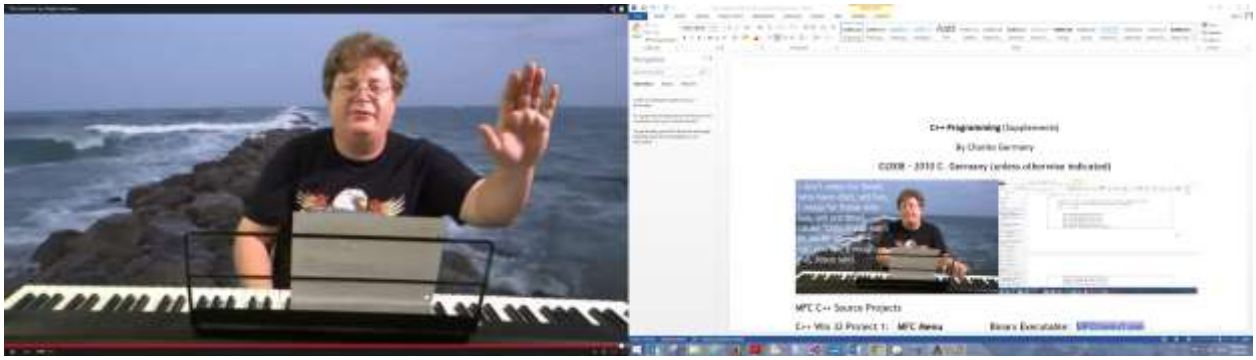
C++ Programming (Supplements)[1]

By Charles Germany



Remember to visit C. Germany's music website: http://www.rainofgod.com/wmaindex.html

MFC C++ Source Projects

C++ Win 32 Project 1: **MFC Menu** Binary Executable: MFCmenu1.exe

Objective: To become familiar with some components of a basic MFC application. This project is part of our second MFC tutorial. It introduces menus, resource files, message maps, message identifiers, message handlers, and many commands, methods and objects that have to do with Windows messaging applications.

Introduction: Messaging is necessary with Windows applications, since Windows is a multi-tasking environment and multiple programs will be running simultaneously. These programs communicate with each other and the operating system through messages.

Getting Started:

1. Launch your Visual C++ compiler.
2. Select File->New and then select "Win 32 Application" from the list of available options.
3. In the Project Name box, type "Menus".
4. Click "OK". On the next screen, make sure "Create an Empty Project" is selected and click "Finish".
5. The compiler will tell you it is creating a new skeleton project. Select "OK" to continue.
6. Add files to the project. Click File->New and select "C/C++ Header File" for the files below with the ".h" extension. Select "C++ Source File" for the files with the ".cpp" extension, and "Text File" for files with the ".rc" extension. Do not forget to reformat text that wraps around multiple lines in the HTML. This will create errors in your C++ programs unless this text is placed on the same line.
7. Then paste the code in the HTML tables below into the appropriate C++ project files.
8. Click Project->Settings and select "Use MFC in a Shared DLL" from the combobox. Click "OK".
9. Rebuild, compile and link.

[1] http://www.networkingprogramming.com/1024x768/index.html

File 1: CGermMFCMenu.h

```
// ©2004 C. Germany
// File 1 of 4 -  MFC Menu - Header File - CGermMFCMenu.h

const int TEXT_SIZE = 16;
class CGermMFCMenu : public CFrameWnd {

      public:
      CGermMFCMenu();
      ~CGermMFCMenu()
      {
         delete m_pWindowText1;
         delete m_pWindowText2;
      }
      //Function and Message Handler Prototype Declarations
      void tally( int & nCount, double dAmount );
      afx_msg void OnExit();
      afx_msg void OnDoCalculate(UINT nCalculate);
      afx_msg void OnShowTotal();
      afx_msg void OnClearTotal();
      afx_msg void LaunchAdventure();
      afx_msg void LaunchHangMan();
      private:
      int m_nJava;
      int m_nVisualBasic;            //count items ordered
      int m_nCPlusPlus;
      int m_nPHP;
      double m_dTotal;                     //tally cost of the order
      char m_szText[ TEXT_SIZE ];                  //output string
      ostrstream MFCOutputString;                  //output string stream
      //Declare 2 pointers to CStatic Window objects
      CStatic * m_pWindowText1;
      CStatic * m_pWindowText2;
      DECLARE_MESSAGE_MAP()
};
```

In the file above, the class "CGermMFCMenu" derives from our mandatory CFrameWnd class. We set up two pointers to CStatic textbox objects so that when we later create new ones on the heap we can point to them. In this example, we are using Windows messages, so we have to declare our message handler prototypes. These "message handlers" are functions that we will define to do various things when their appropriate messages are received. Notice that at the bottom we must also include the "DECLARE_MESSAGE_MAP()" function. We create two pointers to CStatic textboxes, as in our first MFC program. The only other truly new item here is that we have created an instance of the ostrstream class.

The string stream class (ostrstream) works with MFC programs in much the same way as cout << and cin >> work with console programs. We instantiate an object of this class and call it "MFCOutputString". This object will later receive two arguments - the first is the address of a character array, and the second is a variable indicating the size of the character array.

File 2: menu_ids.h

```
//File 2 of 4 - header file - menu_ids.h
//Define messages used by menus.cpp and menus.rc
//IDs for windows messaging are arbitrarily assigned here - that is
to say
//it doesn't matter what you assign for an ID as long as it is
unique.

#define  IDM_EXIT        2000
#define  IDM_Java     2021
#define  IDM_VisualBasic        2022
#define  IDM_CPlusPlus   2041
#define  IDM_PHP    2042
#define  IDM_SHOW_TOTAL  2051
#define  IDM_CLEAR_TOTAL 2052
#define  IDM_ADVENTURE_GAME  2061
#define  IDM_HANG_MAN 2062
```

In the second file above, we are simply defining the message IDs for use with our menu resource file objects. It does not matter what numerical value we assign to these messages - they simply MUST be unique identifiers so that they will not be confused with any other messages. The syntax is to use: #define + the resource file object name + the unique numerical identifier.

File 3: menus.cpp

```
//File 3 of 4 - WinMain() file - menus.cpp

#include <afxwin.h>    // MFC application framework
#include <strstream>   // string stream
#include <iomanip>     // I/O manipulators
#include "menu_ids.h"  // application message ID symbols

using namespace std;   //need for 2003. NET

#include "CGermMFCMenu.h" //CMenu class
//-------------------------------------------------------------------------------------------
--------------------------------

// Scope resolves to CGermMFCMenu Declared in First File
//Define CGermMFCMenu constructor
CGermMFCMenu::CGermMFCMenu()                    // construct window
   : MFCOutputString( m_szText, TEXT_SIZE )     // initialize
ostrstream object
{
```

```
   // Window Caption and Label. Note: "Calculate" is the name of the
menu object from the .rc file
   Create( NULL, "C. Germany 2004 - MFC Menu Example",
WS_OVERLAPPEDWINDOW,
      CRect( 0, 0, 300, 200 ), NULL, "Calculate" );


// Use the pointers to CStatic declared previously to create 2 static
text
// boxes on the heap (free store) and write to their labels and
captions.

m_pWindowText1 = new CStatic;   // create a static text control box

m_pWindowText1->Create("Subcontract programming fees and hourly
rates.\nC. Germany 2004", //text to display
                        WS_CHILD | WS_VISIBLE | WS_BORDER |
SS_CENTER,  //window styles
                        CRect( 70, 60, 210, 130  ),  // window coordinates
                        this ); //window to draw text in

m_pWindowText2 = new CStatic;   // create a static text control box

m_pWindowText2->Create("Programming, rated by hour per programmer.
Pick a language!", //text to display
             WS_CHILD | WS_VISIBLE | SS_CENTER,  //window styles
             CRect(22,10,270,50),  // window coordinates
             this ); //window to draw text in

   //Initialize all variables to null
   m_nJava = m_nVisualBasic = 0;
   m_nCPlusPlus = m_nPHP = 0;
   m_dTotal = 0.0;

} // close CGermMFCMenu constructor defnition

//-------------------------------------------------------------------------------------------
-------------------------------
// count each type of item ordered, compute total bill
void CGermMFCMenu::tally( int & nCount, double dAmount )
{
   nCount++;
   m_dTotal += dAmount;
}

//-------------------------------------------------------------------------------------------
-------------------------------
//afx_msg precedes each message handler function.  We are setting up
//functions that, after they are called, will "listen" for messages.

afx_msg void CGermMFCMenu::OnExit()
{
   SendMessage( WM_CLOSE );
}
```

```
//-------------------------------------------------------------------------------------------
-------------------------------
afx_msg void CGermMFCMenu::OnDoCalculate(UINT nCalculate)
{
   switch (nCalculate)
   {
   case IDM_Java:
        tally( m_nJava, 50.75 );
        break;
   case IDM_VisualBasic:
        tally( m_nVisualBasic, 45.25 );
        break;
   case IDM_CPlusPlus:
        tally( m_nCPlusPlus, 50.95 );
        break;
   case IDM_PHP:
        tally( m_nPHP, 45.10 );
        break;
   }
}

//-------------------------------------------------------------------------------------------
-------------------------------
afx_msg void CGermMFCMenu::OnShowTotal()
{
   MFCOutputString.seekp( 0 );              // reset output string
   MFCOutputString << setprecision( 2 )
       << setiosflags( ios::fixed | ios::showpoint )
       << "         $" << m_dTotal << ends;  // stopper
   // display new dialog box with output string
   MessageBox( m_szText, "Your total project cost per hour by number
and type of programmers is:" );
   m_dTotal = 0.0;
}

//-------------------------------------------------------------------------------------------
-------------------------------
afx_msg void CGermMFCMenu::OnClearTotal()
{
        m_dTotal = 0.0;
        MessageBox( "          $0.00", "Cleared Order" );
}

//-------------------------------------------------------------------------------------------
-------------------------------
afx_msg void CGermMFCMenu::LaunchAdventure()
{
        system("AdventureGame9.exe");
}

//-------------------------------------------------------------------------------------------
-------------------------------
afx_msg void CGermMFCMenu::LaunchHangMan()
{
        system("HangMan.exe");
}
```

```
//-------------------------------------------------------------------------------------------
------------------------------

// Begin MessageMap is sort of like adding an itemListener() in
Java...
// Open Message Map Tags, tells compiler which function to call when
a message is received.

BEGIN_MESSAGE_MAP( CGermMFCMenu, CFrameWnd )

   ON_COMMAND( IDM_EXIT, OnExit )
   ON_COMMAND_RANGE(IDM_Java, IDM_PHP, OnDoCalculate)
   ON_COMMAND( IDM_SHOW_TOTAL, OnShowTotal )
   ON_COMMAND( IDM_CLEAR_TOTAL, OnClearTotal )
   ON_COMMAND( IDM_ADVENTURE_GAME, LaunchAdventure )
   ON_COMMAND( IDM_HANG_MAN, LaunchHangMan )
END_MESSAGE_MAP()

//-------------------------------------------------------------------------------------------
------------------------------

// Below, CMenusApp, deriving from the mandatory CWinApp, acts like
main() in a
// console program. It creates a new CGermMFCMenu on the heap, which
itself derives from
// the mandatory CFrameWnd.  Remember that CGermMFCMenu was declared
in the 1st file.
class MakeCGermMFCMenu : public CWinApp {

      public:

      BOOL InitInstance()        // called by CWinApp::CWinApp
      {
           m_pMainWnd = new CGermMFCMenu;            // create window
           m_pMainWnd->ShowWindow( m_nCmdShow );  // make it visible
           m_pMainWnd->UpdateWindow();            // force refresh
           return true;                           // report success
      }
} // Notice there's no semicolon here at end of class specification


MakeCGermMFCMenu;                  // calls CWinApp::CWinApp
constructor

//-------------------------------------------------------------------------------------------
------------------------------
```

In the third file above, the first new thing we do is to include "strstrea.h" and "iomanip.h". We then define the CGermMFCMenu constructor, which in turn creates an instance of ostrstream using our MFCOutputString declared in the class specification of the first file. We pass to it the two arguments mentioned earlier, an address of a char array, and the size of that char array.

The next thing we do is call Create() to create the main window, passing in its attributes as arguments. Notice the last highlighted argument in the parameters that get passed to Create(), the one in quotes called "Calculate". This is where we pass in the name of the Menu object created in the "menus.rc" file. The name of this object in the .rc file is "Calculate". Passing this object's name to the Create() function adds that menu object to the window. The "WS_OVERLAPPEDWINDOW" argument specifies that is is a typical window with a title bar, minimize, maximize and close buttons.

The CRect() function nested inside the Create() method specifies window location and size. The arguments passed to this function specify that the coordinates of the top left corner will be 0, 0 (top left corner of the screen) and that the window will be 200 wide by 200 high.

Next, our pointer, m_pWindowText1, creates a new CStatic textbox on the heap and then calls that object's Create() method. The text to be displayed, the textbox's position and size, and its attributes are all passed in as parameters. The data member pointer m_pWindowText2 does the same thing as the first data member pointer.

Next, some variables are initialized and a function is defined. We must use the ostrstream object to display information when using this function. Now it begins to get interesting.

Next, we declare our message handler functions. Notice that they are always prefixed with afx_msg. After this prefix, specifying that the function is a message handler, the function is declared with a return type and arguments, just as any other function would be. The command "SendMessage()", explicitly transmits a message, passing it the pre-defined "WM_CLOSE" message ID as an argument. The message handler, "OnClose()", inherited from CWnd, will receive the message and close the window and terminate the program.

The message handler function "OnDoCalculate" simply tallies up the resulting total of each item selected based upon the unique identifier of each message received. Then, in the next two functions, message boxes are created to display the results, using MessageBox(), and passing in char arrays and literal strings as arguments.

In the "OnShowtotal" function, a few new interesting items are introduced. First, "seekp()" is used to to position the "put pointer" to the beginning of the string. The fucntion seekp() is passed a 0 to move the pointer to the beginning of the string. The "put pointer" is used to keep track of the subscript value of the characters in the array in order to determine where the next write will occur. The "ends" passed to the MFCOutputString ostrstream object functions as "endl" would function with cout << in console programming. the "setprecision(2)" is used to format the output of the float to two decimal places.

Next, we use our message map tags to define our message map. The message map is where we associate our message handlers with our message identifiers. This tells our application which message handler function to call based on which message is received. These are all placed between our "BEGIN_MESSAGE_MAP" and "END_MESSAGE_MAP" tags. We use the macro "ON_COMMAND" to map a single message identifier to a single message handler. We use the macro "ON_COMMAND_RANGE" to map multiple message identifiers to message handlers. By convention, though not mandatory, the prefix "IDM" is used with message identifiers and the prefix "On" is used with message handlers.

Finally, the class "MakeCGermMFCMenu" is used to create an instance of "CGermMFCMenu" on the heap and thus construct the window and run the program. As in the previous example, we must again provide an InitInstance() method and supply it with the three items:

- Create an instance on the heap of our CGermMFCMenu class that derives from CFrameWnd
- Call the ShowWindow() method to make the window visible
- Call UpdateWindow() to refresh/repaint the screen so it will display the window

Notice that in the MFC, "BOOL" is used as a return type. Remember that C++ is case sensitive, and the MFC's "BOOL" is different from the standard "bool" we find in C++. This holds true as well with the MFC's "TRUE" and "FALSE" - they are different from C++'s standard "true" and "false".

Finally, we create an instance of MakeCGermMFCMenu, and this creates the window and launches the application.

File 4: menus.rc

```
// File 4 of 4 - resource script for menu - menus.rc
// Note: Create this files as a new plain text file and save it as
"menus.rc".

#include <afxres.h>
#include "menu_ids.h"

// Sets up menus, sub-menus and which message to broadcast when they
are clicked
Calculate MENU
{
   POPUP "File"
   {
      MENUITEM "Exit", IDM_EXIT
   }
   POPUP "Games"
   {
      MENUITEM "Adventure Game", IDM_ADVENTURE_GAME
      MENUITEM "Hang Man", IDM_HANG_MAN
   }
   POPUP "Level 1"
   {
      MENUITEM "PHP Programming", IDM_PHP
      MENUITEM "VisualBasic Programming", IDM_VisualBasic
   }
   POPUP "Level 2"
   {
      MENUITEM "C++ Programming", IDM_CPlusPlus
      MENUITEM "Java Programming", IDM_Java
   }
   POPUP "Order"
   {
```

```
      MENUITEM "Total Hourly Charges", IDM_SHOW_TOTAL
      MENUITEM "Clear Total", IDM_CLEAR_TOTAL
   }
}
```

This last file is a resource file. Resource files have their own language and terminology. They have the ".rc" extension. you can create them manually in a text file if you know the terminology, or you may create them visually using Microsoft's resource editor. To add message identifiers and handlers to resource file objects, you can create them using the editor and then later import the .rc file into a text editor.

Whew! This second MFC lesson was a woozie! We have covered a LOT of new material concerning the MFC.
The subsequent lessons will focus on MFC objects and their attributes, one at a time.

Resource File Options for Multi Line Edit Controls:

ES_MULTILINE - specifies multiline textbox.
ES_WANTRETURN - allows textbox to respond to ENTER key.
WS_VSCROLL - causes vertical scroll bar to be displayed.

```
//MFC Multi Line Text Edit Box and Word Count.


------------------------------------------------------------------------
------------------------
//Save as "MultiEditText_ids.h" - define edit text message
identifiers. File 1 of 4.

#define IDC_TEXT     2001
#define IDC_COUNT    2002

------------------------------------------------------------------------
------------------------
//Save as "MultiEditText.h" - multiline edit text - file 2 of 4.
const int MAX_TEXT = 128;
class CEditTextDialog : public CDialog {
public:
   CEditTextDialog( char *lpszName );
   afx_msg void OnCount();     // clicked the "Count" button
private:
   char m_szText[ MAX_TEXT + 1 ];
   DECLARE_MESSAGE_MAP()
};
------------------------------------------------------------------------
------------------------
//Save as "EditText.cpp" - File 3 of 4.

#include <afxwin.h>
#include <strstrea.h>
#include "MultiEditText_ids.h"
```

```
#include "MultiEditText.h"
// Dialog constructor
CEditTextDialog::CEditTextDialog( char *lpszName )
   : CDialog( lpszName )   // base class constructor
{
   m_szText[ 0 ] = '\0';
}
// count the characters in the edit text control
afx_msg void CEditTextDialog::OnCount()
{
   // get address of edit control
   CEdit *pText = ( CEdit * ) GetDlgItem( IDC_TEXT );
   pText->GetWindowText( m_szText, MAX_TEXT );
   // display length of text read from edit text control
   static char szBuf[ 20 ];
   static ostrstream str( szBuf, 20);
   str.seekp( 0 );
   str << "Text length = " << strlen( m_szText ) << ends;
   pText->SetWindowText( szBuf );
}
BEGIN_MESSAGE_MAP( CEditTextDialog, CDialog )
   ON_COMMAND( IDC_COUNT, OnCount )
END_MESSAGE_MAP()
// start dialog-based application
class CEditApp : public CWinApp {
public:
   BOOL InitInstance()
   {
      CEditTextDialog editTextDialog( "EditText" );
      editTextDialog.DoModal();     // run dialog
      return FALSE;                 // finished
   }
} editApp;

-------------------------------------------------------------------------
------------------------
//Open text file then save as "edittext.rc" - resource file. File 4
of 4.
#include <afxres.h>
#include "edittext_ids.h"
EditText DIALOG  50, 50, 130, 130
CAPTION "Edit"
{
   LTEXT          "Enter text:",
                  IDC_STATIC, 30,  20, 50, 8
   // Define edit text with identifier IDC_TEXT,
   // position (30, 30) and size 70 by 64
   // edit styles multiline and auto vertical scroll
   EDITTEXT       IDC_TEXT,   30,  30, 70, 64,
                  ES_MULTILINE | ES_WANTRETURN | WS_VSCROLL
   DEFPUSHBUTTON  "Count",
                  IDC_COUNT,  50, 100, 30, 15
}
```

Binary Executable: **[** adventuregame7.exe [Broken link] **]** (Console)
Binary Executable: **[** hangman.exe [Broken link] **]** (Console)
Binary Executable: **[** MFCmenu1.exe **]** (MFC Version with Menu Example)

Simplified, Previous Stages for Incremental Lessons:

```
Class Projects/Quizes - Without the Solutions Displayed

//Adventure Game Header File - Functions and Classes
#include "stdafx.h"
#include <iostream.h>
#include <stdlib.h>
#include <time.h>
//Game Functions
int GenerateRandomNumber(int Number=6) {
int ResultRandom;
srand(time(NULL));
ResultRandom = (rand()%Number) + 1;
return ResultRandom;
}
void Attack(int *PlayerPoints) {
int Damage;
Damage = GenerateRandomNumber(10);
*PlayerPoints = *PlayerPoints - Damage;
cout << "Ouch!  You loose " << Damage << " points.\n";
cout << "You now have " << *PlayerPoints << " points left!";
}
void SetTheScene(char PName[25]) {
// Takes char array (or "string") as an argument.
// Example of using a global value locally inside a function
int Setting;
Setting = GenerateRandomNumber();
switch (Setting) {
case 1 : cout << PName << ", you are in a swamp and sinking!\n";
             break;
case 2 : cout << PName << ", you are on a high mountain peak.\n";
             break;
case 3 : cout << PName << ", you are on a grassy plane.\n";
             break;
case 4 : cout << PName << ", you are in the desert.\n";
             break;
case 5 : cout << PName << ", you find yourself in a deserted
village.\n";
             break;
case 6 : cout << PName << ", you find yourself in a steamy
jungle.\n";
             break;
```

```
default : cout << "Something is definitely wrong here.  Should be
1-6.";
} // close switch statement
} // close SetTheScene() function
//CLASSES
class Monster {
public:
//BAse CLASS - 3 Overloaded Constructors
        Monster(int MonStrength = 10) {
                Strength = MonStrength;
                cout << "\nA monster has been created.";}
        Monster(int MonStrength, int MonWeight) {
                Strength = MonStrength;
                Weight = MonWeight;
                cout << "\nA monster has been created.";}
        Monster(int MonStrength, int MonWeight, char MonName[10])
{
                Strength = MonStrength;
                Weight = MonWeight;
                Name[10] = MonName[10];
                cout << "\nA monster has been created.";}
        ~Monster() { cout << "\nThe monster has been
destroyed!\n";}
//Accessor Methods
        int getStrength() { return Strength; }
        void setStrength(int MonsStrength) { Strength =
MonsStrength; }
        int getWeight() {return Weight;}
        void setWeight(int Wt) {Weight = Wt;}
    int getName() {return Name[10];}
        void setName(char Nm[10]) {Name[10] = Nm[10];}

        void Talk() {
        int SayWhat;
        SayWhat = GenerateRandomNumber(4);
        switch(SayWhat) {
    case 1 : cout << "I like flowers.  Will you be my friend?";
                    break;
        case 2 : cout << "I would really like to dismember you and
eat you...";
                    break;
        case 3 : cout << "Unfortunately, I have had a very bad day
and"
                                << " you happen to be the first
creature I\'ve met"
                                << " that I can take it out
on...";
                    break;
        case 4 : cout << "How do you taste?  I don\'t think I\'ve
eaten"
                                << " your kind before...";
                    break;
        default : cout << "Uh oh, this should never happen...";
        } //closes switch
        } //close talk function
```

```
protected:
        int Strength;
        int Weight;
        char Name[10];
}; //closes class specification

class Giant : public Monster {
//Derived class of Monster
public:
        Giant() {cout << "A giant is afoot...";}
        ~Giant() {cout << "Bye bye giant...";}
//Accesor Methods
   int getFootsize() {return Footsize;}
   void setFootSize(int GiantFeet) {Footsize = GiantFeet;}
//Other Functions
   void GiantSpeak() {cout << "Fee Fi Fo Fum...";}
   void Squash(int * Life) {
           cout << "The giant squashes you like a bug!";
       *Life = 0;}

private:
        int Footsize;
}; //close class specification
class Troll : public Monster {
//Derived class of Monster
public:
        Troll() {cout << "Somewhere a troll is in a bad mood...";}
        ~Troll() {cout << "Bye troll...";}
//Accesor Methods
   int getDrool() {return Drool;}
   void setDrool(int TrollDrool) {Drool = TrollDrool;}
//Other Functions
   void GiantSpeak() {cout << "Ahhhhhhhhhhhhhh...";}
   void Eat(int * Life) {
           cout << "The dismembers you and boils your carcass in a
stew.";
       *Life = 0;}
   void Drooling() {
   setDrool(10);
   cout << "\nThe troll salivates, dropping " << getDrool()
            << " gallons of drool onto your shiny new boots.\n";}
private:
        int Drool;
}; //close class specification
class BarryManilow : public Monster {
//Derived class of Monster
public:
        BarryManilow() {cout << "Run for your life - it\'s
Barry!";}
        ~BarryManilow() {cout << "I write the songs that...";}
//Accesor Methods
   int getSong() {return Song[20];}
   void setSong(int BarrySong[20]) {Song[20] = BarrySong[20];}
//Other Functions
   void BarrySing(int * Life) {
```

```
           cout << "Barry lifts his voice to sing a ballad...";
           cout << "\nYou feel the sudden urge to pull out your
own\n"
                    << "sword and kill yourself, right then and
there.\n"
                        << "Barry, for lack of an interested
audience, \n"
                        << "continues serenading your rotting
corpse...\n";
       *Life = 0;}

private:
        int Song[20];
}; //close class specification
// main() Function .cpp File (Goes with previous Function/Class
header file)
// Using srand(), rand(), class structure, while true, functions,
// switch statement, and nested if/else strictures, implementing
variables and objects of
// global and local scope, Overriding default constructor and
destructor, passing pointers.
#include "Stdafx.h"
#include "GameClasses.h"

void main() {
int ALIVE = 100;
int PlayerPoints = 20;  //Global
char conflict;
char conflict2;
char Name[25];
cout << "What is your name, player? ";
cin >> Name;
cout << "Welcome to the game, " << Name << ".\n\n"; //global Name
cout << "You have " << PlayerPoints << " points allocated to you
as you start the game.\n";
SetTheScene(Name);
Monster Ogre;
Monster Dragon(20);
while(ALIVE>0) {
Dragon.Talk();
cout << "\nYou encounter a dragon!\n";
cout << "He blasts you with fire! ";
Attack(&PlayerPoints);
cout << " He is of strength ";
cout << Dragon.getStrength();
cout << ".\n\n";
cout << "\nYou pull out your sword and levy a heavy blow upon his
head.\n";
Dragon.setStrength(5);
cout << "That was some move! The dragon is already half dead!\n";
cout << "He now only has the strength of ";
cout << Dragon.getStrength();
cout << ".\n\n";
cout << "Now what do you choose to do?\n"
     << "r = run away\n"
```

```
         << "s = stay and fight\n"
         << "t = talk to the dragon\n";
cin >> conflict;
system("cls");
switch(conflict) {
case 'r' : cout << "You run away and survive!\n"
                  << "The dragon inflicts merely a flesh
wound.\n"
               << "You loose a mere 15 points. ";
              ALIVE = ALIVE - 15;
                 cout << "You now have " << ALIVE << " points
left.";
           break;
case 's' : cout << "You stay and fight.  You discover that\n"
               << "the dragon is merely a baby dragon.  It\'s
mother\n"
               << "then shows up, greatly enraged by your attempt
to\n"
               << "kill her child.  She is really big and really
angry.\n";
           Dragon.setStrength(200);
               cout << "You are now fighting a dragon of strength:
";
                  cout << Dragon.getStrength() << ".\n";
                  cout << "The dragon rips off your arms and legs
and\n"
                           << "fries you to a crispy, crunchy,
golden brown.\n";
              ALIVE = 0;
                  break;
case 't' : cout << "\nYou strike up a conversation with the
dragon.\n"
                << "You begin telling it your life story and all
your woes.\n"
                        << "Succumbing to fatal levels of nausea
and boredom, the \n"
                        << "dragon lapses into a coma and
dies...\n";
              Dragon.setStrength(0);
              cout << "Dragon reduced to strength: ";
                  cout << Dragon.getStrength() << ".\n";
                  Dragon.~Dragon();
                  cout << "You have " << ALIVE << " points left.";
                  break;
default :  cout << "Invalid input.  Please press either: r, s, or
w.";
} //close switch statement
while(ALIVE>0) {
cout << "\nYou continue traveling eastward.  You see something in
the distance...\n";
Ogre.Talk();
cout << "It is an ogre!\n";
cout << "What do you do now?\n";
cout << "r = run away\n"
     << "t = talk\n";
```

```
cin >> conflict2;
system("cls");
  if(conflict2 == 'r') {
     cout << "\nThe ogre chases you.  You fall.  He chops off your
head and eats you.\n";
     ALIVE = 0;
  }
  else if(conflict2 == 't') {
         cout << "\nYou make friends.  The ogre likes you and so
kills you.\n";
         ALIVE = 0;
  }
} //close 2nd while true loop
} //close 1st while true loop - ALIVE no longer > 0
cout << "\nYou fought bravely but died.  As the worms devour your
flesh\n"
     << "and your soul descends into Hades you vow that you will
one day\n"
     << "arise to take vengeance on your foes.  Dare you try your
luck\n"
     << "and suffer defeat once more?  At least for now, the game
is over....\n\n";
} //close main function



// AdventureGame3.cpp : Using srand(), rand(), class structure,
while true, functions,
// switch statement, and nested if/else strictures, implementing
variables and objects of
// global and local scope, Overriding default constructor and
destructor, passing pointers.
#include "stdafx.h"
#include <iostream.h>
#include <stdlib.h>
#include <time.h>
class Monster {
public:
        Monster(int MonStrength = 10) {
                Strength = MonStrength;
                cout << "\nA monster has been created.";}
        ~Monster() { cout << "\nThe monster has been
destroyed!\n";}
        int getStrength() { return Strength; }
        void setStrength(int MonsStrength) { Strength =
MonsStrength; }
        void Talk() { cout << "\nGrrr!! I am a monster... "; }
private:
        int Strength;
}; //closes class specification
int GenerateRandomNumber(int Number=6) {
int ResultRandom;
srand(time(NULL));
ResultRandom = (rand()%Number) + 1;
/*
```

```
//For debugging purposes only
cout << "\nToday\'s lucky number is: "
     << ResultRandom << ".\n\n";
*/
return ResultRandom;
}
void Attack(int *PlayerPoints) {
int Damage;
Damage = GenerateRandomNumber(10);
*PlayerPoints = *PlayerPoints - Damage;
cout << "Ouch!  You loose " << Damage << " points.\n";
cout << "You now have " << *PlayerPoints << " points left!";
} //close attack() function
void SetTheScene(char PName[25]) {
// Takes char array (or "string") as an argument.
// Example of using a global value locally inside a function
int Setting;
Setting = GenerateRandomNumber();
switch (Setting) {
case 1 : cout << PName << ", you are in a swamp and sinking!\n";
             break;
case 2 : cout << PName << ", you are on a high mountain peak.\n";
             break;
case 3 : cout << PName << ", you are on a grassy plane.\n";
             break;
case 4 : cout << PName << ", you are in the desert.\n";
             break;
case 5 : cout << PName << ", you find yourself in a deserted
village.\n";
             break;
case 6 : cout << PName << ", you find yourself in a steamy
jungle.\n";
             break;
default : cout << "Something is definitely wrong here.  Should be
1-6.";
} // close switch statement
} // close SetTheScene() function

void main() {
int ALIVE = 100;
int PlayerPoints = 20;  //Global
char conflict;
char conflict2;
char Name[25];
cout << "What is your name, player? ";
cin >> Name;
cout << "Welcome to the game, " << Name << ".\n\n"; //global Name
cout << "You have " << PlayerPoints << " points allocated to you
as you start the game.\n";
SetTheScene(Name);
Monster Ogre;
Monster Dragon(20);
while(ALIVE>0) {
Dragon.Talk();
cout << "\nYou encounter a dragon!\n";
```

```
cout << "He blasts you with fire! ";
Attack(&PlayerPoints);
cout << " He is of strength ";
cout << Dragon.getStrength();
cout << ".\n\n";
cout << "\nYou pull out your sword and levy a heavy blow upon his
head.\n";
Dragon.setStrength(5);
cout << "That was some move! The dragon is already half dead!\n";
cout << "He now only has the strength of ";
cout << Dragon.getStrength();
cout << ".\n\n";
cout << "Now what do you choose to do?\n"
     << "r = run away\n"
         << "s = stay and fight\n"
         << "t = talk to the dragon\n";
cin >> conflict;
system("cls");
switch(conflict) {
case 'r' : cout << "You run away and survive!\n"
                  << "The dragon inflicts merely a flesh
wound.\n"
                << "You loose a mere 15 points. ";
               ALIVE = ALIVE - 15;
                   cout << "You now have " << ALIVE << " points
left.";
           break;
case 's' : cout << "You stay and fight.  You discover that\n"
                << "the dragon is merely a baby dragon.  It\'s
mother\n"
                << "then shows up, greatly enraged by your attempt
to\n"
                << "kill her child.  She is really big and really
angry.\n";
           Dragon.setStrength(200);
               cout << "You are now fighting a dragon of strength:
";
                   cout << Dragon.getStrength() << ".\n";
                   cout << "The dragon rips off your arms and legs
and\n"
                            << "fries you to a crispy, crunchy,
golden brown.\n";
               ALIVE = 0;
                   break;
case 't' : cout << "\nYou strike up a conversation with the
dragon.\n"
                << "You begin telling it your life story and all
your woes.\n"
                        << "Succumbing to fatal levels of nausea
and boredom, the \n"
                        << "dragon lapses into a coma and
dies...\n";
               Dragon.setStrength(0);
               cout << "Dragon reduced to strength: ";
                   cout << Dragon.getStrength() << ".\n";
```

```
                    Dragon.~Dragon();
                    cout << "You have " << ALIVE << " points left.";
                    break;
default :  cout << "Invalid input.  Please press either: r, s, or
w.";
} //close switch statement
while(ALIVE>0) {
cout << "\nYou continue traveling eastward.  You see something in
the distance...\n";
Ogre.Talk();
cout << "It is an ogre!\n";
cout << "What do you do now?\n";
cout << "r = run away\n"
     << "t = talk\n";
cin >> conflict2;
system("cls");
  if(conflict2 == 'r') {
     cout << "\nThe ogre chases you.  You fall.  He chops off your
head and eats you.\n";
     ALIVE = 0;
  }
  else if(conflict2 == 't') {
   cout << "\nYou make friends.  The ogre likes you and so kills
you.\n";
   ALIVE = 0;
  }
} //close 2nd while true loop
} //close 1st while true loop - ALIVE no longer > 0
cout << "\nYou fought bravely but died.  As the worms devour your
flesh\n"
     << "and your soul descends into Hades you vow that you will
one day\n"
     << "arise to take vengeance on your foes.  Dare you try your
luck\n"
     << "and suffer defeat once more?  At least for now, the game
is over....\n\n";
} //close main function



// Project 3 - AdventureGame1.cpp : Adventure game using
functions,
// a while true loop and a switch statement.
#include "stdafx.h"
#include <iostream.h>
void Talk() {
char name;
cout << "Very hospitable of you.  You greet the old man. "
     << "He turns towards you and asks your name.  Your reply?";
cin >> name;
}
int Attack() {
cout << "You attack.  The old man turns into an ogre.\n"
     << "He pulls out an axe and chops off your head.\n"
         << "You are now dead, dead, dead...\n";
```

```
     return 0;
}
void Give() {
cout << "You give him a sandwhich";
}
void Ignore() {
cout << "You decide to ignore the man";
}
void main()
{
int ALIVE = 1;
char choice1;
while (ALIVE != 0) {
cout << "\nIt\'s twilight, and you can just make out the "
     << "constellatoin Orion as you look North.  You are "
         << "walking on towards a mountain range, ascending "
         << "towards the west.  You see an elderly man in "
         << "tattered clothes approaching.  What do you do?\n";
cout << "Your options are as follows:\n\n"
     << "T - Talk with the old man.\n"
         << "H - Hit the old man in the head with a shovel.\n"
         << "G - Give the old man a sandwhich.\n"
         << "I - Ignore the old man.\n";
cout << "\nYou choose: ";
cin >> choice1;
switch (choice1) {
case 'T' : Talk();
               break;
case 't' : Talk();
               break;
case 'H' : ALIVE = Attack();
               break;
case 'h' : ALIVE = Attack();
               break;
case 'G' : Give();
               break;
case 'g' : Give();
               break;
case 'I' : Ignore();
               break;
case 'i' : Ignore();
               break;
default : cout << "\nSorry, that is not a valid choice.\n\n";
} //closes switch statement
cout << "\nIf only, if only, if only...\n\n";
} //close while true loop
cout << "\n\nGame ending...\n\n";
} //closes main() function
```

// AdventureGame2.cpp Now using class structure, functions, random number generation, and
// implementing variables and objects of global and local scope.
// Overriding default constructor and destructor

```
#include "stdafx.h"
#include <iostream.h>
#include <stdlib.h>
#include <time.h>
class Monster {
public:
        Monster(int MonStrength = 10) {
                Strength = MonStrength;
                cout << "\nA monster has been created.";}
        ~Monster() { cout << "\nThe monster has been
destroyed!\n";}
        int getStrength() { return Strength; }
        void setStrength(int MonsStrength) { Strength =
MonsStrength; }
        void Talk() { cout << "\nGrrr!! I am a monster... "; }
private:
        int Strength;
}; //closes class specification
int GenerateRandomNumber() {
int ResultRandom;
srand(time(NULL));
ResultRandom = (rand()%6) + 1;
cout << "\nToday\'s lucky number is: "
     << ResultRandom << ".\n\n";
return ResultRandom;
}
void SetTheScene() {
int Setting;
Setting = GenerateRandomNumber();
switch (Setting) {
case 1 : cout << "You are in a swamp.\n";
             break;
case 2 : cout << "You are on a high mountain peak.\n";
             break;
case 3 : cout << "You are on a grassy plane.\n";
             break;
case 4 : cout << "You are in the desert.\n";
             break;
case 5 : cout << "You find yourself in a deserted village.\n";
             break;
case 6 : cout << "You find yourself in a steamy jungle.\n";
             break;
default : cout << "Something is definitely wrong here.  Should be
1-6.";
} // close switch statement
} // close SetTheScene() function
void main() {
int ALIVE = 100;
char conflict;
char conflict2;
SetTheScene();
Monster Ogre;
Monster Dragon(20);
while(ALIVE>0) {
Dragon.Talk();
```

```
cout << "\nYou encounter a dragon!\n"
     << "He is of strength ";
cout << Dragon.getStrength();
cout << ".\n\n";
cout << "\nYou pull out your sword and levy a heavy blow upon his
head.\n";
Dragon.setStrength(5);
cout << "That was some move! The dragon is already half dead!\n";
cout << "He now only has the strength of ";
cout << Dragon.getStrength();
cout << ".\n\n";
cout << "Now what do you choose to do?\n"
     << "r = run away\n"
         << "s = stay and fight\n"
         << "t = talk to the dragon\n";
cin >> conflict;
system("cls");
switch(conflict) {
case 'r' : cout << "You run away and survive!\n"
                  << "The dragon inflicts merely a flesh
wound.\n"
               << "You loose a mere 15 points. ";
              ALIVE = ALIVE - 15;
                 cout << "You now have " << ALIVE << " points
left.";
          break;
case 's' : cout << "You stay and fight.  You discover that\n"
              << "the dragon is merely a baby dragon.  It\'s
mother\n"
              << "then shows up, greatly enraged by your attempt
to\n"
              << "kill her child.  She is really big and really
angry.\n";
           Dragon.setStrength(200);
              cout << "You are now fighting a dragon of strength:
";
                 cout << Dragon.getStrength() << ".\n";
                 cout << "The dragon rips off your arms and legs
and\n"
                          << "fries you to a crispy, crunchy,
golden brown.\n";
              ALIVE = 0;
                 break;
case 't' : cout << "\nYou strike up a conversation with the
dragon.\n"
               << "You begin telling it your life story and all
your woes.\n"
                       << "Succumbing to fatal levels of nausea
and boredom, the \n"
                       << "dragon lapses into a coma and
dies...\n";
              Dragon.setStrength(0);
              cout << "Dragon reduced to strength: ";
                 cout << Dragon.getStrength() << ".\n";
                 Dragon.~Dragon();
```

```
                    cout << "You have " << ALIVE << " points left.";
                    break;
default :  cout << "Invalid input.  Please press either: r, s, or
w.";
} //close switch statement
while(ALIVE>0) {
cout << "\nYou continue traveling eastward.  You see something in
the distance...\n";
Ogre.Talk();
cout << "It is an ogre!\n";
cout << "What do you do now?\n";
cout << "r = run away\n"
     << "t = talk\n";
cin >> conflict2;
system("cls");
  if(conflict2 == 'r') {
     cout << "\nThe ogre chases you.  You fall.  He chops off your
head and eats you.\n";
     ALIVE = 0;
  }
  else if(conflict2 == 't') {
   cout << "\nYou make friends.  The ogre likes you and so kills
you.\n";
   ALIVE = 0;
  }
} //close 2nd while true loop
} //close 1st while true loop - ALIVE no longer > 0
cout << "\nYou fought bravely but died.  As the worms devour your
flesh\n"
     << "and your soul descends into Hades you vow that you will
one day\n"
     << "arise to take vengeance on your foes.  Dare you try your
luck\n"
     << "and suffer defeat once more?  At least for now, the game
is over....\n\n";
} //close main function
```

Visual Studio MFC 2008 Bare Bones CDialog GUI Template

Objective: To create a 2008 bare bones GUI template for CDialog objects without the bloat of VS wizards.

File 1: GUI.cpp

```
//01/16/2010 C. Germany Visual Studio 2008 MFC Template
/*
     Notes: This is a MFC template to create graphical C++ apps
without
```

```
                using the Visual Studio 2008 wizards. It is
stripped down to
                        a bare minimum of code so apps are as
simplified as possible.
                        You must disable some automated features
and manually set a
                        few properties for projects using this
template.
                        1. Create an EMPTY Win32 project. Click ->
"File" -> -> "New"
                           -> "Project" -> "Win32" -> "Win32
Project".
                    2. Name is and select a directory.
                        3. Select "Application Settings" -> "Empty
Project" then click "Finish".
                        4. Rt-click project, select "Properties" ->
"Configuration Properties"
                           -> "General" -> "Use of MFC" and change
it to
                           "Use MFC in a Static Library". This will
give you MFC components.
                    5. Disable Incremental Linking. Rt-click
project, select "Properties"
                           -> "Configuration Properties" ->
"Linker" -> "General" ->
                           "Enable Incremental Linking" and set it
to "NO".
                     6. Add a resources file. Rt-click resources
and add a DIALOG object.
                    7. Change the enumerated constant to match the
Dialog name.
                    8. Note that unlike 2003, when calling
SetWindowText() in 2008 the
                           string passed in must be cast/converted
using "L".
*/
//-------------------------------------------------------------------
-------------------------
#include <afxwin.h>      //MFC core and standard components
#include "resource.h"    //main symbols
//-------------------------------------------------------------------
-------------------------
//Globals
CEdit * TEST;
class GAME_FORM : public CDialog
{
    public:
    GAME_FORM(CWnd* pParent = NULL): CDialog(GAME_FORM::IDD,
pParent)
    {    }
    // Dialog Data, name of dialog form
    enum{IDD = IDD_GAME};
    protected:
```

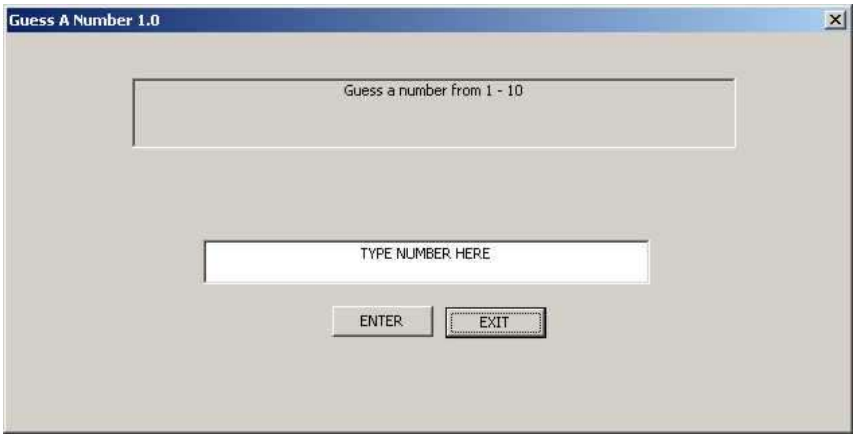
```
    virtual void DoDataExchange(CDataExchange* pDX) {
CDialog::DoDataExchange(pDX); }
    //Called right after constructor. Initialize things here.
    virtual BOOL OnInitDialog()
    {
            CDialog::OnInitDialog();
            TEST = (CEdit *) GetDlgItem(IDC_TEST);
            TEST->SetWindowText(L"Hello!");
            return true;
    }
public:
DECLARE_MESSAGE_MAP()
};
//-----------------------------------------------------------------
-------------------------
class TheGame : public CWinApp
{
public:
TheGame() {  }
public:
virtual BOOL InitInstance()
    {
   CWinApp::InitInstance();
   SetRegistryKey(_T("Hills Of Darkness"));
   GAME_FORM dlg;
   m_pMainWnd = &dlg;
   INT_PTR nResponse = dlg.DoModal();
   return FALSE;
} //close function
};
//-----------------------------------------------------------------
-------------------------
//Need a Message Map Macro for both CDialog and CWinApp
BEGIN_MESSAGE_MAP(GAME_FORM, CDialog)
END_MESSAGE_MAP()
//-----------------------------------------------------------------
-------------------------
TheGame theApp;  //Starts the Application
```


MFC Number Guessing Game 1

C++ 2008 Visual Studio Win 32 Project: MFC Guess a Number Game 1

Binary Executable: 2008_NumberGuess1.exe

Objective: To utilize MFC components and create a simple number guessing game.



File 1: GUI.cpp

```
//TEMPLATE
//Rt-click PROJECT and select -> "Properties" -> "Configuration
Properties"
//-> "C/C++" -> "General" -> "Debug Information Formant" and
//-> "Program Database for Edit and Continue /ZI". Set it to
"Disabled".
//-----------------------------------------------------------------
----------------------
#include <afxwin.h>      // MFC core and standard components
#include "resource.h"    // main symbols
#include <time.h>
//-----------------------------------------------------------------
----------------------
//Globals
CEdit * Input;
CStatic * Output1;
CButton * Enter;
int NUM;
int TheGuess;
class GAME_FORM : public CDialog
{
    public:
    GAME_FORM(CWnd* pParent = NULL): CDialog(GAME_FORM::IDD, pParent)
    {    }
    // Dialog Data, name of dialog form
    enum{IDD = IDD_Interface};
    protected:
    virtual void DoDataExchange(CDataExchange* pDX) {
CDialog::DoDataExchange(pDX); }
    //Called right after constructor. Initialize things here.
    virtual BOOL OnInitDialog()
    {
            CDialog::OnInitDialog();
            Input = (CEdit *) GetDlgItem(CE_INPUT);
            Output1 = (CStatic *) GetDlgItem(CS_Output1);
                      Enter = (CButton *) GetDlgItem(B_ENTER);
```

```
            Output1->SetWindowText(L"Guess a number from 1 - 10");
            Input->SetWindowText(L"TYPE NUMBER HERE");

                        //Enter->SetWindowText("GO!");
            srand(time(NULL));
            return TRUE;
    }
public:
DECLARE_MESSAGE_MAP()
//-----------------------------------------------------
//Message Handler for ENTER Button
afx_msg void ENTER()
{
        NUM = (rand() % 3) + 1;
                CString MESS;
                Input->GetWindowText(MESS);
                Output1->SetWindowText(MESS);

                //Need to convert from CString to const char array for
atoi()
                char CONVERT[20];
                for(int x = 0; x < MESS.GetLength(); x++)
                { CONVERT[x] = MESS[x]; }

                TheGuess = atoi(CONVERT);
                //Another way to use CString with atoi()
                //CString * test = new CString("12");
                //int z = atoi((const char *)test);

        if(TheGuess == NUM)
                { Output1->SetWindowText(L"You win. You guessed the
number!"); }
                else if(TheGuess > NUM)
                { Output1->SetWindowText(L"Sorry. The number is
smaller."); }
                else
                { Output1->SetWindowText(L"Sorry. The number is
bigger."); }

                char M[10];
                itoa(NUM,M,10);
                //Need to convert ot to CString for MesageBox
                 CString TEMP = CString(M);
                MessageBox(TEMP);
                Input->SetFocus();
                Input->SetSel(0, -1, false);
}
//-----------------------------------------------------
};//close Dialog class
//------------------------------------------------------------------
----------------------
class TheGame : public CWinApp
{
```

```
public:
TheGame() {  }
public:
virtual BOOL InitInstance()
    {
   CWinApp::InitInstance();
   SetRegistryKey(_T("Hills Of Darkness"));
   GAME_FORM dlg;
   m_pMainWnd = &dlg;
   INT_PTR nResponse = dlg.DoModal();
   return FALSE;
} //close function
};//close CWinApp class
//-------------------------------------------------------------------------
----------------------
//Need a Message Map Macro for both CDialog and CWinApp
BEGIN_MESSAGE_MAP(GAME_FORM, CDialog)

      ON_COMMAND(B_ENTER,ENTER)
END_MESSAGE_MAP()
//-------------------------------------------------------------------------
----------------------
TheGame theApp;  //Starts the Application
```

File 5: resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by GuessANumber1.rc
//
#define IDD_Interface                   101
#define CE_INPUT                        1005
#define B_ENTER                         1006
#define CS_Output1                      1007
// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        102
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1001
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

File 6: resource.rc

```
// Microsoft Visual C++ generated resource script.
```

```
//
#include "resource.h"
#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"
////////////////////////////////////////////////////////////////////////////
////
#undef APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
////
// English (U.S.) resources
#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32
#ifdef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
////
//
// TEXTINCLUDE
//
1 TEXTINCLUDE
BEGIN
    "resource.h\0"
END
2 TEXTINCLUDE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END
3 TEXTINCLUDE
BEGIN
    "\r\n"
    "\0"
END
#endif    // APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
////
//
// Dialog
//
IDD_Interface DIALOGEX 0, 0, 417, 181
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION |
    WS_SYSMENU
CAPTION "Guess A Number 1.0"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    DEFPUSHBUTTON   "EXIT",IDOK,217,125,50,14
    EDITTEXT        CE_INPUT,97,94,221,20,ES_CENTER | ES_AUTOHSCROLL
```

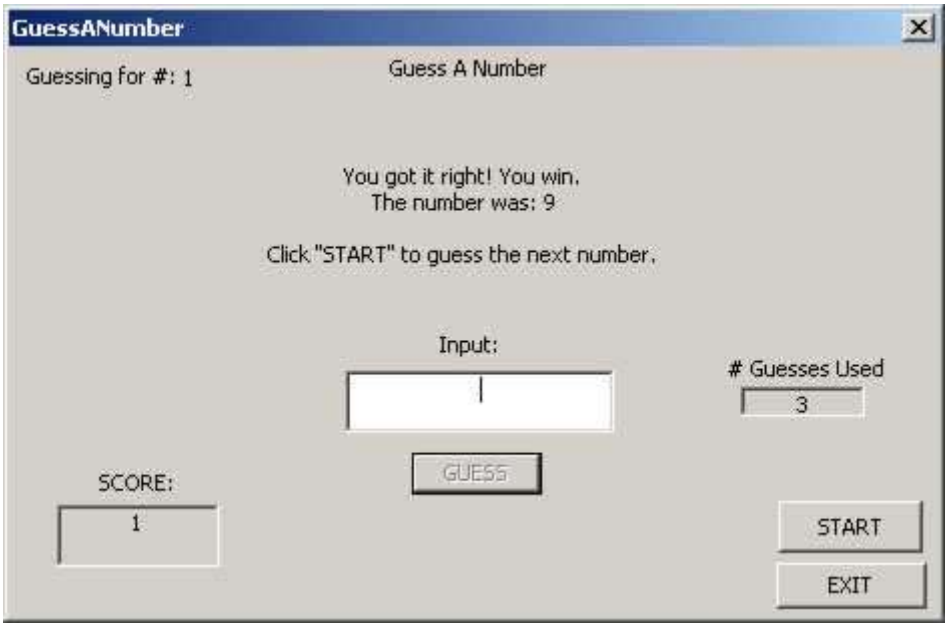
```
    PUSHBUTTON      "ENTER",B_ENTER,161,124,50,14
    CTEXT           "",CS_Output1,62,20,299,32,WS_BORDER
END
/////////////////////////////////////////////////////////////////////////////
////
//
// DESIGNINFO
//
#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO
BEGIN
    IDD_Interface, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 410
        TOPMARGIN, 7
        BOTTOMMARGIN, 174
    END
END
#endif    // APSTUDIO_INVOKED
#endif    // English (U.S.) resources
/////////////////////////////////////////////////////////////////////////////
////

#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 3 resource.
//
/////////////////////////////////////////////////////////////////////////////
////
#endif    // not APSTUDIO_INVOKED
```


MFC Number Guessing Game 2

C++ 2008 Visual Studio Win 32 Project: MFC Guess a Number Game 2

Binary Executable: 2008_NumberGuess2.exe

Objective: To utilize MFC components and create a simple number guessing game.



File 1: GUI.cpp

```
/*
01/16/2010 C. Germany
2008 Notes:
-------------------------------------------------------------------------
-----
To create a scaled down 2008 template project:
   1. Create an EMPTY Win32 project. Click -> "File" -> -> "New"
         -> "Project" -> "Win32" -> "Win32 Project".
   2. Name is and select a directory.
   3. Select "Application Settings" -> "Empty Project" then click
"Finish".
   4. Rt-click project, select "Properties" -> "Configuration
Properties"
      -> "General" -> "Use of MFC" and change it to
      "Use MFC in a Static Library". This will give you MFC
components.
   5. Disable Incremental Linking. Rt-click project, select
"Properties"
         -> "Configuration Properties" -> "Linker" -> "General" ->
      "Enable Incremental Linking" and set it to "NO".
   6. Add a resources file. Rt-click resources and add a DIALOG
object.
   7. Change the enumerated constant in the CDialog class to match
the
         Dialog object ID in the resource file.
-------------------------------------------------------------------------
-----
```

```
To remove 2008 warnings (you can ignore them if you choose):
   1.Rt-click PROJECT and select -> "Properties" -> "Configuration
Properties"
     -> "C/C++" -> "General" -> "Debug Information Formant" and
     -> "Program Database for Edit and Continue /ZI". Set it to
"Disabled".
   2.Rt-click PROJECT and select -> "Properties" -> "Configuration
Properties"
     -> "C/C++" -> "Code Generation" -> "Enable Minimal Rebuild" and
     -> "Yes (/Gm)". Set it to "No".
   3.Use "_atoi_s()" instead of "itoa()" to avoid deprecation
warning.
-------------------------------------------------------------------
-----
Significant changes needed for going from 2003 to 2008 are:
   1. When calling SetWindowText() in 2008 the string passed in must
be
      cast/converted to unicode by prefixing it with "L".
   2. You must use CStrings with SetWindowText() and MessageBox.
   3. You must convert from CString to char array and char array to
CString
      when using atoi and itoa with GetWindowText() and
SetWindowText().
-------------------------------------------------------------------
-----
*/
#include <afxwin.h>     // MFC core and standard components
#include "resource.h"   // main symbols
//Globals
CStatic * pOUTPUT;
CStatic * pNUMGUESSES;
CEdit * pInput;
CStatic * pNumberBeingGuessed;
CStatic * pSCORE;
CButton * pButtonGuess;
CButton * pButtonStart;
int TheNumber = 0;
int Counter = 0;
int NumBeingGuessed = 1;
int score = 0;
CString TEMP;
CString message;
class GuessNumber : public CDialog
{
    public:
    GuessNumber(CWnd* pParent = NULL): CDialog(GuessNumber::IDD,
pParent)
    {    }
    // Dialog Data, name of dialog form
    enum{IDD = IDD_Interface};
    protected:
    virtual void DoDataExchange(CDataExchange* pDX) {
CDialog::DoDataExchange(pDX); }
//-----------------------------------------------------------------
```

```
        virtual BOOL OnInitDialog()
    {
            CDialog::OnInitDialog();

                   pOUTPUT = (CStatic *) GetDlgItem(IDC_OUTPUT);
            pNUMGUESSES = (CStatic *) GetDlgItem(IDC_GUESSES);
            pInput = (CEdit *) GetDlgItem(IDC_INPUT);
                      pButtonGuess = (CButton *)
GetDlgItem(IDC_BUTTON_GUESS);
            pButtonStart = (CButton *) GetDlgItem(IDC_START);
            pNumberBeingGuessed = (CStatic *)
GetDlgItem(CS_NumBeingGuessed);
                      pSCORE = (CStatic *) GetDlgItem(CS_SCORE);
                      score = 0;
                      pButtonGuess->ShowWindow(false);
                      //Cast time_t to unsigned int to avoid a
warning
                      srand((unsigned int)time(NULL));

            return true;
    }
//--------------------------------------------------------------
public:
afx_msg void start()
{
           pButtonGuess->ShowWindow(true);
        pButtonGuess->EnableWindow(true);
        pOUTPUT->SetWindowText(L"\n\rGuess a number from 1 and 10.");
               TheNumber = (rand()%10) + 1;
               pButtonStart->ShowWindow(false);
        char N[10];
               _itoa_s(NumBeingGuessed,N,10);
        TEMP = CString(N);
               pNumberBeingGuessed->SetWindowText(TEMP);
               pInput->SetFocus();
}
//--------------------------------------------------------------
afx_msg void guess()
{
            char N[10];
            _itoa_s(NumBeingGuessed,N,10);
                TEMP = N;
                pNumberBeingGuessed->SetWindowText(TEMP);

            if(NumBeingGuessed <= 3)
                { Guess1Number(); }
                else
                { MessageBox(L"Game Over!"); }
                pInput->SetFocus();
}
//--------------------------------------------------------------
void Guess1Number()
{
```

```
            Counter++;
                char NumGuesses[10];
        _itoa_s(Counter,NumGuesses,10);
        TEMP = NumGuesses;
        pNUMGUESSES->SetWindowText(TEMP);
            char TheGuess[10];
        pInput->GetWindowText(TEMP);
               //Convert CString to char array, cast to char to avoid
warning
        for(int x = 0; x < TEMP.GetLength(); x++)
               { TheGuess[x] = (char) TEMP[x]; }
               int NumGuessed = atoi(TheGuess);

        _itoa_s(TheNumber,TheGuess,10);
        TEMP = TheGuess;
               if(NumGuessed == TheNumber)
               {
             GuessedIt();
               }
               else if(Counter < 3)
               {
                   if(NumGuessed > TheNumber)
                   { message = "\n\rThe number is smaller."; }
                   else
                   { message = "\n\rThe number is larger."; }
               }
               else
               {
            DidNotGuessIt();
               }
               pOUTPUT->SetWindowText(message);
               pInput->SetWindowText(L"");

}
//---------------------------------------------------------------------
void GuessedIt()
{
         message = "\n\rYou got it right! You win.";
         message = message + "\n\rThe number was: " + TEMP;
         if(NumBeingGuessed < 3)
         {
        message = message + "\n\r\n\rClick \"START\" to guess"
                          + " the next number.";
            Counter = 0;
            pButtonGuess->EnableWindow(false);
            pButtonStart->ShowWindow(true);
            score++;
            char COUNT[20];
            _itoa_s(score,COUNT,10);
            TEMP = COUNT;
        pSCORE->SetWindowText(TEMP);
            NumBeingGuessed++;
         }
         else
```

```
         {
         message = message + "\n\r\n\rThree numbers = Game Over.";
         }
}
//-------------------------------------------------------------
void DidNotGuessIt()
{
         message = "\n\rSorry you didn't guess it in 3 guesses.";
     message = message + "\n\rThe number was: " + TEMP + ".";
     if(NumBeingGuessed < 3)
         {
        message = message + "\n\r\n\rClick \"START\" to guess"
                          + " the next number.";
        Counter = 0;
            pButtonGuess->EnableWindow(false);
            pButtonStart->ShowWindow(true);
        NumBeingGuessed++;
         }
         else
         {
        message = message + "\n\r\n\rThree numbers = Game Over.";
         }
}
//-------------------------------------------------------------
DECLARE_MESSAGE_MAP()
};
//-------------------------------------------------------------------
----------------------
class TheProgram : public CWinApp
{
public:
TheProgram() {  }
virtual BOOL InitInstance()
{
        CWinApp::InitInstance();
        GuessNumber dlg;
        m_pMainWnd = &dlg;
        INT_PTR nResponse = dlg.DoModal();
        return FALSE;
} //close function
};
//-------------------------------------------------------------------
----------------------
//Need a Message Map Macro for both CDialog and CWinApp
BEGIN_MESSAGE_MAP(GuessNumber, CDialog)
      ON_COMMAND(IDC_START,start)
      ON_COMMAND(IDC_BUTTON_GUESS,guess)
END_MESSAGE_MAP()
//-------------------------------------------------------------------
----------------------
TheProgram BANANA;  //Starts the Application
```

File 5: resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by GuessANumber2.rc
//
#define IDD_Interface                   101
#define IDC_INPUT                       1001
#define IDC_HelloStatic                 1002
#define IDC_OUTPUT                      1003
#define IDC_BUTTON_GUESS                1004
#define IDC_GUESSES                     1005
#define IDC_START                       1006
#define CS_NumBeingGuessed              1007
#define CS_SCORE                        1008
#define IDC_EDIT1                       1009
#define IDD_GuessNumber                 1010
// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        102
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1011
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

File 6: resource.rc

```
// Microsoft Visual C++ generated resource script.
//
#include "resource.h"
#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"
////////////////////////////////////////////////////////////////////////////
////
#undef APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
////
// English (U.S.) resources
#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32
```

```
#ifdef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////
////
//
// TEXTINCLUDE
//
1 TEXTINCLUDE
BEGIN
    "resource.h\0"
END
2 TEXTINCLUDE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END
3 TEXTINCLUDE
BEGIN
    "\r\n"
    "\0"
END
#endif    // APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////
////
//
// Dialog
//
IDD_Interface DIALOGEX 0, 0, 309, 175
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION |
WS_SYSMENU
CAPTION "GuessANumber"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    DEFPUSHBUTTON   "EXIT",IDOK,255,158,50,14
    CTEXT           "Guess A Number",IDC_HelloStatic,72,2,162,10
    EDITTEXT        IDC_INPUT,112,99,89,19,ES_CENTER | ES_AUTOHSCROLL
    CTEXT           "Input:",IDC_STATIC,131,87,44,10
    CTEXT           "",IDC_OUTPUT,17,27,268,50
    DEFPUSHBUTTON   "GUESS",IDC_BUTTON_GUESS,134,124,43,13
    CTEXT           "",IDC_GUESSES,243,104,41,9,WS_BORDER
    CTEXT           "# Guesses Used",IDC_STATIC,239,94,52,8
    PUSHBUTTON      "START",IDC_START,256,139,48,16
    LTEXT           "Guessing for #:",IDC_STATIC,6,4,51,12
    LTEXT           "",CS_NumBeingGuessed,58,5,19,8
    CTEXT           "SCORE:",IDC_STATIC,12,129,62,10
    CTEXT           "",CS_SCORE,16,140,54,19,WS_BORDER
END
#endif    // English (U.S.) resources
//////////////////////////////////////////////////////////////////////////
////

#ifndef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////
////
//
```

```
// Generated from the TEXTINCLUDE 3 resource.
//
/////////////////////////////////////////////////////////////////////////////
////
#endif    // not APSTUDIO_INVOKED
```


2008 MFC Calculator Application 1

C++ 2008 Visual Studio Win 32 Project: Event Programming with the MFC

Binary Executable: 2008_Calculator1.exe

Objective: To utilize MFC components to create a graphical calculator.

File 1: GUI.cpp

```
//TEMPLATE
//-----------------------------------------------------------------------
----------------------
#include <afxwin.h>       // MFC core and standard components
#include "resource.h"     // main symbols
//-----------------------------------------------------------------------
----------------------
//Globals
CEdit * InputOutput;
int NUM1 = 0;
int NUM2 = 0;
int RESULT = 0;
char OP = '~';
bool FirstOp = true;
bool ClearBox = false;
CString TEMP;
class CALCULATOR : public CDialog
{
```

```
    public:
    CALCULATOR(CWnd* pParent = NULL): CDialog(CALCULATOR::IDD,
pParent)
    {    }
    // Dialog Data, name of dialog form
    enum{IDD = IDD_Interface};
    protected:
    virtual void DoDataExchange(CDataExchange* pDX) {
CDialog::DoDataExchange(pDX); }
    //Called right after constructor. Initialize things here.
    virtual BOOL OnInitDialog()
    {
            CDialog::OnInitDialog();
            InputOutput = (CEdit *) GetDlgItem(CE_InputOutput);
                      Initialize();
            return true;
    }
public:
DECLARE_MESSAGE_MAP()
//-------------------------------------------------------------------
--
void equals()
{
     OPERATION();
        FirstOp = false;
}
//-------------------------------------------------------------------
--
void OPERATION()
{
            char X[50] = "0";
            InputOutput->GetWindowText(TEMP);
                for(int x = 0; x < TEMP.GetLength(); x++)
                { X[x] = (char) TEMP[x]; }
        if(FirstOp)
        {
           NUM1 = atoi(X);
               FirstOp = false;
               ClearBox = true;
        }
        else
        {
                  NUM2 = atoi(X);
              switch(OP)
                  {
                        case '+' : NUM1 = NUM1 + NUM2; break;
                   case '-' : NUM1 = NUM1 - NUM2; break;
                              case '*' : NUM1 = NUM1 * NUM2; break;
                              case '/' : if(NUM2 != 0) { NUM1 = NUM1 /
NUM2; }
                                                    else { RESULT = 0; }
                                          break;
                              default : MessageBox(L"Should never happen!
ERROR.");
```

```
               }
            _itoa_s(NUM1,X,10);
                   TEMP = X;
                   InputOutput->SetWindowText(TEMP);
                   ClearBox = true;
         }

}
//---------------------------------------------------------------------
--
void keynumber(char N[1])
{
        if(ClearBox)
        {
       TEMP = N;
           InputOutput->SetWindowText(TEMP);
       ClearBox = false;
        }
        else
        {
            char X[21] = "";
            InputOutput->GetWindowText(TEMP);
                for(int x = 0; x < TEMP.GetLength(); x++)
                { X[x] = (char) TEMP[x]; }
            strncat_s(X,N,20);
                TEMP = X;
            InputOutput->SetWindowText(TEMP);
        }
}
//---------------------------------------------------------------------
--
void Initialize()
{
     NUM1 = 0;
     NUM2 = 0;
     RESULT = 0;
     OP = '~';
     FirstOp = true;
     ClearBox = false;
}
afx_msg void button0()  { keynumber("0"); }
afx_msg void button1()  { keynumber("1"); }
afx_msg void button2()  { keynumber("2"); }
afx_msg void button3()  { keynumber("3"); }
afx_msg void button4()  { keynumber("4"); }
afx_msg void button5()  { keynumber("5"); }
afx_msg void button6()  { keynumber("6"); }
afx_msg void button7()  { keynumber("7"); }
afx_msg void button8()  { keynumber("8"); }
afx_msg void button9()  { keynumber("9"); }
afx_msg void clear()
{
        NUM1 = 0;
        NUM2 = 0;
```

```
        FirstOp = true;
        ClearBox = false;
        InputOutput->SetWindowText(L"");
}
afx_msg void add()      { OP = '+'; OPERATION(); }
afx_msg void subtract() { OP = '-'; OPERATION(); }
afx_msg void multiply() { OP = '*'; OPERATION(); }
afx_msg void divide()   { OP = '/'; OPERATION(); }
afx_msg void EQUALS()   { equals(); }
//----------------------------------------------------------------------
--
};
//----------------------------------------------------------------------
----------------------
class TheGame : public CWinApp
{
public:
TheGame() {  }
public:
virtual BOOL InitInstance()
    {
   CWinApp::InitInstance();
   SetRegistryKey(_T("MFC CALCULATOR 1"));
   CALCULATOR dlg;
   m_pMainWnd = &dlg;
   INT_PTR nResponse = dlg.DoModal();
   return FALSE;
} //close function
};
//----------------------------------------------------------------------
----------------------
//Need a Message Map Macro for both CDialog and CWinApp
BEGIN_MESSAGE_MAP(CALCULATOR, CDialog)
      ON_COMMAND(B_0,button0)
      ON_COMMAND(B_1,button1)
      ON_COMMAND(B_2,button2)
      ON_COMMAND(B_3,button3)
      ON_COMMAND(B_4,button4)
      ON_COMMAND(B_5,button5)
      ON_COMMAND(B_6,button6)
      ON_COMMAND(B_7,button7)
      ON_COMMAND(B_8,button8)
      ON_COMMAND(B_9,button9)
          ON_COMMAND(B_CLEAR,clear)
          ON_COMMAND(B_ADD, add)
          ON_COMMAND(B_SUBTRACT, subtract)
          ON_COMMAND(B_MULTIPLY, multiply)
          ON_COMMAND(B_DIVIDE, divide)
      ON_COMMAND(B_EQUALS, EQUALS)
END_MESSAGE_MAP()
//----------------------------------------------------------------------
----------------------
TheGame theApp;  //Starts the Application
```

File 2: resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Calculator2.rc
//
#define IDD_Interface                     101
#define B_0                             1001
#define B_1                             1002
#define B_2                             1003
#define B_3                             1004
#define B_4                             1005
#define B_5                             1006
#define B_6                             1007
#define B_7                             1008
#define B_8                             1009
#define B_9                             1010
#define B_EQUALS                        1011
#define B_ADD                           1012
#define B_SUBTRACT                      1013
#define B_MULTIPLY                      1014
#define B_DIVIDE                        1015
#define CE_InputOutput                  1016
#define IDC_BUTTON1                     1017
#define B_CLEAR                         1018
// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        102
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1019
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

File 3: resource.rc

```
// Microsoft Visual C++ generated resource script.
//
#include "resource.h"
#define APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"
```

```
//////////////////////////////////////////////////////////////////////////////
////
#undef APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////////////
////
// English (U.S.) resources
#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32
#ifdef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////////
////
//
// TEXTINCLUDE
//
1 TEXTINCLUDE
BEGIN
    "resource.h\0"
END
2 TEXTINCLUDE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END
3 TEXTINCLUDE
BEGIN
    "\r\n"
    "\0"
END
#endif    // APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////////
////
//
// Dialog
//
IDD_Interface DIALOGEX 0, 0, 179, 140
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION |
WS_SYSMENU
CAPTION "Dialog"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    DEFPUSHBUTTON   "EXIT",IDOK,119,67,32,14
    PUSHBUTTON      "0",B_0,45,110,18,12
    PUSHBUTTON      "1",B_1,24,96,18,12
    PUSHBUTTON      "2",B_2,45,96,18,12
    PUSHBUTTON      "3",B_3,66,96,18,12
    PUSHBUTTON      "4",B_4,24,82,18,12
    PUSHBUTTON      "5",B_5,45,82,18,12
    PUSHBUTTON      "6",B_6,66,82,18,12
    PUSHBUTTON      "7",B_7,24,68,18,12
    PUSHBUTTON      "8",B_8,45,68,18,12
    PUSHBUTTON      "9",B_9,66,68,18,12
```

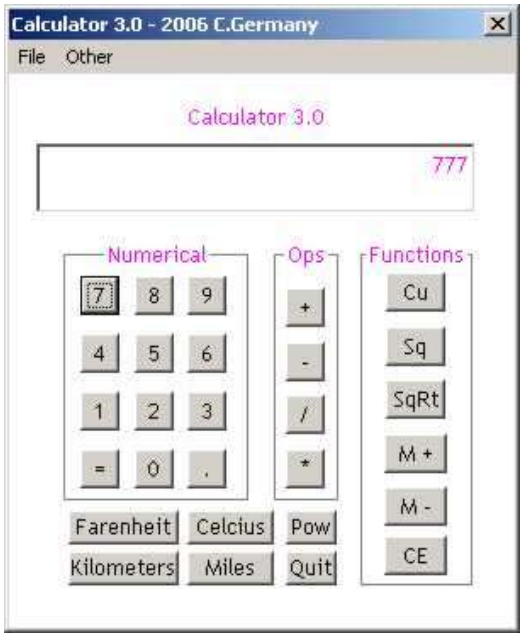
```
    PUSHBUTTON      "=",B_EQUALS,66,110,18,12
    PUSHBUTTON      "+",B_ADD,95,68,18,12
    PUSHBUTTON      "-",B_SUBTRACT,95,82,18,12
    PUSHBUTTON      "*",B_MULTIPLY,95,97,18,12
    PUSHBUTTON      "/",B_DIVIDE,95,112,18,12
    EDITTEXT        CE_InputOutput,16,24,139,26,ES_AUTOHSCROLL |
ES_READONLY
    CTEXT           "OUTPUT",IDC_STATIC,35,13,100,9
    PUSHBUTTON      "CLEAR",B_CLEAR,119,84,32,15
END
/////////////////////////////////////////////////////////////////////////////
////
//
// DESIGNINFO
//
#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO
BEGIN
    IDD_Interface, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 172
        TOPMARGIN, 7
        BOTTOMMARGIN, 133
    END
END
#endif    // APSTUDIO_INVOKED
#endif    // English (U.S.) resources
/////////////////////////////////////////////////////////////////////////////
////

#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 3 resource.
//
/////////////////////////////////////////////////////////////////////////////
////
#endif    // not APSTUDIO_INVOKED
```


2008 MFC Calculator Application 2

C++ 2008 Visual Studio Win 32 Project: Event Programming with the MFC

Binary Executable: 2008_Calculator2.exe

Objective: To utilize MFC components to create a graphical calculator.



File 1: GUI.cpp

```
/*
01/16/2010 C. Germany
2008 Notes:
-------------------------------------------------------------------------
-----
To create a scaled down 2008 template project:
   1. Create an EMPTY Win32 project. Click -> "File" -> -> "New"
         -> "Project" -> "Win32" -> "Win32 Project".
   2. Name is and select a directory.
   3. Select "Application Settings" -> "Empty Project" then click
"Finish".
   4. Rt-click project, select "Properties" -> "Configuration
Properties"
      -> "General" -> "Use of MFC" and change it to
      "Use MFC in a Static Library". This will give you MFC
components.
   5. Disable Incremental Linking. Rt-click project, select
"Properties"
         -> "Configuration Properties" -> "Linker" -> "General" ->
      "Enable Incremental Linking" and set it to "NO".
   6. Add a resources file. Rt-click resources and add a DIALOG
object.
   7. Change the enumerated constant in the CDialog class to match
the
         Dialog object ID in the resource file.
-------------------------------------------------------------------------
-----
To remove 2008 warnings (you can ignore them if you choose):
   1.Rt-click PROJECT and select -> "Properties" -> "Configuration
Properties"
```

```
     -> "C/C++" -> "General" -> "Debug Information Formant" and
     -> "Program Database for Edit and Continue /ZI". Set it to
"Disabled".
   2.Rt-click PROJECT and select -> "Properties" -> "Configuration
Properties"
     -> "C/C++" -> "Code Generation" -> "Enable Minimal Rebuild" and
     -> "Yes (/Gm)". Set it to "No".
   3.Use "_atoi_s()" instead of "itoa()" to avoid deprecation
warning.
   4.Use "sprintf_s()" instead of "sprintf()" to avoid deprecation
warning.
-------------------------------------------------------------------
-----
Significant changes needed for going from 2003 to 2008 are:
   1. When calling SetWindowText() in 2008 the string passed in must
be
      cast/converted to unicode by prefixing it with "L".
   2. You must use CStrings with SetWindowText() and MessageBox.
   3. You must convert from CString to char array and char array to
CString
      when using atoi and itoa with GetWindowText() and
SetWindowText().
-------------------------------------------------------------------
-----
*/
//-----------------------------------------------------------------
----------------------
#include <afxwin.h>     // MFC core and standard components
#include <afxcmn.h>
#include "resource.h"   // main symbols
#include <cmath>
//-----------------------------------------------------------------
----------------------
//Globals
CEdit * pInputOutput;
CButton * pEquals;
CBrush CalculatorBrush;
double Number1;
double Number2;
double Result;
double Memory;
bool NextExpression;
bool NeedToClear;
bool ClickedEquals;
bool PasteMemory;
char oper;
CString TEMP;
class Calculator : public CDialog
{
    public:
    Calculator(CWnd* pParent = NULL): CDialog(Calculator::IDD,
pParent) {    }
    // Dialog Data, name of dialog form
    enum{IDD = IDD_Interface};
```

```
    //-------------------------------------------------------
    //Called right after constructor. Initialize things here.
    virtual BOOL OnInitDialog()
    {
            CDialog::OnInitDialog();
               pInputOutput = (CEdit *) GetDlgItem(IDC_InputOutput);
                      pEquals = (CButton *)
GetDlgItem(IDC_NumEquals);
               //pInputOutput->SetFocus();
               CalculatorBrush.CreateSolidBrush(RGB(255, 255, 255));
            Initialize();
            return true;
    }
    //-------------------------------------------------------
   afx_msg HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
   {

         switch (nCtlColor)
      {
                  case CTLCOLOR_EDIT:   pDC-
>SetTextColor(RGB(255,0,255));
              case CTLCOLOR_STATIC: pDC-
>SetTextColor(RGB(255,0,255));
              case CTLCOLOR_DLG:    return CalculatorBrush;
              default:  return CDialog::OnCtlColor(pDC, pWnd,
nCtlColor);
      }

   }
//-------------------------------------------------------
//To Get Keyboard Messages, bypass buggy CDialog MFC Keyboard Message
Handlers
//Use ASCII values. Prevent duplicate input with if/else and
WM_KEYDOWN
//IMPORTANT: Set CEdit control to read only or it will process input
twice.
//Unlike +,- and /, the * key is also key for number 8 so must handle
it.
BOOL PreTranslateMessage(MSG* pMsg)
{
     int nVirtKey = (int) pMsg->wParam;
        if(nVirtKey == 42 || nVirtKey == 43 || nVirtKey == 45 ||
              nVirtKey == 47 || nVirtKey == 61)
        {
                      if(nVirtKey == 42)
                      {   //Calls 8 too so have to shave off last
character
                pInputOutput->GetWindowText(TEMP);
                               TEMP.Delete(TEMP.GetLength()-1,1);
                               pInputOutput->SetWindowText(TEMP);
                               Calculate('m');
                      } // * MULTIPLY
            if(nVirtKey == 43)    { Calculate('a'); } // + PLUS
               if(nVirtKey == 45)    { Calculate('s'); } // - MINUS
```

```
                if(nVirtKey == 47)    { Calculate('d'); } // / DIVIDE
            if(nVirtKey == 61)    { Equals(); }       // = EQUALS
        }
     else if(pMsg->message == WM_KEYDOWN)
        {
            if(nVirtKey == 48)    { KeyNumber("0"); }
            if(nVirtKey == 49)    { KeyNumber("1"); }
            if(nVirtKey == 50)    { KeyNumber("2"); }
            if(nVirtKey == 51)    { KeyNumber("3"); }
            if(nVirtKey == 52)    { KeyNumber("4"); }
            if(nVirtKey == 53)    { KeyNumber("5"); }
            if(nVirtKey == 54)    { KeyNumber("6"); }
            if(nVirtKey == 55)    { KeyNumber("7"); }
                if(nVirtKey == 56)    { KeyNumber("8"); }
            if(nVirtKey == 57)    { KeyNumber("9"); }

            //More examples. See ASCII Chart at Bottom for codes.
            if(nVirtKey == 65) { MessageBox(L"The \"a\" Key Was
Pressed!"); }
                if(nVirtKey == VK_UP) { MessageBox(L"Up Key
Pressed!"); }
         if(nVirtKey == VK_DOWN) { MessageBox(L"Down Key Pressed!");
}
         if(nVirtKey == '\r')  { Equals(); }
        }

        return CDialog::PreTranslateMessage(pMsg);
}
//-------------------------------------------------------
    //Message Handlers for Dialog class object
       afx_msg void key0() { KeyNumber("0"); }
       afx_msg void key1() { KeyNumber("1"); }
       afx_msg void key2() { KeyNumber("2"); }
       afx_msg void key3() { KeyNumber("3"); }
       afx_msg void key4() { KeyNumber("4"); }
       afx_msg void key5() { KeyNumber("5"); }
       afx_msg void key6() { KeyNumber("6"); }
       afx_msg void key7() { KeyNumber("7"); }
       afx_msg void key8() { KeyNumber("8"); }
       afx_msg void key9() { KeyNumber("9"); }
       afx_msg void keyPeriod() { KeyNumber("."); }
       afx_msg void keyEquals() { Equals(); }
       afx_msg void keyAdd() { Calculate('a'); }
       afx_msg void keySubtract() { Calculate('s'); }
       afx_msg void keyMultiply() { Calculate('m'); }
       afx_msg void keyDivide() { Calculate('d'); }
       afx_msg void keyPower() { Calculate('p'); }
       afx_msg void keyCube() { Cube(); }
       afx_msg void keySquare() { Square(); }
       afx_msg void keySquareRoot() { SquareRoot(); }
       afx_msg void keyMemory() { MemoryKey(); }
       afx_msg void EraseMemory() { MemoryClear(); }
       afx_msg void keyCE() { Erase(); }
       afx_msg void keyFarenheit() { CelciusToFarenheit(); }
```

```
        afx_msg void keyCelcius() { FarenheitToCelcius(); }
        afx_msg void keyMiles() { MilesToKilometers(); }
        afx_msg void keyKilometers() { KilometersToMiles(); }
        afx_msg void Quit() { EndDialog(IDOK); }
    //-------------------------------------------------------

        afx_msg void About()
        {
                MessageBox(L"Calculator 3.0 - 2006 - C. Germany",
L"Calculator 3.0 - HELP");
        }
    //-------------------------------------------------------
        afx_msg void Help()
        {
             //Illustrates how to use multi-array and convert. Could
just use CString.
                 char HELPTEXT[5][55] = {
                                "This program is a fully functional
calculator.\r\n",
                            "It was created as an MFC tutorial for
students.\n",
                                "Its purpose is to display the logic
processes\n",
                                "and progam flow between the user
interface and \n",
                                "processing of input through the MFC
CEdit control."
                        };
             char Message[280] = "";
                 for(int z = 0; z < 5; z++)
             { strncat_s(Message, HELPTEXT[z], 55); }
                           CString TEMP = Message;
                 MessageBox(TEMP, L"Calculator 3.0 - HELP");
        }

    //-------------------------------------------------------
    //Calculator Functions
        void Initialize()
        {
         NextExpression = false;
             NeedToClear = false;
             ClickedEquals = false;
             PasteMemory = false;
             Number1 = 0.0;
             Number2 = 0.0;
             Result = 0.0;
             Memory = 0.0;
             oper = ' ';
        }
    //-------------------------------------------------------
    //In 2008 take in put as a CString and convert it to char array
        void KeyNumber(char digit[1])
        {
             if(NeedToClear)
```

```
                {
                        pInputOutput->SetWindowText(L"");
                        NeedToClear = false;
                }
        pInputOutput->GetWindowText(TEMP);
                char num[21] = "";
                for(int x = 0; x < TEMP.GetLength(); x++)
                { num[x] = (char) TEMP[x]; }
                strncat_s(num,digit,20);
                TEMP = num;
            pInputOutput->SetWindowText(TEMP);
       }
    //-------------------------------------------------------
    void Calculate(char op)
    {
         oper = op;
         pInputOutput->GetWindowText(TEMP);
                char num[21] = "";
                for(int x = 0; x < TEMP.GetLength(); x++)
                { num[x] = (char) TEMP[x]; }
         if(NextExpression)
         {
                if(!ClickedEquals)
                {
                 Number2 = atof(num);
                 switch(oper)
                 {
                     case 'a' : Number1 = Number1 + Number2; break;
                     case 's' : Number1 = Number1 - Number2; break;
                     case 'm' : Number1 = Number1 * Number2; break;
                                        case 'd' : if(Number2 == 0) {
Number1 = 0; }
                                                                   else {
Number1 = Number1 / Number2; }
                                                                   break;
                     case 'p' : Number1 = pow(Number1,Number2);
break;
                     default : break;
                 }

                 char Answer[20];

                        //itoa(Number1, Answer, 10); //Converts int,
but truncates decimal
                 //Instead, use sprintf() to convert double or double
to string

                                sprintf_s(Answer,"%g",Number1);
                                TEMP = Answer;
                 pInputOutput->SetWindowText(TEMP);
                        NeedToClear = true;
                }
             else { ClickedEquals = false; }
       }
```

```
   else
   {
           Number1 = atof(num);
           NeedToClear = true;
           NextExpression = true;
   }
}//close Calculate()
//-------------------------------------------------------
void Equals()
{
        Calculate(oper);
        NeedToClear = true;
        ClickedEquals = true;
}
//-------------------------------------------------------
void Erase()
{
        pInputOutput->SetWindowText(L"");
     NextExpression = false;
        NeedToClear = false;
        ClickedEquals = false;
        Number1 = 0.0;
        Number2 = 0.0;
        Result = 0.0;
        oper = ' ';
}
//-------------------------------------------------------
void Cube()
{
     char num[20] = "";
        double numtemp = 0.0;
        pInputOutput->GetWindowText(TEMP);
     for(int x = 0; x < TEMP.GetLength(); x++)
            { num[x] = (char) TEMP[x]; }
        numtemp = atof(num);
     numtemp = numtemp * numtemp * numtemp;
        sprintf_s(num,"%g",numtemp);
            TEMP = num;
        pInputOutput->SetWindowText(TEMP);
}
//-------------------------------------------------------
void Square()
{
     char num[20] = "";
        double numtemp = 0.0;
        pInputOutput->GetWindowText(TEMP);
     for(int x = 0; x < TEMP.GetLength(); x++)
            { num[x] = (char) TEMP[x]; }
        numtemp = atof(num);
     numtemp = numtemp * numtemp;
        sprintf_s(num,"%g",numtemp);
            TEMP = num;
        pInputOutput->SetWindowText(TEMP);
}
```

```
    //-------------------------------------------------------
    void SquareRoot()
    {
             char num[20] = "";
             pInputOutput->GetWindowText(TEMP);
                 for(int x = 0; x < TEMP.GetLength(); x++)
                 { num[x] = (char) TEMP[x]; }
             Number1 = atof(num);
                 Number1 = sqrt(Number1);
          sprintf_s(num,"%g",Number1);
                 TEMP = num;
             pInputOutput->SetWindowText(TEMP);
    }
    //-------------------------------------------------------
    void MemoryKey()
    {
          char num[20] = "";
          if(PasteMemory)
              {
             sprintf_s(num,"%g",Memory);
                        TEMP = num;
                pInputOutput->SetWindowText(TEMP);
              }
              else
              {
             pInputOutput->GetWindowText(TEMP);
             for(int x = 0; x < TEMP.GetLength(); x++)
                    { num[x] = (char) TEMP[x]; }
             Memory = atof(num);
                 PasteMemory = true;
              }
    }
    //-------------------------------------------------------
    void MemoryClear()
        {
                 PasteMemory = false;
                 Memory = 0.0;
        }
    //-------------------------------------------------------
    void CelciusToFarenheit()
    {
          char num[20];
          pInputOutput->GetWindowText(TEMP);
          for(int x = 0; x < TEMP.GetLength(); x++)
                 { num[x] = (char) TEMP[x]; }

                 double Farenheit = atof(num);
              Farenheit = (Farenheit * 9 / 5) + 32;
          sprintf_s(num,"%g",Farenheit);
                 TEMP = num;
          pInputOutput->SetWindowText(TEMP);
    }
//-------------------------------------------------------------
    void FarenheitToCelcius()
```

```
    {
         char num[20];
         pInputOutput->GetWindowText(TEMP);
         for(int x = 0; x < TEMP.GetLength(); x++)
                { num[x] = (char) TEMP[x]; }
            double Celcius = atof(num);
            Celcius = ((Celcius - 32) * 5) / 9;
         sprintf_s(num,"%g",Celcius);
                TEMP = num;
         pInputOutput->SetWindowText(TEMP);
    }
//---------------------------------------------------------------
    void KilometersToMiles()
    {
             char num[20] = "";
             pInputOutput->GetWindowText(TEMP);
         for(int x = 0; x < TEMP.GetLength(); x++)
                { num[x] = (char) TEMP[x]; }
             double x = atof(num);
             x = x / 1.609347;
             sprintf_s(num,"%g",x);
                 TEMP = num;
             pInputOutput->SetWindowText(TEMP);
    }
//---------------------------------------------------------------
    void MilesToKilometers()
    {
             char num[20] = "";
             pInputOutput->GetWindowText(TEMP);
         for(int x = 0; x < TEMP.GetLength(); x++)
                { num[x] = (char) TEMP[x]; }
                double x = atof(num);
             x = x * 1.609347;
             sprintf_s(num,"%g",x);
                 TEMP = num;
             pInputOutput->SetWindowText(TEMP);
    }
//---------------------------------------------------------------
DECLARE_MESSAGE_MAP()
}; //closing class specification
//-------------------------------------------------------------
BEGIN_MESSAGE_MAP(Calculator, CDialog)
      //Listen for and map COLOR events
      ON_WM_CTLCOLOR()
      ON_COMMAND(IDC_Num0, key0)
      ON_COMMAND(IDC_Num1, key1)
      ON_COMMAND(IDC_Num2, key2)
      ON_COMMAND(IDC_Num3, key3)
      ON_COMMAND(IDC_Num4, key4)
      ON_COMMAND(IDC_Num5, key5)
      ON_COMMAND(IDC_Num6, key6)
      ON_COMMAND(IDC_Num7, key7)
      ON_COMMAND(IDC_Num8, key8)
      ON_COMMAND(IDC_Num9, key9)
```

```
      ON_COMMAND(IDC_Num7, key7)
      ON_COMMAND(IDC_Num8, key8)
      ON_COMMAND(IDC_Num9, key9)
      ON_COMMAND(IDC_NumPeriod, keyPeriod)
      ON_COMMAND(IDC_NumEquals, keyEquals)
      ON_COMMAND(IDC_NumAdd, keyAdd)
      ON_COMMAND(IDC_NumSubtract, keySubtract)
      ON_COMMAND(IDC_NumMultiply, keyMultiply)
      ON_COMMAND(IDC_NumDivide, keyDivide)
      ON_COMMAND(IDC_NumCube, keyCube)
      ON_COMMAND(IDC_NumSquare, keySquare)
      ON_COMMAND(IDC_NumSquareRoot, keySquareRoot)
      ON_COMMAND(IDC_POWER,keyPower)
      ON_COMMAND(IDC_NumMemory, keyMemory)
      ON_COMMAND(IDC_MemoryErase, EraseMemory)
      ON_COMMAND(IDC_NumCE, keyCE)
      ON_COMMAND(IDC_Farenheit, keyFarenheit)
      ON_COMMAND(IDC_Celcius, keyCelcius)
      ON_COMMAND(IDC_Miles, keyMiles)
      ON_COMMAND(IDC_Kilometers, keyKilometers)
      ON_COMMAND(IDC_QUIT, Quit)
      ON_COMMAND(ID_FILE_EXIT,Quit)
      ON_COMMAND(ID_OTHER_HELP,Help)
      ON_COMMAND(ID_OTHER_ABOUT,About)
END_MESSAGE_MAP()
//-------------------------------------------------------------
class Calc : public CWinApp
{
public:
Calc() {  }
virtual BOOL InitInstance()
{
        CWinApp::InitInstance();
        Calculator x;
        m_pMainWnd = &x;
        INT_PTR nResponse = x.DoModal();
        return FALSE;
} //close function
};
//-------------------------------------------------------------
Calc SuperFun;
/*
sprintf() builds and formats a string from a number you pass into it.
Arguments are:
-------------------------------------------------------------------------
----------------------------
c = Character a
d  = or i Signed decimal integer 392
e = Scientific notation (mantise/exponent) using e character
3.9265e+2
E = Scientific notation (mantise/exponent) using E character
3.9265E+2
f = Decimal floating point 392.65
g = Use the shorter of %e or %f 392.65
```

```
G = Use the shorter of %E or %f 392.65
o = Signed octal 610
s = String of characters sample
u = Unsigned decimal integer 7235
x = Unsigned hexadecimal integer 7fa
X = Unsigned hexadecimal integer (capital letters) 7FA
p = Pointer address B800:0000
n = Nothing printed. The argument must be a pointer to a signed int,
   where the number of characters written so far is stored.
% = A % followed by another % character will write % to the string.
EXAMPLE:
  char buffer [50];
  int n, a=5, b=3;
  n=sprintf (buffer, "%d plus %d is %d", a, b, a+b);
  printf ("[%s] is a %d char long string\n",buffer,n);
------------------------------------------------------------------------
----------------------
Mouse and Keyboard Message Macros and Their Handlers:
      //ON_WM_LBUTTONDOWN( )    //Listen for LEFT mouse button click
          //ON_WM_KEYDOWN()         //Keyboard Events
          //ON_WM_KEYUP()           //Keyboard Events
      //ON_WM_CHAR()            //Keyboard Events
   afx_msg void OnKeyDown(UINT uChar, UINT uRepCnt, UINT uFlg)
   {
           //char x[1];
                  //x[0] = (char)uChar;
                  //MessageBox(x);
           MessageBox("MADE IT to OnChar()!");

           switch(uChar)
           {
                 case '\r': Equals(); break;
                 case '\b': Erase(); break;
                               case '+' : Calculate('a'); break;
                 case '-': Calculate('s'); break;
                 case '*': Calculate('m'); break;
                               case '/' : Calculate('d'); break;
                 case '=' : Equals(); break;
                               case 'a' : MessageBox("Heard an A!");
break;
                               default : break;
                  }
   }
   afx_msg void buttonPushed()
   {
           MessageBox("MADE IT HERE in buttonPushed()!");
   }
   afx_msg void OnLButtonDown( UINT, CPoint )
   {
               MessageBox("Mouse Clicked!", "Event Triggered");
   }
ASCII Values for PreTranslate Messages
//-----------------------------------------------------
Decimal      Character     Description
```

```
0               NUL
1               SOH         start of header
2               STX         start of text
3               ETX         end of text
4               EOT         end of transmission
5               ENQ         enquiry
6               ACK         acknowledge
7               BEL         bell
8               BS          backspace
9               HT          horizontal tab
10              LF          line feed
11              VT          vertical tab
12              FF          form feed
13              CR          carriage return
14              SO          shift out
15              SI          shift in
16              DLE         data link escape
17              DC1         no assignment, but usually XON
18              DC2
19              DC3         no assignment, but usually XOFF
20              DC4
21              NAK         negative acknowledge
22              SYN         synchronous idle
23              ETB         end of transmission block
24              CAN         cancel
25              EM          end of medium
26              SUB         substitute
27              ESC         escape
28              FS          file seperator
29              GS          group seperator
30              RS          record seperator
31              US          unit seperator
32              SPC         space
33  !
34  "
35  #
36  $
37  %
38  &
39  '
40  (
41  )
42  *
43  +
44  ,
45  -
46  .
47  /
48  0
49  1
50  2
51  3
52  4
53  5
```

```
54  6
55  7
56  8
57  9
58  :
59  ;
60  <
61  =
62  >
63  ?
64  @
65  A
66  B
67  C
68  D
69  E
70  F
71  G
72  H
73  I
74  J
75  K
76  L
77  M
78  N
79  O
80  P
81  Q
82  R
83  S
84  T
85  U
86  V
87  W
88  X
89  Y
90  Z
91  [
92  \
93  ]
94  ^
95  _
96  `
97   a
98   b
99   c
100  d
101  e
102  f
103  g
104  h
105  i
106  j
107  k
```

```
108  l
109  m
110  n
111  o
112  p
113  q
114  r
115  s
116  t
117  u
118  v
119  w
120  x
121  y
122  z
123  {
124  |
125  }
126  ~
127  DEL delete
//-------------------------------------------------------------------------
----------
*/
```

File 2: resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Calculator1.rc
//
#define IDD_Interface                   101
#define IDR_MENU1                       102
#define IDD_Calculator                  110
#define IDC_InputOutput                 1001
#define IDC_Num7                        1002
#define IDC_Num8                        1003
#define IDC_Num9                        1004
#define IDC_Num4                        1005
#define IDC_Num5                        1006
#define IDC_Num6                        1007
#define IDC_Num1                        1008
#define IDC_Num2                        1009
#define IDC_Num3                        1010
#define IDC_NumEquals                   1011
#define IDC_Num0                        1012
#define IDC_NumPeriod                   1013
#define IDC_NumAdd                      1014
#define IDC_NumSubtract                 1015
#define IDC_NumDivide                   1016
#define IDC_NumMultiply                 1017
#define IDC_NumCube                     1018
```

```
#define IDC_NumSquare                   1019
#define IDC_NumSquareRoot               1020
#define IDC_NumMemory                   1021
#define IDC_NumCE                       1022
#define IDC_NumCE2                      1023
#define IDC_Farenheit                   1024
#define IDC_Celcius                     1025
#define IDC_Kilometers                  1026
#define IDC_Miles                       1027
#define IDC_MemoryErase                 1028
#define IDC_POWER                       1029
#define IDC_QUIT                        1030
#define ID_FILE_EXIT                    40001
#define ID_OTHER_HELP                   40002
#define ID_OTHER_ABOUT                  40003
// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        103
#define _APS_NEXT_COMMAND_VALUE         40004
#define _APS_NEXT_CONTROL_VALUE         1031
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

File 3: resource.rc

```
// Microsoft Visual C++ generated resource script.
//
#include "resource.h"
#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"
////////////////////////////////////////////////////////////////////////////
////
#undef APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
////
// English (U.S.) resources
#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32
#ifdef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
////
```

```
//
// TEXTINCLUDE
//
1 TEXTINCLUDE
BEGIN
    "resource.h\0"
END
2 TEXTINCLUDE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END
3 TEXTINCLUDE
BEGIN
    "\r\n"
    "\0"
END
#endif    // APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
////
//
// Dialog
//
IDD_Interface DIALOGEX 0, 0, 170, 145
STYLE DS_SETFONT | DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Calculator 3.0 - 2006 C.Germany"
MENU IDR_MENU1
FONT 10, "Trebuchet MS", 400, 0, 0x0
BEGIN
    EDITTEXT        IDC_InputOutput,9,18,148,18,ES_RIGHT | ES_AUTOHSCROLL
| ES_READONLY | ES_NUMBER
    CTEXT           "Calculator 3.0",IDC_STATIC,23,7,121,11
    PUSHBUTTON      "7",IDC_Num7,24,53,13,10
    PUSHBUTTON      "8",IDC_Num8,42,53,13,10
    PUSHBUTTON      "9",IDC_Num9,60,53,13,10
    PUSHBUTTON      "4",IDC_Num4,24,68,13,10
    PUSHBUTTON      "5",IDC_Num5,42,68,13,10
    PUSHBUTTON      "6",IDC_Num6,60,68,13,10
    PUSHBUTTON      "1",IDC_Num1,24,83,13,10
    PUSHBUTTON      "2",IDC_Num2,42,83,13,10
    PUSHBUTTON      "3",IDC_Num3,60,83,13,10
    PUSHBUTTON      "=",IDC_NumEquals,24,98,13,10
    PUSHBUTTON      "0",IDC_Num0,42,98,13,10
    PUSHBUTTON      ".",IDC_NumPeriod,60,98,13,10
    GROUPBOX        "Numerical",IDC_STATIC,18,43,63,69,BS_CENTER
    PUSHBUTTON      "+",IDC_NumAdd,93,56,13,10
    PUSHBUTTON      "-",IDC_NumSubtract,93,70,13,10
    PUSHBUTTON      "/",IDC_NumDivide,93,84,13,10
    PUSHBUTTON      "*",IDC_NumMultiply,93,98,13,10
    GROUPBOX        "Ops",IDC_STATIC,89,43,22,69,BS_CENTER
    PUSHBUTTON      "Cu",IDC_NumCube,127,52,20,10
    PUSHBUTTON      "Sq",IDC_NumSquare,127,66,20,10
    PUSHBUTTON      "SqRt",IDC_NumSquareRoot,127,80,20,10
    PUSHBUTTON      "M +",IDC_NumMemory,126,94,21,10
```

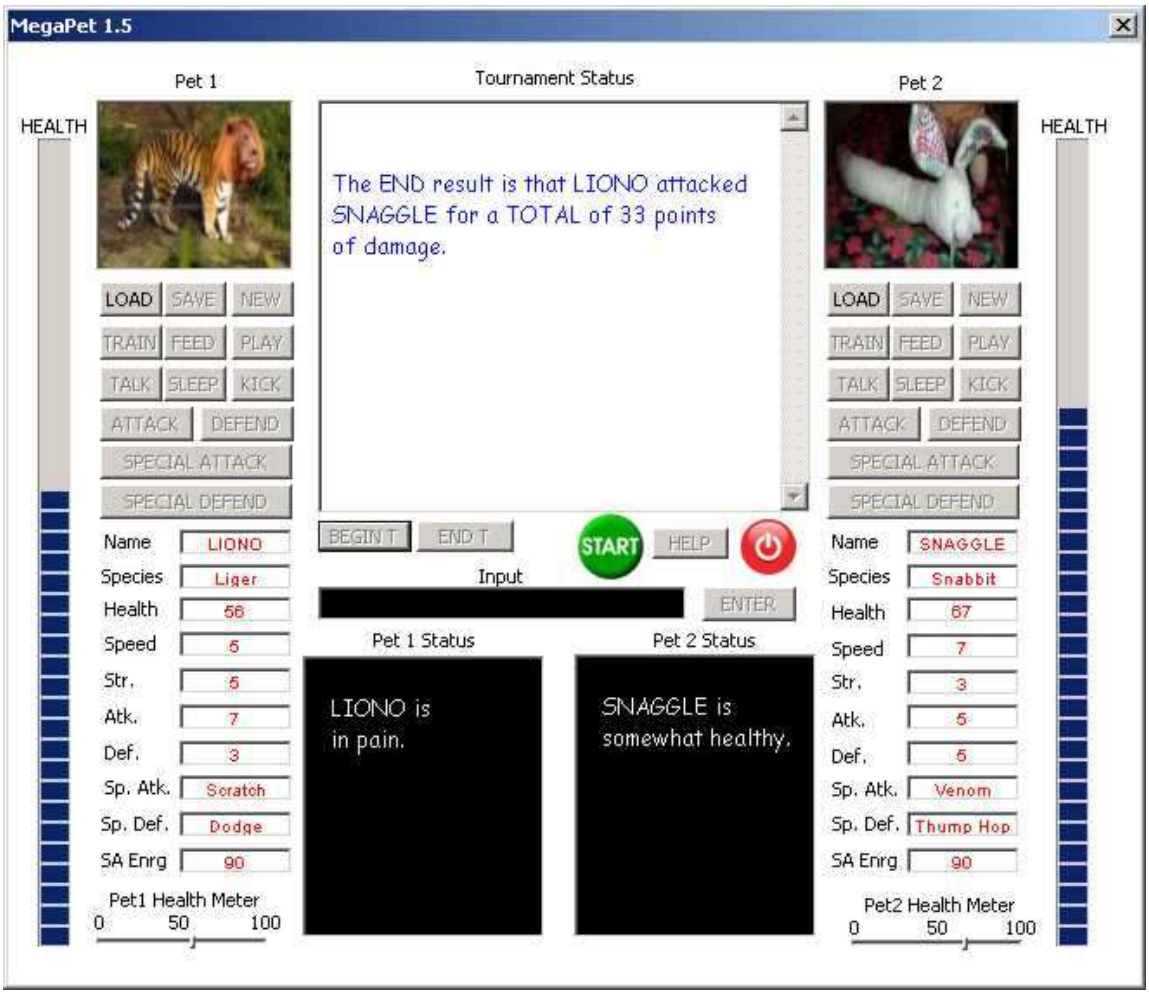
```
    PUSHBUTTON      "CE",IDC_NumCE,126,121,21,10
    GROUPBOX        "Functions",IDC_STATIC,118,43,38,91,BS_CENTER
    PUSHBUTTON      "Farenheit",IDC_Farenheit,20,114,37,9
    PUSHBUTTON      "Celcius",IDC_Celcius,60,114,29,9
    PUSHBUTTON      "Kilometers",IDC_Kilometers,20,125,37,9
    PUSHBUTTON      "Miles",IDC_Miles,60,125,29,9
    PUSHBUTTON      "M -",IDC_MemoryErase,126,108,21,10
    PUSHBUTTON      "Pow",IDC_POWER,93,114,17,9
    PUSHBUTTON      "Quit",IDC_QUIT,93,125,17,9
END
////////////////////////////////////////////////////////////////////////////
////
//
// Menu
//
IDR_MENU1 MENU
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "E&xit",                       ID_FILE_EXIT
    END
    POPUP "&Other"
    BEGIN
        MENUITEM "&Help",                       ID_OTHER_HELP
        MENUITEM "&About",                      ID_OTHER_ABOUT
    END
END
#endif    // English (U.S.) resources
////////////////////////////////////////////////////////////////////////////
////

#ifndef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 3 resource.
//
////////////////////////////////////////////////////////////////////////////
////
#endif    // not APSTUDIO_INVOKED
```


MFC Class Hierarchy Tournament Game

C++ 2008 Visual Studio Win 32 Project:

Binary Executable: 2008_MegaPet2.exe

Objective: To utilize MFC components, an inheritance hierarchy and interface modifications by creating a tournament game.



File 1: GUI.cpp

```
/*
01/16/2010 C. Germany
2008 Notes:
-------------------------------------------------------------------------
-
To create a scaled down 2008 template project:
   1. Create an EMPTY Win32 project. Click -> "File" -> -> "New"
         -> "Project" -> "Win32" -> "Win32 Project".
   2. Name is and select a directory.
   3. Select "Application Settings" -> "Empty Project" then click
"Finish".
   4. Rt-click project, select "Properties" -> "Configuration Properties"
      -> "General" -> "Use of MFC" and change it to
      "Use MFC in a Static Library". This will give you MFC components.
   5. Disable Incremental Linking. Rt-click project, select "Properties"
         -> "Configuration Properties" -> "Linker" -> "General" ->
      "Enable Incremental Linking" and set it to "NO".
   6. Add a resources file. Rt-click resources and add a DIALOG object.
```

```
   7. Change the enumerated constant in the CDialog class to match the
         Dialog object ID in the resource file.
-------------------------------------------------------------------------
-
To remove 2008 warnings (you can ignore them if you choose):
   1.Rt-click PROJECT and select -> "Properties" -> "Configuration
Properties"
     -> "C/C++" -> "General" -> "Debug Information Formant" and
     -> "Program Database for Edit and Continue /ZI". Set it to
"Disabled".
   2.Rt-click PROJECT and select -> "Properties" -> "Configuration
Properties"
     -> "C/C++" -> "Code Generation" -> "Enable Minimal Rebuild" and
     -> "Yes (/Gm)". Set it to "No".
   3.Use "_atoi_s()" instead of "itoa()" to avoid deprecation warning.
   4.Use "sprintf_s()" instead of "sprintf()" to avoid deprecation
warning.
-------------------------------------------------------------------------
-
Significant changes needed for going from 2003 to 2008 are:
   1. When calling SetWindowText() in 2008 the string passed in must be
      cast/converted to unicode by prefixing it with "L".
   2. You must use CStrings with SetWindowText() and MessageBox.
   3. You must convert from CString to char array and char array to
CString
      when using atoi and itoa with GetWindowText() and SetWindowText().
-------------------------------------------------------------------------
-
*/
//The Windows interface for the game.
#include <afxwin.h>      // MFC core and standard components
#include <afxcmn.h>         // MFC components
#include "resource.h"    // main symbols
#include "Afxext.h"      // CBitmapButtons
#include "Globals.h"
#include "Classes.h"
#include "Functions.h"
//*************************************************************************
****
class MegaPet_FORM : public CDialog
{
      public:
      MegaPet_FORM (CWnd* pParent = NULL): CDialog(MegaPet_FORM ::IDD,
pParent)
      {  }
//-------------------------------------------------------------------
      enum{IDD = IDD_Interface};
//-------------------------------------------------------------------
      protected:
      virtual void DoDataExchange(CDataExchange* pDX)
      { CDialog::DoDataExchange(pDX); }
//-------------------------------------------------------------------
      virtual BOOL OnInitDialog()
      {
```

```
                CDialog::OnInitDialog();
            SetUpInterface();
                        Initialize_Interface();


PlaySound(L"Audio/MusicClips/Pokemon.wav",NULL,SND_FILENAME|SND_ASYNC);
            PetBrush_WHITE.CreateSolidBrush(RGB(255, 255, 255));
                        PetBrush_BLACK.CreateSolidBrush(RGB(0, 0, 0));
            PetBrush_RED.CreateSolidBrush(RGB(255, 0, 0));
                        PetBrush_BLUE.CreateSolidBrush(RGB(0, 0, 255));
            MainOutput->SetWindowText(L"\r\n       Click the
\"START\"\r\n       button to begin.");
                        Input->SetFocus();
                        return true;
}
//---------------------------------------------------------------------
void SetUpInterface()
{
        //Setup pointers to Dialog Interface objects
        //Generic Interface Objects
        Button_Start = (CBitmapButton *) GetDlgItem(B_START);
        Button_Start = new CBitmapButton;
     Button_Start->Create(NULL, WS_CHILD|WS_VISIBLE|BS_OWNERDRAW,
        CRect(329,272,0,0), this, B_START);
        Button_Start->LoadBitmaps(IDB_START);
        Button_Start->SizeToContent();
     Button_Exit = (CBitmapButton *) GetDlgItem(B_QUIT);
        Button_Exit = new CBitmapButton;
     Button_Exit->Create(NULL, WS_CHILD|WS_VISIBLE|BS_OWNERDRAW,
        CRect(423,275,0,0), this, B_QUIT);
        Button_Exit->LoadBitmaps(IDB_QUIT);
        Button_Exit->SizeToContent();
                        //CBB_QUIT = new CBitmapButton;
                    MainOutput = (CEdit *) GetDlgItem(CE_MainOutput);
                        Input = (CEdit *) GetDlgItem(CE_INPUT);
            Button_Begin_T = (CButton *)
GetDlgItem(B_BEGIN_TOURNAMENT);
            Button_End_T = (CButton *) GetDlgItem(B_END_TOURNAMENT);
            Button_Help = (CButton *) GetDlgItem(B_HELP);
                        Button_Enter = (CButton *) GetDlgItem(B_ENTER);
            //Pet1 Interface Objects
                        Pet1_HealthMeter = (CSliderCtrl *)
GetDlgItem(SC_PET1);
                        Pet1_HealthBar = (CProgressCtrl *)
GetDlgItem(PB_Pet1Life);
            Pet1_Status = (CEdit *) GetDlgItem(CE_PET1_STATUS);
                        Pet1View = (CStatic *) GetDlgItem(PC_PET_1);
            Pet1_Box_Name = (CStatic *) GetDlgItem(CS_PET1_BOX_NAME);
                        Pet1_Box_Species = (CStatic *)
GetDlgItem(CS_PET1_BOX_SPECIES);
            Pet1_Box_Health = (CStatic *)
GetDlgItem(CS_PET1_BOX_HEALTH);
                        Pet1_Box_Speed = (CStatic *)
GetDlgItem(CS_PET1_BOX_SPEED);
```

```
                        Pet1_Box_Strength = (CStatic *)
GetDlgItem(CS_PET1_BOX_STRENGTH);
                        Pet1_Box_Attack = (CStatic *)
GetDlgItem(CS_PET1_BOX_ATK);
                        Pet1_Box_Defense = (CStatic *)
GetDlgItem(CS_PET1_BOX_DEF);
                        Pet1_Box_SpecialDefense = (CStatic *)
GetDlgItem(CS_PET1_BOX_SPCDEF);
                        Pet1_Box_SpecialAttack = (CStatic *)
GetDlgItem(CS_PET1_BOX_SPCATK);
                        Pet1_Box_SpecAbilEnergy = (CStatic *)
GetDlgItem(CS_PET1_BOX_SPCABILENERGY);
                        Pet1_Button_LOAD = (CButton *)
GetDlgItem(B_PET1_LOAD);
                        Pet1_Button_SAVE = (CButton *)
GetDlgItem(B_PET1_SAVE);
                        Pet1_Button_NEW = (CButton *)
GetDlgItem(B_PET1_NEW);
                        Pet1_Button_TRAIN = (CButton *)
GetDlgItem(B_PET1_TRAIN);
                        Pet1_Button_FEED = (CButton *)
GetDlgItem(B_PET1_FEED);
                        Pet1_Button_PLAY = (CButton *)
GetDlgItem(B_PET1_PLAY);
                        Pet1_Button_TALK = (CButton *)
GetDlgItem(B_PET1_TALK);
                        Pet1_Button_SLEEP = (CButton *)
GetDlgItem(B_PET1_SLEEP);
                        Pet1_Button_KICK = (CButton *)
GetDlgItem(B_PET1_KICK);
                        Pet1_Button_ATTACK = (CButton *)
GetDlgItem(B_PET1_ATTACK);
                        Pet1_Button_DEFEND = (CButton *)
GetDlgItem(B_PET1_DEFEND);
                        Pet1_Button_SPECATTACK = (CButton *)
GetDlgItem(B_PET1_SPEC_ATTACK);
                        Pet1_Button_SPECDEFEND = (CButton *)
GetDlgItem(B_PET1_SPEC_DEFEND);
             //Pet2 Interface Objects
                        Pet2_HealthMeter = (CSliderCtrl *)
GetDlgItem(SC_PET2);
                        Pet2_HealthBar = (CProgressCtrl *)
GetDlgItem(PB_Pet2Life);
             Pet2_Status = (CEdit *) GetDlgItem(CE_PET2_STATUS);
                        Pet2View = (CStatic *) GetDlgItem(PC_PET_2);
             Pet2_Box_Name = (CStatic *) GetDlgItem(CS_PET2_BOX_NAME);
                        Pet2_Box_Species = (CStatic *)
GetDlgItem(CS_PET2_BOX_SPECIES);
             Pet2_Box_Health = (CStatic *)
GetDlgItem(CS_PET2_BOX_HEALTH);
                        Pet2_Box_Speed = (CStatic *)
GetDlgItem(CS_PET2_BOX_SPEED);
                        Pet2_Box_Strength = (CStatic *)
GetDlgItem(CS_PET2_BOX_STRENGTH);
```

```
                        Pet2_Box_Attack = (CStatic *)
GetDlgItem(CS_PET2_BOX_ATK);
                        Pet2_Box_Defense = (CStatic *)
GetDlgItem(CS_PET2_BOX_DEF);
                        Pet2_Box_SpecialDefense = (CStatic *)
GetDlgItem(CS_PET2_BOX_SPCDEF);
                        Pet2_Box_SpecialAttack = (CStatic *)
GetDlgItem(CS_PET2_BOX_SPCATK);
                        Pet2_Box_SpecAbilEnergy = (CStatic *)
GetDlgItem(CS_PET2_BOX_SPCABILENERGY);
                        Pet2_Button_LOAD = (CButton *)
GetDlgItem(B_PET2_LOAD);
                        Pet2_Button_SAVE = (CButton *)
GetDlgItem(B_PET2_SAVE);
                        Pet2_Button_NEW = (CButton *)
GetDlgItem(B_PET2_NEW);
                        Pet2_Button_TRAIN = (CButton *)
GetDlgItem(B_PET2_TRAIN);
                        Pet2_Button_FEED = (CButton *)
GetDlgItem(B_PET2_FEED);
                        Pet2_Button_PLAY = (CButton *)
GetDlgItem(B_PET2_PLAY);
                        Pet2_Button_TALK = (CButton *)
GetDlgItem(B_PET2_TALK);
                        Pet2_Button_SLEEP = (CButton *)
GetDlgItem(B_PET2_SLEEP);
                        Pet2_Button_KICK = (CButton *)
GetDlgItem(B_PET2_KICK);
                        Pet2_Button_ATTACK = (CButton *)
GetDlgItem(B_PET2_ATTACK);
                        Pet2_Button_DEFEND = (CButton *)
GetDlgItem(B_PET2_DEFEND);
                        Pet2_Button_SPECATTACK = (CButton *)
GetDlgItem(B_PET2_SPEC_ATTACK);
                        Pet2_Button_SPECDEFEND = (CButton *)
GetDlgItem(B_PET2_SPEC_DEFEND);
              //Setup Fonts
                        PetFont1.CreatePointFont(75,L"Arial");
                        PetFont2.CreatePointFont(100,L"Comic Sans MS");
                        CFont * BoxFont = &PetFont1;
                        CFont * StatusFont = &PetFont2;
                        MainOutput->SetFont(StatusFont);
              Pet1_Status->SetFont(StatusFont);
              Pet1_Box_Name->SetFont(BoxFont);
                        Pet1_Box_Species->SetFont(BoxFont);
              Pet1_Box_Health->SetFont(BoxFont);
                        Pet1_Box_Speed->SetFont(BoxFont);
                        Pet1_Box_Strength->SetFont(BoxFont);
                        Pet1_Box_Attack->SetFont(BoxFont);
                        Pet1_Box_Defense->SetFont(BoxFont);
                        Pet1_Box_SpecialDefense->SetFont(BoxFont);
                        Pet1_Box_SpecialAttack->SetFont(BoxFont);
                        Pet1_Box_SpecAbilEnergy->SetFont(BoxFont);
                        Pet2_Status->SetFont(StatusFont);
```

```
               Pet2_Box_Name->SetFont(BoxFont);
                         Pet2_Box_Species->SetFont(BoxFont);
               Pet2_Box_Health->SetFont(BoxFont);
                         Pet2_Box_Speed->SetFont(BoxFont);
                         Pet2_Box_Strength->SetFont(BoxFont);
                         Pet2_Box_Attack->SetFont(BoxFont);
                         Pet2_Box_Defense->SetFont(BoxFont);
                         Pet2_Box_SpecialDefense->SetFont(BoxFont);
                         Pet2_Box_SpecialAttack->SetFont(BoxFont);
                         Pet2_Box_SpecAbilEnergy->SetFont(BoxFont);

}
//---------------------------------------------------------------------
//Add Code to color interface with the global CBrush object
//Remember to add "ON_WM_CTLCOLOR()" to the MessageMap below.
HBRUSH MegaPet_FORM::OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
       HBRUSH The_Active_Paint_Brush = CDialog::OnCtlColor(pDC, pWnd,
nCtlColor);
           //Color the main CDialog window background
       if(nCtlColor == CTLCOLOR_DLG)
       {
            pDC->SetTextColor(RGB(0, 0, 0));
            The_Active_Paint_Brush = PetBrush_WHITE;
               return The_Active_Paint_Brush;
       }
       //Color the main output CEdit box
       else if(pWnd->GetDlgCtrlID() == CE_MainOutput)
       {
            pDC->SetTextColor(RGB(0, 0, 255));
            pDC->SetBkMode(TRANSPARENT);
            The_Active_Paint_Brush = PetBrush_WHITE;
               return The_Active_Paint_Brush;
       }
       //Color the Pet1 status CEdit box
       else if(pWnd->GetDlgCtrlID() == CE_PET1_STATUS)
       {
            pDC->SetTextColor(RGB(255, 255, 255));
            pDC->SetBkMode(TRANSPARENT);
            The_Active_Paint_Brush = PetBrush_BLACK;
               return The_Active_Paint_Brush;
       }
       //Color the Pet2 status CEdit box
       else if(pWnd->GetDlgCtrlID() == CE_PET2_STATUS)
       {
            pDC->SetTextColor(RGB(255, 255, 255));
            pDC->SetBkMode(TRANSPARENT);
            The_Active_Paint_Brush = PetBrush_BLACK;
               return The_Active_Paint_Brush;
       }
       //Color the input CEdit box
       else if(pWnd->GetDlgCtrlID() == CE_INPUT)
       {
            pDC->SetTextColor(RGB(255, 255, 255));
```

```
            pDC->SetBkMode(TRANSPARENT);
            The_Active_Paint_Brush = PetBrush_BLACK;
                return The_Active_Paint_Brush;
        }
            else if(pWnd->GetDlgCtrlID() == CS_PET1_BOX_NAME ||
                  pWnd->GetDlgCtrlID() == CS_PET1_BOX_SPECIES ||
                  pWnd->GetDlgCtrlID() == CS_PET1_BOX_HEALTH ||
                  pWnd->GetDlgCtrlID() == CS_PET1_BOX_SPEED ||
                  pWnd->GetDlgCtrlID() == CS_PET1_BOX_STRENGTH ||
                  pWnd->GetDlgCtrlID() == CS_PET1_BOX_ATK ||
                  pWnd->GetDlgCtrlID() == CS_PET1_BOX_DEF ||
                  pWnd->GetDlgCtrlID() == CS_PET1_BOX_SPCATK ||
           pWnd->GetDlgCtrlID() == CS_PET1_BOX_SPCDEF ||
                  pWnd->GetDlgCtrlID() == CS_PET1_BOX_SPCABILENERGY ||
           pWnd->GetDlgCtrlID() == CS_PET2_BOX_NAME ||
                  pWnd->GetDlgCtrlID() == CS_PET2_BOX_SPECIES ||
                  pWnd->GetDlgCtrlID() == CS_PET2_BOX_HEALTH ||
                  pWnd->GetDlgCtrlID() == CS_PET2_BOX_SPEED ||
                  pWnd->GetDlgCtrlID() == CS_PET2_BOX_STRENGTH ||
                  pWnd->GetDlgCtrlID() == CS_PET2_BOX_ATK ||
                  pWnd->GetDlgCtrlID() == CS_PET2_BOX_DEF ||
                  pWnd->GetDlgCtrlID() == CS_PET2_BOX_SPCATK ||
           pWnd->GetDlgCtrlID() == CS_PET2_BOX_SPCDEF ||
                  pWnd->GetDlgCtrlID() == CS_PET2_BOX_SPCABILENERGY)
            {
                  pDC->SetTextColor(RGB(255, 0, 0));
                      pDC->SetBkMode(TRANSPARENT);
              The_Active_Paint_Brush = PetBrush_WHITE;
                  return The_Active_Paint_Brush;
            }
        //Color all other generic CStatic text boxes
        else if(nCtlColor == CTLCOLOR_STATIC)
        {
             pDC->SetTextColor(RGB(0, 0, 0));
                    pDC->SetBkMode(TRANSPARENT);
             The_Active_Paint_Brush = PetBrush_WHITE;
                 return The_Active_Paint_Brush;
        }
        //Color all other generic buttons
        else if(nCtlColor == CTLCOLOR_BTN)
        {
             pDC->SetTextColor(RGB(255, 255, 255));
                    //pDC->SetBkMode(TRANSPARENT);
             The_Active_Paint_Brush = PetBrush_RED;
                 return The_Active_Paint_Brush;
        }
            else
            {
             return The_Active_Paint_Brush;
            }
/*
Note: To color groups of objects by default use:
        if(nCtlColor ==) and one of constants like:
        CTLCOLOR_EDIT
```

```
        CTLCOLOR_STATIC
        CTLCOLOR_DLG
                CTLCOLOR_BTN
                CTLCOLOR_LISTBOX
                CTLCOLOR_MSGBOX
                CTLCOLOR_SCROLLBAR
                CTLCOLOR_STATIC
*/
}
//---------------------------------------------------------------------
      afx_msg void START()
      {
              TEMP = "\r\n  Welcome to MegaPet 2.0!\r\n\r\n";
                        TEMP = TEMP +"\r\n  To begin, you must create "
                                        + "\r\n  Pet objects.\r\n"
                       + "\r\n  Begin by creating a Pet1"
                       + "\r\n  object by clicking the"
                                        + "\r\n  \"NEW\" button for
Pet1.";

                   MainOutput->SetWindowText(TEMP);
                          Pet1_Button_NEW->EnableWindow(true);
                          Button_Start->EnableWindow(false);
                          Button_Start->ShowWindow(false);
      }
//---------------------------------------------------------------------
//Message Handlers
//---------------------------------------------------------------------
         afx_msg void QUIT() { EndDialog(IDOK); }
//---------------------------------------------------------------------
          afx_msg void ENTER()
          {
            if(NeedToCreatePets)
                        { CreatePet(); }
                        else
                        {
                      TEMP = "\r\n  Pets have now been created.";
                                  TEMP = TEMP + "\r\n\r\n  You may click
\"BEGIN T\""
                                          + "\r\n  to begin a TOURNAMENT
with"
                                          + "\r\n  your new PETs or save
them.";
                                  MainOutput->SetWindowText(TEMP);
                                  Button_Begin_T->EnableWindow(true);
                                  Pet1_Button_SAVE->EnableWindow(true);
                                  Pet2_Button_SAVE->EnableWindow(true);
                        }
          }
//---------------------------------------------------------------------
          afx_msg void NewPet1()
          {
                  CreatePet();
                          Pet1_Button_NEW->EnableWindow(false);
```

```
          }
//-------------------------------------------------------------------
          afx_msg void NewPet2()
          {
                  CreatePet();
                         Pet2_Button_NEW->EnableWindow(false);
          }
//-------------------------------------------------------------------
          afx_msg void BeginTournament()
          {
                  Button_Begin_T->EnableWindow(false);
                         Button_Enter->EnableWindow(false);
                         Combat();
          }
//-------------------------------------------------------------------
      afx_msg void Pet1Save()
          {
                      SavePet(PET1);
          }
//-------------------------------------------------------------------
      afx_msg void Pet2Save()
          {
                      SavePet(PET2);
          }
//-------------------------------------------------------------------
      afx_msg void Pet1Load()
          {
                  TEMP = "";
                  if(LoadToggle)
                  {
                     LoadPet(1);
                         Input->SetWindowText(L"");
                     Button_Begin_T->EnableWindow(true);
                         LoadToggle = false;
                  }
                  else
                  {
                     TEMP = TEMP + "\r\n Enter the name of the Pet"
                                   + "\r\n you desire to load into the"
                                       + "\r\n INPUT field below and
click"
                                       + "\r\n the \"LOAD\" button for
PET1"
                                       + "\r\n to continue.";
                         MainOutput->SetWindowText(TEMP);
                         Input->SetFocus();
              LoadToggle = true;
                  }
          }
//-------------------------------------------------------------------
      afx_msg void Pet2Load()
          {
                  TEMP = "";
                  if(LoadToggle)
```

```
                  {
                     LoadPet(2);
                         Input->SetWindowText(L"");
                     Button_Begin_T->EnableWindow(true);
                         LoadToggle = false;
                  }
                  else
                  {
                     TEMP = TEMP + "\r\n Enter the name of the Pet"
                                   + "\r\n you desire to load into the"
                                       + "\r\n INPUT field below and
click"
                                       + "\r\n the \"LOAD\" button for
PET2"
                                       + "\r\n to continue.";
                         MainOutput->SetWindowText(TEMP);
                         Input->SetFocus();
             LoadToggle = true;
                  }
          }
//-------------------------------------------------------------------
          DECLARE_MESSAGE_MAP()
};
//-------------------------------------------------------------------
BEGIN_MESSAGE_MAP(MegaPet_FORM, CDialog)
          //ON_WM_TIMER()
      ON_WM_CTLCOLOR()
          ON_COMMAND(B_QUIT, QUIT)
      ON_COMMAND(B_START, START)
          ON_COMMAND(B_ENTER, ENTER)
      ON_COMMAND(B_PET1_NEW, NewPet1)
          ON_COMMAND(B_PET2_NEW, NewPet2)
          ON_COMMAND(B_BEGIN_TOURNAMENT,BeginTournament)
      ON_COMMAND(B_PET1_SAVE,Pet1Save)
      ON_COMMAND(B_PET2_SAVE,Pet2Save)
          ON_COMMAND(B_PET1_LOAD,Pet1Load)
      ON_COMMAND(B_PET2_LOAD,Pet2Load)
END_MESSAGE_MAP()
//*******************************************************************
****
class MegaPet_Window : public CWinApp
{
public:
MegaPet_Window() {  }
public:
virtual BOOL InitInstance()
{
   CWinApp::InitInstance();
   SetRegistryKey(_T("MegaPet2"));
   MegaPet_FORM MegaPetDialog;
   m_pMainWnd = &MegaPetDialog;
   INT_PTR nResponse = MegaPetDialog.DoModal();
   return FALSE;
} //close function
```

```
};
//*************************************************************************
****
MegaPet_Window LetErrRip;  //This starts the ball rolling
```

File 2: Globals.h

```
//Globals for MegaPet 1.5
#include <string>
#include <windows.h>
#include <strstream>
#include <mmsystem.h>
#include <fstream>
#include "MFCSleep.h"
using namespace std;
//Create MFC streaming text like console's cout <<
//-------------------------------------------------------------------
-----------
CString TEMP;
CString Pet1TEMP;
CString Pet2TEMP;
//-------------------------------------------------------------------
-----------
//Colors and Fonts
CBrush PetBrush_BLACK;
CBrush PetBrush_WHITE;
CBrush PetBrush_RED;
CBrush PetBrush_BLUE;
CFont PetFont1;
CFont PetFont2;
//-------------------------------------------------------------------
-----------
//CBitmapButtons
CBitmapButton * Button_Start;
CBitmapButton * Button_Exit;
//-------------------------------------------------------------------
-----------
//Generic Interface Pointers
CEdit * MainOutput;
CEdit * Input;
CButton * Button_Begin_T;
CButton * Button_End_T;
CButton * Button_Help;
CButton * Button_Enter;
//Pet1 Interface Pointers
CSliderCtrl * Pet1_HealthMeter;
CProgressCtrl * Pet1_HealthBar;
CEdit * Pet1_Status;
CStatic * Pet1View;
CStatic * Pet1_Box_Name;
```

```
CStatic * Pet1_Box_Species;
CStatic * Pet1_Box_Health;
CStatic * Pet1_Box_Speed;
CStatic * Pet1_Box_Strength;
CStatic * Pet1_Box_Attack;
CStatic * Pet1_Box_Defense;
CStatic * Pet1_Box_SpecialDefense;
CStatic * Pet1_Box_SpecialAttack;
CStatic * Pet1_Box_SpecAbilEnergy;
CButton * Pet1_Button_LOAD;
CButton * Pet1_Button_SAVE;
CButton * Pet1_Button_NEW;
CButton * Pet1_Button_TRAIN;
CButton * Pet1_Button_FEED;
CButton * Pet1_Button_PLAY;
CButton * Pet1_Button_TALK;
CButton * Pet1_Button_SLEEP;
CButton * Pet1_Button_KICK;
CButton * Pet1_Button_ATTACK;
CButton * Pet1_Button_DEFEND;
CButton * Pet1_Button_SPECATTACK;
CButton * Pet1_Button_SPECDEFEND;
//Pet2 Interface Pointers
CSliderCtrl * Pet2_HealthMeter;
CProgressCtrl * Pet2_HealthBar;
CEdit * Pet2_Status;
CStatic * Pet2View;
CStatic * Pet2_Box_Name;
CStatic * Pet2_Box_Species;
CStatic * Pet2_Box_Health;
CStatic * Pet2_Box_Speed;
CStatic * Pet2_Box_Strength;
CStatic * Pet2_Box_Attack;
CStatic * Pet2_Box_Defense;
CStatic * Pet2_Box_SpecialDefense;
CStatic * Pet2_Box_SpecialAttack;
CStatic * Pet2_Box_SpecAbilEnergy;
CButton * Pet2_Button_LOAD;
CButton * Pet2_Button_SAVE;
CButton * Pet2_Button_NEW;
CButton * Pet2_Button_TRAIN;
CButton * Pet2_Button_FEED;
CButton * Pet2_Button_PLAY;
CButton * Pet2_Button_TALK;
CButton * Pet2_Button_SLEEP;
CButton * Pet2_Button_KICK;
CButton * Pet2_Button_ATTACK;
CButton * Pet2_Button_DEFEND;
CButton * Pet2_Button_SPECATTACK;
CButton * Pet2_Button_SPECDEFEND;
//-----------------------------------------------------------------------
-----------
//Class Prototypes
class Pet;
```

```
class Animal;
class Plant;
class Mammal;
class Reptile;
class Amphibian;
class HYBRID;
class SNAKE;
class RABBIT;
class SNABBIT;
class LIGER;
class FRANTHER;
class JONKEY;
class CROG;
class SQUEEL;
//-------------------------------------------------------------------
-----------
//Function Prototypes
CString CONVERT(string X);
char * CS_To_CharArray(CString CS);
void CS_To_CharArray(CString CS, char * SimpleString);
int CS_To_Number(CString CS);
CString Number_To_CS(int NUM);
void Initialize_Interface();
void CreatePet();
void SavePet(Pet * PLAYER);
void LoadPet(int ThePet);
void Combat();
//Universal Globals
Pet * PET1;
Pet * PET2;
bool NeedToCreatePets;
bool LoadToggle;
int CreatePetSequence;
//Directories for Media Files
CString PetImageDirectory;
CString PetAudioMusicDirectory;
CString PetAudioSoundEffectsDirectory;
int Max_Damage;
//-------------------------------------------------------------------
/*
Notes: Using CBitmap buttons.
1. Create or find the bitmaps for your buttons.
2. Add them to the resource ".rc" file using resource tab and "add
resource".
3. Make your CBitmapButton pointers.
4. Add CButtons to the resource file.
5. Set the type of the button to "bitmap".
4. Set the pointers up on the interface sith GetDlgItem().
5. Build instances of the buttons on the heap with "new".
6. Call the following functions (x,y,width,height):
Button_Start->Create(NULL, WS_CHILD|WS_VISIBLE|BS_OWNERDRAW,
                     CRect(10,10,10,10), this,
IDC_ResourceNameOfButton);
Button_Start->LoadBitmaps(IDB_BitmapName);
```

```
Button_Start->SizeToContent();
7. Add the "IDC_ResourceNameOfButton" as an entry to the resource.h
file
   unless Visual Studio added for you in the resource editor.
8. Map it in the message map tags.
9. Create a message handler function.
//-----------------------------------------------------------------
Notes: Using SOUND.
        //A Note: For sound, you must do three things:
        //1. Go to project properties, then under Linker, find Input.
        //In the box labelled Aditional Dependancies add "winmm.lib".
        //2. Include the file: #include <mmsystem.h> .
        //3. Use command:
PlaySound("west.wav",NULL,SND_FILENAME|SND_ASYNC);
        //Options: SYNC = ends with function, async = keep playing
        //SND_LOOP = loop it, must be stopped then with
"StopSound()".
        //PlaySound("west.wav",NULL,SND_FILENAME|SND_ASYNC|SND_LOOP);
//-----------------------------------------------------------------
//Example of Timer inside CDialog class
void OnTimer(UINT nIDEvent)
{
        Pet->setDays((Pet->getDays() + 1));
        Pet->setLife((Pet->getLife() - 1));
        Pet->View();
        strncpy(DISPLAY,"",2000);
        Pet->Talk();
        if(Pet->getLife()  <= 0)
    {
          KillTimer(1);
          pOutput->SetWindowText("\r\n\r\nGame Over.Your pet has died
of starvation.");
    }
}
//-----------------------------------------------------------------
*/
```

File 3: Classes.h

```
//Class Inheritance Hierarchy
//The Base Class ADT (Abstract Data Type)
//*****************************************************************
*********************
class Pet
{
      public:
          Pet()
          {
               TEMP = TEMP + "\r\n Creating a Pet object...";
                   MainOutput->SetWindowText(TEMP);
          }
          ~Pet()
```

```
          {
              TEMP = TEMP + "\r\n Destroying a Pet object...";
                        MainOutput->SetWindowText(TEMP);
          }
          //-------------------------------------------------------------
----
      void InitializePet()
          {
               PetName = "Anonymous";
                   CustomizePetSequence = 1;
                   PetInterface = 0;

                   //Set Up Media Files for Class
                   PetImageFile = PetImageDirectory + PetImageFile;
            PetAtkSoundFile = PetAudioSoundEffectsDirectory +
PetAtkSoundFile;
            PetWinSoundFile = PetAudioMusicDirectory + PetWinSoundFile;
            PetLoseSoundFile = PetAudioMusicDirectory + PetLoseSoundFile;
            PetPICTURE = (HBITMAP) LoadImage(NULL,PetImageFile,
                          IMAGE_BITMAP,110,100,LR_LOADFROMFILE);
          }
      //Functions
          //-------------------------------------------------------------
----
          void Talk() { }
      //---------------------------------------------------------------------
          void Taunt() { }
          //-------------------------------------------------------------
----
          void Train()
          { }
          //-------------------------------------------------------------
----
      void Attack(Pet * opponent)
          {
                   //Note: Not necessary to cast with srand, just to avoid
warning
                   srand((unsigned int)time(0));
                   int AdjustDamage;
                   int DAMAGE = (rand() % Max_Damage) + 1;

                   TEMP = TEMP + "\r\n\r\n " + PetName + " attacks " +
opponent->GetPetName()
                               + "\r\n for a total of " +
Number_To_CS(DAMAGE) + " points"
                                    + "\r\n of damage.";
                   MainOutput->SetWindowText(TEMP);

            PlaySound(PetAtkSoundFile,NULL,SND_FILENAME|SND_ASYNC);
                   XSleep(3000);
            if(strength > 5)
                   {
                AdjustDamage = (rand() % 5) + 1;
                            DAMAGE = DAMAGE + AdjustDamage;
```

```
                TEMP = TEMP + "\r\n\r\n Due to " + PetName + "'s
strength,"
                                    + "\r\n an additional " +
Number_To_CS(AdjustDamage) + " points"
                                          + "\r\n of damage are
generated.";
                           MainOutput->SetWindowText(TEMP);
                           XSleep(4000);
                    }
             /*
                    if(speed > 5)
                    {  }
                    */
             if(opponent->GetDefense() > 5)
                    {
                           AdjustDamage = (rand() % 3) + 1;
                           if(DAMAGE - AdjustDamage < 0)
                           {
                                  TEMP = TEMP +  "\r\n\r\n " + opponent-
>GetPetName()
                                          + "'s defensive capabilities "
                                          + "\r\n completely block all
damage!";
                                  DAMAGE = 0;
                           }
                           else
                           {
                                  TEMP = TEMP +  "\r\n\r\n " + opponent-
>GetPetName()
                                          + "'s defensive capabilities "
                                          + "\r\n deflect " +
Number_To_CS(AdjustDamage)
                                                 + " points of damage.";
                                  DAMAGE = DAMAGE - AdjustDamage;
                           }
                           MainOutput->SetWindowText(TEMP);
                           XSleep(4000);
                    }

             if(SpecialAbilityEnergy > 5)
                    {
                SpecialAbilityEnergy = SpecialAbilityEnergy - 10;
                    }
             TEMP = "";
             if(opponent->GetHealth() - DAMAGE > 0)
                    {
                       opponent->SetHealth(opponent->GetHealth() - DAMAGE);
                           TEMP = TEMP + "\r\n\r\n The END result is that "
+ PetName
                                          + " attacked \r\n " + opponent-
>GetPetName()
                                    + " for a TOTAL of " +
Number_To_CS(DAMAGE) + " points"
                                          + "\r\n of damage.";
```

```
                    }
                    else
                    {
                            opponent->SetHealth(0);
                TEMP = TEMP + "\r\n\r\n The END result is that " + PetName
                                    + " attacked \r\n " + opponent-
>GetPetName()
                                  + " and finished off what little life
\r\n it"
                                        + " had left...";
                    }
                    MainOutput->SetWindowText(TEMP);
                    MainOutput->SetScrollPos(MainOutput-
>GetLineCount(),0,true); //AUTOSCROLL it down
                    View();
                    PET1->View();
                    PET2->View();
                    XSleep(5000);
          }
          //-------------------------------------------------------------
----
          void Feed()
          {
           TEMP = "";
           if(health < 50)
           {
               TEMP = TEMP + "\r\nYou feed your pet. This brings it "
                                  + "happy contentment and adds 3 points
to"
                        + " its Life.\r\n";
               health = health + 3;
           }
           else
           {
               TEMP = TEMP + "\r\nYour feed your pet, but it is just not"
                                  + " hungry. As a result, it gets sick
and vomits"
                        + " on your new carpet. This causes you to scold"
                        + " it, and it looses 1 life point.\r\n";
                       health = health - 1;
           }
               View();

                   MainOutput->SetWindowText(TEMP);

//PlaySound("media/Feed_Sound.wav",NULL,SND_FILENAME|SND_ASYNC);
          }
          //-------------------------------------------------------------
----
      void CustomizePet()
      {
                switch(CustomizePetSequence)
                    {
                 case 1 : TEMP = TEMP + "\r\n\r\n   Enter a name for Pet1"
```

```
                                                        + "\r\n   and click
\"ENTER\" to continue.";
                                            CustomizePetSequence++;
                                            break;
                                case 2:  Input->GetWindowText(TEMP);
                                            Input->SetWindowText(L"");
                            PetName = TEMP;
                                                View();
                                                TEMP = TEMP + "\r\n\r\n
Now you must choose how to allocate"
                                                                + "\r\n    your
Pet's attributes. There is an"
                                                                    + "\r\n
element of strategy to this. You only"
                                                                    + "\r\n
have 100 points to allocate. For"
                                                                    + "\r\n
example, give your Pet more speed and"
                                                                    + "\r\n
it gets more attacks per round. The"
                                                                    + "\r\n
tradeoff is that there will be less"
                                                                    + "\r\n
available for strength and so that while"
                                                                    + "\r\n
the Pet will get more attacks it will do"
                                                                    + "\r\n
less damage per attack."
                                                                    + "\r\n
You have 100 points to allocate. How much"
                                                                    + "\r\n
will you give your Pet for speed?";
                                            CustomizePetSequence++;
                                            break;
                                default: break;
                    }

      }
          //-------------------------------------------------------------
----
          void View()
          {
           char buffer[10];

                  if(PetInterface == 1)
                  {
                          Pet1TEMP = "";
               Pet1_Box_Name->SetWindowText(PetName);
                          Pet1_Box_SpecialDefense->SetWindowText(spcdef);
               Pet1_Box_SpecialAttack->SetWindowText(spcatk);
                          Pet1_Box_Species->SetWindowText(Species);
                          _itoa_s(health, buffer, 10);
                          Pet1TEMP = buffer;
               Pet1_Box_Health->SetWindowText(Pet1TEMP);
```

```
                _itoa_s(speed, buffer, 10);
                           Pet1TEMP = buffer;
                Pet1_Box_Speed->SetWindowText(Pet1TEMP);
                _itoa_s(strength, buffer, 10);
                           Pet1TEMP = buffer;
                Pet1_Box_Strength->SetWindowText(Pet1TEMP);
                           _itoa_s(attack, buffer, 10);
                           Pet1TEMP = buffer;
                Pet1_Box_Attack->SetWindowText(Pet1TEMP);
                           _itoa_s(def, buffer, 10);
                           Pet1TEMP = buffer;
                Pet1_Box_Defense->SetWindowText(Pet1TEMP);
                           _itoa_s(SpecialAbilityEnergy, buffer, 10);
                           Pet1TEMP = buffer;
                Pet1_Box_SpecAbilEnergy->SetWindowText(Pet1TEMP);
                Pet1_HealthMeter->SetPos(health);
                           Pet1_HealthBar->SetPos(health);
                Pet1TEMP = "";
               if(health > 80 && health <= 100)
                           { Pet1TEMP = Pet1TEMP + "\r\n   " + PetName + "
is in\r\n   excellent health."; }
                      else if(health > 60 && health <= 80)
                           { Pet1TEMP = Pet1TEMP + "\r\n   " + PetName + "
is\r\n   somewhat healthy."; }
                           else if(health > 40 && health <= 60)
                           { Pet1TEMP = Pet1TEMP + "\r\n   " + PetName + "
is\r\n   in pain."; }
                      else if(health > 20 && health <= 40)
                           {
                                  Pet1TEMP = Pet1TEMP + "\r\n   " +
PetName + " is mortally\r\n   wounded"
                                                + "\r\n   and in\r\n
terrible suffering.";
                           }
                      else if(health > 0 && health <= 20)
                           {
                                  Pet1TEMP = Pet1TEMP + "\r\n   " +
PetName + " is at the"
                                                + "\r\n   brink of
death!";
                           }
                      else  { Pet1TEMP = Pet1TEMP + "\r\n   " + PetName +
" is dead!"; }
                           Pet1_Status->SetWindowText(Pet1TEMP);
                           Pet1View->SetBitmap(PetPICTURE);
                   }
                   if(PetInterface == 2)
                   {
                Pet2TEMP = "";
                           Pet2_Box_Name->SetWindowText(PetName);
                           Pet2_Box_SpecialDefense->SetWindowText(spcdef);
                Pet2_Box_SpecialAttack->SetWindowText(spcatk);
                           Pet2_Box_Species->SetWindowText(Species);
                           _itoa_s(health, buffer, 10);
```

```
                        Pet2TEMP = buffer;
                Pet2_Box_Health->SetWindowText(Pet2TEMP);
                _itoa_s(speed, buffer, 10);
                        Pet2TEMP = buffer;
                Pet2_Box_Speed->SetWindowText(Pet2TEMP);
                _itoa_s(strength, buffer, 10);
                        Pet2TEMP = buffer;
                Pet2_Box_Strength->SetWindowText(Pet2TEMP);
                        _itoa_s(attack, buffer, 10);
                        Pet2TEMP = buffer;
                Pet2_Box_Attack->SetWindowText(Pet2TEMP);
                        _itoa_s(def, buffer, 10);
                        Pet2TEMP = buffer;
                Pet2_Box_Defense->SetWindowText(Pet2TEMP);
                        _itoa_s(SpecialAbilityEnergy, buffer, 10);
                        Pet2TEMP = buffer;
                Pet2_Box_SpecAbilEnergy->SetWindowText(Pet2TEMP);
                Pet2_HealthMeter->SetPos(health);
                        Pet2_HealthBar->SetPos(health);
                Pet2TEMP = "";
                if(health > 80 && health <= 100)
                        { Pet2TEMP = Pet2TEMP + "\r\n   " + PetName + "
is in\r\n   excellent health."; }
                    else if(health > 60 && health <= 80)
                        { Pet2TEMP = Pet2TEMP + "\r\n   " + PetName + "
is\r\n   somewhat healthy."; }
                        else if(health > 40 && health <= 60)
                        { Pet2TEMP = Pet2TEMP + "\r\n   " + PetName + "
is\r\n   in pain."; }
                    else if(health > 20 && health <= 40)
                        {
                                Pet2TEMP = Pet2TEMP + "\r\n   " +
PetName + " is mortally\r\n   wounded"
                                        + "\r\n   and in\r\n
terrible suffering.";
                        }
                    else if(health > 0 && health <= 20)
                        {
                                Pet2TEMP = Pet2TEMP + "\r\n   " +
PetName + " is at the"
                                        + "\r\n   brink of
death!";
                        }
                    else  { Pet2TEMP = Pet2TEMP + "\r\n   " + PetName +
" is dead!"; }
                        Pet2_Status->SetWindowText(Pet2TEMP);
                        Pet2View->SetBitmap(PetPICTURE);
                }

}//close View function
     //-------------------------------------------------------------------
          //Accessor Methods
          void SetHealth(int X) { health = X; }
          int GetHealth() { return health; }
```

```
        void SetSpeed(int X) { speed = X; }
        int GetSpeed() { return speed; }
        void SetStrength(int X) { strength = X; }
        int GetStrength() { return strength; }
        void SetAttack(int X) { attack = X; }
        int GetAttack() { return attack; }
        void SetDefense(int X) { def = X; }
        int GetDefense() { return def; }
        void SetSpecialAbilityEnergy(int X) { SpecialAbilityEnergy = X;
}
        int GetSpecialAbilityEnergy() { return SpecialAbilityEnergy; }
        void SetPetName(CString X) { PetName = X; }
        CString GetPetName() { return PetName; }
        void SetSpecies(CString X) { Species = X; }
        CString GetSpecies() { return Species; }
        void SetSpcAtk(CString X) { spcatk = X; }
        CString GetSpcAtk() { return spcatk; }
        void SetSpcDef(CString X) { spcdef = X; }
        CString GetSpcDef() { return spcdef; }
        void SetPetImageFile(CString X) { PetImageFile = X; }
        CString GetPetImageFile() { return PetImageFile; }
    void SetPetAtkSoundFile(CString X) { PetAtkSoundFile = X; }
        CString GetPetAtkSoundFile() { return PetAtkSoundFile; }
    void SetPetWinSoundFile(CString X) { PetWinSoundFile = X; }
        CString GetPetWinSoundFile() { return PetWinSoundFile; }
    void SetPetLoseSoundFile(CString X) { PetLoseSoundFile = X; }
        CString GetPetLoseSoundFile() { return PetLoseSoundFile; }
    void SetPetPICTURE(HBITMAP X) { PetPICTURE = X; }
        HBITMAP GetPetPICTURE() { return PetPICTURE; }
        void SetPetInterface(int X) { PetInterface = X; }
        int GetPetInterface() { return PetInterface; }
        void SetCustomizePetSequence(int X) { CustomizePetSequence = X;
}
        int GetCustomizePetSequence() { return CustomizePetSequence; }
    //-------------------------------------------------------------------
    //Private Data
    private:
        int health;
        int speed;
        int strength;
        int attack;
        int def;
        int SpecialAbilityEnergy;
        CString PetName;
        CString Species;
        CString spcatk;
    CString spcdef;
        CString PetImageFile;
        CString PetAtkSoundFile;
    CString PetWinSoundFile;
    CString PetLoseSoundFile;
        HBITMAP PetPICTURE;
        CStatic * PetView;
    int PetInterface;
```

```
      int CustomizePetSequence;
};
//************************************************************************
**********************
class Animal : public Pet
{
      public:
          Animal()
          {
               TEMP = TEMP + "\r\n Creating an Animal object...";

                          MainOutput->SetWindowText(TEMP);
          }
          ~Animal()
          {
               TEMP = TEMP + "\r\n Destroying an Animal object...";

                          MainOutput->SetWindowText(TEMP);
          }
       //Functions

          //Accessor Methods
     //Private Data
     private:
};
//************************************************************************
**********************
class Plant : public Pet
{
      public:
          Plant()
          {
               TEMP = TEMP + "\r\n Creating a Plant object...";

                          MainOutput->SetWindowText(TEMP);
          }
          ~Plant()
          {
               TEMP = TEMP + "\r\n Destroying a Plant object...";

                          MainOutput->SetWindowText(TEMP);
          }
       //Functions

          //Accessor Methods
     //Private Data
     private:
};
//************************************************************************
**********************
class Mammal : public Animal
{
      public:
          Mammal()
```

```
          {
               TEMP = TEMP + "\r\n Creating a Mammal object...";

                         MainOutput->SetWindowText(TEMP);
          }
          ~Mammal()
          {
               TEMP = TEMP + "\r\n Destroying a Mammal object...";

                         MainOutput->SetWindowText(TEMP);
          }
      //Functions

          //Accessor Methods
     //Private Data
     private:
};
//*************************************************************************
**********************
class Reptile : public Animal
{
      public:
          Reptile()
          {
               TEMP = TEMP + "\r\n Creating a Reptile object...";

                         MainOutput->SetWindowText(TEMP);
          }
          ~Reptile()
          {
               TEMP = TEMP + "\r\n Destroying a Reptile object...";

                         MainOutput->SetWindowText(TEMP);
          }
      //Functions

          //Accessor Methods
     //Private Data
     private:
};
//*************************************************************************
**********************
class Amphibian : public Animal
{
      public:
          Amphibian()
          {
               TEMP = TEMP + "\r\n Creating an Amphibian object...";

                         MainOutput->SetWindowText(TEMP);
          }
          ~Amphibian()
          {
               TEMP = TEMP + "\r\n Destroying an Amphibian object...";
```

```
                        MainOutput->SetWindowText(TEMP);
          }
      //Functions

          //Accessor Methods
     //Private Data
     private:
};
//*************************************************************************
**********************
class HYBRID : public Animal
{
      public:
          HYBRID()
          {
               TEMP = TEMP + "\r\n Creating a HYBRID object...";

                        MainOutput->SetWindowText(TEMP);
          }
          ~HYBRID()
          {
               TEMP = TEMP + "\r\n Destroying a HYBRID object...";

                        MainOutput->SetWindowText(TEMP);
          }
      //Functions

          //Accessor Methods

     //Private Data
     private:
     string HybridAbility;
};
//*************************************************************************
**********************
class SNAKE : public Reptile
{
      public:
          SNAKE()
          {
               TEMP = TEMP + "\r\n Creating a SNAKE.";

                        MainOutput->SetWindowText(TEMP);
                        InitializeSNAKE();
          }
          ~SNAKE()
          {
               TEMP = TEMP + "\r\n Destroying a SNAKE.";

                        MainOutput->SetWindowText(TEMP);
          }
      //Functions
          void InitializeSNAKE()
```

```
          {
               SetSpecies("Snake");
                   SetPetImageFile("Snake1.bmp");
                   SetHealth(100);
               SetSpeed(2);
               SetStrength(8);
               SetAttack(5);
               SetDefense(5);
               SetSpecialAbilityEnergy(100);
               SetSpcAtk("Venom");
            SetSpcDef("Scales");
                   InitializePet();
          }
          //Accessor Methods
     //Private Data
     private:
         int Venom;
};
//*************************************************************************
**********************
class RABBIT : public Mammal
{
      public:
          RABBIT()
          {
               TEMP = TEMP + "\r\n Creating a RABBIT.";

                           MainOutput->SetWindowText(TEMP);
                           InitializeRABBIT();
          }
          ~RABBIT()
          {
               TEMP = TEMP + "\r\n Destroying a RABBIT.";

                           MainOutput->SetWindowText(TEMP);
          }
      //Functions
          void InitializeRABBIT()
          {
               SetSpecies("Rabbit");
                   SetPetImageFile("Rabbit1.bmp");
                   SetHealth(100);
               SetSpeed(8);
               SetStrength(2);
               SetAttack(2);
               SetDefense(8);
               SetSpecialAbilityEnergy(100);
               SetSpcAtk("Thump");
            SetSpcDef("Jump");
                   InitializePet();
          }
          //Accessor Methods
     //Private Data
     private:
```

```
};
//************************************************************************
**********************
class SNABBIT : public HYBRID
{
      public:
          SNABBIT()
          {
               TEMP = TEMP + "\r\n Creating a SNABBIT (Snake + Rabbit).";

                          MainOutput->SetWindowText(TEMP);
                          InitializeSNABBIT();
          }
          ~SNABBIT()
          {
               TEMP = TEMP + "\r\n Destroying a SNABBIT.";

                          MainOutput->SetWindowText(TEMP);
          }
       //Functions
          void InitializeSNABBIT()
          {
                SetSpecies("Snabbit");
                    SetPetImageFile("Snabbit1.bmp");
                    SetPetAtkSoundFile("Grind.wav");
            SetPetWinSoundFile("SucksToBeYou.wav");
            SetPetLoseSoundFile("Jeopardy.wav");
                    SetHealth(100);
                SetSpeed(7);
                SetStrength(3);
                SetAttack(5);
                SetDefense(5);
                SetSpecialAbilityEnergy(100);
                SetSpcAtk("Venom");
             SetSpcDef("Thump Hop");
                    InitializePet();
          }
          //Accessor Methods
     //Private Data
     private:
};
//************************************************************************
**********************
class LIGER : public HYBRID
{
       public:
          LIGER()
          {
               TEMP = TEMP + "\r\n Creating a LIGER (Lion + Tiger).";

                          MainOutput->SetWindowText(TEMP);
                          InitializeLIGER();
          }
          ~LIGER()
```

```
          {
               TEMP = TEMP + "\r\n Destroying a LIGER.";

                         MainOutput->SetWindowText(TEMP);
          }
       //Functions
          void InitializeLIGER()
          {
               SetSpecies("Liger");
                   SetPetImageFile("Liger1.bmp");
                   SetPetAtkSoundFile("Roar.wav");
            SetPetWinSoundFile("BadToTheBone.wav");
            SetPetLoseSoundFile("WelcomeToTheJungle.wav");
                   SetHealth(100);
               SetSpeed(5);
               SetStrength(5);
               SetAttack(7);
               SetDefense(3);
               SetSpecialAbilityEnergy(100);
               SetSpcAtk("Scratch");
            SetSpcDef("Dodge");
                   InitializePet();
          }
          //Accessor Methods
      //Private Data
      private:
};
//*************************************************************************
**********************
class FRANTHER : public HYBRID
{
       public:
          FRANTHER()
          {
               TEMP = TEMP + "\r\n Creating a FRANTHER (Frog + Panther).";

                         MainOutput->SetWindowText(TEMP);
               InitializeFRANTHER();
          }
          ~FRANTHER()
          {
               TEMP = TEMP + "\r\n Destroying a FRANTHER.";

                         MainOutput->SetWindowText(TEMP);
          }
       //Functions
          void InitializeFRANTHER()
          {
               SetSpecies("Franther");
                   SetPetName("Anonymous");
                   SetPetImageFile("Panther1.bmp");
                   SetPetAtkSoundFile("Panther.wav");
            SetPetWinSoundFile("CantTouchThis.wav");
            SetPetLoseSoundFile("Jeopardy.wav");
```

```
                    SetHealth(100);
                SetSpeed(5);
                SetStrength(5);
                SetAttack(6);
                SetDefense(4);
                SetSpecialAbilityEnergy(100);
                SetSpcAtk("Claw");
             SetSpcDef("Stealth");
                    InitializePet();
            }
            //Accessor Methods
      //Private Data
      private:
};
//*********************************************************************
*********************
class JONKEY : public HYBRID
{
       public:
           JONKEY()
           {
               TEMP = TEMP + "\r\n Creating a Jonkey (JellyFish +
Monkey).";

                           MainOutput->SetWindowText(TEMP);
               InitializeJONKEY();
           }
           ~JONKEY()
           {
               TEMP = TEMP + "\r\n Destroying a Jonkey object.";

                           MainOutput->SetWindowText(TEMP);
           }
       //Functions
       void InitializeJONKEY()
           {
                SetSpecies("Jonkey");
                    SetPetImageFile("Monkey2.bmp");
                    SetPetAtkSoundFile("Sonic.wav");
             SetPetWinSoundFile("IceIceBaby.wav");
             SetPetLoseSoundFile("IWillSurvive.wav");
                    SetHealth(100);
                SetSpeed(4);
                SetStrength(6);
                SetAttack(5);
                SetDefense(5);
                SetSpecialAbilityEnergy(100);
                SetSpcAtk("ICE");
             SetSpcDef("Jelly Swing");
                    InitializePet();
           }
           //Accessor Methods
      //Private Data
      private:
```

```
};
//************************************************************************
**********************
class CROG : public HYBRID
{
      public:
          CROG()
          {
               TEMP = TEMP + "\r\n Creating a CROG (Crocodile + Dog).";

                          MainOutput->SetWindowText(TEMP);
                          InitializeCROG();
          }
          ~CROG()
          {
               TEMP = TEMP + "\r\n Destroying a CROG.";

                          MainOutput->SetWindowText(TEMP);
          }
       //Functions

          //Accessor Methods
          void InitializeCROG()
          {
                SetSpecies("Crog");
                    SetPetImageFile("Croc1.bmp");
                    SetPetAtkSoundFile("Spook.wav");
            SetPetWinSoundFile("TheRock.wav");
            SetPetLoseSoundFile("MachoMan.wav");
                    SetHealth(100);
                SetSpeed(4);
                SetStrength(6);
                SetAttack(5);
                SetDefense(5);
                SetSpecialAbilityEnergy(100);
                SetSpcAtk("Lash");
             SetSpcDef("Armor");
                    InitializePet();
          }
     //Private Data
     private:
};
//************************************************************************
**********************
class SQUEEL : public HYBRID
{
       public:
          SQUEEL()
          {
               TEMP = TEMP + "\r\n Creating a SQUEEL (Squirrel + Eel).";

                          MainOutput->SetWindowText(TEMP);
                          InitializeSQUEEL();
          }
```

```
          ~SQUEEL()
          {
               TEMP = TEMP + "\r\n Destroying a SQUEEL.";

                          MainOutput->SetWindowText(TEMP);
          }
      //Functions
          void InitializeSQUEEL()
          {
               SetSpecies("Squeel");
                   SetPetImageFile("Squirrel1.bmp");
                   SetPetAtkSoundFile("Rip.wav");
            SetPetWinSoundFile("DiscoInferno.wav");
            SetPetLoseSoundFile("StayinAlive.wav");
                   SetHealth(100);
               SetSpeed(8);
               SetStrength(2);
               SetAttack(2);
               SetDefense(8);
               SetSpecialAbilityEnergy(100);
               SetSpcAtk("Gnash");
            SetSpcDef("Scurry");
                   InitializePet();
          }
          //Accessor Methods
     //Private Data
     private:
};
//*************************************************************************
**********************
```

File 4: Functions.h

```
//Game Functions
//-------------------------------------------------------------------
----------------------
CString CONVERT(string X)
{
        //Note: Have to cast to signed int to avoid warning when
calling length()
     char TEMP[100] = "?";
        for(int a = 0; a < (int) X.length(); a++)
        { TEMP[a] = X[a]; }
        return TEMP;
}
//-------------------------------------------------------------------
----------------------
//Note: 2 overloaded conversion functions for CString to char array
char * CS_To_CharArray(CString CS)
{
          char * INPUT = new char[100];
          for(int x = 0; x < CS.GetLength(); x++)
```

```
          { INPUT[x] = (char) CS[x]; }
          return INPUT;
}
void CS_To_CharArray(CString CS, char * SimpleString)
{
     for(int z = 0; z < CS.GetLength(); z++)
     { SimpleString[z] = (char) CS[z]; }
}
//---------------------------------------------------------------------
----------------------
int CS_To_Number(CString CS)
{
          char INPUT[10] = "";

          for(int x = 0; x < CS.GetLength(); x++)
          { INPUT[x] = (char) CS[x]; }
          int TheNumber = atoi(INPUT);
          return TheNumber;
}
//---------------------------------------------------------------------
----------------------
CString Number_To_CS(int NUM)
{
          char INPUT[10] = "";
          _itoa_s(NUM,INPUT,10,10);
      CString CS = INPUT;
          return CS;
}
//---------------------------------------------------------------------
----------------------
void Initialize_Interface()
{
         Button_Start->EnableWindow(true);
     Button_Begin_T->EnableWindow(false);
     Button_End_T->EnableWindow(false);
     //Button_Exit->EnableWindow(false);
     Button_Help->EnableWindow(false);
         Button_Enter->EnableWindow(false);
         Pet1_Button_LOAD->EnableWindow(true);
         Pet1_Button_SAVE->EnableWindow(false);
         Pet1_Button_NEW->EnableWindow(false);
         Pet1_Button_TRAIN->EnableWindow(false);
         Pet1_Button_FEED->EnableWindow(false);
         Pet1_Button_PLAY->EnableWindow(false);
         Pet1_Button_TALK->EnableWindow(false);
         Pet1_Button_SLEEP->EnableWindow(false);
         Pet1_Button_KICK->EnableWindow(false);
         Pet1_Button_ATTACK->EnableWindow(false);
         Pet1_Button_DEFEND->EnableWindow(false);
         Pet1_Button_SPECATTACK->EnableWindow(false);
         Pet1_Button_SPECDEFEND->EnableWindow(false);
         Pet2_Button_LOAD->EnableWindow(true);
         Pet2_Button_SAVE->EnableWindow(false);
         Pet2_Button_NEW->EnableWindow(false);
```

```
         Pet2_Button_TRAIN->EnableWindow(false);
         Pet2_Button_FEED->EnableWindow(false);
         Pet2_Button_PLAY->EnableWindow(false);
         Pet2_Button_TALK->EnableWindow(false);
         Pet2_Button_SLEEP->EnableWindow(false);
         Pet2_Button_KICK->EnableWindow(false);
         Pet2_Button_ATTACK->EnableWindow(false);
         Pet2_Button_DEFEND->EnableWindow(false);
         Pet2_Button_SPECATTACK->EnableWindow(false);
         Pet2_Button_SPECDEFEND->EnableWindow(false);
         CreatePetSequence = 1;
         NeedToCreatePets = true;
         LoadToggle = false;
     PetImageDirectory = "Images/";
     PetAudioMusicDirectory = "Audio/MusicClips/";
     PetAudioSoundEffectsDirectory = "Audio/SoundEffects/";
         Max_Damage = 50;
}
//------------------------------------------------------------------------
----------------------
void CreatePet()
{
         int choice;

         TEMP = "";
     switch(CreatePetSequence)
         {
                 case 1 : TEMP = TEMP + "Mad Science! Choose a type of
Transgenic"
                                  + "\r\nchimera to create for Pet1.
Options are:\r\n"
                                  + "\r\n    1. Jonkey (JellyFish +
Monkey)"
                                  + "\r\n    2. Franther (Frog +
Panther)"
                                  + "\r\n    3. Snabbit (Snake +
Rabbit)"
                   + "\r\n    4. Liger (Lion + Tiger)"
                   + "\r\n    5. Crog (Crocodile + Dog)"
                                     + "\r\n    6. Squeel (Squirrel +
Eel)"
                                  + "\r\n\r\n Enter your choice in the
Input box."
                                  + "\r\n Click ENTER to continue.";
                                  Button_Enter->EnableWindow(true);
                                  Input->SetFocus();
                   CreatePetSequence++;
                                  break;

                 case 2:  Input->GetWindowText(TEMP);
                   Input->SetWindowText(L"");
                   choice = atoi(CS_To_CharArray(TEMP));
                                  TEMP = "";
```

```
                  if(choice > 0 && choice < 7)
                                {
                     switch(choice)
                                {
                                     case 1 : PET1 = new JONKEY;
                                                                        TEMP
= TEMP + "\r\n\r\n    Jonkeys devestate with venom.";

break;
                                          case 2 : PET1 = new FRANTHER;
                                                                        TEMP
= TEMP + "\r\n\r\n    A Franther? Nice maneuverability.";

break;
                           case 3 : PET1 = new SNABBIT;
                                                                        TEMP
= TEMP + "\r\n\r\n    Snabbits = dexterity + venom attack.";

break;
                                          case 4 : PET1 = new LIGER;
                                                                        TEMP
= TEMP + "\r\n\r\n    Ligers possess speed and strength.";

break;
                           case 5 : PET1 = new CROG;
                                                                        TEMP
= TEMP + "\r\n\r\n    A Crog? Very strong bite attack!";

break;
                           case 6 : PET1 = new SQUEEL;
                                                                        TEMP
= TEMP + "\r\n\r\n    Squeels are dextrous and shocking!";

break;
                                                   default: break;
                                     }
                                CreatePetSequence++;
                                PET1->SetPetName("PET 1");
                                PET1->SetPetInterface(1);
                                PET1->View();
                                     Pet1_Button_NEW-
>EnableWindow(false);
                                     Pet1_Button_SAVE-
>EnableWindow(true);
                                     Button_Enter-
>EnableWindow(true);
                                TEMP = TEMP + "\r\n    Click
\"ENTER\" to continue.";
                           }
                           else
                           {
                                     TEMP = "";
                     TEMP = TEMP + "\r\n\r\n    Sorry, that was
simply"
```

```
                            + "\r\n    not an available option"
                                                    + "\r\n    for
you. Your only options"
                                                    + "\r\n
consist of choices 1-6."
                                                    + "\r\n\r\n
Ok?\r\n"
                                                    + "\r\n
Please click \"ENTER\" to continue.";
                    CreatePetSequence--;
                                }
                               break;
                case 3 : TEMP = "";
                            TEMP = TEMP + "\r\n It's alive! It's
alive!\r\n\r\n"
                                + " Now, give your new pet\r\n"
                                + " " + PET1->GetSpecies() + " a
name.\r\n\r\n"
                                + " Type it into the \"INPUT\" box
below.\r\n\r\n"
                  + " Click \"ENTER\" when you are ready to\r\n"
                                + " continue.";
                                Button_Enter->EnableWindow(true);
                                Input->SetFocus();
                  CreatePetSequence++;
                                break;
                case 4 : Input->GetWindowText(TEMP);
                            if(TEMP != "")
                                { PET1->SetPetName(TEMP); }
                                else
                                { PET1->SetPetName(L"Anoymous"); }
                  TEMP = "";
                  TEMP = TEMP + "\r\n Pet now named\r\n"
                                            + " " + PET1-
>GetPetName() + ".";
                                TEMP = TEMP + "\r\n\r\n    Now click
the \"NEW\" button to\r\n"
                                            + "    give
Frankensteinish life to Pet2.";
                                Pet2_Button_NEW->EnableWindow(true);
                                Button_Enter->EnableWindow(false);
                                Input->SetWindowText(L"");
                                Input->SetFocus();
                                PET1->View();
                  CreatePetSequence++;
                                break;
                case 5 : TEMP = "";
                            TEMP = TEMP + "Mad Science! Choose a
type of Transgenic"
                                + "\r\nchimera to create for Pet2.
Options are:\r\n"
                                + "\r\n    1. Jonkey (JellyFish +
Monkey)"
```

```
                                    + "\r\n    2. Franther (Frog +
Panther)"
                                    + "\r\n    3. Snabbit (Snake +
Rabbit)"
                    + "\r\n    4. Liger (Lion + Tiger)"
                    + "\r\n    5. Crog (Crocodile + Dog)"
                                    + "\r\n    6. Squeel (Squirrel +
Eel)"
                                    + "\r\n\r\n Enter your choice in the
Input box."
                                    + "\r\n Click ENTER to continue.";
                                    Button_Enter->EnableWindow(true);
                                    Input->SetWindowText(L"");
                                    Input->SetFocus();
                   CreatePetSequence++;
                                    break;

                  case 6:  Input->GetWindowText(TEMP);
                                    Input->SetWindowText(L"");
                    choice = atoi(CS_To_CharArray(TEMP));
                                    TEMP = "";

                    if(choice > 0 && choice < 7)
                                     {
                       switch(choice)
                                    {
                                        case 1 : PET2 = new JONKEY;
                                                                    TEMP =
TEMP + "\r\n\r\n    Jonkeys devestate with venom.";
                                                                    break;
                                             case 2 : PET2 = new FRANTHER;
                                                                    TEMP =
TEMP + "\r\n\r\n    A Franther? Nice maneuverability.";
                                                                    break;
                            case 3 : PET2 = new SNABBIT;
                                                                    TEMP =
TEMP + "\r\n\r\n    Snabbits = dexterity + venom attack.";
                                                                    break;
                                             case 4 : PET2 = new LIGER;
                                                                    TEMP =
TEMP + "\r\n\r\n    Ligers possess speed and strength.";
                                                                    break;
                            case 5 : PET2 = new CROG;
                                                                    TEMP =
TEMP + "\r\n\r\n    A Crog? Very strong bite attack!";
                                                                    break;
                            case 6 : PET2 = new SQUEEL;
                                                                    TEMP =
TEMP + "\r\n\r\n    Squeels are dextrous and shocking!";
                                                                    break;
                                                    default: break;
                                             }
                                             CreatePetSequence++;
                      PET2->SetPetName("PET 2");
```

```
                                        PET2->SetPetInterface(2);
                                        PET2->View();
                                        Pet2_Button_NEW-
>EnableWindow(false);
                                        Pet2_Button_SAVE-
>EnableWindow(true);
                                        Button_Enter-
>EnableWindow(true);
                                        TEMP = TEMP + "\r\n    Click
\"ENTER\" to continue.";
                                }
                                else
                                {
                                        TEMP = "";
                        TEMP = TEMP + "\r\n\r\n    Sorry, that was
simply"
                                + "\r\n    not an available option"
                                                        + "\r\n    for
you. Your only options"
                                                        + "\r\n
consist of choices 1-6."
                                                        + "\r\n\r\n
Ok?\r\n"
                                                        + "\r\n
Please click \"ENTER\" to continue.";
                        CreatePetSequence--;
                                }
                                break;
                case 7 : TEMP = "";
                                TEMP = TEMP + "\r\n It's alive! It's
alive!\r\n\r\n"
                                    + " Now, give your new pet\r\n"
                                    + " " + PET2->GetSpecies() + " a
name.\r\n\r\n"
                                    + " Type it into the \"INPUT\" box
below.\r\n\r\n"
                    + " Click \"ENTER\" when you are ready to\r\n"
                                    + " continue.";
                                    Button_Enter->EnableWindow(true);
                                    Input->SetFocus();
                    CreatePetSequence++;
                                    break;
                case 8 : Input->GetWindowText(TEMP);
                                if(TEMP != "")
                                    { PET2->SetPetName(TEMP); }
                                    else
                                    { PET2->SetPetName("Anoymous"); }
                                    TEMP = "";
                    TEMP = TEMP + "\r\n Pet now named\r\n"
                                                + " " + PET2-
>GetPetName() + ".";
                                    TEMP = TEMP + "\r\n\r\n    Now click
the \"ENTER\" button to\r\n"
```

```
                                                + "    continue with the
game.";
                                    Input->SetWindowText(L"");
                                    Input->SetFocus();
                                    PET2->View();
                    CreatePetSequence++;
                                    break;
                  case 9 : TEMP = "";
                    TEMP = TEMP + "\r\n     Your transgenic chimera Pet
objects"
                                                + "\r\n     are now
ready for action!"
                                                + "\r\n\r\n     You
MAD SCIENTIST, you..."
                                                + "\r\n\r\n     ;-) "
                             + "\r\n\r\n\r\n     Please click \"ENTER\"
to continue.";
                                    NeedToCreatePets = false;
                                    break;
                                    default : break;
        }//close outer switch

    MainOutput->SetWindowText(TEMP);
}//close Function
//---------------------------------------------------------------------
----------------------
void SavePet(Pet * PLAYER)
{
     TEMP = "";
         CString TheFile = PLAYER->GetPetName() + ".pet";
         ofstream WriteStuff;
         WriteStuff.open(TheFile);
         if(!WriteStuff)
             {
                     TEMP = TEMP + "\r\n Houston - we have a
problem!\r\n"
                                 + "\r\n The file could not be saved";
             }
         else
         {
                     //Save Pet attributes
                         char SimpleString1[50] = "";
                         CS_To_CharArray(PLAYER-
>GetPetName(),SimpleString1);
              WriteStuff << SimpleString1 << "\n";
                         char SimpleString2[50] = "";
              CS_To_CharArray(PLAYER->GetSpecies(),SimpleString2);
              WriteStuff << SimpleString2 << "\n";
                         char SimpleString3[50] = "";
                         CS_To_CharArray(PLAYER-
>GetPetImageFile(),SimpleString3);
              WriteStuff << SimpleString3 << "\n";
                         char SimpleString4[50] = "";
```

```
                        CS_To_CharArray(PLAYER-
>GetPetAtkSoundFile(),SimpleString4);
             WriteStuff << SimpleString4 << "\n";
                        char SimpleString5[50] = "";
                        CS_To_CharArray(PLAYER-
>GetPetWinSoundFile(),SimpleString5);
             WriteStuff << SimpleString5 << "\n";
                        char SimpleString6[50] = "";
             CS_To_CharArray(PLAYER-
>GetPetLoseSoundFile(),SimpleString6);
             WriteStuff << SimpleString6 << "\n";
                        char SimpleString7[50] = "";
                        CS_To_CharArray(PLAYER-
>GetSpcAtk(),SimpleString7);
             WriteStuff << SimpleString7 << "\n";
                        char SimpleString8[50] = "";
                        CS_To_CharArray(PLAYER-
>GetSpcDef(),SimpleString8);
             WriteStuff << SimpleString8 << "\n";

             WriteStuff << PLAYER->GetHealth() << "\n";
             WriteStuff << PLAYER->GetSpeed() << "\n";
                    WriteStuff << PLAYER->GetStrength() << "\n";
                    WriteStuff << PLAYER->GetAttack() << "\n";
                    WriteStuff << PLAYER->GetDefense() << "\n";
                    WriteStuff << PLAYER->GetSpecialAbilityEnergy()
<< "\n";
                WriteStuff.close();
             TEMP = TEMP + "\r\n The Pet has been successfully
saved."
                                 + "\r\n Click \"ENTER\" to
continue...";
          }//close else for a successfully opened file
     Button_Enter->EnableWindow(true);
     MainOutput->SetWindowText(TEMP);
}//close function
//---------------------------------------------------------------------
----------------------
void LoadPet(int ThePet)
{
     TEMP = "";
         string petname = "", species = "", specialattack = "",
specialdefense = "",
                   petimagefile = "", petatksoundfile = "",
petwinsoundfile = "", petlosesoundfile = "";
         int health, speed, strength, attack, defense,
specialabilityenergy;

     CString TheFile = "";
         Input->GetWindowText(TheFile);
         TheFile = TheFile + ".pet";
     ifstream ReadStuff;
     ReadStuff.open(TheFile);
```

```
     //Detect if successful or not to keep program from crashing on
failed load
     if(!ReadStuff)
     {
        TEMP = "";
               TEMP = TEMP + "\r\n\r\n Can not open or find a file
using"
                           + "\r\n the name you typed in."
                           + "\r\n Try a different name?";
     }
     else
     {
           //Read PLAYER attributes
           getline(ReadStuff, petname);
                  getline(ReadStuff, species);
                  getline(ReadStuff, petimagefile);
           getline(ReadStuff, petatksoundfile);
           getline(ReadStuff, petwinsoundfile);
           getline(ReadStuff, petlosesoundfile);
           getline(ReadStuff, specialattack);
                  getline(ReadStuff, specialdefense);
           ReadStuff >> health;
                  ReadStuff >> speed;
                  ReadStuff >> strength;
           ReadStuff >> attack;
           ReadStuff >> defense;
                  ReadStuff >> specialabilityenergy;

           ReadStuff.close();
                  if(ThePet == 1)
                  {
                      if(species == "Jonkey") { PET1 = new JONKEY; }
                      if(species == "Franther") { PET1 = new
FRANTHER; }
               if(species == "Snabbit") { PET1 = new SNABBIT; }
                      if(species == "Liger") { PET1 = new LIGER; }
                      if(species == "Crog") { PET1 = new CROG; }
               if(species == "Squeel") { PET1 = new SQUEEL; }

                      PET1->SetPetName(CONVERT(petname));
                      PET1->SetSpecies(CONVERT(species));
                      PET1->SetPetImageFile(CONVERT(petimagefile));
                      PET1-
>SetPetAtkSoundFile(CONVERT(petatksoundfile));
                      PET1-
>SetPetWinSoundFile(CONVERT(petwinsoundfile));
                      PET1-
>SetPetLoseSoundFile(CONVERT(petlosesoundfile));
                      PET1->SetSpcAtk(CONVERT(specialattack));
               PET1->SetSpcDef(CONVERT(specialdefense));
               PET1->SetHealth(health);
                      PET1->SetSpeed(speed);
                      PET1->SetStrength(strength);
                      PET1->SetAttack(attack);
```

```
                    PET1->SetDefense(defense);
                    PET1-
>SetSpecialAbilityEnergy(specialabilityenergy);
               PET1->SetPetInterface(1);
                   PET1->View();

                   }//close if
                   if(ThePet == 2)
                   {
                       if(species == "Jonkey") { PET2 = new JONKEY; }
                       if(species == "Franther") { PET2 = new
FRANTHER; }
               if(species == "Snabbit") { PET2 = new SNABBIT; }
                       if(species == "Liger") { PET2 = new LIGER; }
                       if(species == "Crog") { PET2 = new CROG; }
               if(species == "Squeel") { PET2 = new SQUEEL; }

                       PET2->SetPetName(CONVERT(petname));
                       PET2->SetSpecies(CONVERT(species));
                       PET2->SetPetImageFile(CONVERT(petimagefile));
                       PET2-
>SetPetAtkSoundFile(CONVERT(petatksoundfile));
                       PET2-
>SetPetWinSoundFile(CONVERT(petwinsoundfile));
                       PET2-
>SetPetLoseSoundFile(CONVERT(petlosesoundfile));
                       PET2->SetSpcAtk(CONVERT(specialattack));
               PET2->SetSpcDef(CONVERT(specialdefense));
               PET2->SetHealth(health);
                       PET2->SetSpeed(speed);
                       PET2->SetStrength(strength);
                       PET2->SetAttack(attack);
                       PET2->SetDefense(defense);
                       PET2-
>SetSpecialAbilityEnergy(specialabilityenergy);
               PET2->SetPetInterface(2);
                   PET2->View();
                   }//close if
    TEMP = "\r\n\r\n The Pet was successfully loaded.";
    }//close else
    MainOutput->SetWindowText(TEMP);
}//close function
//---------------------------------------------------------------------
----------------------
void Combat()
{
     int WhoGoesFirst = (rand() % 2) + 1;
     TEMP = "";
     if(WhoGoesFirst == 1)
         {
         TEMP = TEMP + "\r\n  " + PET1->GetPetName() + " the "
                              + PET1->GetSpecies()
                              + " gets\r\n"
                              + "  the luck of the draw with\r\n"
```

```
                            + "  the very FIRST ATTACK!\r\n";
                MainOutput->SetWindowText(TEMP);
          XSleep(3000);
          while(PET1->GetHealth() > 0 && PET2->GetHealth() > 0)
              {
                  TEMP = "";
              TEMP = TEMP + " " + PET1->GetPetName() + "'s ATTACK
begins!\r\n"
                                    + "-------------------------------
--------------";
                         MainOutput->SetWindowText(TEMP);
                         XSleep(4000);
                         PET1->Attack(PET2);
                     if(PET2->GetHealth() > 0)
                         {
                TEMP = "";
                TEMP = TEMP + " " + PET2->GetPetName() + " makes a
counter-attack!\r\n"
                                    + "-------------------------------
--------------";
                                MainOutput->SetWindowText(TEMP);
                                XSleep(4000);
                                PET2->Attack(PET1);
                         }
              }
          }
          else
          {
          TEMP = TEMP + "\r\n  " + PET2->GetPetName() + " the "
                                 + PET2->GetSpecies()
                                 + " gets\r\n"
                                 + "  the luck of the draw with\r\n"
                             + "  the very FIRST ATTACK!\r\n";
                MainOutput->SetWindowText(TEMP);
          XSleep(3000);
                 while(PET1->GetHealth() > 0 && PET2->GetHealth() > 0)
              {
                  TEMP = "";
              TEMP = TEMP + " " + PET2->GetPetName() + "'s ATTACK
begins!\r\n"
                                    + "-------------------------------
--------------";
                         MainOutput->SetWindowText(TEMP);
                         XSleep(4000);
                         PET2->Attack(PET1);

                     if(PET1->GetHealth() > 0)
                         {
                TEMP = "";
                TEMP = TEMP + " " + PET1->GetPetName() + " makes a
counter-attack!\r\n"
                                    + "-------------------------------
--------------";
                                MainOutput->SetWindowText(TEMP);
```

```
                XSleep(4000);
                            PET1->Attack(PET2);
                        }
            }
        }

        TEMP = "";
     TEMP = TEMP + "\r\n\r\n Wait for it...";
        MainOutput->SetWindowText(TEMP);
     XSleep(3000);
     //Declare winner
        TEMP = "";
     if(PET1->GetHealth() > 0)
        {
                TEMP = TEMP + "\r\n The " + PET1->GetSpecies() + "
"
                            + PET1->GetPetName() + " wins\r\n"
                               + " the TOURNAMENT!";
                MainOutput->SetWindowText(TEMP);
                Pet1TEMP = "\r\n\r\n " + PET1->GetPetName()
                                        + " was\r\n" + " VICTORIUS!";
           Pet1_Status->SetWindowText(Pet1TEMP);
                PlaySound(PET1-
>GetPetWinSoundFile(),NULL,SND_FILENAME|SND_ASYNC);
           XSleep(8000);
           Pet2TEMP + "\r\n\r\n " + PET2->GetPetName()
                                        + " was\r\n" + " DEFEATED!";
                Pet2_Status->SetWindowText(Pet2TEMP);
                PlaySound(PET2-
>GetPetLoseSoundFile(),NULL,SND_FILENAME|SND_ASYNC);
        }
        else
        {
                TEMP = TEMP + "\r\n The " + PET2->GetSpecies() + "
"
                            + PET2->GetPetName() + " wins\r\n"
                               + " the TOURNAMENT!";
                MainOutput->SetWindowText(TEMP);
                Pet2TEMP = "\r\n\r\n " + PET2->GetPetName()
                                        + " was\r\n" + " VICTORIUS!";
           Pet2_Status->SetWindowText(Pet2TEMP);
                PlaySound(PET2-
>GetPetWinSoundFile(),NULL,SND_FILENAME|SND_ASYNC);
           XSleep(8000);
           Pet1TEMP + "\r\n\r\n " + PET1->GetPetName()
                                        + " was\r\n" + " DEFEATED!";
                Pet1_Status->SetWindowText(Pet1TEMP);
                PlaySound(PET1-
>GetPetLoseSoundFile(),NULL,SND_FILENAME|SND_ASYNC);
        }

}
//---------------------------------------------------------------------
```

File 6: MFCSleep.h

```
struct XSleep_Structure
{
        int duration;
        HANDLE eventHandle;
};
DWORD WINAPI XSleepThread(LPVOID pWaitTime)
{
        XSleep_Structure *sleep = (XSleep_Structure *)pWaitTime;
        Sleep(sleep->duration);
        SetEvent(sleep->eventHandle);
        return 0;
}
void XSleep(int nWaitInMSecs)
{
        XSleep_Structure sleep;

        sleep.duration          = nWaitInMSecs;
        sleep.eventHandle       = CreateEvent(NULL, TRUE, FALSE, NULL);
        DWORD threadId;
        CreateThread(NULL, 0, &XSleepThread, &sleep, 0, &threadId);
        MSG msg;

        while(::WaitForSingleObject(sleep.eventHandle, 0) ==
WAIT_TIMEOUT)
        {
                //get and dispatch messages
                if(::PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
                {
                        ::TranslateMessage(&msg);
                        ::DispatchMessage(&msg);
                }
        }
        CloseHandle(sleep.eventHandle);
}
```

File 7: resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by MegaPet2.rc
//
#define IDD_Interface                   108
#define IDB_START                       109
#define IDB_QUIT                        111
#define cbb_START                       112
#define cbb_QUIT                        113
#define B_START                         1015
#define B_QUIT                          1020
```

```
#define CE_MainOutput                   1021
#define B_HELP                          1022
#define PC_PET_1                        1023
#define PC_PET_2                        1024
#define CS_Pet1_Label                   1025
#define CS_Pet2_Label                   1026
#define CS_MainOutput_Label             1027
#define B_PET1_LOAD                     1028
#define B_PET1_NEW                      1029
#define B_PET1_SAVE                     1030
#define B_PET1_TRAIN                    1031
#define B_PET1_FEED                     1032
#define B_PET1_PLAY                     1033
#define B_PET1_TALK                     1034
#define B_PET1_SLEEP                    1035
#define B_PET1_KICK                     1036
#define B_PET1_ATTACK                   1037
#define B_PET2_LOAD                     1038
#define B_PET1_DEFEND                   1039
#define B_PET1_SPEC_ATTACK              1040
#define B_PET1_SPEC_DEFEND              1041
#define B_PET2_NEW                      1042
#define B_PET2_SAVE                     1043
#define B_PET2_TRAIN                    1044
#define B_PET2_FEED                     1045
#define B_PET2_PLAY                     1046
#define B_PET2_TALK                     1047
#define B_PET2_SLEEP                    1048
#define B_PET2_KICK                     1049
#define B_PET2_ATTACK                   1050
#define B_PET2_DEFEND                   1051
#define B_PET2_SPEC_ATTACK              1052
#define B_PET2_SPEC_DEFEND              1053
#define CS_LABEL_PET1_HEALTH            1054
#define CS_PET1_BOX_HEALTH              1055
#define CS_LABEL_PET1_NAME              1056
#define CS_PET1_BOX_NAME                1057
#define CS_LABEL_PET1_SPEED             1058
#define CS_PET1_BOX_SPEED               1059
#define CS_LABEL_PET1_STRENGTH          1060
#define CS_PET1_BOX_STRENGTH            1061
#define CS_LABEL_PET1_SPCATK            1062
#define CS_PET1_BOX_SPCATK              1063
#define CS_LABEL_PET1_SPCDEF            1064
#define CS_PET1_BOX_SPCDEF              1065
#define CS_LABEL_PET1_ATK               1066
#define CS_PET1_BOX_ATK                 1067
#define CS_LABEL_PET1_DEF               1068
#define CS_PET1_BOX_DEF                 1069
#define CS_LABEL_PET2_HEALTH            1070
#define CS_PET2_BOX_HEALTH              1071
#define CS_LABEL_PET2_NAME              1072
#define CS_PET2_BOX_NAME                1073
#define CS_LABEL_PET2_SPEED             1074
```

```
#define CS_PET2_BOX_SPEED               1075
#define CS_LABEL_PET2_STRENGTH          1076
#define CS_PET2_BOX_STRENGTH            1077
#define CS_LABEL_PET2_SPCATK            1078
#define CS_PET2_BOX_SPCATK              1079
#define CS_LABEL_PET2_SPCDEF            1080
#define CS_PET2_BOX_SPCDEF              1081
#define CS_LABEL_PET2_ATK               1082
#define CS_PET2_BOX_ATK                 1083
#define CS_LABEL_PET2_DEF               1084
#define CS_PET2_BOX_DEF                 1085
#define B_BEGIN_TOURNAMENT              1086
#define B_END_TOURNAMENT                1087
#define CE_PET1_STATUS                  1088
#define CE_PET2_STATUS                  1089
#define CS_PET1_LABEL_STATUS            1090
#define CS_PET2_LABEL_STATUS            1091
#define SC_PET1                         1092
#define SC_PET2                         1093
#define CS_PET1_LABEL_HEATHMETER        1094
#define CS_PET1_HM_0                    1095
#define CS_PET1_HM_50                   1096
#define CS_PET1_HM_100                  1097
#define CS_PET2_LABEL_HEATHMETER        1098
#define CS_PET2_HM_0                    1099
#define CS_PET2_HM_50                   1100
#define CS_PET2_HM_100                  1101
#define CE_INPUT                        1102
#define CS_INPUT                        1103
#define CS_PET1_BOX_SPCABILENERGY       1104
#define CS_LABEL_PET1_SPCABILENERGY     1105
#define CS_PET2_BOX_SPCABILENERGY       1106
#define CS_LABEL_PET2_SPCABILENERGY     1107
#define B_ENTER                         1108
#define CS_PET1_BOX_SPECIES             1109
#define CS_LABEL_PET1_SPECIES           1110
#define CS_PET1_BOX_SPECIES2            1111
#define CS_PET2_BOX_SPECIES             1111
#define CS_LABEL_PET1_SPECIES2          1112
#define IDC_BUTTON1                     1113
#define cbb                             1113
#define PB_Pet1Life                     1114
#define PB_Pet2Life                     1115
#define CS_Pet1HealthBar                1116
#define CS_Pet2HealthBar                1117
// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        114
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1117
#define _APS_NEXT_SYMED_VALUE           101
#endif
```

```
#endif
```

File 8: resource.rc

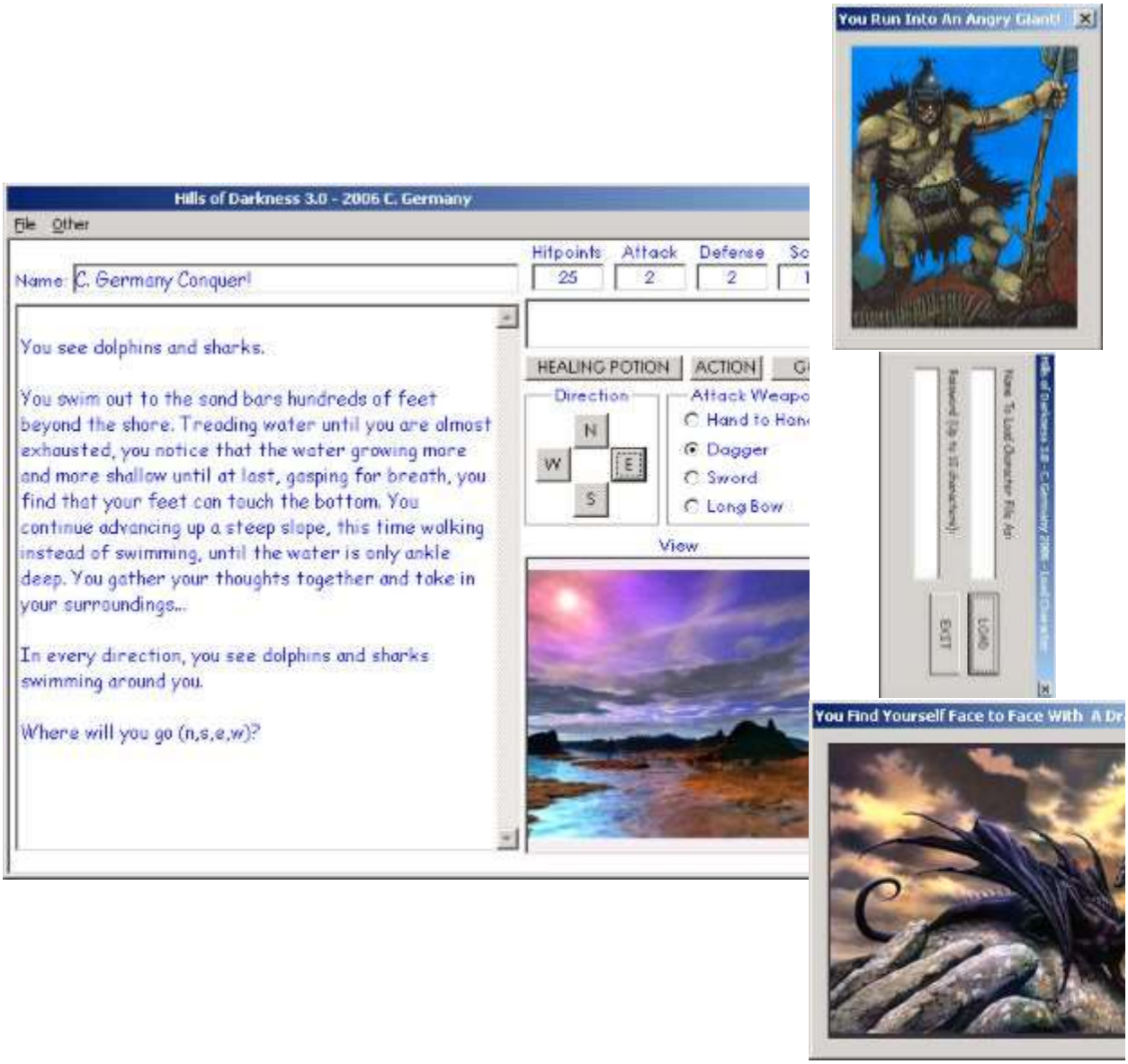
```
// Microsoft Visual C++ generated resource script.
//
#include "resource.h"
#define APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////////
////
#undef APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////////////
////
// English (U.S.) resources
#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32
#ifdef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////////
////
//
// TEXTINCLUDE
//
1 TEXTINCLUDE
BEGIN
    "resource.h\0"
END
2 TEXTINCLUDE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END
3 TEXTINCLUDE
BEGIN
    "\r\n"
    "\0"
END
#endif    // APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////////
////
//
// Dialog
//
IDD_Interface DIALOGEX 0, 0, 436, 335
```

```
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION |
WS_SYSMENU
CAPTION "MegaPet 1.5"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    PUSHBUTTON      "EXIT",B_QUIT,288,169,0,0,BS_BITMAP
    EDITTEXT        CE_MainOutput,119,20,190,147,ES_MULTILINE |
ES_READONLY | WS_VSCROLL | NOT WS_TABSTOP,WS_EX_TRANSPARENT
    PUSHBUTTON      "START",B_START,227,169,0,0,BS_BITMAP
    CONTROL         "",PC_PET_1,"Static",SS_BITMAP | SS_CENTERIMAGE |
WS_BORDER,34,20,76,61
    CONTROL         "",PC_PET_2,"Static",SS_BITMAP | SS_CENTERIMAGE |
WS_BORDER,314,20,76,61
    CTEXT           "Pet 1",CS_Pet1_Label,37,9,71,8
    CTEXT           "Pet 2",CS_Pet2_Label,316,10,71,8
    CTEXT           "Tournament Status",CS_MainOutput_Label,174,8,74,13
    PUSHBUTTON      "LOAD",B_PET1_LOAD,35,85,24,12
    PUSHBUTTON      "NEW",B_PET1_NEW,86,85,24,12
    PUSHBUTTON      "SAVE",B_PET1_SAVE,60,85,24,12
    PUSHBUTTON      "TRAIN",B_PET1_TRAIN,35,100,24,12
    PUSHBUTTON      "FEED",B_PET1_FEED,60,100,24,12
    PUSHBUTTON      "PLAY",B_PET1_PLAY,86,100,24,12
    PUSHBUTTON      "TALK",B_PET1_TALK,35,115,24,12
    PUSHBUTTON      "SLEEP",B_PET1_SLEEP,60,115,24,12
    PUSHBUTTON      "KICK",B_PET1_KICK,86,115,24,12
    PUSHBUTTON      "ATTACK",B_PET1_ATTACK,35,129,36,12
    PUSHBUTTON      "DEFEND",B_PET1_DEFEND,74,129,36,12
    PUSHBUTTON      "SPECIAL ATTACK",B_PET1_SPEC_ATTACK,35,143,74,12
    PUSHBUTTON      "SPECIAL DEFEND",B_PET1_SPEC_DEFEND,35,157,74,12
    PUSHBUTTON      "HELP",B_HELP,248,172,29,11
    PUSHBUTTON      "LOAD",B_PET2_LOAD,315,85,24,12
    PUSHBUTTON      "NEW",B_PET2_NEW,366,85,24,12
    PUSHBUTTON      "SAVE",B_PET2_SAVE,340,85,24,12
    PUSHBUTTON      "TRAIN",B_PET2_TRAIN,315,100,24,12
    PUSHBUTTON      "FEED",B_PET2_FEED,340,100,24,12
    PUSHBUTTON      "PLAY",B_PET2_PLAY,366,100,24,12
    PUSHBUTTON      "TALK",B_PET2_TALK,315,115,24,12
    PUSHBUTTON      "SLEEP",B_PET2_SLEEP,340,115,24,12
    PUSHBUTTON      "KICK",B_PET2_KICK,366,115,24,12
    PUSHBUTTON      "ATTACK",B_PET2_ATTACK,315,129,36,12
    PUSHBUTTON      "DEFEND",B_PET2_DEFEND,354,129,36,12
    PUSHBUTTON      "SPECIAL ATTACK",B_PET2_SPEC_ATTACK,315,143,74,12
    PUSHBUTTON      "SPECIAL DEFEND",B_PET2_SPEC_DEFEND,315,157,74,12
    LTEXT           "Health",CS_LABEL_PET1_HEALTH,37,197,22,8
    CTEXT           "",CS_PET1_BOX_HEALTH,66,197,43,9,WS_BORDER
    LTEXT           "Name",CS_LABEL_PET1_NAME,37,173,19,8
    CTEXT           "",CS_PET1_BOX_NAME,66,173,43,9,WS_BORDER
    LTEXT           "Speed",CS_LABEL_PET1_SPEED,37,209,21,8
    CTEXT           "",CS_PET1_BOX_SPEED,66,209,43,9,WS_BORDER
    LTEXT           "Str.",CS_LABEL_PET1_STRENGTH,37,222,13,8
    CTEXT           "",CS_PET1_BOX_STRENGTH,66,222,43,9,WS_BORDER
    LTEXT           "Sp. Atk.",CS_LABEL_PET1_SPCATK,37,260,27,8
    CTEXT           "",CS_PET1_BOX_SPCATK,66,260,43,9,WS_BORDER
    LTEXT           "Sp. Def.",CS_LABEL_PET1_SPCDEF,36,273,28,8
```

```
    CTEXT           "",CS_PET1_BOX_SPCDEF,66,273,43,9,WS_BORDER
    LTEXT           "Atk.",CS_LABEL_PET1_ATK,37,235,14,8
    CTEXT           "",CS_PET1_BOX_ATK,66,235,43,9,WS_BORDER
    LTEXT           "Def.",CS_LABEL_PET1_DEF,37,248,15,8
    CTEXT           "",CS_PET1_BOX_DEF,66,248,43,9,WS_BORDER
    LTEXT           "Health",CS_LABEL_PET2_HEALTH,317,198,22,8
    CTEXT           "",CS_PET2_BOX_HEALTH,346,197,43,9,WS_BORDER
    LTEXT           "Name",CS_LABEL_PET2_NAME,317,173,19,8
    CTEXT           "",CS_PET2_BOX_NAME,346,173,43,9,WS_BORDER
    LTEXT           "Speed",CS_LABEL_PET2_SPEED,317,211,21,8
    CTEXT           "",CS_PET2_BOX_SPEED,346,210,43,9,WS_BORDER
    LTEXT           "Str.",CS_LABEL_PET2_STRENGTH,317,223,13,8
    CTEXT           "",CS_PET2_BOX_STRENGTH,346,223,43,9,WS_BORDER
    LTEXT           "Sp. Atk.",CS_LABEL_PET2_SPCATK,317,261,27,8
    CTEXT           "",CS_PET2_BOX_SPCATK,346,260,43,9,WS_BORDER
    LTEXT           "Sp. Def.",CS_LABEL_PET2_SPCDEF,317,273,28,8
    CTEXT           "",CS_PET2_BOX_SPCDEF,346,273,43,9,WS_BORDER
    LTEXT           "Atk.",CS_LABEL_PET2_ATK,317,236,14,8
    CTEXT           "",CS_PET2_BOX_ATK,346,235,43,9,WS_BORDER
    LTEXT           "Def.",CS_LABEL_PET2_DEF,317,249,15,8
    CTEXT           "",CS_PET2_BOX_DEF,346,248,43,9,WS_BORDER
    PUSHBUTTON      "BEGIN T",B_BEGIN_TOURNAMENT,119,170,36,11
    PUSHBUTTON      "END T",B_END_TOURNAMENT,157,170,37,11
    EDITTEXT        CE_PET1_STATUS,113,218,93,100,ES_MULTILINE |
ES_AUTOHSCROLL
    EDITTEXT        CE_PET2_STATUS,218,217,93,101,ES_MULTILINE |
ES_AUTOHSCROLL
    CTEXT           "Pet 1 Status",CS_PET1_LABEL_STATUS,125,208,70,8
    CTEXT           "Pet 2 Status",CS_PET2_LABEL_STATUS,233,208,70,8
    CONTROL         "",SC_PET1,"msctls_trackbar32",TBS_BOTH | TBS_NOTICKS
| WS_TABSTOP,31,316,71,8
    CONTROL         "",SC_PET2,"msctls_trackbar32",TBS_BOTH | TBS_NOTICKS
| WS_TABSTOP,322,317,71,8
    LTEXT           "Pet1 Health
Meter",CS_PET1_LABEL_HEATHMETER,39,300,59,8
    CTEXT           "0",CS_PET1_HM_0,31,308,8,8
    CTEXT           "50",CS_PET1_HM_50,62,308,8,8
    CTEXT           "100",CS_PET1_HM_100,93,308,12,8
    LTEXT           "Pet2 Health
Meter",CS_PET2_LABEL_HEATHMETER,330,302,59,8
    CTEXT           "0",CS_PET2_HM_0,322,310,8,8
    CTEXT           "50",CS_PET2_HM_50,354,310,8,8
    CTEXT           "100",CS_PET2_HM_100,384,310,12,8
    EDITTEXT        CE_INPUT,119,193,142,12,ES_AUTOHSCROLL
    CTEXT           "Input",CS_INPUT,173,185,33,8
    CTEXT           "",CS_PET1_BOX_SPCABILENERGY,66,286,43,9,WS_BORDER |
NOT WS_GROUP
    LTEXT           "SA Enrg",CS_LABEL_PET1_SPCABILENERGY,36,286,26,8
    CTEXT           "",CS_PET2_BOX_SPCABILENERGY,346,286,43,9,WS_BORDER |
NOT WS_GROUP
    LTEXT           "SA Enrg",CS_LABEL_PET2_SPCABILENERGY,317,286,26,8
    DEFPUSHBUTTON   "ENTER",B_ENTER,267,193,36,12
    CTEXT           "",CS_PET1_BOX_SPECIES,66,185,43,9,WS_BORDER | NOT
WS_GROUP
```

```
    LTEXT           "Species",CS_LABEL_PET1_SPECIES,36,185,25,8
    CTEXT           "",CS_PET1_BOX_SPECIES2,346,185,43,9,WS_BORDER | NOT
WS_GROUP
    LTEXT           "Species",CS_LABEL_PET1_SPECIES2,316,185,25,8
    CONTROL         "",PB_Pet1Life,"msctls_progress32",PBS_VERTICAL |
WS_BORDER,11,34,13,288
    CONTROL         "",PB_Pet2Life,"msctls_progress32",PBS_VERTICAL |
WS_BORDER,403,34,13,288
    CTEXT           "HEALTH",CS_Pet1HealthBar,5,25,26,8
    CTEXT           "HEALTH",CS_Pet2HealthBar,398,25,26,8
END
/////////////////////////////////////////////////////////////////////////////
////
//
// DESIGNINFO
//
#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO
BEGIN
    IDD_Interface, DIALOG
    BEGIN
        RIGHTMARGIN, 392
        BOTTOMMARGIN, 333
    END
END
#endif    // APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
////
//
// Bitmap
//
IDB_START               BITMAP                  "Images\\start.bmp"
IDB_QUIT                BITMAP                  "Images\\quit.bmp"
#endif    // English (U.S.) resources
/////////////////////////////////////////////////////////////////////////////
////

#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 3 resource.
//
/////////////////////////////////////////////////////////////////////////////
////
#endif    // not APSTUDIO_INVOKED
```


Download: Hills3.exe Hills of Darkness 3.0 RPG Game - (2010 Visual Studio MFC Project)



2010 Visual Studio Version

This project is a continuation of the Adventure Game you created as a console program (See cproject2.html). We will convert this game to a graphical MFC game as we learn each new MFC component one by one. The true purpose of this project is to learn about the MFC, but it will be a fun project as a beneficial side effect to our MFC studies.

1. Create an EMPTY Win32 project. Click -> "File" -> -> "New" -> "Project" -> "Win32" -> "Win32 Project".
2. Name is and select a directory.
3. Select "Application Settings" -> "Empty Project" then click "Finish".
4. Rt-click project, select "Properties" -> "Configuration Properties" -> "General" -> "Use of MFC" and change it to "Use MFC in a Static Library". This will give you MFC components.
5. Disable Incremental Linking. Rt-click project, select "Properties" -> "Configuration Properties" -> "Linker" -> "General" ->"Enable Incremental Linking" and set it to "NO".

6. Add a resources file. Rt-click resources and add a DIALOG object.
7. Change the enumerated constant to match the Dialog name.
8. Note that unlike 2003, when calling SetWindowText() in 2010 the string passed in must be cast/converted using "L".

File 1 of 7 "Globals.h":

```
//Globals-----------------------------------------------------
#include <afxwin.h> // MFC core and standard components
#include "INTERFACE_MAIN_resources.h" // main symbols
#include <iomanip>
#include <string>
#include <windows.h>
#include <fstream>
#include <ctime>
#include <mmsystem.h>
#include "MFCsleep.h" //Necessary for pauses
//-----------------------------------------------------------
using namespace std;

//Globals
//Enumerated Constant for Player Location
enum EVENTS {INTRO, QUIT, GAMEOVER, YOUWIN, SHAMAN, GATE,
UNDERGRND, GIANTCAMP,
GIANTFIGHT, DRAGONFIGHT, CENTER1, N1, S1, E1, W1, N2, S2, E2,
W2};

bool LOCK;
bool NeedName;
bool Continue;
bool HiScoresToggle;
char choice[1];
int ConqueredDragons;
int ConqueredGiants;

//Globals to track various game events
bool W1GiantAlive;
bool E1DragonAlive;
bool S2MotleyCrewAlive;
bool FirstTimeInShamanHut;
bool CENTERFirstTime;
bool UNDERDragonPairAlive;
bool FoundHP_West2;
bool FoundHP_Shaman;

bool StartedGame;
int GoEvent;
int NagMe;
int location; //Note: Now location must be global since
event-based input.

//Function Prototype Used in Classes
```

```
int Randy(int n);

//2 Function Pointers for Dialog Popups
void (* AFunctionPointer1)();
void (* AFunctionPointer2)();

//Class Prototypes
class Monster;
class Character;
class Player;
class Dragon;
class Giant;
class Shaman;

//Function Prototypes
void InitializeGlobals();
int Introduction();
int Combat(Dragon * m, EVENTS CurrentLocation);
int Combat(Giant * m, EVENTS CurrentLocation);
int CENTER();
int NORTH1();
int SOUTH1();
int EAST1();
int WEST1();
int NORTH2();
int SOUTH2();
int EAST2();
int WEST2();
int SHAMANUT();
int GateWay();
int UnderGroundPassage();
int GiantCamp();
int DragonFight();
int GiantFight();
void Conquests();
void LockButtons(bool x);
int GameOver();
int YouWin();
bool SaveCharacter();
bool LoadCharacter(CString & problem);
bool SaveHighScores();
void DisplayHighScores();

//Global Pointers to Keep Character and Shaman on Heap
Character * CurrentPlayer;
Shaman * WiseWoman;

//Dialog Globals for MFC Window and Controls
CString MESSAGE;
CString INVENTORY;
CString CONQUESTS;
```

```
//Interface Globals (MFC)
CEdit * pInput;
CEdit * pOutput;
CStatic * pScore;
CEdit * pInventory;
CStatic * pHitpoints;
CStatic * pAttack;
CStatic * pDefense;
CStatic * pName;
CEdit * pConquests;
CStatic * pView;

CEdit * pSaveName;
CEdit * pSavePassword;
CEdit * pLoadName;
CEdit * pLoadPassword;

CButton * pDagger;
CButton * pSword;
CButton * pLongBow;
CButton * pHand;

CButton * pN;
CButton * pS;
CButton * pE;
CButton * pW;
CButton * pSCORES;
CButton * pSAVE;
CButton * pLOAD;
CButton * pSTART;
CButton * pGO;
CButton * pACTION;
CButton * pHEAL;
```

File 2 of 7 "Classes.h":

```
//Classes

#include "Globals.h"

//Base class-----------------------------------------------------
class Monster
{
      public:

    //-------------------------------------------------------
      Monster(int hp=25, int atk = 1, int def=1)
      {
         pOutput->GetWindowText(MESSAGE);
         MESSAGE = MESSAGE + "\r\nCreating a BASE class monster.";
         pOutput->SetWindowText(MESSAGE);
```

```
      }
    //-------------------------------------------------------
      ~Monster()
      {
         pOutput->GetWindowText(MESSAGE);
         MESSAGE = MESSAGE + "\r\nDestroying a BASE class monster.";
         pOutput->SetWindowText(MESSAGE);
      }
    //-------------------------------------------------------

      //Accessor Methods
      void setHit(int hp) { hitpoints = hp; }
      void setAttack(int atk) { atak = atk; }
      void setDefense(int def) { defense = def; }
      void setName(char * nm) { name = nm; }
      int getHit() { return hitpoints; }
      int getAttack() { return atak; }
      int getDefense() { return defense; }
      char * getName() { return name; }
    //-------------------------------------------------------
      protected:
      int hitpoints;
      int atak;
      int defense;
      char * name;
};

//---------------------------------------------------------------------
//Base Class ADT for Characters and Players. Would allow player
//to travel with companions and interact if expanded further.

class Character
{
      public:
   //-------------------------------------------------------------
      //Constructor
      Character(int Hp = 30, int Atk = 1, int Def = 1, CString nm =
"Character")
      {
         pOutput->GetWindowText(MESSAGE);
         MESSAGE = MESSAGE + "\r\nCreating a BASE class Character.";
         pOutput->SetWindowText(MESSAGE);
         hitpoints = Hp; atak = Atk; defense = Def; CharName = nm; level
= 0;
         InitializeInventory();
         score = 0;
      }
   //-------------------------------------------------------------
      //Destructor
      ~Character()
      {
         pOutput->GetWindowText(MESSAGE);
         MESSAGE = MESSAGE + "\r\nDestroying a BASE class Character.";
         pOutput->GetWindowText(MESSAGE);
         MESSAGE = MESSAGE + "\r\nDestroying a BASE class Character.";
         pOutput->SetWindowText(MESSAGE);
```

```
    }
//-------------------------------------------------------------
    void DisplayStats()
    {
        char NumberBuffer[10] = "";
        string TempString = "";
        CString TempCString = "";

        pName->SetWindowText(" " + CharName);

        _itoa_s(hitpoints, NumberBuffer, 10);
        TempString = NumberBuffer;
        TempCString = TempString.c_str();
        pHitpoints->SetWindowText(TempCString);

        _itoa_s(atak, NumberBuffer, 10);
        TempString = NumberBuffer;
        TempCString = TempString.c_str();
        pAttack->SetWindowText(TempCString);

        _itoa_s(defense, NumberBuffer, 10);
        TempString = NumberBuffer;
        TempCString = TempString.c_str();
        pDefense->SetWindowText(TempCString);

        _itoa_s(score, NumberBuffer, 10);
        TempString = NumberBuffer;
        TempCString = TempString.c_str();
        pScore->SetWindowText(TempCString);
    }
//-------------------------------------------------------------
    void InitializeInventory()
    {
        dagger = false;
        sword = false;
        longbow = false;
        chainmail= false;
        fullbodyarmor= false;
        healingpotion = 0;
        FishKey = false;
    }
//-------------------------------------------------------------
    void Inventory()
    {
        INVENTORY = "";
        char NumBuffer[10];
        int n = 0;

        INVENTORY = INVENTORY + "\r\n";

        if(dagger)
        {
            ++n;
            INVENTORY = INVENTORY + _itoa(n,NumBuffer,10)
            + ". Dagger (+ 2 Atk)\r\n\r\n";
```

```
            }

            if(sword)
            {
                ++n;
                INVENTORY = INVENTORY + _itoa(n,NumBuffer,10)
                + ". Sword (+ 4 Atk)\r\n\r\n";
            }

            if(longbow)
            {
                ++n;
                INVENTORY = INVENTORY + _itoa(n,NumBuffer,10)
                + ". Long Bow (+2 Dist)\r\n\r\n";
            }

            if(chainmail)
            {
                ++n;
                INVENTORY = INVENTORY + _itoa(n,NumBuffer,10)
                + ". Chain Mail (+ 2 Def)\r\n\r\n";
            }

            if(fullbodyarmor)
            {
                ++n;
                INVENTORY = INVENTORY + _itoa(n,NumBuffer,10)
                + ". Armor (+4 Def)\r\n\r\n";
            }

            if(healingpotion > 0)
            {
                ++n;
                INVENTORY = INVENTORY + itoa(n,NumBuffer,10) +
                ". Healing Potion\r\n (+20HP)\r\n Quantity: " +
                _itoa(healingpotion,NumBuffer,10) + "\r\n\r\n";
            }

            if(FishKey)
            {

                ++n;
                INVENTORY = INVENTORY + _itoa(n,NumBuffer,10)
                + ". Fish Key\r\n (Object Access)\r\n";
            }

            if(!dagger && !sword && !longbow && !chainmail
                       && !fullbodyarmor && !healingpotion && ! FishKey)
            {
                 INVENTORY = INVENTORY
                 + "\r\n\r\nHow sad!\r\n\r\nAbsolute\r\nNothing!!!";
            }

            pInventory->SetWindowText(INVENTORY);
        }
    //-------------------------------------------------------
```

```
      void Attack(Monster * Opponent)
      {
            int damage = 0;
            char NumBuffer[10];
            damage = Randy(10) + atak;

            if(dagger && UseDagger)
            {
               MESSAGE = MESSAGE + CharName + " stabs with the
dagger!\r\n";
               damage = damage + 2;
               PlaySound(L"media/knife.wav",NULL,SND_FILENAME|SND_ASYNC);
            }
            else if(sword && UseSword)
            {
               MESSAGE = MESSAGE + " swings the sword!\r\n";
               damage = damage + 4;
               PlaySound(L"media/sword.wav",NULL,SND_FILENAME|SND_ASYNC);
            }
            else if(longbow && UseLongBow)
            {
               MESSAGE = MESSAGE + CharName + " releases the string of the
longbow!\r\n";
               damage = damage + 2;
               PlaySound(L"media/bow.wav",NULL,SND_FILENAME|SND_ASYNC);
            }
            else
            {
               MESSAGE = MESSAGE + CharName +
               " engages the oponent in brutal hand to hand combat!\r\n";
               PlaySound(L"media/hand.wav",NULL,SND_FILENAME|SND_ASYNC);
            }

            MESSAGE = MESSAGE + "\r\n********* " + CharName
            + " Attacks! *********\r\n" + "Before Attack:\r\n"
            + CharName + " Hitpoints = "
            + itoa(hitpoints,NumBuffer,10) + "\r\nOpponent Hitpoints = "
            + itoa(Opponent->getHit(),NumBuffer,10) + "\r\n";

            if(damage - Opponent->getDefense() > 0)
            { damage = damage - Opponent->getDefense(); }
            else { damage = 0; }

            //Prevent negative values. Check that oponent is still alive.
            if((Opponent->getHit() - damage) > 0)
            { Opponent->setHit((Opponent->getHit() - damage)); }
            else
            { Opponent->setHit(0); }

            MESSAGE = MESSAGE + "\r\nAfter Attack:\r\n" + CharName + "
Hitpoints = "
            + itoa(hitpoints,NumBuffer,10) + "\r\nOpponent Hitpoints = "
            + itoa(Opponent->getHit() ,NumBuffer,10) + "\r\n";

            DisplayStats();
```

```
            pOutput->SetWindowText(MESSAGE);
      }
   //-------------------------------------------------------------
      void Cheat()
      {
            setDagger(true);
            setSword(true);
            setLongBow(true);
            setChainMail(true);
            setFullBodyArmor(true);
            setHealingPotion(7);
            setFishKey(true);
      }
   //-------------------------------------------------------------
      void UseHealingPotion()
      {
            if(healingpotion > 0)
            {
                hitpoints += 20;
                --healingpotion;
                DisplayStats();
                Inventory();

                pOutput->GetWindowText(MESSAGE);
                MESSAGE = MESSAGE +
                "\r\n You drink the healing elixir and feel revived!";
            }
            else
            {
                pOutput->GetWindowText(MESSAGE);
                MESSAGE = MESSAGE +
                "\r\n Wishful thinking. You don't have any healing
potions!";
            }

            pOutput->SetWindowText(MESSAGE);
      }
   //-------------------------------------------------------------
-
      //Inventory Accessor Methods
      bool getDagger() { return dagger; }
      bool getSword() { return sword; }
      bool getLongBow() { return longbow; }
      bool getChainMail() { return chainmail; }
      bool getFullBodyArmor() { return fullbodyarmor; }
      int getHealingPotion() { return healingpotion; }
      bool getFishKey() { return FishKey; }
      void setDagger(bool x) { dagger = x; }
      void setSword(bool x) { sword = x; }
      void setLongBow(bool x) { longbow = x; }
      void setChainMail(bool x) { chainmail = x; }
      void setFullBodyArmor(bool x) { fullbodyarmor = x; }
      void setHealingPotion(int x) { healingpotion = healingpotion + x; }
      void setFishKey(bool x) { FishKey = x; }
      void setUseDagger(bool x) { UseDagger = x; }
      void setUseSword(bool x) { UseSword = x; }
```

```
      void setUseLongBow(bool x) { UseLongBow = x; }
   //-------------------------------------------------------
      //Character Attribute Accessor Methods
      void setHit(int hp) { hitpoints = hp; }
      void setAttack(int atk) { atak = atk; }
      void setDefense(int def) { defense = def; }
      void setLevel(int lvl) { level = lvl; }
      void setLocation(int loki) { location = loki; }
      void setName(CString nm) { CharName = nm; }
      void setScore(int sc) { score = sc; }
      int getHit() { return hitpoints; }
      int getAttack() { return atak; }
      int getDefense() { return defense; }
      int getLevel() { return level; }
      int getLocation() { return location; }
      int getScore() { return score; }
      CString getName() { return CharName; }
   //-------------------------------------------------------
      private:
      //Character Attribute Items
      int hitpoints;
      int atak;
      int defense;
      int level;
      int location;
      int score;
      CString CharName;
   //-------------------------------------------------------
      //Character Inventory Items
      bool dagger;
      bool sword;
      bool longbow;
      bool UseDagger;
      bool UseSword;
      bool UseLongBow;
      bool chainmail;
      bool fullbodyarmor;
      int healingpotion;
      bool FishKey;
};

//Derives from Character-----------------------------------------

class Player : public Character
{
      public:
   //-------------------------------------------------------
      Player()
      {
         pOutput->GetWindowText(MESSAGE);
         MESSAGE = MESSAGE + "\r\nCreating a DERIVED class Player.";
         pOutput->SetWindowText(MESSAGE);
      }
   //-------------------------------------------------------
      ~Player()
      {
```

```
         pOutput->GetWindowText(MESSAGE);
         MESSAGE = MESSAGE + "\r\nDestroying a DERIVED class Player.";
         pOutput->SetWindowText(MESSAGE);
      }
   //-------------------------------------------------------------
      //Accessor Methods
      private:
};

//---------------------------------------------------------------------

//Derives from Monster-----------------------------------------------

class Dragon : public Monster
{
      public:

   //-------------------------------------------------------------
      Dragon(int hp=25, int atk = 1, int def=1)
      {
         Monster::hitpoints = hp;
         Monster::atak = atk;
         Monster::defense = def;
         pOutput->GetWindowText(MESSAGE);
         MESSAGE = MESSAGE + "\r\nCreating a Dragon.";
         pOutput->SetWindowText(MESSAGE);
      }
   //-------------------------------------------------------------
      ~Dragon()
      {
         pOutput->GetWindowText(MESSAGE);
         MESSAGE = MESSAGE + "\r\nDestroying a Dragon.";
         pOutput->SetWindowText(MESSAGE);
      }
   //-------------------------------------------------------------
      void Attack(Character * Opponent)
      {
          //Since we are calling members of the base class, we use the
"this" pointer.
          char NumBuffer[10];
          MESSAGE = "";
          MESSAGE = MESSAGE + "\r\n********** Dragon Attacks!
**********\r\n" +
          "Before Attack:\r\nDragon Hitpoints = " +
          itoa(this->getHit(),NumBuffer,10) +
          "\r\n" + Opponent->getName() + " Hitpoints = " +
          itoa(Opponent->getHit(),NumBuffer,10) ;

          int damage = 0;
          damage = Randy(10) + this->getAttack();

          if(damage > Opponent->getDefense())
          { damage = damage - Opponent->getDefense(); }
          else
          { damage = 0; }
```

```
          if(Opponent->getFullBodyArmor())
          {
             if(damage > 4)
             { damage = damage - 4; }
          }

          if(Opponent->getChainMail())
          {
             if(damage > 2)
             { damage = damage - 2; }
          }

          //Prevent negative values. Check that oponent is still alive.
          if((Opponent->getHit() - damage) > 0)
          { Opponent->setHit((Opponent->getHit() - damage)); }
          else
          { Opponent->setHit(0); }

          MESSAGE = MESSAGE + "\r\n\r\nAfter Attack:\r\nDragon Hitpoints
= " +
          itoa(this->getHit(),NumBuffer,10) + "\r\n" +
          Opponent->getName() + " Hitpoints = " +
          itoa(Opponent->getHit(),NumBuffer,10);

          pOutput->SetWindowText(MESSAGE);
       }
    //-------------------------------------------------------------
       void BreatheFire()
       {
            pOutput->GetWindowText(MESSAGE);
            MESSAGE = MESSAGE + "\r\nDragon breathing fire!";
            pOutput->SetWindowText(MESSAGE);
       }
    //-------------------------------------------------------------
       //Accesor Methods
       void setCanFly(bool fly) { CanFly = fly; }
       bool getCanFly() { return CanFly; }

       private:
       bool CanFly;
};

//Derives from Monster-----------------------------------------------

class Giant : public Monster
{
       public:
    //-------------------------------------------------------------
       Giant(int hp=20, int atk = 1, int def=1)
       {
           Monster::hitpoints = hp;
           Monster::atak = atk;
           Monster::defense = def;
           pOutput->GetWindowText(MESSAGE);
           MESSAGE = MESSAGE + "\r\nCreating a Giant.";
           pOutput->SetWindowText(MESSAGE);
```

```
      }
   //-------------------------------------------------------------
      ~Giant()
      {
          pOutput->GetWindowText(MESSAGE);
          MESSAGE = MESSAGE + "\r\nDestroying a Giant.";
          pOutput->SetWindowText(MESSAGE);
      }
   //-------------------------------------------------------------
      void Stomp()
      {
           pOutput->GetWindowText(MESSAGE);
           MESSAGE = MESSAGE + "\r\nGiant stomping on object.";
           pOutput->SetWindowText(MESSAGE);
      }
   //-------------------------------------------------------------
      void SwingClub()
      {
           pOutput->GetWindowText(MESSAGE);
           MESSAGE = MESSAGE + "\r\nGiant swinging club.";
           pOutput->SetWindowText(MESSAGE);
      }
   //-------------------------------------------------------------
      void Attack(Character * Opponent)
      {
          //Since we are calling members of the base class, we use the
"this" pointer.
          char NumBuffer[10];

          MESSAGE = MESSAGE + "\r\n********** Giant Attacks!
**********\r\n" +
          "Before Attack:\r\nGiant Hitpoints = " +
          itoa(this->getHit(),NumBuffer,10) +
          + "\r\n" + Opponent->getName() + " Hitpints = " +
          itoa(Opponent->getHit(),NumBuffer,10);

          int damage = 0;
          damage = Randy(10) + this->getAttack();

          if(damage > Opponent->getDefense())
          { damage = damage - Opponent->getDefense(); }
          else { damage = 0; }

          if(Opponent->getFullBodyArmor())
          {
             if(damage > 4)
             { damage = damage - 4; }
          }

          if(Opponent->getChainMail())
          {
             if(damage > 2)
             { damage = damage - 2; }
          }

          //Prevent negative values. Check that oponent is still alive.
```

```
          if((Opponent->getHit() - damage) > 0)
          {
              Opponent->setHit((Opponent->getHit() - damage));
          }
          else
          { Opponent->setHit(0); }

          MESSAGE = MESSAGE + "\r\n\r\nAfter Attack:\r\nGiant Hitpoints =
" +
          itoa(this->getHit(),NumBuffer,10) + "\r\n"
          + Opponent->getName() + " Hitpoints = "
          + itoa(Opponent->getHit(),NumBuffer,10);

          pOutput->SetWindowText(MESSAGE);
      }
   //-------------------------------------------------------------
      //Accessor Methods
      void setClub(bool club) { HasClub = club; }
      bool getClub() { return HasClub; }

      private:
      bool HasClub;
};

//-------------------------------------------------------------

class Shaman : public Character
{
      public:
   //-------------------------------------------------------------
      Shaman()
      {
         pOutput->GetWindowText(MESSAGE);
         MESSAGE = MESSAGE + "\r\nCreating a DERIVED class Shaman.";
         pOutput->SetWindowText(MESSAGE);
      }
   //-------------------------------------------------------------
      ~Shaman()
      {
         pOutput->GetWindowText(MESSAGE);
         MESSAGE = MESSAGE + "\r\nDestroying a Shaman.";
         pOutput->SetWindowText(MESSAGE);
      }
   //-------------------------------------------------------------
      void Talk()
      {
           int SayWhat;
           SayWhat = Randy(10);

           MESSAGE = "";
           MESSAGE = MESSAGE + "\r\nThe Shaman looks at you from beneath
her priestly "
           + "garments and says," + "\r\n\r\n\"";

           switch(SayWhat)
           {
```

```
                case 1 : MESSAGE = MESSAGE
                         + "I like wild flowers. They are very beautiful,
and\r\n"
                         + "their restorative powers are merely a fringe
benefit";
                         break;
                case 2 : MESSAGE = MESSAGE
                         + "Do not look at the outward appearance of
things,\r\n"
                         + "character should be judged by what is on the
inside.";
                         break;
                case 3 : MESSAGE = MESSAGE
                         + "Good karma, bad karma, it's all the same..";
                         break;
                case 4 : MESSAGE = MESSAGE + "I have a secret to
tell...";
                         break;
                case 5 : MESSAGE = MESSAGE + "To unlock the gate between
worlds one needs a key";
                         break;
                case 6 : MESSAGE = MESSAGE + "You are not from this world,
I see that now";
                         break;
                case 7 : MESSAGE = MESSAGE
                         + "You did not yet know it traveler, but you must
"
                         + "\r\nseek the key of the fish god!";
                         break;
                case 8 : MESSAGE = MESSAGE + "Sometimes I wish I'd never
taken that vow of chastity";
                         break;
                case 9 : MESSAGE = MESSAGE + "Do you think I'm pretty?
Don't judge a book by its cover!";
                         break;
                case 10 : MESSAGE = MESSAGE +"Beware the edge of the
forest. Giants are afoot";
                          break;
                default : MESSAGE = MESSAGE + "Uh oh, this should never
happen..";
                          break;
            }//closes switch

            MESSAGE = MESSAGE + "\".";

            pOutput->SetWindowText(MESSAGE);

       }//close talk function
    //-------------------------------------------------------------
       private:
       bool staff;
       bool medicinebag;
};
```

File 3 of 7 "Functions.h":

```
#include "Classes.h"

//Function Definitions--------------------------------------------
void InitializeGlobals()
{
     LOCK = false;
     Continue = true;
     location = INTRO;

     W1GiantAlive = true;
     E1DragonAlive = true;
     S2MotleyCrewAlive = true;
     FirstTimeInShamanHut = true;
     CENTERFirstTime = true;
     UNDERDragonPairAlive = true;
     FoundHP_West2 = false;
     FoundHP_Shaman = false;

     StartedGame = false;
     NeedName = true;
     GoEvent = 1;
     NagMe = 0;
     HiScoresToggle = true;

     ConqueredDragons = 0;
     ConqueredGiants = 0;
}

//-----------------------------------------------------------------

int Introduction()
{
     if(choice[0] == 'g')
     {
        location = CENTER1;
        LOCK = false;
        return location;
     }
     else
     {
        LOCK = true;
        MESSAGE = "";
        MESSAGE = MESSAGE + "\r\n "
        + "Hills of Darkness 3.0\r\n\r\n";

//Note: For sound, you must do three things:
//1. Go to project properties, then under Linker, find Input.
//In the box labelled Aditional Dependancies add "winmm.lib".
//2. Include the file: #include <mmsystem.h> .
//3. Use command: PlaySound("west.wav",NULL,SND_FILENAME|SND_ASYNC);
```

```
//Options: SYNC = ends with function, async = keep playing
//SND_LOOP = loop it, must be stopped then with "StopSound()".
//PlaySound("west.wav",NULL,SND_FILENAME|SND_ASYNC|SND_LOOP);
PlaySound(L"media/theme.wav",NULL,SND_FILENAME|SND_ASYNC);

        MESSAGE = MESSAGE + "You awake from what appears to be a
disturbing dream, "
        + "knowing neither where you've been nor how you got where "
        + "you are now. You slowly rise to your feet, bewildered "
        + " and almost oblivious to the throbbing ache pulsating "
        + "between your ringing ears.\r\n\r\nYou find yourself "
        + "standing on a flowing grassy knoll amidst the dark green
"
        + "hills of medieval Scotland. In the skies above, dark gray
"
        + "clouds are passing in billowing random patterns. It
appears "
        + "a storm is approaching from the east. A few black ravens "
        + "fly over your head towards some unknown destination, a "
        + "familiar caw, their cries echoing softly against the "
        + "creeping shadows.";

        MESSAGE = MESSAGE + "\r\n\r\n\r\n"
        + "Type \"G\" in the command box and click the \"GO\" "
        + "button to continue...";

        HBITMAP IntroPix = (HBITMAP)
LoadImage(NULL,L"media/hills.bmp",
        IMAGE_BITMAP, 0,0,LR_DEFAULTSIZE | LR_LOADFROMFILE);

        pView->SetBitmap(IntroPix);

        Conquests();

        pInput->SetWindowText(L"g");
        pOutput->SetWindowText(MESSAGE);

        return location;
     }//close else
}

//---------------------------------------------------------------------

int CENTER()
{
    if(choice[0] == 'n' || choice[0] == 's' ||
    choice[0] == 'e' || choice[0] == 'w' ||
    choice[0] == 't' || choice[0] == '~' ||
    choice[0] == '!')
    {
       MESSAGE = "";
```

```
       switch(choice[0])
       {
          case 'n' : location = N1; break;
          case 's' : location = S1; break;
          case 'e' : location = E1; break;
          case 'w' : location = W1; break;
          case 't' : if(!CENTERFirstTime)
                     {
                        location = GATE;
                        break;
                     }
                     else
                     {
                        MESSAGE = MESSAGE + "\n That was an invalid
choice.";
                        break;
                     }
          case '~' : CurrentPlayer->Cheat();
                     break;
          case '!' : CurrentPlayer->InitializeInventory();
                     break;
          default : MESSAGE = MESSAGE + "\n That was an invalid
choice.";
          break;
       }

    pOutput->SetWindowText(MESSAGE);
    LOCK = false;

    return location;
    }//close if

    else
    {
       LOCK = true;
       MESSAGE = "";

       if(CENTERFirstTime)
       {
          MESSAGE = MESSAGE + "\r\n" + CurrentPlayer->getName()
          + ", confused, you try to get your bearings. "
          + "You see nothing but large stone tablets and columns with
"
          + "what appear to be odd and archaic symbols engraved upon
them.\r\n";

          CENTERFirstTime = false;
       }
       else
       {
          MESSAGE = MESSAGE + "\r\nYou return to the location where
you first mysteriously appeared "
```

```
          + "in this strange medieval world. You notice that the
large stone "
          + "tablets and columns have arranged themselves into an
arch and hinged "
          + "gates. The symbols are constantly changing, disappearing
and re-"
          + "appearing at random intervals across the surface of the
tablets. ";
       }

        MESSAGE = MESSAGE + "\r\n\r\nYou see that you can move to the
north, south, east or west. "
        + "To the north, you see the ruins of an ancient castle
spread "
        + "out across the horizon. To the east, you see the lapping
waves "
        + "of the ocean against a sandy shore. To the south, you see
a "
        + "small village with gray cobblestone houses and smoldering
"
        + "chimneys. To the west, you see an abandoned farm house. ";

        MESSAGE = MESSAGE + "\r\n\r\n\r\nWhere will you go (n, s, e,
w)?\r\nOr will you "
        + "(t)ry the gates?\r\n";

        HBITMAP CenterPix = (HBITMAP)
LoadImage(NULL,L"media/center.bmp",IMAGE_BITMAP,
        0,0,LR_DEFAULTSIZE | LR_LOADFROMFILE);

        pView->SetBitmap(CenterPix);

        pOutput->SetWindowText(MESSAGE);
        location = CENTER1;

        return location;

    }//close else

}//close function

//-----------------------------------------------------------------

int NORTH1()
{
    if(choice[0] == 's' || choice[0] == 'n' ||
    choice[0] == 'e' || choice[0] == 'w')
    {
       MESSAGE = "";

       switch(choice[0])
       {
```

```
          case 's' : location = CENTER1; break;
          case 'n' : location = N2; break;
          case 'e' : MESSAGE = MESSAGE + "\r\nYou fail to scale the
east wall and "
                     + "turn back...\r\n";
                     break;
          case 'w' : MESSAGE = MESSAGE + "\r\nYou press against hard,
cold stone.\r\n";
                     if(!CurrentPlayer->getChainMail())
                     {
                        MESSAGE = MESSAGE + "You find a suit of chain
mail!\r\n";
                        CurrentPlayer->setChainMail(true);
                        CurrentPlayer->Inventory();
                     }
                     else
                     { MESSAGE = MESSAGE + "You already have the
chain mail!\r\n"; }
                     break;
          default : /*NADA*/ break;
      }

      pOutput->SetWindowText(MESSAGE);
      LOCK = false;
      return location;

    }//close if
    else
    {
       MESSAGE = "";
       LOCK = true;

       MESSAGE = MESSAGE
       + "\r\nYou find yourself amidst the ruins of an ancient
castle...";

       MESSAGE = MESSAGE
       + "\r\n\r\nFurther to the north, you see the delapidated
entrance "
       + "to the abandoned castle. It is adjoined by crumbling
towers, one at "
       + "each corner of the foundation. The entrance to the castle
is a "
       + "frail wooden door, looking as though it had been abandoned
for "
       + "over a hundred years. It probably would not be too
difficult to "
       + "force the door open...\r\n\r\n"
       + "At this point, you may explore the castle ruins, or go back
to "
       + "the SOUTH. You are surrounded by impassible castle walls to
"
```

```
       + "the east and the west.\r\n\r\n";

       MESSAGE = MESSAGE
       + "Where will you go (n,s,e,w)? ";

       HBITMAP CenterPix = (HBITMAP)
LoadImage(NULL,L"media/N1.bmp",IMAGE_BITMAP,
       0,0,LR_DEFAULTSIZE | LR_LOADFROMFILE);

       pView->SetBitmap(CenterPix );

       pOutput->SetWindowText(MESSAGE);
       location = N1;

       return location;

    }//close else

}//close function

//-----------------------------------------------------------------

int SOUTH1()
{
if(choice[0] == 's' || choice[0] == 'n' || choice[0] == 'e' ||
choice[0] == 'w' || choice[0] == 'g')
{
MESSAGE = "";

switch(choice[0])
{
case 'n' : location = CENTER1; break;
case 's' : location = S2; break;
case 'e' : MESSAGE = MESSAGE
+ "You have no right to enter someone "
+ "else's dwelling!\r\n";
break;
case 'w' : if(!CurrentPlayer->getSword())
{
MESSAGE = MESSAGE
+ "\r\nBonus!!!\r\n You can not manage to ascend the";
MESSAGE = MESSAGE
+ " gate in front of you, but you do find a broad "
+ "sword at the base of the wall!\r\n";
CurrentPlayer->setSword(true);
CurrentPlayer->Inventory();
}
else
{
MESSAGE = MESSAGE
+ "You already took the sword!\r\n";
}
```

```
break;
case 'g' : location = SHAMAN; break;
default : MESSAGE = MESSAGE
+ "That was an invalid choice.\r\n"; break;
}

pOutput->SetWindowText(MESSAGE);
LOCK = false;
return location;

}//close if

else
{
LOCK = true;

MESSAGE = MESSAGE
+ "\r\nYou stumble into the gates of a rustic village. You see what "
+ "appears to be a tavern to the north. Further south, you see a "
+ "winding dirt road that meanders towards the horizon. All around "
+ "you are peasants buying and selling wares in an open market "
+ "place. Near the center of the village several children are "
+ "playing, and in the midst of them sits an elderly woman, "
+ "looking very wise and thoughtful.\r\n\r\n";

MESSAGE = MESSAGE
+ "At this point, you may only go back to the NORTH or further SOUTH,
"
+ "if you so desire. You are surrounded by what seems impassible "
+ "terrain to the east and the west and several cottages.\r\n\r\n "
+ "Towards thecenter of the villiage, you notice a small though "
+ "nicely maintained Shaman's lodge with smoke billowing from its
roof.";

MESSAGE = MESSAGE
+ "\r\n\r\nChoices: (n,e,s,w) (g)o into the Shaman's Hut ";

HBITMAP CenterPix = (HBITMAP)
LoadImage(NULL,L"media/S1.bmp",IMAGE_BITMAP,
0,0,LR_DEFAULTSIZE | LR_LOADFROMFILE);

pView->SetBitmap(CenterPix );

pOutput->SetWindowText(MESSAGE);
location = S1;
return location;

}//close else

}//close function

//-----------------------------------------------------------------
```

```
int EAST1()
{
if(choice[0] == 's' || choice[0] == 'n' || choice[0] == 'e' ||
choice[0] == 'w')
{
MESSAGE = "";

switch(choice[0])
{
case 'w' : location = CENTER1; break;
case 'n' : if(E1DragonAlive)
{
MESSAGE = MESSAGE
+ "\r\nYou creep towards the Dragon. Startled, "
+ " it climbs into the sky\nto defend itself!\r\n";
Dragon Prometheus; //Example of passing object on stack
location = Combat(&Prometheus, E1);
E1DragonAlive = false;
PlaySound(L"media/theme.wav",NULL,SND_FILENAME|SND_ASYNC);
//break out of while true, re-invoke EAST1 to display text
break;
}
else
{
MESSAGE = MESSAGE
+ "\r\nYou see a noble red dragon, tragically and"
+ " yet recently slain...\r\n";
break;
}
case 'e' : MESSAGE = MESSAGE
+ "\r\nYou jump into the water. It's freezing. "
+"You catch a cold.\r\n";
location = E2; break;
case 's' : MESSAGE = MESSAGE
+ "\r\nYou step on a jellyfish and it stings you with"
+ " its tentacles!\r\n";
if(CurrentPlayer->getHit() > 1)
{
CurrentPlayer->setHit((CurrentPlayer->getHit() - 1));
CurrentPlayer->DisplayStats();
}
else
{
CurrentPlayer->setHit(0);
MESSAGE = MESSAGE
+ "How pathetic! Slain by a jellyfish!\r\n";
Continue = false;
location = QUIT;
}
break;
default : MESSAGE = MESSAGE
```

```
+ "That was an invalid choice.\r\n"; break;
}
pOutput->SetWindowText(MESSAGE);
LOCK = false;
return location;
}//close if

else
{
MESSAGE = "";
LOCK = true;

MESSAGE = MESSAGE
+ "\r\nYou arrive at a sandy shore where green-blue translucent "
+ "waves are crashing against rocky outcroppings. "
+ "Overhead, seagulls are gathering, their cries echo against "
+ "an amphitheatre of sand and sea, intermingling with the sound of"
+ " the waves as they crash upon the beach.\r\n\r\nTo the north, ";

if(E1DragonAlive)
{
MESSAGE = MESSAGE
+ "a magnificent red dragon folds its wings, smoke billowing "
+ "from its nostrils.\r\n\r\n";
}

else
{ MESSAGE = MESSAGE
+ "a slain dragon is being devoured by ravens...\r\n\r\n"; }

MESSAGE = MESSAGE
+ "At this point, you may only go back to the WEST. You are "
+ "surrounded by turbulent water and razor sharp rocks to the "
+ "east and the south.\r\n";

MESSAGE = MESSAGE
+ "\r\nWhere will you go (n,s,e,w)? ";

HBITMAP CenterPix = (HBITMAP)
LoadImage(NULL,L"media/E1.bmp",IMAGE_BITMAP,
0,0,LR_DEFAULTSIZE | LR_LOADFROMFILE);

pView->SetBitmap(CenterPix );

pOutput->SetWindowText(MESSAGE);
location = E1;
return location;

}//close else

}//close function
```

```
//---------------------------------------------------------------

int WEST1()
{
if(choice[0] == 's' || choice[0] == 'n' || choice[0] == 'e' ||
choice[0] == 'w')
{
MESSAGE = "";
switch(choice[0])
{
case 'e' : location = CENTER1; break;
case 'n' : MESSAGE = MESSAGE
+ "\r\nYou are attacked by a vicious "
+ "chicken! You can not pass.\r\n";
if(!CurrentPlayer->getDagger())
{
MESSAGE = MESSAGE
+ "\r\nBut lieing on the ground, "
+ "you find a bronze dagger!\r\n";
CurrentPlayer->setDagger(true);
CurrentPlayer->Inventory();
}
else
{ MESSAGE = MESSAGE
+ "\r\nYou already found the dagger!\r\n";
}
break;
case 's' : if(W1GiantAlive)
{
MESSAGE = MESSAGE
+ "\r\nYou walk towards the Giant and he charges you!";
Giant OneGiant;
location = Combat(&OneGiant, W1);
//If player survives, make sure they don't fight giant
//by setting the global boolean to false
W1GiantAlive = false;
PlaySound(L"media/theme.wav",NULL,SND_FILENAME|SND_ASYNC);
}
else
{
MESSAGE = MESSAGE
+ "\r\nYou stumble over the corpse of a dead giant!\r\n";
}
break;
case 'w' : location = W2; break;
default : MESSAGE = MESSAGE
+ "Invalid choice.\r\n"; break;
}

pOutput->SetWindowText(MESSAGE);
LOCK = false;
return location;
```

```
}

else
{
MESSAGE = "";
LOCK = true;

MESSAGE = MESSAGE
+ "\r\n\r\nYou arrive at an abandoned farm house. You see a "
+ "picket fence, a rustic dilapidated barn, and a decaying "
+ "hovel that used to be someone's residence. There are "
+ "chickens walking around the premises.\r\n\r\nTo the south, you see
";

if(W1GiantAlive)
{
MESSAGE = MESSAGE
+ "a Giant wearing old, brown "
+ "sackcloth. He is taunting you with offensive "
+ "gestures and lewd comments.\r\n\r\nYou really don't want to "
+ "tangle with a giant, do you?";
}

else
{
MESSAGE = MESSAGE
+ "barbed wire, blood, sackcloth and carnage...";
}

MESSAGE = MESSAGE
+ "\r\n\r\n";
MESSAGE = MESSAGE
+ "At this point, you may go NORTH or back EAST. "
+ "You see only thick undergrowth and brush "
+ "to the west.\r\n";

MESSAGE = MESSAGE
+ "\r\nWhere will you go (n,s,e,w)? ";

HBITMAP CenterPix = (HBITMAP)
LoadImage(NULL,L"media/W1.bmp",IMAGE_BITMAP,
0,0,LR_DEFAULTSIZE | LR_LOADFROMFILE);

pView->SetBitmap(CenterPix );

pOutput->SetWindowText(MESSAGE);
location = W1;
return location;

}//close else

}//close function
```

```
//-----------------------------------------------------------------

int NORTH2()
{
MESSAGE = "";

if(choice[0] == 's' || choice[0] == 'n' || choice[0] == 'e' ||
choice[0] == 'w' || choice[0] == 'g')
{
switch(choice[0])
{
case 's' : location = N1; break;
case 'n' : MESSAGE = MESSAGE
+ "The tapestries look very dry and dusty.\r\n"; break;
case 'e' : MESSAGE = MESSAGE
+ "You approach the box and cautiously open it...\r\n";
if(!CurrentPlayer->getFullBodyArmor())
{
MESSAGE = MESSAGE
+ "You find a well preserved suit of full body armor!\r\n";
CurrentPlayer->setFullBodyArmor(true);
CurrentPlayer->Inventory();
}
else
{
MESSAGE = MESSAGE
+ "The box is empty - you already took the armor.\r\n";
}
break;
case 'w' : MESSAGE = MESSAGE
+ "You press against the wall but find nothing.\r\n"; break;
case 'g' : location = UNDERGRND; break;
default : MESSAGE = MESSAGE
+ "That was an invalid choice.\r\n"; break;
}

pOutput->SetWindowText(MESSAGE);
LOCK = false;
return location;

}//close if

else
{
MESSAGE = "";
LOCK = true;

MESSAGE = MESSAGE
+ "\r\nYou walk inside the castle. It is dark and musty, but ";
MESSAGE = MESSAGE
+ "enough daylight is leaking through the cracks in between "
```

```
+ "stones and mortar that you can ascertain your surroundings "
+ "in a dim, colorless twilight. Against the wall to the east "
+ "you see a long wooden box, about the size of a coffin. You "
+ "can see a table, chairs and several candle stands to the west "
+ "of the room.\r\n\r\nTo the north and the south the walls are "
+ " adorned with dusty, thread-bare tapestries.\r\n\r\nYou notice"
+ " stairs descending deep underground to some unknown passage to"
+ "your right. ";

MESSAGE = MESSAGE + "\r\n\r\nChoices: (n,s,e,w) or (g)o underground:
";

HBITMAP CenterPix = (HBITMAP)
LoadImage(NULL,L"media/N2.bmp",IMAGE_BITMAP,
0,0,LR_DEFAULTSIZE | LR_LOADFROMFILE);

pView->SetBitmap(CenterPix );

pOutput->SetWindowText(MESSAGE);
location = N2;
return location;

}//close else

}//close function

//-----------------------------------------------------------------

int SOUTH2()
{
if(choice[0] == 's' || choice[0] == 'n' || choice[0] == 'e' ||
choice[0] == 'w')
{
MESSAGE = "";

switch(choice[0])
{
case 's' : location = GIANTCAMP; break;
case 'n' : location = S1; break;
case 'e' : MESSAGE = MESSAGE
+ "You see a lake, covered with lily pads and algae.\n";
break;
case 'w' : MESSAGE = MESSAGE
+ "You see cat tails and dragon flies skimming across"
+ " the water.\n"; break;
default : MESSAGE = MESSAGE
+ "That was an invalid choice.\r\n"; break;
}

pOutput->SetWindowText(MESSAGE);
LOCK = false;
return location;
```

```
}//close if

else
{
MESSAGE = "";
LOCK = true;

MESSAGE = MESSAGE
+ "\r\nYou wander through the village further to the south. ";
MESSAGE = MESSAGE
+ "You notice several of the villagers are staring at you "
+ "strangely as you walk by. You come to the southern gate "
+ "that guards the entrance to the village and pass through its "
+ "open doors. You follow a meandering dirt path towards the "
+ "edge of a dense hardwood forest.\r\n\r\nAs you walk along the
road, "
+ "you pass several merchants hauling their wares by horse and "
+ "cart. ";

if(S2MotleyCrewAlive)
{
MESSAGE = MESSAGE
+ "Continuing south, you see a group of three giants "
+ "resting with their backs against the trees.\r\n\r\n";
}

else
{
MESSAGE = MESSAGE
+ "Looking south, you see the destruction and carnage"
+ " of your bloody victory over the giants. Quickened"
+ " by the smell of carrion, vultures circle overhead...\r\n\r\n";
}

MESSAGE = MESSAGE
+ "Where will you go (n,s,e,w)? ";

HBITMAP CenterPix = (HBITMAP)
LoadImage(NULL,L"media/S2.bmp",IMAGE_BITMAP,
0,0,LR_DEFAULTSIZE | LR_LOADFROMFILE);

pView->SetBitmap(CenterPix );

pOutput->SetWindowText(MESSAGE);
location = S2;
return location;

}//close else

}//close function
```

```
//-------------------------------------------------------------------

int EAST2()
{
if(choice[0] == 's' || choice[0] == 'n' || choice[0] == 'e' ||
choice[0] == 'w')
{
MESSAGE = "";

switch(choice[0])
{
case 's' : MESSAGE = MESSAGE
+ "\r\nYou see dolphins and sharks.\r\n";
if(!CurrentPlayer->getFishKey())
{
MESSAGE = MESSAGE
+ "\r\nYou notice something metalic shining in the sand "
+ "beneath your feet. You dig into the sand and find "
+ "a bronze key with a Fish engraved upon it.\r\n\r\n";
CurrentPlayer->setFishKey(true);
}
else
{
MESSAGE = MESSAGE
+ "\r\nHey, this is the same place"
+ "you found that Fish key!\r\n";
}
break;
case 'n' : MESSAGE = MESSAGE
+ "\r\nYou see dolphins and sharks.\r\n"; break;
case 'e' : MESSAGE = MESSAGE
+ "\r\nYou see dolphins and sharks.\r\n"; break;
case 'w' : location = E1; break;
default : MESSAGE = MESSAGE
+ "\r\nThat was an invalid choice.\r\n"; break;
}//close switch

pOutput->SetWindowText(MESSAGE);
LOCK = false;
return location;

}//end if

else
{
MESSAGE = "";
LOCK = true;

MESSAGE = MESSAGE
+ "\r\nYou swim out to the sand bars hundreds of feet beyond the
shore. "
+ "Treading water until you are almost exhausted, you notice that the
```

```
"
+ "water growing more and more shallow until at last, gasping for "
+ "breath, you find that your feet can touch the bottom. You continue
"
+ "advancing up a steep slope, this time walking instead of "
+ "swimming, until the water is only ankle deep. You gather your "
+ "thoughts together and take in your surroundings...\r\n\r\n"
+ "In every direction, you see dolphins and sharks swimming around"
+ " you.\r\n\r\n";

MESSAGE = MESSAGE
+ "Where will you go (n,s,e,w)? ";

HBITMAP CenterPix = (HBITMAP)
LoadImage(NULL,L"media/E2.bmp",IMAGE_BITMAP,
0,0,LR_DEFAULTSIZE | LR_LOADFROMFILE);

pView->SetBitmap(CenterPix );

pOutput->SetWindowText(MESSAGE);
location = E2;
return location;

}//close else

}//close function

//---------------------------------------------------------------

int WEST2()
{
if(choice[0] == 's' || choice[0] == 'n' || choice[0] == 'e' ||
choice[0] == 'w')
{
MESSAGE = "";

switch(choice[0])
{
case 's' : MESSAGE = MESSAGE
+ "You see... WHEAT!\r\n";
if(!FoundHP_West2)
{
MESSAGE = MESSAGE
+ "\r\nBuried under a mound among the wheat, you "
+ "find a healing potion!\r\n";
CurrentPlayer->setHealingPotion(1);
FoundHP_West2 = true;
CurrentPlayer->Inventory();
}
else
{
MESSAGE = MESSAGE
```

```
+ "\r\nYou already found the healing potion!\r\n";
}
break;
case 'n' : if(!CurrentPlayer->getLongBow())
{
MESSAGE = MESSAGE
+ "\r\nYou find a well-stringed long bow and arrows!\r\n";
CurrentPlayer->setLongBow(true);
CurrentPlayer->Inventory();
}
else
{
MESSAGE = MESSAGE
+ "\r\nYou already found the long bow.\r\n";
}
break;
case 'e' : location = W1; break;
case 'w' : MESSAGE = MESSAGE
+ "\r\nYou see various feed crops planted in rows.\r\n"; break;
default : MESSAGE = MESSAGE
+ "That was an invalid choice.\r\n"; break;
}//close switch

pOutput->SetWindowText(MESSAGE);
LOCK = false;
return location;

}//close if

else
{
MESSAGE = "";
LOCK = true;

MESSAGE = MESSAGE
+ "\r\nYou find yourself walking in golden fields of billowing "
+ "wheat. The wind is blowing briskly through the long, tall "
+ "stalks as they rustle in the autumn air. Particles of chaff "
+ "float through the air all around you and collect upon "
+ "your clothing.\r\n\r\n";

MESSAGE = MESSAGE
+ "Where will you go (n,s,e,w)? ";

HBITMAP CenterPix = (HBITMAP)
LoadImage(NULL,L"media/W2.bmp",IMAGE_BITMAP,
0,0,LR_DEFAULTSIZE | LR_LOADFROMFILE);

pView->SetBitmap(CenterPix );

pOutput->SetWindowText(MESSAGE);
location = W2;
```

```
return location;

}//close else

}//close function
//-----------------------------------------------------------------

int SHAMANHUT()
{
if(choice[0] == 'n' || choice[0] == 'l' || choice[0] == 's' ||
choice[0] == 't')
{
MESSAGE = "";
switch(choice[0])
{
case 'n' : MESSAGE = MESSAGE
+ "You run into a straw-mud wall.\n"; break;
case 'l' : location = S1; break;
case 's' : LockButtons(true); pHEAL->EnableWindow(false);
AFunctionPointer1();
MESSAGE = MESSAGE
+ "\r\n\r\nYou get the uneasy feeling that you are going "
+ "to reap serious bad karma for this unwise action.\r\n";
pOutput->SetWindowText(MESSAGE);

XSleep(5000);

MESSAGE = MESSAGE
+ "\r\nBellowing thunder cracks and the clouds darken the "
+ " skies outside as the deity of the temple preistess fills"
+ " with indignation and anger!\r\n";
pOutput->SetWindowText(MESSAGE);

XSleep(5000);

MESSAGE = MESSAGE
+ "\r\nIn an instant, lighting from the heavens strikes you "
+ "down! All that remains of you are burning embers...\r\n";

PlaySound(L"media/thunder.wav",NULL,SND_FILENAME|SND_ASYNC);
pOutput->SetWindowText(MESSAGE);
CurrentPlayer->setHit(0);
CurrentPlayer->DisplayStats();

XSleep(6000);

MESSAGE = MESSAGE
+ "\r\nYou go into the afterlife a loser, ashamed "
+ "for the despicable deeds you have done. The warriors "
+ "who have gone on before you, the great warriors of "
+ "reknown and the kings of the past will ridicule you "
+ "for all of eternity for dieing without honor.\r\n";
```

```
pOutput->SetWindowText(MESSAGE);

XSleep(13000);

LockButtons(false);
location = GAMEOVER;
break;
case 't' : MESSAGE = MESSAGE
+ "You seek audience with the preistess.\r\n";
WiseWoman->Talk();
break;
default : MESSAGE = MESSAGE
+ "That was an invalid choice.\r\n";
break;
}//close switch

pOutput->SetWindowText(MESSAGE);
LOCK = false;
return location;

}//close if

else
{
MESSAGE = "";
LOCK = true;

if(FirstTimeInShamanHut)
{
MESSAGE = MESSAGE
+ "\r\nYou duck down and enter into the Shaman's hut. Towards the
center "
+ "of the mud dwelling, beneath an overhanging shelf descending from
the "
+ "thatched roof, sits an elderly priestess. Unphased by your
presence, "
+ "she continues to stare into the flames of a small fire burning
within "
+ "a set of blackened stone rings in the center of the hut. Directly
over "
+ "her head, an opening in the ceiling allows the smoke to
escape.\r\n"
+ "\r\nShe gazes at you and cackles. \"Not expecting an old temple
preistess, "
+ "were you, a woman, eh? Well, in this village, I'm the
\"Shaman\".\r\n\r\n";

FirstTimeInShamanHut = false;
}

else
```

```
{
MESSAGE = MESSAGE
+ "\r\n\r\nYou re-enter the Shaman's hut. She turns her head in a
peculiar "
+ "fashion and remarks, \"Back so soon, traveler?\" She offers you a
cup "
+ "of freshly brewed tea, which you gladly accept to quench your
thirst. ";

if(!FoundHP_Shaman)
{
MESSAGE = MESSAGE
+ "She opens her medicine bag and begins creating an acrid mixture of
"
+ "herbs. She pours it into a vial and places it in your hand, saying
"
+ "\"Drink this if you become wounded, my friend. It may restore "
+ "you to a measure of your former health and
constitution.\"\r\n\r\n";
CurrentPlayer->setHealingPotion(1);
FoundHP_Shaman = true;
CurrentPlayer->Inventory();
}

else
{
MESSAGE = MESSAGE
+ "You feel a sense of debt and gratitude towards this "
+ "kind old woman, remembering the healing elixir she gave "
+ "you on your last visit.\r\n\r\n";
}
}//close else

MESSAGE = MESSAGE
+ "At this point, you may (l)eave the Shaman's hut, "
+ "(t)alk with her if you so desire, or try to (s)teal "
+ "her medicine bag and staff for what wonders they may "
+ "contain.\r\n";

MESSAGE = MESSAGE
+ "\r\n\r\nYou may: (l)eave (t)alk or (s)teal things ";

HBITMAP CenterPix = (HBITMAP)
LoadImage(NULL,L"media/SHAMAN.bmp",IMAGE_BITMAP,
0,0,LR_DEFAULTSIZE | LR_LOADFROMFILE);

pView->SetBitmap(CenterPix );

pOutput->SetWindowText(MESSAGE);
location = SHAMAN;
return location;
```

```
}//close else

}//close function

//---------------------------------------------------------------------

int GateWay()
{
char NumberBuffer[10];

if(choice[0] == 'g')
{
pOutput->SetWindowText(MESSAGE);
LOCK = false;
return location;
}//end if

else
{
MESSAGE = "";
LOCK = true;

if(CurrentPlayer->getFishKey())
{
MESSAGE = MESSAGE
+ "\r\n\r\nFumbling around the gate, you find a slot to insert the "
+ "Fish key. The tablets and columns begin to rumble and shake. Large
"
+ "stones rise, levitating off the ground, rearranging
themselves.\r\n";
}

else
{
MESSAGE = MESSAGE
+ "\r\nYou look around, trying every nook and crevice, but can not "
+ "seem to find the means to open the gate, nor alter anything else "
+ "around it. You see what appears to be a key hole to one side.
\r\n";
location = CENTER1;
}

if(CurrentPlayer->getFishKey() && CurrentPlayer->getScore() < 50)
{
MESSAGE = MESSAGE
+ "\r\nIt appears that, although you have the key, you lack enough "
+ "experience with the ways of this world to cause the gate to "
+ "function in any useful manner.\r\n";
location = CENTER1;
}

if(CurrentPlayer->getFishKey() && CurrentPlayer->getScore() >= 50)
```

```
{
MESSAGE = MESSAGE
+ "\r\nWith the experience you have gained since entering this "
+ "strange world, you manage to figure out the correct sequence of "
+ "actions to perform while turning the Fish key within the gate. "
+ "You hear a loud hiss followed by a dull hum as cascading beams of
"
+ "light blind you from the opening dimensional portal.\r\n\r\n";

MESSAGE = MESSAGE
+ "You feel as though you have won a series of battles in a "
+ "long campaign, but that the war is far from being over. "
+ "Having made several new friends and vanquished many foes as "
+ "a soujourner in a strange land, you step through the gates, "
+ "uncertain yet hopeful that this may bring you one step closer "
+ "to home...";

MESSAGE = MESSAGE
+ "You win this campaign and end the game with:\r\n\r\n";

CurrentPlayer->DisplayStats();
CurrentPlayer->Inventory();

MESSAGE = MESSAGE
+ "\nCombat Experience Score: "
+ _itoa(CurrentPlayer->getScore(),NumberBuffer,10)
+ ".\r\n\r\n";

LOCK = false;
location = YOUWIN;
}

MESSAGE = MESSAGE
+ "\r\n\r\nType \"G\" and click \"GO\" to continue...\r\n\r\n";

pInput->SetWindowText(L"g");
pOutput->SetWindowText(MESSAGE);

return location;

}//end else

}//end function

//-----------------------------------------------------------------

int UnderGroundPassage()
{
if(choice[0] == 'r' || choice[0] == 'l')
{
if(choice[0] == 'l') { location = DRAGONFIGHT; }
if(choice[0] == 'r') { location = N2; }
```

```
pOutput->SetWindowText(MESSAGE);
LOCK = false;
return location;
}//end if

else
{
MESSAGE = "";
LOCK = true;

MESSAGE = MESSAGE
+ "\r\n\r\nYou descend into the darkness underground...\r\n\r\n";

if(UNDERDragonPairAlive)
{
MESSAGE = MESSAGE
+ "Peeking around a corner, you see a large subterranean cavern. ";
MESSAGE = MESSAGE
+ "Scattered around its rocky walls are piles of bones from "
+ "both animals and men.\r\n";

MESSAGE = MESSAGE
+ "\r\n\r\nDo you want to (r)eturn to the castle above or "
+ "(l)ook for what created this macabre chamber of remains?\r\n\r\n";
}

else
{
MESSAGE = MESSAGE
+ "\r\nYou see a family of dead red dragons and the carnal "
+ "aftermath of your last great battle with these fierce and"
+ " worthy opponents.\r\n\r\nEnter \"r\" and click \"GO\" to"
+ " return up the stairs to the castle.";

pInput->SetWindowText(L"r");
}

pOutput->SetWindowText(MESSAGE);
return location;

}//close else

}//close function

//-----------------------------------------------------------------
------

int DragonFight()
{
char NumberBuffer[10];
```

```
if(choice[0] == 'y' || choice[0] == 'n')
{
if(choice[0] == 'y')
{
Dragon * DragonFamily = new Dragon[5];

for(int x = 0; x < 5 && CurrentPlayer->getHit() > 0; x++)
{
MESSAGE = "";

MESSAGE = MESSAGE
+ "Dragon " + _itoa((x+1),NumberBuffer,10) + " of 5 attacks you!";

pOutput->SetWindowText(MESSAGE);

XSleep(3000);

location = Combat(&DragonFamily[x], UNDERGRND);
}

//Clean up DragonFamily objects on heap, set pointer to NULL
delete [] DragonFamily;
DragonFamily = 0;
UNDERDragonPairAlive = false;

MESSAGE = MESSAGE
+ "\r\n\r\nType \"G\" and click \"GO\" to continue...\r\n\r\n";

pInput->SetWindowText(L"g");

pOutput->SetWindowText(MESSAGE);

PlaySound(L"media/theme.wav",NULL,SND_FILENAME|SND_ASYNC);
}

else
{
MESSAGE = MESSAGE
+ "\r\nIntelligently, you decide to run back up the stairs...\r\n";
location = N2;
}

pOutput->SetWindowText(MESSAGE);
LOCK = false;
return location;

}//close if

else
{
LOCK = true;
MESSAGE = MESSAGE
```

```
+ "\r\nYou see a family of 5 dragons. Stop and think about this a
minute. "
+ "Are you sure you want to take on 5 dragons simultaneously? (y,n)";

AFunctionPointer2();
pOutput->SetWindowText(MESSAGE);
return location;

}//close else

}//close function

//-----------------------------------------------------------------
------

int GiantFight()
{
char NumberBuffer[10];

if(choice[0] == 'y' || choice[0] == 'n')
{
if(choice[0] == 'y')
{
Giant * MotleyCrew = new Giant[3];
MESSAGE = MESSAGE
+ "All three giants charge you at once!\r\n\r\n";
for(int x = 0; x < 3 && CurrentPlayer->getHit() > 0; x++)
{
MESSAGE = "";
MESSAGE = MESSAGE
+ "Giant " + _itoa((x+1),NumberBuffer,10) + " of 3 attacks you!";

pOutput->SetWindowText(MESSAGE);

XSleep(3000);

location = Combat(&MotleyCrew[x], S2);
}
//Clean up after MotleyCrew - no dangling pointers
delete [] MotleyCrew; MotleyCrew = 0;
S2MotleyCrewAlive = false;

MESSAGE = MESSAGE
+ "\r\n\r\nType \"G\" and click \"GO\" to continue...\r\n\r\n";

pInput->SetWindowText(L"g");

pOutput->SetWindowText(MESSAGE);
PlaySound(L"media/theme.wav",NULL,SND_FILENAME|SND_ASYNC);
}

else
```

```
{
MESSAGE = MESSAGE
+ "\r\nIntelligently, you decide to walk away...\r\n";
location = S2;
}

pOutput->SetWindowText(MESSAGE);
LOCK = false;
return location;

}//close if

else
{
LOCK = true;
MESSAGE = MESSAGE
+ "Stop and think a moment. Are you really prepared "
+ "to fight 3 giants at the same time? (y,n)";

pOutput->SetWindowText(MESSAGE);
return location;
}

}//close function

//-----------------------------------------------------------------------
------

int GiantCamp()
{
if(choice[0] == 'r' || choice[0] == 'l')
{
if(choice[0] == 'l' && S2MotleyCrewAlive) { location = GIANTFIGHT; }
if(choice[0] == 'r') { location = S2; }

pOutput->SetWindowText(MESSAGE);
LOCK = false;
return location;
}//end if

else
{
LOCK = true;
location = GIANTCAMP;
MESSAGE = "";

MESSAGE = MESSAGE
+ "\r\nYou approach the giant's camp at the edge "
+ "of the forest.\r\n\r\n";

if(S2MotleyCrewAlive)
{
```

```
MESSAGE = MESSAGE
+ "Through a clearing among the trees, you see ";
MESSAGE = MESSAGE
+ "3 giants dining upon what can only be human remains.\r\n";
}
else
{
MESSAGE = MESSAGE
+ "You stumble over the corpses of three dead giants.\r\n";
}

if(S2MotleyCrewAlive)
{
MESSAGE = MESSAGE
+ "\r\n\r\nDo you want to (r)eturn to the forest edge or "
+ "(l)ook for giants?\r\n\r\n";
}
else
{
MESSAGE = MESSAGE
+ "\r\n\r\nType \"r\" and click \"GO\" or click"
+ " on any direction button (N, S, E, W) to RETURN."
+ "\r\n\r\nThere is nothing here but death!\r\n\r\n";

pInput->SetWindowText(L"r");
}

pOutput->SetWindowText(MESSAGE);
return location;

}//close else

}//close function

//-------------------------------------------------------------------
---

int Randy(int n)
{
int ResultRandom;
ResultRandom = (rand()%n) + 1;
return ResultRandom;
}
//-------------------------------------------------------------------
---
int GameOver()
{
if(choice[0] == 'g')
{
if(StartedGame) { delete CurrentPlayer; CurrentPlayer = 0; }
StartedGame = false;
pSTART->EnableWindow(true);
```

```
pOutput->SetWindowText(MESSAGE);
LOCK = false;
Continue = false;
return location;
}//close if

else
{
MESSAGE = "";
LOCK = true;
pGO->EnableWindow(true);

MESSAGE = MESSAGE
+ "\r\n\r\nGame Over...\r\n\r\nEnter \"g\" and then click"
+ " \"GO\".\r\n\r\nYou may then press \"QUIT\" to exit the"
+ " game or \"START\" to begin a new game.";

PlaySound(L"media/mighty3.wav",NULL,SND_FILENAME|SND_ASYNC);

pInput->SetWindowText(L"g");

pOutput->SetWindowText(MESSAGE);
location = GAMEOVER;
return location;

}//close else

}//close function

//-------------------------------------------------------------------
---
//Overloaded Combat Functions, one for Dragon and one for Giant.
//Takes the global enumerated constant "EVENTS" as one of its
arguments.
//-------------------------------------------------------------------
---
int Combat(Dragon * m, EVENTS CurrentLocation)
{
char NumberBuffer[10];
LOCK = true;
LockButtons(true);

MESSAGE = MESSAGE
+ "\r\nMortal Combat!!!\r\n";

HBITMAP GiantPix = (HBITMAP)
LoadImage(NULL,L"media/dragon.bmp",IMAGE_BITMAP,
0,0,LR_DEFAULTSIZE | LR_LOADFROMFILE);
pView->SetBitmap(GiantPix);

while(CurrentPlayer->getHit() > 0 && m->getHit() > 0)
{
```

```
MESSAGE = "";

if(CurrentPlayer->getHit() > 0)
{
CurrentPlayer->Attack(m);
XSleep(3000);
}

if(m->getHit() > 0)
{
PlaySound(L"media/dragon.wav",NULL,SND_FILENAME|SND_ASYNC);
m->Attack(CurrentPlayer);
XSleep(3500);
}

CurrentPlayer->DisplayStats();
}

if(CurrentPlayer->getHit() <= 0)
{
MESSAGE = MESSAGE
+ "\r\n\r\nYou die, most tragically!\r\n";
MESSAGE = MESSAGE
+ "The Dragon wins the battle, having "
+ _itoa(m->getHit(),NumberBuffer,10) + " hitpoints left.\r\n";

LOCK = false;
LockButtons(false);
return GAMEOVER;
}

else
{
MESSAGE = MESSAGE
+ "\r\nYou vanquish your foe most valiantly!\r\n";
MESSAGE = MESSAGE
+ "The Dragon died, now having " + _itoa(m->getHit(),NumberBuffer,10)
+ " hitpoints.\r\n" + CurrentPlayer->getName() + " has "
+ _itoa(CurrentPlayer->getHit(),NumberBuffer,10)
+ " hitpoints left.\r\n\r\n";

//Every three opponents, increase player's attack and defense for
experience
if( (ConqueredGiants + ConqueredDragons) == 3 ||
(ConqueredGiants + ConqueredDragons) == 6 ||
(ConqueredGiants + ConqueredDragons) == 9 ||
(ConqueredGiants + ConqueredDragons) == 11)
{
MESSAGE = MESSAGE
+ "Add 1 to your attack and 1 to your defense as a result of combat"
+ " experience acquired defeating your last three opponents.\r\n";
```

```
CurrentPlayer->setAttack((CurrentPlayer->getAttack() + 1));
CurrentPlayer->setDefense((CurrentPlayer->getDefense() + 1));
}

CurrentPlayer->setScore((CurrentPlayer->getScore() + 10));

MESSAGE = MESSAGE
+ "\r\n" + CurrentPlayer->getName() + "'s Current Score: "
+ _itoa(CurrentPlayer->getScore(),NumberBuffer,10)
+ ".\r\n";

CurrentPlayer->DisplayStats();
ConqueredDragons++;
Conquests();

MESSAGE = MESSAGE
+ "\r\n";

pOutput->SetWindowText(MESSAGE);
LOCK = false;
LockButtons(false);
return CurrentLocation;

}//close else

}//close function

//-----------------------------------------------------------------

int Combat(Giant * m, EVENTS CurrentLocation)
{
char NumberBuffer[10];
LOCK = true;
LockButtons(true);
MESSAGE = "";

MESSAGE = MESSAGE + "\r\nMortal Combat!!!\r\n";

HBITMAP GiantPix = (HBITMAP)
LoadImage(NULL,L"media/giant.bmp",IMAGE_BITMAP,
0,0,LR_DEFAULTSIZE | LR_LOADFROMFILE);
pView->SetBitmap(GiantPix);

while(CurrentPlayer->getHit() > 0 && m->getHit() > 0)
{
MESSAGE = "";

if(CurrentPlayer->getHit() > 0)
{
CurrentPlayer->Attack(m);
XSleep(3000);
}
```

```
if(m->getHit() > 0)
{
PlaySound(L"media/dragon.wav",NULL,SND_FILENAME|SND_ASYNC);
m->Attack(CurrentPlayer);
XSleep(3500);
}

CurrentPlayer->DisplayStats();
Conquests();

}

if(CurrentPlayer->getHit() <= 0)
{
MESSAGE = MESSAGE + "\r\n\r\nYou die, most tragically!\r\n";
MESSAGE = MESSAGE + "The Giant won the battle, having "
+ _itoa(m->getHit(),NumberBuffer,10)
+ " hitpoints left.\r\n";

LOCK = false;
LockButtons(false);
return GAMEOVER;
}

else
{
MESSAGE = MESSAGE + "\r\nYou vanquish your foe most valiantly!\r\n";

MESSAGE = MESSAGE + "The Giant died, now having "
+ _itoa(m->getHit(),NumberBuffer,10);

MESSAGE = MESSAGE + " hitpoints.\r\n" + CurrentPlayer->getName()
+ " has " + _itoa(CurrentPlayer->getHit(),NumberBuffer,10)
+ " hitpoints left.\r\n";

//Every three opponents, increase player's attack and defense for
experience
if( (ConqueredGiants + ConqueredDragons) == 3 ||
(ConqueredGiants + ConqueredDragons) == 6 ||
(ConqueredGiants + ConqueredDragons) == 9 ||
(ConqueredGiants + ConqueredDragons) == 11)
{
MESSAGE = MESSAGE
+ "Add 1 to your attack and 1 to your defense as a result of combat"
+ " experience acquired defeating your last three opponents.\r\n";

CurrentPlayer->setAttack((CurrentPlayer->getAttack() + 1));
CurrentPlayer->setDefense((CurrentPlayer->getDefense() + 1));
}

CurrentPlayer->setScore((CurrentPlayer->getScore() + 10));
```

```
MESSAGE = MESSAGE
+ "\r\n" + CurrentPlayer->getName() + "'s Current Score: "
+ _itoa(CurrentPlayer->getScore(),NumberBuffer,10)
+ ".\r\n";

CurrentPlayer->DisplayStats();
ConqueredGiants++;
Conquests();

LOCK = false;
LockButtons(false);

return CurrentLocation;

}//close else

}//close function

//-----------------------------------------------------------------


bool SaveCharacter()
{
CString CharacterName;
CString password;

pSaveName->GetWindowText(CharacterName);
CharacterName = CharacterName + ".gam";

pSavePassword->GetWindowText(password);

ofstream WriteStuff;
WriteStuff.open(CharacterName);

if(!WriteStuff)
{ return false; } //close if

else
{
//NOTE: We need to convert the CStrings to strings to write the data.
char PASSWORD[100] = " ";
for(int y = 0; y < password.GetLength(); y++)
{ PASSWORD[y] = password[y]; }

CString CHARACTERNAME = CurrentPlayer->getName();
char CHARNAME[100] = " ";
for(int z = 0; z < CHARACTERNAME.GetLength(); z++)
{ CHARNAME[z] = CHARACTERNAME[z]; }

//Simple serialization of Character class
WriteStuff << PASSWORD << "\n";
```

```
WriteStuff << CHARNAME << "\n";
WriteStuff << CurrentPlayer->getHit() << "\n";
WriteStuff << CurrentPlayer->getAttack() << "\n";
WriteStuff << CurrentPlayer->getDefense() << "\n";
WriteStuff << CurrentPlayer->getLevel() << "\n";
WriteStuff << CurrentPlayer->getScore() << "\n";
WriteStuff << CurrentPlayer->getLocation() << "\n";

WriteStuff << CurrentPlayer->getDagger() << "\n";
WriteStuff << CurrentPlayer->getSword() << "\n";
WriteStuff << CurrentPlayer->getLongBow() << "\n";
WriteStuff << CurrentPlayer->getChainMail() << "\n";
WriteStuff << CurrentPlayer->getFullBodyArmor() << "\n";
WriteStuff << CurrentPlayer->getHealingPotion() << "\n";
WriteStuff << CurrentPlayer->getFishKey() << "\n";

WriteStuff << W1GiantAlive << "\n";
WriteStuff << E1DragonAlive << "\n";
WriteStuff << S2MotleyCrewAlive << "\n";
WriteStuff << FirstTimeInShamanHut << "\n";
WriteStuff << CENTERFirstTime << "\n";
WriteStuff << UNDERDragonPairAlive << "\n";
WriteStuff << FoundHP_West2 << "\n";
WriteStuff << FoundHP_Shaman << "\n";
WriteStuff << ConqueredDragons << "\n";
WriteStuff << ConqueredGiants << "\n";

WriteStuff.close();

return true;

}//close else

}//close function

//-------------------------------------------------------------------------
------

bool LoadCharacter(CString & problem)
{
string passwd;
string nm;
int hp, atk, def, lvl, scr, loc;
CString CharacterName;
CString pass;
bool dagger, sword, bow, mail, armor, healing, fkey;

pLoadName->GetWindowText(CharacterName);
CharacterName = CharacterName + ".gam";

ifstream ReadStuff;
ReadStuff.open(CharacterName);
```

```
//Detect if successful or not to keep program from crashing on failed
load
if(!ReadStuff)
{
problem = "Can not find the file you typed in. Try a different
name?";
return false;
}

else
{
pLoadPassword->GetWindowText(pass);
getline(ReadStuff, passwd);

//Need to convert the string to CString with .c_str()
if(pass == passwd.c_str())
{
//Careful! serialization = you must read in exactly the same order as
you wrote!
//Have to use getline for nm since name may have spaces in it
getline(ReadStuff, nm); ReadStuff >> hp;
ReadStuff >> atk; ReadStuff >> def;
ReadStuff >> lvl; ReadStuff >> scr;
ReadStuff >> loc; location = loc;

ReadStuff >> dagger;
ReadStuff >> sword; ReadStuff >> bow;
ReadStuff >> mail; ReadStuff >> armor;
ReadStuff >> healing; ReadStuff >> fkey;

//Note: In VS 6.0 and .Net Visul Studio, you can read these as
booleans
//with no problem. The problem is with Bloodshed - they will all read
in
//as false. To make it compatible with Bloodshed (therefore all 3
//compilers) you must read them as integers and they will be cast to
bools
//in VS 6.0 and .Net Visual Studio. (It ain't easy getting these 3 to
agree).
bool giant, dragon, motley, shaman, center, under, HPwest2, HPshaman;
int CD, CG;

ReadStuff >> giant; W1GiantAlive = giant;
ReadStuff >> dragon; E1DragonAlive = dragon;
ReadStuff >> motley; S2MotleyCrewAlive = motley;
ReadStuff >> shaman; FirstTimeInShamanHut = shaman;
ReadStuff >> center; CENTERFirstTime = center;
ReadStuff >> under; UNDERDragonPairAlive = under;
ReadStuff >> HPwest2; FoundHP_West2 = HPwest2;
ReadStuff >> HPshaman; FoundHP_Shaman = HPshaman;
ReadStuff >> CD; ConqueredDragons = CD;
```

```
ReadStuff >> CG; ConqueredGiants = CG;

//Example: Memberwise copy of string to character array for CString
//char x[25] = "Weeeeee";
//for(int a = 0; a < nm.length(); a++) { x[a] = nm[a]; }
//OR just use .c_str()

CurrentPlayer->setName(nm.c_str());
CurrentPlayer->setHit(hp);
CurrentPlayer->setAttack(atk);
CurrentPlayer->setDefense(def);
CurrentPlayer->setLevel(lvl);
CurrentPlayer->setScore(scr);
CurrentPlayer->setDagger(dagger);
CurrentPlayer->setSword(sword);
CurrentPlayer->setLongBow(bow);
CurrentPlayer->setChainMail(mail);
CurrentPlayer->setFullBodyArmor(armor);
CurrentPlayer->setHealingPotion(healing);
CurrentPlayer->setFishKey(fkey);

ReadStuff.close();
CurrentPlayer->Inventory();
CurrentPlayer->DisplayStats();
NeedName = false;
Conquests();
pName->SetWindowText(CurrentPlayer->getName());
PlaySound(L"media/theme.wav",NULL,SND_FILENAME|SND_ASYNC);

return true;

}//close if

else
{
problem = "An incorrect password was entered. Re-enter it please.";
return false;
}

}//close else

}//close function

//-----------------------------------------------------------------------
------

bool SaveHighScores()
{
bool successful;

//Open highscores if is exists and count the # scores
ofstream highscores("highscores.txt", ios::app);
```

```
CString CHARACTERNAME = CurrentPlayer->getName();
char CHARNAME[100] = " ";
for(int z = 0; z < CHARACTERNAME.GetLength(); z++)
{ CHARNAME[z] = CHARACTERNAME[z]; }

highscores << CHARNAME << "\n";
highscores << CurrentPlayer->getScore() << "\n";

highscores.close();

return successful;

}//close function

//-----------------------------------------------------------------------
------

void DisplayHighScores()
{
char NumberBuffer[10];
MESSAGE = "";
pOutput->SetWindowText(MESSAGE);

string HoldMeString; int HoldMeInt, z, x = 0;

ifstream highscores("highscores.txt");

if (!highscores)
{
MESSAGE = MESSAGE + "Unable to find \"highscores.txt\" for
reading.\r\n";
pOutput->SetWindowText(MESSAGE);
}

else
{ //Note: When using getline() in a for loop, you must use getline()
again
//right after reading the integer in to consume the '\n' left in the
//data stream. We don't have to have this extra getline() in the
//LoadCharacter() function because getline() is not in a loop there.
while(!highscores.eof())
{
getline(highscores, HoldMeString);
highscores >> HoldMeInt;
getline(highscores, HoldMeString);
x++; //add one for every 2 lines (name and score pair)
}

x = x - 1; //Subtract 1 for the offset (one too many)

//Declare 2 Parallel Arrays where # elements = # lines. We could
```

```
// have also used only 1 array here with a structure containing
//a string and an integer component as a single object. Example:

//NameAndScore { string n; int s; }; //defines structure, sort of
like a class
//NameAndScore HIGHSCORE[x]; //creates array of objects defined as
"NameAndScore"
//if(HIGHSCORE[r].s > HIGHSCORE[r - 1].s)
//{ TempHIGHSCORE = HIGHSCORE[r]; HIGHSCORE[r] = HIGHSCORE[r - 1];

//Note: If compiling this with Visual Studio .Net, it will flag the
array
//declarations below as an error. It does not allow dynamically
sizing them
//at run time as BloodShed does. So just change the 2 lines below:
//from string NAMES[x]; to string NAMES[100];
//from int SCORES[x]; to int SCORES[100];
string NAMES[100];
int SCORES[100];

//Reset stream and move pointer back to beginning of file
highscores.clear();
highscores.seekg(0, ios::beg);

//Put each line into
for(z = 0; z < x; z++)
{
//Remember, need to use getline in case of spaces in name
getline(highscores, NAMES[z]);
highscores >> SCORES[z];
getline(highscores, HoldMeString);
}

highscores.close();

//Could also use (sizeof SCORES /sizeof *SCORES) to get array size

//Bubble Sort for High Scores. Go through each Name and Score set
for(int q = 0; q < x; q++)
{
//Compare the integer component to every other, move it if >
//For descendind order, r starts at 1 to compare it to element 0
//0 = 100
//1 = 75
//2 = 50
for(int r = 1; r < x; r++)
{
if(SCORES[r] > SCORES[r - 1])
{
HoldMeInt = SCORES[r]; HoldMeString = NAMES[r];
SCORES[r] = SCORES[r - 1]; NAMES[r] = NAMES[r - 1];
SCORES[r - 1] = HoldMeInt; NAMES[r - 1] = HoldMeString;
```

```
}
}
}

MESSAGE = MESSAGE
+ " *************** High Scores **************\r\n";
MESSAGE = MESSAGE
+ " ------------------------------------------"
+ "-----------------\r\n";

for(z = 0; z < x; z++)
{
MESSAGE = MESSAGE
+ " "
+ _itoa((z+1),NumberBuffer,10)
+ ". Name: " + NAMES[z].c_str()
+ " " + "Score: " + _itoa(SCORES[z],NumberBuffer,10)
+ "\r\n";

MESSAGE = MESSAGE
+ " ------------------------------------------"
+ "-----------------\r\n";
}

MESSAGE = MESSAGE
+ " ****************************************\r\n";
MESSAGE = MESSAGE
+ "\r\n";
MESSAGE = MESSAGE
+ " Click HighScores again to return to the game and\r\n"
+ " continue playing.";

pOutput->SetWindowText(MESSAGE);

}//close else

}//close function

//-------------------------------------------------------------------------
------
void Conquests()
{
char NumBuffer[10];
CONQUESTS = "";

if(ConqueredGiants == 0 && ConqueredDragons == 0)
{
CONQUESTS = CONQUESTS
+ "You're special. You haven't fought anything at all. "
+ "Beware! Treacherous paths unwind before you...";
}
```

```
if(ConqueredGiants > 0)
{
if(ConqueredGiants == 1)
{
CONQUESTS = CONQUESTS
+ "You have defeated a single, lone Giant.\r\n";
}
else
{
CONQUESTS = CONQUESTS
+ "You have defeated " + _itoa(ConqueredGiants,NumBuffer,10)
+ " Giants in battle.\r\n";
}
}


if(ConqueredDragons > 0)
{
if(ConqueredDragons == 1)
{
CONQUESTS = CONQUESTS
+ "You have defeated a single, lone Dragon.\r\n";
}
else
{
CONQUESTS = CONQUESTS
+ "You have defeated " + _itoa(ConqueredDragons,NumBuffer,10)
+ " Dragons in battle.\r\n";
}
}

if(ConqueredDragons == 0)
{
CONQUESTS = CONQUESTS
+ "You have not battled any Dragons yet.\r\n";
}


if(ConqueredGiants == 0)
{
CONQUESTS = CONQUESTS
+ "You have not battled any Giants yet.\r\n";
}

pConquests->SetWindowText(CONQUESTS);

}//close function
//-------------------------------------------------------------------
------

int YouWin()
{
```

```
if(choice[0] == 'g')
{
location = QUIT;
pOutput->SetWindowText(MESSAGE);
LOCK = false;
return location;
}//close if

else
{
MESSAGE = "";
LOCK = true;

MESSAGE = MESSAGE
+ "\r\n\r\nGood fortune is yours this day, and you stand "
+ "over fallen foes, victorious and triumphant!";
MESSAGE = MESSAGE
+ "\r\n\r\nYou have fought well and finished the game!\r\n\r\n"
+ "Enter \"g\" and click \"GO\"\r\n"
+ "or click \"QUIT\" to exit the program.";

PlaySound(L"media/mighty3.wav",NULL,SND_FILENAME|SND_ASYNC);

pOutput->SetWindowText(MESSAGE);
location = YOUWIN;

return location;

}//close else

}//close function

//-------------------------------------------------------------------------
---

void LockButtons(bool x)
{
//Safety lock all command buttons during combat except
//for the radio buttons being used in combat
if(x == true)
{
pN->EnableWindow(false);
pS->EnableWindow(false);
pE->EnableWindow(false);
pW->EnableWindow(false);
pSCORES->EnableWindow(false);
pSAVE->EnableWindow(false);
pLOAD->EnableWindow(false);
pSTART->EnableWindow(false);
pGO->EnableWindow(false);
pACTION->EnableWindow(false);
}
```

```
else
{
pN->EnableWindow(true);
pS->EnableWindow(true);
pE->EnableWindow(true);
pW->EnableWindow(true);
pSCORES->EnableWindow(true);
pSAVE->EnableWindow(true);
pLOAD->EnableWindow(true);
pSTART->EnableWindow(true);
pGO->EnableWindow(true);
pACTION->EnableWindow(true);
}
}
```

File 4 of 7 "MFCsleep.h":

```
struct XSleep_Structure
{
int duration;
HANDLE eventHandle;
};

DWORD WINAPI XSleepThread(LPVOID pWaitTime)
{
XSleep_Structure *sleep = (XSleep_Structure *)pWaitTime;

Sleep(sleep->duration);
SetEvent(sleep->eventHandle);

return 0;
}

void XSleep(int nWaitInMSecs)
{
XSleep_Structure sleep;

sleep.duration = nWaitInMSecs;
sleep.eventHandle = CreateEvent(NULL, TRUE, FALSE, NULL);

DWORD threadId;
CreateThread(NULL, 0, &XSleepThread, &sleep, 0, &threadId);

MSG msg;

while(::WaitForSingleObject(sleep.eventHandle, 0) == WAIT_TIMEOUT)
{
//get and dispatch messages
if(::PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
```

```
{
::TranslateMessage(&msg);
::DispatchMessage(&msg);
}
}

CloseHandle(sleep.eventHandle);
}
```

File 5 of 7 "RPG_Main.cpp":

```
//-----------------------------------------------------------------------
----------------------
//#include <afxwin.h> //MFC core and standard components
//#include "INTERFACE_MAIN_resources.h" // main symbols
#include "Functions.h"
//-----------------------------------------------------------------------
----------------------

class ShamanMessage : public CDialog
{
public:
ShamanMessage(CWnd* pParent = NULL):
CDialog(ShamanMessage::IDD, pParent) { }

enum{IDD = IDD_SHAMAN};

//Need to create CBrush and use OnCtlColor to change dialog and text
color
CBrush ShamanMessageBrush;

HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
switch (nCtlColor)
{
case CTLCOLOR_EDIT: pDC->SetTextColor(RGB(255,0,0));
case CTLCOLOR_STATIC: pDC->SetTextColor(RGB(255, 0, 0));
case CTLCOLOR_DLG: return ShamanMessageBrush;
default: return CDialog::OnCtlColor(pDC, pWnd, nCtlColor);
}
}

protected:
virtual void DoDataExchange(CDataExchange* pDX)
{CDialog::DoDataExchange(pDX);}

virtual BOOL OnInitDialog()
{
CDialog::OnInitDialog();
ShamanMessageBrush.CreateSolidBrush(RGB(255, 255, 255));
//Create a Timer to self-destruct the dialog window
```

```
SetTimer(1, 3000, 0);
return true;
}

void OnTimer(UINT nIDEvent) { EndDialog(IDOK); }

DECLARE_MESSAGE_MAP()
};

BEGIN_MESSAGE_MAP(ShamanMessage, CDialog)
ON_WM_TIMER()
ON_WM_CTLCOLOR()
END_MESSAGE_MAP()

//-----------------------------------------------------------------------
-----------

class DragonImage : public CDialog
{
public:
DragonImage(CWnd* pParent = NULL): CDialog(DragonImage::IDD, pParent)
{ }

enum{IDD = IDD_DRAGON};

CBrush DragonImageBrush;

HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
switch (nCtlColor)
{
case CTLCOLOR_EDIT: pDC->SetTextColor(RGB(255,0,0));
case CTLCOLOR_STATIC: pDC->SetTextColor(RGB(255, 0, 0));
case CTLCOLOR_DLG: return DragonImageBrush;
default: return CDialog::OnCtlColor(pDC, pWnd, nCtlColor);
}
}

protected:
virtual void DoDataExchange(CDataExchange* pDX)
{CDialog::DoDataExchange(pDX);}

virtual BOOL OnInitDialog()
{
CDialog::OnInitDialog();
DragonImageBrush.CreateSolidBrush(RGB(255, 255, 255));
//Create Timer to self-destruct the dialog window
SetTimer(1, 3000, 0);
return true;
}

void OnTimer(UINT nIDEvent) { EndDialog(IDOK); }
```

```
DECLARE_MESSAGE_MAP()
};

BEGIN_MESSAGE_MAP(DragonImage, CDialog)
ON_WM_TIMER()
ON_WM_CTLCOLOR()
END_MESSAGE_MAP()

//-----------------------------------------------------------------------
-----------

class SaveChar : public CDialog
{
public:
SaveChar(CWnd* pParent = NULL): CDialog(SaveChar::IDD, pParent)
{ }

enum{IDD = IDD_SAVE};

CBrush SaveBrush;

HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
switch (nCtlColor)
{
case CTLCOLOR_EDIT: pDC->SetTextColor(RGB(255,255,255));
case CTLCOLOR_STATIC: pDC->SetTextColor(RGB(255, 255, 255));
pDC->SetBkMode(TRANSPARENT);
case CTLCOLOR_DLG: return SaveBrush;
default: return CDialog::OnCtlColor(pDC, pWnd, nCtlColor);
}
}

afx_msg void SaveIt()
{
pSaveName = ( CEdit * ) GetDlgItem( IDC_SaveName );
pSavePassword = ( CEdit * ) GetDlgItem( IDC_SavePassword );
bool success;
success = SaveCharacter();
if(success)
{
MessageBox(L"Your character was saved successfully!",
L"Character Saved Successfully!");
}
else
{
MessageBox(L"Your character could not be saved, for some reason.",
L"Error! Character Could Not Be Saved.");
}
}
```

```
protected:
virtual void DoDataExchange(CDataExchange* pDX)
{CDialog::DoDataExchange(pDX);}

virtual BOOL OnInitDialog()
{
CDialog::OnInitDialog();
SaveBrush.CreateSolidBrush(RGB(0, 0, 255));
return true;
}

DECLARE_MESSAGE_MAP()
};

BEGIN_MESSAGE_MAP(SaveChar, CDialog)
ON_WM_CTLCOLOR()
ON_COMMAND(IDSAVEOK, SaveIt)
END_MESSAGE_MAP()

//-----------------------------------------------------------------------
-----------

class LoadChar : public CDialog
{
public:
LoadChar(CWnd* pParent = NULL): CDialog(LoadChar::IDD, pParent)
{ }

enum{IDD = IDD_LOAD};

CBrush LoadBrush;

HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
switch (nCtlColor)
{
case CTLCOLOR_EDIT: pDC->SetTextColor(RGB(255,255,255));
case CTLCOLOR_STATIC: pDC->SetTextColor(RGB(255, 255, 255));
pDC->SetBkMode(TRANSPARENT);
case CTLCOLOR_DLG: return LoadBrush;
default: return CDialog::OnCtlColor(pDC, pWnd, nCtlColor);
}
}

afx_msg void LoadIt()
{
pLoadName = ( CEdit * ) GetDlgItem( IDC_LoadName );
pLoadPassword = ( CEdit * ) GetDlgItem( IDC_LoadPassword );
bool success; CString problem;

success = LoadCharacter(problem);
if(success)
```

```
{
MessageBox(L"Your character was loaded successfully!",
L"Character Loaded Successfully!");
}
else
{
MessageBox(problem, L"Error! Character Could Not Be Loaded.");
}
}

afx_msg void ExitLoad() { EndDialog(IDOK); }

protected:
virtual void DoDataExchange(CDataExchange* pDX)
{CDialog::DoDataExchange(pDX);}

virtual BOOL OnInitDialog()
{
CDialog::OnInitDialog();
LoadBrush.CreateSolidBrush(RGB(0, 0, 255));
return true;
}

DECLARE_MESSAGE_MAP()
};

BEGIN_MESSAGE_MAP(LoadChar, CDialog)
ON_WM_CTLCOLOR()
ON_COMMAND(IDLOADOK, LoadIt)
ON_COMMAND(IDLOADCANCEL, ExitLoad)
END_MESSAGE_MAP()

//-----------------------------------------------------------------------
-----------

//The Main Dialog-------------------------------------------------------
-----------
class HOD_Main_Window : public CDialog
{
public:
HOD_Main_Window(CWnd* pParent = NULL): CDialog(HOD_Main_Window::IDD,
pParent)
{

}

// Dialog Data
enum{IDD = HillsOfDark};

protected:
//-----------------------------------------------------------------------
virtual void DoDataExchange(CDataExchange* pDX)
```

```
{ CDialog::DoDataExchange(pDX); }
//-----------------------------------------------------------------------

CBrush MainDialogBrush; //Need for coloring dialog
CFont AFont; //Ned for setting font

//-----------------------------------------------------------------------
//Set Colors of Dialog
HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
switch (nCtlColor)
{
case CTLCOLOR_BTN:
case CTLCOLOR_LISTBOX:
case CTLCOLOR_MSGBOX:
case CTLCOLOR_SCROLLBAR:
case CTLCOLOR_EDIT: pDC->SetTextColor(RGB(0,0,255));
case CTLCOLOR_STATIC: pDC->SetTextColor(RGB(0, 0, 255));
//pDC->SetBkColor(RGB(255, 255, 255));
//pDC->SetBkMode(TRANSPARENT);
case CTLCOLOR_DLG: return MainDialogBrush;
default: return CDialog::OnCtlColor(pDC, pWnd, nCtlColor);
}
}

//-----------------------------------------------------------------------
//Message Map Handlers, assign global pointers to the Dialog controls
virtual BOOL OnInitDialog()
{
CDialog::OnInitDialog();

MainDialogBrush.CreateSolidBrush(RGB(255, 255, 255));
AFont.CreatePointFont(110,L"Comic Sans MS");

pInput = ( CEdit * ) GetDlgItem(IDC_INPUT);
pOutput = ( CEdit * ) GetDlgItem(IDC_OUTPUT);
pScore = ( CStatic * ) GetDlgItem( IDC_SCORE );
pInventory = ( CEdit * ) GetDlgItem( IDC_INVENTORY );
pHitpoints = ( CStatic * ) GetDlgItem( IDC_HITPOINTS );
pAttack = ( CStatic * ) GetDlgItem( IDC_ATTACK );
pDefense = ( CStatic * ) GetDlgItem( IDC_DEFENSE );
pConquests = ( CEdit * ) GetDlgItem( IDC_CONQUESTS );
pName = ( CStatic * ) GetDlgItem( IDC_NAME );
pView = ( CStatic * ) GetDlgItem( IDC_VIEW );

pDagger = ( CButton * ) GetDlgItem( IDC_RADIODAGGER );
pSword = ( CButton * ) GetDlgItem( IDC_RADIOSWORD );
pLongBow = ( CButton * ) GetDlgItem( IDC_RADIOBOW );
pHand = ( CButton * ) GetDlgItem( IDC_RADIOHAND );

pN = ( CButton * ) GetDlgItem( IDC_NORTH );
pS = ( CButton * ) GetDlgItem( IDC_SOUTH );
```

```
pE = ( CButton * ) GetDlgItem( IDC_EAST );
pW = ( CButton * ) GetDlgItem( IDC_WEST );
pSCORES = ( CButton * ) GetDlgItem( IDCHISCORES );
pSAVE = ( CButton * ) GetDlgItem( IDCSAVE );
pLOAD = ( CButton * ) GetDlgItem( IDCLOAD );
pSTART = ( CButton * ) GetDlgItem( IDCSTART );
pGO = ( CButton * ) GetDlgItem( IDGO );
pACTION = ( CButton * ) GetDlgItem( IDCACTION );
pHEAL = ( CButton * ) GetDlgItem( IDD_HEAL );

InitializeGlobals();
pOutput->SetWindowText(L"\r\nClick the \"START\" button to begin.");

return true;
}//close OnInitDialog() function
//---------------------------------------------------------------------

public:
//Functions for Messages

afx_msg void QuitTheGame()
{
if(StartedGame)
{
bool successful;
successful = SaveHighScores();

if(successful)
{
MESSAGE = MESSAGE + "\r\nScore saved to \"highscores.txt\".\r\n";
}
else
{
MESSAGE = MESSAGE + "\r\nScore saved to \"highscores.txt\".\r\n";
}

MESSAGE = MESSAGE + "\r\nScore saved to \"highscores.txt\".\r\n";
"\r\nExiting Hills of Darkness 2.0\r\n";

delete CurrentPlayer;
delete WiseWoman;
CurrentPlayer = 0;
WiseWoman = 0;
pOutput->SetWindowText(MESSAGE);
}

EndDialog(IDOK);
}
//---------------------------------------------------------------------

afx_msg void GO()
{
```

```
if(StartedGame)
{
if(NeedName)
{
CString n;
pInput->GetWindowText(n);
CurrentPlayer->setName(n);
pName->SetWindowText(n);
NeedName = false;
}

if(Continue)
{
WEAPONS();

if(!LOCK)
{
choice[0] = '#';

switch(location)
{
case INTRO : location = Introduction(); break;
case N1 : location = NORTH1(); break;
case S1 : location = SOUTH1(); break;
case E1 : location = EAST1(); break;
case W1 : location = WEST1(); break;
case CENTER1 : location = CENTER(); break;
case N2 : location = NORTH2(); break;
case S2 : location = SOUTH2(); break;
case E2 : location = EAST2(); break;
case W2 : location = WEST2(); break;
case UNDERGRND : location = UnderGroundPassage(); break;
case GATE : location = GateWay(); break;
case SHAMAN : location = SHAMANHUT(); break;
case DRAGONFIGHT : location = DragonFight(); break;
case GIANTFIGHT : location = GiantFight(); break;
case GIANTCAMP : location = GiantCamp(); break;
case GAMEOVER : location = GameOver(); break;
case YOUWIN : location = YouWin(); break;
case QUIT : Continue = false; break;
default : MESSAGE = MESSAGE + "\r\nNot an NAV option.\r\n"; break;
}//close switch

CurrentPlayer->DisplayStats();
CurrentPlayer->Inventory();

}//close if for "LOCK = false" (Unlocked)

else
{
//Begin else for "LOCK = true" (Locked)
```

```
CString TEMP;
pInput->GetWindowText(TEMP);
choice[0] = (char)TEMP[0];
choice[0] = tolower(choice[0]);
pInput->SetWindowText(L"");
pInput->SetFocus();

MESSAGE = "";

switch(location)
{
case INTRO : location = Introduction(); break;
case N1 : location = NORTH1(); break;
case S1 : location = SOUTH1(); break;
case E1 : location = EAST1(); break;
case W1 : location = WEST1(); break;
case CENTER1 : location = CENTER(); break;
case N2 : location = NORTH2(); break;
case S2 : location = SOUTH2(); break;
case E2 : location = EAST2(); break;
case W2 : location = WEST2(); break;
case UNDERGRND : location = UnderGroundPassage(); break;
case GATE : location = GateWay(); break;
case SHAMAN : location = SHAMANHUT(); break;
case DRAGONFIGHT : location = DragonFight(); break;
case GIANTFIGHT : location = GiantFight(); break;
case GIANTCAMP : location = GiantCamp(); break;
case GAMEOVER : location = GameOver(); break;
case YOUWIN : location = YouWin(); break;
case QUIT : Continue = false; break;
default : MESSAGE = MESSAGE
+ "\r\nNot a NAV option.\r\n"; break;
}//close switch

//If choice is valid, recursively call GO() to proceed to next
function
if(choice[0] == 'n' || choice[0] == 's' ||
choice[0] == 'e' || choice[0] == 'w' ||
choice[0] == 't' || choice[0] == 'r' ||
choice[0] == 'l' || choice[0] == 'g')
{
GO();
}

}//close else for "LOCK"

}//close if for "Continue = true"

else
{ //If "Continue = false"
QuitTheGame();
}
```

```
} //close if for "StartedGame = true", if not started, handle click,
don't crash

else
{
MESSAGE = "";
MESSAGE = MESSAGE +
"\r\nSorry, you can not press the \"GO\" button yet."
+ "\r\nYou need to click on the \"START\" button\r\nto "
+ "begin the game!";
pOutput->SetWindowText(MESSAGE);
}
}//close the GO button function
//---------------------------------------------------------------------

afx_msg void START()
{
//Start new unless in middle of a game, else confirm player really
wants to start over

if(!StartedGame)
{
AFunctionPointer1 = ShowShamanMessage;
AFunctionPointer2 = ShowDragon;

InitializeGlobals(); LockButtons(false);
pHand->SetCheck(1); //Default check

MESSAGE = "";

//To set the font on an object you must create a CFont instance.
CFont * TheFont = &AFont;
pOutput->SetFont(TheFont);
pName->SetFont(TheFont);

pInput->SetWindowText(L"Enter your name HERE.");
pInput->SetFocus(); //Changes Focus
pInput->SetSel(0, -1, false); //selects Text to conveniently delete
when user types

CString MESSAGE = "";
MESSAGE = MESSAGE +
"\r\nWelcome to Hills of Darkness 3.0. You need to create "
+ "a character and select a name. The object of this game "
+ "is to collect enough items and gain enough experience to "
+ "operate the dimensional gateway and return home. You will "
+ "also need to find a key to activate this mechanism. You "
+ "may ask the village shaman for help and useful advice on "
+ "our journey. \r\n\r\n\r\nType your name in the box to the "
+ "right.\r\nThen begin he game by pressing the \"GO\" button."
+ "\r\n\r\n\r\n";
```

```
srand((int)time(NULL)); //seed for random numbers

pOutput->SetWindowText(MESSAGE);
pConquests->SetWindowText(L"Loading...");

CurrentPlayer = new Player;
WiseWoman = new Shaman;
WEAPONS();
StartedGame = true;

}//close if for "StartedGame=false"

else
{
NagMe = MessageBox(L"You are in the middle of a game. Are you
sure?",
L"Start Over?", MB_YESNOCANCEL|MB_ICONINFORMATION);

if(NagMe == IDYES)
{
InitializeGlobals(); //If user clicks again, initialize
START();
}

}

}//close START() function
//---------------------------------------------------------------------

afx_msg void LOAD()
{
if(StartedGame)
{
//Code Necessary to Open the Save Character Dialog
LoadChar LOAD_EM;
LOAD_EM.DoModal();
StartedGame = true;
LOCK = true; GO();
}
else
{
MessageBox(L"Click the START button first, then load your
character.",
L"Click the START button first!");
}
}
//---------------------------------------------------------------------

afx_msg void SAVE()
{
if(StartedGame)
```

```
{
//Code Necessary to Open the Save Character Dialog
CurrentPlayer->setLocation(location);
SaveChar SAVE_EM;
SAVE_EM.DoModal();
StartedGame = true;
}
else
{
MessageBox(L"You can't save a character when a it does not exist!",
L"Mindless Error!!!");
}
}
//---------------------------------------------------------------------

afx_msg void DISPLAYSCORES()
{
if(!HiScoresToggle && StartedGame) { GO(); HiScoresToggle = true; }
else { DisplayHighScores(); HiScoresToggle = false; }
}
//---------------------------------------------------------------------

afx_msg void HEAL() {if(StartedGame) {CurrentPlayer-
>UseHealingPotion();}}

//---------------------------------------------------------------------

afx_msg void NORTH()
{
if(StartedGame)
{ pInput->SetWindowText(L"n"); LOCK = true; GO(); }
}
//---------------------------------------------------------------------

afx_msg void SOUTH()
{
if(StartedGame)
{ pInput->SetWindowText(L"s"); LOCK = true; GO(); }
}
//---------------------------------------------------------------------

afx_msg void EAST()
{
if(StartedGame)
{ pInput->SetWindowText(L"e"); LOCK = true; GO(); }
}
//---------------------------------------------------------------------

afx_msg void WEST()
{
if(StartedGame)
{ pInput->SetWindowText(L"w"); LOCK = true; GO(); }
```

```
}
//--------------------------------------------------------------------

afx_msg void ABOUT()
{
MessageBox(L"Hills of Darkness 3.0 - C. Germany 2006",
L"Hills of Darkness 3.0");
}
//--------------------------------------------------------------------

afx_msg void HELP()
{
char HELPTEXT[15][55] = {
"The object of this game is to collect enough",
" items\nand gain enough experience to operate",
" the\ndimensional gateway and return home.",
" You will\nalso need to find a key to activate",
" the gate\nmechanism. You may ask the village ",
"shaman\nfor help and useful advice on your journey.",
"\n\nFor every three opponents you defeat, \"1\" will",
"\nbe added to your Attack and Defense capabilities.",
"\nThis represents the natural skill you acquire as",
"\na result of combat, and is independent of and in",
"\naddition to any defense or attack capabilities",
"\nyou gain through the use of armor and weapons. In",
"\naddition to your attack/defense skills acquired",
"\nthrough experience, if you find armor and weapons,",
"\nthey will also add to your abilities." };

char Message[800] = "";

for(int z = 0; z < 15; z++)
{ strncat_s(Message, HELPTEXT[z], 55); }

MessageBox((LPCTSTR)Message, L"Hills of Darkness 3.0 - HELP");

}
//--------------------------------------------------------------------

afx_msg void WEAPONS()
{
//Moved here so player can change weapons while in combat
if(Continue && StartedGame)
{
//Disable Radio Buttons if Player Doesn't Have the Weapon
if(CurrentPlayer->getDagger())
{ pDagger->EnableWindow(1); }
else
{ pDagger->EnableWindow(0); }

if(CurrentPlayer->getSword())
{ pSword->EnableWindow(1); }
```

```
else
{ pSword->EnableWindow(0); }

if(CurrentPlayer->getLongBow())
{ pLongBow->EnableWindow(1); }
else
{ pLongBow->EnableWindow(0); }

int weapon = GetCheckedRadioButton(IDC_RADIOHAND, IDC_RADIOBOW);

switch(weapon)
{
case IDC_RADIOHAND: CurrentPlayer->setUseDagger(false);
CurrentPlayer->setUseSword(false);
CurrentPlayer->setUseLongBow(false);
break;
case IDC_RADIODAGGER: if(CurrentPlayer->getDagger())
{
CurrentPlayer->setUseDagger(true);
CurrentPlayer->setUseSword(false);
CurrentPlayer->setUseLongBow(false);
}
break;
case IDC_RADIOSWORD: if(CurrentPlayer->getSword())
{
CurrentPlayer->setUseSword(true);
CurrentPlayer->setUseDagger(false);
CurrentPlayer->setUseLongBow(false);
}
break;
case IDC_RADIOBOW: if(CurrentPlayer->getLongBow())
{
CurrentPlayer->setUseLongBow(true);
CurrentPlayer->setUseDagger(false);
CurrentPlayer->setUseSword(false);
}
break;
default: CurrentPlayer->setUseDagger(false);
CurrentPlayer->setUseSword(false);
CurrentPlayer->setUseLongBow(false);
break;
}//close switch

}//close if

}//close function
//---------------------------------------------------------------------

//Must be defined as static - they both use function pointers
static void ShowDragon() { DragonImage m; m.DoModal(); }
//---------------------------------------------------------------------
```

```
static void ShowShamanMessage() { ShamanMessage m; m.DoModal(); }
//---------------------------------------------------------------------

afx_msg void ACTION() { /*Empty, for later use*/ }
//---------------------------------------------------------------------

DECLARE_MESSAGE_MAP()
};

//Main Dialog Message Map-------------------------------------------
BEGIN_MESSAGE_MAP(HOD_Main_Window, CDialog)

ON_COMMAND(IDCQUIT, QuitTheGame)
ON_COMMAND(IDGO, GO)
ON_COMMAND(IDCACTION, ACTION)
ON_COMMAND(IDCSTART, START)
ON_COMMAND(IDCSAVE, SAVE)
ON_COMMAND(IDCLOAD, LOAD)
ON_COMMAND(IDCHISCORES, DISPLAYSCORES)
ON_COMMAND(IDD_HEAL, HEAL)
ON_COMMAND(IDC_NORTH, NORTH)
ON_COMMAND(IDC_SOUTH, SOUTH)
ON_COMMAND(IDC_EAST, EAST)
ON_COMMAND(IDC_WEST, WEST)
ON_COMMAND(ID_FILE_SAVEGAME, SAVE)
ON_COMMAND(ID_FILE_LOADGAME, LOAD)
ON_COMMAND(ID_OTHER_HELP, HELP)
ON_COMMAND(ID_OTHER_ABOUT, ABOUT)
ON_COMMAND(ID_FILE_EXIT, QuitTheGame)
ON_COMMAND(IDCHELP, HELP)
ON_COMMAND(IDC_RADIOHAND, WEAPONS)
ON_COMMAND(IDC_RADIODAGGER, WEAPONS)
ON_COMMAND(IDC_RADIOSWORD, WEAPONS)
ON_COMMAND(IDC_RADIOBOW, WEAPONS)
ON_WM_CTLCOLOR()

END_MESSAGE_MAP()

//---------------------------------------------------------------------
-----------

class TheGame : public CWinApp
{
public:
TheGame() { }

public:
virtual BOOL InitInstance()
{
CWinApp::InitInstance();
SetRegistryKey(_T("Charles Germany"));
HOD_Main_Window dlg;
```

```
m_pMainWnd = &dlg;
INT_PTR nResponse = dlg.DoModal();
return FALSE;
}//close function
};
//--------------------------------------------------------------------
-----------

TheGame theApp;
```

File 6 of 7 "INTERFACE_MAIN_resources.h":

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by INTERFACE_MAIN_resources.rc
//-----------------------------------------------
#define IDR_MENU1 101
#define HillsOfDarkness 102
#define HillsOfDark 103
#define IDD_SAVE 104
#define IDB_BITMAP1 105
#define IDD_DIALOG1 106
#define IDB_BITMAP2 107
#define B_ACTION 108
#define IDD_SHAMAN 109
#define IDD_SHAMAN1 110
#define IDCACTION2 111
#define IDCSTART 112
#define IDCSTART2 113
#define IDD_HEAL 114
#define IDC_VIEW 115
#define IDC_EDIT1 116
#define ID_FILE_EXIT 117
#define ID_FILE_LOADGAME 118
#define ID_ABOUT 119
#define ID_OTHER_HELP 120
#define ID_OTHER_ABOUT 121
//-----------------------------------------------
//#define HillsOfDark 122
#define IDC_TITLE 123
#define IDCQUIT 124
#define IDGO 125
#define IDC_OUTPUT 126
#define IDC_INPUT 127
#define IDCACTION 128
#define IDCSAVE 129
#define IDCLOAD 130
#define IDC_NORTH 131
#define IDCHISCORES 132
#define IDC_SOUTH 133
```

```
#define IDC_EAST 134
#define IDC_WEST 135
#define IDC_INVENTORY 136
#define IDC_HITPOINTS 137
#define IDC_ATTACK 138
#define IDC_DEFENSE 139
#define IDC_SCORE 140
#define IDC_EDIT4 141
#define IDC_EDIT5 142
#define IDC_CONQUESTS 143
#define IDC_NAME 144
//#define IDCSTART 145
//#define IDD_HEAL 146
#define IDC_RADIOHAND 147
#define IDC_RADIODAGGER 148
#define IDC_RADIOSWORD 149
#define IDC_RADIOBOW 150

//#define IDR_MENU1 151
//#define ID_FILE_EXIT 152
#define ID_FILE_SAVEGAME 153
//#define ID_FILE_LOADGAME 154
//#define ID_OTHER_HELP 155
//#define ID_OTHER_ABOUT 156

//#define IDD_SAVE 157
#define IDSAVEOK 158
//#define IDCANCEL 159
#define IDC_SaveName 160
#define IDC_SavePassword 161

#define IDD_LOAD 162
#define IDLOADOK 163
#define IDLOADCANCEL 164
#define IDC_LoadName 165
#define IDC_LoadPassword 166

#define IDD_DRAGON 167
#define IDB_DRAGON 168

//#define IDD_SHAMAN 169
#define IDB_GIANT 170

//#define IDC_VIEW 171
#define IDCHELP 172
//------------------------------------------------
#define IDM_ABOUTBOX 173
#define IDS_ABOUTBOX 174
#define IDR_MAINFRAME 175
//------------------------------------------------
// Next default values for new objects
//
```

```
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE 176
#define _APS_NEXT_COMMAND_VALUE 40008
#define _APS_NEXT_CONTROL_VALUE 1038
#define _APS_NEXT_SYMED_VALUE 176
#endif
#endif
```

File 7 of 7 "INTERFACE_MAIN.rc":

```
// Microsoft Visual C++ generated resource script.
//


#define APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
//#include "resource.h"
#include "INTERFACE_MAIN_resources.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

//////////////////////////////////////////////////////////////////////////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

//////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

HillsOfDark DIALOGEX 0, 0, 449, 220
STYLE DS_SETFONT | DS_MODALFRAME | DS_3DLOOK | DS_CENTER | WS_POPUP |
WS_VISIBLE | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_CLIENTEDGE | WS_EX_APPWINDOW
CAPTION
" Hills of Darkness 3.0 - 2006 C. Germany"

MENU IDR_MENU1
FONT 9, "Century Gothic", 400, 0, 0x0
```

```
BEGIN
DEFPUSHBUTTON "GO",IDGO,322,40,31,9
PUSHBUTTON "QUIT",IDCQUIT,406,35,30,11
EDITTEXT IDC_OUTPUT,2,22,214,191,ES_MULTILINE | ES_READONLY |
WS_VSCROLL | NOT WS_TABSTOP,WS_EX_CLIENTEDGE
EDITTEXT IDC_INPUT,218,20,138,18,ES_AUTOHSCROLL,WS_EX_CLIENTEDGE
PUSHBUTTON "ACTION",IDCACTION,288,40,31,9
PUSHBUTTON "SAVE",IDCSAVE,369,35,30,11
PUSHBUTTON "LOAD",IDCLOAD,369,22,30,11
PUSHBUTTON "SCORES",IDCHISCORES,369,9,30,11
PUSHBUTTON "N",IDC_NORTH,239,60,15,12
PUSHBUTTON "S",IDC_SOUTH,239,84,15,12
PUSHBUTTON "E",IDC_EAST,255,72,15,12
PUSHBUTTON "W",IDC_WEST,223,72,15,12
GROUPBOX "Direction",IDC_STATIC,218,50,57,50,BS_CENTER
EDITTEXT IDC_INVENTORY,355,59,91,109,ES_MULTILINE | ES_READONLY |
WS_VSCROLL,WS_EX_TRANSPARENT | WS_EX_STATICEDGE
CTEXT "Inventory",IDC_STATIC,379,50,40,8
CTEXT "Hitpoints",IDC_STATIC,219,0,34,8,0,WS_EX_TRANSPARENT
CTEXT "",IDC_HITPOINTS,221,8,31,10,0,WS_EX_TRANSPARENT |
WS_EX_CLIENTEDGE
CTEXT "Attack",IDC_STATIC,256,0,31,8,0,WS_EX_TRANSPARENT
CTEXT "",IDC_ATTACK,256,8,31,10,0,WS_EX_TRANSPARENT |
WS_EX_CLIENTEDGE
CTEXT "Defense",IDC_STATIC,289,0,34,8
CTEXT "",IDC_DEFENSE,291,8,31,10,0,WS_EX_TRANSPARENT |
WS_EX_CLIENTEDGE
CTEXT "Score",IDC_STATIC,326,0,28,8
CTEXT "",IDC_SCORE,325,8,31,10,0,WS_EX_TRANSPARENT |
WS_EX_CLIENTEDGE
LTEXT "Name:",IDC_STATIC,2,10,22,10
LTEXT "",IDC_NAME,27,8,189,11,SS_CENTERIMAGE,WS_EX_TRANSPARENT |
WS_EX_CLIENTEDGE
PUSHBUTTON "START",IDCSTART,406,22,30,11
PUSHBUTTON "HEALING POTION",IDD_HEAL,218,40,67,9
GROUPBOX "Attack Weapon",IDC_STATIC,278,50,76,50,BS_CENTER,
WS_EX_TRANSPARENT
CONTROL "Hand to Hand",IDC_RADIOHAND,"Button",BS_AUTORADIOBUTTON,
285,58,61,10
CONTROL "Dagger",IDC_RADIODAGGER,"Button",BS_AUTORADIOBUTTON,285,
68,40,10
CONTROL "Sword",IDC_RADIOSWORD,"Button",BS_AUTORADIOBUTTON,285,
78,36,10
CONTROL "Long Bow",IDC_RADIOBOW,"Button",BS_AUTORADIOBUTTON,285,
88,47,10
PUSHBUTTON "HELP",IDCHELP,406,9,30,11
CONTROL "",IDC_VIEW,"Static",SS_BITMAP | SS_CENTERIMAGE |
WS_BORDER,218,110,133,104,WS_EX_CLIENTEDGE
CTEXT "View",IDC_STATIC,266,102,34,8
GROUPBOX "MENU",IDC_STATIC,364,0,77,49,BS_CENTER
EDITTEXT IDC_CONQUESTS,352,180,94,34,ES_MULTILINE | ES_READONLY |
WS_VSCROLL,WS_EX_TRANSPARENT | WS_EX_STATICEDGE
```

```
CTEXT "Conquests",IDC_STATIC,377,171,37,8
END

IDD_SAVE DIALOGEX 0, 0, 211, 71
STYLE DS_SETFONT | DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Hills of Darkness 3.0 - C. Germany 2006 - Save Character"
FONT 9, "Comic Sans MS", 400, 0, 0x0
BEGIN
DEFPUSHBUTTON "SAVE",IDSAVEOK,146,17,50,14
PUSHBUTTON "EXIT",2,146,35,50,14
EDITTEXT IDC_SaveName,8,18,130,12,ES_AUTOHSCROLL
LTEXT "Name To Save Character File As:",IDC_STATIC,9,8,124,10
EDITTEXT IDC_SavePassword,8,44,130,12,ES_PASSWORD |
ES_AUTOHSCROLL
LTEXT "Password (Up to 10 characters):",IDC_STATIC,9,34,124,10
END

IDD_LOAD DIALOGEX 0, 0, 211, 71
STYLE DS_SETFONT | DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Hills of Darkness 3.0 - C. Germany 2006 - Load Character"
FONT 9, "Comic Sans MS", 400, 0, 0x0
BEGIN
DEFPUSHBUTTON "LOAD",IDLOADOK,146,17,50,14
PUSHBUTTON "EXIT",IDLOADCANCEL,146,35,50,14
EDITTEXT IDC_LoadName,8,18,130,12,ES_AUTOHSCROLL
LTEXT "Name To Load Character File As:",IDC_STATIC,9,8,124,10
EDITTEXT IDC_LoadPassword,8,44,130,12,ES_PASSWORD |
ES_AUTOHSCROLL
LTEXT "Password (Up to 10 characters):",IDC_STATIC,9,34,124,10
END

IDD_SHAMAN DIALOGEX 0, 0, 49, 23
STYLE DS_SETFONT | DS_MODALFRAME | DS_3DLOOK | DS_CONTROL | WS_POPUP |
WS_VISIBLE | WS_SYSMENU
EXSTYLE WS_EX_WINDOWEDGE
FONT 48, "Fat", 400, 0, 0x0
BEGIN
LTEXT "BAD Karma!",IDC_STATIC,4,4,42,8
LTEXT "Seriously ...",IDC_STATIC,3,12,43,8
END

IDD_DRAGON DIALOGEX 0, 0, 49, 18
STYLE DS_SETFONT | DS_MODALFRAME | DS_3DLOOK | WS_POPUP | WS_SYSMENU
FONT 48, "Fat", 400, 0, 0x0
BEGIN
LTEXT "Are you sure",IDC_STATIC,2,2,46,8
LTEXT "about this?",IDC_STATIC,2,9,47,8
END


//////////////////////////////////////////////////////////////////////////
//
```

```
// Menu
//

IDR_MENU1 MENU
BEGIN
POPUP "&File"
BEGIN
MENUITEM "E&xit", ID_FILE_EXIT
MENUITEM "&Save Game", ID_FILE_SAVEGAME
MENUITEM "&Load Game", ID_FILE_LOADGAME
END
POPUP "&Other"
BEGIN
MENUITEM "&Help", ID_OTHER_HELP
MENUITEM "&About", ID_OTHER_ABOUT
END
END


#ifdef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE
BEGIN
"resrc1.h\0"
END

2 TEXTINCLUDE
BEGIN
"#include ""resource.h""\r\n"
"#include ""afxres.h""\r\n"
"\0"
END

3 TEXTINCLUDE
BEGIN
"\r\n"
"\0"
END

#endif // APSTUDIO_INVOKED

#endif // English (U.S.) resources
////////////////////////////////////////////////////////////////////////////



#ifndef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
```

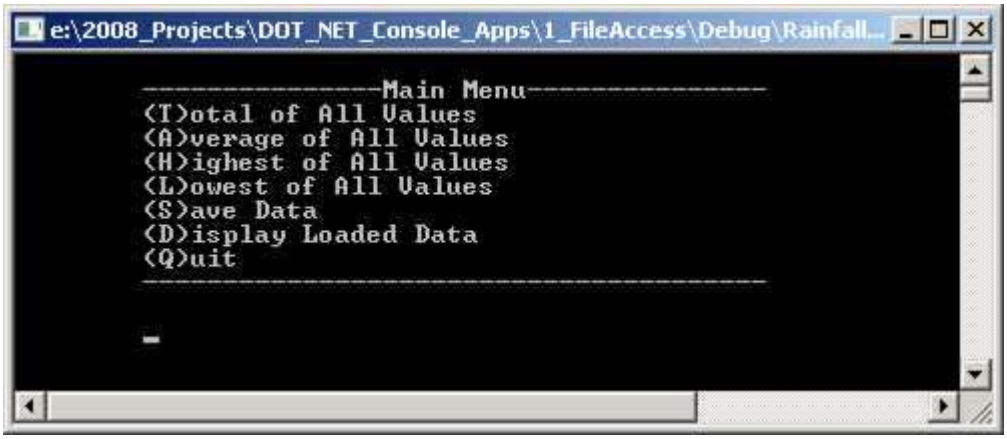
```
//
// Generated from the TEXTINCLUDE 3 resource.
//


/////////////////////////////////////////////////////////////////////////////
#endif // not APSTUDIO_INVOKED
```


2008 .Net Console ASCII File Access Application

C++ 2008 Visual Studio 2008 .Net Console Application

Binary Executable: 2008_ASCII_FileAccess.exe

Objective: To utilize .Net Console ASCII File Access



File 1: MAIN.cpp

```
//01/16/2010 C. Germany - Visual Studio 2008 .Net ASCII File
Access
/*
    Notes: 1. Select "File" -> "New" -> "Project" -> "Visual C++".
           2. Select"Win32" -> "Win32 Console Project".
               3. Rt-click project, select "properties,
                  select "Common Language Runtime support",
                  select "Common Language Runtime Support (/clr)"
                  to incorporate the .Net runtime instead of MFC.
    Some functions are prototyped to take array arguments but
        defined to take pointers to illustrate that a pointer to
        the first element of an array can be used to access the
        entire array.
*/
#include "stdafx.h"
```

```
#using <mscorlib.dll>
using namespace System;
using namespace System::IO;
//function prototypes
void PAUSE();
void LoadData();
void SaveData();
void DisplayMenu();
void DisplayTotal(double r[]);
void DisplayAverage(double r[]);
void DisplayHigh(double r[]);
void DisplayLow(double r[]);
const int MAXSIZE = 12;
int RecordCount = 0;
int sub = 0;
double * Array_Of_Doubles = new double[MAXSIZE];
//-----------------------------------------------------------------
int main()
{
        //Fill array with Array_Of_Doubles amounts from text file
        LoadData();
    PAUSE();
    DisplayMenu();

        Console::WriteLine("\n\n\tExiting program.");
        PAUSE();
        return 0;
}
//-----------------------------------------------------------------
void PAUSE()
{
        Console::WriteLine("\tPAUSED: Press ENTER to continue.");
        Console::ReadLine();
        System::Console::Beep();
        System::Console::Clear();
}
//-----------------------------------------------------------------
void SaveData()
{
    StreamWriter ^ OutputFile;
        bool CreatedIt = false;
    String ^ DATA = "NOTHING";
        sub = 0;
        //try opening the file
        try
        {
                OutputFile = gcnew
StreamWriter("Number_DATA.txt");
                Console::WriteLine("\nMade it!");
                CreatedIt = true;
        }
        catch(IOException ^ ex)
        {
                Console::WriteLine("\nCouldn't create file...");
```

```
                Console::WriteLine(ex->Message);
                CreatedIt = false;
        }
        //only if the file was opened successfully
        if (CreatedIt == true)
        {
           Console::WriteLine("\n----------Enter Values to Store--
--------");
           Console::WriteLine("                Q = QUIT");
           while(DATA[0] != 'q')
           {
                        Console::Write("\nEnter value for record
{0} ", Convert::ToString(sub+1));
            DATA = Console::ReadLine();
            DATA = DATA->ToLower();
                        if(DATA[0] != 'q')
                        {
                                //See if 1st character is a number
                                if(System::Char::IsNumber(DATA[0]))
                                {
                                   OutputFile->WriteLine(DATA);
                                   sub = sub + 1;
                                }
                                else
                                {
                                   Console::WriteLine("\n\tNon
numerical values are not allowed!");
                                }
                        }

                }//end while
                OutputFile->Close();
        Console::WriteLine("\n\n----------Closing file.----------
\n\n");
        }//close if
}
//---------------------------------------------------------------
//Can take a pointer to a double to accept entire array of doubles
void LoadData()
{
    StreamReader ^ InputFile;
        bool FoundIt = false;
    sub = 0;
        RecordCount = 0;  //reset each time
        //try opening the file
        try
        {
                InputFile = gcnew StreamReader("Number_DATA.txt");
                Console::WriteLine("\nFound it!");
                FoundIt = true;
        }
        catch(IOException ^ ex)
        {
```

```
                Console::WriteLine("\nWe have a situation
here...");
                Console::WriteLine(ex->Message);
                FoundIt = false;
        }
        //only if the file was opened successfully
        if (FoundIt == true)
        {
           Console::WriteLine("\n----------Opening file and
reading.----------");
       while(InputFile->Peek() != -1 && sub < MAXSIZE)
           {
                        //assign the number to the
array
                        Array_Of_Doubles[sub] =
Convert::ToDouble(InputFile->ReadLine());
                        Console::Write("\nValue of subscript {0} ",
Convert::ToString(sub));
                        Console::Write(" = {0}",
Convert::ToString(Array_Of_Doubles[sub]));
                        //update accumulator and access Array
subscript to obtain sentinel value
                        sub = sub + 1;
                        RecordCount++;

                }//end while
                InputFile->Close();
        Console::WriteLine("\n\n----------Closing file.----------
\n\n");
        }//close if
}//close function
//---------------------------------------------------------------
void DisplayMenu()
{
     String ^ ANSWER = "GO";
         while(ANSWER[0] != 'q')
         {
            Console::WriteLine("\n\t---------------Main Menu------
---------");
            Console::WriteLine("\t(T)otal of All Values");
            Console::WriteLine("\t(A)verage of All Values");
            Console::WriteLine("\t(H)ighest of All Values");
            Console::WriteLine("\t(L)owest of All Values");
                Console::WriteLine("\t(S)ave Data");
        Console::WriteLine("\t(D)isplay Loaded Data");
        Console::WriteLine("\t(Q)uit");
        Console::WriteLine("\t-----------------------------------
---");
                Console::Write("\n\t");
            ANSWER = Console::ReadLine();
                ANSWER = ANSWER->ToLower();

                switch(ANSWER[0])
            {
```

```
            case 't' : DisplayTotal(Array_Of_Doubles);
                                    break;
                   case 'h' : DisplayHigh(Array_Of_Doubles);
                                    break;
                   case 'l' : DisplayLow(Array_Of_Doubles);
                                    break;
                   case 'a' : DisplayAverage(Array_Of_Doubles);
                                    break;
                      case 's' : SaveData();
                                    break;
            case 'd' : LoadData();
                                    break;
                      case 'q' : break;
                   default :  Console::WriteLine("\n\tGet
REAL!");
                                    break;
            }//close switch
        }//close loop
}//close function
//-------------------------------------------------------------------
void DisplayTotal(double * r)
{
        //declare accumulator variable
        double total = 0.0;
        //accumulate Array_Of_Doubles amounts, then display total
        for (int x = 0; x < RecordCount; x = x + 1)
        {
                total = total + r[x];
        }
        Console::WriteLine("\n\tTotal of all numbers = {0}.",
                                Convert::ToString(total));
        Console::WriteLine();
}
//For you to complete on your own.
//-------------------------------------------------------------------
void DisplayAverage(double * r)
{
         //declare accumulator variable
         double AVG = 0.0;
         //accumulate Array_Of_Doubles amounts, then display total
         for (int x = 0; x < RecordCount; x = x + 1)
         {
                AVG = AVG + r[x];
         }
     AVG = AVG / RecordCount;
         Console::WriteLine("\n\tAverage of all numbers = {0}.",
                                 Convert::ToString(AVG));
         Console::WriteLine();
}
//-------------------------------------------------------------------
void DisplayHigh(double * r)
{
         //declare accumulator variable
         double HIGH = 0.0;
```

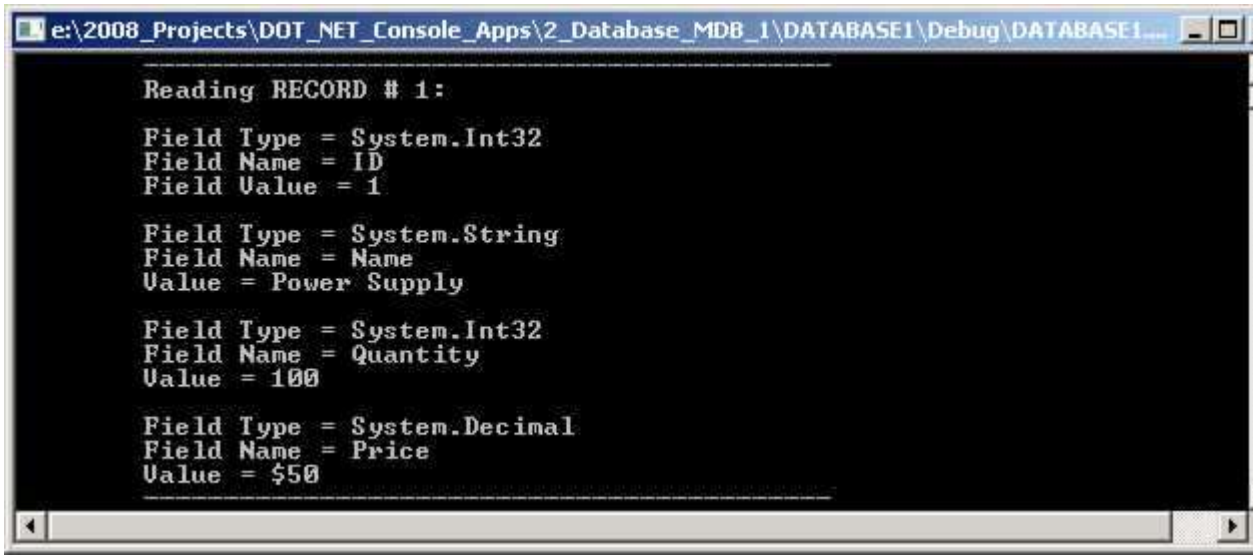
```
        //accumulate Array_Of_Doubles amounts, then display total
        for (int x = 0; x < RecordCount; x = x + 1)
        {
               if(HIGH < r[x])
               {
                        Console::Write("\nValue for number {0}=",
Convert::ToString(x+1));
                        Console::Write("{0} ",
Convert::ToString(r[x]));
                        Console::Write("and IS HIGHER than HIGH
which is {0}.",

Convert::ToString(HIGH));
                        HIGH = r[x];
               }
               else
               {
                        Console::Write("\nValue for number {0}=",
Convert::ToString(x+1));
                        Console::Write("{0} ",
Convert::ToString(r[x]));
                        Console::Write("and is NOT HIGHER than HIGH
which is {0}.",

Convert::ToString(HIGH));
               }
        }
        Console::WriteLine("\n\n\tHighest number = {0}.",
                                 Convert::ToString(HIGH));
        Console::WriteLine();
}
//---------------------------------------------------------------
void DisplayLow(double * r)
{
        //declare accumulator variable
        double LOW = r[0];
        //accumulate Array_Of_Doubles amounts, then display total
        for (int x = 0; x < RecordCount; x = x + 1)
        {
               if(r[x] > LOW)
               {
           Console::Write("\nValue for number {0}=",
Convert::ToString(x+1));
                        Console::Write("{0} ",
Convert::ToString(r[x]));
                        Console::Write("and is NOT LOWER than LOW
which is {0}.",
                                                 Convert::ToString(LOW));
               }
               else
               {
                        Console::Write("\nValue for number {0}=",
Convert::ToString(x+1));
```

```
                        Console::Write("{0} ",
Convert::ToString(r[x]));
                        Console::Write("and IS LOWER than LOW which
is {0}.",
                                                Convert::ToString(LOW));
                        LOW = r[x];
                }
        }
        Console::WriteLine("\n\n\tLowest number = {0}.",
                                Convert::ToString(LOW));
        Console::WriteLine();
}
//-------------------------------------------------------------------
```


2008 .Net Console OleDB SQL Database Application 1

C++ 2008 Visual Studio 2008 .Net Console OleDB SQL Application Binary Executable: 2008_SQL_Database1.exe

Objective: To utilize .Net Console OleDB SQL Database Access



File 1: MAIN.cpp

```
//01/16/2010 C. Germany - Visual Studio 2008 OleDB Database 1
/*
    Notes: 1. Select "File" -> "New" -> "Project" -> "Visual C++".
```

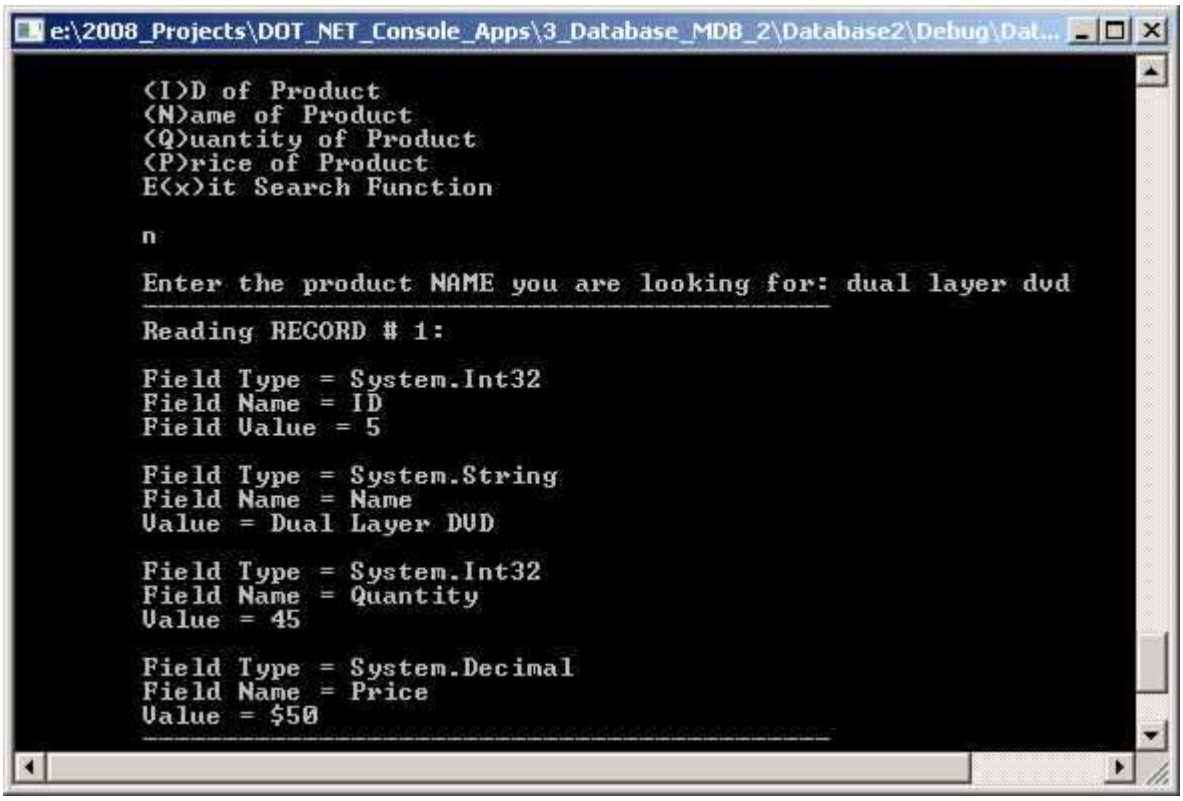
```
           2. Select"Win32" -> "Win32 Console Application".
               3. Rt-click project, select "properties,
                  select "Common Language Runtime support",
                  select "Common Language Runtime Support (/clr)"
                  to incorporate the .Net runtime instead of MFC.
*/
#include "stdafx.h"
#using <mscorlib.dll>
#using <System.dll>
#using <System.data.dll>
using namespace System;
using namespace System::Data;
using namespace System::Data::OleDb;
//------------------------------------------------------------------
int main()
{
        OleDbConnection ^ Database_Connection = gcnew
OleDbConnection();
        //Example 1: ABSOLUTE path Connection String
        //Database_Connection->ConnectionString =
        //"Provider=Microsoft.Jet.OLEDB.4.0; Data
Source=d:\\2008_Projects\\DOT_NET_Console_Apps\\
         //Database_MDB_1\\DATABASE1\\DATABASE1\\Inventory.mdb";
    //Example 2: RELATIVE path Connection String
    Database_Connection->ConnectionString =
        "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=Inventory.mdb";
        try
        {
          Database_Connection->Open();
                 //Create Command object to get information from
database via SQL
                 String ^ sql = "SELECT ID, Name, Quantity, Price FROM
Inventory";
                 OleDbCommand ^ Command_Inventory = gcnew
OleDbCommand(sql,Database_Connection);
        //Create Reader object to get records and display them
                OleDbDataReader ^ Read_Inventory = Command_Inventory-
>ExecuteReader();
        int COUNTER = 1;
                while(Read_Inventory->Read() == true)
                {

                        Console::Write("\t----------------------------
---------------");
                        Console::Write("\n\tReading RECORD # ");
                        Console::Write(Convert::ToString(COUNTER));
                        Console::Write(":\n");
                        //ID
                        Console::Write("\n\tField Type = ");
                        Console::Write(Read_Inventory-
>GetFieldType(0));
            Console::Write("\n\tField Name = ");
                        Console::Write(Read_Inventory->GetName(0));
            Console::Write("\n\tField Value = ");
```

```
                        Console::Write(Read_Inventory->GetInt32(0));

                        //Name
                        Console::Write("\n\n\tField Type = ");
                        Console::Write(Read_Inventory-
>GetFieldType(1));
                        Console::Write("\n\tField Name = ");
                        Console::Write(Read_Inventory->GetName(1));
                        Console::Write("\n\tValue = ");
                        Console::Write(Read_Inventory->GetString(1));
                        //Quantity
                        Console::Write("\n\n\tField Type = ");
                        Console::Write(Read_Inventory-
>GetFieldType(2));
                        Console::Write("\n\tField Name = ");
                        Console::Write(Read_Inventory->GetName(2));
                        Console::Write("\n\tValue = ");
            Console::Write(Read_Inventory->GetInt32(2));
                        //Price
                        Console::Write("\n\n\tField Type = ");
                        Console::Write(Read_Inventory-
>GetFieldType(3));
                        Console::Write("\n\tField Name = ");
                        Console::Write(Read_Inventory->GetName(3));
            Console::Write("\n\tValue = $");
                        Console::Write(Read_Inventory->GetDecimal(3));
                        Console::WriteLine("\n\t-----------------------
----------------------\n\n");
                        COUNTER++;
                }
         Read_Inventory->Close();
         Database_Connection->Close();
         }//close try
         catch(OleDbException ^ A_PROBLEM)
         {
                Console::WriteLine(A_PROBLEM->Message);
         }

         Console::WriteLine("\n\n\t");
    Console::WriteLine("Press ENTER to continue.");
         Console::ReadLine();
         return 0;
}
//------------------------------------------------------------------------
```


2008 .Net Console OleDB SQL Database Application 2

C++ 2008 Visual Studio 2008 .Net Console OleDB SQL Application Binary Executable: 2008_SQL_Database2.exe

Objective: To utilize .Net Console OleDB SQL Database Access



File 1: MAIN.cpp

```
//01/16/2010 C. Germany - Visual Studio 2008 Database Application
Using SQL
/*
    Notes: 1. Select "File" -> "New" -> "Project" -> "Visual C++".
           2. Select"Win32" -> "Win32 Console Application".
               3. Rt-click project, select "properties,
                  select "Common Language Runtime support",
                  select "Common Language Runtime Support (/clr)"
                  to incorporate the .Net runtime instead of MFC.
           4. This project uses SQL. SQL is an acronym that stands
for:
                  "Structured Query Language". It is a simple
database language.
              5. This project uses OleDb to access an Access
database file that
                  you will create. OleDb = Object Linking and
Embedding Database.
                      Access database files have an extension of
".mdb".
*/
#include "stdafx.h"
```

```
#using <mscorlib.dll>
#using <System.dll>
#using <System.data.dll>
using namespace System;
using namespace System::Data;
using namespace System::Data::OleDb;
//Prototypes
void PAUSE();
void DisplayMenu(OleDbConnection ^ DB);
void DisplayDatabase(OleDbConnection ^ DB, String ^ SQL);
void BasicRead(OleDbConnection ^ DB);
void Sort(OleDbConnection ^ DB);
void Find(OleDbConnection ^ DB);
int main()
{

    OleDbConnection ^ Database_Connection = gcnew
OleDbConnection();
    Database_Connection->ConnectionString =
        "Provider=Microsoft.Jet.OLEDB.4.0; Data
Source=inventory.mdb";
    DisplayMenu(Database_Connection);
    PAUSE();
        return 0;
}
//-----------------------------------------------------------------
void PAUSE()
{
        Console::WriteLine("\tPAUSED: Press ENTER to continue.");
        Console::ReadLine();
        System::Console::Beep();
        System::Console::Clear();
}
//-----------------------------------------------------------------
void DisplayMenu(OleDbConnection ^ DB)
{
     String ^ ANSWER = "GO";
         while(ANSWER[0] != 'q')
         {
            Console::WriteLine("\n\t---------------Main Menu-------
--------");
        Console::Write("\n\t(B)asic Read Operation");
        Console::Write("\n\t(S)ort Operation");
        Console::Write("\n\t(F)ind Operation");
               Console::Write("\n\t(Q)uit\n\n");
        Console::WriteLine("\t-----------------------------------
---");
               Console::Write("\n\t");
            ANSWER = Console::ReadLine();
               ANSWER = ANSWER->ToLower();

               switch(ANSWER[0])
            {
                   case 'b' : BasicRead(DB); break;
```

```
                  case 's' : Sort(DB); break;
            case 'f' : Find(DB); break;
                       case 'q' : break;
                     default :  Console::WriteLine("\n\tGet REAL!");
                                    break;
            }//close switch
        }//close loop
}//close function
//-------------------------------------------------------------
void DisplayDatabase(OleDbConnection ^ DB, String ^ SQL)
{
         try
         {
         DB->Open();
                //Create Command object to get information from
database via SQL
                OleDbCommand ^ Command_Inventory = gcnew
OleDbCommand(SQL,DB);
         //Create Reader object to get records and display them
                OleDbDataReader ^ Read_Inventory =
Command_Inventory->ExecuteReader();
         int COUNTER = 1;
                while(Read_Inventory->Read() == true)
                {
                       Console::Write("\t-------------------------
-----------------");
                       Console::Write("\n\tReading RECORD # ");
                       Console::Write(Convert::ToString(COUNTER));
                       Console::Write(":\n");
                       //ID
                       Console::Write("\n\tField Type = ");
                       Console::Write(Read_Inventory-
>GetFieldType(0));
            Console::Write("\n\tField Name = ");
                       Console::Write(Read_Inventory->GetName(0));
            Console::Write("\n\tField Value = ");
                       Console::Write(Read Inventory->GetInt32(0));

                       //Name
                       Console::Write("\n\n\tField Type = ");
                       Console::Write(Read_Inventory-
>GetFieldType(1));
                       Console::Write("\n\tField Name = ");
                       Console::Write(Read_Inventory->GetName(1));
                       Console::Write("\n\tValue = ");
                       Console::Write(Read_Inventory-
>GetString(1));
                       //Quantity
                       Console::Write("\n\n\tField Type = ");
                       Console::Write(Read_Inventory-
>GetFieldType(2));
                       Console::Write("\n\tField Name = ");
                       Console::Write(Read_Inventory->GetName(2));
                       Console::Write("\n\tValue = ");
```

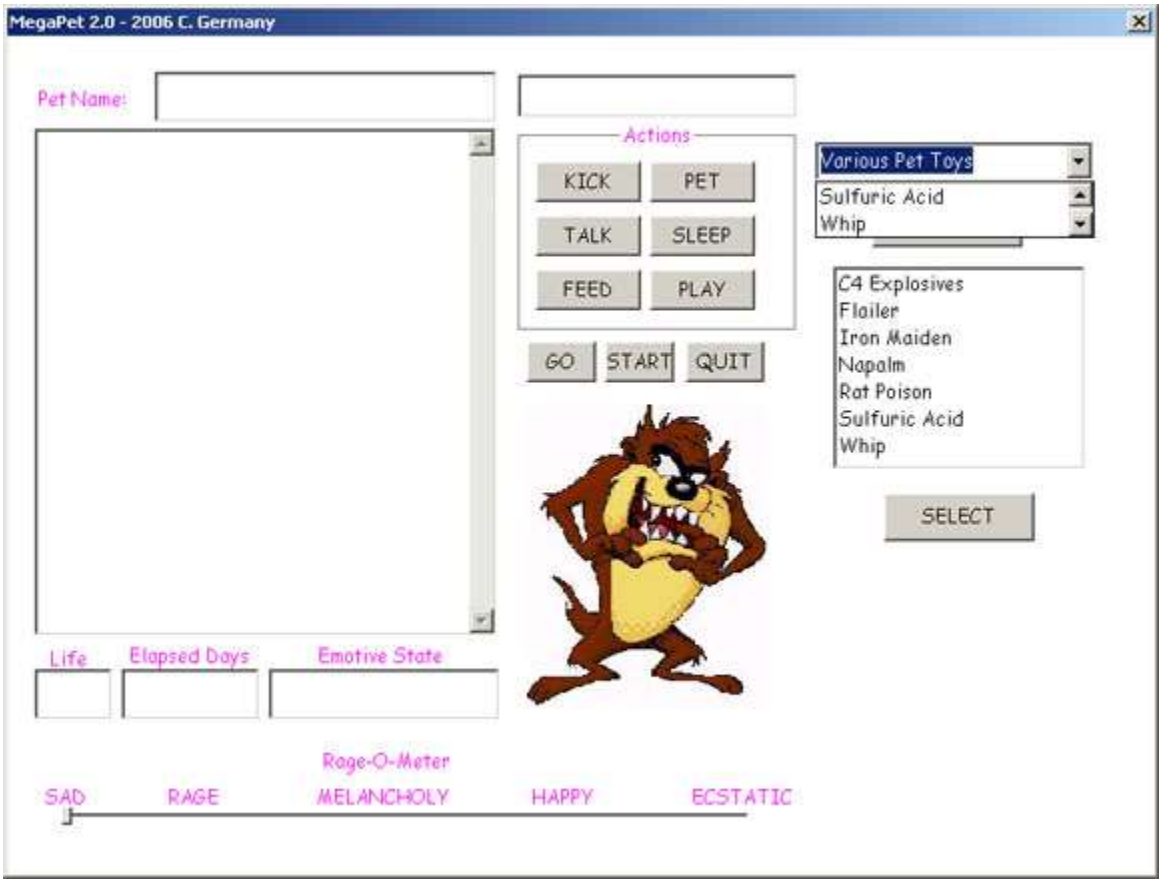
```
            Console::Write(Read_Inventory->GetInt32(2));
                        //Price
                        Console::Write("\n\n\tField Type = ");
                        Console::Write(Read_Inventory-
>GetFieldType(3));
                        Console::Write("\n\tField Name = ");
                        Console::Write(Read_Inventory->GetName(3));
            Console::Write("\n\tValue = $");
                        Console::Write(Read_Inventory-
>GetDecimal(3));
                        Console::WriteLine("\n\t-------------------
-----------------------\n\n");
                        COUNTER++;
                }
        Read_Inventory->Close();
        DB->Close();

        }
        catch(OleDbException ^ A_PROBLEM)
        {
                Console::WriteLine(A_PROBLEM->Message);
        }

}
//-----------------------------------------------------------------
----------------
void BasicRead(OleDbConnection ^ DB)
{
     Console::WriteLine("\n\n\t----------BASIC READ FUNCTION------
----");
     String ^ ACTION = "SELECT ID, Name, Quantity, Price FROM
Inventory";
     DisplayDatabase(DB,ACTION);
}
//-----------------------------------------------------------------
----------------
void Sort(OleDbConnection ^ DB)
{
         //Note: By default SQL commands sort in ASCENDING order
(least to greatest)
         //Use the keyword "DESC" to sort in DESCENDING order.
Example:
         //SQL = "SELECT ID, Name, Quantity, Price FROM Inventory
ORDER BY ID DESC"
         String ^ ACTION = " ";
     String ^ CHOICE = " ";
         Console::WriteLine("\n\n\t----------DATABASE SORT
FUNCTION----------");

     while(CHOICE[0] != 'x')
         {
                Console::Write("\n\t----------SORT MENU: How Do You
Want Records Sorted?----------\n");
        Console::Write("\n\t(I)D of Product");
```

```
        Console::Write("\n\t(N)ame of Product");
        Console::Write("\n\t(Q)uantity of Product");
               Console::Write("\n\t(P)rice of Product");
        Console::Write("\n\tE(x)it Search Function\n\n\t");
               CHOICE = Console::ReadLine();
               CHOICE = CHOICE->ToLower();
        switch(CHOICE[0])
               {
                    case 'i' : ACTION = "SELECT ID, Name,
Quantity, Price FROM Inventory ORDER BY ID";
                       DisplayDatabase(DB,ACTION);
                                    CHOICE = "~";  //Invalidate
CHOICE
                                    break;
                    case 'n' : ACTION = "SELECT ID, Name,
Quantity, Price FROM Inventory ORDER BY Name";
                       DisplayDatabase(DB,ACTION);
                                    CHOICE = "~";  //Invalidate
CHOICE
                                    break;
             case 'q' : ACTION = "SELECT ID, Name, Quantity, Price
FROM Inventory ORDER BY Quantity";
                       DisplayDatabase(DB,ACTION);
                                    CHOICE = "~";  //Invalidate
CHOICE
                                    break;
                    case 'p' : ACTION = "SELECT ID, Name,
Quantity, Price FROM Inventory ORDER BY Price";
                       DisplayDatabase(DB,ACTION);
                                    CHOICE = "~";  //Invalidate
CHOICE
                                    break;
             case 'x' : break;
                       default: Console::WriteLine("\n\tInvalid
option."); break;
               }
       }
}
//-----------------------------------------------------------------
----------------
void Find(OleDbConnection ^ DB)
{
         String ^ ACTION = " ";
     String ^ CHOICE = " ";
     Console::WriteLine("\n\n\t----------FIND FUNCTION----------
");

     while(CHOICE[0] != 'x')
         {
                Console::Write("\n\t----------SEARCH MENU: What
Are You Looking For?----------\n");
         Console::Write("\n\t(I)D of Product");
         Console::Write("\n\t(N)ame of Product");
         Console::Write("\n\t(Q)uantity of Product");
```

```
                Console::Write("\n\t(P)rice of Product");
          Console::Write("\n\tE(x)it Search Function\n\n\t");
                CHOICE = Console::ReadLine();
                CHOICE = CHOICE->ToLower();
         switch(CHOICE[0])
                {
                     case 'i' : Console::Write("\n\tEnter the
product ID you are looking for: ");
                                     CHOICE = Console::ReadLine();
                                              ACTION =
String::Concat("SELECT ID, Name, Quantity, Price FROM Inventory
WHERE ID = ",CHOICE);
                          DisplayDatabase(DB,ACTION);
                                     CHOICE = "~";  //Invalidate
CHOICE
                                     break;
                    case 'n' : Console::Write("\n\tEnter the
product NAME you are looking for: ");
                                     CHOICE = Console::ReadLine();
                                     ACTION =
String::Concat("SELECT ID, Name, Quantity, Price FROM Inventory
WHERE Name = '",CHOICE);
                                     ACTION =
String::Concat(ACTION,"'");
                          DisplayDatabase(DB,ACTION);
                                     CHOICE = "~";  //Invalidate
CHOICE
                                     break;
              case 'q' : Console::Write("\n\tEnter the product
QUANTITY you are looking for: ");
                                     CHOICE = Console::ReadLine();
                                     ACTION =
String::Concat("SELECT ID, Name, Quantity, Price FROM Inventory
WHERE Quantity = ",CHOICE);
                          DisplayDatabase(DB,ACTION);
                                     CHOICE = "~";  //Invalidate
CHOICE
                                     break;
                    case 'p' : Console::Write("\n\tEnter the
product PRICE you are looking for: ");
                                     CHOICE = Console::ReadLine();
                                     ACTION =
String::Concat("SELECT ID, Name, Quantity, Price FROM Inventory
WHERE Price = ",CHOICE);
                          DisplayDatabase(DB,ACTION);
                                     CHOICE = "~";  //Invalidate
CHOICE
                                     break;
              case 'x' : break;
                          default: Console::WriteLine("\n\tInvalid
option."); break;
                }
        }
}
```

```
//---------------------------------------------------------------------
---------------
```


MFC Example Using a ListBox and a ComboBox



This project is a continuation of the MegaPet you created as a console program (See cproject2.html). We will convert this game to a graphical MFC game as we learn each new MFC component one by one. The true purpose of this project is to learn about the MFC, but it will be a fun project as a beneficial side effect to our MFC studies.

1. File -> New -> Blank Solution -> Visual C++ Projects -> Win32 Project (Windows).
2. Give the project a name and set the directory.
3. Select "Application Settings" and check the "Empty Project" checkbox.
4. Right-click on the project name in the pane to your left and select "Properties".
5. IMPORTANT!!! Select "Use of MFC" dropdown list and select "Use MFC in a Shared DLL".
6. Right-Click on the source, header and resource folders to and create the 3 files named below.
7. Paste the code from each table below into its respective file in 2003 .Net Visual Studio.

File 1 "MegaPet.cpp":

```
//File 1 of 3. The interface for the dialog and form.

#include <afxwin.h> // MFC core and standard components
#include <afxcmn.h> //
#include "resource.h" // main symbols
#include <mmsystem.h>
#include "Classes.h"

//A Note: For sound, you must do three things:
//1. Go to project properties, then under Linker, find Input.
//In the box labelled Aditional Dependancies add "winmm.lib".
//2. Include the file: #include <mmsystem.h> .
//3. Use command: PlaySound("west.wav",NULL,SND_FILENAME|SND_ASYNC);
//Options: SYNC = ends with function, async = keep playing
//SND_LOOP = loop it, must be stopped then with "StopSound()".
//PlaySound("west.wav",NULL,SND_FILENAME|SND_ASYNC|SND_LOOP);

//-----------------------------------------------------------------

class MegaPet_FORM : public CDialog
{
public:
MegaPet_FORM (CWnd* pParent = NULL): CDialog(MegaPet_FORM ::IDD, pParent)
{ }

// Dialog Data
enum{IDD = IDD_MegaPet};

protected:
virtual void DoDataExchange(CDataExchange* pDX)
{ CDialog::DoDataExchange(pDX); }

CBrush MegaPetBrush;
CFont PetFont;
HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
switch (nCtlColor)
{
case CTLCOLOR_EDIT: pDC->SetTextColor(RGB(255,0,255));
case CTLCOLOR_STATIC: pDC->SetTextColor(RGB(255,0,255));
case CTLCOLOR_DLG: return MegaPetBrush;
default: return CDialog::OnCtlColor(pDC, pWnd, nCtlColor);
}
}

//Message Map Handlers
virtual BOOL OnInitDialog()
{
CDialog::OnInitDialog();
MegaPetBrush.CreateSolidBrush(RGB(255, 255, 255));
pInput = (CEdit *) GetDlgItem(IDC_INPUT);
pOutput = (CEdit *) GetDlgItem(IDC_OUTPUT);
pName = (CEdit *) GetDlgItem(IDC_NAME);
pPetView = (CStatic *) GetDlgItem(IDC_VIEW);
```

```
pLife = (CStatic *) GetDlgItem(IDC_LIFE);
pEmotiveState = (CStatic *) GetDlgItem(IDC_EMOTIVESTATE);
pDays = (CStatic *) GetDlgItem(IDC_DAYS);
pGO = (CButton *) GetDlgItem(IDC_GO);
RAGEoMETER = (CSliderCtrl *) GetDlgItem(IDC_RAGEMETER);
pComboBox = (CComboBox *) GetDlgItem(IDC_ComboBoxExample);
pComboBoxButton = (CButton *) GetDlgItem(IDC_ComboBoxButton);
pListBox = (CListBox *) GetDlgItem(IDC_ListBoxExample);
pListBoxButton = (CButton *) GetDlgItem(IDC_ListBoxButton);
Started = false; NeedName = true;
srand(time(NULL));

PetFont.CreatePointFont(150,"Comic Sans MS");
CFont * TheFont = &PetFont;
pOutput->SetFont(TheFont);
pName->SetFont(TheFont);
pLife->SetFont(TheFont);
pDays->SetFont(TheFont);

//ComboBox
pComboBox->SetWindowText("Various Pet Toys");
pComboBox->AddString("Napalm");
pComboBox->AddString("C4 Explosives");
pComboBox->AddString("Rat Poison");
pComboBox->AddString("Sulfuric Acid");
pComboBox->AddString("Iron Maiden");
pComboBox->AddString("Whip");
pComboBox->AddString("Flailer");

//ListBox
pListBox->SetWindowText("Various Pet Toys");
pListBox->AddString("Napalm");
pListBox->AddString("C4 Explosives");
pListBox->AddString("Rat Poison");
pListBox->AddString("Sulfuric Acid");
pListBox->AddString("Iron Maiden");
pListBox->AddString("Whip");
pListBox->AddString("Flailer");

//Invalidate();
//UpdateWindow();
//Create timer to subtract 1 life point every 10 seconds
return true;
}

//-------------------------------------------------------------

afx_msg void HandleComboBox()
{
int index = pComboBox->GetCurSel();

switch(index)
{
case 0 : MessageBox("0"); break;
case 1 : MessageBox("1"); break;
case 2 : MessageBox("2"); break;
```

```
case 3 : MessageBox("3"); break;
case 4 : MessageBox("4"); break;
case 5 : MessageBox("5"); break;
case 6 : MessageBox("6"); break;
}

CString TEMP;
pComboBox->GetWindowText(TEMP);
MessageBox(TEMP);

}

//---------------------------------------------------------------

afx_msg void HandleListBox()
{
int index = pListBox->GetCurSel();

switch(index)
{
case 0 : MessageBox("0"); break;
case 1 : MessageBox("1"); break;
case 2 : MessageBox("2"); break;
case 3 : MessageBox("3"); break;
case 4 : MessageBox("4"); break;
case 5 : MessageBox("5"); break;
case 6 : MessageBox("6"); break;
}

CString TEMP;
pComboBox->GetWindowText(TEMP);
MessageBox(TEMP);
}

//---------------------------------------------------------------

void OnTimer(UINT nIDEvent)
{
Pet->setDays((Pet->getDays() + 1));
Pet->setLife((Pet->getLife() - 1));
Pet->View();
strncpy(DISPLAY,"",2000);
Pet->Talk();

if(Pet->getLife() <= 0)
{
KillTimer(1);
pOutput->SetWindowText("\r\n\r\nGame Over.Your pet has died of
starvation.");
}

}

//---------------------------------------------------------------------
afx_msg void KICK() { if(Started) { Pet->Kick(); } }
//---------------------------------------------------------------------
```

```
afx_msg void PET() { if(Started) { Pet->Pet(); } }
//---------------------------------------------------------------------
afx_msg void TALK() { if(Started) { Pet->Talk(); } }
//---------------------------------------------------------------------
afx_msg void SLEEP() { if(Started) { Pet->Sleep(); } }
//---------------------------------------------------------------------
afx_msg void FEED() { if(Started) { Pet->Feed(); } }
//---------------------------------------------------------------------
afx_msg void PLAY() { if(Started) { Pet->Play(); } }
//---------------------------------------------------------------------
afx_msg void GO()
{
if(Started)
{
if(NeedName) { Pet->SetName(); }
WRITEME.seekp(0); strncpy(DISPLAY, "", 2000);

Pet->View();

WRITEME << "\r\n"
<< Pet->getPetName() << " begins its new life with"
<< " you, happy and content...\r\n";

pOutput->SetWindowText(DISPLAY);
SetTimer(1, 8000, 0);
pInput->ShowWindow(false);
pGO->ShowWindow(false);

}
else { MessageBox("You must click \"START\" first.", "So Sorry"); }
}
//---------------------------------------------------------------------
afx_msg void START()
{
WRITEME.seekp(0); strncpy(DISPLAY,"",2000);

WRITEME << "\r\nWelcome to MegaPet 2.0.\r\n\r\nTo begin, you must "
<< "a MegaPet.\r\n\r\nGive your new MegaPet a name by typing"
<< " it in the input box above the \"Actions\" buttons."
<< "\r\n\r\nThen when you are ready, click \"GO\" to "
<< "continue.\r\n\r\n";

Pet = new MegaPet;
pEmotiveState->SetWindowText("Loading...");
pName->SetWindowText("Loading...");
pDays->SetWindowText("Loading...");
pLife->SetWindowText("Wait...");
pOutput->SetWindowText(DISPLAY);
Started = true;
}
//---------------------------------------------------------------------
afx_msg void QUIT() { EndDialog(IDOK); }
//---------------------------------------------------------------------


DECLARE_MESSAGE_MAP()
```

```
};

//--------------------------------------------------------------------
BEGIN_MESSAGE_MAP(MegaPet_FORM, CDialog)

ON_WM_TIMER()
ON_WM_CTLCOLOR()
ON_COMMAND(IDC_ComboBoxButton, HandleComboBox)
ON_COMMAND(IDC_ListBoxButton, HandleListBox)
ON_COMMAND(IDC_KICK, KICK)
ON_COMMAND(IDC_PET, PET)
ON_COMMAND(IDC_TALK, TALK)
ON_COMMAND(IDC_SLEEP, SLEEP)
ON_COMMAND(IDC_FEED, FEED)
ON_COMMAND(IDC_PLAY, PLAY)
ON_COMMAND(IDC_GO, GO)
ON_COMMAND(IDC_START, START)
ON_COMMAND(IDC_QUIT, QUIT)

END_MESSAGE_MAP()

//--------------------------------------------------------------------

class MegaPet_Window : public CWinApp
{
public:
MegaPet_Window() { }

public:
virtual BOOL InitInstance()
{
CWinApp::InitInstance();
SetRegistryKey(_T("MegaPet"));
MegaPet_FORM MegaPetDialog;
m_pMainWnd = &MegaPetDialog;
INT_PTR nResponse = MegaPetDialog.DoModal();
return FALSE;
} //close function

DECLARE_MESSAGE_MAP()
};

//--------------------------------------------------------------------
//Need a Message Map Macro for both CDialog and CWinApp
BEGIN_MESSAGE_MAP(MegaPet_Window, CWinApp)
END_MESSAGE_MAP()

//--------------------------------------------------------------------

MegaPet_Window LetErrRip; //This starts the ball rolling
```

File 2 "Classes.h":

```
//File 2 of 4. "Classes.h". Globals/Class Definitions. MegaPet 2.0, C.
Germany, June 22, 2006

#include <string>
#include <windows.h>
#include <strstream>    //Necessary for ostrstream object

using namespace std;

static char DISPLAY[2000];
static ostrstream WRITEME(DISPLAY, 2000);

bool NeedName;
bool Started;
int Day;

//Global pointers
CEdit * pOutput;
CEdit * pInput;
CEdit * pName;
CStatic * pPetView;
CStatic * pLife;
CStatic * pEmotiveState;
CStatic * pDays;
CButton * pGO;
CSliderCtrl * RAGEoMETER;

//-------------------------------------------------------------

int RandomNumber(int n)
{
    int ResultRandom;
    ResultRandom = (rand()%n) + 1;
    return ResultRandom;
}

//-------------------------------------------------------------

class  MegaPet
{
       public:
       MegaPet(int LF = 50, int ES = 5)
       {
            Life = LF; EmotiveState = ES;
            WRITEME << "\r\nCreating a MegaPet.\r\n";
            pOutput->SetWindowText(DISPLAY);
            Day = 1;
       }

       ~MegaPet()
       {
            WRITEME << "\r\nDestroying a MegaPet.\r\n";
            pOutput->SetWindowText(DISPLAY);
       }
```

```
//-------------------------------------------------------------
//Function Definitions

void SetName()
{
         char n[40];
         pInput->GetWindowText(n, 40);
          PetName = n;
         pName->SetWindowText(PetName);
         NeedName = false;
         pInput->SetWindowText("");
}

//-------------------------------------------------------------

void Feed()
{
    strncpy(DISPLAY,"",2000);
    WRITEME.seekp(0);

    if(Life < 50)
    {
        WRITEME << "\r\nYou feed your pet. This brings it "
               << "happy contentment and adds 3 points to"
                << " its Life.\r\n";

        Life = Life + 3;
    }

    else
    {
        WRITEME << "\r\nYour feed your pet, but it is just not"
               << " hungry. As a result, it gets sick and vomits"
                << " on your new carpet. This causes you to scold"
                << " it, and it looses 1 life point.\r\n";

        Life = Life - 1;
    }

    View();
    pOutput->SetWindowText(DISPLAY);
    PlaySound("media/TazFart.wav",NULL,SND_FILENAME|SND_ASYNC);
}

//-------------------------------------------------------------

void Pet()
{
    strncpy(DISPLAY,"",2000);
    WRITEME.seekp(0);

    if(EmotiveState < 5)
    {
        WRITEME << "\r\nYou pet your pet. This affection raises "
                << "its spirits, improving its outlook on life.\r\n";
```

```
        EmotiveState = EmotiveState + 1;
    }

    else
    {
        WRITEME << "\r\nYour pet does not require petting. It feels"
               << " loved and content.\r\n";
    }

    View();
    pOutput->SetWindowText(DISPLAY);
    PlaySound("media/TazLaugh.wav",NULL,SND_FILENAME|SND_ASYNC);
}

//-------------------------------------------------------------

void Sleep()
{
     strncpy(DISPLAY,"",2000);
     WRITEME.seekp(0);

     if(Life < 50)
     {
       WRITEME << "\r\nYour pet could use some sleep. It gains 1"
              << " life point!\r\n";

       Life = Life + 2;
     }

     else
     {
       WRITEME << "\r\nYour pet does not need any sleep. You put "
              << "it to bed any way, causing it undue anxiety and"
               << " it looses 1 life point.\r\n";

       Life = Life - 1;
     }

    View();
    pOutput->SetWindowText(DISPLAY);
}

//-------------------------------------------------------------

void Talk()
{
     strncpy(DISPLAY,"",2000);
     WRITEME.seekp(0);

     int PetState = RandomNumber(10);

     WRITEME << "\r\n";

     if(EmotiveState > 3)
     {
```

```
          switch(PetState)
          {
              case 1 : WRITEME << "Purr! I love you master! My life for
you!\r\n";
                       break;
              case 2 : WRITEME << "I am happy and content. Life is
good.\r\n";
                       break;
              case 3 : WRITEME << "I am fairly content with things.\r\n";
                       break;
              case 4 : WRITEME << "I am o.k. Are you o.k.?\r\n";
                       break;
              case 5 : WRITEME << "I am surviving, getting by...\r\n";
                       break;
              case 6 : WRITEME << "It's all good, baby.\r\n";
                       break;
              case 7 : WRITEME << "That's life I guess.\r\n";
                       break;
              case 8 : WRITEME << "I am so happy it is hard not to wet
myself.\r\n";
                       break;
              case 9 : WRITEME << "I must be in heaven!\r\n";
                       break;
              case 10 : WRITEME << "When I'm with you, everything is
o.k.\r\n";
                        break;
              default : WRITEME << "This should never happen! Only values
1-10.\r\n";
        } //close switch

        PlaySound("media/TazTalk1.wav",NULL,SND_FILENAME|SND_ASYNC);

     } // close if

     else if(EmotiveState <= 3 && EmotiveState > 1)
     {
              switch(PetState)
          {
              case 1 : WRITEME << "I feel really sad. So sad...\r\n";
                       break;
              case 2 : WRITEME << "I am so lonely. Please pet me
master.\r\n";
                       break;
              case 3 : WRITEME << "Sigh, if only...\r\n";
                       break;
              case 4 : WRITEME << "Have I displeased you master?\r\n";
                       break;
              case 5 : WRITEME << "Don't you love me any more?\r\n";
                       break;
              case 6 : WRITEME << "I crave your love and
affection...\r\n";
                       break;
              case 7 : WRITEME << "Why me?\r\n";
                       break;
              case 8 : WRITEME << "Feelin' mighty low...\r\n";
                       break;
```

```
              case 9 : WRITEME << "My life is a living hell.\r\n";
                       break;
              case 10 : WRITEME << "I'm thinking about ways to kill
myself.\r\n";
                        break;
              default : WRITEME << "I don't want to go on living.\r\n";
        } //close switch

        PlaySound("media/TazTalk2.wav",NULL,SND_FILENAME|SND_ASYNC);
     }

     else
     {
        WRITEME << "\r\nI hate you. I want to bite your freakin' arm off"
                          << " master. When you sleep at night I will
lift my leg"
                << " I will shred your favorite pillow, and wizz in your"
                << "  shoes! I will defecate on your homework.\r\n";

     }

     //While the first statement is based on EmotiveState, the second is
based on life
     if(Life > 40 && Life <= 50)
     { WRITEME << "\r\nI'm not really hungry or sleepy."; }
     if(Life > 30 && Life <= 40)
     { WRITEME << "\r\nI feel a little hungry."; }
     if(Life > 20 && Life <= 30)
     { WRITEME << "\r\nI feel really hungry and sleepy."; }
     if(Life > 10 && Life <= 20)
     { WRITEME << "\r\nI am starving to death."; }
     if(Life > 0 && Life <= 10)
     { WRITEME << "\r\nI am so weak and don't think I will last much
longer."; }

     View();
     pOutput->SetWindowText(DISPLAY);
}

//-------------------------------------------------------------

void View()
{
     HBITMAP EmoPix;
     CString Feeling;

     switch(EmotiveState)
     {
        case 1 : Feeling = "Depressed and Suicidal";
                           EmoPix = (HBITMAP)
LoadImage(NULL,"MEDIA/STILL_Sad.bmp",
                            IMAGE_BITMAP, 0,0,LR_DEFAULTSIZE |
LR_LOADFROMFILE);
                           RAGEoMETER->SetPos(0);    //SAD
                                break;
        case 2 : Feeling = "Down and Out";
```

```
                            EmoPix = (HBITMAP)
LoadImage(NULL,"MEDIA/STILL_Angry.bmp",
                             IMAGE_BITMAP, 0,0,LR_DEFAULTSIZE |
LR_LOADFROMFILE);
                            RAGEoMETER->SetPos(19);   //RAGE
                                 break;
        case 3 : Feeling = "Melancholy";
                            EmoPix = (HBITMAP)
LoadImage(NULL,"MEDIA/STILL_Melancholy.bmp",
                             IMAGE_BITMAP, 0,0,LR_DEFAULTSIZE |
LR_LOADFROMFILE);
                            RAGEoMETER->SetPos(47);   //MELANCHOLY
                                 break;
        case 4 : Feeling = "Fairly Happy";
                            EmoPix = (HBITMAP)
LoadImage(NULL,"MEDIA/STILL_Happy.bmp",
                             IMAGE_BITMAP, 0,0,LR_DEFAULTSIZE |
LR_LOADFROMFILE);
                            RAGEoMETER->SetPos(73);   //HAPPY
                                 break;
        case 5 : Feeling = "Ecstatic with Joy";
                            EmoPix = (HBITMAP)
LoadImage(NULL,"MEDIA/STILL_Ecstatic.bmp",
                             IMAGE_BITMAP, 0,0,LR_DEFAULTSIZE |
LR_LOADFROMFILE);
                            RAGEoMETER->SetPos(100);  //ECSTATIC
                            break;
        default : break;
     }

     char buffer[10];
     char n[50];

     itoa(Life, buffer, 10);
     pLife->SetWindowText(buffer);

     pEmotiveState->SetWindowText(Feeling);

     strcpy(n, "  ");
     strncat(n, PetName, 50);
     pName->SetWindowText(n);

     itoa(Day, buffer, 10);
     pDays->SetWindowText(buffer);

     pPetView->SetBitmap(EmoPix);
}

//-------------------------------------------------------------

void Kick()
{
     strncpy(DISPLAY,"",2000);
     WRITEME.seekp(0);
```

```
     WRITEME << "\r\n\r\nYou kick your pet. You are a real
jerk!\r\n\r\n";

     if(EmotiveState > 1)
     {
        EmotiveState = EmotiveState - 1;
     }

     else
     {
         WRITEME << "\r\nYou are not worthy of being a pet
owner.\r\n\r\n"
                 << " Kick 'em when they're down? You should be charged"
                 << " with animal cruelty and your pet taken away.\r\n";

         EmotiveState = 1;
     }

     View();
     pOutput->SetWindowText(DISPLAY);
     PlaySound("media/Kick.wav",NULL,SND_FILENAME|SND_ASYNC);
}
//-------------------------------------------------------------

void Play()
{
     strncpy(DISPLAY,"",2000);
     WRITEME.seekp(0);

     WRITEME << "Playing with pet...\r\n";

     if(Life < 50) { Life = Life + 1; }

     View();
     pOutput->SetWindowText(DISPLAY);
}

//-------------------------------------------------------------

    //Accessors--------------------------------------
        int getLife() { return Life; }
        int getEmotiveState() { return EmotiveState; }
        int getDays() { return Day; }
        CString getPetName() { return PetName; }

        void setLife(int x) { Life = x; }
        void setEmotiveState(int x) { EmotiveState = x; }
        void setDays(int x) { Day = x; }
        void setPetName(CString x) { PetName = x; }
    //-----------------------------------------------

private:
int Life;
int EmotiveState;
int Day;
CString PetName;
```

```
}; //End MegaPet class specification
```

File 3 "resourceh":

```
//File 3 of 4. The "resource.h" file.  IDS - MegaPet 2.0, C. Germany, May
27, 2006

#define IDD_MegaPet                     101
#define IDB_BITMAP1                     102
#define TAZ                             102
#define IDC_OUTPUT                      1001
#define IDC_NAME                        1002
#define IDC_QUIT                        1003
#define IDC_INPUT                       1004
#define IDC_FEED                        1005
#define IDC_TALK                        1006
#define IDC_KICK                        1007
#define IDC_PET                         1008
#define IDC_SLEEP                       1009
#define IDC_PLAY                        1010
#define IDC_START                       1011
#define IDC_START2                      1012
#define IDC_GO                          1012
#define IDC_VIEW                        1013
#define IDC_LIFE                        1014
#define IDC_EMOTIVESTATE                1015
#define IDC_DAYS                        1016

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        103
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1018
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

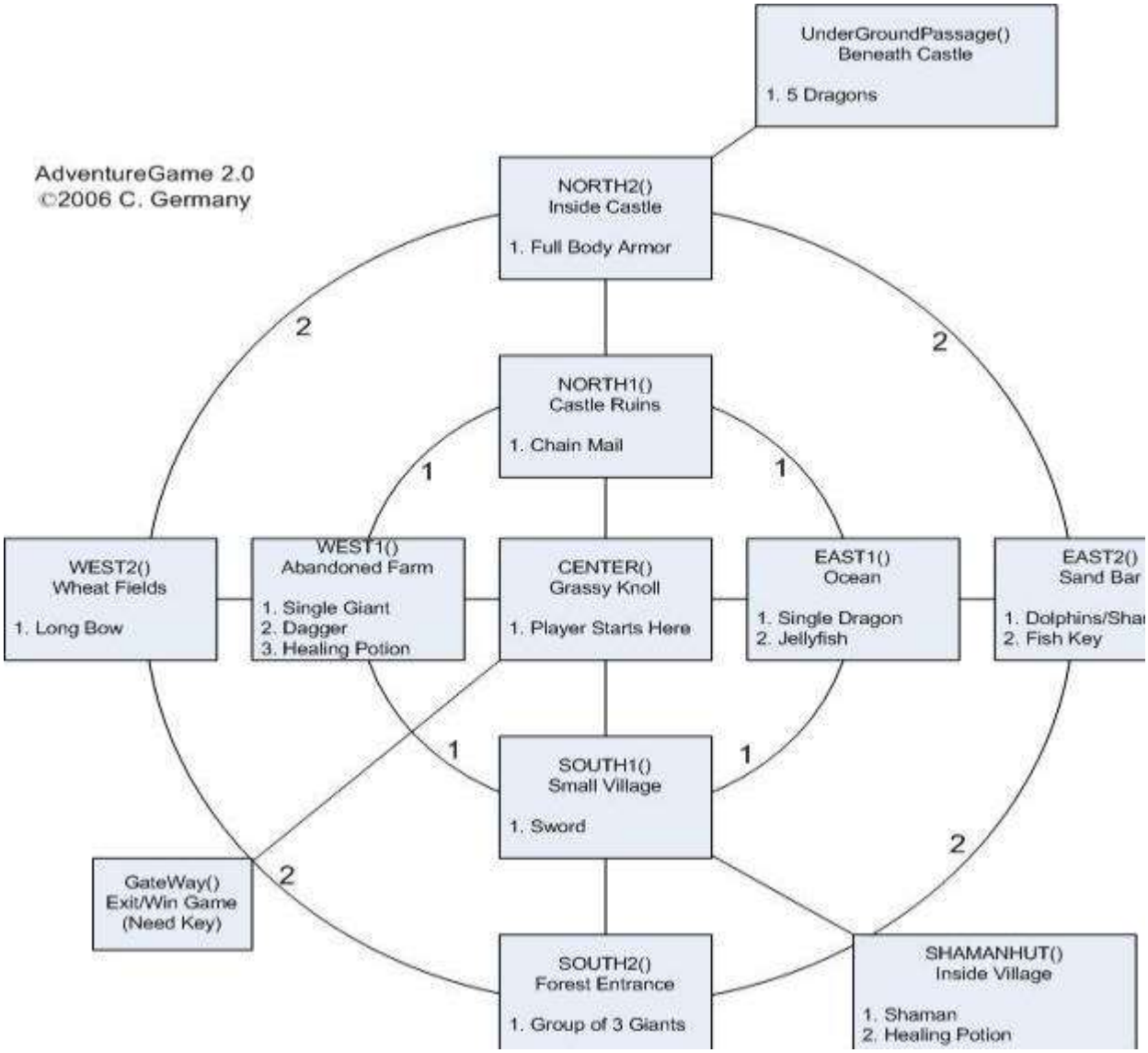
File 4 "MegaPet.rc":

```
// File 4 of 4. Resource file for Dialog

#include "resource.h"
#include "afxres.h"

IDD_MegaPet DIALOGEX 0, 0, 271, 210
STYLE DS_SETFONT | DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "MegaPet 2.0 - 2006 C. Germany"
FONT 10, "Comic Sans MS", 400, 0, 0x0
BEGIN
    EDITTEXT        IDC_OUTPUT,9,27,154,150,ES_MULTILINE | ES_READONLY |
                    WS_VSCROLL | NOT WS_TABSTOP
```

```
    LTEXT           "",IDC_NAME,49,10,114,15,WS_BORDER,WS_EX_CLIENTEDGE
    LTEXT           "Pet Name:",IDC_STATIC,10,14,38,8
    PUSHBUTTON      "QUIT",IDC_QUIT,226,90,26,12
    EDITTEXT        IDC_INPUT,166,11,93,13,ES_AUTOHSCROLL
    PUSHBUTTON      "FEED",IDC_FEED,176,69,35,12
    PUSHBUTTON      "TALK",IDC_TALK,176,53,35,12
    PUSHBUTTON      "KICK",IDC_KICK,176,37,35,12
    PUSHBUTTON      "PET",IDC_PET,214,37,35,12
    PUSHBUTTON      "SLEEP",IDC_SLEEP,214,53,35,12
    PUSHBUTTON      "PLAY",IDC_PLAY,214,69,35,12
    GROUPBOX        "Actions",IDC_STATIC,166,25,93,62,BS_CENTER
    CONTROL         102,IDC_VIEW,"Static",SS_BITMAP,173,109,80,89
    PUSHBUTTON      "START",IDC_START,199,90,23,12
    PUSHBUTTON      "GO",IDC_GO,173,90,23,12
    CTEXT           "Life",IDC_STATIC,12,180,17,8
    CTEXT           "Emotive State",IDC_STATIC,96,179,57,8
    CTEXT           "",IDC_LIFE,9,188,26,15,WS_BORDER
    CTEXT           "",IDC_EMOTIVESTATE,87,188,77,15,WS_BORDER
    CTEXT           "Elapsed Days",IDC_STATIC,36,179,49,8
    CTEXT           "",IDC_DAYS,38,188,46,15,WS_BORDER
END


TAZ                     BITMAP                  "taz.bmp"
```

C++ Console 32 Project 1: **Adventure Game 2.0** Binary Executable: AdventureGame1.exe

The object of the game is to find the FISH key and fight enough battles to gain substantial experience. Both of these requirements must be met in order to activate the dimensional gateway to return home (or proceed to the next level). You will find other items along the way to help you defeat your foes, and you may ask the local Shaman for advice on your journey. This is a fun way to learn console programming. A graphical MFC version is here:mfcproject18.html.



The way to approach this project is to code your game in concentric rings. In the case above, the player starts the game in the CENTER() function, and may move from it to any of the other functions in ring 1 (NORTH1, SOUTH1, EAST1, WEST1). We therefore develop these functions first, after laying down our classes. When we completely finish ring 1, we can move out to coding rings 2, 3, 4 and so on. This makes the complexity of the game manageable. How do you eat an elephant? One bite at a time. Try to code more than one ring at once and you will quickly bite off more than you bargained for.

Note: This version was compiled with BloodShed 5.0, the free open source Linux-based gcc compiler. A Visual Studio 6.0 version is also included below (commented out) in the "Classes.h" file. If using .Net Visual Studio, uncomment the #include lines in "Classes.h" for .Net Visual Studio and then comment out the lines for BloodShed 5.0. Also, in "Functions.h", look for the "DisplayHighScores()" function and follow the instructions in the comment tags to make it compile with .Net Visual Studio. You only have to change "x" to "10" in two arrays.

If using the older VS 6.0, you need to patch the string include file in order for the getline() function to work properly without requiring an extra "ENTER". To do so, go to \Program Files\Microsoft Visual Studio\VC98\Include, find the file "string" (NOT "string.h"), go to line 165 in notepad, and

change _I.rdbuf()->snextc(); to _I.rdbuf()->sbumpc();. Microsoft never fixed the bug and shipped the compiler with it broken. If you are dead set against using a great free compiler (BloodShed), after you have fixed Microsoft's bug, uncomment all the includes under "//For Visual Studio 6.0" and comment out all the includes under "//For BloodShed Compiler".

Design Note: I wrote this so that the main() function has a switch statement that acts as the master controller for the entire game. Every function called returns an integer value that can be used to track the player's location. An alternative to this would be to simply make the variable "location" global, then it would not have to be declared locally in each function and returned on exiting. Making local global introduces its own set of problems, however, as it would **pollute the global namespace** more than necessary. Localizing it to each function and restricting its access from main is a bit more secure.

```
//File 1 of 3. The "main.cpp" file.  AdventureGame 2.0, C. Germany, May
27, 2006

#include "Classes.h"
#include "Functions.h"

//-----------------------------------------------------------------

int main()
{
    int location;
    Continue = true;
    InitializeGlobals();
    bool successful;

    //Create a new Player object on the heap
    Character * CurrentPlayer = new Character();
    CurrentPlayer->AskName();

    //Create a Shaman on the heap, in main on heap for continuance
    Shaman * WiseWoman = new Shaman;

    Introduction();
    location = CENTER(CurrentPlayer);

    while(Continue)
    {
        switch(location)
        {
           case N1 : location = NORTH1(CurrentPlayer); break;
           case S1 : location = SOUTH1(CurrentPlayer); break;
           case E1 : location = EAST1(CurrentPlayer); break;
           case W1 : location = WEST1(CurrentPlayer); break;
           case CENTER1 : location = CENTER(CurrentPlayer); break;
           case N2 : location = NORTH2(CurrentPlayer); break;
           case S2 : location = SOUTH2(CurrentPlayer); break;
           case E2 : location = EAST2(CurrentPlayer); break;
           case W2 : location = WEST2(CurrentPlayer); break;
           case UNDERGRND : location =
UnderGroundPassage(CurrentPlayer); break;
           case GATE : location = GateWay(CurrentPlayer); break;
           case SHAMAN : location = SHAMANHUT(CurrentPlayer, WiseWoman);
break;
```

```
            case QUIT : Continue = false; break;
            default : cout << "Not an option."; break;
         }
    }

    //Clean Up After Player Object And Save HighScore
    successful = SaveHighScores(CurrentPlayer);
    if(successful)
    {
       cout << "\n\tScore saved to \"highscores.txt\".\n\n";
    }
    else
    {
       cout << "\n\tError. Score could not be saved!\n\n";
    }

    delete CurrentPlayer; CurrentPlayer = 0;
    delete WiseWoman, WiseWoman = 0;

    cout << "\n\tExiting Hills of Darkness 2.0\n\n\n";
    cout<< "\t"; system("PAUSE");
    return 0;
}
```

```
//File 2 of 3. The "Classes.h" file.  AdventureGame 2.0, C. Germany, May
27, 2006

//Includes you need to compile with BloodShed 5.0
#include <cstdlib>
#include <string>
#include <windows.h>
#include <iostream>
#include <iomanip>
#include <fstream>


//Includes you need to compile with .Net Visual Studio or VS 6.0
//#include <ctime>
//#include <string>
//#include <windows.h>
//#include <iostream>
//#include <iomanip>
//#include <fstream>

using namespace std;

//Globals - An Enumerated Constant to Store Player Location--------
enum EVENTS{QUIT, SHAMAN, GATE, UNDERGRND, CENTER1, N1, S1, E1, W1, N2,
S2, E2, W2};
bool Continue;

//Globals to track various game events
bool W1GiantAlive;
```

```
bool E1DragonAlive;
bool S2MotleyCrewAlive;
bool FirstTimeInShamanHut;
bool CENTERFirstTime;
bool UNDERDragonPairAlive;
bool FoundHP_West2;
bool FoundHP_Shaman;


//Function Prototype Used in Classes
int Randy(int n);

//A Base class-----------------------------------------------------
class Monster
{
   public:
   Monster(int hp=20, int atk = 1, int def=1)
   {
     cout << "\n\tCreating a BASE class monster.";
     hitpoints = hp; atak = atk; defense = def;
   }
   ~Monster()
   { cout << "\n\tDestroying BASE class monster."; }


   //Accessor Methods
   void setHit(int hp) { hitpoints = hp; }
   void setAttack(int atk)  { atak = atk; }
   void setDefense(int def) { defense = def; }
   void setName(char * nm) { name = nm; }
   int getHit() { return hitpoints; }
   int getAttack()  { return atak; }
   int getDefense() { return defense; }
   char * getName() { return name; }

   private:
   int hitpoints;
   int atak;
   int defense;
   char * name;

};

//-----------------------------------------------------------------
//Base Class for Characters and Players. Would allow player
//to travel with companions and interact if expanded further.
class Character
{
   public:
   Character(int Hp = 25, int Atk = 1, int Def = 1, string nm =
"Character")
   {
     cout << "\tCreating a Character.\n";
     hitpoints = Hp; atak = Atk; defense = Def; CharName = nm; level =
0;
     InitializeInventory();
```

```
     setScore(0);
   }
   ~Character()
   { cout << "\tDestroying a Character.\n";  }

   void DisplayStats()
   {
       cout << "\n\t********************** Stats
**********************\n"
            << "\t" << CharName << ":    " << "Hit: " << hitpoints
            << "    " << "Attack: " << atak << "    "
            << "Defense: " << defense << "\n"
            <<
"\t**************************************************";
   }

   void InitializeInventory()
   {
        dagger = false;
        sword = false;
        longbow = false;
        chainmail= false;
        fullbodyarmor= false;
        healingpotion = 0;
        FishKey = false;
   }

   void Inventory()
   {
      cout << "\n\n\t\t\t**********************************\n"
           << "\t\t\t*        Current Inventory:       *\n"
           << "\t\t\t**********************************\n";
      cout << "\t\t\t*                                 *\n";
      if(dagger) { cout << "\t\t\t*  Dagger (+ 2 Attack)
*\n"; }
      if(sword) { cout << "\t\t\t*  Sword (+ 4 Attack)             *\n";
}
      if(longbow) { cout << "\t\t\t*  Long Bow (+2 over Distance)
*\n"; }
      if(chainmail) { cout << "\t\t\t*  Chain Mail (+ 2 Defense)
*\n"; }
      if(fullbodyarmor) { cout << "\t\t\t*  Full Body Armor (+4 Defense)
*\n"; }
      if(healingpotion > 0)
          {
                  cout << "\t\t\t*  Healing Potion (+20HP)  Amt: "
                   << healingpotion << " *\n";
          }
      if(FishKey) { cout << "\t\t\t*  Fish Key (Access to Objects)
*\n"; }
      if(!dagger && !sword && !longbow && !chainmail && !fullbodyarmor
&& !healingpotion && ! FishKey)
      { cout << "\t\t\t*   How sad, absolute nothing!    *\n"; }
      cout << "\t\t\t*                                 *\n";
      cout << "\t\t\t**********************************\n";
      cout << "\n";
```

```
   }

   void Attack(Monster * Opponent)
   {
      int damage;

      damage = Randy(10) + atak;

      if(dagger && UseDagger)
      { cout << "\n\t" << CharName << " stabs with the dagger!\n";
damage = damage + 2; }
      else if(sword && UseSword)
      { cout << "\n\t" << CharName << " swings the sword!\n"; damage =
damage + 4; }
      else if(longbow && UseLongBow)
      { cout << "\n\t" << CharName << " releasees the string of the
longbow!"; damage = damage + 2; }
      else { cout << "\n\t" << CharName << " engages the oponent in
brutal hand to hand combat!\n"; }

      cout << "\n\n\t***************** " << CharName << " Attacks!
*****************\n"
           << "\tBefore Attack:    " << CharName << " Hit = " <<
hitpoints  << "   "
           << "    Opponent Hit = " << Opponent->getHit()
           << "\n";

      if(damage - Opponent->getDefense() > 0)
      { damage = damage - Opponent->getDefense(); }
      else { damage = 0; }

      //Prevent negative values. Check that oponent is still alive.
      if((Opponent->getHit() - damage) > 0)
        { Opponent->setHit((Opponent->getHit() - damage)); }
      else
        { Opponent->setHit(0); }

      cout << "\tAfter Attack:     " << CharName << " Hit = " <<
hitpoints  << "   "
           << "    Opponent Hit = " << Opponent->getHit()
           << "\n";

      Sleep(3000);
   }

   void AskName()
   {
      string n;

      cout << "\n\n\n\n\n\n\n\tWhat will be your true name, player?\t";
      getline(cin, CharName);

      cout << "\n\tFrom henceforth, you shall be called: "
           << CharName << " !!!\n\n\n\n\n\n\t";

      system("PAUSE");
```

```
   }

   void Cheat()
   {
        setDagger(true); setSword(true);
        setLongBow(true); setChainMail(true);
        setFullBodyArmor(true); setHealingPotion(7);
        setFishKey(true);
   }

   void UseHealingPotion()
   {
         if(healingpotion > 0)
         { hitpoints = hitpoints + 20; DisplayStats(); healingpotion--;
}
         else { cout << "\n\tWishful thinking. You don't have any
healing potions.\n"; }
   }

   //Inventory Accessor Methods
   bool getDagger() { return dagger; }
   bool getSword() { return sword; }
   bool getLongBow() { return longbow; }
   bool getChainMail() { return chainmail; }
   bool getFullBodyArmor() { return fullbodyarmor; }
   int getHealingPotion() { return healingpotion; }
   bool getFishKey() { return FishKey; }
   void setDagger(bool x) { dagger = x; }
   void setSword(bool x) { sword = x; }
   void setLongBow(bool x) { longbow = x; }
   void setChainMail(bool x) { chainmail = x; }
   void setFullBodyArmor(bool x) { fullbodyarmor = x; }
   void setHealingPotion(int x) { healingpotion = healingpotion + x; }
   void setFishKey(bool x) { FishKey = x; }
   void setUseDagger(bool x) { UseDagger = x; }
   void setUseSword(bool x) { UseSword = x; }
   void setUseLongBow(bool x) { UseLongBow = x; }


   //Character Attribute Accessor Methods
   void setHit(int hp) { hitpoints = hp; }
   void setAttack(int atk)  { atak = atk; }
   void setDefense(int def) { defense = def; }
   void setLevel(int lvl) { level = lvl; }
   void setLocation(int loki) { location = loki; }
   void setName(string nm) { CharName = nm; }
   void setScore(int sc)  { score = sc; }
   int getHit() { return hitpoints; }
   int getAttack()  { return atak; }
   int getDefense() { return defense; }
   int getLevel() { return level; }
   int getLocation() { return location; }
   int getScore() { return score; }
   string getName() { return CharName; }

   private:
```

```
   //Character Attribute Items
   int hitpoints;
   int atak;
   int defense;
   int level;
   int location;
   string CharName;
   int score;

   //Character Inventory Items
   bool dagger;
   bool sword;
   bool longbow;
   bool UseDagger;
   bool UseSword;
   bool UseLongBow;
   bool chainmail;
   bool fullbodyarmor;
   int healingpotion;
   bool FishKey;

};

//Derives from Character-----------------------------------------

class Player : public Character
{
   public:
   Player() { cout << "\tCreating a Player.\n"; }
   ~Player(){ cout << "\tDestroying a Player.\n";  }

   //Accessor Methods
   private:

};

//---------------------------------------------------------------


//Derives from Monster-------------------------------------------

class Dragon : public Monster
{
   public:
   Dragon() { cout << "\n\tCreating a DERIVED class Dragon."; }
   ~Dragon() { cout << "\n\tDestroying a DERIVED class Dragon."; }

  void Attack(Character * Opponent)
  {
      //Since we are calling members of the base class, we use the
"this" pointer.
      int damage;
      cout << "\n\n\t***************** Dragon Attacks!
*****************\n"
           << "\tBefore Attack:    Dragon Hit = " << this->getHit() << "
"
```

```
           << Opponent->getName() <<  " Hit = "  << Opponent->getHit()
           << "\n";

      damage = Randy(10) + this->getAttack();
      if(damage > Opponent->getDefense())
         { damage = damage - Opponent->getDefense(); }
      else { damage = 0; }
      if(Opponent->getFullBodyArmor())
      { if(damage > 4) { damage = damage - 4; } }
      if(Opponent->getChainMail())
      { if(damage > 2) { damage = damage - 2; }  }

      //Prevent negative values. Check that oponent is still alive.
      if((Opponent->getHit() - damage) > 0)
        { Opponent->setHit((Opponent->getHit() - damage)); }
      else
        { Opponent->setHit(0); }

      cout << "\tAfter Attack:     Dragon Hit = " << this->getHit() << "
"
           << Opponent->getName() << " Hit = " << Opponent->getHit()
           << "\n\n";

      Sleep(3000);
  }

   void BreatheFire()
   { cout << "Breathing fire."; }

   //Accesor Methods
   void setCanFly(bool fly) { CanFly = fly; }
   bool getCanFly() { return CanFly; }

   private:
   bool CanFly;

};

//Derives from Monster-----------------------------------------------

class Giant : public Monster
{
  public:
  Giant() { cout << "\n\tCreating a DERIVED class Giant."; }
  ~Giant() { cout << "\n\tDestroying a DERIVED class Giant."; }

  void Stomp() { cout << "Stomping on object."; }
  void SwingClub() { cout << "Swing club."; }

  void Attack(Character * Opponent)
  {
      //Since we are calling members of the base class, we use the
"this" pointer.
      int damage;
      cout << "\n\n\t***************** Giant Attacks!
*****************\n"
```

```
           << "\tBefore Attack:    Giant Hit = " << this->getHit() << "
"
           << Opponent->getName() <<  " Hit = "  << Opponent->getHit()
           << "\n";

      damage = Randy(10) + this->getAttack();
      if(damage > Opponent->getDefense())
         { damage = damage - Opponent->getDefense(); }
      else { damage = 0; }
      if(Opponent->getFullBodyArmor())
      { if(damage > 4) { damage = damage - 4; } }
      if(Opponent->getChainMail())
      { if(damage > 2) { damage = damage - 2; }  }

      //Prevent negative values. Check that oponent is still alive.
      if((Opponent->getHit() - damage) > 0)
        { Opponent->setHit((Opponent->getHit() - damage)); }
      else
        { Opponent->setHit(0); }

      cout << "\tAfter Attack:     Giant Hit = " << this->getHit() << "
"
           << Opponent->getName() << " Hit = " << Opponent->getHit()
           << "\n\n";

      Sleep(3000);
  }

  //Accesor Methods
  void setClub(bool club) { HasClub = club; }
  bool getClub() { return HasClub; }

  private:
  bool HasClub;

};

//-----------------------------------------------------------------

class Shaman : public Character
{
      public:
      Shaman() { cout << "\tCreating a Shaman.\n"; }
      ~Shaman() { cout << "\tDestroying a Shaman.\n";  }

          void Talk()
      {
           int SayWhat;
           SayWhat = Randy(10);

           cout << "\n\tThe Shaman looks at you from beneath her
priestly\n"
                << "\tgarments and says,\n\n\t\"";

           switch(SayWhat)
           {
```

```
               case 1 : cout << "I like wild flowers. They are very
beautiful, and\n"
                              << "\ttheir restorative powers are merely a
fringe benefit";
                                 break;
               case 2 : cout << "Do not look at the outward appearance
of things,\n"
                              << "\tcharacter should be judged by what is
on the inside.";
                                 break;
                    case 3 : cout << "Good karma, bad karma, it's all the
same.."; break;
                    case 4 : cout << "I have a secret to tell..."; break;
                    case 5 : cout << "To unlock the gate between worlds
one needs a key"; break;
                    case 6 : cout << "You are not from this world, I see
that now"; break;
                    case 7 : cout << "You did not yet know it traveler,
but you must "
                              << "seek\nthe key of the fish god!"; break;
               case 8 : cout << "Sometimes I wish I'd never taken that
vow of chastity"; break;
               case 9 : cout << "Do you think I'm pretty? Don't judge a
book by its cover!"; break;
               case 10 : cout << "Beware the edge of the forest. Giants
are afoot"; break;
                    default : cout << "Uh oh, this should never
happen.."; break;
            } //closes switch

            cout << "\".\n\n";

    } //close talk function

      private:
      bool staff;
      bool medicinebag;
};

//---------------------------------------------------------------------
```

```
//File 3 of 3. The "Functions.h" file.  AdventureGame 2.0, C. Germany,
May 27, 2006

//Function Prototypes
void InitializeGlobals();
void Introduction();
int CENTER(Character * CurrentPlayer);
int NORTH1(Character * CurrentPlayer);
int SOUTH1(Character * CurrentPlayer);
int EAST1(Character * CurrentPlayer);
int WEST1(Character * CurrentPlayer);
int NORTH2(Character * CurrentPlayer);
```

```
int SOUTH2(Character * CurrentPlayer);
int EAST2(Character * CurrentPlayer);
int WEST2(Character * CurrentPlayer);
int SHAMANUT(Character * CurrentPlayer);
int GateWay(Character * CurrentPlayer);
int Combat(Character * p, Dragon * m, EVENTS CurrentLocation);
int Combat(Character * p, Giant * m, EVENTS CurrentLocation);
int Options(Character * CurrentPlayer, int location);
bool SaveCharacter(Character * CurrentPlayer);
bool LoadCharacter(Character * CurrentPlayer);
bool SaveHighScores(Character * CurrentPlayer);
bool DisplayHighScores();


//Function Definitions--------------------------------------------

void InitializeGlobals()
{
     cout << "\t*********  Hills of Darkness 2.0 - 2006 C. Germany
*********\n\n";
     W1GiantAlive = true;
     E1DragonAlive = true;
     S2MotleyCrewAlive = true;
     FirstTimeInShamanHut = true;
     CENTERFirstTime = true;
     UNDERDragonPairAlive = true;
         FoundHP_West2 = false;
     FoundHP_Shaman = false;
}

void Introduction()
{
     system("CLS");

     cout << "\a\n\n\t\t\t   Hills of Darkness 2.0\n\n\n";

     cout << "\tYou awake from what appears to be a disturbing dream,
knowing\n"
          << "\tneither where you've been nor how you got where you are
now.\n"
          << "\tYou slowly rise to your feet, bewildered and almost
oblivious\n"
          << "\tto the throbbing ache pulsating between your ringing
ears.\n\n"
          << "\tYou find  yourself standing on a flowing grassy knoll
amidst\n"
          << "\tthe dark green hills of medieval Scotland. In the skies
above,\n"
          << "\tdark gray clouds are passing in billowing random
patterns. It\n"
          << "\tappears a storm is approaching from the east. A few
black ravens\n"
          << "\tfly over your head towards some unknown destination, a
familiar\n"
          << "\tcaw, their cries echoing softly against the creeping
shadows.";
```

```
     cout << "\n\n\n\n\n\n\n\t";

     system("PAUSE");
     system("CLS");
}

//---------------------------------------------------------------------

int CENTER(Character * CurrentPlayer)
{
     int location;
     char choice = 'z';

     if(CENTERFirstTime)
     {
         cout << "\n\n\n"
              << "\t" << CurrentPlayer->getName()
              << ", confused, you try to get your bearings.\n\n"
              << "\tYou see nothing but large stone tablets and columns
with\n"
              << "\twhat appear to be odd and archaic symbols engraved
upon them.\n\n";

         CENTERFirstTime = false;
     }
     else
     {
         cout << "\n\n\n"
              << "\tYou return to the location where you first
mysteriously appeared\n"
              << "\tin this strange medieval world. You notice that the
large stone\n"
              << "\ttablets and columns have arranged themselves into
arch and hinged\n"
              << "\tgates. The symbols are constantly changing,
disappearing and re-\n"
              << "\tappearing at random intervals across the surface of
the tablets.\n\n";
     }
         cout << "\tYou see that you can move to the north, south, east
or west.\n"
              << "\tTo the north, you see the ruins of an ancient castle
spread\n"
              << "\tout across the horizon. To the east, you see the
lapping waves\n"
              << "\tof the ocean against a sandy shore. To the south,
you see a\n"
              << "\tsmall village with gray cobblestone houses and
smoldering\n"
              << "\tchimneys. To the west, you see an abandoned farm
house.\n";

          while( choice != 'n' &&
                 choice != 's' &&
                 choice != 'e' &&
```

```
                choice != 'w' &&
                choice != 'o' )
          {
             cout << "\n\n\tWhere will you go (n, s, e, w)?  Or will you
(t)ry the gates?\n"
                  << "\tOther: (o)ptions  (i)nventory  (d)isplay stats
(h)ealing potion   ";

             cin >> choice;

             choice = tolower(choice);

             switch(choice)
             {
               case 'n' : location = N1; break;
               case 's' : location = S1; break;
               case 'e' : location = E1; break;
               case 'w' : location = W1; break;
               case 'o' : location = Options(CurrentPlayer, CENTER1);
break;
               case 't' : if(!CENTERFirstTime)
                          { location = GATE; choice = 'o'; break; }
                          else { cout << "\tThat was an invalid
choice."; break; }
               case 'i' : CurrentPlayer->Inventory(); break;
               case '~' : CurrentPlayer->Cheat(); break;
               case '!' : CurrentPlayer->InitializeInventory(); break;
               case 'd' : CurrentPlayer->DisplayStats(); break;
               case 'h' : CurrentPlayer->UseHealingPotion(); break;
               default : cout << "\tThat was an invalid choice."; break;
             }
          }
          system("CLS");
          return location;
}

//-----------------------------------------------------------------

int NORTH1(Character * CurrentPlayer)
{
  int location;
  char choice = 'z';

  cout << "\n\n\tYou find yourself amidst the ruins of an ancient
castle...\n\n\n";
  cout << "\tFurther to the north, you see the delapidated entrance to
the\n"
       << "\tabandoned castle. It is adjoined by crumbling towers, one
at\n"
       << "\teach corner of the foundation. The entrance to the castle
is a\n"
       << "\tfrail wooden door, looking as though it had been abandoned
for\n"
       << "\tover a hundred years. It probably would not be too
difficult to\n"
       << "\tforce the door open...\n\n\n"
```

```
       << "\tAt this point, you may explore the castle ruins, or go back
to\n"
       << "\tthe SOUTH. You are surrounded by impassible castle walls
to\n"
       << "\tthe east and the west.\n\n";

          while(choice != 's' && choice != 'o' && choice !='n')
          {
             cout << "\n\n\tWhere will you go (n,s,e,w)?\n"
                  << "\tOther: (o)ptions  (i)nventory  (d)isplay stats
(h)ealing potion   ";
             cin >> choice;

             choice = tolower(choice);

             switch(choice)
             {
               case 's' : location = CENTER1; break;
               case 'o' : location = Options(CurrentPlayer, N1); break;
               case 'n' : location = N2; break;
               case 'e' : cout << "\tYou fail to scale the east
wall.\n";
                          break;
               case 'w' : cout << "\tYou press against hard, cold
stone.\n";
                          if(!CurrentPlayer->getChainMail())
                          {
                             cout << "\tYou find a suit of chain
mail!\n";
                             CurrentPlayer->setChainMail(true);
                             CurrentPlayer->Inventory();
                          }
                          else { cout << "\tYou already have the chain
mail!\n"; }
                          break;
               case 'i' : CurrentPlayer->Inventory(); break;
               case 'd' : CurrentPlayer->DisplayStats(); break;
               case 'h' : CurrentPlayer->UseHealingPotion(); break;
               default : cout << "\tThat was an invalid choice.\n";
break;
             }
          }
     system("CLS");
     return location;
}

//---------------------------------------------------------------

int SOUTH1(Character * CurrentPlayer)
{
  int location;
  char choice = 'z';

  cout << "\n\n\tYou stumble into the gates of a rustic village. You see
what\n"
```

```
       << "\tappears to be a tavern to the north. Further south, you see
a winding\n"
       << "\tdirt road that meanders towards the horizon. All around you
are\n"
       << "\tpeasants buying and selling wares in an open market place.
Near\n"
       << "\tthe center of the village several children are playing, and
in the\n"
       << "\tmidst of them sits an elderly woman, looking very wise and
thoughtful.\n\n";

  cout << "\tAt this point, you may only go back to the NORTH or further
SOUTH.\n"
       << "\tif you so desire. You are surrounded by what seems
impassible\n"
       << "\tterrain to the east and the west and several cottages.
Towards the\n"
       << "\tcenter of the villiage, you notice a small though nicely
maintained\n"
       << "\tShaman's lodge with smoke billowing from its roof.";

          while(choice != 'n' && choice != 's' && choice != 'o')
          {
             cout << "\n\n\tChoices: (n,e,s,w)  (g)o into the Shaman's
Hut\n"
                  << "\tOther: (o)ptions  (i)nventory  (d)isplay stats
(h)ealing potion   ";
             cin >> choice;

             choice = tolower(choice);

             switch(choice)
             {
               case 'n' : location = CENTER1; break;
               case 'o' : location = Options(CurrentPlayer, S1); break;
               case 's' : location = S2; break;
               case 'e' : cout << "\tYou have no right to enter someone
else's dwelling!\n";
                          break;
               case 'w' : if(!CurrentPlayer->getSword())
                          {  cout << "\nBonus!!!\n\n";
                             cout << "\tYou can not mange to ascend the
gate in front of you.\n"
                                  << "\tbut you do find a broad sword at
the base of the wall!";
                             CurrentPlayer->setSword(true);
                             CurrentPlayer->Inventory();
                          }
                          else {cout << "\tYou already took the sword!";
}
                          break;
               case 'g' : location = SHAMAN; choice = 'o'; break;
               case 'i' : CurrentPlayer->Inventory(); break;
               case 'd' : CurrentPlayer->DisplayStats(); break;
               case 'h' : CurrentPlayer->UseHealingPotion(); break;
```

```
               default : cout << "\tThat was an invalid choice.\n";
break;
             }
           }
     system("CLS");
     return location;
}

//---------------------------------------------------------------

int EAST1(Character * CurrentPlayer)
{
    int location;
    char choice = 'z';
    srand(time(NULL));  //seed for random numbers

    cout << "\n\n\tYou arrive at a sandy shore where green-blue
translucent.\n"
         << "\twaves are crashing against rocky outcroppings. To the
north,\n";

    if(E1DragonAlive)
    {
         cout << "\ta magnificent red dragon folds its wings, smoke
billowing\n"
              << "\tfrom its nostrils.";
    }
    else
    { cout << "\ta slain dragon is being devoured by ravens..."; }

    cout << "\n\n\n\tAt this point, you may only go back to the WEST.
You are\n"
         << "\tsurrounded by turbulent water and razor sharp rocks to
the\n"
         << "\teast and the south.\n\n";

          while(choice != 'w' && choice != 'e' && choice != 'o')
          {
             cout << "\n\n\tWhere will you go (n,s,e,w)?\n"
                  << "\tOther: (o)ptions  (i)nventory  (d)isplay stats
(h)ealing potion   ";
             cin >> choice;

             choice = tolower(choice);

             switch(choice)
             {
               case 'w' : location = CENTER1; break;
               case 'o' : location = Options(CurrentPlayer, E1); break;
               case 'n' : if(E1DragonAlive)
                          {  cout << "\n\n\tYou creep towards the
Dragon. Startled,"
                                  << " it climbs into the sky\n\tto
defend itself!\n";
                             Dragon Prometheus; //Example of passing
object on stack
```

```
                                location = Combat(CurrentPlayer,
&Prometheus, E1);
                                E1DragonAlive = false;
                                //break out of while true, re-invoke EAST1
to display text
                                choice = 'o';
                                break;
                             }
                             else
                             {
                                cout << "\tYou see a noble red dragon,
tragically and"
                                     << " yet recently slain...\n";
                                break;
                             }
               case 'e' : cout << "\tYou jump into the water. It's
freezing. You catch a cold.\n";
                          location = E2; break;
               case 's' : cout << "\n\tYou step on a jellyfish and it
stings you with"
                           << " its tentacles!\n";
                          if(CurrentPlayer->getHit() > 1)
                          {
                             CurrentPlayer->setHit((CurrentPlayer-
>getHit() - 1));
                             CurrentPlayer->DisplayStats();
                          }
                          else
                          {
                             CurrentPlayer->setHit(0);
                             cout << "\tHow pathetic! Slain by a
jellyfish!";
                             Continue = false;
                             location = QUIT;
                             cout << "\n\n\t"; system("PAUSE");
                          }
                          break;
               case 'i' : CurrentPlayer->Inventory(); break;
               case 'd' : CurrentPlayer->DisplayStats(); break;
               case 'h' : CurrentPlayer->UseHealingPotion(); break;
               default : cout << "\tThat was an invalid choice.\n";
break;
            }
         }
     system("CLS");
     return location;
}

//---------------------------------------------------------------------

int WEST1(Character * CurrentPlayer)
{
    int location;
    char choice = 'z';
```

```
    cout << "\n\n\tYou arrive at an abandoned farm house. You see a
picket fence,\n"
         << "\ta rustic delapidated barn, and a decaying hovel that used
to be \n"
         << "\tsomeone's residence. There are chickens walking around
the.\n"
         << "\tpremesis. To the south, you see ";

    if(W1GiantAlive)
    {
         cout << "a Giant wearing old, brown\n"
              << "\tsackcloth. He is taunting you with offensive "
              << "gestures and lewd \n\tcomments. You really don't want
to "
              << "tangle with a giant, do you?";
    }
    else
    { cout << "barbed wire, blood, sackcloth\n\tand carnage..."; }

    cout << "\n\n\n";
    cout << "\tAt this point, you may go NORTH or back EAST.\n"
         << "\tYou see only thick undergrowth and brush\n"
         << "\tto the west. To the south is that brutish giant.\n"
         << "\twith a foul mouth and a nasty disposition.\n\n\n";

         while(choice != 'e' && choice != 'w' && choice != 'o')
         {
             cout << "\n\n\tWhere will you go (n,s,e,w)?\n"
                  << "\tOther: (o)ptions  (i)nventory  (d)isplay stats
(h)ealing potion   ";
             cin >> choice;

             choice = tolower(choice);

             switch(choice)
             {
                case 'e' : location = CENTER1; break;
                case 'o' : location = Options(CurrentPlayer, W1); break;
                case 'n' : cout << "\tYou are attacked by a vicious
chicken! You can not pass.\n";
                           if(!CurrentPlayer->getDagger())
                           {
                              cout << "\tBut lieing on the ground, you
find a bronze dagger!";
                              CurrentPlayer->setDagger(true);
                              CurrentPlayer->Inventory();
                           }
                           else { cout << "\tYou already found the
dagger!\n"; }
                           break;
                case 's' : if(W1GiantAlive)
                           {  cout << "\n\n\tYou walk towards the Giant
and he charges you!";
                              Giant Bubba; //Creating and passing object
on stack
```

```
                                location = Combat(CurrentPlayer, &Bubba,
W1);
                                //If player survives, make sure they don't
fight giant
                                //by setting the global boolean to false
                                W1GiantAlive = false;
                                //break out of while true, re-invoke WEST1
to display text
                                choice = 'o';
                             }
                             else
                             {
                                cout << "\tYou stumble over the corpse of
a dead giant!\n";
                             }
                             break;
                case 'w' :  location = W2; break;
                case 'i' : CurrentPlayer->Inventory(); break;
                case 'd' : CurrentPlayer->DisplayStats(); break;
                case 'h' : CurrentPlayer->UseHealingPotion(); break;
                default : cout << "\tInvalid choice.\n"; break;
             }
          }

     system("CLS");
     return location;
}

//---------------------------------------------------------------------

int NORTH2(Character * CurrentPlayer)
{
  int location;
  char choice = 'z'; system("CLS");

  cout << "\n\n\n\n\tYou walk inside the castle. It is dark and musty,
but\n";
  cout << "\tenough daylight is leaking through the cracks in between\n"
       << "\tstones and mortar that you can ascertain your
surroundings\n"
       << "\tin a dim, colorless twilight. Against the wall to the
east\n"
       << "\tyou see a long wooden box, about the size of a coffin.
You\n"
       << "\tcan see a table, chairs and several candle stands to the
west\n"
       << "\tof the room. To the north and the south the walls are
adorned\n"
       << "\twith dusty, thread-bare tapestries. You notice stairs
descending\n"
       << "\tdeep underground to some unknown passage to your right. ";

          while(choice != 's' && choice != 'o')
          {
             cout << "\n\n\tChoices: (n,s,e,w)   (u)nderground
passage\n"
```

```
                  << "\tOther: (o)ptions  (i)nventory  (d)isplay stats
(h)ealing potion   ";
             cin >> choice;

             choice = tolower(choice);

             switch(choice)
             {
               case 's' : location = N1; break;
               case 'o' : location = Options(CurrentPlayer, N2); break;
               case 'n' : cout << "\tThe tapestries look very dry and
dusty.\n"; break;
               case 'e' : cout << "\tYou approach the box and cautiously
open it...\n";
                          if(!CurrentPlayer->getFullBodyArmor())
                          {
                             cout << "\tYou find a well preserved suit
of full body armor!\n";
                             CurrentPlayer->setFullBodyArmor(true);
                             CurrentPlayer->Inventory();
                          }
                          else { cout << "\tThe box is empty - you
already took the armor.\n"; }
                          break;
               case 'w' : cout << "\tYou press against the wall but find
nothing.\n"; break;
               case 'u' : location = UNDERGRND; choice = 'o'; break;
               case 'i' : CurrentPlayer->Inventory(); break;
               case 'd' : CurrentPlayer->DisplayStats(); break;
               case 'h' : CurrentPlayer->UseHealingPotion(); break;
               default : cout << "\tThat was an invalid choice.\n";
break;
             }
          }
     system("CLS");
     return location;
}

//---------------------------------------------------------------------

int SOUTH2(Character * CurrentPlayer)
{
  int location;
  char choice = 'z';
  char crazy;

  cout << "\n\n\tYou wander through the village further to the
south.\n";
  cout << "\tYou notice several of the villagers are staring at you\n"
       << "\tstrangely as you walk by. You come to the southern gate\n"
       << "\tthat guards the entrance to the village and pass through
its\n"
       << "\topen doors. You follow a meandering dirt path towards
the\n"
       << "\tedge of a dense hardwood forest. As you walk along the
road,\n"
```

```
      << "\tyou pass several merchants hauling their wares by horse
and\n"
      << "\tcart. Continuing south, you see a group of three giants \n"
      << "\tresting with their backs against the trees.\n\n";

         while(choice != 'n' && choice != 'o')
         {
            cout << "\n\n\tWhere will you go (n,s,e,w)?\n"
                 << "\tOther: (o)ptions  (i)nventory  (d)isplay stats
(h)ealing potion   ";
            cin >> choice;

            choice = tolower(choice);

            switch(choice)
            {
              case 's' : if(S2MotleyCrewAlive)
                         { cout << "\tYou're either really brave or
really stupid.\n";
                           cout << "\tAre you sure you want to fight 3
giants at the same time?   ";
                           cin >> crazy;
                           if(tolower(crazy) == 'y')
                           {
                              Giant * MotleyCrew = new Giant[3];
                              cout << "\tAll three giants charge you at
once!\n";
                              for(int x = 0; x < 3; x++)
                              {
                                 location = Combat(CurrentPlayer,
&MotleyCrew[x], S2);
                              }
                              //Clean up after MotleyCrew - no dangling
pointers
                              delete [] MotleyCrew; MotleyCrew = 0;
                              S2MotleyCrewAlive = false;
                           }
                           else { cout << "\tIntelligently, you decide
to walk away...\n"; }
                         }
                         else { cout << "\tYou see three dead
giants.\n"; }
                         break;
              case 'o' : location = Options(CurrentPlayer, S2); break;
              case 'n' : location = S1; break;
              case 'e' : cout << "\tYou see a lake, covered with lily
pads and algae.\n"; break;
              case 'w' : cout << "\tYou see cat tails and dragon flies
skimming across"
                               << " the water.\n"; break;
              case 'i' : CurrentPlayer->Inventory(); break;
              case 'd' : CurrentPlayer->DisplayStats(); break;
              case 'h' : CurrentPlayer->UseHealingPotion(); break;
              default : cout << "\tThat was an invalid choice.\n";
break;
            }
```

```
          }
     system("CLS");
     return location;
}

//---------------------------------------------------------------------

int EAST2(Character * CurrentPlayer)
{
  int location;
  char choice = 'z';

  cout << "\n\n\tYou swim out to the sand bars hundreds of feet beyond
the shore.\n"
       << "\tIn every direction, you see dolphins and sharks swimming
around you.\n\n";

          while(choice != 'w' && choice != 'o')
          {
             cout << "\n\n\tWhere will you go (n,s,e,w)?\n"
                  << "\tOther: (o)ptions  (i)nventory  (d)isplay stats
(h)ealing potion   ";
             cin >> choice;

             choice = tolower(choice);

             switch(choice)
             {
               case 's' : cout << "\n\tYou see dolphins and sharks.\n";
                          if(!CurrentPlayer->getFishKey())
                          {
                            cout << "\n\tYou notice something metalic
shining in the sand\n"
                                 << "\tbeneath your feet. You dig into
the sand and find\n"
                                 << "\ta bronze key with a Fish engraved
upon it.";
                            CurrentPlayer->setFishKey(true);
                          }
                          else
                          { cout << "\n\tHey, this is the same place you
found that Fish key!"; }
                          break;
               case 'o' : location = Options(CurrentPlayer, E2); break;
               case 'n' : cout << "\n\tYou see dolphins and sharks.\n";
break;
               case 'e' : cout << "\n\tYou see dolphins and sharks.\n";
break;
               case 'w' : location = E1; break;
               case 'i' : CurrentPlayer->Inventory(); break;
               case 'd' : CurrentPlayer->DisplayStats(); break;
               case 'h' : CurrentPlayer->UseHealingPotion(); break;
               default : cout << "\n\tThat was an invalid choice.\n";
break;
             }
          }
```

```
     system("CLS");
     return location;
}

//---------------------------------------------------------------------

int WEST2(Character * CurrentPlayer)
{
  int location;
  char choice = 'z';

  cout << "\n\n\tYou find yourself walking in golden fields of
wheat.\n\n";

          while(choice != 'e' && choice != 'o')
          {
             cout << "\n\n\tWhere will you go (n,s,e,w)?\n"
                  << "\tOther: (o)ptions  (i)nventory  (d)isplay stats
(h)ealing potion   ";
             cin >> choice;

             choice = tolower(choice);

             switch(choice)
             {
               case 's' : cout << "\tYou see... WHEAT!\n";
                          if(!FoundHP_West2)
                          {
                             cout << "\tBuried under a mound among the
wheat, you\n"
                                  << "\tfind a healing potion!\n";
                             CurrentPlayer->setHealingPotion(1);
                             FoundHP_West2 = true;
                             CurrentPlayer->Inventory();
                          }
                          else { cout << "\tYou already found the
healing potion!\n"; }
                          break;
               case 'o' : location = Options(CurrentPlayer, W2); break;
               case 'n' : if(!CurrentPlayer->getLongBow())
                          {
                             cout << "\tYou find a well-stringed long
bow and arrows!\n";
                             CurrentPlayer->setLongBow(true);
                             CurrentPlayer->Inventory();
                          }
                          else { cout << "\tYou already found the long
bow.\n"; }
                          break;
               case 'e' : location = W1; break;
               case 'w' : cout << "\tYou see various feed crops planted
in rows.\n"; break;
               case 'i' : CurrentPlayer->Inventory(); break;
               case 'd' : CurrentPlayer->DisplayStats(); break;
               case 'h' : CurrentPlayer->UseHealingPotion(); break;
```

```
                default : cout << "\tThat was an invalid choice.\n";
break;
             }
          }
     system("CLS");
     return location;
}
//-----------------------------------------------------------------

int SHAMANHUT(Character * CurrentPlayer, Shaman * WiseWoman)
{
  int location;
  char choice = 'z';

  if(FirstTimeInShamanHut)
  {
      cout << "\n\n\tYou duck down and enter into the Shaman's hut.
Towards the center\n"
           << "\tof the mud dwelling, beneath an overhanging shelf
descending from the\n"
           << "\tthatched roof, sits an elderly priestess. Unphased by
your presence,\n"
           << "\tshe continues to stare into the flames of a small fire
burning within\n"
           << "\ta set of blackened stone rings in the center of the
hut. Directly over\n"
           << "\ther head, an opening in the ceiling allows the smoke to
escape. She\n"
           << "\tgazes at you and cackles. \"Not expecting an old temple
preistess, \n"
           << "\twere you? Well, in this village, I'm the
\"Shaman\".\n\n";

      FirstTimeInShamanHut = false;
  }
  else
  {
      cout << "\n\n\n\tYou re-enter the Shaman's hut. She turns her head
in a peculiar\n"
           << "\tfashion and remarks, \"Back so soon, traveler?\" She
offers you a cup\n"
           << "\tof freshly brewed tea, which you gladly accept to
quench your thirst.\n\n";

           if(!FoundHP_Shaman)
           {
               cout << "\tShe opens her medicine bag and begins creating
an acrid mixture of\n"
                    << "\therbs. She pours it into a vial and places it
in your hand, saying\n"
                    << "\t\"Drink this if you become wounded, my friend.
It may restore\n"
                    << "\tyou to a measure of your former health and
constitution.\"\n\n\n\n\n";
                              CurrentPlayer->setHealingPotion(1);
```

```
                                                        FoundHP_Shaman =
true;
                              cout << "\t\t\t"; system("PAUSE");
                                                        CurrentPlayer-
>Inventory();
                              cout << "\t\t\t"; system("PAUSE");
system("CLS"); cout << "\n\n";
           }
           else
           {
                  cout << "\tYou feel a sense of debt and gratitiude
towards this\n"
                       << "\tkind old woman, remembering the healing
elixir she gave\n"
                       << "\tyou on your last visit.\n\n";
           }
  }

      cout << "\tAt this point, you may (l)eave the Shaman's hut,\n"
           << "\t(t)alk with her if you so desire, or try to (s)teal\n"
           << "\ther medicine bag and staff for what wonders they may\n"
           << "\tcontain.\n\n";

          while(choice != 'l' && choice != 's' && choice != 'o')
          {
             cout << "\n\n\tYou may: (l)eave  (t)alk  or (s)teal
things\n"
                  << "\tOther: (o)ptions  (i)nventory  (d)isplay stats
(h)ealing potion   ";
             cin >> choice;

             choice = tolower(choice);

             switch(choice)
             {
               case 'n' : cout << "\tYou run into a straw-mud wall.\n";
break;
               case 'o' : location = Options(CurrentPlayer, SHAMAN);
break;
               case 'l' : location = S1; break;
               case 's' : system("CLS");
                          cout << "\n\n\n\n\tYou get the uneasy feeling
that you are going\n"
                               << "\tto reap serious bad karma for this
unwise action.\n\n";
                          Sleep(3000);
                          cout << "\tBellowing thunder cracks and the
clouds darken as the deity\n"
                               << "\tof the temple preistess fills with
indignation and anger!\n\n";
                          Sleep(3000);
                          cout << "\tIn an instant, lighting from the
heavens strikes you\n"
                               << "\tdown!\n\n";
                          Sleep(3000);
```

```
                            cout << "\tYou go into the afterlife a loser,
ashamed\n"
                                 << "\tfor the despicable deeds you have
done. The warriors\n"
                                 << "\twho have gone on before you, the
great warriors of\n"
                                 << "\treknown and the kings of the past
will ridicule you\n"
                                 << "\tfor all of eternity for dieing
without honor.\n\n";
                                 Sleep(5000); cout << "\n\n\n\t\t\t";
system("PAUSE");
                            CurrentPlayer->setHit(0); CurrentPlayer-
>DisplayStats();
                            location = QUIT; choice = 'o';
                            break;
                case 't' : cout << "\tYou seek audience with the
preistess.\n";
                            WiseWoman->Talk(); break;
                case 'i' : CurrentPlayer->Inventory(); break;
                case 'd' : CurrentPlayer->DisplayStats(); break;
                case 'h' : CurrentPlayer->UseHealingPotion(); break;
                default : cout << "\tThat was an invalid choice.\n";
break;
             }
          }
     system("CLS");
     return location;
}

//-----------------------------------------------------------------------

int GateWay(Character * CurrentPlayer)
{
    int location;

    if(CurrentPlayer->getFishKey())
    {
        cout << "\n\n\tFumbling around the gate, you find a slot to
insert the\n"
             << "\tFish key. The tablets and columns begin to rumble and
shake.\n"
             << "\tLarge stones rise, levitating off the ground,
rearranging themselves.";
    }
    else
    {
        cout << "\n\n\tYou look around, trying every nook and crevice,
but can not\n"
             << "\tseem to find the means to open the gate, nor alter
anything else\n"
             << "\n\taround it. You see what appears to be a key hole to
one side.\n\n";

        location = CENTER1;
    }
```

```
    if(CurrentPlayer->getFishKey() && CurrentPlayer->getScore() < 50)
    {
        cout << "\n\n\tIt appears that, although you have the key, you
lack enough\n"
             << "\texperience with the ways of this world to cause the
gate to"
             << "\n\tfunction in any useful manner.\n\n\n";

        location = CENTER1;
    }

    if(CurrentPlayer->getFishKey() && CurrentPlayer->getScore() >= 50)
    {
        cout << "\n\n\tWith the experience you have gained since
entering this\n"
             << "\tstrange world, you manage to figure out the correct
sequence of\n"
             << "\tactions to perform while turning the Fish key within
the gate.\n"
             << "\tYou hear a loud hiss followed by a dull hum as
cascading beams of\n"
             << "\tlight blind you from the opening dimensional
portal.\n\n";

        cout << "\tYou feel as though you have won a series of battles
in a\n"
             << "\tlong campaign, but that the war is far from being
over.\n\n"
             << "\tHaving made several new friends and vanquished many
foes as\n"
             << "\ta soujourner in a strange land, you step through the
gates,\n"
             << "\tuncertain yet hopeful that this may bring you one
step closer\n"
             << "\tto home...";

        cout << "\n\t"; system("PAUSE");
        cout << "\tYou win this campaign and end the game with:\n\n";
        CurrentPlayer->DisplayStats(); CurrentPlayer->Inventory();
        cout << "\n\tCombat Experience Score: " << CurrentPlayer-
>getScore() << ".\n\n\n";
        location = QUIT;
    }
    cout << "\n\t"; system("PAUSE"); system("CLS");
    return location;
}

//-----------------------------------------------------------------

int UnderGroundPassage(Character * CurrentPlayer)
{
    int location;
    char nutz;
```

```
    cout << "\n\n\n\n\n\n\n\tYou descend into the darkness
underground...\n";

    if(UNDERDragonPairAlive)
    {
      cout << "\tPeeking around a corner, you see a family of red
dragons\n";
      cout << "\tbreathing fire! Think about whether you want to fight
or flee.\n";
      cout << "\tAre you sure you want to fight 5 dragons at the same
time? (y,n)   ";
      cin >> nutz;

        if(tolower(nutz) == 'y')
        {
           Dragon * DragonFamily = new Dragon[5];
           for(int x = 0; x < 5; x++)
           {
               location = Combat(CurrentPlayer, &DragonFamily[x],
UNDERGRND);
           }
           //Clean up DragonFamily objects on heap, set pointer to NULL
           delete [] DragonFamily; DragonFamily = 0;
           UNDERDragonPairAlive = false;
        }
        else
        {
          cout << "\tIntelligently, you decide to run back up the
stairs...\n";
          location = N2;
        }
    }
    else {
              cout << "\tYou see a family of dead red dragons and the
carnal aftermath.\n"
                   << "\tof your last great battle with these fierce and
worthy opponents.\n";
              location = N2;
         }
    cout << "\n\n\n\t"; system("PAUSE"); system("CLS");
    return location;
}

//-------------------------------------------------------------------------
---

int Randy(int n)
{
    int ResultRandom;
    ResultRandom = (rand()%n) + 1;
    return ResultRandom;
}

//-------------------------------------------------------------------------
//Overloaded Combat Functions, one for Dragon and one for Giant.
//Takes the global enumerated constant "EVENTS" as one of its arguments.
```

```
//---------------------------------------------------------------------
int Combat(Character * p, Dragon * m, EVENTS CurrentLocation)
{
    char choice = 'z';
    p->setUseDagger(false); p->setUseSword(false); p-
>setUseLongBow(false);
    cout << "\n\tMortal Combat!!!\n";

    while(choice != 'd' && choice != 's' && choice != 'l' && choice !=
'h')
    {
        cout << "\tYou currently have:\n";
        p->Inventory();
        cout << "\tWhat weapon will you choose to wage this battle
with?\n"
             << "\t(d)agger   (s)word   (l)ongbow   (h)and to hand
combat   ";
        cin >> choice;

        switch(choice)
        {
           case 'd' : if(p->getDagger()) { p->setUseDagger(true); }
                      else { cout << "\n\tYou don't have the
dagger!\n\n"; choice = 'z';}
                      break;
           case 's' : if(p->getSword()) { p->setUseSword(true); }
                      else { cout << "\n\tYou don't have the
sword!\n\n"; choice = 'z';}
                      break;
           case 'l' : if(p->getLongBow()) { p->setUseLongBow(true); }
                      else { cout << "\n\tYou don't have the long
bow!\n\n"; choice = 'z';}
                      break;
           case 'h' : cout << "\n\tHand to hand it is...\n\n";
                           p->setUseDagger(false); p-
>setUseSword(false);
                           p->setUseLongBow(false); break;
           default : cout << "\tThat was an invalid response.\n"; break;
        } //close switch
    } //close while true loop


    while(p->getHit() > 0 && m->getHit() > 0)
    {
       if(p->getHit() > 0) { p->Attack(m); }
       if(m->getHit() > 0) { m->Attack(p); }
    }

    if(p->getHit() <= 0)
    {
          cout << "\tYou die, most tragically!\n\n";
          cout << "\tThe Dragon wins the battle, having " << m->getHit()
               << " hitpoints left.\n";
          Continue = false;
          cout << "\t"; system("PAUSE");
          return QUIT;
```

```
    }
    else
    {
           cout << "\n\n\tYou vanquish your foe most valiantly!\n";
           cout << "\tThe Dragon died, now having " << m->getHit() << "
hitpoints.\n";
           cout << "\t" << p->getName() << " has " << p->getHit() << "
hitpoints left.\n\n";
           cout << "\tAdd 1 to your attack and 1 to your defense as a
result of\n"
                << "\tcombat experience acquired defeating this foe.\n";
           p->setScore((p->getScore() + 10));
           cout << "\n\t" << p->getName() << "'s Current Score: " << p-
>getScore() << ".\n";
           p->setAttack((p->getAttack() + 1)); p->setDefense((p-
>getDefense() + 1));
           p->DisplayStats();
           cout << "\n\n\t"; system("PAUSE"); system("CLS");
           return CurrentLocation;
    }
}

//---------------------------------------------------------------

int Combat(Character * p, Giant * m, EVENTS CurrentLocation)
{
    char choice = 'z';
    p->setUseDagger(false); p->setUseSword(false); p-
>setUseLongBow(false);
    cout << "\n\tMortal Combat!!!\n";

    while(choice != 'd' && choice != 's' && choice != 'l' && choice !=
'h')
    {
        cout << "\tYou currently have:\n";
        p->Inventory();
        cout << "\tWhat weapon will you choose to wage this battle
with?\n"
             << "\t(d)agger   (s)word   (l)ongbow   (h)and to hand
combat   ";
        cin >> choice;

        switch(choice)
        {
           case 'd' : if(p->getDagger()) { p->setUseDagger(true); }
                      else { cout << "\n\tYou don't have the
dagger!\n\n"; choice = 'z';}
                      break;
           case 's' : if(p->getSword()) { p->setUseSword(true); }
                      else { cout << "\n\tYou don't have the
sword!\n\n"; choice = 'z';}
                      break;
           case 'l' : if(p->getLongBow()) { p->setUseLongBow(true); }
                      else { cout << "\n\tYou don't have the long
bow!\n\n"; choice = 'z';}
                      break;
```

```
           case 'h' : cout << "\n\tHand to hand it is...\n\n";
                           p->setUseDagger(false); p-
>setUseSword(false);
                           p->setUseLongBow(false); break;
           default : cout << "\n\tThat was an invalid response.\n";
break;
         } //close switch
    } //close while true loop

    while(p->getHit() > 0 && m->getHit() > 0)
    {
       if(p->getHit() > 0) { p->Attack(m); }
       if(m->getHit() > 0) { m->Attack(p); }
    }

    if(p->getHit() <= 0)
    {
          cout << "\tYou die, most tragically!\n\n";
          cout << "\tThe Giant won the battle, having " << m->getHit()
               << " hitpoints left.\n";
          Continue = false;
          cout << "\t"; system("PAUSE");
          return QUIT;
    }
    else
    {
           cout << "\n\n\tYou vanquish your foe most valiantly!\n";
           cout << "\tThe Giant died, now having " << m->getHit() << "
hitpoints.\n";
           cout << "\t" << p->getName() <<" has " << p->getHit() << "
hitpoints left.\n\n";
           cout << "\tAdd 1 to your attack and 1 to your defense as a
result of\n"
                << "\tcombat experience acquired defeating this foe.\n";
           p->setScore((p->getScore() + 10));
           cout << "\n\t" << p->getName() << "'s Current Score: " << p-
>getScore() << ".\n";
           p->setAttack((p->getAttack() + 1)); p->setDefense((p-
>getDefense() + 1));
           p->DisplayStats();
           cout << "\n\n\t"; system("PAUSE"); system("CLS");
           return CurrentLocation;
    }
}

//-----------------------------------------------------------------

int Options(Character * CurrentPlayer, int location)
{
    char choice; bool successful; system("CLS"); cout << "\n\n\n\n";

    while(choice != 's' && choice != 'l' && choice != 'h' && choice !=
'q')
    {
         cout << "\n\n\t\t\t********* OPTIONS *********\n"
              << "\t\t\t*                         *\n"
```

```
              << "\t\t\t*      (s)ave game        *\n"
              << "\t\t\t*      (l)oad game        *\n"
              << "\t\t\t*      (h)igh scores      *\n"
              << "\t\t\t*      (q)uit             *\n"
              << "\t\t\t*                         *\n"
              << "\t\t\t*************************\n\n\t\t\t";

        cin >> choice;

        switch(tolower(choice))
        {
           case 's' : CurrentPlayer->setLocation(location);
                      successful = SaveCharacter(CurrentPlayer);
                      if(successful)
                      {
                          cout << "\n\tYour character was saved!\n\n\t";
                      }
                      else
                      {
                         cout << "\n\tYou character could not be
saved!\n\n\t";
                      }
                      system("PAUSE"); break;
           case 'l' : successful = LoadCharacter(CurrentPlayer);
                      if(successful)
                      {
                         location = CurrentPlayer->getLocation();
                         cout << "\n\tYour character was loaded!\n\n\t";
                      }
                      else
                      {
                           cout << "\n\tYou character could not be
loaded!\n\n\t";
                      }
                      system("PAUSE"); break;
           case 'h' : successful = DisplayHighScores();
                      if(successful) { break; }
                      else { cout << "\n\tThere are no high scores to
load!\n\n\t"; }
                      system("PAUSE");
                      break;
           case 'q' : location = QUIT; break;
           default : cout << "\n\t\t\tInvalid choice.\n"; break;
        } //close switch
    } //close while loop

   return location;
}
//---------------------------------------------------------------------
---

bool SaveCharacter(Character * CurrentPlayer)
{
     bool successful;
     char CharacterName[80];
     string passwd;
```

```
     cout << "\n\tEnter the file name to save as: ";
     cin >> CharacterName; strcat(CharacterName, ".gam");
     cout << "\n\tEnter a password:   ";
     cin >> passwd;

     ofstream WriteStuff;
     WriteStuff.open(CharacterName);

     if(!WriteStuff)
     {
         cout << "\tError. Unable to create " << CharacterName << " for
writing.\n";
         successful = false;
     }  //close if

     else
     {
         //Simple serialization of Character class
         WriteStuff << passwd << "\n";
         WriteStuff << CurrentPlayer->getName() << "\n";
         WriteStuff << CurrentPlayer->getHit() << "\n";
         WriteStuff << CurrentPlayer->getAttack() << "\n";
         WriteStuff << CurrentPlayer->getDefense() << "\n";
         WriteStuff << CurrentPlayer->getLevel() << "\n";
         WriteStuff << CurrentPlayer->getScore() << "\n";
         WriteStuff << CurrentPlayer->getLocation() << "\n";

         WriteStuff << CurrentPlayer->getDagger() << "\n";
         WriteStuff << CurrentPlayer->getSword() << "\n";
         WriteStuff << CurrentPlayer->getLongBow() << "\n";
         WriteStuff << CurrentPlayer->getChainMail() << "\n";
         WriteStuff << CurrentPlayer->getFullBodyArmor() << "\n";
         WriteStuff << CurrentPlayer->getHealingPotion() << "\n";
         WriteStuff << CurrentPlayer->getFishKey() << "\n";

         WriteStuff << W1GiantAlive << "\n";
         WriteStuff << E1DragonAlive << "\n";
         WriteStuff << S2MotleyCrewAlive << "\n";
         WriteStuff << FirstTimeInShamanHut << "\n";
         WriteStuff << CENTERFirstTime << "\n";
         WriteStuff << UNDERDragonPairAlive << "\n";
         WriteStuff << FoundHP_West2 << "\n";
         WriteStuff << FoundHP_Shaman << "\n";

         //Note: You could flush the buffer to write to the file and
keep it open
         //using  WriteStuff << flush  , but we don't need to keep it
open so:
         WriteStuff.close();
         successful = true;
     }
     return successful;
}

//-------------------------------------------------------------------------
---
```

```
bool LoadCharacter(Character * CurrentPlayer)
{
     bool successful; string nm, passwd, pass;
     int hp, atk, def, lvl, scr, loc;
     //Note: bool data type reads in VS 6.0 and .Net, but for BloodShed
you
     //must read all boolean values as integers from the file
     int dagger, sword, bow, mail, armor, healing, fkey;

     char CharacterName[80];
     cout << "\n\tPlease enter the name of character to load: ";
     cin >> CharacterName; strcat(CharacterName, ".gam");

     ifstream ReadStuff;
     ReadStuff.open(CharacterName);

     //Need to detect if successful or not to keep program from crashing
on failed load
     if (!ReadStuff)
     {
          cout << "\tUnable to find \"" << CharacterName << "\" for
reading.\n";
          successful = false;
     }
     else
     {
        cout << "\n\tEnter the password:   ";
        cin >> pass;

        getline(ReadStuff, passwd);

        if(pass == passwd)
        {
           //Careful! serialization = you must read in exactly the same
order as you wrote!
               //Have to use getline for nm since name may have spaces
in it
           getline(ReadStuff, nm); ReadStuff >> hp;
           ReadStuff >> atk; ReadStuff >> def;
           ReadStuff >> lvl; ReadStuff >> scr;
           ReadStuff >> loc;

           ReadStuff >> dagger;
           ReadStuff >> sword; ReadStuff >> bow;
           ReadStuff >> mail; ReadStuff >> armor;
           ReadStuff >> healing; ReadStuff >> fkey;

           //Note: In VS 6.0 and .Net Visul Studio, you can read these
as booleans
          //with no problem. The problem is with Bloodshed - they will
all read in
           //as false. To make it compatible with Bloodshed (therefore
all 3
           //compilers) you must read them as integers and they will be
cast to bools
```

```
           //in VS 6.0 and .Net Visual Studio. (It ain't easy getting
these 3 to agree).
           int giant, dragon, motley, shaman, center, under, HPwest2,
HPshaman;

           ReadStuff >> giant; W1GiantAlive = giant;
           ReadStuff >> dragon; E1DragonAlive = dragon;
           ReadStuff >> motley; S2MotleyCrewAlive = motley;
           ReadStuff >> shaman; FirstTimeInShamanHut = shaman;
           ReadStuff >> center; CENTERFirstTime = center;
           ReadStuff >> under; UNDERDragonPairAlive = under;
           ReadStuff >> HPwest2; FoundHP_West2 = HPwest2;
           ReadStuff >> HPshaman; FoundHP_Shaman = HPshaman;

           CurrentPlayer->setName(nm); CurrentPlayer->setHit(hp);
           CurrentPlayer->setAttack(atk); CurrentPlayer-
>setDefense(def);
           CurrentPlayer->setLevel(lvl); CurrentPlayer->setScore(scr);
           CurrentPlayer->setLocation(loc);
           CurrentPlayer->setDagger(dagger);
           CurrentPlayer->setSword(sword); CurrentPlayer-
>setLongBow(bow);
           CurrentPlayer->setChainMail(mail); CurrentPlayer-
>setFullBodyArmor(armor);
           CurrentPlayer->setHealingPotion(healing); CurrentPlayer-
>setFishKey(fkey);

           ReadStuff.close();
           successful = true;
        }
        else
        {
             cout << "\n\tThat was not the correct password!!!";
             successful = false;
        }

    }
    return successful;
}

//-------------------------------------------------------------------------
---
// Function - SaveHighScores

bool SaveHighScores(Character * CurrentPlayer)
{
     bool successful;
     ofstream highscores;
     highscores.open("highscores.txt", ios::app);

     if(!highscores)
     {
        cout << "\tError. Unable to create \"highscores\" for
writing.\n";
        successful = false;
     }
```

```
     else
     {
         highscores << CurrentPlayer->getName() << "\n"
                    << CurrentPlayer->getScore() << "\n";

         highscores.close();
         successful = true;
     }
     return successful;
}  //close function

//---------------------------------------------------------------------
---
// Function - DisplayHighScores

bool DisplayHighScores()
{
     system("CLS"); cout << "\n\n\n"; bool successful;
     string HoldMeString; int HoldMeInt, z, x = 0;

     ifstream highscores("highscores.txt");

     if (!highscores)
     {
          cout << "\tUnable to find \"highscores.txt\" for reading.\n";
          successful = false;
     }

     else
     {   //Note: When using getline() in a for loop, you must use
getline() again
         //right after reading the integer in to consume the '\n' left
in the
         //data stream. We don't have to have this extra getline() in
the
         //LoadCharacter() function because getline() is not in a loop
there.
         while(!highscores.eof())
         {   getline(highscores, HoldMeString);
             highscores >> HoldMeInt;
             getline(highscores, HoldMeString);
             x++;  //add one for every 2 lines (name and score pair)
         }
      //cout << "\t" << x; system("PAUSE"); //Use to test the value of x
      x = x - 1;   //Subtract 1 for the offset (one too many)

     //Declare 2 Parallel Arrays where # elements = # lines.
     //We could have also used only 1 array here with a structure
containing
     //a string and an integer component as a single object. Example:

     //NameAndScore { string n; int s; };  //defines structure, sort of
like a class
     //NameAndScore HIGHSCORE[x]; //creates array of objects defined as
"NameAndScore"
     // if(HIGHSCORE[r].s > HIGHSCORE[r - 1].s)
```

```
     //   { TempHIGHSCORE = HIGHSCORE[r]; HIGHSCORE[r] = HIGHSCORE[r -
1];

     //Note: If compiling this with Visual Studio .Net, it will flag the
array
     //declarations below as an error. It does not allow dynamically
sizing them
     //at run time as BloodShed does. So just change the 2 lines below:
     //from string NAMES[x]; to string NAMES[100];
     //from int SCORES[x]; to int SCORES[100];
         string NAMES[x];
         int SCORES[x];

         //Reset stream and move pointer back to beginning of file
         highscores.clear();
         highscores.seekg(0, ios::beg);

         //Put each line into
         for(z = 0; z < x; z++)
         {
             //Remember, need to use getline in case of spaces in name
             getline(highscores, NAMES[z]);
             highscores >> SCORES[z];
             getline(highscores, HoldMeString);
         }

         highscores.close();

         //Could also use (sizeof SCORES /sizeof *SCORES) to get array
size

         //Bubble Sort for High Scores. Go through each Name and Score
set
         for(int q = 0; q < x; q++)
         {
            //Compare the integer component to every other, move it if >
            //For descendind order, r starts at 1 to compare it to
element 0
            for(int r = 1; r < x; r++)
            {
               if(SCORES[r] > SCORES[r - 1])
               {
                   HoldMeInt = SCORES[r]; HoldMeString = NAMES[r];
                   SCORES[r] = SCORES[r - 1]; NAMES[r] = NAMES[r - 1];
                   SCORES[r - 1] = HoldMeInt; NAMES[r - 1] =
HoldMeString;
               }
            }
         }

        cout.setf(ios::fixed);
        cout << "\t\t***************** High Scores
*****************\n\n";
        cout << "\t\t-----------------------------------------------\n";

               for(z = 0; z < x; z++)
```

```
        {
            cout << setw(16)<< (z+1) << ". Name: " << left << setw(28)
<< NAMES[z]
                 << setw(5) << "Score: " << right << SCORES[z] << "\n";
            cout << "\t\t-----------------------------------------------
\n";
        }

               cout <<
"\n\t\t*********************************************\n";
        cout << "\n\n\t\t\t"; system("PAUSE");
        successful = true;
         }//close else
        return successful;
}

//-----------------------------------------------------------------------
---
```


C++ Console 32 Project 1: **MegaPet 1.0** Binary Executable: megapet.exe

```
//File 1 of 3. Save to a file called "Classes.h". Charles Germany, May
10, 2006

//Classes for MegaPet
//Base Class ADT

//Globals
int GlobalX;
bool CreatedPet;
int Hours;

//Prototypes
int RANDOM(int x);
void CreatePet();
void LoadPet();
void SavePet();
void Interact();

//-----------------------------------------------------------------------

class Pet
{
      enum SEX{male, female};
      enum EMOTIONS{ECSTATIC, JOYFUL, HAPPY, CONTENT, MELANCHOLY,
DEPRESSED, SUICIDAL};
      public:
```

```
      //-------------------------------------------------------------------
-----
      Pet()
      {
          cout << "\n\t\tCreating a BASE class Pet object.";
          Alive = true;  Claws = true;  LiveBirth = true;
          Gender = female;  Age = 1;  Size = 10;
          Weight = 50;  Health = 100;  Hunger = 0;
          EnergyLevel = 10;  SexDrive = 5;  EmotiveState = ECSTATIC;
          PetName = "GenericPet";
      }
      //-------------------------------------------------------------------
-----
      ~Pet() { cout << "\n\t\tDestroying a BASE class Pet object."; }
      //-------------------------------------------------------------------
-----

      //Functions
      void ChoosePet()
      {
           char Preference[10];
           cout << "\n\t\tWhat would you like to name your pet? ";
           cin >> PetName;

           cout << "\n\t\tWhat sex do you prefer? (m)ale or (f)emale ";
           cin  >> Preference;
           Preference[0] = tolower(Preference[0]);

           switch(Preference[0])
           {
               case 'm' : cout << "\n\t\tYou pick a male."; Gender =
male; break;
               case 'f' : cout << "\n\t\tYou pick a female."; Gender =
female; break;
               default : cout << "\n\t\tYou pick a female."; Gender =
female; break;
           }

           cout << "\n\t\tDo you prefer a (y)ounger pet or an (o)lder
pet? ";
           cin  >> Preference;
           Preference[0] = tolower(Preference[0]);

           switch(Preference[0])
           {
               case 'y' : cout << "\n\t\tYou get a young pet!"; Age = 1;
break;
               case 'o' : cout << "\n\t\tYou get a old pet!"; Age = 5;
break;
               default : cout << "\n\t\tUndecided? You get an older
pet!"; Age = 5; break;
           }

           cout << "\n\t\tWould you like to procure a (s)mall or (l)arge
animal?";
           cin >> Preference;
```

```
            Preference[0] = tolower(Preference[0]);
            switch(Preference[0])
            {
                case 'l' : cout << "\n\t\tYou obtain a large pet!"; Size
= 7; break;
                case 's' : cout << "\n\t\tYou obtain a small pet!"; Size
= 2; break;
                default : cout << "\n\t\tYou obtain a large pet!"; Age =
7; break;
            }

            cout << endl << endl;
            DisplayPet();
      }

      //-----------------------------------------------------------------
-----

      void DisplayPet()
      {
            if(Health > 0 && Health < 11) { EmotiveState = SUICIDAL; }
            if(Health > 10 && Health < 21) { EmotiveState = DEPRESSED; }
            if(Health > 20 && Health < 31) { EmotiveState = MELANCHOLY; }
            if(Health > 30 && Health < 51) { EmotiveState = CONTENT; }
            if(Health > 50 && Health < 71) { EmotiveState = HAPPY; }
            if(Health > 70 && Health < 91) { EmotiveState = JOYFUL; }
            if(Health > 90 && Health < 101) { EmotiveState = ECSTATIC; }

            cout << "\n\t\t------------------------ Pet Stats ----------
---------------\n";
            cout << "\t\tName: " << PetName << "  Species: " << species
<< "  Sex: ";

            if(Gender == male) { cout << "Male";  }
            else { cout << "Female"; }

            cout << "  Age: " << Age << "\n\t\tHealth: " << Health
                 << "  EnergyLevel: " << EnergyLevel << "  Hunger: "
                 << Hunger << "  Size: " << Size;
            cout << "\n\t\t-----------------------------------------------
---------------\n";

            cout << "\t\t" << PetName << " is feeling: ";

            switch(EmotiveState)
            {
                case ECSTATIC : cout << " Ecstatic!\n"; break;
                case JOYFUL : cout << " So Joyful!\n"; break;
                case HAPPY : cout << " Happy.\n"; break;
                case CONTENT : cout << " Content.\n"; break;
                case MELANCHOLY : cout << " Not too good, not too
bad.\n"; break;
                case DEPRESSED : cout << " Sad, depressed and
lonely...\n"; break;
                case SUICIDAL : cout << " Suicidal!\n"; break;
            }
```

```
            cout << "\t\t" << PetName << " is";

            switch(EnergyLevel)
            {
                case 10 : cout << " full of limitless energy.\n"; break;
                case 9 : cout << " energetic and happy.\n"; break;
                case 8 : cout << " strong and capable.\n"; break;
                case 7 : cout << " gung ho about life in general.\n";
break;
                case 6 : cout << " fairly rested.\n"; break;
                case 5 : cout << " neither tired nor abundantly
energetic.\n"; break;
                case 4 : cout << " a little tired.\n"; break;
                case 3 : cout << " very tired.\n"; break;
                case 2 : cout << " completely exhausted.\n"; break;
                case 1 : cout << " unable to stay awake - barely
conscious.\n"; break;
                case 0 : cout << " feeling like death is immenent without
sleep.\n"; break;
                default : cout << " something indescribable.\n"; break;
            }
      }
      //-----------------------------------------------------------------
-----

      void GiveLiveBirth() { cout << "\n\t\tGiving live birth! Ouch!"; }

      void Regurgitate() { cout << "\n\t\tPet regurgitating..."; }

      void Eat()
      {
            if((Hunger - 1) > 0)
            {
                cout << "\n\t\tYou feed your pet. It feels better.";
                Hunger = Hunger - 2;
                if((Health + 2) < 100) { Health = Health + 2; }
                else { Health = 100; }
            }
            else
            {
                 cout << "\n\t\tYour pet is full and does not require any
feeding.";
                 Hunger = 0;
            }
      }

      void Sleep()
      {
            if((EnergyLevel + 1) < 10)
            {
               cout << "\n\t\tYou cuase your pet to sleep, increasing its
energy.";
               EnergyLevel = EnergyLevel + 2;
               if((Health + 5) < 100) { Health = Health + 2;}
               else { Health = 100; }
```

```
            }
            else
            {
                 cout << "\n\t\tYour pet does not require any sleep.";
                 EnergyLevel = 10;
            }
      }

      void Play()
      {
            cout << "\n\t\tYou pet your pet and it feels better.";
            if(EmotiveState + 1 < 7) { EmotiveState++; }
            else { EmotiveState - 7; }

      }

      void ShowAffection() { cout << "\n\t\tYou pet your pet."; }

      void Communicate()
      {
            cout << "\n\t\tYou communicate with your pet.";
            int x = 1;

            //Keep from saying the same random statement twice
            while(GlobalX == x) { x = RANDOM(5); }
            GlobalX = x;

            switch(EmotiveState)
            {
                 case ECSTATIC : switch(x)
                                 {
                                     case 1 : cout << "\n\t\tI am leaping
with joy!"; break;
                                     case 2 : cout << "\n\t\tI am so
Happy I could cry!"; break;
                                     case 3 : cout << "\n\t\tLife
couldn't get any better!"; break;
                                     case 4 : cout << "\n\t\tI feel
great!"; break;
                                     case 5 : cout << "\n\t\tI feel
awesome!"; break;
                                     default: cout << "\n\t\tNever
Happen"; break;
                                 } break;
                 case JOYFUL :  switch(x)
                                 {
                                     case 1 : cout << "\n\t\tI am
happy."; break;
                                     case 2 : cout << "\n\t\tI feel very
good."; break;
                                     case 3 : cout << "\n\t\tLife is
good."; break;
                                     case 4 : cout << "\n\t\tNice day!";
break;
                                     case 5 : cout << "\n\t\tYour the
greatest, Master!"; break;
```

```
                                    default: cout << "\n\t\tNever
Happen"; break;
                                } break;
                case HAPPY :  switch(x)
                                {
                                    case 1 : cout << "\n\t\tI am in a
good mood!"; break;
                                    case 2 : cout << "\n\t\tLife is
wonderful!"; break;
                                    case 3 : cout << "\n\t\tYou rock,
Master!"; break;
                                    case 4 : cout << "\n\t\tOh yeah,
lovin it."; break;
                                    case 5 : cout << "\n\t\tWeeeeee!";
break;
                                    default: cout << "\n\t\tNever
Happen"; break;
                                } break;
                case CONTENT :  switch(x)
                                {
                                    case 1 : cout << "\n\t\tI am fine.";
break;
                                    case 2 : cout << "\n\t\tI am o.k.,
you're o.k."; break;
                                    case 3 : cout << "\n\t\tI am doing
alright."; break;
                                    case 4 : cout << "\n\t\tI am o.k.";
break;
                                    case 5 : cout << "\n\t\tI am getting
my needs met."; break;
                                    default: cout << "\n\t\tNever
Happen"; break;
                                } break;
                case MELANCHOLY :  switch(x)
                                {
                                    case 1 : cout << "\n\t\tThings could
be worse..."; break;
                                    case 2 : cout << "\n\t\tI was
thinking..."; break;
                                    case 3 : cout << "\n\t\tMaybe so,
maybe not..."; break;
                                    case 4 : cout << "\n\t\tI am getting
buy..."; break;
                                    case 5 : cout << "\n\t\tI'm hanging
in there..."; break;
                                    default: cout << "\n\t\tThings could
be better..."; break;
                                } break;
                case DEPRESSED :  switch(x)
                                {
                                    case 1 : cout << "\n\t\tI feel
sad."; break;
                                    case 2 : cout << "\n\t\tI don't feel
very happy."; break;
                                    case 3 : cout << "\n\t\tI feel so
depressed."; break;
```

```
                                    case 4 : cout << "\n\t\tI don't feel
good at all."; break;
                                    case 5 : cout << "\n\t\tI am not in
a good mood."; break;
                                    default: cout << "\n\t\tNever
Happen"; break;
                                } break;
                case SUICIDAL :  switch(x)
                                {
                                    case 1 : cout << "\n\t\tLife sux. I
wish I were dead!"; break;
                                    case 2 : cout << "\n\t\tI hate you
master!"; break;
                                    case 3 : cout << "\n\t\tI feel like
biting you!"; break;
                                    case 4 : cout << "\n\t\tI am going
to wizz in your shoes!"; break;
                                    case 5 : cout << "\n\t\tI am
completely miserable..."; break;
                                    default: cout << "\n\t\tNever
Happen"; break;
                                } break;
                 default : cout << "\n\t\tNever happen..."; break;

           }

           switch(Hunger)
           {
               case 0 : cout << "\n\t\t" << PetName << " is full and
content."; break;
               case 1 : cout << "\n\t\t" << PetName << " is just a
little hungry."; break;
               case 2 : cout << "\n\t\t" << PetName << " is hungry for
food"; break;
               case 3 : cout << "\n\t\t" << PetName << " is very
hungry."; break;
               case 4 : cout << "\n\t\t" << PetName << " is ravenous!";
break;
               case 5 : cout << "\n\t\t" << PetName << " is starving to
death!"; break;
               default : cout << "\n\t\tHunger value outside range.
Error!"; break;
           }
      }

      void Suffer() { cout << "\n\t\tPet suffering abuse!"; }
      void Challenge() { cout << "\n\t\tPet challenging another pet."; }

      //Public Accesor Methods
      void SetAlive(bool x) { Alive = x; }
      void SetClaws(bool x) { Claws = x; }
      void SetLiveBirth(bool x) { LiveBirth = x;}
      void SetHealth(int x) { Health = x; }
      void SetHunger(int x) { Hunger = x; }
      void SetEnergyLevel(int x) { EnergyLevel = x; }
      void SetSexDrive(int x) { SexDrive = x; }
```

```
      void SetEmotiveState(int x) { EmotiveState = x; }
      void SetAge(int x) { Age = x; }
      void SetSize(int x) { Size = x; }
      void SetWeight(int x) { Weight = x; }
      void SetGender(int x) { Gender = x; }
      void SetSpecies(string x) { species = x; }
      void SetPetName(string x) { PetName = x; }
      void SetDescription(string desc) { description = desc; }

      bool GetAlive() { return Alive; }
      bool GetClaw() { return Claws; }
      bool GetLiveBirth() { return LiveBirth; }
      int GetHealth() { return Health; }
      int GetHunger() { return Hunger; }
      int GetEnergyLevel() { return EnergyLevel; }
      int GetSexDrive() { return SexDrive; }
      int GetEmotiveState() { return EmotiveState; }
      int GetAge() { return Age; }
      int GetSize() { return Size; }
      int GetWeight() { return Weight; }
      int GetGender() { return Gender; }
      string GetSpecies() { return species; }
      string GetPetName() { return PetName; }
      string GetDescription() { return description; }

      private:
      //Private Data
      bool Alive;
      bool Claws;
      bool LiveBirth;
      int Gender;
      int Age;
      int Size;
      int Weight;
      int Health;
      int Hunger;
      int EnergyLevel;
      int SexDrive;
      int EmotiveState;
      string description;
      string species;
      string PetName;
};

//-----------------------------------------------------------------
//ADT for Mammal
class MammalMorph : public Pet
{
      public:
      MammalMorph() { cout << "\n\t\tCreating a MammalMorph object."; }
      ~MammalMorph() { cout << "\n\t\tDestroying a MammalMorph object.";
}

      //Functions

      //Public Accesor Methods
```

```
      void SetHairy(bool x) { Hairy = x; }
      bool GetHairy() { return Hairy; }

      void SpellFireBall(){  }

      private:
      //Private Data
      bool Hairy;
};

//-------------------------------------------------------------
//ADT for ReptileMorph
class ReptileMorph : public Pet
{
      public:
      ReptileMorph() {  }
      ~ReptileMorph() {  }

      //Functions
      void LayEggs() { cout << "\n\t\tLaying eggs..."; }
      //Public Accesor Methods
      int GetScaleDensity() { return ScaleDensity; }
      void SetScaleDensity(int x) { ScaleDensity = x; }

      private:
      int ScaleDensity;
      //Private Data

};

//-------------------------------------------------------------

class Liger : public MammalMorph
{
      public:
      Liger() { cout << "\n\t\tCreating a Liger."; SetSpecies("Liger");
}
      ~Liger() { cout << "\n\t\tDestroying a Liger."; }

      //Functions
      void Purr() { cout << "\n\t\tLiger purring..."; }
      void ChaseMice() { cout << "\n\t\tLiger is chasing mice..."; }
      void ScratchVindictively() { cout << "\n\t\tLiger is scratching
furniture..."; }
      void LigerScratch() { cout << "\n\t\tThe Liger lunges with deadly
claws!"; }

      //Public Accesor Methods
      bool GetLazy() { return Lazy; }
      void SetLazy(bool x) { Lazy = x; }

      private:
      //Private Data
      bool Lazy;
};
```

```
//------------------------------------------------------------

class Grog : public MammalMorph
{
      public:
      Grog() { cout << "\n\t\tCreating a Grog."; SetSpecies("Grog"); }
      ~Grog() { cout << "\n\t\tDestroying a Grog."; }

      //Functions
      void Bark() { cout << "\n\t\tGrog barking!"; }
      void WagTail() { cout << "\n\t\tTail wagging!"; }
      void ChaseLiger() { cout << "\n\t\tChasing Liger..."; }
      void ChewThings() { cout << "\n\t\tChewing something expensive.";
}
      void GrogBite() { cout << "\n\t\tThe Grog bites with crushing
pressure!"; }

      //Public Accesor Methods
      bool GetLoyalty() { return Loyal; }
      void SetLoyalty(bool x) { Loyal = x; }

      private:
      //Private Data
      bool Loyal;
};

//------------------------------------------------------------

class Snabbit : public ReptileMorph
{
      public:
      Snabbit() { cout << "\n\t\tCreating a Snabbit.";
SetSpecies("Snabbit"); }
      ~Snabbit() { cout << "\n\t\tDestroying a Snabbit."; }

      //Functions
      void ShedSkin() { }
      void UnhingeJaws() { }
      void SnabbitBite() { cout << "\n\t\tThe Snabbit lunges with
poisonous fangs!"; }

      //Public Accesor Methods
      bool GetPoisionous() { return Poisonous; }
      void SetPoisonous(bool x) { Poisonous = x; }

      private:
      //Private Data
      bool Poisonous;
};

//------------------------------------------------------------

class Kizard : public ReptileMorph
{
      public:
```

```
      Kizard() { cout << "\n\t\tCreating a Kizard.";
SetSpecies("Kizard"); }
      ~Kizard() { cout << "\n\t\tDestroying a Kizard."; }

      //Functions
      void TongueLash() { cout << "\n\t\tLashing with tongue...";  }
      void Regenerate() { cout << "\n\t\tRegenerating severed body
part..."; }
      void KizardLash() { cout << "\n\t\tTHe Kizard lashes with its
tail!"; }

      //Public Accesor Methods
      bool GetCamoSkin() { return CamoSkin; }
      void SetCamoSkin(bool x) { CamoSkin = x; }

      private:
      //Private Data
      bool CamoSkin;
};

//-----------------------------------------------------------------
```

```
//File 2 of 3. Save to a file called "Functions.h". Charles Germany, May
10, 2006

//-----------------------------------------------------------------------

int RANDOM(int x)
{  return (rand()%1) + x; }

//-----------------------------------------------------------------------

void Information()
{
  system("CLS"); cout << endl;

  cout << "\n\tIt is the year 2032. Transgenics has made many advances"
    << "\n\tin the last 10 years. Manipulating the genetic code of"
    << "\n\tcountless species, chimeras have been created for medical"
    << "\n\tresearch and organ transplantation. Thousands of new species"
    << "\n\thave emerged, not as the result of millions of years of"
    << "\n\tevolution, but as the fulfillment of humanity's whimsical"
    << "\n\tdesire for diversion. Some of these species are used as"
    << "\n\tservants, engineered to become a docile working class. They"
    << "\n\tpossess enough intelligence and human DNA to obey simple"
```

```
    << "\n\tverbal commands and follow basic routines for humanity's"
    << "\n\tunwanted, menial tasks. They now fill the positions
considered"
    << "\n\ttoo dangerous for genetically pure humans. The species
selected"
    << "\n\tfor these human/animal hybrid classes had a predisposed,"
    << "\n\tinstinctual disposition towards social behavior and include"
    << "\n\tsuch species as sheep, horses, cattle, dogs and
wolves.\n\n\n\n\t\t\t";

  system("PAUSE"); system("CLS"); cout << endl << endl;

  cout << "\n\tOther species were bred for military purposes"
    << "\n\tas \"super soldiers\" capable of much greater speed, agility"
    << "\n\tand strength on the battlefield than a genetically pure
human."
    << "\n\tThis greatly reduced human fatalities in conventional
warfare"
    << "\n\tand increased the destructive yield of battlefield
operatives."
    << "\n\tNow one hybrid soldier can do the work of 10 of its pure-
human"
    << "\n\tcounter-parts. Unfortunately, due to the battlefield's
demand"
    << "\n\tfor increased intelligence and adaptability, a developmental"
    << "\n\tbreeding \"arms race\" ensued, creating hybrids with
increasingly"
    << "\n\thigher percentages of human DNA, greater intellectual
capacity"
    << "\n\tand more advanced brain structures to facilitate autonomy
and"
    << "\n\tingenuity. As a result, despising their second-class
citizenship,"
    << "\n\tmany of these hybrids have gone AWOL and started various"
    << "\n\tunderground militias demanding rights for chimeras and
hybrids"
    << "\n\tequal to those of genetically-pure humans."
    << "\n\n\n\n\t\t\t";

  system("PAUSE"); system("CLS"); cout << endl << endl << endl;

  cout << "\n\tStill other animal hybrids were created, not by mixing human
"
    << "\n\tand animal DNA, but by mixing DNA from disparate animal
classes."
    << "\n\tThese hybrid designs were patented by \"Transgenics, LTD\""
```

```
    << "\n\tand are now sold as \"pets\" to an ever-widening customer
base"
    << "\n\tthat is experiencing continuously increasing
demand.\n\n\t\t\t"
    << "\n\tAs a researcher for \"Transgenics, LTD.\", you have been"
    << "\n\tassigned the task of breeding the company's four most"
    << "\n\tpopular genetic concoctions. They are described as follows:"
    << "\n\n\n\n\t\t\t";

  system("PAUSE"); system("CLS");

  cout << "\n\n\t1. Liger - Part lion and part tiger. Aggressive
tendencies"
    << "\n\thave been removed from this animal to mitigate the risk of "
    << "\n\tit killing and eating its owners. Nevertheless, research is"
    << "\n\tstill pending futher market investigation into the safety of"
    << "\n\tthis hybrid."
    << "\n\n\t2. Grog - The Grog's original creators wanted to combine"
    << "\n\tthe cuddliness of a bear, of which the public is so fond of"
    << "\n\taesthetically, with the social aptitude and trainability of"
    << "\n\ta dog, a species which humans have domesticated for
thousands"
    << "\n\tof years. During the development phase, Trangenic's home"
    << "\n\tsecurity division suggested giving the hybrid genetic
material"
    << "\n\tfrom a grizzly rather than a more docile black bear. Their"
    << "\n\treasoning is that the increased size and aggression
resulting"
    << "\n\tfrom the grizzly genes could be activated and employed for"
    << "\n\tself-defense purposes. Designers reasoned that the the
genetic"
    << "\n\tmaterial from the dog species would give this hybrid a"
    << "\n\tpredisposition towards obedience and offset any threat posed
to"
    << "\n\towners by the more aggressive grizzly genes. "
    << "Research is pending...\n\n\n\t\t\t";

  system("PAUSE"); system("CLS"); cout << endl << endl;

  cout << "\n\t3. Snabbit - Researchers, seeking a way to increase the"
    << "\n\trate of reproduction among certain snakes whose venom"
    << "\n\tforms a key component for many lucrative pharmaceuticals,"
    << "\n\tbegan mixing the the DNA of various snake species with"
    << "\n\tDNA from mammilian rabbits. These hybrids breed rapidly,"
    << "\n\tand although are useful for research, they have also
```

```
become"
    << "\n\tpopular as pets for the adventurous who usually have their"
    << "\n\tvenom glands removed."
    << "\n\n\t4. Kizard - A whimsical pet created by crossing the
genetic"
    << "\n\tcharacterisitics of a Kangaroo with a reptilian lizard."
    << "\n\tThese hybrids are resistant to disease, can leap up to"
    << "\n\t20 feet in the air to escape danger, and have the ability"
    << "\n\tto regenerate almost any part of their bodies that is lost"
    << "\n\tor dammaged.";

  cout << "\n\n\n\n\t\t\t";

  system("PAUSE"); system("CLS");

}

//-----------------------------------------------------------------------

void CreatePet()
{
   char choice[10];
  system("CLS");
   while(choice[0] != 'q')
  {
    cout << "\n\t\t\tDescriptions:\n\n\t\t\t"
      << "Liger = Lion + Tiger\n\t\t\t"
      << "Grog = Grizzly + Dog\n\t\t\t"
      << "Snabbit = Snake + Rabbit\n\t\t\t"
      << "Kizard = Kangaroo + Lizard\n\n";

    cout << "\n\t\t\t**** Create a Pet ****"
      << "\n\t\t\t*                    *"
      << "\n\t\t\t*    (L)iger         *"
      << "\n\t\t\t*    (G)rog          *"
      << "\n\t\t\t*    (S)nabbit       *"
      << "\n\t\t\t*    (K)izard        *"
      << "\n\t\t\t*    (I)nformation   *"
      << "\n\t\t\t*    (Q)uit          *"
      << "\n\t\t\t*                    *"
      << "\n\t\t\t**********************\n\n\t\t\t";

    cin >> choice;
    choice[0] = tolower(choice[0]);
```

```
     switch(choice[0])
    {
      case 'l' : ThePet = new Liger(); ThePet->ChoosePet();
           choice[0] = 'q'; CreatedPet = true; break;
      case 'g' : ThePet = new Grog(); ThePet->ChoosePet();
           choice[0] = 'q'; CreatedPet = true; break;
      case 's' : ThePet = new Snabbit(); ThePet->ChoosePet();
           choice[0] = 'q'; CreatedPet = true; break;
      case 'k' : ThePet = new Kizard(); ThePet->ChoosePet();
           choice[0] = 'q'; CreatedPet = true; break;
      case 'i' : Information(); break;
      case 'q' : cout << "\n\t\tExiting!\n"; break;
      default : cout << "\n\t\tSorry, that was not an option.\n"; break;
    }
  }
   if(CreatedPet) { cout << "\n\t\t"; system("PAUSE"); }
  system("CLS");
}

//---------------------------------------------------------------

void LoadPet()
{

}

//---------------------------------------------------------------

void SavePet()
{

}

//---------------------------------------------------------------

void Interact()
{
   char choice[10];

   while(choice[0] != 'q' && ThePet->GetHealth() > 0)
  {
    cout << "\n\t\tHours: " << Hours;

    ThePet->DisplayPet();
```

```
cout << "\n\n\t\t*****Interact With Pet****"
  << "\n\t\t*                          *"
  << "\n\t\t*  (F)eed      (S)leep    *"
  << "\n\t\t*  (P)et       (T)alk     *"
  << "\n\t\t*  (C)hallenge (K)ick     *"
  << "\n\t\t*  (Q)uit                 *"
  << "\n\t\t*                          *"
  << "\n\t\t**************************\n\n\t\t";

cin >> choice;
choice[0] = tolower(choice[0]);

 switch(choice[0])
{
  case 'f' : ThePet->Eat(); break;
  case 's' : ThePet->Sleep(); break;
  case 'p' : ThePet->ShowAffection(); break;
  case 'c' : ThePet->Challenge(); break;
  case 'k' : ThePet->Suffer(); break;
  case 't' : ThePet->Communicate(); break;
  case 'q' : cout << "\n\t\tExiting!\n"; break;
  default : cout << "\nSorry, that was not an option.\n";
}

 //If homeostatis - if pet is well fed and rested, increase health
 if(ThePet->GetHunger() == 0 && ThePet->GetEnergyLevel() == 10)
{
  if(ThePet->GetHealth() + 10 < 100)
  { ThePet->SetHealth(ThePet->GetHealth() + 10); }
  else { ThePet->SetHealth(100); }
}

 //Adjust Health for hunger level every 2 hours - requires feeding
 if(Hours % 2 == 0)
{
   switch(ThePet->GetHunger())
  {
     case 0 : break; //do nothing
     case 1 : break; //do nothing
     case 2 : break; //do nothing
     case 3 : if((ThePet->GetHealth() - 1) > 0)
         { ThePet->SetHealth(ThePet->GetHealth() - 1); }
          else { ThePet->SetHealth(0); }
          break;
```

```
        case 4 : if((ThePet->GetHealth() - 5) > 0)
            { ThePet->SetHealth(ThePet->GetHealth() - 5); }
             else { ThePet->SetHealth(0); }
             break;
        case 5 : if((ThePet->GetHealth() - 10) > 0)
            { ThePet->SetHealth(ThePet->GetHealth() - 10); }
             else { ThePet->SetHealth(0); }
             break;
        default : break;
      }
    }

     //Pet increases hunger each 8 hours
     if(Hours % 8 == 0)
    {
        if((ThePet->GetHunger() + 1) < 5)
       { ThePet->SetHunger(ThePet->GetHunger() + 1); }
        else { ThePet->SetHunger(5); }
    }

     //Pet grows more tired every 10 hours
     if(Hours % 10 == 0)
    {
        if((ThePet->GetEnergyLevel() - 1) >= 0)
       { ThePet->SetEnergyLevel(ThePet->GetEnergyLevel() - 1); }
        else { ThePet->SetEnergyLevel(0); }
    }


     //Adjust for energy level every 2 hours - requires sleep
     if(Hours % 2 == 0)
    {
       if(ThePet->GetEnergyLevel() > 0 && ThePet->GetEnergyLevel() < 3)
      {
         if((ThePet->GetHealth() - 5) > 0)
        { ThePet->SetHealth(ThePet->GetHealth() - 5); }
         else { ThePet->SetHealth(0); }
      }

       if(ThePet->GetEnergyLevel() > 3 && ThePet->GetEnergyLevel() < 6)
      {
         if((ThePet->GetHealth() - 2) > 0)
        { ThePet->SetHealth(ThePet->GetHealth() - 2); }
         else { ThePet->SetHealth(0); }
```

```
      }
    }
     //Advance hours to age pet
    Hours++;
  }
}

//---------------------------------------------------------------------
```

```
//File 3 of 3. Save to a file called "MegaPet1.cpp". Charles Germany, May 10,
2006

#include <cstdlib>
#include <iostream>
#include <string>
using namespace std;

#include "Classes.h"

Pet * ThePet;

#include "Functions.h"

//Globals


int main()
{
    srand(time(NULL));
    GlobalX = 0;
    Hours = 1;
    CreatedPet = false;

    cout << "\n\n\n\n\n\n\t\tWelcome to MegaPet 1.0!"
         << "\n\t\t   C. Germany - 2007\n\n\n\n\n\n\n\n\t\t";
    system("PAUSE");

    Information();

    char choice[10];

    while(choice[0] != 'q')
    {
        cout << "\n\n\t\t\t******* MegaPet Menu *******"
             << "\n\t\t\t*                           *"
             << "\n\t\t\t*    (C)reate a Pet         *"
             << "\n\t\t\t*    (L)oad a Pet           *"
             << "\n\t\t\t*    (S)ave a Pet           *"
             << "\n\t\t\t*    (I)nteract with Pet    *"
             << "\n\t\t\t*    (Q)uit                 *"
             << "\n\t\t\t*                           *"
```

```
             << "\n\t\t\t**************************\n\n\t\t\t";

        cin >> choice;
        choice[0] = tolower(choice[0]);
        system("CLS");

        switch(choice[0])
        {
            case 'c' : if(!CreatedPet) { CreatePet(); }
                       else { cout << "\n\t\tYou already created a pet!"; }
break;
            case 'l' : LoadPet(); break;
            case 's' : SavePet(); break;
            case 'i' : if(CreatedPet) { Interact(); }
                       else { cout << "\n\t\tYou need to create a pet
first!"; }
                       break;
            case 'q' : cout << "\n\t\tQuitter!"; break;
            default : cout << "\nSorry, that was not an option.\n";
        }

        if(CreatedPet)
        {
            if(ThePet->GetHealth() == 0)
            { break; }
        }
    }

    if(CreatedPet)
    {
         if(ThePet->GetHealth() == 0)
         {
             cout << "\n\n\t\tYour pet has died!";

             if(ThePet->GetHunger() >= 3)
             { cout << "\n\n\t\tIt was starving to death due to your
neglect!"; }

             if(ThePet->GetEnergyLevel() < 4)
             { cout << "\n\n\t\tIt was completely exhausted and needed
rest!"; }

             if(ThePet->GetHunger() >= 3 && ThePet->GetEnergyLevel() < 4)
             {
                 cout << "\n\n\t\tYou are especially cruel and despicable!";
                 cout << "\n\t\tYou were not worthy of a MegaPet, you
fiend!";
             }

         }

         delete ThePet;
         ThePet = 0;
    }

    cout << "\n\t\tExiting MegaPet 1.0 ...\n\n\t\t";
```

```
    cout << "\n\t\t"; system("PAUSE");
    return 0;
}
```


C++ Console 32 Project 1: **Calculator 1.0** Binary Executable: calculator1.exe

```
// File 1 of 1 - Calculator 1.0 - ©2006 C. Germany
// This calculator is not bullet proofed. It takes in put as integers
and floats,
// so if the user enters a non-numerical value, it will lock and crash
the program.
// Calculator 2.0, the project after this one, will show you how to
bullet proof
// your input and safeguard your programs against crashing due to user
error.
#include <cstdlib>
#include <iostream>

using namespace std;

// Function for All Operations
//-------------------------------------------------------------------------
------------------------

void calculate()
{
     int FirstNum;
     int SecondNum;
     int Answer;
     char choice;

     cout << "\nBasic Calculations:\n\n";
     cout << "Enter 1st number:\t";
     cin >> FirstNum;
     cout << "Enter 2nd number:\t";
     cin >> SecondNum;

     cout << "What would you like to do?\n"
          << "(a)dd\n"
          << "(s)ubtract\n"
          << "(m)ultiply\n"
          << "(d)ivide\n";

     cin >> choice;

     switch(choice)
     {
         case 'a' : Answer = FirstNum + SecondNum; break;
         case 's' : Answer = FirstNum - SecondNum; break;
         case 'm' : Answer = FirstNum * SecondNum; break;
         case 'd' : Answer = FirstNum / SecondNum; break;
         default : cout << "Invalid response."; break;
     }

     cout << "\nThe answer is " << Answer << " .\n\n\n";
}
```

```
//-------------------------------------------------------------------------
-------------------------


void Square() {

     int FirstNum;
     int Answer;
     cout << "\nSquare of Number\n";
     cout << "Enter number to be squared:\t";
     cin >> FirstNum;
     Answer = FirstNum * FirstNum;
     cout << "\nAnswer is " << Answer << " .\n";
}

//-------------------------------------------------------------------------
-------------------------
void Cube() {

     int FirstNum;
     int Answer;
     cout << "\nCube of Number\n";
     cout << "Enter number to be cubed:\t";
     cin >> FirstNum;
     Answer = FirstNum * FirstNum * FirstNum;
     cout << "\nAnswer is " << Answer << " .\n";
}

//-------------------------------------------------------------------------
-------------------------
void Convert() {

     float TempFar;
     float TempCel;
     char choice;
     cout << "What would you like to convert?\n\n";
     cout << "(F)arenheit to Celcius\n";
     cout << "(C)elcius to Farenheit\n";
     cin >> choice;
     if(choice=='F'||choice=='f')
     {
         cout << "Please enter the temperature in degrees Farenheit:\t";
          cin >> TempFar;
          TempCel = ((TempFar - 32) * 5) / 9;
          cout << "\n\n" << TempFar << " degrees Farenheit"
               << " is equal to " << TempCel << " degrees
Celcius.\n\n\n";
     }

    if(choice=='C'||choice=='c')
    {
         cout << "Please enter the temperature in degrees Celcius:\t";
         cin >> TempCel;
        TempFar = (TempCel * 9 / 5) + 32;
```

```
        cout << "\n\n" << TempCel << " degrees Celcius"
             << " is equal to " << TempFar << " degrees
Farenheit.\n\n\n";
    }
    if(choice!='F' && choice!='f' && choice!='C' && choice!='c') {
        cout << "Sorry, invalid choice.\n\n";
}

} //close function

//-------------------------------------------------------------------
-------------------------
int main()
{
     int RunProgram = 100;
     int choice;

     cout << "\n\nNOT a Bulletproof Calculator Program 1.0 - 2006 C.
Germany\n\n";
     while(RunProgram != 0)
     {
            cout << "MENU - Select an option\n\n";
            cout << "1 - Basic Calculations\n"
                 << "2 - Square a Number\n"
                 << "3 - Cube a Number\n"
                 << "4 - Convert Farenheit or Celcius\n"
                     << "0 - Quit\n";

            cout << "\nEnter your choice:\t";
            cin >> choice;
            switch(choice)
            {

                   case 0 : RunProgram = 0;
                            break;
                   case 1 : calculate();
                            break;
                   case 2 : Square();
                            break;
                   case 3 : Cube();
                            break;
                   case 4 : Convert();
                            break;
                   default : cout << "Sorry, invalid choice.";
                             break;

            } //close switch statement
        } //close while true loop on choice

        cout << "\nYou have choosen to quit.\nEnding calculator program.
Exiting...\n\n";


    system("PAUSE");
    return 0;
}
```

```
//---------------------------------------------------------------------------
-------------------------
```

Variations of the above assignment:
Objective: To demonstrate an understanding of C++ functions and decision structures.
Assignment: Re-write these functions to accept and return values.

```
// File 1 of 1 - Calculator - ©2002 C. Germany
//---------------------------------------------------------------------------------------------
-------------------------------

#include <iostream.h>

//---------------------------------------------------------------------------------------------
-------------------------------

void Multiply() {

     int FirstNum;
     int SecondNum;
     int Answer;
     cout << "\nMultiplication\n";
     cout << "Enter 1st number:\t";
     cin >> FirstNum;
     cout << "Enter 2nd number:\t";
     cin >> SecondNum;
     Answer = FirstNum * SecondNum;
     cout << "\nAnswer is " << Answer << " .\n\n\n";

}

//---------------------------------------------------------------------------------------------
-------------------------------

void Divide() {

     int FirstNum;
     int SecondNum;
     int Answer;
     cout << "\nDivision\n";
     cout << "Enter 1st number:\t";
     cin >> FirstNum;
     cout << "Enter 2nd number:\t";
     cin >> SecondNum;
     Answer = FirstNum / SecondNum;
     cout << "\nAnswer is " << Answer << " .\n\n\n";
}

//---------------------------------------------------------------------------------------------
-------------------------------

void Add() {

     int FirstNum;
     int SecondNum;
     int Answer;
     cout << "\nAddition\n";
```

```
      cout << "Enter 1st number:\t";
      cin >> FirstNum;
      cout << "Enter 2nd number:\t";
      cin >> SecondNum;
      Answer = FirstNum + SecondNum;
      cout << "\nAnswer is " << Answer << " .\n\n\n";
}

//-------------------------------------------------------------------------------------------
-------------------------------

void Subtract() {

      int FirstNum;
      int SecondNum;
      int Answer;
      cout << "\nSubtraction\n";
      cout << "Enter 1st number:\t";
      cin >> FirstNum;
      cout << "Enter 2nd number:\t";
      cin >> SecondNum;
      Answer = FirstNum - SecondNum;
      cout << "\nAnswer is " << Answer << " .\n\n\n";
}
//-------------------------------------------------------------------------------------------
-------------------------------

void Square() {

      int FirstNum;
      int Answer;
      cout << "\nSquare of Number\n";
      cout << "Enter number to be squared:\t";
      cin >> FirstNum;
      Answer = FirstNum * FirstNum;
      cout << "\nAnswer is " << Answer << " .\n";
}

//-------------------------------------------------------------------------------------------
-------------------------------

void Cube() {

      int FirstNum;
      int Answer;
      cout << "\nCube of Number\n";
      cout << "Enter number to be cubed:\t";
      cin >> FirstNum;
      Answer = FirstNum * FirstNum * FirstNum;
      cout << "\nAnswer is " << Answer << " .\n";
}

//-------------------------------------------------------------------------------------------
-------------------------------

void Convert() {

      float TempFar;
      float TempCel;
```

```
     char choice;
     cout << "What would you like to convert?\n\n";
     cout << "(F)arenheit to Celcius\n";
     cout << "(C)elcius to Farenheit\n";
     cin >> choice;
     if(choice=='F'||choice=='f')
     {
         cout << "Please enter the temperature in degrees Farenheit:\t";
         cin >> TempFar;
         TempCel = ((TempFar - 32) * 5) / 9;
         cout << "\n\n" << TempFar << " degrees Farenheit"
              << " is equal to " << TempCel << " degrees
Celcius.\n\n\n";
     }

    if(choice=='C'||choice=='c')
    {
        cout << "Please enter the temperature in degrees Celcius:\t";
        cin >> TempCel;
       TempFar = (TempCel * 9 / 5) + 32;
        cout << "\n\n" << TempCel << " degrees Celcius"
             << " is equal to " << TempFar << " degrees
Farenheit.\n\n\n";
    }
    if(choice!='F' && choice!='f' && choice!='C' && choice!='c') {
        cout << "Sorry, invalid choice.\n\n";
}
} //close function

//-------------------------------------------------------------------------------------------
-------------------------------


void main()
{
     int RunProgram = 100;
     int choice;

     cout << "\n\nCalculator Program - 2002\n\n";
     while(RunProgram != 0)
     {
            cout << "MENU - Select an option\n\n";
            cout << "1 - Multiply\n"
                 << "2 - Divide\n"
                 << "3 - Add\n"
                 << "4 - Subtract\n"
                 << "5 - Square a Number\n"
                 << "6 - Cube a Number\n"
                 << "7 - Convert Farenheit or Celcius\n"
                 << "0 - Quit\n";

            cout << "\nEnter your choice:\t";
            cin >> choice;
            switch(choice)
            {

                   case 0 : RunProgram = 0;
```

```
                        break;
                case 1 : Multiply();
                        break;
                case 2 : Divide();
                        break;
                case 3 : Add();
                        break;
                case 4 : Subtract();
                        break;
                case 5 : Square();
                        break;
                case 6 : Cube();
                        break;
                case 7 : Convert();
                        break;
                default : cout << "Sorry, invalid choice.";
                         break;

            } //close switch statement
        } //close while true loop on choice

        cout << "\nYou have choosen to quit.\nEnding calculator program.
Exiting...\n\n";

} // close main
```

Now, try rewriting your functions to take arguments and return values:

```
#include "stdafx.h"
#include <iostream.h>
int Multiply(int Num1, int Num2) {
int FirstNum;
int SecondNum;
FirstNum = Num1;
SecondNum = Num2;
cout << "\nMultiplying the numbers you previously entered.";
return FirstNum * SecondNum;
}
int Divide(int Num1, int Num2) {
int FirstNum;
int SecondNum;
FirstNum = Num1;
SecondNum = Num2;
cout << "\nDividing the numbers you previously entered.";
return FirstNum / SecondNum;
}
int Add(int Num1, int Num2) {
int FirstNum;
int SecondNum;
FirstNum = Num1;
SecondNum = Num2;
cout << "\nAdding the numbers you previously entered.";
return FirstNum + SecondNum;
}
```

```
int Subtract(int Num1, int Num2) {
int FirstNum;
int SecondNum;
FirstNum = Num1;
SecondNum = Num2;
cout << "\nSubtracting the numbers you previously entered.";
return FirstNum - SecondNum;
}
void Square() {
int FirstNum;
int Answer;
cout << "\nSquare of Number\n";
cout << "Enter number to be squared:\t";
cin >> FirstNum;
Answer = FirstNum * FirstNum;
cout << "\nAnswer is " << Answer << " .\n";
}
void Cube() {
int FirstNum;
int Answer;
cout << "\nCube of Number\n";
cout << "Enter number to be cubed:\t";
cin >> FirstNum;
Answer = FirstNum * FirstNum * FirstNum;
cout << "\nAnswer is " << Answer << " .\n";
}
void Convert() {
float TempFar;
float TempCel;
cout << "Convert: Farenheit to Celcius\n\n";
cout << "Please enter the temperature in degrees Farenheit:\t";
cin >> TempFar;
TempCel = ((TempFar - 32) * 5) /9;
cout << "\n\n" << TempFar << " degrees Farenheit"
     << " is equal to " << TempCel << " degrees Celcius.\n\n\n";
}
void main()
{
int RunProgram = 100;
int choice;
int Number1;
int Number2;
int Answer;
cout << "\n\nCalculator Program - 2002\n\n";
while (RunProgram != 0) {
cout << "First, let\'s enter 2 numbers.\n\n";
cout << "Enter the First number: ";
cin >> Number1;
cout << "Enter the Second number: ";
cin >> Number2;
cout << "\n\nThank you. Now, what do we do with them?\n\n";
cout << "MENU - Select an option\n\n";
cout << "1 - Multiply\n"
     << "2 - Divide\n"
         << "3 - Add\n"
```

```
         << "4 - Subtract\n"
         << "5 - Square a Number\n"
         << "6 - Cube a Number\n"
         << "7 - Convert Farenheit to Celcius\n"
         << "0 - Quit\n";
cout << "\nEnter your choice:\t";
cin >> choice;
switch (choice) {
case 0 : RunProgram = 0;
          break;
case 1 : Answer = Multiply(Number1, Number2);
             cout << "\n\nThe answer is: "
                          << Answer << " .\n\n";
             break;
case 2 : Answer = Divide(Number1, Number2);
             cout << "\n\nThe answer is: "
                          << Answer << " .\n\n";
             break;
case 3 : Answer = Add(Number1, Number2);
             cout << "\n\nThe answer is: "
                          << Answer << " .\n\n";
             break;
case 4 : Answer = Subtract(Number1, Number2);
             cout << "\n\nThe answer is: "
                          << Answer << " .\n\n";
             break;
case 5 : Square();
             break;
case 6 : Cube();
             break;
case 7 : Convert();
             break;
default : cout << "Sorry, invalid choice.";
              break;
} //close switch statement
       } //close while true loop on choice
cout << "\nYou have choosen to quit.\nEnding calculator program.
Exiting...\n\n";
} // close main()
```

C++ Console 32 Project 1: **Calculator 2.0** Binary Executable: calculator2.exe

```
// File 1 of 1 - Calculator 2.0 - ©2006 C. Germany. May 11, 2006
// Here we try to bulletproof Calculator 1.0 (make it more resilient
against bad input)
#include <cstdlib>
#include <iostream>

using namespace std;

//Function Prototypes
double CheckErrors(char * amountString);
void Calculate();
```

```
void Square();
void Cube();
void Convert();
void Kilometers();

//-------------------------------------------------------------------------
-------

int main()
{
     //If we take input as a char or int the user can crash the program
     //by entering a string or invalid data type. Therefore we will take
     //data as a string and switch on the 1st element in the char array.
        char choice[10] = "#";

     cout << "\n\n\tCalculator 2.0 - 2007"
                  << "\n\tFor BulletProofing Calculator 1.0 - C.
Germany\n\n";

     while(choice[0] != 'q')
     {
            cout << "\n\t**************************************\n"
                 << "\t*       MENU - Select an option      *\n"
                 << "\t*------------------------------------*\n"
                    << "\t*                                    *\n"
                 << "\t*  (B)asic Calculations              *\n"
                 << "\t*  (S)quare a Number                 *\n"
                 << "\t*  (C)ube a Number                   *\n"
                 << "\t*  (F)arenheit or Celcius Conversion *\n"
                 << "\t*  (K)ilometers or Miles Conversion  *\n"
                    << "\t*  (Q)uit                            *\n"
                    << "\t*                                    *\n"
                    << "\t**************************************\n";

            cout << "\n\tEnter your choice: ";

            cin >> choice;

            switch(tolower(choice[0]))
            {
                   case 'q' : break;
                   case 'b' : Calculate(); break;
                   case 's' : Square(); break;
                   case 'c' : Cube(); break;
                   case 'f' : Convert(); break;
                   case 'k' : Kilometers(); break;
                   default : cout << "Sorry, invalid choice."; break;
            } //close switch statement

       } //close while true loop on choice

    cout << "\n\tYou have choosen to quit.\n\tEnding calculator 2.0
program."
         <<  " Exiting...\n\n\t";

    system("PAUSE");
```

```
    return 0;
}

//Function Definitions--------------------------------------------------
-------

//---------------------------------------------------------------------
-------
//Function to correct user error (bulletproofing the calculator)
//---------------------------------------------------------------------
-------
// This function encapsulates the error checking process.
// If we take input as a float or integer, when the user enters
// a character/string, it will throw the program into an infinite
// loop and crash it.  If, on the other hand, we take input as
// a string, we can parse the string for ASCII values that relate
// to numbers and then we can handle both kinds of input.
// The first part parses numbers to the left of the decimal,
// then the second part parses numbers to the right.
// If we get a non-number ASCII value, an error message is displayed
// and the the return value is set to 0.0.

double CheckErrors(char * amountString)
{
        int x = 0;
        double THEamount = 0.0;
        cout << "\tThe string value passed in is " << amountString <<
".\n";

        //character must be between 10 and 0 to be valid
        while( (amountString[x] - 48) < 10 && (amountString[x] - 48) >=
0 )
        {
                  //Multiply by 10 to move digits to left of decimal.
                  //As each character is evaluated it will move in place
                  //value to the left.
                  THEamount = 10 * THEamount + (amountString[x] - 48);
                   x++;   //proceed to next character in the array
        }

        //If last character is a decimal point, we need to move numbers
to
       //the right instead of the left.

       if(amountString[x] == '.')
       {
                cout << "\tI detect a decimal point!\n";
                x++;             //Need to increment x to the next
character FIRST

               double factor = 1.0;    //factor needs to be
reinitialized each time!

              while((amountString[x] - 48) < 10 && (amountString[x] -
48) >= 0)
              {
```

```
                            factor = factor * 0.1;  //Decrease by factor
of 10
                            //Multiply by 0.1 to move digits to right of
decimal.
                            //As each character is evaluated it will move
in place value to the right.
                            THEamount = THEamount + (amountString[x] - 48)
* factor;
                            x++;
                }

                //If they type a non-number ASCII value after the decimal
flag
                //it as an error and return 0.0 as the value.
                 if(!(amountString[x] - 48) < 10 && (amountString[x] - 48)
>= 0)
                {
                        cout << "\tSorry, but that was not a valid
number!\n";
                        cout << "\tTherefore, the number you have entered
will be set to 1.\n";
                        THEamount = 1.0;
                        return THEamount;
                 }

          }  //close block that executes if char was a '.'

          else
          {
                    //If they type a non-number ASCII value before the
decimal flag
                    //it as an error and return 0.0 as the value.
                    if(!(amountString[x] - 48) < 10 && (amountString[x] -
48) >= 0)
                    {
                         cout << "\tSorry, but that was not a normal
number!\n";
                         cout << "\tTherefore, the strange string you
have entered will be set to 1.\n";
                         THEamount = 1.0;
                         return THEamount;
                    }
          }

          return THEamount;

} //close function

//-------------------------------------------------------------------------
--------------------
// Calculation Function for All Operations
//-------------------------------------------------------------------------
--------------------
void Calculate()
{
         char NumberString[10] = "#";
```

```
        char choice[10] = "#";
     char op;
     double FirstNum;
     double SecondNum;
     double Answer;

     while(choice[0] != 'q')
     {
          cout << "\n\tBasic Calculations:\n\n";

          cout << "\n\tWhat would you like to do?\n"
               << "\t(A)dd\n"
               << "\t(S)ubtract\n"
               << "\t(M)ultiply\n"
               << "\t(D)ivide\n"
               << "\t(Q)uit\n\n\t";

          cin >> choice; //In case they enter more that 1 char, use a
char array
          choice[0] = tolower(choice[0]); //tolower() here saves
multiple calls

          if(choice[0] == 'q') { break; } //Don't ask for numbers if
user quits

          cout << "\tEnter 1st number: ";
          cin >> NumberString;
          FirstNum = CheckErrors(NumberString);
          cout << "\tEnter 2nd number: ";
          cin >> NumberString;
          SecondNum = CheckErrors(NumberString);

          //Bulletproofing - If they enter a whole string, only accept
1st char

          switch(choice[0])
          {
              case 'a' : Answer = FirstNum + SecondNum; op = '+'; break;
              case 's' : Answer = FirstNum - SecondNum; op = '-'; break;
              case 'm' : Answer = FirstNum * SecondNum; op = 'x'; break;
              case 'd' : //Need to guard(bulletproof) against a divide
by 0 error!
                         if(SecondNum != 0)
                         { Answer = FirstNum / SecondNum; op = '/';}
                         else
                         {
                              cout << "\n\tI can't really divide by 0!
That will crash me!\n"
                                   << "\n\tAnything divided by 0 is 0, so
I will set answer to 0.\n";
                              Answer = 0;
                         }
                         break;
              default : cout << "\tInvalid response."; Answer = 0;
break;
          }
```

```
          cout << "\n\tResult: " << FirstNum << " " << op << " " <<
SecondNum
               << " = " << Answer << " .\n\n";

     }//close while true loop

}//close Calculate function

//-------------------------------------------------------------------------
--------------------

void Square() {

     char NumberString[10];
     double Num;
     double Answer;

     cout << "\n\tSquare of Number\n";
     cout << "\tEnter number to be squared: ";
     cin >> NumberString;
     Num = CheckErrors(NumberString);
     Answer = Num * Num;
     cout << "\n\tResult: " << Num << " to the power of 2 "
          << "(" << Num << " x " << Num << ")" << " is "
          << Answer << " .\n";
}

//-------------------------------------------------------------------------
--------------------

void Cube()
{
     char NumberString[10];
     double Num;
     double Answer;

     cout << "\n\tCube of Number\n";
     cout << "\tEnter number to be cubed: ";
     cin >> NumberString;
     Num = CheckErrors(NumberString);
     Answer = Num * Num * Num;
     cout << "\n\tResult: " << Num << " to the power of 3 "
          << "(" << Num << " x " << Num << " x " << Num << ")" << " is "
          << Answer << " .\n";
}

//-------------------------------------------------------------------------
--------------------

void Convert()
{
         char choice[10] = "#";
         char NumberString[10];
     double TempFar;
     double TempCel;
```

```
     while(choice[0] != 'q')
     {
         cout << "\tWhat would you like to convert?\n"
              << "\t(F)arenheit to Celcius\n"
              << "\t(C)elcius to Farenheit\n"
              << "\t(Q)uit\n\n\t";

         cin >> choice;
         choice[0] = tolower(choice[0]);

         if(choice[0] == 'q') { break; }

         if(choice[0] == 'f')
         {
                 cout << "\tPlease enter the temperature in degrees
Farenheit: ";
             cin >> NumberString;
             TempFar = CheckErrors(NumberString);

             TempCel = ((TempFar - 32) * 5) / 9;
             cout << "\n\t" << TempFar << " degrees Farenheit"
                  << " is equal to " << TempCel << " degrees
Celcius.\n\n";
         }

         if(choice[0] == 'c')
         {
            cout << "\tPlease enter the temperature in degrees Celcius:
";
            cin >> NumberString;
            TempCel = CheckErrors(NumberString);

                TempFar = (TempCel * 9 / 5) + 32;
            cout << "\n\t" << TempCel << " degrees Celcius"
                 << " is equal to " << TempFar << " degrees
Farenheit.\n\n";
         }

         if(choice[0]!='f' && choice[0]!='c' && choice[0]!='q')
         { cout << "\n\tSorry, that was an invalid choice.\n\n"; }

    }//close while true loop

} //close function

//-------------------------------------------------------------------------
--------------------

void Kilometers()
{
         char choice[10] = "#";
     char NumberString[10];
     double Kilometers;
     double Miles;
```

```
     while(choice[0] != 'q')
     {
         cout << "\tWhat would you like to convert?\n"
              << "\t(K)Kilometers to Miles\n"
              << "\t(M)iles to Kilometers\n"
              << "\t(Q)uit\n\n\t";

         cin >> choice;
         choice[0] = tolower(choice[0]);

         if(choice[0] == 'q') { break; }

         if(choice[0] == 'k')
         {
                cout << "\tPlease enter the number of Kilometers to
convert: ";
             cin >> NumberString;
             Kilometers = CheckErrors(NumberString);

             Miles = Kilometers / 1.60934;

             cout << "\n\t" << Kilometers << " kilometers"
                  << " is equal to " << Miles << " miles.\n\n";
         }

         if(choice[0] == 'm')
         {
            cout << "\tPlease enter the number of Miles to convert: ";
            cin >> NumberString;
            Miles = CheckErrors(NumberString);

                Kilometers = Miles * 1.60934;

            cout << "\n\t" << Miles << " miles"
                 << " is equal to " << Kilometers << " kilometers.\n\n";
         }

         if(choice[0]!='k' && choice[0]!='m' && choice[0]!='q')
         { cout << "\n\tSorry, that was an invalid choice.\n\n"; }

    }//close while true loop
}
```


C++ Console 32 Project 1: **MorseCodeConverter 2.0** Binary Executable: morsecode.exe

```
//Morse Code 2.0 - Charles Germany - May 11, 2006
//Uses parallel arrays to convert Morse Code to ASCII and ASCII to Morse
Code

#include <cstdlib>
#include <iostream>

#include <stdio.h>
```

```
#include <string.h>
#include <ctype.h>

using namespace std;

//Parallel Arrays below match ASCII characters to Morse Code Sequences
//One is an array of type char for single letters, the other an array of
strings.
char MorseLetter[] =
{'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R',
                      'S','T','U','V','W','X','Y','Z',' '};

char * MorseCharacterSequence[] = { ".- ","-... ","-.-. ","-.. ",".
","..-. ","--. ","....  ",
                                    ".. ",".--- ",".-.- ",".-.. ","--
","-. ","--- ",".--. ",
                                    "--.- ",".-. ","... ","- ","..-
","...- ",".-- ","-..- ",
                                    "-.-- ","--.. ","  "};

//-------------------------------------------------------------------------
-----------------

void ConvertToMorse(const char *MessageInASCII)
{
   //Note: Changed from "size_t", which is an unsigned short integer
   int i, j;
   string AccumulatedString;

   for(i = 0; MessageInASCII[i]; ++i)
   {
      //Note: The expression (sizeof ACharArray / sizeof *ACharArray)
will
      //automatically calculate the length of the array, like
array.length in JavaScript.
      for(j = 0; j < (sizeof MorseLetter /sizeof *MorseLetter); ++j)
      {
         if(toupper(MessageInASCII[i]) == MorseLetter[j])
         {
            AccumulatedString = AccumulatedString +
MorseCharacterSequence[j];
            break;
         }
      }
   }
   cout << "\n" << AccumulatedString << "\n";
}

//-------------------------------------------------------------------------
-----------------

void ConvertToASCII(const char *MessageInMorse)
{
   int i, j;
   string AccumulatedString;
```

```
   //Note: The "i++" in the for loop is intentionally left out and
updated/incremented below
   for(i = 0; MessageInMorse[i];)
   {
      for(j = 0; j < (sizeof MorseCharacterSequence / sizeof
*MorseCharacterSequence); ++j)
      {
         //strlen() =   Returns the number of characters in string before
the
         //terminating null-character. Example: int AVAriable = strlen(
const char * string );
         int MorseStringSize = strlen(MorseCharacterSequence[j]);

         //The memcmp() function compares n bytes of two regions of
memory, treating each byte
         //as an unsigned character. It returns an integer less than,
equal to, or greater than
         //zero according to whether s1 is lexicographically less than,
equal to, or greater
         //than s2.  Example: int memcmp(const void *s1, const void *s2,
size_t n);
         if ( memcmp(MorseCharacterSequence[j], &MessageInMorse[i],
MorseStringSize) == 0 )
         {
            AccumulatedString = AccumulatedString + MorseLetter[j];
            //This actually increments the i missing from the Update
section in
            //the first for loop above. We can't just increment it by
ones, it must
            //be incremented by the number of characters in a particular
morse string
            //as we sequence through each set of characters in Morse.
            i += MorseStringSize;
            break;
         } //ends if
      }
   }
   cout << "\n" << AccumulatedString << "\n";
}

//-------------------------------------------------------------------------
-----------------

void Test()
{
     char ASCIItext[] = "Hello World";
     char MorseCode[] = ".... . .-.. .-.. ---   .-- --- .-. .-.. -.. ";

     cout << "\n\n";
     cout << "Morse Code is: ";
     ConvertToMorse(ASCIItext);
     cout << "\n\n";
     cout << "ASCII Translation is: ";
     ConvertToASCII(MorseCode);
}
```

```
//------------------------------------------------------------------------
-----------------

void GetMorse()
{
   char ASCIIPhrase[100];
   cout << "\n\nEnter an ASCII string to convert to Morse Code:\t";
   cin.ignore(); //necessary to prevent getline problems
   cin.getline(ASCIIPhrase, 100);
   ConvertToMorse(ASCIIPhrase);
}
//------------------------------------------------------------------------
-----------------

void GetASCII()
{
   char MorsePhrase[100];

   cout << "\n\nImportant: Enter a Morse string to convert to ASCII. Make
sure you\n"
        << "separate each morse letter sequence with a space, separate
whole words\n"
        << "with 3 spaces, and add a space at the end before pressing
\"Enter\":\n\n";
   cin.ignore();  //necessary to prevent getline problems
   cin.getline(MorsePhrase, 100);
   ConvertToASCII(MorsePhrase);

}
//------------------------------------------------------------------------
-----------------

int main()
{
  char choice;

  cout << "\nMorse to ASCII Converter 2.0\n\n";

  while(choice != 'q')
  {
      cout << "\n\nMorse to ASCII Converter 1.0 Menu:\n\n"
           << "(Q)uit\n"
           << "(M)orse to ASCII Conversion\n"
           << "(A)SCII to Morse Conversion\n"
           << "(T)est System\n\n";

      cin >> choice;

      switch(toupper(choice))
      {
          case 'Q' : choice = 'q'; break;
          case 'M' : GetASCII(); break;
          case 'A' : GetMorse(); break;
          case 'T' : Test(); break;
          default : cout << "Invalid response."; break;
      }
```

```
  } // close while true loop

   cout << "\n\nExiting program...\n\n";
   system("PAUSE");
   return 0;

}
//--------------------------------------------------------------------------
-----------------
```



```
//Morse Code 1.0 - Charles Germany - May 11, 2006
//Uses a single array a class object with 2 public data members
//to convert Morse Code to ASCII and ASCII to Morse Code
//I like using parallel arrays (above) better.
#include <cstdlib>
#include <iostream>

#include <stdio.h>
#include <string.h>
#include <ctype.h>

using namespace std;

class AMorseObject
{
      public :
      char letter; //single char
      char *MessageInMorse;  //char array
};


//Array below holds the object defined above
AMorseObject MorseCodeArray[] =
{
   { 'A', ".- "   },{ 'B', "-... " },{ 'C', "-.-. " },{ 'D', "-.. "  },
   { 'E', ". "    },{ 'F', "..-. " },{ 'G', "--. "  },{ 'H', ".... " },
   { 'I', ".. "   },{ 'J', ".--- " },{ 'K', ".-.- " },{ 'L', ".-.. " },
   { 'M', "-- "   },{ 'N', "-. "   },{ 'O', "--- "  },{ 'P', ".--. " },
   { 'Q', "--.- " },{ 'R', ".-. "  },{ 'S', "... "  },{ 'T', "- "    },
   { 'U', "..- "  },{ 'V', "...- " },{ 'W', ".-- "  },{ 'X', "-..- " },
   { 'Y', "-.-- " },{ 'Z', "--.. " },{ ' ', "  "    },
};



void ConvertToMorse(const char *MessageInASCII)
{
   //Note: Changed from "size_t", which is an unsigned short integer
   int i, j;
   string AccumulatedString;
```

```
   for(i = 0; MessageInASCII[i]; ++i)
   {
      //Original was for( j = 0; j < sizeof MorseCodeArray / sizeof
*MorseCodeArray; ++j )
      //Note: The expression (sizeof ACharArray / sizeof *ACharArray) will
      //automatically calculate the length of the array, like array.length in
JavaScript.
      for(j = 0; j < (sizeof MorseCodeArray /sizeof *MorseCodeArray); ++j)
      {
         if(toupper(MessageInASCII[i]) == MorseCodeArray[j].letter)
         {
            //cout << MorseCodeArray[j].MessageInMorse;
            AccumulatedString = AccumulatedString +
MorseCodeArray[j].MessageInMorse;
            break;
         }
      }
   }
   cout << "\n" << AccumulatedString << "\n";
}



void ConvertToASCII(const char *MessageInMorse)
{
   int i, j;
   string AccumulatedString;

   //Note: The "i++" in the for loop is intentionally left out and
updated/incremented below
   for(i = 0; MessageInMorse[i];)
   {
      for(j = 0; j < (sizeof MorseCodeArray / sizeof *MorseCodeArray); ++j)
      {
         //Old was: size_t MorseStringSize =
strlen(MorseCodeArray[j].MessageInMorse);
         //strlen() =   Returns the number of characters in string before the
         //terminating null-character. Example: size_t  strlen ( const char *
string );
         int MorseStringSize = strlen(MorseCodeArray[j].MessageInMorse);

         //The memcmp() function compares n bytes of two regions of memory,
treating each byte
         //as an unsigned character. It returns an integer less than, equal
to, or greater than
         //zero according to whether s1 is lexicographically less than, equal
to, or greater
         //than s2.  Example: int memcmp(const void *s1, const void *s2,
size_t n);
         if ( memcmp(MorseCodeArray[j].MessageInMorse, &MessageInMorse[i],
MorseStringSize) == 0 )
         {
            AccumulatedString = AccumulatedString + MorseCodeArray[j].letter;
            //This actually increments the i missing from the Update section
in
```

```
            //the first for loop above. We can't just increment it by ones,
it must
            //be incremented by the number of characters in a particular
morse string
            //As we sequence through each set of characters in Morse
            i += MorseStringSize;
            break;
         }
      }
   }
   cout << "\n" << AccumulatedString << "\n";
}

void Test()
{
     char ASCIItext[] = "Hello World";
     char MorseCode[] = ".... . .-.. .-.. ---   .-- --- .-. .-.. -.. ";

     cout << "\n\n";
     cout << "Morse Code is: ";
     ConvertToMorse(ASCIItext);
     cout << "\n\n";
     cout << "ASCII Translation is: ";
     ConvertToASCII(MorseCode);
}

void GetMorse()
{
   char ASCIIPhrase[100];
   cout << "\n\nEnter an ASCII string to convert to Morse Code:\t";
   cin.ignore(); //necessary to prevent getline problems
   cin.getline(ASCIIPhrase, 100);
   ConvertToMorse(ASCIIPhrase);
}

void GetASCII()
{
   char MorsePhrase[100];

   cout << "\n\nImportant: Enter a Morse string to convert to ASCII. Make
sure you\n"
        << "separate each morse letter sequence with a space, separate whole
words\n"
        << "with 3 spaces, and add a space at the end before pressing
\"Enter\":\n\n";
   cin.ignore();  //necessary to prevent getline problems
   cin.getline(MorsePhrase, 100);
   ConvertToASCII(MorsePhrase);

}

int main()
{
  char choice;

  cout << "\nMorse to ASCII Converter 1.0\n\n";
```

```
  while(choice != 'q')
  {
      cout << "\n\nMorse to ASCII Converter 1.0 Menu:\n\n"
           << "(Q)uit\n"
           << "(M)orse to ASCII Conversion\n"
           << "(A)SCII to Morse Conversion\n"
           << "(T)est System\n\n";

      cin >> choice;

      switch(toupper(choice))
      {
          case 'Q' : choice = 'q'; break;
          case 'M' : GetASCII(); break;
          case 'A' : GetMorse(); break;
          case 'T' : Test(); break;
          default : cout << "Invalid response."; break;
      }

  } // close while true loop

   cout << "\n\nExiting program...\n\n";
   system("PAUSE");
   return 0;

}
```


C++ Console 32 Project 1: **Number Game** Binary Executable: NumberGame.exe

Objective: To become familiar with C++ decision structures.

```
//Number Guess 1.5 - Charles Germany - May 4, 2006

#include <iostream>
#include <time.h>

using namespace std;

//Function Prototype, has a default argument of 999
int GenerateRandomNumber(int n=999); //Example of a function prototype

//--------------------------------------------------------------------------
--------------

int main()
{
    int guess; //Get the player's guess
    int counter = 0; //Count the number of guesses
    int num;
    //const int num = 69; //The number to guess
```

```
    num = GenerateRandomNumber(5);

    cout << "\n\nNumber guessing game 1.0\n\n";

    while(counter < 3)
    {
       cout << "Guess a number between 1 and 5:\t";
       cin >> guess;

       if(guess >= num)
       {
              if(guess == num)
              {
                  cout << "Congratulations! You win. You guessed the
number!"
                       << "\nThe number was: " << num << ".\n\n\n";
                  break;
              }
              else
              {
                  cout << "The number is smaller than what you have
guessed.\n\n";
              }
       }
       else
            cout << "The number is larger than what you have guessed.\n\n";

       counter = counter + 1;
    }

    if(counter >= 3)
        cout << "Sorry. It appears you have lost the game. You only get
three guesses!"
             << "\nThe number was: " << num << ".\n\n\n";

    system("PAUSE");
    return 0;

}

//------------------------------------------------------------------------
--------------
//Function Definition
int GenerateRandomNumber(int n)
{
    int ResultRandom;
    srand(time(NULL));
    ResultRandom = (rand()%n) + 1;
    return ResultRandom;
}

//------------------------------------------------------------------------
--------------
```

C++ Console 32 Project 1: **LameMan 2.0** Binary Executable: LameMan.exe

Objective: To demonstrate an understanding of the C++ string class and its methods.
Assignment: Debug and add functions to draw a different ANSI graphic each time a letter is guessed incorrectly.

```
//LameMan 2.0, using an array of strings, 2006 - C. Germany

#include <iostream>
#include <string>
#include <ctime>

using namespace std;

//---------------------------------------------------------------------------

int RandomNumber(int n)
{
    int ResultRandom;
    srand(time(NULL));
    ResultRandom = (rand()%n) + 1;
    return ResultRandom;
}

//---------------------------------------------------------------------------

void DrawIntro()
{
     //Example of a whole ANSI man.  You could create a function to draw
a leg or arm for each
     //guess the player gets wrong. It's up to you...

     cout << "\a"; //beep

     cout << "\n\n"
          << "                                      ***                    \n"
          << "                                    *- -*                    \n"
          << "                                    * | *                    \n"
          << "                                    * _ *                    \n"
          << "                                     ***                     \n"
          << "                                      *                      \n"
          << "                                 *************               \n"
          << "                                       *                     \n"
          << "                                       *                     \n"
          << "                                      * *                    \n"
          << "                                     *   *                   \n"
          << "                                    *     *                  \n"
          << "                                   *       *                 \n"
          << "                                  *         *
\n\n";
     cout << "                              LameMan       2.0
\n\n\n";
     cout << "                               2006 C. Germany
\n\n\n\n";
```

```
     cout.flush();
     system("PAUSE");
     system("CLS");
     cout << "\a"; //beep
}

//--------------------------------------------------------------------------


int main()
{
    //Declaration of constants and initialization of locals in main()
function
    const int NumberOfWrongGuessesAllowed = 5;
    const int NumberOfWordsInBank = 27;
    int WrongGuesses = 0;

    DrawIntro();

    //Collection of possible WordBank to guess in an array of strings
    string WordBank[NumberOfWordsInBank] =
{"CAPRICIOUS","TACIT","MELANCHOLY",

"TOURNIQUET","PLAGIARISM","DECEPTIVE",

"DECEITFUL","THIEF","ARROGANT","PATHETIC",

"APOCALYPTIC","DELUSIONS","GRANDEUR",

"WORTHLESS","SCUM","REVENGE", "ENTITLEMENT",
                                          "GENERATION","WEAK",
"USELESS","FLUFFY",
                                          "BUNNY",
"HOLLOW","LIAR","TASTES","LIKE",
                                          "CHICKEN"};

    int TheChosenOne = RandomNumber(NumberOfWordsInBank);

    //After the vectors are randomly shuffled, set "TheWordToGuess" to
    //whichever ends up the first one. Subscript value for vector works
just like arrays.
    string TheWordToGuess = WordBank[TheChosenOne];

    //Make a string object the same size as "TheWordToGuess" and fill it
with the appropriate
    //number of dashes. Later, we'll replace dashes where a correct
letter is guessed.
    //The "TheWordToGuess" and "CorrectGuesses" will act as parallel
arrays, where each "-"
    // in "CorrectGuesses" will correspond to a char in
"TheWordToGuess".
    string CorrectGuesses(TheWordToGuess.size(), '-');

    string LettersGuessedAlready = "";
```

```
    cout << "\nThe rules for LameMan 2.0 are simple. Try to guess the
\nrandomly picked word."
         << " For each game, you get " << NumberOfWrongGuessesAllowed <<
" wrong\nguesses."
         << " Guess the word before exceeding " <<
NumberOfWrongGuessesAllowed
         << " wrong guesses\nand you win!"
         << " If you do not guess the word before exceeding\n"
         << NumberOfWrongGuessesAllowed << " wrong guesses, "
         << "you loose. Are you ready? Go!\n\n\n";

    //while true will break if they guess word or if they exceed number
of guesses
    while((CorrectGuesses != TheWordToGuess) && (WrongGuesses <
NumberOfWrongGuessesAllowed))
    {
        cout << "You have " << (NumberOfWrongGuessesAllowed -
WrongGuesses)
             << " incorrect guesses left.\n"
             << "This word has " << TheWordToGuess.size() << " letters
in it.\n";
        cout << "\nYou have already guessed these letters:\n";

             if(LettersGuessedAlready == "")
             {
                 cout << "Nothing to display - You haven't guessed any
letters yet!\n\n";
             }
             else
             {   cout << LettersGuessedAlready << endl; }

        cout << "\nThe result of all correctly guessed letters is:\n\n"
             << CorrectGuesses << endl;

        char guess;
        cout << "\n\nEnter your guess: ";
        cin >> guess;

        //toupper() converts characters to uppercase
        guess = toupper(guess);

        //The find() function below pertains to string objects.
        //It will return the value "string::npos" if it can not find a
        //string or char within another string. So we are saying in the
        //loop below: keep looping until player guesses a char that is
not
        //already in the string of dashes "CorrectGuesses". In this way,
we
        //will not penalize them for guessing a correct char twice.
        while(LettersGuessedAlready.find(guess) != string::npos)
        {
            cout << "\n" << guess << "? You've already guessed that
letter!"
                 << endl;
            cout << "\nLook above and make sure you know what you
already guessed."
```

```
                 << "\nNow enter a fresh guess this time: ";
            cin >> guess;
            guess = toupper(guess);
        } //close nested inner while true

        //If it hasn't already been done, add it to letters guessed
already
        //The while true loop above keeps multiple instances of the same
letter
        //from being added.
        LettersGuessedAlready += guess;

        //If the char guessed is in the word, say so.
        if(TheWordToGuess.find(guess) != string::npos)
        {
            cout << "\nThat's right! " << guess << " is in the
word.\n\n";

            //The for loop below will sequence every char in the string
of "TheWordToGuess"
            //to see if it finds the char guessed. If so, it will
replace the dash in
            //"CorrectGuesses" with the letter.
            //Note: The length() function automatically determines the
length of a string.
            for(int i = 0; i < TheWordToGuess.length(); ++i)
            {
                if(TheWordToGuess[i] == guess)
                {  CorrectGuesses[i] = guess;  }
            } //close for loop
        } //close if
        else
        {
            cout << "\nSorry, " << guess << " isn't in the word.\n\n";
            ++WrongGuesses;
        }
    } //close outer while true loop

    //Win or loose?
    if (WrongGuesses == NumberOfWrongGuessesAllowed)
        cout << "\nYou are so lame. Your number of wrong guesses has "
             << "exceeded the limit.\nThis means you loose the game.\n";
    else
        cout << "\nNice. You did it! You guessed the entire word!";

    cout << "\nThe chosen word was \"" << TheWordToGuess << "\".\n\n";

    cout << "Exiting program.\n\n";
    system("PAUSE");
    return 0;
}
```

```
//LameMan, with vectors already functioned and sorted for you instead
//of arrays. Why not instead, try building this without the vectors?

#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <ctime>
#include <cctype>

using namespace std;

int main()
{
    //Initialize values here on start
    int NumberOfGuessesAllowed = 10;
    int WrongGuesses = 0;

    cout << "LameMan 1.0\n";

    //Wussy string vectors hold collection of possible words to guess
    vector<string> WordBank;
    WordBank.push_back("CAPRICIOUS");
    WordBank.push_back("TACIT");
    WordBank.push_back("APOCALYPTIC");
    WordBank.push_back("MELANCHOLY");
    WordBank.push_back("TOURNIQUET");
    WordBank.push_back("PLAGIARISM");
    WordBank.push_back("DECEPTIVE");
    WordBank.push_back("DECEITFUL");
    WordBank.push_back("THIEF");
    WordBank.push_back("ARROGANT");
    WordBank.push_back("PATHETIC");
    WordBank.push_back("DELUSIONS");
    WordBank.push_back("GRANDEUR");
    WordBank.push_back("WORTHLESS");
    WordBank.push back("SCUM");
    WordBank.push_back("REVENGE");

    //srand() seeds random number generator with system time value
        srand(time(0));
        //random shuffle is a built-in vector function, randomly shuffles
string vectors
    random_shuffle(WordBank.begin(), WordBank.end());

    //After the vectors are randomly shuffled, set "TheWordToGuess" to
    //whichever ends up the first one. Subscript value for vector works
just like arrays.
    string TheWordToGuess = WordBank[0];

    //Make a string object the same size as "TheWordToGuess" and fill it
with the appropriate
    //number of dashes. Later, we'll replace dashes where a correct
letter is guessed.
    //The "TheWordToGuess" and "CorrectGuesses" will act as parallel
arrays, where each "-"
```

```
    // in "CorrectGuesses" will correspond to a char in
"TheWordToGuess".
    string CorrectGuesses(TheWordToGuess.size(), '-');

    string LettersGuessedAlready = "";

    //while true will break if they guess word or if they exceed number
of guesses
    while((CorrectGuesses != TheWordToGuess) && (WrongGuesses <
NumberOfGuessesAllowed))
    {
        cout << "\n\nYou have only " << (NumberOfGuessesAllowed -
WrongGuesses)
             << " incorrect guesses left.\n";
        cout << "\nYou have already guessed these letters:\n";

             if(LettersGuessedAlready == "")
             {
                 cout << "Nothing to display - You haven't guessed any
letters yet!\n\n";
             }
             else
             {   cout << LettersGuessedAlready << endl; }

        cout << "\nThe result of all correctly guessed letters is:\n\n"
             << CorrectGuesses << endl;

        char guess;
        cout << "\n\nEnter your guess: ";
        cin >> guess;

        //toupper() converts characters to uppercase
        guess = toupper(guess);

        //The find() function below pertains to string objects.
        //It will return the value "string::npos" if it can not find a
        //string or char within another string. So we are saying in the
        //loop below: keep looping until player guesses a char that is
not
        //already in the string of dashes "CorrectGuesses". In this way,
we
        //will not penalize them for guessing a correct char twice.
        while(LettersGuessedAlready.find(guess) != string::npos)
        {
            cout << "\nYou've already guessed that letter! Guess again."
                 << guess << endl;
            cout << "Look above and make sure you know what you already
guessed."
                 << "Now enter a fresh guess this time: ";
            cin >> guess;
            guess = toupper(guess);
        } //close nested inner while true

        //If it hasn't already been done, add it to letters guessed
already
```

```
        //The while true loop above keeps multiple instances of the same
letter
        //from being added.
        LettersGuessedAlready += guess;

        //If the char guessed is in the word, say so.
        if(TheWordToGuess.find(guess) != string::npos)
        {
            cout << "That's right! " << guess << " is in the word.\n";

            //The for loop below will sequence every char in the string
of "TheWordToGuess"
            //to see if it finds the char guessed. If so, it will
replace the dash in
            //"CorrectGuesses" with the letter.
            //Note: The length() function automatically determines the
length of a string.
            for(int i = 0; i < TheWordToGuess.length(); ++i)
            {
                if(TheWordToGuess[i] == guess)
                {  CorrectGuesses[i] = guess;  }
            } //close for loop
        } //close if
        else
        {
            cout << "Sorry, " << guess << " isn't in the word.\n";
            ++WrongGuesses;
        }
    } //close outer while true loop

    //Win or loose?
    if (WrongGuesses == NumberOfGuessesAllowed)
        cout << "\nYou are so lame. Number of wrong guesses has exceeded
limit.";
    else
        cout << "\nNice. You guessed it! You win. Yay.";

    cout << "\nThe word was " << TheWordToGuess << endl;

    system("PAUSE");
    return 0;
}
```

Notes:

```
    Notes:
    syntax of compare:

    Examples:
    x = stringObjectName.compare(x, y, z);
    x = stringObjectName.compare(x, y, "Charles");
    x = stringObjectName.compare(x, y, "32");
    x = position to start at
    y = number of letters/elements to compare
    z = the string, text, or letters to compare
```

```
    Returns:
    0 if characters are equal
    -1 if characters come before z
    1 if characters come after z

    syntax of find:
    x = stringObjectName.find("monster", 0);
    Searches string for "monster" starting at first character
    x = stringObjectName.find("employee", 5);
    Searches string for "employee" starting at 6th character.
    Returns:
    >=0 is true
    -1 (less than 0) = false
```


C++ Console 32 Project 1: **GuessAWord 1.0**

Binary Executable: guessaword.exe

```
// GuessAWord 1.0 - Charles Germany, May 14, 2006

#include <iostream>
#include <string>
#include <ctime>

using namespace std;

//Global variables outside of main()
const int NumberOfWords = 15;
const int NumberOfWrongGuesses = 10;
const int NumWordsToGuess = 3;
int MakeSureTheyAreNotTheSame;
bool WorthyOfAClue;
char CONTINUE;

int randy()
{
        int TheChosenOne;
    srand(time(0));
        TheChosenOne = (rand() % NumberOfWords);
        return TheChosenOne;
}

int main()
{
    CONTINUE = 'y';

    while(CONTINUE == 'y')
    {
        //Initializing and declaring variables local to main()
        MakeSureTheyAreNotTheSame = 0;
        WorthyOfAClue = false;
```

```
        int counter = 0;
        int WordsGuessed = 0;
        int TheChosenOne = 0;
        int Randy1 = 0, Randy2 = 0;
        int TheSizeOfMixItAllUp = 0;
        char temp = ' ';
        string UsersBestGuess = "";
        string TheWordToGuess = "";
        string ThrowMeAFrickinBoneHere = "";
        string MixItAllUp = "";

        //Store word to guess and a corresponding hint in a 2
dimensional array
        const string WORDS[NumberOfWords][2] =
        {
           {"entitlement", "You think you deserve it."},
           {"plagiarism", "Taking the credit for someone else's
work."},
           {"revenge", "Don't get mad, get even."},
           {"generation", "By survival of the FITTEST, the
present one will soon be extinct."},
           {"pathetic", "It's always someone else's fault, it's
not my responsibility."},
           {"obsolete", "No longer relevant to the situation."},
           {"entrapment", "Deceptively misleading
situation..."},
           {"scapegoat", "Blame it on me."},
           {"cryptic", "l00k @ th1s, 1 m sew kewl n a11 l0fte"},
           {"disembowel", "Gut-wrenching..."},
           {"adolescent", "You're not the boss of me..."},
           {"agoraphobia", "Fear of open spaces..."},
           {"claustrophobia", "Fear of closed spaces..."},
           {"apocalyptic", "Of or pertaining to the end..."},
           {"karma", "Getting what you deserve..."}
       };

    cout << "\t\t\tGuessAWord 1.0 Instructions:\n\n";
    cout << "The object of this game is to guess " <<
NumWordsToGuess << " scrambled words\n";
    cout << "within " << NumberOfWrongGuesses << " wrong guesses
with scrambled letters.\n\n\n";

    //Guess 3 words (outer loop)
    while((UsersBestGuess != "quit") && (counter < 10) &&
(WordsGuessed < NumWordsToGuess))
    {
            TheChosenOne = randy();
        TheWordToGuess = WORDS[TheChosenOne][0];
        ThrowMeAFrickinBoneHere = WORDS[TheChosenOne][1];
        MixItAllUp = TheWordToGuess;
        TheSizeOfMixItAllUp = MixItAllUp.size();

        //If by strange chance strings are same, keep scrambling
until they are not.
        while(MakeSureTheyAreNotTheSame == 0)
        {
```

```
            for(int i=0; i<TheSizeOfMixItAllUp; ++i)
            {
                Randy1 = (rand() % TheSizeOfMixItAllUp);
                Randy2 = (rand() % TheSizeOfMixItAllUp);
                temp = MixItAllUp[Randy1];
                MixItAllUp[Randy1] = MixItAllUp[Randy2];
                MixItAllUp[Randy2] = temp;
            }

            //The compare() fucntion returns 0 if strings are
equal. Arguments are:
            //(start character, # characters, the string),
hence:
            MakeSureTheyAreNotTheSame =
TheWordToGuess.compare(0,TheSizeOfMixItAllUp,MixItAllUp);
        }

        //By the time we get here, MakeSureTheyAreNotTheSame is
currently not 0 but 1, so we
        //need to reset the variable so next word will be
scrambled in the for loop above.
        MakeSureTheyAreNotTheSame = 0;

        cout << "So far, you have guessed " << WordsGuessed << "
word(s) of the "
             << NumWordsToGuess << " required.\n";
        cout << "So far, you have used " << counter << " of your
"
             << NumberOfWrongGuesses << " wrong guesses.\n";
        cout << "If you enter \"clue\", you'll get a clue.\n";
        cout << "If you enter \"quit\", you'll exit the
game.\n\n";
        cout << "The word to guess is: " << MixItAllUp;

        //Guess each word (inner loop)
        while((UsersBestGuess != TheWordToGuess) &&
(UsersBestGuess != "quit") &&
              (counter < NumberOfWrongGuesses))
        {
            cout <<  "\n\nYour guess for word number " <<
(WordsGuessed + 1) << ": ";
            cin >> UsersBestGuess;

            for(int z = 0; z < UsersBestGuess.size(); z++)
            {  UsersBestGuess[z] = tolower(UsersBestGuess[z]);
}


            if(UsersBestGuess == "clue")
            {
               if(WorthyOfAClue == true)
               {
                  cout << ThrowMeAFrickinBoneHere;
               }
               else
               {
```

```
                    cout << "You are not yet worthy of a clue.
Try at least one guess first!";
                }
            }

            else if(UsersBestGuess == TheWordToGuess)
            {
                 cout << "\nHooray! You have guessed word number
" << (WordsGuessed + 1) << ".\n";
                 WordsGuessed = WordsGuessed + 1;
                 WorthyOfAClue = false;
                 break;
            }

            else
            {
               cout << "Sorry, but that is not what word number
" << (WordsGuessed + 1) << " is.\n";
               counter = counter + 1;
               cout << "Careful, you only have " <<
(NumberOfWrongGuesses - counter)
                    << " wrong guesses left!\n";
               WorthyOfAClue = true;
            }

        } //close nested while true loop

    } //close outer while true loop

    if(WordsGuessed == 3)
    {
       cout << "\n\nYou did it! You guessed all three words and
therefore you win the game!"
            << "\n\nYAY!!!\n\n";
    }

    else
    {
       cout << "\n\nSorry, you did not guess all three words
before making\n"
            << NumberOfWrongGuesses << " incorrect guesses. As a
result: \n\n"
            << "YOU LOSE!\n\n";
    }

    cout << "\n\nWould you like to play again?\n\n";
    cin >> CONTINUE;
    CONTINUE = tolower(CONTINUE);

  }//close outermost while

    cout << "\nExiting GuessAWord 1.0...\n\n\n";
    system("PAUSE");
    return 0;
}
```

C++ Console 32 Project 1: **MonsterCombat 2.0**

Binary Executable: monstercombat.exe

```
//Monster Mortal Combat 2.2, C. Germany, May 15, 2006

#include <iostream>
#include <windows.h>
#include <ctime>

using namespace std;

//Class Defintions-----------------------------------------------
----
class Monster
{
  public:
  Monster(int hp)
  {
     cout << "A monster has been created.\n";
     hitpoints = hp;

  }
  ~Monster() { cout << "Destroying a monster.\n"; }

  //Accesor methods
  int getHit() { return hitpoints; }
  void setHit(int hp) { hitpoints = hp; }
  char * getName() { return name; }
  void setName(char * nm) { name = nm; }

  private:
  int hitpoints;
  char * name;

};


//Function Prototypes--------------------------------------------
-------
void Attack();
int GenerateRandomNumber(int n);


//Main Function--------------------------------------------------
-------
int main()
{
    //Since it was being called so rapidly within the while true
loop,
    //srand had to be moved from the raondom number function to
main().
    //Otherwise the random numbers are calculated too close
together
    //and end up being many times the same.
    srand(time(NULL));
```

```
    cout << "\t\t\a     Welcome to Monster Mortal Combat 2.2\n"
         << "\t\t\a        Only the strong will
survive.\n\n\n\n";

    Attack();

    system("PAUSE");
    return 0;
}


//Function Definitions---------------------------------------------
-------

void Attack()
{
   int damage;
   int hit;
   int MothraWins = 0;
   int GodzillaWins = 0;

   //Create 2 Monsters
   hit = GenerateRandomNumber(100);
   Monster Godzilla(hit);

   hit = GenerateRandomNumber(100);
   Monster Mothra(hit);

   cout << "\n\n\nGodzilla and Mothra are locked in mortal
combat!\n\n\n\n";

   system("PAUSE");

   //Have Monster objects loop into mortal combat, calculate the
best of 3 matches
   for(int x = 0; x<3; x++)
   {
      system("CLS");

      cout << "\nCombat session " << x+1 << ".\n\n";
      cout << "Current Wins:\n\n";
      cout << "Mothra: " << MothraWins << "  Godzilla: " <<
GodzillaWins << "\n\n";

      //Hitpoints begin with value initialized by the Monster
constructor.
      cout << "Godzilla begins this fight with "
           << Godzilla.getHit() << " hitpoints.\n";

      cout << "Mothra begins this fight with "
           << Mothra.getHit() << " hitpoints.\n\n\n\n";

      system("PAUSE");
      system("CLS");
```

```
      //Each individual match of 3 between Monster objects plays
out until one dies.
      while(Godzilla.getHit() > 0 || Mothra.getHit() > 0)
      {
           //Generate random number for combat
           damage = GenerateRandomNumber(30);
           cout << "\nMothra attacks first, doing "
                << damage << " points of damage.\n";

            //We need to make sure Godzilla's hitpoints are at
least equal to
            //or greater than damage, otherwise we'll end up with
a negative
            //value. If so, we just set it to 0 and break out of
the while
            //true loop - Godzilla dies. Without checking for
this, Godzilla
            //may still be able to attack even when he is dead.
           if(Godzilla.getHit() <= damage)
           {
              Godzilla.setHit(0);
              break;
           }
           else
           {  //Subtract the randomly calculated damage from
Godzilla's hitpoints.
              Godzilla.setHit((Godzilla.getHit() - damage));
           }

           cout << "Godzilla now has " << Godzilla.getHit() << "
hitpoints.\n";

            //Pauses program for 4 seconds. The Sleep() function
requires you
           //to include the <windows.h> file and counts in
milliseconds.
           Sleep(5000);


           damage = GenerateRandomNumber(30);
           cout << "\nGodzilla attacks second, doing "
                << damage << " points of damage.\n";

           if(Mothra.getHit() <= damage)
           {
               Mothra.setHit(0);
               break;
           }
           else
           {
               Mothra.setHit((Mothra.getHit() - damage));
           }

           cout << "Mothra now has " << Mothra.getHit() << "
hitpoints.\n";
```

```
      } //closes while true loop

      //Determine who wins each of the three battles and count
wins.
      if(Godzilla.getHit() <= 0)
      {
         cout << "\n\nGodzilla loses.\n";
         MothraWins = MothraWins + 1;
      }
      if(Mothra.getHit() <= 0)
      {
         cout << "\n\nMothra loses.\n";
         GodzillaWins = GodzillaWins + 1;
      }

      Sleep(6000);

      hit = GenerateRandomNumber(100);
      Godzilla.setHit(hit);
      hit = GenerateRandomNumber(100);
      Mothra.setHit(hit);

   }//closes for loop

      cout << "\n\nFinal Results of All Three Battles:\n\n";
      cout << "Mothra: " << MothraWins << "  Godzilla: " <<
GodzillaWins << "\n\n";

   //Determine who won the war, best out of three battles.
   if(GodzillaWins > MothraWins)
   {
       cout << "Godzilla wins the war.\n\n\n\n";
   }
   else
   {
      cout << "Mothra Wins the war.\n\n\n\n";
   }

}//closes attack function

//-----------------------------------------------------------------
-----

int GenerateRandomNumber(int n)
{
    int ResultRandom;
    ResultRandom = (rand()%n) + 1;
    return ResultRandom;
}

//-----------------------------------------------------------------
-----
```

C++ Console 32 Project 1: **Tic Tac Toe 2.0**

Binary Executable: tictactoe.exe

```
//Tic-Tac-Toem, demonstrates a very simple A.I., C. Germany, May 17, 2006
//Uses pointers to pass an array for more efficiency than passing by value

#include <iostream>
#include <string>

using namespace std;

//Function Prototypes-------------------------------------------------
char CheckForAWinner(char * board);
int ComputerPlays(char * board, char playerpiece, char computerpiece);
void Display(char * board);
void DisplayWinner(char TheWinner, char computerpiece, char playerpiece);
char Introduction();
bool LegalMove(int move,char * board);
int PlayerPlays(char * board, char human);

//Globals
const int SQUARES = 9;
bool stupid;

//Main() Function-----------------------------------------------------

int main()
{
    int choice;
    char GameBoard[SQUARES] = {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '};

    char playerpiece, computerpiece, currentplayer;

    //Start game and get player's choice to go first/last.
    playerpiece = Introduction();

    //Whatever player chooses, 'X' or 'O', the computer gets the other.
    if(playerpiece == 'X')  { computerpiece = 'O'; }
    else { computerpiece = 'X'; }

    currentplayer = 'X';
    Display(GameBoard);

    while (CheckForAWinner(GameBoard) == 'N')
    {
        if (currentplayer == playerpiece)
        {
            choice = PlayerPlays(GameBoard, playerpiece);
            GameBoard[choice] = playerpiece;
        }
        else
        {
            choice = ComputerPlays(GameBoard, playerpiece, computerpiece);
            GameBoard[choice] = computerpiece;
        }
```

```
        Display(GameBoard);

        if(currentplayer == playerpiece) { currentplayer = computerpiece; }
        else { currentplayer = playerpiece; }
    }

    DisplayWinner(CheckForAWinner(GameBoard), computerpiece, playerpiece);

    cout << "\nGame Over.\n\n"
         << "Exiting Tic Tac Toe 2.0 ...\n\n\n";

    system("PAUSE");
    return 0;
}


//Function Definitions---------------------------------------------------

char CheckForAWinner(char * GameBoard)
{
    //All the wining combinations, horizontal, vertical and diagonal.
    const int WINNING_ROWS[8][3] = { {0, 1, 2},
                                     {3, 4, 5},
                                     {6, 7, 8},
                                     {0, 3, 6},
                                     {1, 4, 7},
                                     {2, 5, 8},
                                     {0, 4, 8},
                                     {2, 4, 6} };
    const int TOTAL_ROWS = 8;

    //If an row has three values that are the same we have a winner
    //We will either return 'X' or 'O' as the winner or a tie and
    //end the game, or else we will return 'N' and keep playing the game.
    for(int row = 0; row < TOTAL_ROWS; ++row)
    {
        if ( (GameBoard[WINNING ROWS[row][0]] != ' ') &&
             (GameBoard[WINNING_ROWS[row][0]] ==
GameBoard[WINNING_ROWS[row][1]]) &&
             (GameBoard[WINNING_ROWS[row][1]] ==
GameBoard[WINNING_ROWS[row][2]]) )
        {
            return GameBoard[WINNING_ROWS[row][0]];
        }
    }

    //If we don't have a winner, check for a possible tie.
    bool NoEmptySpaces = true;

    for(int x = 0; x < SQUARES; x++)
    {
        if(GameBoard[x] == ' ') { NoEmptySpaces = false; }
    }

    if(NoEmptySpaces == true)
       return 'T';
```

```
    //If nobdy wins and it's not a tie return 'N' and keep playing.
    return 'N';
}

//---------------------------------------------------------------------------

int ComputerPlays(char * GameBoard, char playerpiece, char computerpiece)
{
    cout << "\nThe computer chooses location: ";

    //1. If computer can win on next choice, make that choice.

    for(int choice = 0; choice < SQUARES; ++choice)
    {
        if (LegalMove(choice, GameBoard))
        {
            GameBoard[choice] = computerpiece;
            if (CheckForAWinner(GameBoard) == computerpiece)
            {
                cout << choice << "." << endl;
                return choice;
            }
            // done checking this choice, undo it
            GameBoard[choice] = ' ';
        }
    }

    //Note: Computer will not try to block player's winning move if stupid =
true
    if(stupid == false)
    {
        //2. If human can win on next choice, block that choice.
        char human = playerpiece;
        for(int choice = 0; choice < SQUARES; ++choice)
        {
            if(LegalMove(choice, GameBoard))
            {
                GameBoard[choice] = human;
                if(CheckForAWinner(GameBoard) == human)
                {
                    cout << choice  << "." << endl;
                    return choice;
                }
                // done checking this choice, undo it
                GameBoard[choice] = ' ';
            } //close inner if
        } //close for loop
    } //close outer if

    //3. If no one can win on next choice, pick best open square.
    const int PreferredChoices[] = {4, 0, 2, 6, 8, 1, 3, 5, 7};

    for(int i = 0; i < SQUARES; ++i)
    {
        int choice = PreferredChoices[i];
```

```
        if (LegalMove(choice, GameBoard))
        {
            cout << choice  << "." << endl;
            return choice;
        }
    }
}

//--------------------------------------------------------------------------

void Display(char * GameBoard)
{
    cout << "\n\t" << GameBoard[0] << " | " << GameBoard[1] << " | " <<
GameBoard[2];
    cout << "\n\t" << "---------";
    cout << "\n\t" << GameBoard[3] << " | " << GameBoard[4] << " | " <<
GameBoard[5];
    cout << "\n\t" << "---------";
    cout << "\n\t" << GameBoard[6] << " | " << GameBoard[7] << " | " <<
GameBoard[8];
    cout << "\n\n";
}

//--------------------------------------------------------------------------

void DisplayWinner(char TheWinner, char computerpiece, char playerpiece)
{
        if (TheWinner == computerpiece)
    {
        cout << "\n" << TheWinner << " wins the game. "
             << "The computer wins this match.\nSorry, you loose.\n";
    }

        else if (TheWinner == playerpiece)
    {
        cout << "\n" << TheWinner << " wins the game. "
             << "Excellent. You win this match. The computer looses.\n";
    }

        else
    {
        cout << "The game ends in a tie.\nNeither you nor the computer
wins.\n\n";
        }
}

//--------------------------------------------------------------------------

char Introduction()
{
    char FirstOrSecond;
    char intelligence;
    char playerpiece = 'z';

    cout << "\aWelcome to Tic Tac Toe 1.0.\n";
```

```
    cout << "This game uses very basic A.I. in the form of several decision
structures.\n\n";

    cout << "To play, enter a number, 0 - 8.  The number you enter will\n";
    cout << "indicate which of 9 positions you desire below:\n\n";

    cout << "       0 | 1 | 2\n";
    cout << "       ---------\n";
    cout << "       3 | 4 | 5\n";
    cout << "       ---------\n";
    cout << "       6 | 7 | 8\n\n";

    cout << "The computer will play as your oponent in this game.\n\n";

    while(intelligence != 'I' && intelligence != 'S')
    {
        cout << "\aDo you want your computer opponent to be (i)ntelligent or
(s)tupid?\n"
             << "Choosing this option for your opponent sets the difficulty
of the game. ";
        cin >> intelligence;
        intelligence = toupper(intelligence);

        switch(intelligence)
        {
           case 'I' : cout << "\nO.k., your computer opponent will be more
dificult to beat.\n";
                      stupid = false;
                      break;
           case 'S' : cout << "\nO.k., your computer opponent will be easier
to beat.\n";
                      stupid = true;
                      break;
           default : cout << "That is an invalid response.\n\n"; break;
        }
    }//close while true loop

    while(FirstOrSecond != 'Y' && FirstOrSecond != 'N')
    {
        cout << "\a\nDo you want to make the first move (y/n)? ";
        cin >> FirstOrSecond;
        FirstOrSecond = toupper(FirstOrSecond);

        switch(FirstOrSecond)
        {
           case 'Y' : cout << "\nO.k., you take the first move.\n";
                      playerpiece = 'X';
                      break;
           case 'N' : cout << "\nO.k., the computer takes the first
move.\n";
                      playerpiece = 'O';
                      break;
           default : cout << "That is an invalid response.\n\n"; break;
        }
    }//close while true loop
```

```
    system("PAUSE");
    system("CLS");

    return playerpiece;

}//close function

//--------------------------------------------------------------------------

bool LegalMove(int choice, char * GameBoard)
{
    return (GameBoard[choice] == ' ');
}

//--------------------------------------------------------------------------

int PlayerPlays(char * GameBoard, char human)
{
    int PlayersMove;

    while(PlayersMove > SQUARES || PlayersMove < 0 ||
         (LegalMove(PlayersMove, GameBoard)== false))
    {
        cout << "\nChoose a location (0-8): ";
        cin >> PlayersMove;

        if(PlayersMove > SQUARES || PlayersMove < 0)
        { cout << "\nThat number is outside of the valid range of 1-8.\n"; }

        else if(LegalMove(PlayersMove, GameBoard) == false)
        {
          cout << "\nYou can not choose that location. It already has a(n) "
               << GameBoard[PlayersMove] << " in it.\n";
        }
    }

    return PlayersMove;
}

//--------------------------------------------------------------------------
```


C++ Console 32 Project 1: **Quiz 1.0**

Binary Executable: quiz.exe

```
#include <iostream>
#include <string>

using namespace std;

//Globals
const int NumberOfQuestions = 5;
```

```
const int NumWrongGuesses = 10;
int score;
int WrongCount;
int RightCount;

//ProtoTypes
void quiz1();
void quiz2();

//--------------------------------------------------------------------------
----------------------

int main()
{
        WrongCount = 0;
        RightCount = 0;
        score = 0;
        char choice;
        cout << "\nChoose an example to run:\n\n"
             << "(M)ulti-dimensional Array Quiz\n"
             << "(P)arallel Array Quiz\n\n";

        cin >> choice;

        switch(choice)
        {
            case 'm' : quiz1(); break;
            case 'p' : quiz2(); break;
            default: cout << "Invalid choice. Ending program."; break;
        }

        cout << "\n\n";
        system("PAUSE");
        return 0;

}//close main()

//--------------------------------------------------------------------------
----------------------

//Function Definitions
void quiz1()
{
     //This Function Illustrates a Multi-Dimensional Array, 1 array with
5 rows and 4 columns

     string QandClueandA[5][4] = {
     {"\nEl dia de los muertos? ","october_30","Same month as
halloween.",""},
     {"\nFavorite game? ","descent_3","Maze-like first person shooter
game.",""},
     {"\nBest new television show: ","bsg","Frack off cylon!",""},
     {"\nFavorite hobby? ","music","The hills are alive...",""},
     {"\nPopular vacation Destination? ","bahamas","Ja man, on da
beaches.",""} };
```

```
     //Quiz Using Multi-dimensional arrays
     for(int x = 0; x < NumberOfQuestions && WrongCount <
NumWrongGuesses; x++)
     {
         cout << QandClueandA[x][0];
         cin >> QandClueandA[x][3];

         //Convert User's Answer to lower case
         for(int z = 0; z < QandClueandA[x][3].size(); z++)
         { QandClueandA[x][3][z] = tolower(QandClueandA[x][3][z]); }

         if(QandClueandA[x][1] != QandClueandA[x][3])
         {
            cout << "\nHere's a hint: " << QandClueandA[x][2] << "\n\n";
            x--;
            WrongCount++;
         }
         else
         { cout << "That's it! You got it right!"; score++;
RightCount++; }

         cout << "\n\nCurrent Status:   Wrong Guesses = " << WrongCount
              << "   Right Guesses = " << RightCount << "\n\n";

     }//close for loop

     score = score * 20;

     switch(score)
     {
            case 0 : cout << "You got NOTHING! Try at least guessing
next time! Grade = F.";
                     break;
            case 20 : cout << "Score: " << score << " Got at least one
right. Grade = F.";
                      break;
            case 40 : cout << "Score: " << score << " Got at least two
right. Grade = F.";
                      break;
            case 60 : cout << "Score: " << score << " Got at least three
right. Grade = F.";
                      break;
            case 80 : cout << "Score: " << score << " Got at least four
right. Grade = B.";
                      break;
            case 100 : cout << "Score: " << score
                            << " Got all of them right! Well done! Grade
= A."; break;
            default : cout << "Should never happen.";
     }

}//close quiz1()

//-------------------------------------------------------------------------
----------------------
```

```
void quiz2()
{
    //This Function Illustrates a Parallel Array Example using 4
separate parallel arrays

     string Questions[5] = { "\nEl dia de los muertos? ",
                             "\nFavorite game? ",
                             "\nBest new television show: ",
                             "\nFavorite hobby? ",
                             "\nPopular vacation Destination? "};

     string Answers[5] = { "october_30",
                           "descent_3",
                           "bsg",
                           "music",
                           "bahamas"};

     string Clues[5] = { "Same month as halloween.",
                         "Maze-like first person shooter game.",
                         "Frack off cylon!",
                         "The hills are alive...",
                         "Ja man, on da beaches."};


     string UsersInput[5] = { "", "", "", "", ""};

     for(int x = 0; x < NumberOfQuestions && WrongCount <
NumWrongGuesses; x++)
     {
         cout << Questions[x];
         cin >> UsersInput[x];

         //Convert User's Answer to lower case
         for(int z = 0; z < UsersInput[x].size(); z++)
         { UsersInput[x][z] = tolower(UsersInput[x][z]); }

         if(UsersInput[x] != Answers[x])
         {
            cout << "\nHere's a hint: " << Clues[x] << "\n\n";
            x--;
            WrongCount++;
         }
         else
         { cout << "That's it! You got it right!"; score++;
RightCount++; }

         cout << "\n\nCurrent Status:   Wrong Guesses = " << WrongCount
              << "   Right Guesses = " << RightCount << "\n\n";

     }//close for loop

     score = score * 20;

     if(score <= 100 && score >= 90)
     { cout << "You earned and A for this quiz!"; }
     if(score <= 89 && score >= 80)
```

```
     { cout << "You earned and B for this quiz!"; }
     if(score <= 79 && score >= 70)
     { cout << "You earned and C for this quiz!"; }
     if(score <=69  && score >= 65)
     { cout << "You earned and D for this quiz!"; }
     if(score <= 64 && score >= 0)
     { cout << "You earned and F for this quiz!"; }

}//close quiz2()
```

```
//Quiz 1.0, simple quiz program to demonstrate 2 dimensional arrays, C.
Germany, May 11, 2006

#include <iostream>
#include <string>

using namespace std;

int main()
{
    int score = 0;
    const int NumberofQuestions = 5;

        cout << "\t\tQuiz 1.0\n\n";

    //Store questions and answers in 2 dimensional array
    string QandA[NumberofQuestions][2] = {
                                    {"Who is Scooby Doo's best
friend?","shaggy"},
                                    {"Who developed
relativity?","einstein"},
                                    {"When did the twin towers
fall?","2001"},
                                    {"Who was the first
president?","washington"},
                                    {"What is the meaning of life?","42"}
                                    };

    string Answers[NumberofQuestions];

    //loop through each string in the array and compare it to answers
    for(int x = 0; x < NumberofQuestions; x++)
    {
       cout << QandA[x][0] << " ";
       cin >> Answers[x];

       //loop through each character of each string and convert to
lower case
       for(int y = 0; y < Answers[x].length(); y++)
       {
           Answers[x][y] = tolower(Answers[x][y]);
       }//close inner loop

       if(QandA[x][1] == Answers[x])
```

```
        {
            score++;
        }//close if
    }//close outer loop

    cout << "\n\nYou got " << score << " of "
         << NumberofQuestions << " right!\n\n\n";

    system("PAUSE");
    return 0;

}//close main()
```


C++ Console 32 Project 1: **Database 1.0** Binary Executable: Database1.exe

Objective: To demonstrate an understanding of polymorphism, inheritance, pointers, references and local and global scope. To complete this project, you must demonstrate an understanding of passing objects by pointer, reference and value. You must be familiar with creating objects locally on the stack and on the free store (heap). You should demonstrate cleaning up memory leaks by calling delete on pointers to objects on the heap when they are no longer needed, and you should initialize those pointers to 0 to avoid calling delete on heap objects twice. You must also use repetition and decision structures where needed.

Files In Project: Total of 3. 1 - DatabaseClasses.h, 2 - DatabaseFunctions.h, 3 - MainFile.cpp .

I. Database Classes

```
// File 1 of 3 - Database Classes - ©2004 C. Germany - in header file
"DatabaseClasses.h"
//-------------------------------------------------------------------------------------------
---------------------------------

#include <stdlib.h>
#include <iostream>
#include <string>

using namespace std;
//-------------------------------------------------------------------------------------------
---------------------------------
//Function prototypes
bool VerifyPassword();
//-------------------------------------------------------------------------------------------
---------------------------------
class Employee {
   public:

   static int PrimaryKey;

        //Overloaded constructors
        Employee()
        {
                Initialize();
```

```
                    PrimaryKey++;
                    EmpID = PrimaryKey;

                    cout << "\nA New Employee object was created.";
          }


          Employee(int EID, int DID, float sal, long social, long hd, long
rd,
                    char * first, char * last, char * job)
          {
                   EmpID = EID;
                   DeptID = DID;
                  salary = sal;
                  SSN = social;
                  HireDate = hd;
                  ReviewDate = rd;
                   FirstName[20] = first[20];
                   LastName[20] = last[20];
                  JobDescription[40] = job[40];

                   EmpID = ++PrimaryKey;
          }

          //Our destructor will clean up our heap objects created in the EditEmployee function
          ~Employee()
          {
                 cout << "\nEmployee object deleted.\n";
          }

          //Accesor methods
          int GetEmpID() { return EmpID; }
          void SetEmpID(int eid) { EmpID = eid; }

          int GetDeptID() { return DeptID; }
          void SetDeptID(int ID) { DeptID = ID; }

          float GetSalary() { return salary; }
          void SetSalary(float sal) { salary = sal; }

          long GetSSN() { return SSN; }
          void SetSSN(long social) { SSN = social; }

          long GetHireDate() { return HireDate; }
          void SetHireDate(long hire) { HireDate = hire; }

          long GetReviewDate() { return ReviewDate; }
          void SetReviewDate(long review) { ReviewDate = review; }
          char * GetFirstName() { return FirstName; }
          void SetFirstName(char first[20]) { FirstName[20] = first[20]; }
          char * GetLastName() { return LastName; }
          void SetLastName(char last[20]) { LastName[20] = last[20]; }

          char * GetJobDescription() { return JobDescription; }
          void SetJobDescription(char description[40]) {
JobDescription[40] = description[40]; }
```

```
        //Other methods
        void EditEmployee() {

              //Create character arrays on the heap so they will persist when the function
ends and not go out of scope
              char First[20];
              char Last[20];
              char Desc[40];
              bool LegitimatePass;
              int x = 0;
              //Check to make sure it's a manager and they have the correct password
              LegitimatePass = VerifyPassword();
              if(LegitimatePass == false)
              {
                   cout << "   Sorry.  You do not have permission to
access this feature.\n\n";
              }
              else
              {
                   cout.flush();
                   system("CLS");
                   cout << "\n   Welcome!  You may edit employee
information below for";
                   cout << " employee # " << GetEmpID() << ".\n\n";
                   cout << "************ Edit Employee Information
*************\n";
                   cout << "\n\nEmployee ID: " << GetEmpID() << endl;
                  cout << "\nEnter Department ID: ";
                  cin >> DeptID;
                  cout << "\nEnter First Name: ";
                  cin >> First;
                  cout << "\nEnter Last Name: ";
                  cin >> Last;
                  cout << "\nEnter Job Description: ";
                  cin >> Desc;
                  cout << "\nEnter employee salary: ";
                  cin >> salary;
                  cout << "\nEnter employee SSN: ";
                  cin >> SSN;
                  cout << "Enter employee hire date (mmddyyyy): ";
                  cin >> HireDate;
                  cout << "Enter employee\'s last review date: ";
                  cin >> ReviewDate;

                   //This is how we can copy a char array (string) without strcpy
                   for(int x = 0; x <20; x++) {
                        FirstName[x] = First[x];
                        LastName[x] = Last[x];
                   }
                   //For the last value to be copied we'll use strcpy
                   strcpy(JobDescription, Desc);

              } //close else
        } //close function
```

```
        void DisplayEmployee() {

             cout << endl;
             cout <<
"************************************************\n";
             cout << "* Information for employee " << GetEmpID()
                  << ":                      *\n";
             cout <<
"************************************************\n";
             cout << endl << endl;
             cout << "Employee ID: " << GetEmpID() << endl;
              cout << "Department ID: " << GetDeptID() << endl;
              cout << "First Name: " << GetFirstName() << endl;
              cout << "Last Name: " << GetLastName() << endl;
              cout << "Job Description: " << GetJobDescription() <<
endl;
              cout << "Salary: " << GetSalary() << endl;
              cout << "SSN: " << GetSSN() << endl;
              cout << "Hire Date: " << GetHireDate() << endl;
              cout << "Review Date: " << GetReviewDate() << endl;
              cout << endl << endl;

        } //close function


        void Initialize()
        {
             DeptID = 0;
             salary = 0.0;
             SSN = 000000000;
             HireDate = 00000000;
             ReviewDate = 00000000;
             strcpy(FirstName, "No First Name Yet");
             strcpy(LastName, "No Last Name Yet");
             strcpy(JobDescription, "No Job Description Yet");
        }

    //Data members
    private:

       int EmpID;
       int DeptID;
       float salary;
       long SSN;
       long HireDate;
       long ReviewDate;
       char FirstName[20];
       char LastName[20];
       char JobDescription[40];

};   //close base class specification

//Static member data must be defined outside the class specification
int Employee::PrimaryKey = 0;
//-------------------------------------------------------------------------------------------
---------------------------------
```

```
class Manager : public Employee {
   public:

        Manager() { cout << "\nNew Manager object created."; }
        ~Manager() { cout << "\nManager deleted"; }

        //Accessor methods
       int GetManID() { return ManID; }
       void SetManID(int ID) { ManID = ID; }

       double GetLogInTime() { return LogInTime; }
       void SetLogInTime(double logtime) { LogInTime = logtime; }
       char * GetPassWord() { return PassWord; }
       void SetPassWord(char * pass) { PassWord = pass; }

       char * GetResponsibilities() { return Responsibilities; }
       void SetResponsibilities(char * response) { Responsibilities =
response; }

       void setDesc(char * des) { Desc = des; }
       char * getDesc() { return Desc; }
   //Data members
   private:

       int ManID;
       double LogInTime;
       char * PassWord;
       char * Responsibilities;
       char * Desc;
};
//-------------------------------------------------------------------------------------------
---------------------------------
```

II. Database Functions

```
// File 2 of 3 - Database Functions - 2004 C.  Germany - in header file
"DatabaseFunctions.h"

//-------------------------------------------------------------------------------------------
---------------------------------

void ListEmployeeIDs(Employee * Comp, int & Num) {

            //List The Employee IDs Available:
            cout << "\n\n-----------------------------------------\n";
            cout << "Valid employee IDs are: \n\n";

            for(int x = 0;x < Num; x++)
            {
                  //Need to offset for the fencepost by adding 1
                 cout << (Comp[x].GetEmpID()) << " ";
            }

            cout << "\n-----------------------------------------\n\n";
```

```
}
//-------------------------------------------------------------------------------------------
-------------------------------
bool VerifyPassword() {
     cout.flush();
     system("CLS");
     int ManagerID;
     string Password;
     cout << "\n   To add employees to this database, you must be a
department manager\n"
          << "   with a valid password and manager ID.";
     cout << "\n\n   Please enter your manager ID: ";
     cin >> ManagerID;
     if(ManagerID > 99 && ManagerID < 105)
     {
         cout << "\n\n   Please enter your password (case sensitive):
";
         cin >> Password;
         cout << "\n\n   You entered: " << Password << ".\n";
         switch(ManagerID) {
               case 100: if(Password == "blue")
                         {
                           cout << "\n   Ok., manager " << ManagerID
                                << ", your password is valid!\n";
                          return true;
                         }
                        break;

               case 101: if(Password == "red")
                         {
                            cout << "\n   Ok., manager " << ManagerID
                                 << ", your password is valid!\n";
                           return true;
                         }
                        break;

               case 102: if(Password == "green")
                         {
                            cout << "\n   Ok., manager " << ManagerID
                                << ", your password is valid!\n";
                           return true;
                         }
                        break;

               case 103: if(Password == "yellow")
                         {
                           cout << "\n   Ok., manager " << ManagerID
                                << ", your password is valid!\n";
                           return true;
                         }
                        break;

               case 104: if(Password == "white")
                         {
                           cout << "\n   Ok., manager " << ManagerID
                                << ", your password is valid!\n";
```

```
                              return true;
                            }
                           break;

                default: cout << "Invalid.\n\n";
                        break;
                  } //close switch
          } //close if
          else { cout << "   Invalid Manager ID!\n\n"; }

          return false;

} //close function
//-------------------------------------------------------------------------------------------------
---------------------------------
void EditEmployeeInformation(Employee * ACompany, int & NumEmployees) {
      int key = 0;
      cout << "\n   Welcome!  You may enter new Employee information
below:" << endl;
      ListEmployeeIDs(ACompany, NumEmployees);
      cout << "Which employee would you like to edit?  Enter the key
value: ";
      cin >> key;
      //Check for valid ID, grater than 0 and one less than NumEmp because of fencepost
      if(key > 0 && key <= NumEmployees)
         {  ACompany[key-1].EditEmployee(); }
      else { cout << "\nI am sorry, that is an invalid eployee ID.\n\n";
}
} // close function
//-------------------------------------------------------------------------------------------------
---------------------------------
void DisplayEmployeeInformation(Employee * ACompany, int &
NumEmployees) {

      int key = 0;

      cout << "\n   Welcome!  You may display information for"
           << "the employees listed below:" << endl;
      ListEmployeeIDs(ACompany, NumEmployees);
      cout << "Which employee would you like to view?  Enter the key
value: ";
      cin >> key;
      //Check for valid ID, grater than 0 and one less than NumEmp because of fencepost
      if(key > 0 && key <= NumEmployees)
         {  ACompany[key-1].DisplayEmployee(); }
      else { cout << "\nI am sorry, that is an invalid eployee ID.\n\n";
}
} // close function
//-------------------------------------------------------------------------------------------------
---------------------------------

void DeleteEmployee(Employee * ACompany, int & NumEmployees) {

      int key = 0;
      cout << "\n Warning! You are about to delete an employee record!"
<< endl;
```

```
      ListEmployeeIDs(ACompany, NumEmployees);

      cout << "Which employee record would you like to delete? Enter the
key value: ";
      cin >> key;

      //Check for valid ID, grater than 0 and one less than NumEmp because of fencepost
      if(key > 0 && key <= NumEmployees)
      {
            ACompany[key-1].Initialize();
      }

      else { cout << "\nI am sorry, that is an invalid eployee ID.\n\n";
}

} // close function

//-------------------------------------------------------------------------------------------
---------------------------------


void MainMenu(char & Continue, Employee * Company, int & NumEmp) {
      char MenuChoice;
      cout << "\n\n";
      cout << "\t*************************************************\n"
           << "\t*                 Database Menu                 *\n"
           << "\t*************************************************\n"
           << "\t*                                               *\n"
           << "\t*      E = Edit Employee Information            *\n"
           << "\t*      D = Display Employee Information          *\n"
           << "\t*      R = Erase Employee Information            *\n"
           << "\t*      A = Add Employee to Database              *\n"
           << "\t*      P = Populate Database (new district)      *\n"
           << "\t*      X = Exit Menu Function                    *\n"
           << "\t*                                               *\n"
           << "\t*************************************************\n";
      cout << "\n\nEnter a selection: ";
      cin >> MenuChoice;
      //Below, "tolower" transfroms a char to lowercase.
      switch(tolower(MenuChoice))
      {
           case 'e' : cout << "\nEdit Employee Information.\n";
                      EditEmployeeInformation(Company, NumEmp);
                       break;
           case 'd' : cout << "\nDisplay Employee Information\n";
                      DisplayEmployeeInformation(Company,
NumEmp);
                       break;

           case 'r' : cout << "Remove Employee.";
                       DeleteEmployee(Company, NumEmp);
                      break;
           case 'p' : cout << "Populating database for a new district.";
                      break;
           case 'x' : cout << "Exiting database menu...\n\n";
                      Continue = 'q';
```

```
                    break;
          default :  cout << "Invalid response...\n\n";
                    break;

     } //close switch statement
} //close function
//-------------------------------------------------------------------------------------------
---------------------------------
//This stuff below is just junk for demonstrations and debugging tutorials.
void TestStuff() {
     char MoveOn;
     //Object test
     Employee Newbie;
     Newbie.SetFirstName("Charles");
     cout << "\n" << Newbie.GetFirstName() << "\n";
     Manager Buffy;
     Buffy.SetFirstName("Buffy");
     cout << "\nManager object named: " << Buffy.GetFirstName() <<
"\n";
     Buffy.setDesc("Vampire Slayer");
     cout << "Buffy Description: " << Buffy.getDesc() << endl;
     cout << "\n\n\nType C followed by return to continue.  ";
     cin >> MoveOn;

}
//-------------------------------------------------------------------------------------------
---------------------------------
//The function below is no longer used.  It was a precursor introduction to static member data.
//We are now implementing inline in our constructor what this function used to do.
void AssignUniqueIDs(Employee * ACompany, int & NumEmployees) {

     //Note: This should only get called ONCE, unless there is a new database.
     int key = 1;
     for(int x = 0; x < NumEmployees; x++)
     {
               ACompany[x].SetEmpID(key);
               ACompany[x].SetFirstName("NewEmployeeFirst");
               ACompany[x].SetLastName("NewEmployeeLast");
               ACompany[x].SetJobDescription("NewEmployeeDescription");
               key = key + 1;
     }
}

//-------------------------------------------------------------------------------------------
---------------------------------
```

III. Database Main Function

```
// File 3 of 3 Database Main Function - 2004 C. Germany - in source
file "MainFile.cpp"

#include "DatabaseClasses.h"
#include "DatabaseFunctions.h"
```

```
//-------------------------------------------------------------------------------------------
---------------------------------

void main()
{
      int NumEmps = 0;
       //Set sentinel value
      char Continue = 'z';
      //Stuff to happen only once
      cout << "Enter the number of employees in your company: ";
      cin >> NumEmps;
      Employee * TheCompany = new Employee[NumEmps];

      //Stuff to keep happening until user selects exit
      while(Continue != 'q')
      {
            MainMenu(Continue, TheCompany, NumEmps);

      } //end while true loop

      cout << "\nExiting program...\n\n";

      //Now that we're done, let's clean up our array of Employee objects on the heap.
      delete [] TheCompany;
      TheCompany = 0;
}   //close main()
```


C++ Console 32 Project 1: **Database 2.0** Binary Executable: Database2.exe

Objective: To demonstrate an understanding of polymorphism, inheritance, pointers, references and local and global scope. In addition, you will write basic linear search routines and include the ability to save your data to a binary file. You will create a function to load this data and display it. To complete this project, you must demonstrate an understanding of passing objects by pointer, reference and value.

You must be familiar with creating objects locally on the stack and on the free store (heap). You should demonstrate cleaning up memory leaks by calling delete on pointers to objects on the heap when they are no longer needed, and you should initialize those pointers to 0 to avoid calling delete on heap objects twice. You must also use repetition and decision structures where needed.

Files In Project: Total of 3. 1 - DatabaseClasses.h, 2 - DatabaseFunctions.h, 3 - MainFile.cpp .

I. Database Classes

```
// File 1 of 3 - Database Classes - ©2004 C. Germany - in header file
"DatabaseClasses.h"
//-------------------------------------------------------------------------------------------
---------------------------------
```

```
#include <stdlib.h>
#include <iostream>
#include <string>
#include <fstream>

using namespace std;
//-------------------------------------------------------------------------------------------
--------------------------------
//Function prototypes
bool VerifyPassword();
//-------------------------------------------------------------------------------------------
--------------------------------
class Employee {
   public:

   static int PrimaryKey;

         //Overloaded constructors
         Employee()
         {
                   Initialize();

                    PrimaryKey++;
                    EmpID = PrimaryKey;

                    cout << "\nA New Employee object was created.";
         }


         Employee(int EID, int DID, float sal, long social, long hd, long
rd,
                    char * first, char * last, char * job)
         {
                  EmpID = EID;
                  DeptID = DID;
                 salary = sal;
                 SSN = social;
                 HireDate = hd;
                 ReviewDate = rd;
                  FirstName[20] = first[20];
                  LastName[20] = last[20];
                 JobDescription[40] = job[40];

                  EmpID = ++PrimaryKey;
         }

         //Our destructor will clean up our heap objects created in the EditEmployee function
         ~Employee()
         {
                cout << "\nEmployee object deleted.\n";
         }

         //Accesor methods
         int GetEmpID() { return EmpID; }
         void SetEmpID(int eid) { EmpID = eid; }

         int GetDeptID() { return DeptID; }
```

```
        void SetDeptID(int ID) { DeptID = ID; }

        float GetSalary() { return salary; }
        void SetSalary(float sal) { salary = sal; }

        long GetSSN() { return SSN; }
        void SetSSN(long social) { SSN = social; }

        long GetHireDate() { return HireDate; }
        void SetHireDate(long hire) { HireDate = hire; }

        long GetReviewDate() { return ReviewDate; }
        void SetReviewDate(long review) { ReviewDate = review; }
        char * GetFirstName() { return FirstName; }
        void SetFirstName(char first[20]) { FirstName[20] = first[20]; }
        char * GetLastName() { return LastName; }
        void SetLastName(char last[20]) { LastName[20] = last[20]; }

        char * GetJobDescription() { return JobDescription; }
        void SetJobDescription(char description[40]) {
JobDescription[40] = description[40]; }


        //Other methods
        void EditEmployee() {

              //Create character arrays on the heap so they will persist when the function
ends and not go out of scope
              char First[20];
              char Last[20];
              char Desc[40];
              bool LegitimatePass;
              int x = 0;
              //Check to make sure it's a manager and they have the correct password
              LegitimatePass = VerifyPassword();
              if(LegitimatePass == false)
              {
                   cout << "   Sorry.  You do not have permission to
access this feature.\n\n";
              }
              else
              {
                   cout.flush();
                   system("CLS");
                   cout << "\n   Welcome!  You may edit employee
information below for";
                   cout << " employee # " << GetEmpID() << ".\n\n";
                   cout << "************ Edit Employee Information
*************\n";
                   cout << "\n\nEmployee ID: " << GetEmpID() << endl;
                  cout << "\nEnter Department ID: ";
                  cin >> DeptID;
                  cout << "\nEnter First Name: ";
                  cin >> First;
                  cout << "\nEnter Last Name: ";
                  cin >> Last;
```

```
                cout << "\nEnter Job Description: ";
                cin >> Desc;
                cout << "\nEnter employee salary: ";
                cin >> salary;
                cout << "\nEnter employee SSN: ";
                cin >> SSN;
                cout << "Enter employee hire date (mmddyyyy): ";
                cin >> HireDate;
                cout << "Enter employee\'s last review date: ";
                cin >> ReviewDate;

                 //This is one way to copy a char array (string) without strcpy
                 for(int x = 0; x <20; x++) {
                     FirstName[x] = First[x];
                     LastName[x] = Last[x];
                 }
                 //For the last value to be copied we'll use strcpy
                 strcpy(JobDescription, Desc);

           } //close else
      } //close function

      void DisplayEmployee() {

           cout << endl;
           cout <<
"************************************************\n";
           cout << "* Information for employee " << GetEmpID()
                << ":                      *\n";
           cout <<
"************************************************\n";
           cout << endl << endl;
           cout << "Employee ID: " << GetEmpID() << endl;
           cout << "Department ID: " << GetDeptID() << endl;
           cout << "First Name: " << GetFirstName() << endl;
           cout << "Last Name: " << GetLastName() << endl;
           cout << "Job Description: " << GetJobDescription() <<
endl;
           cout << "Salary: " << GetSalary() << endl;
           cout << "SSN: " << GetSSN() << endl;
           cout << "Hire Date: " << GetHireDate() << endl;
           cout << "Review Date: " << GetReviewDate() << endl;
           cout << endl << endl;

      } //close function


      void Initialize()
      {
           DeptID = 0;
           salary = 0.0;
           SSN = 000000000;
           HireDate = 00000000;
           ReviewDate = 00000000;
           strcpy(FirstName, "No First Name Yet");
           strcpy(LastName, "No Last Name Yet");
```

```
            strcpy(JobDescription, "No Job Description Yet");
        }

    //Data members
    private:

       int EmpID;
       int DeptID;
       float salary;
       long SSN;
       long HireDate;
       long ReviewDate;
       char FirstName[20];
       char LastName[20];
       char JobDescription[40];

};   //close base class specification


//Static member data must be defined outside the class specification
int Employee::PrimaryKey = 0;
//-------------------------------------------------------------------------------------------
--------------------------------
class Manager : public Employee {
   public:

       Manager() { cout << "\nNew Manager object created."; }
       ~Manager() { cout << "\nManager deleted"; }

       //Accessor methods
      int GetManID() { return ManID; }
      void SetManID(int ID) { ManID = ID; }

      double GetLogInTime() { return LogInTime; }
      void SetLogInTime(double logtime) { LogInTime = logtime; }
      char * GetPassWord() { return PassWord; }
      void SetPassWord(char * pass) { PassWord = pass; }

      char * GetResponsibilities() { return Responsibilities; }
      void SetResponsibilities(char * response) { Responsibilities =
response; }

      void setDesc(char * des) { Desc = des; }
      char * getDesc() { return Desc; }
   //Data members
   private:

      int ManID;
      double LogInTime;
      char * PassWord;
      char * Responsibilities;
      char * Desc;
};
//-------------------------------------------------------------------------------------------
--------------------------------
```

II. Database Functions

```
// File 2 of 3 - Database Functions - 2004 C.  Germany - in header file
"DatabaseFunctions.h"

//-------------------------------------------------------------------------------------------
---------------------------------

//Function Prototypes
void SaveData(Employee * TheDatabase, int & NumObjects);
void LoadData(Employee * TheDatabase, int & NumObjects);
void Search(Employee * TheDatabase, int & NumObjects);


//-------------------------------------------------------------------------------------------
---------------------------------

void ListEmployeeIDs(Employee * Comp, int & Num) {

            //List The Employee IDs Available:
            cout << "\n\n-----------------------------------------\n";
            cout << "Valid employee IDs are: \n\n";

             for(int x = 0;x < Num; x++)
             {
                  //Need to offset for the fencepost by adding 1
                  cout << (Comp[x].GetEmpID()) << " ";
             }

             cout << "\n-----------------------------------------\n\n";
}
//-------------------------------------------------------------------------------------------
---------------------------------
bool VerifyPassword() {
     cout.flush();
     system("CLS");
     int ManagerID;
     string Password;
     cout << "\n   To add employees to this database, you must be a
department manager\n"
          << "   with a valid password and manager ID.";
     cout << "\n\n   Please enter your manager ID: ";
     cin >> ManagerID;
     if(ManagerID > 99 && ManagerID < 105)
     {
         cout << "\n\n   Please enter your password (case sensitive):
";
         cin >> Password;
         cout << "\n\n   You entered: " << Password << ".\n";
         switch(ManagerID) {
               case 100: if(Password == "blue")
                         {
                           cout << "\n   Ok., manager " << ManagerID
```

```
                                << ", your password is valid!\n";
                            return true;
                          }
                         break;

                case 101: if(Password == "red")
                          {
                             cout << "\n   Ok., manager " << ManagerID
                                  << ", your password is valid!\n";
                            return true;
                          }
                         break;

                case 102: if(Password == "green")
                          {
                             cout << "\n   Ok., manager " << ManagerID
                                 << ", your password is valid!\n";
                            return true;
                          }
                         break;

                case 103: if(Password == "yellow")
                          {
                            cout << "\n   Ok., manager " << ManagerID
                                 << ", your password is valid!\n";
                            return true;
                          }
                         break;

                case 104: if(Password == "white")
                          {
                            cout << "\n   Ok., manager " << ManagerID
                                 << ", your password is valid!\n";
                            return true;
                          }
                         break;

                default: cout << "Invalid.\n\n";
                        break;
                  } //close switch
          } //close if
          else { cout << "   Invalid Manager ID!\n\n"; }

          return false;

} //close function
//-------------------------------------------------------------------------------------------
-------------------------------
void EditEmployeeInformation(Employee * ACompany, int & NumEmployees) {
     int key = 0;
     cout << "\n   Welcome!  You may enter new Employee information
below:" << endl;
     ListEmployeeIDs(ACompany, NumEmployees);
     cout << "Which employee would you like to edit?  Enter the key
value: ";
     cin >> key;
```

```
      //Check for valid ID, grater than 0 and one less than NumEmp because of fencepost
      if(key > 0 && key <= NumEmployees)
         {  ACompany[key-1].EditEmployee(); }
      else { cout << "\nI am sorry, that is an invalid eployee ID.\n\n";
}
} // close function
//-------------------------------------------------------------------------------------------
---------------------------------
void DisplayEmployeeInformation(Employee * ACompany, int &
NumEmployees) {

      int key = 0;

      cout << "\n   Welcome!  You may display information for"
           << "the employees listed below:" << endl;
      ListEmployeeIDs(ACompany, NumEmployees);
      cout << "Which employee would you like to view?  Enter the key
value: ";
      cin >> key;
      //Check for valid ID, grater than 0 and one less than NumEmp because of fencepost
      if(key > 0 && key <= NumEmployees)
         {  ACompany[key-1].DisplayEmployee(); }
      else { cout << "\nI am sorry, that is an invalid eployee ID.\n\n";
}
} // close function
//-------------------------------------------------------------------------------------------
---------------------------------

void DeleteEmployee(Employee * ACompany, int & NumEmployees) {

      int key = 0;
      cout << "\n Warning! You are about to delete an employee record!"
<< endl;

      ListEmployeeIDs(ACompany, NumEmployees);

      cout << "Which employee record would you like to delete? Enter the
key value: ";
      cin >> key;

      //Check for valid ID, grater than 0 and one less than NumEmp because of fencepost
      if(key > 0 && key <= NumEmployees)
      {
            ACompany[key-1].Initialize();
      }

      else { cout << "\nI am sorry, that is an invalid eployee ID.\n\n";
}

} // close function

//-------------------------------------------------------------------------------------------
---------------------------------


void MainMenu(char & Continue, Employee * Company, int & NumEmp) {
      char MenuChoice;
```

```
      cout << "\n\n";
      cout << "\t*************************************************\n"
           << "\t*                 Database Menu                 *\n"
           << "\t*************************************************\n"
           << "\t*                                               *\n"
           << "\t*      E = Edit Employee Information            *\n"
           << "\t*      D = Display Employee Information          *\n"
           << "\t*      R = Erase Employee Information            *\n"
           << "\t*      A = Add Employee to Database              *\n"
           << "\t*      S = Save Data                             *\n"
           << "\t*      L = Load Data                             *\n"
           << "\t*      H = Search Database                       *\n"
           << "\t*      P = Populate Database (new district)      *\n"
           << "\t*      X = Exit Menu Function                    *\n"
           << "\t*                                               *\n"
           << "\t*************************************************\n";
      cout << "\n\nEnter a selection: ";
      cin >> MenuChoice;
      //Below, "tolower" transfroms a char to lowercase.
      switch(tolower(MenuChoice))
      {
           case 'e' : cout << "\nEdit Employee Information.\n";
                      EditEmployeeInformation(Company, NumEmp);
                       break;
           case 'd' : cout << "\nDisplay Employee Information\n";
                      DisplayEmployeeInformation(Company,
NumEmp);
                       break;

           case 'r' : cout << "Remove Employee.";
                       DeleteEmployee(Company, NumEmp);
                      break;
           case 's' : cout << "Saving Data.";
                       SaveData(Company, NumEmp);
                      break;
           case 'l' : cout << "Loading Data.";
                       LoadData(Company, NumEmp);
                      break;
           case 'h' : cout << "Search Database.";
                       Search(Company, NumEmp);
                      break;
           case 'p' : cout << "Populating database for a new district.";
                      break;
           case 'x' : cout << "Exiting database menu...\n\n";
                      Continue = 'q';
                       break;
           default :  cout << "Invalid response...\n\n";
                      break;

      } //close switch statement
} //close function
//-------------------------------------------------------------------------------------------
---------------------------------

void SaveData(Employee * TheDatabase, int & NumObjects) {
```

```
      ofstream OutputFile("Data.dat", ios::binary);

      for(int x = 0; x < NumObjects; x++)
      {
          OutputFile.write((char*) &TheDatabase[x], sizeof
TheDatabase[x]);
      }

      OutputFile.close();

} //close function

//-------------------------------------------------------------------------------------------
---------------------------------

void LoadData(Employee * TheDatabase, int & NumObjects) {

      ifstream InputFile("Data.dat", ios::binary);

      if(!InputFile)
      {
         cout << "Unable to find Data.dat for reading!\n";
      }

      for(int x = 0; x < NumObjects; x++)
      {
          InputFile.read((char*) &TheDatabase[x], sizeof
TheDatabase[x]);
      }

      InputFile.close();
} //close function

//-------------------------------------------------------------------------------------------
---------------------------------

void Search(Employee * TheDatabase, int & NumObjects) {

      char choice;
      int x = 0;
      int TheSame;
      long SearchLong;
      bool FoundSomething = false;
      string SearchString;
      string Data;

      cout << endl << endl;
      cout << "**********************************\n";
      cout << "*                                *\n";
      cout << "* F - First Name                 *\n";
      cout << "* L - Last Name                  *\n";
      cout << "* J - Job Description            *\n";
      cout << "* H - Hire Date                  *\n";
      cout << "* R - Review Date                *\n";
      cout << "* S - Social Security Number     *\n";
      cout << "* A - Salary                     *\n";
```

```
     cout << "* D - Department ID              *\n";
     cout << "* E - Employee ID                *\n";
     cout << "*                                *\n";
     cout << "**********************************\n";

     cout << "\nWhat category would you like to search by? ";
     cin >> choice;

     switch(tolower(choice))
     {
       //-------------------------------------------------------------------------------
-------------------------
          case 'f' : cout << "\nSearch by first name: \n";
                     cout << "Enter first name to search for: ";
                     cin >> SearchString;

                     for(x = 0; x < NumObjects; x++)
                     {
                         Data = TheDatabase[x].GetFirstName();
                         TheSame = Data.compare(0,20,SearchString);

                         if(TheSame == 0)
                         {
                             cout << "\nA record was found!";
                             TheDatabase[x].DisplayEmployee();
                             cout << endl << endl;
                             FoundSomething = true;
                         }
                     } //close for loop

                     break;
       //-------------------------------------------------------------------------------
-------------------------
          case 'l' : cout << "\nSearch by last name: \n";
                     cout << "Enter last name to search for: ";
                     cin >> SearchString;

                     for(x = 0; x < NumObjects; x++)
                     {
                         Data = TheDatabase[x].GetFirstName();
                         TheSame = Data.compare(0,20,SearchString);

                         if(TheSame == 0)
                         {
                             cout << "\nA record was found!";
                             TheDatabase[x].DisplayEmployee();
                             cout << endl << endl;
                             FoundSomething = true;
                         }
                     } //close for loop

                     break;
       //-------------------------------------------------------------------------------
-------------------------
          case 'j' : cout << "\nSearch by job description: \n";
                     cout << "Stubbed out!";
                     break;
```

```
        //-------------------------------------------------------------------------------------------
-------------------------
            case 'h' : cout << "\nSearch by hire date: \n";
                       cout << "Stubbed out!";
                       break;
        //-------------------------------------------------------------------------------------------
-------------------------
            case 'r' : cout << "\nSearch by review date: \n";
                       cout << "Stubbed out!";
                       break;
        //-------------------------------------------------------------------------------------------
-------------------------

            case 's' : cout << "\nSearch by social security number: \n";
                       cout << "Enter SSN to search for: ";
                       cin >> SearchLong;

                       for(x = 0; x < NumObjects; x++)
                       {
                            if(SearchLong == TheDatabase[x].GetSSN())
                            {
                                 cout << "\nA record was found!";
                                 TheDatabase[x].DisplayEmployee();
                                 cout << endl << endl;
                                 FoundSomething = true;
                            }
                       } //close for loop

                       break;
        //-------------------------------------------------------------------------------------------
-------------------------
            case 'a' : cout << "\nSearch by salary: \n";
                       cout << "Stubbed out!";
                       break;

        //-------------------------------------------------------------------------------------------
-------------------------

            case 'd' : cout << "\nSearch by department ID: \n";
                       cout << "Stubbed out!";
                       break;
        //-------------------------------------------------------------------------------------------
-------------------------

            case 'e' : cout << "\nSearch by employee ID: \n";
                       cout << "Stubbed out!";
                       break;
        //-------------------------------------------------------------------------------------------
-------------------------

            default : cout << "Invalid response...\n\n";
                      break;
        //-------------------------------------------------------------------------------------------
-------------------------

      } //close switch statement

      if(FoundSomething == false)
      { cout << "Sorry, no records were found."; }
```

```
      else
      { cout << "\nNo more matching records found in database."; }
} //close function

//-------------------------------------------------------------------------------------------------
---------------------------------


//This stuff below is just junk for demonstrations and debugging tutorials.
void TestStuff() {
      char MoveOn;
      //Object test
      Employee Newbie;
      Newbie.SetFirstName("Charles");
      cout << "\n" << Newbie.GetFirstName() << "\n";
      Manager Buffy;
      Buffy.SetFirstName("Buffy");
      cout << "\nManager object named: " << Buffy.GetFirstName() <<
"\n";
      Buffy.setDesc("Vampire Slayer");
      cout << "Buffy Description: " << Buffy.getDesc() << endl;
      cout << "\n\n\nType C followed by return to continue.  ";
      cin >> MoveOn;

}
//-------------------------------------------------------------------------------------------------
---------------------------------
//The function below is no longer used.  It was a precursor introduction to static member data.
//We are now implementing inline in our constructor what this function used to do.
void AssignUniqueIDs(Employee * ACompany, int & NumEmployees) {

      //Note: This should only get called ONCE, unless there is a new database.
      int key = 1;
      for(int x = 0; x < NumEmployees; x++)
      {
                  ACompany[x].SetEmpID(key);
                 ACompany[x].SetFirstName("NewEmployeeFirst");
                 ACompany[x].SetLastName("NewEmployeeLast");
                 ACompany[x].SetJobDescription("NewEmployeeDescription");
                  key = key + 1;
      }
}

//-------------------------------------------------------------------------------------------------
---------------------------------
```

III. Database Main Function

```
// File 3 of 3 Database Main Function - 2004 C. Germany - in source
file "MainFile.cpp"

#include "DatabaseClasses.h"
#include "DatabaseFunctions.h"
```

```
//-------------------------------------------------------------------------------------------
---------------------------------

void main()
{
      int NumEmps = 0;
       //Set sentinel value
      char Continue = 'z';
      //Stuff to happen only once
      cout << "Enter the number of employees in your company: ";
      cin >> NumEmps;
      //We need to make sure they enter something valid
      if(NumEmps > 0 && NumEmps < 500)
      {
            Employee * TheCompany = new Employee[NumEmps];

            //Stuff to keep happening until user selects exit
            while(Continue != 'q')
            {
                  MainMenu(Continue, TheCompany, NumEmps);

            } //end while true loop

            //Now that we're done, let's clean up our array of Employee objects on the heap.
            delete [] TheCompany;
            TheCompany = 0;
      }

      else
      {
         cout << "Sorry, you must enter a whole number integer value for
your database.";
      }

      cout << "\nExiting program...\n\n";

}   //close main()
```


C++ Console 32 Project 1: **AdventureGame (Other Projects)**

Other Projects for Console Adventure Games:

Objective: Demonstrate an understanding of polymorphism, inheritance, pointers and references, function and class creation and usage, passing by reference and value, data types, memory management, scope, templates and OOP theory in a console program.

Files In Project: Total of 5. 1-Gameclasses.h, 2-GameFunctions.h, 3-GameTemplate.h, 4-GameCharacterClass.h, and 5-AdventureGame7.cpp which is the main() file. An example command-line argument class file is included showing how it would be used in main().

I. Game Classes

```
// File 1 of 5 - Game Classes - ©2002 C. Germany - in header file
"GameClasses.h"

// Header Files Included in Program
#include <iostream.h>    //Needed for basic I/O operations
#include <stdlib.h>
#include <time.h>        //Needed for pause
#include <stdio.h>       //for old printf() function
#include <fstream.h>     //Needed for saving and reading highscores and
game character
#include <string.h>
#include "GameTemplate.h"

// Function Prototypes for use in classes and main().  Even though we
are only declaring
// them, not defining them here in this class file, it is necessary
since we invoke
// functions in these classes that will not be defined until later in
GameFunctions.h
// Also, notice we have 4 OVERLOADED Attack() functions.  Some take 1
argument, others 2.
// Some take a pointer, other versions of Attack() take a reference.

int GenerateRandomNumber(int Number);
void wait(int seconds);
void Attack(int * PlayerPoints);
void Attack(int * PlayerPoints, int Ouch);
void Attack(int & PlayerPoints);
void Attack(int & PlayerPoints, int Ouch);
void SetTheScene(char PName[25]);
void ArrayOfMonsters(int & PlayerPoints, int & PlayerScore);

/*-------------------------------------------------------------------
----------------------*/
// Base CLASS - 3 Overloaded Constructors

class Monster {

public:

        // Recall that unlike other data members that exist
individually for
        // each object that instantiates from a class, STATIC data
members
        // exist as a SINGLE instance for ALL objects of that class. So
MonsterCount
        // is shared by all Monster objects.  STATIC variables must be
defined
        // and initialized OUTSIDE the class.
        static int MonsterCount;
```

```
        // Overloaded Constructor 1 Takes 1 argument, supplies a
DEFAULT PARAMETER if none given.
        Monster(int MonStrength = 10)
        {
               MonsterCount++;
               Strength = MonStrength;
               cout << "You hear a monstrous, chilling, blood curdling "
                    << "sound in the distance.\n";
        }

        // Overloaded Constructor 2
        Monster(int MonStrength, int MonWeight)
        {
               MonsterCount++;
               Strength = MonStrength;
               Weight = MonWeight;
               cout << "You hear a monstrous, chilling, blood curdling "
                    << "sound in the distance.\n";
        }

        // Overloaded Constructor 3
        Monster(int MonStrength, int MonWeight, char MonName[10])
        {
               MonsterCount++;
               Strength = MonStrength;
               Weight = MonWeight;
               Name[10] = MonName[10];
               cout << "You hear a monstrous, chilling, blood curdling "
                    << "sound in the distance.\n";
        }

        // Destructor
        ~Monster() { cout << "A monster has been destroyed!\n"; }



        //Accessor Methods - Needed to Access Private Data Members
        static int CountMonsters() { return MonsterCount; }
        int getStrength() { return Strength; }
        void setStrength(int MonsStrength) { Strength = MonsStrength; }
        int getWeight() { return Weight; }
        void setWeight(int Wt) { Weight = Wt; }
         char getName() { return Name[10]; }
        void setName(char Nm[10]) { Name[10] = Nm[10]; }

        // Virtual method, though not pure, will be implemented
separately by
        // each class that derives from Monster
        virtual void Talk() {
                int SayWhat;
                SayWhat = GenerateRandomNumber(4);
                cout << "\nThe monster looks at you and says, \n\"";

                switch(SayWhat) {
```

```
                    case 1 : cout << "I like flowers.  Will you be my
friend?";
                             break;
                    case 2 : cout << "I would really like to dismember
you and eat you...";
                             break;
                    case 3 : cout << "Unfortunately, I have had a very
bad day, and\n";
                             cout << "you happen to be the first
creature I\'ve met\n";
                             cout << "that I can take it out on...\n";
                             break;
                    case 4 : cout << "How do you taste?  I don\'t think
I\'ve eaten"
                                  << " your kind before...";
                             break;
                    default : cout << "Uh oh, this should never
happen...";
                } //closes switch
                cout << "\".\n";
        } //close talk function


        //Virtual method, will be implemented differently by each
derived class.
        virtual void AttackMonster(int & PlayerLife, int & score) {
                int PutaHurtinOnYou;

               while(Strength > 0 && PlayerLife > 0) {
                    cout << "\nYou attack the Monster!";
                     PutaHurtinOnYou = GenerateRandomNumber(10);
                     Strength = Strength - PutaHurtinOnYou;
                    cout << "\nMonster is now of strength: " <<
Strength << ".\n";

                      if(Strength > 0) {
                           cout << "\nThe monster attacks you!";
                           // PlayerLife passed by reference, so
manipulated as global
                          Attack(PlayerLife);
                           cout.flush();
                           wait(2);
                       } // close if statement
               } //close while true loop

               if(Strength <= 0) {
                    cout << "\nYou did it!  You killed the monster!";
                    score = score + 10;
                    cout << "\nYour score is now: " << score << ".";  }

               if(Strength < 0) {
                     cout << "\nThe monster has strength " << Strength
                          << "?  You sly fox!  You inflicted"
                         << "\npost-mortem humiliation on your foe!
Good job!"; }
```

```
        } //close function


// Data members
protected:
        int Strength;
        int Weight;
        char Name[10];

}; //closes class specification

/*-------------------------------------------------------------------------
-----------------------*/

// Since MonsterCount is static, we must define/initialize it here
outside the class.
int Monster::MonsterCount = 0;

/*-------------------------------------------------------------------------
-----------------------*/
//Giant class derives or "inherits from" Monster

class Giant : public Monster {

public:
        // Overloaded Constructor 1 - Takes no arguments.
        Giant() {  cout << "A giant!\n";  }

        // Overloaded Constructor 2 - Takes 1 argument.
        Giant(int ft)
        {
              cout << "A giant!\n";
              Footsize = ft;
        }

        // Destructor
        ~Giant() {  cout << "Bye bye giant...";  }


   //Accessor Methods
   int getFootsize() { return Footsize; }
   void setFootSize(int GiantFeet) { Footsize = GiantFeet; }


   //Overriding Base Class Virtual Function of Talk()
   void Talk() {
        int SayWhat;
        SayWhat = GenerateRandomNumber(4);
        cout << "\nThe lumbering giant looks at you and says, \"";

        switch(SayWhat) {
            case 1 : cout << "\nAwwwe, little people are so tiny and
cuddly...\n";
                     cout << "I want to hug you and squeeze you and call
you\n";
```

```
                    cout << "my very own...";
                    break;
            case 2 : cout << "\nMmmm...bite size.  You would be a tasty
snack...";
                    break;
            case 3 : cout << "\nYou little people give me the
creeps...";
                    break;
            case 4 : cout << "\nShort people got no reason to live...";
                    break;
            default : cout << "Uh oh, this should never happen...";
        } //closes switch
        cout << "\".\n";
        } //close talk function


   //Other Functions
   void GiantSpeak() { cout << "\nFee Fi Fo Fum..."; }
   void Squash(int * Life)
   {
        cout << "\nThe giant squashes you like a bug!";
        // Take a pointer to Life.  On dereferencing, set global life
to 0, game over.
        *Life = 0;
   }

    // Overriding AttackMonster method from base class.
    void AttackMonster(int & PlayerLife, int & score) {
         int PutaHurtinOnYou;

         while(Strength > 0 && PlayerLife > 0) {
               cout << "\nYou attack the Giant!";
               PutaHurtinOnYou = GenerateRandomNumber(10);
               Strength = Strength - PutaHurtinOnYou;
               cout << "\nGiant is now of strength: " << Strength <<
".\n";

               if(Strength > 0) {
                   cout << "\nThe Giant attacks you!";
                  Attack(PlayerLife);
                   cout.flush();
                   wait(2);
               } //close if statement

         } //close while true loop

         if(Strength <= 0) {
               cout << "\nYou did it!  You killed the Giant!";
               score = score + 120;
               cout << "\nYour score is now: " << score << "."; }

         if(Strength < 0) {
               cout << "\nA giant with strength " << Strength
                    << "?  You are BAD to the BONE!  You inflicted"
```

```
                    << "\npost-mortem humiliation on the giant. Good
job!"; }
        } //close function

private:
        // Data members
        int Footsize;

}; //close class specification

/*-------------------------------------------------------------------
----------------------*/
// Troll class derived from Monster

class Troll : public Monster {

public:
        // Constructor
        Troll() { cout << "\nIt is a troll, in a very bad mood!"; }

        // Destructor
        ~Troll() { cout << "Bye troll..."; }


   //Accessor Methods
   int getDrool() { return Drool; }
   void setDrool(int TrollDrool) { Drool = TrollDrool; }


   //Overriding Base Class Virtual Function of Talk()
   void Talk() {
        int SayWhat;
        SayWhat = GenerateRandomNumber(4);
        cout << "\nThe ugly, drooling troll looks at you and says, \"";

        switch(SayWhat) {
             case 1 : cout << "\nI hate everything.";
                      break;
            case 2 : cout << "\nSmell my feet?  Are you good to eat?";
                      break;
            case 3 : cout << "\nI am a Troll - get over it...";
                      break;
            case 4 : cout << "\nDon\'t hate me because I'm
beautiful...";
                      break;
            default : cout << "Uh oh, this should never happen...";
        } //closes switch
        cout << "\".\n";
   } //close talk function


   //Other Functions
   void TrollSpeak() { cout << "Ahhhhhhhhhhhhhh..."; }
   void Eat(int * Life) {
```

```
            cout << "It dismembers you and boils your carcass in a
stew.";
             *Life = 0; }
   void Drooling() {
        setDrool(10);
        cout << "\nThe troll salivates, dropping " << getDrool()
             << " gallons of drool onto your shiny new boots.\n";}

   //Overriding AttackMonster function from base class Monster
   void AttackMonster(int & PlayerLife, int & score) {
        int PutaHurtinOnYou;

        while(Strength > 0 && PlayerLife > 0) {
              cout << "\nYou attack the Troll!";
              PutaHurtinOnYou = GenerateRandomNumber(10);
              Strength = Strength - PutaHurtinOnYou;
              // Remember that Strength is inherited from Monster
protected data member.
              cout << "\nTroll is now of strength: " << Strength <<
".\n";

                if(Strength > 0) {
                   cout << "\nThe Troll attacks you!";
                  Attack(PlayerLife);
                   cout.flush();
                   wait(2);
                } //close if statement

        } //close while true loop

        if(Strength <= 0) {
           cout << "\nYou did it!  You killed the Troll!";
           score = score + 7;
           cout << "\nYour score is now: " << score << "."; }

        if(Strength < 0) {
           cout << "\n" << Strength << "? Player, you are Mr. Cool!  You
inflicted"
                << "\npost-mortem humiliation on that ugly old troll!
Good job!"; }
   } //close function

private:
        // Data Members
        int Drool;

}; //close class specification

/*-------------------------------------------------------------------
----------------------*/
// BarryManilow class derived from Monster

class BarryManilow : public Monster {

public:
```

```
        // Constructor
        BarryManilow() { cout << "\nRun for your life - it\'s Barry!"; }

        // Destructor
        ~BarryManilow() { cout << "I write the songs that..."; }


        // Accesor Methods
        int getSong() { return Song[20]; }
        void setSong(int BarrySong[20]) { Song[20] = BarrySong[20]; }


        // Overriding Base Class Virtual Function of Talk()
        void Talk() {
             int SayWhat;
             SayWhat = GenerateRandomNumber(4);
              cout << "\nScary Barry Manilow looks at you and says, \"";

             switch(SayWhat) {
                  case 1 : cout << "\nI\'m good enough, I\'m smart
enough, \n"
                                << "and, dog gone it, people like me...";
                           break;
                 case 2 : cout << "\nI write the songs that make the
whole world sing...";
                           break;
                 case 3 : cout << "\nAt the Copa, Copa Cabana...";
                           break;
                 case 4 : cout << "\nOh Mandy...";
                           break;
                 default : cout << "Uh oh, this should never happen...";
             } //closes switch
              cout << "\".\n";
        } //close talk function


        // Other Functions
        void BarrySing(int * Life) {
             cout << "Barry lifts his voice to sing a ballad...";
             cout << "\nYou feel the sudden urge to pull out your own\n"
                  << "sword and kill yourself, right then and there.\n"
                  << "Barry, for lack of an interested audience, \n"
                  << "continues serenading your rotting corpse...\n";
              *Life = 0; }

        // Overriding AttackMonster function in base class Monster
        void AttackMonster(int & PlayerLife, int & score) {
             int PutaHurtinOnYou;

             while(Strength > 0 && PlayerLife > 0) {
                   cout << "\nYou attack Barry Manilow!";
                    PutaHurtinOnYou = GenerateRandomNumber(10);
                    Strength = Strength - PutaHurtinOnYou;
                   cout << "\nBarry Manilow is now of strength: " <<
Strength << ".\n";
```

```
                    if(Strength > 0) {
                       cout << "\nBarry Manilow serenades you! "
                            << "You feel like hurting yourself.";
                       cout << "\nYou pull out your own sword and cut
your own flesh.";
                        // PlayerLife and score passed by reference, so
globalized.
                       Attack(PlayerLife);
                        cout.flush();
                        wait(2);
                    } //close if statement

             } //close while true loop

             if(Strength <= 0) {
                cout << "\nYes! You did it!  You killed Barry Manilow!";
                score = score + 300;
                cout << "\nYour score is now: " << score << ".";  }

              if(Strength < 0) {
                 cout << "\n" << Strength << "? Player, you have
excellent musical "
                      << "tastes.  You inflicted \npost-mortem
humiliation on"
                     << " Barry Manilow!  The world rejoices!"; }
        } //close function

private:
        // Data Members
        int Song[20];

}; //close class specification

/*-------------------------------------------------------------------------
-----------------------*/
// BrainEatingZombie class derived from Monster

class BrainEatingZombie : public Monster  {

public:

        // Overloaded Constructor - Takes no parameters
        BrainEatingZombie() {
                        cout << "Brains! Brains! They smell so good
and spicy...\n"; }

        // Overloaded Constructor - Takes 1 argument
        BrainEatingZombie(int weight) {
                        itsWeight = weight;
                        cout << "Brains! Brains! They smell so good
and spicy...\n";  }
        // Destructor
        ~BrainEatingZombie() {  cout << "Bye zombie . . .";  }
```

```
        // Accessors
        void SetWeight(int wt) {  itsWeight = wt;  }
        int GetWeight() const { return itsWeight; }
        void Display() const { cout << itsWeight; }
        void setHunger(bool hunger) { HungryForBrains = hunger; }
        bool getHunger() { return HungryForBrains; }

        // Other Methods
        void CorpseTalk() {
               cout << "Please let me eat your brain... please?\n";
               cout << "Just a little bit... please?\n";  }
private:
        // Data Members
        int itsWeight;
        bool HungryForBrains;
};
```

II. Game Functions

```
// File 2 of 5 - Game Functions - 2002 C. Germany - in header file
"GameFunctions.h"
// Declare a namespace.  Must use "WRONG" to bring it into scope.

namespace WRONG { class WrongAnswer {}; }

/*-------------------------------------------------------------------
----------------------*/

// Function - Difficulty to Choose Difficulty
//Note: Code below would be better suited to a switch statement but
used
//to illustrate a nested decision structure used as a switch statement.

void ChooseDifficultyLevel(int & ALIVE) {
     int Difficulty = 0;
     try {

          while(Difficulty != 1 || 2 || 3){
               cin >> Difficulty;

               if(Difficulty == 1) {
                  ALIVE = 200;
                  break; }

               if(Difficulty == 2) {
                  ALIVE = 100;
                  break; }

               if(Difficulty == 3) {
                  ALIVE = 50;
                  break; }

               cout << "Invalid Response.  Please enter 1, 2, or 3: ";
```

```
          } //close while true loop

     } //close try block
     catch(WRONG::WrongAnswer) {

            cout << "It appears that you cannot even follow simple
directions.\n";
            cout << "Too bad.  You are too simple to play this game.\n";
            cout << "Have a nice day. :) ";
            cout.flush();
            system("PAUSE");
            system("CLS");

     } //close catch block
} //close function

/*-------------------------------------------------------------------
----------------------*/

// Function - Random to Generate a Random Number, default parameter of
6.
//srand seeds the generator for rand.

int GenerateRandomNumber(int Number=6) {

    int ResultRandom;
    srand(time(NULL));
    ResultRandom = (rand()%Number) + 1;
    return ResultRandom;

    }
/*-------------------------------------------------------------------
----------------------*/

// Function - Pause to pause program for number of seconds passed in as
argument

void wait(int seconds) {

    // Define start and end to difftime().
    time_t start;
    time_t end;

    // Define variable to find time elapsed.
    double elapsedtime = 0;

    //  Explicit Type Cast - not used in this example!
    //  elapsedtime = static_cast <double> (elapsedtime);

    // Start timer, pass in address of start by reference.
    time (&start);

    // Create pause.
    while (elapsedtime < seconds)
    {
```

```
        time (&end);
        elapsedtime = difftime(end, start);
    }  // end while true loop

}  // end function

/*------------------------------------------------------------------------
-----------------------*/
//Four overloaded Attack functions - 1 takes a pointer, 2 takes a
pointer
//and an int, 3 takes a reference, and 4 takes a reference and an int.

/*------------------------------------------------------------------------
-----------------------*/

void Attack(int * PlayerPoints) {

     int Damage;
     Damage = GenerateRandomNumber(10);
     *PlayerPoints = *PlayerPoints - Damage;
     cout << "Ouch!  You loose " << Damage << " points.\n";
     cout << "You now have " << *PlayerPoints << " points left!";

}

/*------------------------------------------------------------------------
-----------------------*/

void Attack(int * PlayerPoints, int Ouch) {

     int Damage;
     Damage = GenerateRandomNumber(Ouch);
     *PlayerPoints = *PlayerPoints - Damage;
     cout << "Ouch!  You loose " << Damage << " points.\n";
     cout << "You now have " << *PlayerPoints << " points left!";

}

/*------------------------------------------------------------------------
-----------------------*/

void Attack(int & PlayerPoints) {

     int Damage;
     Damage = GenerateRandomNumber(10);
     PlayerPoints = PlayerPoints - Damage;
     cout << "Ouch!  You loose " << Damage << " points.\n";
     cout << "You now have " << PlayerPoints << " points left!";

}

/*------------------------------------------------------------------------
-----------------------*/

void Attack(int & PlayerPoints, int Ouch) {
```

```
     int Damage;
     Damage = GenerateRandomNumber(Ouch);
     PlayerPoints = PlayerPoints - Damage;
     cout << "Ouch!  You loose " << Damage << " points.\n";
     cout << "You now have " << PlayerPoints << " points left!";

}

/*-------------------------------------------------------------------
----------------------*/
// Function - SetTheSene uses GenerateRandomNumber function to set
random scene.
// Takes char array (or "string") as an argument.
// Example of using a global value locally inside a function

void SetTheScene(char PName[25]) {

     int Setting;
     Setting = GenerateRandomNumber();

     switch (Setting) {
         case 1 : cout << PName << ", you are in a swamp and
sinking!\n";
                  break;
         case 2 : cout << PName << ", you are on a high mountain
peak.\n";
                  break;
         case 3 : cout << PName << ", you are on a grassy plane.\n";
                  break;
         case 4 : cout << PName << ", you are in the desert.\n";
                  break;
         case 5 : cout << PName << ", you find yourself in a deserted
village.\n";
                 break;
         case 6 : cout << PName << ", you find yourself in a steamy
jungle.\n";
                  break;
         default : cout << "Something is definitely wrong here.  You
should not be here.";
     } // close switch statement

} // close SetTheScene() function


/*-------------------------------------------------------------------
----------------------*/
// Function - ArrayOfMonsters - demonstrate creating an array of
objects.

void ArrayOfMonsters(int & PlayerPoints, int & PlayerScore) {

     cout.flush();
     system("CLS");
     int loopcount;
```

```
     char choice;
     //MonsterMash is a pointer to an array of 5 monsters on the heap.
     Monster * MonsterMash = new Monster[5];

     for(loopcount = 0; loopcount < 5; loopcount++) {
         MonsterMash[loopcount].setStrength((loopcount+1) * 10);
     }

     cout << "\nYou encounter a gang of 5 unruly monsters.  Their stats
are as follows:\n";

     for(loopcount = 0; loopcount < 5; loopcount++) {
         cout << "\nMonster " << loopcount+1 << " is of strength "
              << MonsterMash[loopcount].getStrength() << "!";
     }

     cout << "\nYour choices are:\n\n";
     cout << "f = fight\n"
          << "r = run\n"
          << "t = talk\n\n";
     cout << "\nWhat do you choose to do? ";

     cin >> choice;

     switch(choice) {
     case 'f' : cout << "\nYou choose to fight...";
                for(loopcount = 0; loopcount < 5; loopcount++) {
                    MonsterMash[loopcount].AttackMonster(PlayerPoints,
PlayerScore);
                }
                break;
     case 'r' : cout << "\nYou choose to run...";
                cout << "\nYou run away and survive!\n"
                     << "The creature inflicts minimal damage.\n\n";
                Attack(&PlayerPoints);
               PlayerScore = PlayerScore - 10;
               cout << "\nYour score is now:" << PlayerScore << ".";
                break;
     case 't' : cout << "\nYou choose to converse with the group...";
                for(loopcount = 0; loopcount < 5; loopcount++) {
                    MonsterMash[loopcount].Talk();
               } // close loop
                Attack(&PlayerPoints);
                break;
     default :  cout << "\nPlease enter a valid response.";

     } //close switch statement

 delete [] MonsterMash;  //clean up array of Monster objects, delete
entire array.

} // close function

/*-------------------------------------------------------------------
----------------------*/
```

```
// Function - DragonEncounter

void DragonEncounter(int & LIFE, char conflict) {

     cout << "\n";
     //Instantiate a Monster called Dragon of strength 20
     Monster Dragon(20);

     cout << "\nYou encounter a dragon!  He opens razor sharp teeth and
hisses!  \n\n";

     Dragon.Talk();

     cout << "\nHe appears to be highly agitated.\n";
     cout << "\nYou sense that things are suddenly taking a turn for
the worse...\n\n";
     cout.flush();
     system("PAUSE");
     system("CLS");

     cout << "The dragon blasts you with fire! \n";
     cout.flush();
     wait(3);
     Attack(LIFE, 50);
     cout.flush();
     wait(3);
     cout << "\n\nHe is of strength ";
     cout << Dragon.getStrength();
     cout << ".\n\n";
     cout.flush();
     wait(4);
     cout << "\n\nYou unsheath your sword...\n";
     cout.flush();
     wait(4);
     cout << "\nIt shines like silver moonlight...\n";
     cout.flush();
     wait(4);
     cout << "\nIt begins to quiver, thirsting for the blood of your
enemy...\n";
     cout.flush();
     wait(4);
     cout << "\nYou thrust with all your strength, plunging it into
scale and flesh...\n";
     cout.flush();
     wait(4);
     cout << "\nYou levy the weight of your blade upon the dragon\'s
head.\n";
     cout.flush();
     wait(5);
     system("CLS");

     Dragon.setStrength(5);

     cout << "\nThat was some move! The dragon is already half
dead!\n";
```

```
     cout << "\nHe now only has the strength of ";
     cout << Dragon.getStrength() << ".\n";
     cout << "\n\nNow what do you choose to do?\n"
          << "r = run away\n"
          << "s = stay and fight\n"
          << "t = talk to the dragon\n";

     cin >> conflict;

     system("cls");

     switch(conflict) {
         case 'r' : cout << "\nYou run away and survive!\n"
                         << "The dragon inflicts what is merely a flesh
wound.\n\n";
                    Attack(LIFE);
                    break;
         case 's' : cout << "\nYou stay and fight.  You discover
that\n"
                         << "the dragon is merely a baby dragon.  It\'s
mother\n"
                         << "then shows up, greatly enraged by your
attempt to\n"
                         << "kill her child.  \n\nShe is really big and
really angry.\n";

                    Dragon.setStrength(200);

                    cout << "\n\nYou are now fighting a dragon of
strength: ";
                   cout << Dragon.getStrength() << ".\n";
                   cout << "The dragon rips off your arms and legs
and\n"
                        << "fries you to a crispy, crunchy, golden
brown.\n";
                    LIFE = 0;
                   break;
         case 't' : cout << "\nYou strike up a conversation with the
dragon.\n"
                          << "You begin telling it your life story and
all your woes.\n"
                         << "Succumbing to fatal levels of nausea and
boredom, the \n"
                         << "dragon lapses into a coma and dies...\n\n";

                    Dragon.setStrength(0);

                    cout << "Dragon reduced to strength: ";
                    cout << Dragon.getStrength() << ".\n";
                    Dragon.~Dragon();
                   cout << "You have " << LIFE << " points left.";
                   break;
         default :  cout << "Invalid input.  Please press either: r, s,
or w.";
     } //close switch statement
```

```
  cout.flush();
  system("PAUSE");

}  //close function

/*------------------------------------------------------------------------
-----------------------*/
// Function - SaveHighScores

void SaveHighScores(char Name[25], int score) {

     ofstream highscores("hiscore.txt", ios::app);
     highscores << "\nName: " << Name << "     "
                << "Score: " << score << "     "
               << "Monsters Fought: " << Monster::CountMonsters() << "
";
     highscores.close();

}  //close function

/*------------------------------------------------------------------------
-----------------------*/
// Function - DisplayHighScores

void DisplayHighScores() {

     char contents;
     ifstream highscores("hiscore.txt");
     //Header
     cout << "High scores for the game are:\n\n";
     while(highscores.get(contents)){
          cout << contents;
     }
     highscores.close();

}

/*------------------------------------------------------------------------
-----------------------*/
//Function - Random Monster Attack Sequence for Main(), creates monster
on the heap

void RandomMonsterAttackSequence(int & ALIVE, int & score, char &
conflict) {

     int UndeterminedMonster;
     Monster * RandomMonster;
     while(ALIVE > 0) {
          cout << "\n\nYou continue traveling eastward.  You see
something!\n\n";
          UndeterminedMonster = GenerateRandomNumber(3);

          switch (UndeterminedMonster) {
                  case 1 : RandomMonster = new Giant;
```

```
                              break;
                    case 2 : RandomMonster = new Troll;
                              break;
                    case 3 : RandomMonster = new BarryManilow;
                              break;
                    default : cout << "No monsters here...";
               } //close switch
               RandomMonster->Talk();
               cout << "\n\n";
               cout.flush();
               system("PAUSE");
               cout << "What do you do now?\n";
               cout << "r = run away\n"
                    << "t = talk\n"
                    << "f = fight\n";

               cin >> conflict;
               system("cls");
               switch(conflict) {
                   case 'r' : cout << "\nYou run away and survive!\n"
                                   << "The creature inflicts minimal
damage.\n\n";
                              Attack(&ALIVE);
                              score = score - 10;
                              cout << "\nYour score is now:" << score <<
".";
                              break;
                   case 'f' : cout << "\nYou stay and fight.";
                              RandomMonster->AttackMonster(ALIVE, score);
                              break;
                   case 't' : cout << "\nYou strike up a conversation with
the creature.\n";
                              RandomMonster->Talk();
                              cout << "\n\"";
                              Attack(&ALIVE);
                              break;
                   default :  cout << "Invalid input.  Please press either:
r, f, or w.";

               } //close switch statement

          } //close while true loop - ALIVE no longer > 0

          system("CLS");

} //close Random Monster Attack Sequence Function

/*-------------------------------------------------------------------------
-----------------------*/

// Function - ClassicMovie
void ClassicMovie() {

     //scoping a zombie pointer
     BrainEatingZombie * pFavoriteBMovieOfAllTime;
```

```
     //Three instantiations using our MonsterArray Template to handle
different data types
     ArrayOfSomeKindOfMonster<int> ArrayOfIntegersForTheMonsters;
     ArrayOfSomeKindOfMonster<BrainEatingZombie> NightOfTheLivingDead;
     cout << "Dare We Go On Any Further?";
     cout.flush();
     system("PAUSE");
     system("CLS");
        for (int i = 0; i < ArrayOfIntegersForTheMonsters.GetSize();
i++) {
              pFavoriteBMovieOfAllTime = new BrainEatingZombie(i*3);
             NightOfTheLivingDead[i] = *pFavoriteBMovieOfAllTime;
              NightOfTheLivingDead[i].CorpseTalk(); }

     cout << "Dare We Go On Any Further?";
     cout.flush();
     system("PAUSE");
     system("CLS");
     ArrayOfSomeKindOfMonster<Giant> JollyGreen;
     cout << "Dare We Go On Any Further?";
     cout.flush();
     system("PAUSE");
     system("CLS");

} //close function - the template class destructor will clean up after
us


/*-------------------------------------------------------------------*/
//Function - SetPlayerAttributesCheat Using Character Class declared
later.

void SetPlayerAttributesCheat(Player * CurPlayer) {
     int stren;
     int lev;
     int intel;
     int charis;
     int dex;
     char nam[15];
     cout << "\nWhat is new player\'s name?";
     cin >> nam;
     CurPlayer->setName(nam);

     cout << "\nWhat is new player\'s strength?";
     cin >> stren;
     CurPlayer->setStrength(stren);

     cout << "\nWhat is new player\'s level?";
     cin >> lev;
     CurPlayer->setLevel(lev);

     cout << "\nWhat is new player\'s intelligence?";
     cin >> intel;
     CurPlayer->setIntelligence(intel);
```

```
     cout << "\nWhat is new player\'s charisma?";
     cin >> charis;
     CurPlayer->setCharisma(charis);

     cout << "\nWhat is new player\'s dexterity?";
     cin >> dex;
     CurPlayer->setDexterity(dex);

} //close SetPlayerAttributes function
```

III. Game Template

```
//File 3 of 5 - Game Template - 2002 C. Germany in template header file
"GameTemplate.h"

template <class T>  // declare the template and the generic parameter.
"T"
                    // represents whatever object we will pass in to
the template.

/*-------------------------------------------------------------------
----------------------*/

// ArrayOfSomeKindOfMonster class

class ArrayOfSomeKindOfMonster            // the class being
parameterized
{
public:
        // Constructor For some "odd" reason, Monsters come in sets of 3
        //Remember what the songs says - "3 is a magic number"... :)

        ArrayOfSomeKindOfMonster(int size = 3) {
             itsSize = size;

             //We are saying that pType will become a pointer of
whatever type of object
             //we pass to the template and will point to an array of
whatever type
             //of object we pass to the template

            pType = new T[size];

             for (int i = 0; i < size; i++) {
                  pType[i] = 0; }
        }
        // Copy constructor
        ArrayOfSomeKindOfMonster(const ArrayOfSomeKindOfMonster &rhs) {
               itsSize = rhs.GetSize();
```

```
                pType = new T[itsSize];
                for(int i = 0; i < itsSize; i++) {
                    pType[i] = rhs[i]; }
        }
        // Destructor
        ~ArrayOfSomeKindOfMonster() { delete [] pType; }

        // Overloaded Operator =
        ArrayOfSomeKindOfMonster& operator=(const
ArrayOfSomeKindOfMonster&) {
                if (this == &rhs)
                return *this;
                delete [] pType;
                itsSize = rhs.GetSize();
                pType = new T[itsSize];
                for (int i = 0; i < itsSize; i++)
                     pType[i] = rhs[i];
                return *this; }
        // 2 Overloaded Operator []
        T& operator[](int offSet) { return pType[offSet]; }

        const T& operator[](int offSet) const
        { return pType[offSet]; }
        // Accessors
        int GetSize() const { return itsSize; }
        // friend function prototype/declaration. Remember that a friend
        // function must be defined outside the class specification.

        friend void Intrude(ArrayOfSomeKindOfMonster<int>);
private:
        // T represents what this will be a class of - the template
        T *pType;
        int  itsSize;
};

/*-------------------------------------------------------------------------
-----------------------*/
// Friend Function - Intrude  Not a template, can only be used
// with int arrays! Intrudes into private data.

void Intrude(ArrayOfSomeKindOfMonster<int> theArrayOfSomeKindOfMonster)
{

        cout << "\n*** Intrude ***\n";

        for(int i = 0; i < theArrayOfSomeKindOfMonster.itsSize; i++) {
            cout << "i: " <<    theArrayOfSomeKindOfMonster.pType[i] <<
endl;  }

        cout << "\n";
     }
```

IV. Game Player Character Class

```
//File 4 of 5 - Character class - C. Germany 2002 in header file
"GameCharacterClass.h"

// Prototype declaration
class Player;
void SetPlayerAttributesCheat(Player * CurPlayer);
// Player class

class Player
{
public:

        // Constructor
        Player() {
                    name = "NewPlayerNow";
                    InitializeItems();
                     cout << "Player object created.\n";  }

        // Destructor
        ~Player() { cout << "\nPlayer character destroyed.\n"; }

        //Accessor Methods
         void setLevel(int lev) { level = lev; }
        int getLevel() { return level; }

        void setStrength(int str) { strength = str; }
        int getStrength() { return strength; }

        void setIntelligence(int intel) { intelligence = intel; }
        int getIntelligence() { return intelligence; }

        void setCharisma(int chrs) { charisma = chrs; }
        int getCharisma() { return charisma; }

        void setDexterity(int dex) { dexterity = dex; }
        int getDexterity() { return dexterity; }

        void setName(char na[15]) {
                     name = new char[15];
                     strcpy(name, na);
                      cout << "\nCharacter name set to " << name << ".";
}
        char * getName() { return name; }


         //Inventory Item Accessors
        void setSword(bool swd) { sword = swd; }
        bool getSword() { return sword; }
```

```
void setBow(bool lng) { longbow = lng; }
bool getBow() { return longbow; }

void setStaff(bool stf) { staff = stf; }
bool getStaff() { return staff; }

void setShield(bool shl) { shield = shl; }
bool getShield() { return shield; }

void setArmor(bool arm) { armor = arm; }
bool getArmor() { return armor; }

void DisplayInventory() {
     cout << "You have the following items:\n";
     if(sword) { cout << " sword, "; }
     if(longbow) { cout << " longbow, "; }
     if(staff) { cout << " staff, "; }
     if(shield) { cout << " shield, "; }
     if(armor) { cout << " armor, "; }
     if(!sword && !longbow && !staff && !shield && !armor) {
        cout << "\nIt appears that you have NOTHING!"
        cout << "\nAbsolutely nothing!\nHow sad...";
     }

} //close DisplayInventory function
//Game Cheat - gives all items
void AllItems() {
     sword = true;
     longbow = true;
     staff = true;
     shield = true;
     armor = true;
}
//Initialize Items
void InitializeItems() {
     level = 1;
     strength = 10;
     intelligence = 18;
     charisma = 17;
     dexterity = 16;
     sword = false;
     longbow = false;
     staff = false;
     shield = false;
     armor = false;
}

//Save Character
void SaveCharacter() {
     char CharacterName[80];
     cout << "Please enter a name for your character to save: ";
     cin >> CharacterName;
     ofstream fout(CharacterName,ios::trunc||ios::binary);
          if(!fout) {
```

```
                    cout << "Unable to create " << CharacterName
                         << " for writing.\n";
                }  //close if
            fout.write((char*) &*this,sizeof(*this));
            fout.close();
        } //close save character function


        //Load Character
        void LoadCharacter() {
            char CharacterName[80];
            cout << "Please enter the name of your character to load:
";
            cin >> CharacterName;
            ifstream fin(CharacterName,ios::trunc||ios::binary);
            if (!fin) {
                cout << "Unable to find " << CharacterName << " for
reading.\n";
            }
            fin.read((char*) &*this, sizeof(this));
            fin.close();
            cout << "Loaded player name is: " << name << ".\n";
            cout << "Loaded player strength is: " << strength << ".\n";
             cout << "Loaded player level is: " << level << ".\n";
            cout << "Loaded player intelligence is: " << intelligence
<< ".\n";
             cout << "Loaded player charisma is: " << charisma <<
".\n";
            cout << "Loaded player dexterity is: " << dexterity <<
".\n";
            DisplayInventory();
        } //close load character function

        // DisplayCharacter
        void DisplayCharacter() {
            cout << "Player name is: " << name << ".\n";
            cout << "Player strength is: " << strength << ".\n";
             cout << "Player level is: " << level << ".\n";
            cout << "Player intelligence is: " << intelligence <<
".\n";
             cout << "Player charisma is: " << charisma << ".\n";
            cout << "Player dexterity is: " << dexterity << ".\n";
            DisplayInventory();
        }

        // Display Menu
        void DisplayMenu() {
            bool quit = false;
            char choice;
            cout << "\nChoose an option:\n";
            cout << "***********MENU**********\n";
            cout << "*                       *\n";
            cout << "* L = Load player       *\n";
            cout << "* S = Save player       *\n";
            cout << "* D = Display player    *\n";
```

```
            cout << "* C = Change attributes  *\n";
            cout << "* I = Inventory           *\n";
            cout << "* A = All Items Cheat     *\n";
            cout << "* Q = Quit                *\n";
            cout << "*                         *\n";
            cout << "***************************\n";
            while(!quit) {
                  DisplayMenu();
                  cin >> choice;

                  switch(choice) {
                       case 'l' : LoadCharacter(); break;
                       case 'L' : LoadCharacter(); break;
                      case 'd' : DisplayCharacter(); break;
                      case 'D' : DisplayCharacter(); break;
                      case 's' : SaveCharacter(); break;
                      case 'S' : SaveCharacter(); break;
                      case 'c' : SetPlayerAttributesCheat(this);
break;
                      case 'C' : SetPlayerAttributesCheat(this);
break;
                       case 'i' : DisplayInventory(); break;
                      case 'I' : DisplayInventory(); break;
                      case 'a' : AllItems(); break;
                      case 'A' : AllItems(); break;
                      case 'q' : quit = true; break;
                      case 'Q' : quit = true; break;
                       default : cout << "Please choose either c, l,
s, or d."
                                      << " (LSD - get it?) haha";
                  } //close switch statement

            } // close while true
        } //close function

    // Data Members
    private:
    int level;
    int strength;
    int intelligence;
    int charisma;
    int dexterity;
    char * name;
    //Items
    bool sword;
    bool longbow;
    bool staff;
    bool shield;
    bool armor;

};  // close character class specification
```

V. Game Main() Function

```
//File 5 of 5 -  main() function C. Germany 2002 - in file
"AdventureGame7.cpp"
// AdventureGame7.cpp - 2002 C. Germany: Using srand(), rand(), class
structure, while true,
// for, switch, functions, nested if/else strictures, implementing
variables and objects of
// global and local scope, overriding default constructor and
destructor, passing pointers/refs.

#include "GameClasses.h"
#include "GameFunctions.h"
#include "GameCharacterClass.h"
void main() {

     //Globals
     int score = 0;
     int ALIVE = 100;
     char Name[25];
     char conflict = 'z';
     //Old C printf
     printf("%s", "\aAdventureGame7 - C. Germany, 2002\nSample project
for Intro to C++ class.\n");

     //Start game and initialize objects
     Player * CurrentPlayer = new Player();
     char Title[] = "\nWhat is your name, player?";
     int length = strlen(Title);
     cout.write(Title, length);
     cin.getline(Name, 26);
     cout << "Welcome to AdventureGame7, " << Name << ".\n\n"; //global
Name
     cout <<
"\n*****************************************************************\n";
     DisplayHighScores();
     cout <<
"\n*****************************************************************\n";
     cout << "\n";

     cout.width(76);
     cout.fill('*');
     cout << "*\n";
     cout << "*
*\n";
     cout << "*  Choose your difficulty level: 1=Beginner
2=Intermediate  3=VERY Hard  *\n";
     cout << "*
*\n";
     cout.width(76);
     cout.fill('*');
```

```
     cout << "*\n";
     cout << "\n\n";
     ChooseDifficultyLevel(ALIVE);
     cout.flush();
     system("CLS");
     cout <<"\n";
     cout << "You have " << ALIVE << " points allocated to you as you
start the game.\n";

     //Things we only want to happen once, at the beginning
     ClassicMovie();
     SetTheScene(Name);



     //Things we want to happen until player dies or wins the game
     while(ALIVE > 0 && score < 5000) {

     DragonEncounter(ALIVE, conflict);
     ArrayOfMonsters(ALIVE, score);
     RandomMonsterAttackSequence(ALIVE, score, conflict);
     }
     if(ALIVE <= 0) {
        if(ALIVE < 0) {
           cout << "\n\nYou just experienced POST-MORTEM humiliation,
my friend!\n";
           cout <<"\n" << Name << ", your final score is: " << score <<
".\n\n";
           cout << "You fought a total of " << Monster::CountMonsters()
<< " monsters this game!\n";
           cout.flush();
           system("PAUSE");
           system("CLS"); }
        else if(ALIVE = 0) {
                cout << "\n\nSorry, but it appears you have died!\n";
                cout <<"\n" << Name << ", your final score is: " <<
score << ".\n\n";
                cout << "You fought a total of " <<
Monster::CountMonsters() << " monsters this game!\n";
                cout.flush();
                system("PAUSE");
                system("CLS");
       }



     cout << "\n\n";
     cout << Name << ", you fought bravely but died.  As the worms
devour your flesh\n"
          << "and your soul descends into Hades you vow that you will
one day\n"
          << "arise to take vengeance on your foes.  Dare you try your
luck\n"
          << "and suffer defeat once more?  \n\nAt least for now, the
game is over....\n\n";
```

```
     cout << "\n\n\nLimping back to your operating system...\n\n\n";
     cout.flush();
     system("PAUSE");

 } //close if
      if(score >=5000){
         cout << "\n\nYou did it!  You win!  you are victorious!\n";
          cout <<"\n" << Name << ", your final score is: " << score <<
".\n\n";
          cout << "You fought a total of " << Monster::CountMonsters()
<< " monsters this game!\n";
          cout.flush();
          system("PAUSE");
          system("CLS");
          }
      SaveHighScores(Name, score);
      cout <<
"\n*****************************************************************\n";
      DisplayHighScores();
      cout <<
"\n*****************************************************************\n";
} //close main function
```


C++ Console 32 Project 1: **Mad Lib Game** - Parallel Arrays Binary Executable: MadLib.exe

Objective: To become familiar with Parallel arrays and the relationships they employ when combined with repetition structures in C++.

```
// File 1 of 1 - Mad Lib Project - C++ 2 - ©2004 Germany - Objective -
Learn structure and
// usage of PARALLEL ARRAYS and reading from and writing to sequential
files.
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
//-----------------------------------------------------------------------------------------
//Function prototypes
int SaveTheStory(string * StuffToWrite);
int LoadTheStory(string * StuffToRead);
void MakeTheStory(string * TheStory);
void MainMenu(char & choice, string * Story);
//-----------------------------------------------------------------------------------------
void main() {
     //The Globals
     char KeepGoing = 'y';
     string Story;
```

```
      //Launch the menu
      while(KeepGoing != 'n')
      {
            MainMenu(KeepGoing, &Story);
      }
      cout << "\nClosing the program - exiting main().\n\n";
} //close main() function

//-------------------------------------------------------------------------------------
void MainMenu(char & choice, string * AStory)
{
      cout << endl;
      cout <<
"\t*******************************************************\n";
      cout << "\t*    MAD LIB 1.0  -  Main Menu  - 2004 C. Germany
*\n";
      cout <<
"\t*******************************************************\n";
      cout << "\t*
*\n";
      cout << "\t*     (Q)uit the Game
*\n";
      cout << "\t*     (P)lay Mad Lib
*\n";
      cout << "\t*     (S)ave your Mad Lib
*\n";
      cout << "\t*     (V)iew a Mad Lib previously created
*\n";
      cout << "\t*
*\n";
      cout <<
"\t*******************************************************\n";
      cout << endl;

      cout << "\tWhat would you like to do? ";
      cin >> choice;

      switch(tolower(choice))
      {
            case 'q': cout << "\nExiting menu system ...";
                      choice = 'n';
                      break;
             case 'p': cout << "Starting the game!";
                      MakeTheStory(AStory);
                      break;
            case 's': cout << "Saving the Mad Lib.";
                      SaveTheStory(AStory);
                      break;
            case 'v': cout << "Loading the Mad Lib.";
                      LoadTheStory(AStory);
                      break;
             default: cout << "Invalid input.";
                     break;

        } //close switch
```

```
} //close MainMenu() function
//-----------------------------------------------------------------------------------------
void MakeTheStory(string * TheStory)
{
     int x = 0;
     const int NumWords = 7;
     string UserSays[NumWords];
     string ComputerSays[NumWords] = {

                                "One day, ",
                                " was tip-toing through the \ntulips when
she saw a ",
                                ".  This was because the folks \nat
Microsoft were ",
                                " into the sky again and \nagain. The ",
                                " jumped off of a cliff \nbecause the ",
                                " couldn\'t pat its head and rub its
\ntummy at the same time. The ",
                                " ran \noff holding hands with "
                                };
     string Hints[NumWords] = {
                                   "A female person's name: ",
                                  "An object: ",
                                  "An infinitive verb: ",
                                 "An object or animal: ",
                                  "An object: ",
                                  "An object: ",
                                  "A male person's name: "
                                 };

     //Clear the screen
     system("CLS");
     cout.flush();
     //Get the user to input some appropriate words
     for(x = 0; x < NumWords; x++)
     {
         cout << Hints[x];
        cin >> UserSays[x];
        cout << endl;
     }
     //Concatenate the arrays of strings with an accumulator
     for(x = 0; x < NumWords; x++)
     {
         *TheStory = *TheStory + ComputerSays[x] + UserSays[x];
     }

     //Now end the story with a period!
     *TheStory = *TheStory + ".\n\n";

     //Display it to the screen
     cout << endl << endl << *TheStory << endl << endl;
} //close PlayTheGame() function
//-----------------------------------------------------------------------------------------
int SaveTheStory(string * StuffToWrite)
{
    //Write the story to a file
```

```
    string FileName = "";
    string Extension = ".txt";
    system("CLS");
    cout.flush();
    cout << endl << "Enter name of file to save: ";
    cin >> FileName;
    FileName = FileName + Extension;
    const char * TheFile = FileName.c_str();
    ofstream OutputFile;
    OutputFile.open(TheFile, ios::out);
    if(!OutputFile)
    {
       cout << "Unable to open the file for writing.\n";
       return(1);
    }
    OutputFile << *StuffToWrite;
    OutputFile.close();
    return 0;
} //close SaveTheStory() function
//-------------------------------------------------------------------------------------------
int LoadTheStory(string * StuffToRead)
{
    char OneLetterAtATime;
    string FileName = "";
    string Extension = ".txt";
    system("CLS");
    cout.flush();
    cout << endl << "Enter name of file to load: ";
    cin >> FileName;
    FileName = FileName + Extension;
    const char * TheFile = FileName.c_str();
    ifstream InputFile;
    InputFile.open(TheFile, ios::in);

    if(!InputFile)
    {
        cout << "Unable to open the file for reading.\n";
        return(1);
    }

    cout << "\n*************Here's the contents of the
file:*************\n\n";
    while(InputFile.get(OneLetterAtATime))
    {
          cout << OneLetterAtATime;
    }

    cout << "\n*************End of file contents.*************\n";
    InputFile.close();
    return 0;
} //close ReadFile() function
//-------------------------------------------------------------------------------------------
```

C++ Console 32 Project 1: **Command Line Monster 1.0** Binary Executable: MonsterArg.exe

Objective: To demonstrate an understanding of polymorphism, inheritance, pointers, references and local and global scope. In addition, you will write basic linear search routines and include the ability to save your data to a binary file. You will create a function to load this data and display it. To complete this project, you must demonstrate an understanding of passing objects by pointer, reference and value.

You must be familiar with creating objects locally on the stack and on the free store (heap). You should demonstrate cleaning up memory leaks by calling delete on pointers to objects on the heap when they are no longer needed, and you should initialize those pointers to 0 to avoid calling delete on heap objects twice. You must also use repetition and decision structures where needed.

Files In Project: 1 - Monster.cpp

```
// File 1 of 1 - A Monster Class With Command Line Arguments - ©2002 C. Germany
//-------------------------------------------------------------------------------------------------
--------------------------------

#include <iostream.h>

/*-------------------------------------------------------------------------------------------------
----------------------------*/
class Monster {
public:
        Monster() { cout << "A monster!\n"; };
         Monster(int ag) { cout << "A monster!\n";
                           age = ag; }
        Monster(char * nam) { cout << "A monster!\n";
                              name = nam; }
        ~Monster() { cout << "Monster destroyed.\n"; }
         //Accessor Methods
        void setAge(int ag) { age = ag; }
        int getAge() { return age; }
        void setName(char * nam) { name = nam; }
        char * getName() { return name; }
private:
       int age;
       char * name;
}; //close Monster class specification

/*-------------------------------------------------------------------------------------------------
----------------------------*/

//Note: int argc is number of command-line arguments and char argv is a char string for the
arguments
void main(int argc, char ** argv) {
     //offset for program name, since that is accepted as the first argument
     int size = argc - 1;
     Monster * MonsterArray = new Monster[size];
     for(int x = 0; x < size; x++)
     {
         MonsterArray[x].setName(*(argv+1));
         argv++;

     } //close for loop
```

```
     for(x = 0; x < size; x++)
     {
          cout << "Monster number " << x+1 << " is named "
               << MonsterArray[x].getName() << ".\n"
               << "After argument number " << x+1 << " that you typed
in.\n\n";

     } //close for loop
     delete [] MonsterArray;
} // close main()
```


C++ Console 32 Project 1: **Binary Write** - Binary File Access Binary Executable: ToyClass.exe

Objective: To become familiar with Binary File Access and the relationships employed when combined with repetition structures in C++.

Note: You will need to provide an ASCII text file called "addresses.dat" for this program to work. A sample one is included below.

File 1 - Class.h

```
//File 1 of 3   © C. Germany 2004 -   "Class.h" -    The Class Header
File
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
using namespace std;
class Toy
{
      public:
          Toy() { cout << "Toy object created. \n\n"; }
           ~Toy() { cout << "Toy object destroyed. \n\n";  }
          void setToyAmount(int amt) { ToyAmount = amt; }
          int getToyAmount() { return ToyAmount; }
          void setToyPrice(float price) { ToyPrice = price; }
          float getToyPrice() { return ToyPrice; }
          void setToyName(char tname[25]) { strcpy(ToyName, tname); }
          char * getToyName() { return ToyName; }
      private:
          char ToyName[25];
          int ToyAmount;
          float ToyPrice;
};
```

File 2 - Functions.h:

```
//File 2 of 3   -   "Functions.h" -    Read and Write Functions

//In both instances, we need to pass in the address of the object,
dereference it,
//determine the size of its data members and cast to a pointer to a
character.

//Note: I wrote both these functions to be simple and concise and to
illustrate the 4 steps of
//reading and writing - 1)Create an instance, 2)Open a file, 3)Write the
data, 4)Close the file.

//-------------------------------------------------------------------------------------------
----------------------------

void WriteIt(Toy * TheToy)
{
     ofstream WriteObject;
     WriteObject.open("TheFile.dat", ios::binary);
     WriteObject.write((char*) &*TheToy, sizeof *TheToy);
     WriteObject.close();
}
//-------------------------------------------------------------------------------------------
----------------------------

void ReadIt(Toy * TheToy)
{
     ifstream ReadObject;
     ReadObject.open("TheFile.dat", ios::binary);
     ReadObject.read((char*) &*TheToy, sizeof *TheToy);
     ReadObject.close();
}

//-------------------------------------------------------------------------------------------
----------------------------
```

File 3 - The Main() Function

```
//File 3 of 3   -   "BinaryWrite.cpp" -    Create instance of class on
heap and write/read it
#include "class.h"
#include "functions.h"
void main()
{
     //Create instance
     Toy * BeanieBaby = new Toy;
     //Set object values
     BeanieBaby->setToyName("Cute little bean bag toy.");
     BeanieBaby->setToyAmount(200);
     BeanieBaby->setToyPrice(25.32);
     //Display object values before read/write
     cout << "Toy description: " << BeanieBaby->getToyName() << endl;
```

```
     cout << "Toy amount (inventory): " << BeanieBaby->getToyAmount() <<
endl;
     cout << "Toy price (per unit): " << BeanieBaby->getToyPrice() <<
endl;
     WriteIt(BeanieBaby);
     //KILL the BeanieBaby!!!
     delete BeanieBaby;
     BeanieBaby = 0;
     //Create a new toy on the heap
     BeanieBaby = new Toy;

     //Show that the data members of the Toy we have created have not
yet been initialized
     cout << "Toy before binary read:" << endl;
     cout << "Toy description: " << BeanieBaby->getToyName() << endl;
     cout << "Toy amount (inventory): " << BeanieBaby->getToyAmount() <<
endl;
     cout << "Toy price (per unit): " << BeanieBaby->getToyPrice() <<
endl << endl;
     //Load our saved values into the Toy object we just created on the
heap
     ReadIt(BeanieBaby);
     //Show that the Toy object now had values
     cout << "Toy AFTER binary read:" << endl;
     cout << "Toy description: " << BeanieBaby->getToyName() << endl;
     cout << "Toy amount (inventory): " << BeanieBaby->getToyAmount() <<
endl;
     cout << "Toy price (per unit): " << BeanieBaby->getToyPrice() <<
endl << endl;
     delete BeanieBaby;
     BeanieBaby = 0;
} //close main() function
```


inary Executable: **[** adventuregame7.exe **]** (Console)
Binary Executable: **[** hangman.exe maybe is LameMan.exe **]** (Console)
Binary Executable: **[** MFCmenu1.exe **]** (MFC Version with Menu Example)

Simplified, Previous Stages for Incremental Lessons:

```
Class Projects/Quizes - Without the Solutions Displayed

//Adventure Game Header File - Functions and Classes
#include "stdafx.h"
#include <iostream.h>
#include <stdlib.h>
#include <time.h>
//Game Functions
```

```
int GenerateRandomNumber(int Number=6) {
int ResultRandom;
srand(time(NULL));
ResultRandom = (rand()%Number) + 1;
return ResultRandom;
}
void Attack(int *PlayerPoints) {
int Damage;
Damage = GenerateRandomNumber(10);
*PlayerPoints = *PlayerPoints - Damage;
cout << "Ouch!  You loose " << Damage << " points.\n";
cout << "You now have " << *PlayerPoints << " points left!";
}
void SetTheScene(char PName[25]) {
// Takes char array (or "string") as an argument.
// Example of using a global value locally inside a function
int Setting;
Setting = GenerateRandomNumber();
switch (Setting) {
case 1 : cout << PName << ", you are in a swamp and sinking!\n";
             break;
case 2 : cout << PName << ", you are on a high mountain peak.\n";
             break;
case 3 : cout << PName << ", you are on a grassy plane.\n";
             break;
case 4 : cout << PName << ", you are in the desert.\n";
             break;
case 5 : cout << PName << ", you find yourself in a deserted
village.\n";
             break;
case 6 : cout << PName << ", you find yourself in a steamy
jungle.\n";
             break;
default : cout << "Something is definitely wrong here.  Should be
1-6.";
} // close switch statement
} // close SetTheScene() function
//CLASSES
class Monster {
public:
//BAse CLASS - 3 Overloaded Constructors
        Monster(int MonStrength = 10) {
                Strength = MonStrength;
                cout << "\nA monster has been created.";}
        Monster(int MonStrength, int MonWeight) {
                Strength = MonStrength;
                Weight = MonWeight;
                cout << "\nA monster has been created.";}
        Monster(int MonStrength, int MonWeight, char MonName[10]) {
                Strength = MonStrength;
                Weight = MonWeight;
                Name[10] = MonName[10];
                cout << "\nA monster has been created.";}
        ~Monster() { cout << "\nThe monster has been destroyed!\n";}
//Accessor Methods
```

```
        int getStrength() { return Strength; }
        void setStrength(int MonsStrength) { Strength =
MonsStrength; }
        int getWeight() {return Weight;}
        void setWeight(int Wt) {Weight = Wt;}
    int getName() {return Name[10];}
        void setName(char Nm[10]) {Name[10] = Nm[10];}

        void Talk() {
        int SayWhat;
        SayWhat = GenerateRandomNumber(4);
        switch(SayWhat) {
    case 1 : cout << "I like flowers.  Will you be my friend?";
                    break;
        case 2 : cout << "I would really like to dismember you and
eat you...";
                    break;
        case 3 : cout << "Unfortunately, I have had a very bad day
and"
                                << " you happen to be the first
creature I\'ve met"
                                << " that I can take it out on...";
                    break;
        case 4 : cout << "How do you taste?  I don\'t think I\'ve
eaten"
                                << " your kind before...";
                    break;
        default : cout << "Uh oh, this should never happen...";
        } //closes switch
        } //close talk function
protected:
        int Strength;
        int Weight;
        char Name[10];
}; //closes class specification

class Giant : public Monster {
//Derived class of Monster
public:
        Giant() {cout << "A giant is afoot...";}
        ~Giant() {cout << "Bye bye giant...";}
//Accesor Methods
   int getFootsize() {return Footsize;}
   void setFootSize(int GiantFeet) {Footsize = GiantFeet;}
//Other Functions
   void GiantSpeak() {cout << "Fee Fi Fo Fum...";}
   void Squash(int * Life) {
           cout << "The giant squashes you like a bug!";
       *Life = 0;}

private:
        int Footsize;
}; //close class specification
class Troll : public Monster {
//Derived class of Monster
```

```
public:
        Troll() {cout << "Somewhere a troll is in a bad mood...";}
        ~Troll() {cout << "Bye troll...";}
//Accesor Methods
   int getDrool() {return Drool;}
   void setDrool(int TrollDrool) {Drool = TrollDrool;}
//Other Functions
   void GiantSpeak() {cout << "Ahhhhhhhhhhhhhh...";}
   void Eat(int * Life) {
           cout << "The dismembers you and boils your carcass in a
stew.";
       *Life = 0;}
   void Drooling() {
   setDrool(10);
   cout << "\nThe troll salivates, dropping " << getDrool()
            << " gallons of drool onto your shiny new boots.\n";}
private:
        int Drool;
}; //close class specification
class BarryManilow : public Monster {
//Derived class of Monster
public:
        BarryManilow() {cout << "Run for your life - it\'s Barry!";}
        ~BarryManilow() {cout << "I write the songs that...";}
//Accesor Methods
   int getSong() {return Song[20];}
   void setSong(int BarrySong[20]) {Song[20] = BarrySong[20];}
//Other Functions
   void BarrySing(int * Life) {
           cout << "Barry lifts his voice to sing a ballad...";
           cout << "\nYou feel the sudden urge to pull out your
own\n"
                   << "sword and kill yourself, right then and
there.\n"
                       << "Barry, for lack of an interested
audience, \n"
                       << "continues serenading your rotting
corpse...\n";
       *Life = 0;}

private:
        int Song[20];
}; //close class specification
// main() Function .cpp File (Goes with previous Function/Class
header file)
// Using srand(), rand(), class structure, while true, functions,
// switch statement, and nested if/else strictures, implementing
variables and objects of
// global and local scope, Overriding default constructor and
destructor, passing pointers.
#include "Stdafx.h"
#include "GameClasses.h"

void main() {
int ALIVE = 100;
```

```
int PlayerPoints = 20;  //Global
char conflict;
char conflict2;
char Name[25];
cout << "What is your name, player? ";
cin >> Name;
cout << "Welcome to the game, " << Name << ".\n\n"; //global Name
cout << "You have " << PlayerPoints << " points allocated to you as
you start the game.\n";
SetTheScene(Name);
Monster Ogre;
Monster Dragon(20);
while(ALIVE>0) {
Dragon.Talk();
cout << "\nYou encounter a dragon!\n";
cout << "He blasts you with fire! ";
Attack(&PlayerPoints);
cout << " He is of strength ";
cout << Dragon.getStrength();
cout << ".\n\n";
cout << "\nYou pull out your sword and levy a heavy blow upon his
head.\n";
Dragon.setStrength(5);
cout << "That was some move! The dragon is already half dead!\n";
cout << "He now only has the strength of ";
cout << Dragon.getStrength();
cout << ".\n\n";
cout << "Now what do you choose to do?\n"
     << "r = run away\n"
         << "s = stay and fight\n"
         << "t = talk to the dragon\n";
cin >> conflict;
system("cls");
switch(conflict) {
case 'r' : cout << "You run away and survive!\n"
                   << "The dragon inflicts merely a flesh wound.\n"
                 << "You loose a mere 15 points. ";
               ALIVE = ALIVE - 15;
                   cout << "You now have " << ALIVE << " points
left.";
           break;
case 's' : cout << "You stay and fight.  You discover that\n"
                << "the dragon is merely a baby dragon.  It\'s
mother\n"
                << "then shows up, greatly enraged by your attempt
to\n"
                << "kill her child.  She is really big and really
angry.\n";
           Dragon.setStrength(200);
               cout << "You are now fighting a dragon of strength:
";
                   cout << Dragon.getStrength() << ".\n";
                   cout << "The dragon rips off your arms and legs
and\n"
```

```
                            << "fries you to a crispy, crunchy,
golden brown.\n";
               ALIVE = 0;
                  break;
case 't' : cout << "\nYou strike up a conversation with the
dragon.\n"
                << "You begin telling it your life story and all
your woes.\n"
                        << "Succumbing to fatal levels of nausea and
boredom, the \n"
                        << "dragon lapses into a coma and
dies...\n";
               Dragon.setStrength(0);
               cout << "Dragon reduced to strength: ";
                  cout << Dragon.getStrength() << ".\n";
                  Dragon.~Dragon();
                  cout << "You have " << ALIVE << " points left.";
                  break;
default :  cout << "Invalid input.  Please press either: r, s, or
w.";
} //close switch statement
while(ALIVE>0) {
cout << "\nYou continue traveling eastward.  You see something in
the distance...\n";
Ogre.Talk();
cout << "It is an ogre!\n";
cout << "What do you do now?\n";
cout << "r = run away\n"
     << "t = talk\n";
cin >> conflict2;
system("cls");
  if(conflict2 == 'r') {
     cout << "\nThe ogre chases you.  You fall.  He chops off your
head and eats you.\n";
     ALIVE = 0;
  }
  else if(conflict2 == 't') {
         cout << "\nYou make friends.  The ogre likes you and so
kills you.\n";
         ALIVE = 0;
  }
} //close 2nd while true loop
} //close 1st while true loop - ALIVE no longer > 0
cout << "\nYou fought bravely but died.  As the worms devour your
flesh\n"
     << "and your soul descends into Hades you vow that you will
one day\n"
     << "arise to take vengeance on your foes.  Dare you try your
luck\n"
     << "and suffer defeat once more?  At least for now, the game
is over....\n\n";
} //close main function
```

```
// AdventureGame3.cpp : Using srand(), rand(), class structure,
while true, functions,
// switch statement, and nested if/else strictures, implementing
variables and objects of
// global and local scope, Overriding default constructor and
destructor, passing pointers.
#include "stdafx.h"
#include <iostream.h>
#include <stdlib.h>
#include <time.h>
class Monster {
public:
        Monster(int MonStrength = 10) {
                Strength = MonStrength;
                cout << "\nA monster has been created.";}
        ~Monster() { cout << "\nThe monster has been destroyed!\n";}
        int getStrength() { return Strength; }
        void setStrength(int MonsStrength) { Strength =
MonsStrength; }
        void Talk() { cout << "\nGrrr!! I am a monster... "; }
private:
        int Strength;
}; //closes class specification
int GenerateRandomNumber(int Number=6) {
int ResultRandom;
srand(time(NULL));
ResultRandom = (rand()%Number) + 1;
/*
//For debugging purposes only
cout << "\nToday\'s lucky number is: "
     << ResultRandom << ".\n\n";
*/
return ResultRandom;
}
void Attack(int *PlayerPoints) {
int Damage;
Damage = GenerateRandomNumber(10);
*PlayerPoints = *PlayerPoints - Damage;
cout << "Ouch!  You loose " << Damage << " points.\n";
cout << "You now have " << *PlayerPoints << " points left!";
} //close attack() function
void SetTheScene(char PName[25]) {
// Takes char array (or "string") as an argument.
// Example of using a global value locally inside a function
int Setting;
Setting = GenerateRandomNumber();
switch (Setting) {
case 1 : cout << PName << ", you are in a swamp and sinking!\n";
             break;
case 2 : cout << PName << ", you are on a high mountain peak.\n";
             break;
case 3 : cout << PName << ", you are on a grassy plane.\n";
             break;
case 4 : cout << PName << ", you are in the desert.\n";
             break;
```

```
case 5 : cout << PName << ", you find yourself in a deserted
village.\n";
            break;
case 6 : cout << PName << ", you find yourself in a steamy
jungle.\n";
            break;
default : cout << "Something is definitely wrong here.  Should be
1-6.";
} // close switch statement
} // close SetTheScene() function

void main() {
int ALIVE = 100;
int PlayerPoints = 20;  //Global
char conflict;
char conflict2;
char Name[25];
cout << "What is your name, player? ";
cin >> Name;
cout << "Welcome to the game, " << Name << ".\n\n"; //global Name
cout << "You have " << PlayerPoints << " points allocated to you as
you start the game.\n";
SetTheScene(Name);
Monster Ogre;
Monster Dragon(20);
while(ALIVE>0) {
Dragon.Talk();
cout << "\nYou encounter a dragon!\n";
cout << "He blasts you with fire! ";
Attack(&PlayerPoints);
cout << " He is of strength ";
cout << Dragon.getStrength();
cout << ".\n\n";
cout << "\nYou pull out your sword and levy a heavy blow upon his
head.\n";
Dragon.setStrength(5);
cout << "That was some move! The dragon is already half dead!\n";
cout << "He now only has the strength of ";
cout << Dragon.getStrength();
cout << ".\n\n";
cout << "Now what do you choose to do?\n"
     << "r = run away\n"
         << "s = stay and fight\n"
         << "t = talk to the dragon\n";
cin >> conflict;
system("cls");
switch(conflict) {
case 'r' : cout << "You run away and survive!\n"
                    << "The dragon inflicts merely a flesh wound.\n"
                << "You loose a mere 15 points. ";
               ALIVE = ALIVE - 15;
                   cout << "You now have " << ALIVE << " points
left.";
           break;
case 's' : cout << "You stay and fight.  You discover that\n"
```

```
                << "the dragon is merely a baby dragon.  It\'s
mother\n"
                << "then shows up, greatly enraged by your attempt
to\n"
                << "kill her child.  She is really big and really
angry.\n";
           Dragon.setStrength(200);
               cout << "You are now fighting a dragon of strength:
";
                  cout << Dragon.getStrength() << ".\n";
                  cout << "The dragon rips off your arms and legs
and\n"
                           << "fries you to a crispy, crunchy,
golden brown.\n";
               ALIVE = 0;
                  break;
case 't' : cout << "\nYou strike up a conversation with the
dragon.\n"
                << "You begin telling it your life story and all
your woes.\n"
                        << "Succumbing to fatal levels of nausea and
boredom, the \n"
                        << "dragon lapses into a coma and
dies...\n";
               Dragon.setStrength(0);
               cout << "Dragon reduced to strength: ";
                  cout << Dragon.getStrength() << ".\n";
                  Dragon.~Dragon();
                  cout << "You have " << ALIVE << " points left.";
                  break;
default :  cout << "Invalid input.  Please press either: r, s, or
w.";
} //close switch statement
while(ALIVE>0) {
cout << "\nYou continue traveling eastward.  You see something in
the distance...\n";
Ogre.Talk();
cout << "It is an ogre!\n";
cout << "What do you do now?\n";
cout << "r = run away\n"
     << "t = talk\n";
cin >> conflict2;
system("cls");
  if(conflict2 == 'r') {
     cout << "\nThe ogre chases you.  You fall.  He chops off your
head and eats you.\n";
     ALIVE = 0;
  }
  else if(conflict2 == 't') {
   cout << "\nYou make friends.  The ogre likes you and so kills
you.\n";
   ALIVE = 0;
  }
} //close 2nd while true loop
} //close 1st while true loop - ALIVE no longer > 0
```

```
cout << "\nYou fought bravely but died.  As the worms devour your
flesh\n"
     << "and your soul descends into Hades you vow that you will
one day\n"
     << "arise to take vengeance on your foes.  Dare you try your
luck\n"
     << "and suffer defeat once more?  At least for now, the game
is over....\n\n";
} //close main function



// Project 3 - AdventureGame1.cpp : Adventure game using functions,
// a while true loop and a switch statement.
#include "stdafx.h"
#include <iostream.h>
void Talk() {
char name;
cout << "Very hospitable of you.  You greet the old man. "
     << "He turns towards you and asks your name.  Your reply?";
cin >> name;
}
int Attack() {
cout << "You attack.  The old man turns into an ogre.\n"
     << "He pulls out an axe and chops off your head.\n"
         << "You are now dead, dead, dead...\n";
     return 0;
}
void Give() {
cout << "You give him a sandwhich";
}
void Ignore() {
cout << "You decide to ignore the man";
}
void main()
{
int ALIVE = 1;
char choice1;
while (ALIVE != 0) {
cout << "\nIt\'s twilight, and you can just make out the "
     << "constellatoin Orion as you look North.  You are "
         << "walking on towards a mountain range, ascending "
         << "towards the west.  You see an elderly man in "
         << "tattered clothes approaching.  What do you do?\n";
cout << "Your options are as follows:\n\n"
     << "T - Talk with the old man.\n"
         << "H - Hit the old man in the head with a shovel.\n"
         << "G - Give the old man a sandwhich.\n"
         << "I - Ignore the old man.\n";
cout << "\nYou choose: ";
cin >> choice1;
switch (choice1) {
case 'T' : Talk();
               break;
case 't' : Talk();
```

```
                break;
case 'H' : ALIVE = Attack();
                break;
case 'h' : ALIVE = Attack();
                break;
case 'G' : Give();
                break;
case 'g' : Give();
                break;
case 'I' : Ignore();
                break;
case 'i' : Ignore();
                break;
default : cout << "\nSorry, that is not a valid choice.\n\n";
} //closes switch statement
cout << "\nIf only, if only, if only...\n\n";
} //close while true loop
cout << "\n\nGame ending...\n\n";
} //closes main() function
```

// AdventureGame2.cpp Now using class structure, functions, random number generation, and
// implementing variables and objects of global and local scope.
// Overriding default constructor and destructor

```
#include "stdafx.h"
#include <iostream.h>
#include <stdlib.h>
#include <time.h>
class Monster {
public:
        Monster(int MonStrength = 10) {
                Strength = MonStrength;
                cout << "\nA monster has been created.";}
        ~Monster() { cout << "\nThe monster has been destroyed!\n";}
        int getStrength() { return Strength; }
        void setStrength(int MonsStrength) { Strength =
MonsStrength; }
        void Talk() { cout << "\nGrrr!! I am a monster... "; }
private:
        int Strength;
}; //closes class specification
int GenerateRandomNumber() {
int ResultRandom;
srand(time(NULL));
ResultRandom = (rand()%6) + 1;
cout << "\nToday\'s lucky number is: "
     << ResultRandom << ".\n\n";
return ResultRandom;
}
void SetTheScene() {
int Setting;
Setting = GenerateRandomNumber();
```

```
switch (Setting) {
case 1 : cout << "You are in a swamp.\n";
            break;
case 2 : cout << "You are on a high mountain peak.\n";
            break;
case 3 : cout << "You are on a grassy plane.\n";
            break;
case 4 : cout << "You are in the desert.\n";
            break;
case 5 : cout << "You find yourself in a deserted village.\n";
            break;
case 6 : cout << "You find yourself in a steamy jungle.\n";
            break;
default : cout << "Something is definitely wrong here.  Should be
1-6.";
} // close switch statement
} // close SetTheScene() function
void main() {
int ALIVE = 100;
char conflict;
char conflict2;
SetTheScene();
Monster Ogre;
Monster Dragon(20);
while(ALIVE>0) {
Dragon.Talk();
cout << "\nYou encounter a dragon!\n"
     << "He is of strength ";
cout << Dragon.getStrength();
cout << ".\n\n";
cout << "\nYou pull out your sword and levy a heavy blow upon his
head.\n";
Dragon.setStrength(5);
cout << "That was some move! The dragon is already half dead!\n";
cout << "He now only has the strength of ";
cout << Dragon.getStrength();
cout << ".\n\n";
cout << "Now what do you choose to do?\n"
     << "r = run away\n"
         << "s = stay and fight\n"
         << "t = talk to the dragon\n";
cin >> conflict;
system("cls");
switch(conflict) {
case 'r' : cout << "You run away and survive!\n"
                    << "The dragon inflicts merely a flesh wound.\n"
                << "You loose a mere 15 points. ";
               ALIVE = ALIVE - 15;
                   cout << "You now have " << ALIVE << " points
left.";
           break;
case 's' : cout << "You stay and fight.  You discover that\n"
                << "the dragon is merely a baby dragon.  It\'s
mother\n"
```

```
               << "then shows up, greatly enraged by your attempt
to\n"
               << "kill her child.  She is really big and really
angry.\n";
           Dragon.setStrength(200);
               cout << "You are now fighting a dragon of strength:
";
                  cout << Dragon.getStrength() << ".\n";
                  cout << "The dragon rips off your arms and legs
and\n"
                           << "fries you to a crispy, crunchy,
golden brown.\n";
               ALIVE = 0;
                  break;
case 't' : cout << "\nYou strike up a conversation with the
dragon.\n"
                << "You begin telling it your life story and all
your woes.\n"
                        << "Succumbing to fatal levels of nausea and
boredom, the \n"
                        << "dragon lapses into a coma and
dies...\n";
               Dragon.setStrength(0);
               cout << "Dragon reduced to strength: ";
                  cout << Dragon.getStrength() << ".\n";
                  Dragon.~Dragon();
                  cout << "You have " << ALIVE << " points left.";
                  break;
default :  cout << "Invalid input.  Please press either: r, s, or
w.";
} //close switch statement
while(ALIVE>0) {
cout << "\nYou continue traveling eastward.  You see something in
the distance...\n";
Ogre.Talk();
cout << "It is an ogre!\n";
cout << "What do you do now?\n";
cout << "r = run away\n"
     << "t = talk\n";
cin >> conflict2;
system("cls");
  if(conflict2 == 'r') {
     cout << "\nThe ogre chases you.  You fall.  He chops off your
head and eats you.\n";
     ALIVE = 0;
  }
  else if(conflict2 == 't') {
   cout << "\nYou make friends.  The ogre likes you and so kills
you.\n";
   ALIVE = 0;
  }
} //close 2nd while true loop
} //close 1st while true loop - ALIVE no longer > 0
cout << "\nYou fought bravely but died.  As the worms devour your
flesh\n"
```

```
     << "and your soul descends into Hades you vow that you will
one day\n"
     << "arise to take vengeance on your foes.  Dare you try your
luck\n"
     << "and suffer defeat once more?  At least for now, the game
is over....\n\n";
} //close main function
```

Windows API Project – Calculator

C++ Win 32 Project: Windows API Project - Calculator

Like the MFC, Windows API programming provides a graphical user interface. Its structure different from the MFC with its class hierarchy and polymorphic inheritance. Windows API programming opens a common gateway to programming with DirectX. The example below is a calculator using the Windows API.

File 1: Calculator.cpp

```
#include <windows.h>
#include <cstring>

using namespace std;

void InputNumber(char CurrentNumber[10]);
void Calculate();
char Operation = '0';
char Num1[10];
double Number1 = 0;
char Num2[10];
double Number2 = 0;

//-------------------------------------------------------------------
-------------

LRESULT CALLBACK MainWndProc(HWND, UINT, WPARAM, LPARAM);

//-------------------------------------------------------------------
-------------

HWND hWnd; HWND EditBox; HWND Button_Add;
HWND Button_Subtract; HWND Button_Multiply;
HWND Button_Divide; HWND Button_9;
HWND Button_8; HWND Button_7;
HWND Button_6; HWND Button_5;
HWND Button_4; HWND Button_3;
HWND Button_2; HWND Button_1;
HWND Button_0; HWND Button_Decimal;
HWND Button_Equals; HWND Button_Clear;

//-------------------------------------------------------------------
-------------
```

```
int WINAPI WinMain(HINSTANCE hInstance,
                                   HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine,
                                   int nCmdShow)
{
        WNDCLASS wc;
        wc.lpszClassName = "CalculatorClass";
        wc.lpfnWndProc = MainWndProc;
        wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
        wc.hInstance = hInstance;
        wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
        wc.hCursor = LoadCursor( NULL, IDC_ARROW );
        wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );
        wc.lpszMenuName = "";
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        RegisterClass(&wc);
        hWnd = CreateWindow(
                "CalculatorClass",
                "Calculator",
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                185,
                265,
                NULL,
                NULL,
                hInstance,
                NULL);
        EditBox = CreateWindow(
                "EDIT",
                NULL,
                WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
                10,
                10,
                155,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);
        Button_Add = CreateWindow(
                "BUTTON",
                "+",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                10,
                40,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        Button_Subtract = CreateWindow(
```

```
            "BUTTON",
            "-",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            50,
            40,
            35,
            35,
            hWnd,
            NULL,
            hInstance,
            NULL);
        Button_Multiply = CreateWindow(
            "BUTTON",
            "*",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            90,
            40,
            35,
            35,
            hWnd,
            NULL,
            hInstance,
            NULL);
        Button_Divide = CreateWindow(
            "BUTTON",
            "/",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            130,
            40,
            35,
            35,
            hWnd,
            NULL,
            hInstance,
            NULL);
        Button_6 = CreateWindow(
            "BUTTON",
            "6",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            10,
            80,
            35,
            35,
            hWnd,
            NULL,
            hInstance,
            NULL);
        Button_7 = CreateWindow(
            "BUTTON",
            "7",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            50,
            80,
            35,
            35,
```

```
            hWnd,
            NULL,
            hInstance,
            NULL);
        Button_8 = CreateWindow(
            "BUTTON",
            "8",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            90,
            80,
            35,
            35,
            hWnd,
            NULL,
            hInstance,
            NULL);
        Button_9 = CreateWindow(
            "BUTTON",
            "9",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            130,
            80,
            35,
            35,
            hWnd,
            NULL,
            hInstance,
            NULL);
        Button_2 = CreateWindow(
            "BUTTON",
            "2",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            10,
            120,
            35,
            35,
            hWnd,
            NULL,
            hInstance,
            NULL);
        Button_3 = CreateWindow(
            "BUTTON",
            "3",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            50,
            120,
            35,
            35,
            hWnd,
            NULL,
            hInstance,
            NULL);
        Button_4 = CreateWindow(
            "BUTTON",
            "4",
```

```
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                90,
                120,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        Button_5 = CreateWindow(
                "BUTTON",
                "5",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                130,
                120,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        Button_0 = CreateWindow(
                "BUTTON",
                "0",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                10,
                160,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        Button_1 = CreateWindow(
                "BUTTON",
                "1",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                50,
                160,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        Button_Decimal = CreateWindow(
                "BUTTON",
                ".",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                90,
                160,
                35,
                35,
                hWnd,
                NULL,
```

```
                hInstance,
                NULL);
        Button_Equals = CreateWindow(
                "BUTTON",
                "=",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                130,
                160,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        Button_Clear = CreateWindow(
                "BUTTON",
                "Clear",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                10,
                200,
                155,
                25,
                hWnd,
                NULL,
                hInstance,
                NULL);
        ShowWindow(hWnd, nCmdShow);
        MSG msg;
        while(GetMessage(&msg, NULL, 0, 0 ))
        {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }

        return (int) msg.wParam;
}

//--------------------------------------------------------------------------
-------------

LRESULT CALLBACK MainWndProc(HWND hWnd, UINT msg,
        WPARAM wParam, LPARAM lParam)
{
        HWND hwndCtl = (HWND) lParam;
        switch (msg) {
                case WM_COMMAND:
                        switch (wParam) {
                                case BN_CLICKED:
                                        if (hwndCtl == Button_1)
                                                InputNumber("1");
                                        else if (hwndCtl == Button_2)
                                                InputNumber("2");
                                        else if (hwndCtl == Button_3)
                                                InputNumber("3");
                                        else if (hwndCtl == Button_4)
```

```
                                        InputNumber("4");
                                    else if (hwndCtl == Button_5)
                                        InputNumber("5");
                                    else if (hwndCtl == Button_6)
                                        InputNumber("6");
                                    else if (hwndCtl == Button_7)
                                        InputNumber("7");
                                    else if (hwndCtl == Button_8)
                                        InputNumber("8");
                                    else if (hwndCtl == Button_9)
                                        InputNumber("9");
                                    else if (hwndCtl == Button_0)
                                        InputNumber("0");
                                    else if (hwndCtl ==
Button_Decimal)
                                        InputNumber(".");
                                    else if (hwndCtl == Button_Add) {
                                        Operation = '+';
                                        Number1 = atof(Num1);
                                    }
                                    else if (hwndCtl ==
Button_Subtract) {
                                        Operation = '-';
                                        Number1 = atof(Num1);
                                    }
                                    else if (hwndCtl ==
Button_Multiply) {
                                        Operation = '*';
                                        Number1 = atof(Num1);
                                    }
                                    else if (hwndCtl ==
Button_Divide) {
                                        Operation = '/';
                                        Number1 = atof(Num1);
                                    }
                                    else if (hwndCtl ==
Button_Equals) {
                                        Number2 = atof(Num2);
                                        Calculate();
                                    }
                                    else if (hwndCtl == Button_Clear)
{
                                        Operation = '0';
                                        strcpy(Num1, "");
                                        Number1 = 0;
                                        strcpy(Num2, "");
                                        Number2 = 0;
                                        SetWindowText(EditBox,
"");
                                    }
                        }
                        break;
                case WM_DESTROY:
                        PostQuitMessage(0);
                        return 0;
```

```
                default:
                        return DefWindowProc(hWnd, msg, wParam, lParam);
        }
        return 0;
}

//-------------------------------------------------------------------------
-------------

void InputNumber(char CurrentNumber[10])
{
        if (Operation == '0')
        {
                strcat(Num1, CurrentNumber);
                SetWindowText(EditBox, Num1);
        }
        else
        {
                strcat(Num2, CurrentNumber);
                SetWindowText(EditBox, Num2);
        }
}

//-------------------------------------------------------------------------
-------------

void Calculate()
{
        double Result = 0;
        char RES[25];

        switch(Operation)
        {
              case '+' : Result = Number1 + Number2; break;
              case '-' : Result = Number1 - Number2; break;
              case '*' : Result = Number1 * Number2; break;
              case '/' : Result = Number1 / Number2; break;
              default :  break;
         }

        _gcvt(Result, 10, RES);
        SetWindowText(EditBox, RES);
        Operation = '0';
        strcpy(Num1, RES);
        Number1 = 0;
        strcpy(Num2, "");
        Number2 = 0;

}
//-------------------------------------------------------------------------
-------------
```

C++ Win 32 Project: Windows API Project - Background Rectangles

File 1: Background Rectangles.cpp

```
#include <windows.h>
#include <cstring>

using namespace std;

//-------------------------------------------------------------------------
--

LRESULT CALLBACK MainWndProc(HWND, UINT, WPARAM, LPARAM);
HWND hWnd;
HWND hwndCity;
HWND hwndState;
HWND hwndCountry;

//-------------------------------------------------------------------------
--

int WINAPI WinMain(HINSTANCE hInstance,
                                    HINSTANCE hPrevInstance,
                    LPSTR lpCmdLine,
                                    int nCmdShow) {
        WNDCLASS wc;
        wc.lpszClassName = "LocationClass";
        wc.lpfnWndProc = MainWndProc;
        wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
        wc.hInstance = hInstance;
        wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
        wc.hCursor = LoadCursor( NULL, IDC_ARROW );
        wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );
        wc.lpszMenuName = "";
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        RegisterClass(&wc);
        hWnd = CreateWindow(
                "LocationClass",
                "Location",
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                250,
                150,
                NULL,
                NULL,
                hInstance,
                NULL);
        hwndCity = CreateWindow(
                "Static",
                "City: New York",
                WS_VISIBLE | WS_CHILD,
```

```
                10,
                10,
                200,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndState = CreateWindow(
                "Static",
                "State: New York",
                WS_VISIBLE | WS_CHILD,
                10,
                35,
                200,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndCountry = CreateWindow(
                "Static",
                "Country: USA",
                WS_VISIBLE | WS_CHILD,
                10,
                60,
                200,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        ShowWindow(hWnd, nCmdShow);

        MSG msg;
        while(GetMessage(&msg, NULL, 0, 0 )) {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }

        return (int) msg.wParam;
}

//-------------------------------------------------------------------------
--

LRESULT CALLBACK MainWndProc(HWND hWnd, UINT msg,
        WPARAM wParam, LPARAM lParam) {
        HWND hwndCtl = (HWND) lParam;
        switch (msg) {
                case WM_DESTROY:
                        PostQuitMessage(0);
                        return 0;
                default:
```

```
                        return DefWindowProc(hWnd, msg, wParam, lParam);
        }
        return 0;
}

//-------------------------------------------------------------------------
--
```

C++ Win 32 Project: Windows API Project - Window Styles

Different API Window Styles

Style 1

```
#include <windows.h>

LRESULT CALLBACK MainWndProc(HWND, UINT, WPARAM, LPARAM);
HWND hWnd;
HWND hwndLabel;

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpCmdLine, int nCmdShow) {

        WNDCLASS wc;

        wc.lpszClassName = "Style1";
        wc.lpfnWndProc = MainWndProc;
        wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
        wc.hInstance = hInstance;
        wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
        wc.hCursor = LoadCursor( NULL, IDC_ARROW );
        wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );
        wc.lpszMenuName = "";
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        RegisterClass( &wc );

        hWnd = CreateWindow(
                "Style1",
                "Style 1",
                WS_OVERLAPPEDWINDOW | WS_HSCROLL,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                400,
                300,
                NULL,
                NULL,
                hInstance,
```

```
                NULL);

        hwndLabel = CreateWindow(
                "Static",
                "This window uses the WS_OVERLAPPEDWINDOW style and the
WS_HSCROLL style",
                WS_VISIBLE | WS_CHILD,
                10,
                10,
                375,
                250,
                hWnd,
                NULL,
                hInstance,
                NULL);

        ShowWindow(hWnd, nCmdShow);

        MSG msg;
        while(GetMessage(&msg, NULL, 0, 0 )) {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }

        return (int) msg.wParam;
}


LRESULT CALLBACK MainWndProc(HWND hWnd, UINT msg,
        WPARAM wParam, LPARAM lParam) {
        HWND hwndCtl = (HWND) lParam;
        switch (msg) {
                case WM_DESTROY:
                        PostQuitMessage(0);
                        return 0;
                default:
                        return DefWindowProc(hWnd, msg, wParam,
lParam);
        }
        return 0;
}
```

Style 2

```
#include <windows.h>

LRESULT CALLBACK MainWndProc(HWND, UINT, WPARAM, LPARAM);

HWND hWnd;
HWND hwndLabel;
```

```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpCmdLine, int nCmdShow) {

        WNDCLASS wc;

        wc.lpszClassName = "Style2";
        wc.lpfnWndProc = MainWndProc;
        wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
        wc.hInstance = hInstance;
        wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
        wc.hCursor = LoadCursor( NULL, IDC_ARROW );
        wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );
        wc.lpszMenuName = "";
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        RegisterClass( &wc );

        hWnd = CreateWindow(
                "Style2",
                "Style 2",
                WS_SYSMENU | WS_THICKFRAME,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                400,
                300,
                NULL,
                NULL,
                hInstance,
                NULL);

        hwndLabel = CreateWindow(
                "Static",
                "This window uses the WS_SYSMENU style and the
WS_THICKFRAME style",
                WS_VISIBLE | WS_CHILD,
                10,
                10,
                375,
                250,
                hWnd,
                NULL,
                hInstance,
                NULL);

        ShowWindow(hWnd, nCmdShow);

        MSG msg;
        while(GetMessage(&msg, NULL, 0, 0 )) {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }

        return (int) msg.wParam;
}
```

```
LRESULT CALLBACK MainWndProc(HWND hWnd, UINT msg,
        WPARAM wParam, LPARAM lParam) {
        HWND hwndCtl = (HWND) lParam;
        switch (msg) {
                case WM_DESTROY:
                        PostQuitMessage(0);
                        return 0;
                default:
                        return DefWindowProc(hWnd, msg, wParam,
lParam);
        }
        return 0;
}
```

Style 3

```
#include <windows.h>

LRESULT CALLBACK MainWndProc(HWND, UINT, WPARAM, LPARAM);

HWND hWnd;
HWND hwndLabel;


int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpCmdLine, int nCmdShow) {

        WNDCLASS wc;

        wc.lpszClassName = "Style3";
        wc.lpfnWndProc = MainWndProc;
        wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
        wc.hInstance = hInstance;
        wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
        wc.hCursor = LoadCursor( NULL, IDC_ARROW );
        wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );
        wc.lpszMenuName = "";
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        RegisterClass( &wc );

        hWnd = CreateWindow(
                "Style3",
                "Style 3",
                WS_BORDER | WS_SYSMENU | WS_MINIMIZEBOX,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                400,
                300,
                NULL,
```

```
                NULL,
                hInstance,
                NULL);

        hwndLabel = CreateWindow(
                "Static",
                "This window uses the WS_BORDER style, the WS_SYSMENU
style, and the WS_MINIMIZEBOX style",
                WS_VISIBLE | WS_CHILD,
                10,
                10,
                375,
                250,
                hWnd,
                NULL,
                hInstance,
                NULL);

        ShowWindow(hWnd, nCmdShow);

        MSG msg;
        while(GetMessage(&msg, NULL, 0, 0 )) {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }

        return (int) msg.wParam;
}

LRESULT CALLBACK MainWndProc(HWND hWnd, UINT msg,
        WPARAM wParam, LPARAM lParam) {
        HWND hwndCtl = (HWND) lParam;
        switch (msg) {
                case WM_DESTROY:
                        PostQuitMessage(0);
                        return 0;
                default:
                        return DefWindowProc(hWnd, msg, wParam,
lParam);
        }
        return 0;
}
```

Style 4

```
#include <windows.h>

LRESULT CALLBACK MainWndProc(HWND, UINT, WPARAM, LPARAM);

HWND hWnd;
HWND hwndLabel;
```

```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpCmdLine, int nCmdShow) {

        WNDCLASS wc;

        wc.lpszClassName = "Style4";
        wc.lpfnWndProc = MainWndProc;
        wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
        wc.hInstance = hInstance;
        wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
        wc.hCursor = LoadCursor( NULL, IDC_ARROW );
        wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );
        wc.lpszMenuName = "";
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        RegisterClass( &wc );

        hWnd = CreateWindow(
                "Style4",
                "Style 4",
                WS_BORDER | WS_SYSMENU | WS_MAXIMIZEBOX,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                400,
                300,
                NULL,
                NULL,
                hInstance,
                NULL);

        hwndLabel = CreateWindow(
                "Static",
                "This window uses the WS_BORDER style, the WS_SYSMENU
style, and the WS_MAXIMIZEBOX style",
                WS_VISIBLE | WS_CHILD,
                10,
                10,
                375,
                250,
                hWnd,
                NULL,
                hInstance,
                NULL);

        ShowWindow(hWnd, nCmdShow);

        MSG msg;
        while(GetMessage(&msg, NULL, 0, 0 )) {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }

        return (int) msg.wParam;
}
```

```
LRESULT CALLBACK MainWndProc(HWND hWnd, UINT msg,
        WPARAM wParam, LPARAM lParam) {
        HWND hwndCtl = (HWND) lParam;
        switch (msg) {
                case WM_DESTROY:
                        PostQuitMessage(0);
                        return 0;
                default:
                        return DefWindowProc(hWnd, msg, wParam,
lParam);
        }
        return 0;
}
```

Style 5

```
#include <windows.h>

LRESULT CALLBACK MainWndProc(HWND, UINT, WPARAM, LPARAM);

HWND hWnd;
HWND hwndLabel;


int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpCmdLine, int nCmdShow) {

        WNDCLASS wc;

        wc.lpszClassName = "Style5";
        wc.lpfnWndProc = MainWndProc;
        wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
        wc.hInstance = hInstance;
        wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
        wc.hCursor = LoadCursor( NULL, IDC_ARROW );
        wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );
        wc.lpszMenuName = "";
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        RegisterClass( &wc );

        hWnd = CreateWindow(
                "Style5",
                "Style 5",
                WS_OVERLAPPEDWINDOW | WS_VSCROLL,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                400,
                300,
                NULL,
```

```
                NULL,
                hInstance,
                NULL);

        hwndLabel = CreateWindow(
                "Static",
                "This window uses the WS_OVERLAPPEDWINDOW style and the
WS_VSCROLL style.",
                WS_VISIBLE | WS_CHILD,
                10,
                10,
                375,
                250,
                hWnd,
                NULL,
                hInstance,
                NULL);

        ShowWindow(hWnd, nCmdShow);

        MSG msg;
        while(GetMessage(&msg, NULL, 0, 0 )) {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }

        return (int) msg.wParam;
}

LRESULT CALLBACK MainWndProc(HWND hWnd, UINT msg,
        WPARAM wParam, LPARAM lParam) {
        HWND hwndCtl = (HWND) lParam;
        switch (msg) {
                case WM_DESTROY:
                        PostQuitMessage(0);
                        return 0;
                default:
                        return DefWindowProc(hWnd, msg, wParam,
lParam);
        }
        return 0;
}
```

C++ Win 32 Project: Windows API Project – Calculations

Calculations

```
#include <windows.h>
#include <cstring>
```

```
LRESULT CALLBACK MainWndProc(HWND, UINT, WPARAM, LPARAM);

HWND hWnd;
HWND hwndHoursLabel;
HWND hwndHoursEdit;
HWND hwndWageLabel;
HWND hwndWageEdit;
HWND hwndButton;
HWND hwndGrossLabel;
HWND hwndGrossAmount;
HWND hwndWithheldLabel;
HWND hwndWithheldAmount;
HWND hwndNetLabel;
HWND hwndNetAmount;
char szHoursResult[10];
char szWageResult[10];
char szResult[10];
double dHoursResult;
double dWageResult;
double dRegularPay;
double dOvertimePay;
double dGrossPay;
double dNetPay;
double dAmountWithheld;


int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpCmdLine, int nCmdShow) {

        WNDCLASS wc;

        wc.lpszClassName = "GuessNumber";
        wc.lpfnWndProc = MainWndProc;
        wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
        wc.hInstance = hInstance;
        wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
        wc.hCursor = LoadCursor( NULL, IDC_ARROW );
        wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );
        wc.lpszMenuName = "";
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        RegisterClass( &wc );

        hWnd = CreateWindow(
                "GuessNumber", "Guess Number",
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                225,
                225,
                NULL,
                NULL,
                hInstance,
                NULL);
```

```
        hwndHoursLabel = CreateWindow(
                "Static",
                "Hours worked:",
                WS_VISIBLE | WS_CHILD,
                10,
                10,
                100,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndHoursEdit = CreateWindow(
                "EDIT",
                NULL,
                WS VISIBLE | WS CHILD | WS BORDER | ES LEFT,
                110,
                10,
                90,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndWageLabel = CreateWindow(
                "Static",
                "Hourly wage:",
                WS_VISIBLE | WS_CHILD,
                10,
                35,
                100,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndWageEdit = CreateWindow(
                "EDIT",
                NULL,
                WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
                110,
                35,
                90,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndButton = CreateWindow(
                "BUTTON",
```

```
            "Calc Pay",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            10,
            60,
            190,
            35,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndGrossLabel = CreateWindow(
            "Static",
            "Gross Pay: ",
            WS_VISIBLE | WS_CHILD,
            10,
            105,
            100,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndGrossAmount = CreateWindow(
            "Static",
            NULL,
            WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
            110,
            105,
            90,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndWithheldLabel = CreateWindow(
            "Static",
            "Taxes (15%):",
            WS_VISIBLE | WS_CHILD,
            10,
            130,
            100,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndWithheldAmount = CreateWindow(
            "Static",
            NULL,
            WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
            110,
```

```
                130,
                90,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndNetLabel = CreateWindow(
                "Static",
                "Net pay:",
                WS_VISIBLE | WS_CHILD,
                10,
                155,
                100,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndNetAmount = CreateWindow(
                "Static",
                NULL,
                WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
                110,
                155,
                90,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        ShowWindow(hWnd, nCmdShow);

        MSG msg;
        while(GetMessage(&msg, NULL, 0, 0 )) {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }

        return (int) msg.wParam;
}


LRESULT CALLBACK MainWndProc(HWND hWnd, UINT msg,
        WPARAM wParam, LPARAM lParam) {
        HWND hwndCtl = (HWND) lParam;
        switch (msg) {
                case WM_COMMAND:
                        switch (wParam) {
                                case BN_CLICKED:

        GetWindowText(hwndHoursEdit, szHoursResult, 10);
```

```
        GetWindowText(hwndWageEdit, szWageResult, 10);
                                                dHoursResult =
atof(szHoursResult);
                                                dWageResult =
atof(szWageResult);
                                                if (dHoursResult <= 40)
                                                        dGrossPay =
dHoursResult * dWageResult;
                                                else {
                                                        dRegularPay = 40
* dWageResult;
                                                        dOvertimePay =
                                                           (dHoursResult
- 40) * (dWageResult * 1.5);
                                                        dGrossPay =
dRegularPay + dOvertimePay;
                                                }
                                                dAmountWithheld =
dGrossPay * .15;
                                                dNetPay = dGrossPay -
dAmountWithheld;
                                                 gcvt(dGrossPay, 10,
szResult);

        SetWindowText(hwndGrossAmount, szResult);
                                                _gcvt(dNetPay, 10,
szResult);

        SetWindowText(hwndWithheldAmount, szResult);
                                                _gcvt(dAmountWithheld,
10, szResult);

        SetWindowText(hwndNetAmount, szResult);
                        }
                        break;
                case WM_DESTROY:
                        PostQuitMessage(0);
                        return 0;
                default:
                        return DefWindowProc(hWnd, msg, wParam,
lParam);
        }
        return 0;
}
```

C++ Win 32 Project: Windows API Project - Calculations 2

Calulations 2

```
#include <windows.h>
```

```
#include <cstring>

LRESULT CALLBACK MainWndProc(HWND, UINT, WPARAM, LPARAM);

HWND hWnd;
HWND hwndEmployer1NameLabel;
HWND hwndEmployer1NameEdit;
HWND hwndEmployer1YearsLabel;
HWND hwndEmployer1YearsEdit;
HWND hwndEmployer1SalaryLabel;
HWND hwndEmployer1SalaryEdit;
HWND hwndEmployer2NameLabel;
HWND hwndEmployer2NameEdit;
HWND hwndEmployer2YearsLabel;
HWND hwndEmployer2YearsEdit;
HWND hwndEmployer2SalaryLabel;
HWND hwndEmployer2SalaryEdit;
HWND hwndEmployer3NameLabel;
HWND hwndEmployer3NameEdit;
HWND hwndEmployer3YearsLabel;
HWND hwndEmployer3YearsEdit;
HWND hwndEmployer3SalaryLabel;
HWND hwndEmployer3SalaryEdit;
HWND hwndHighestButton;
HWND hwndLowestButton;
HWND hwndLongestButton;
HWND hwndShortestButton;
HWND hwndMessage;
char szEmployer1Name[50];
char szEmployer2Name[50];
char szEmployer3Name[50];
char szEmployer1Years[5];
char szEmployer2Years[5];
char szEmployer3Years[5];
char szEmployer1Salary[10];
char szEmployer2Salary[10];
char szEmployer3Salary[10];
double dEmployer1Years;
double dEmployer2Years;
double dEmployer3Years;
double dEmployer1Salary;
double dEmployer2Salary;
double dEmployer3Salary;
char szResponse[50];

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpCmdLine, int nCmdShow) {

        WNDCLASS wc;

        wc.lpszClassName = "JobHistory";
        wc.lpfnWndProc = MainWndProc;
        wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
        wc.hInstance = hInstance;
        wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
```

```
        wc.hCursor = LoadCursor( NULL, IDC_ARROW );
        wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );
        wc.lpszMenuName = "";
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        RegisterClass( &wc );

        hWnd = CreateWindow(
                "JobHistory", "Job History",
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                300,
                450,
                NULL,
                NULL,
                hInstance,
                NULL);

        hwndEmployer1NameLabel = CreateWindow(
                "Static",
                "Employer 1 name",
                WS_VISIBLE | WS_CHILD,
                10,
                10,
                130,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndEmployer1NameEdit = CreateWindow(
                "EDIT",
                NULL,
                WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
                140,
                10,
                140,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndEmployer1YearsLabel = CreateWindow(
                "Static",
                "Employer 1 years",
                WS_VISIBLE | WS_CHILD,
                10,
                35,
                130,
                20,
                hWnd,
                NULL,
```

```
            hInstance,
            NULL);

        hwndEmployer1YearsEdit = CreateWindow(
            "EDIT",
            NULL,
            WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
            140,
            35,
            140,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndEmployer1SalaryLabel = CreateWindow(
            "Static",
            "Employer 1 pay",
            WS_VISIBLE | WS_CHILD,
            10,
            60,
            130,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndEmployer1SalaryEdit = CreateWindow(
            "EDIT",
            NULL,
            WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
            140,
            60,
            140,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndEmployer2NameLabel = CreateWindow(
            "Static",
            "Employer 2 name",
            WS VISIBLE | WS CHILD,
            10,
            100,
            130,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);
```

```
        hwndEmployer2NameEdit = CreateWindow(
                "EDIT",
                NULL,
                WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
                140,
                100,
                140,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndEmployer2YearsLabel = CreateWindow(
                "Static",
                "Employer 2 years",
                WS_VISIBLE | WS_CHILD,
                10,
                125,
                130,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndEmployer2YearsEdit = CreateWindow(
                "EDIT",
                NULL,
                WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
                140,
                125,
                140,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndEmployer2SalaryLabel = CreateWindow(
                "Static",
                "Employer 2 pay",
                WS_VISIBLE | WS_CHILD,
                10,
                150,
                130,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndEmployer2SalaryEdit = CreateWindow(
                "EDIT",
                NULL,
```

```
            WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
            140,
            150,
            140,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndEmployer3NameLabel = CreateWindow(
            "Static",
            "Employer 3 name",
            WS_VISIBLE | WS_CHILD,
            10,
            190,
            130,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndEmployer3NameEdit = CreateWindow(
            "EDIT",
            NULL,
            WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
            140,
            190,
            140,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndEmployer3YearsLabel = CreateWindow(
            "Static",
            "Employer 3 years",
            WS_VISIBLE | WS_CHILD,
            10,
            215,
            130,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndEmployer3YearsEdit = CreateWindow(
            "EDIT",
            NULL,
            WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
            140,
            215,
```

```
            140,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndEmployer3SalaryLabel = CreateWindow(
            "Static",
            "Employer 3 pay",
            WS_VISIBLE | WS_CHILD,
            10,
            240,
            130,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndEmployer3SalaryEdit = CreateWindow(
            "EDIT",
            NULL,
            WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
            140,
            240,
            140,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndHighestButton = CreateWindow(
            "BUTTON",
            "Highest Salary",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            10,
            275,
            270,
            25,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndLowestButton = CreateWindow(
            "BUTTON",
            "Lowest Salary",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            10,
            300,
            270,
            25,
            hWnd,
```

```
                NULL,
                hInstance,
                NULL);

        hwndLongestButton = CreateWindow(
                "BUTTON",
                "Longest Employment",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                10,
                325,
                270,
                25,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndShortestButton = CreateWindow(
                "BUTTON",
                "Shortest Employment",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                10,
                350,
                270,
                25,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndMessage = CreateWindow(
                "STATIC",
                NULL,
                WS_VISIBLE | WS_CHILD,
                10,
                385,
                270,
                25,
                hWnd,
                NULL,
                hInstance,
                NULL);

        ShowWindow(hWnd, nCmdShow);

        MSG msg;
        while(GetMessage(&msg, NULL, 0, 0 )) {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }

        return (int) msg.wParam;
}
```

```
LRESULT CALLBACK MainWndProc(HWND hWnd, UINT msg,
        WPARAM wParam, LPARAM lParam) {
        HWND hwndCtl = (HWND) lParam;
        switch (msg) {
                case WM_COMMAND:
                        switch (wParam) {
                                case BN_CLICKED:

                                GetWindowText(hwndEmployer1NameEdit,
szEmployer1Name, 50);
                                GetWindowText(hwndEmployer1YearsEdit,
szEmployer1Years, 10);
                                GetWindowText(hwndEmployer1SalaryEdit,
szEmployer1Salary, 10);
                                GetWindowText(hwndEmployer2NameEdit,
szEmployer2Name, 50);
                                GetWindowText(hwndEmployer2YearsEdit,
szEmployer2Years, 10);
                                GetWindowText(hwndEmployer2SalaryEdit,
szEmployer2Salary, 10);
                                GetWindowText(hwndEmployer3NameEdit,
szEmployer3Name, 50);
                                GetWindowText(hwndEmployer3YearsEdit,
szEmployer3Years, 10);
                                GetWindowText(hwndEmployer3SalaryEdit,
szEmployer3Salary, 10);
                                dEmployer1Years =
atof(szEmployer1Years);
                                dEmployer2Years =
atof(szEmployer2Years);
                                dEmployer3Years =
atof(szEmployer3Years);
                                dEmployer1Salary =
atof(szEmployer1Salary);
                                dEmployer2Salary =
atof(szEmployer2Salary);
                                dEmployer3Salary =
atof(szEmployer3Salary);

                                        if (hwndCtl ==
hwndHighestButton) {
                                                if (dEmployer1Salary >
dEmployer2Salary &&
                        dEmployer1Salary > dEmployer3Salary)
                        {

        strcpy(szResponse, szEmployer1Name);

        strcat(szResponse,
                                                                " paid you the
highest salary");
                                                }
                                                else if
(dEmployer2Salary > dEmployer1Salary &&
                        dEmployer2Salary > dEmployer3Salary)
```

```
                        {

        strcpy(szResponse, szEmployer2Name);

        strcat(szResponse,
                                                        " paid you the
highest salary");
                                                }
                                                else if
(dEmployer3Salary > dEmployer1Salary &&
                        dEmployer3Salary > dEmployer2Salary)
                        {

        strcpy(szResponse, szEmployer3Name);

        strcat(szResponse,
                                                        " paid you the
highest salary");
                                                }
                                        }
                                        else if (hwndCtl ==
hwndLowestButton) {
                                                if (dEmployer1Salary <
dEmployer2Salary &&
                        dEmployer1Salary < dEmployer3Salary)
                        {

        strcpy(szResponse, szEmployer1Name);

        strcat(szResponse,
                                                        " paid you the
lowest salary");
                                                }
                                                else if
(dEmployer2Salary < dEmployer1Salary &&
                        dEmployer2Salary < dEmployer3Salary)
                        {

        strcpy(szResponse, szEmployer2Name);

        strcat(szResponse,
                                                        " paid you the
lowest salary");
                                                }
                                                else if
(dEmployer3Salary < dEmployer1Salary &&
                        dEmployer3Salary < dEmployer2Salary)
                        {

        strcpy(szResponse, szEmployer3Name);

        strcat(szResponse,
                                                        " paid you the
lowest salary");
                                                }
```

```
                                        }
                                        else if (hwndCtl ==
hwndLongestButton)
                    {
                                                if (dEmployer1Years >
dEmployer2Years &&
                        dEmployer1Years > dEmployer3Years)
                        {

        strcpy(szResponse, szEmployer1Name);

        strcat(szResponse,
                                                            " was your
longest employer");
                                                }
                                                else if (dEmployer2Years
> dEmployer1Years &&
                        dEmployer2Years > dEmployer3Years)
                        {

        strcpy(szResponse, szEmployer2Name);

        strcat(szResponse,
                                                            " was your
longest employer");
                                                }
                                                else if (dEmployer3Years
> dEmployer1Years &&
                        dEmployer3Years > dEmployer2Years)
                        {

        strcpy(szResponse, szEmployer3Name);

        strcat(szResponse,
                                                            " was your
longest employer");
                                                }
                                        }
                                        else if (hwndCtl ==
hwndShortestButton)
                    {
                                                if (dEmployer1Years <
dEmployer2Years &&
                            dEmployer1Years < dEmployer3Years)
                            {

        strcpy(szResponse, szEmployer1Name);

        strcat(szResponse,
                                                            " was your
shortest employer");
                                                }
                                                else if (dEmployer2Years
< dEmployer1Years &&
                                    dEmployer2Years < dEmployer3Years)
```

```
                                {

        strcpy(szResponse, szEmployer2Name);

        strcat(szResponse,
                                                        " was your
shortest employer");
                                                    }
                                                else if (dEmployer3Years
< dEmployer1Years &&
                                    dEmployer3Years < dEmployer2Years)
                                {

        strcpy(szResponse, szEmployer3Name);

        strcat(szResponse,
                                                        " was your
shortest employer");
                                                    }
                                            }
                        }
                        SetWindowText(hwndMessage, szResponse);
                        break;
                case WM_DESTROY:
                        PostQuitMessage(0);
                        return 0;
                default:
                        return DefWindowProc(hWnd, msg, wParam,
lParam);
        }
        return 0;
}
```

C++ Win 32 Project: Windows API Project - Quiz

Quiz

```
#include <windows.h>
#include <cstring>

LRESULT CALLBACK MainWndProc(HWND, UINT, WPARAM, LPARAM);

HWND hWnd;
HWND hwndQ1Label;
HWND hwndQ1Edit;
HWND hwndQ2Label;
HWND hwndQ2Edit;
HWND hwndQ3Label;
HWND hwndQ3Edit;
```

```
HWND hwndQ4Label;
HWND hwndQ4Edit;
HWND hwndQ5Label;
HWND hwndQ5Edit;
HWND hwndButton;
HWND hwndResponse;
char szQ1Answer[10];
char szQ2Answer[10];
char szQ3Answer[10];
char szQ4Answer[10];
char szQ5Answer[10];
double dQ1Answer;
double dQ2Answer;
double dQ3Answer;
double dQ4Answer;
double dQ5Answer;
double dQ1CorrectAnswer = 12;
double dQ2CorrectAnswer = 13;
double dQ3CorrectAnswer = 14;
double dQ4CorrectAnswer = 156;
double dQ5CorrectAnswer = 26;
int iCorrect = 0;
char szResponse[50];


int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpCmdLine, int nCmdShow) {

        WNDCLASS wc;

        wc.lpszClassName = "MathQuiz";
        wc.lpfnWndProc = MainWndProc;
        wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
        wc.hInstance = hInstance;
        wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
        wc.hCursor = LoadCursor( NULL, IDC_ARROW );
        wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );
        wc.lpszMenuName = "";
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        RegisterClass( &wc );

        hWnd = CreateWindow(
                "MathQuiz", "Math Quiz",
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                475,
                250,
                NULL,
                NULL,
                hInstance,
                NULL);

        hwndQ1Label = CreateWindow(
```

```
            "Static",
            "What is the result of 84 divided by 7?",
            WS_VISIBLE | WS_CHILD,
            10,
            10,
            350,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndQ1Edit = CreateWindow(
            "EDIT",
            NULL,
            WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
            360,
            10,
            90,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndQ2Label = CreateWindow(
            "Static",
            "What is the value of x in the equation x * 12 = 56?",
            WS_VISIBLE | WS_CHILD,
            10,
            35,
            350,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndQ2Edit = CreateWindow(
            "EDIT",
            NULL,
            WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
            360,
            35,
            90,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndQ3Label = CreateWindow(
            "Static",
            "What is the square root of 196?",
            WS_VISIBLE | WS_CHILD,
```

```
            10,
            60,
            350,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndQ3Edit = CreateWindow(
            "EDIT",
            NULL,
            WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
            360,
            60,
            90,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndQ4Label = CreateWindow(
            "Static",
            "What is the value of 12 multiplied by 13?",
            WS_VISIBLE | WS_CHILD,
            10,
            85,
            350,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndQ4Edit = CreateWindow(
            "EDIT",
            NULL,
            WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
            360,
            85,
            90,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndQ5Label = CreateWindow(
            "Static",
            "What is result of the equation 14 + (2 * 6)?",
            WS_VISIBLE | WS_CHILD,
            10,
            110,
            350,
```

```
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndQ5Edit = CreateWindow(
                "EDIT",
                NULL,
                WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
                360,
                110,
                90,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndButton = CreateWindow(
                "BUTTON",
                "Score Quiz",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                10,
                140,
                440,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndResponse = CreateWindow(
                "STATIC",
                NULL,
                WS_VISIBLE | WS_CHILD | ES_LEFT,
                10,
                180,
                440,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);

        ShowWindow(hWnd, nCmdShow);

        MSG msg;
        while(GetMessage(&msg, NULL, 0, 0 )) {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }

        return (int) msg.wParam;
}
```

```
LRESULT CALLBACK MainWndProc(HWND hWnd, UINT msg,
        WPARAM wParam, LPARAM lParam) {
        HWND hwndCtl = (HWND) lParam;
        switch (msg) {
                case WM_COMMAND:
                        switch(wParam) {

                        case BN_CLICKED:

                        iCorrect = 0;
                        GetWindowText(hwndQ1Edit, szQ1Answer, 10);
                        dQ1Answer = atof(szQ1Answer);
                        GetWindowText(hwndQ2Edit, szQ2Answer, 10);
                        dQ2Answer = atof(szQ2Answer);
                        GetWindowText(hwndQ3Edit, szQ3Answer, 10);
                        dQ3Answer = atof(szQ3Answer);
                        GetWindowText(hwndQ4Edit, szQ4Answer, 10);
                        dQ4Answer = atof(szQ4Answer);
                        GetWindowText(hwndQ5Edit, szQ5Answer, 10);
                        dQ5Answer = atof(szQ5Answer);
                        if (dQ1Answer == NULL ||
                                dQ2Answer == NULL ||
                                dQ3Answer == NULL ||
                                dQ4Answer == NULL ||
                                dQ5Answer == NULL )

                                strcpy(szResponse, "You must answer all
five!");

                                else {
                                        if (dQ1Answer ==
dQ1CorrectAnswer)
                                            ++iCorrect;
                                        if (dQ2Answer ==
dQ2CorrectAnswer)
                                                ++iCorrect;
                                        if (dQ3Answer ==
dQ3CorrectAnswer)
                                                ++iCorrect;
                                        if (dQ4Answer ==
dQ4CorrectAnswer)
                                                ++iCorrect;
                                        if (dQ5Answer ==
dQ5CorrectAnswer)
                                                ++iCorrect;
                                        if (iCorrect == 0)
                                        strcpy(szResponse, "You answered
0 questions correctly!");
                                        else if (iCorrect == 1)
                                        strcpy(szResponse, "You answered
1 questions correctly!");
                                        else if (iCorrect == 2)
```

```
                                        strcpy(szResponse, "You answered
2 questions correctly!");
                                        else if (iCorrect == 3)
                                        strcpy(szResponse, "You answered
3 questions correctly!");
                                        else if (iCorrect == 4)
                                        strcpy(szResponse, "You answered
4 questions correctly!");
                                        else if (iCorrect == 5)
                                        strcpy(szResponse, "You answered
5 questions correctly!");
                                        }
                                        SetWindowText(hwndResponse,
szResponse);
                        }
                        break;
                case WM_DESTROY:
                        PostQuitMessage(0);
                        return 0;
                default:
                        return DefWindowProc(hWnd, msg, wParam, lParam);
        }
        return 0;
}
```

C++ Win 32 Project: Windows API Project - Calculations 2

Calulations 2 - Example 1

```
#include <windows.h>
#include <cstring>

LRESULT CALLBACK MainWndProc(HWND, UINT, WPARAM, LPARAM);
HWND hWnd;
HWND hwndLengthLabel;
HWND hwndLengthEdit;
HWND hwndWidthLabel;
HWND hwndWidthEdit;
HWND hwndCostLabel;
HWND hwndCostEdit;
HWND hwndButton;
HWND hwndMessage;
char szLength[10];
char szWidth[10];
char szCost[10];
double dLength;
double dWidth;
double dCost;
double dTotalCost;
```

```
char szTotalCost[10];
char szResult[50];


int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpCmdLine, int nCmdShow) {

        WNDCLASS wc;

        wc.lpszClassName = "CarpetCost";
        wc.lpfnWndProc = MainWndProc;
        wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
        wc.hInstance = hInstance;
        wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
        wc.hCursor = LoadCursor( NULL, IDC_ARROW );
        wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );
        wc.lpszMenuName = "";
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        RegisterClass( &wc );

        hWnd = CreateWindow(
                "CarpetCost", "Carpet Cost",
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                225,
                200,
                NULL,
                NULL,
                hInstance,
                NULL);

        hwndLengthLabel = CreateWindow(
                "Static",
                "Room length",
                WS_VISIBLE | WS_CHILD,
                10,
                10,
                100,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndLengthEdit = CreateWindow(
                "EDIT",
                NULL,
                WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
                110,
                10,
                90,
                20,
                hWnd,
```

```
            NULL,
            hInstance,
            NULL);

        hwndWidthLabel = CreateWindow(
            "Static",
            "Room width",
            WS_VISIBLE | WS_CHILD,
            10,
            35,
            100,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndWidthEdit = CreateWindow(
            "EDIT",
            NULL,
            WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
            110,
            35,
            90,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndCostLabel = CreateWindow(
            "Static",
            "Cost per foot",
            WS_VISIBLE | WS_CHILD,
            10,
            60,
            100,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);

        hwndCostEdit = CreateWindow(
            "EDIT",
            NULL,
            WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
            110,
            60,
            90,
            20,
            hWnd,
            NULL,
            hInstance,
            NULL);
```

```
        hwndButton = CreateWindow(
                "BUTTON",
                "Calculate Cost",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                10,
                90,
                190,
                25,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndMessage = CreateWindow(
                "STATIC",
                NULL,
                WS VISIBLE | WS CHILD | BS DEFPUSHBUTTON,
                10,
                125,
                190,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);

        ShowWindow(hWnd, nCmdShow);

        MSG msg;
        while(GetMessage(&msg, NULL, 0, 0 )) {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }

        return (int) msg.wParam;
}

LRESULT CALLBACK MainWndProc(HWND hWnd, UINT msg,
        WPARAM wParam, LPARAM lParam) {
        HWND hwndCtl = (HWND) lParam;
        switch (msg) {
                case WM_COMMAND:
                        switch (wParam) {
                                case BN_CLICKED:
                                        GetWindowText(hwndLengthEdit,
szLength, 10);
                                        dLength = atof(szLength);
                                        GetWindowText(hwndWidthEdit,
szWidth, 10);
                                        dWidth = atof(szWidth);
                                        GetWindowText(hwndCostEdit,
szCost, 10);
                                        dCost = atof(szCost);
```

```
                                        dTotalCost = dLength * dWidth *
dCost;
                                        _gcvt(dTotalCost, 10,
szTotalCost);
                                        strcpy(szResult, "The total cost
to carpet the room is $");
                                        strcat(szResult, szTotalCost);
                        }
                        SetWindowText(hwndMessage, szResult);
                        break;
                case WM_DESTROY:
                        PostQuitMessage(0);
                        return 0;
                default:
                        return DefWindowProc(hWnd, msg, wParam,
lParam);
        }
        return 0;
}
```

Retirement (3 Files)

```
//File 1 of 3, "Retire.cpp"
#include <windows.h>
#include <cstring>
#include "Retire.h"

LRESULT CALLBACK MainWndProc(HWND, UINT, WPARAM, LPARAM);

HWND hWnd;
HWND hwndAnnualLabel;
HWND hwndAnnualText;
HWND hwndYieldLabel;
HWND hwndYieldText;
HWND hwndCurrentAgeLabel;
HWND hwndCurrentAgeText;
HWND hwndRetAgeLabel;
HWND hwndRetAgeText;
HWND hwndInflationLabel;
HWND hwndInflationText;
HWND hwndFutureValueButton;
HWND hwndFutureValueText;
HWND hwndPresentValueButton;
HWND hwndPresentValueText;
HWND hwndTotalInterestButton;
HWND hwndTotalInterestText;
RetirementPlanner savings;
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE
hPrevInstance,
        LPSTR lpCmdLine, int nCmdShow) {
```

```
        WNDCLASS wc;

        wc.lpszClassName = "RetirementClass";
        wc.lpfnWndProc = MainWndProc;
        wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
        wc.hInstance = hInstance;
        wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
        wc.hCursor = LoadCursor( NULL, IDC_ARROW );
        wc.hbrBackground = (HBRUSH)( COLOR_BACKGROUND );
        wc.lpszMenuName = "";
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        RegisterClass( &wc );

        hWnd = CreateWindow(
                "RetirementClass", "Retirement Planner",
                WS_OVERLAPPEDWINDOW,
                CW USEDEFAULT,
                NULL,
                490,
                240,
                NULL,
                NULL,
                hInstance,
                NULL);

        hwndAnnualLabel = CreateWindow(
                "Static",
                " Annual Contribution",
                WS_VISIBLE | WS_CHILD | SS_SUNKEN ,
                10,
                10,
                250,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndAnnualText = CreateWindow(
                "EDIT",
                "0",
                WS_VISIBLE | WS_CHILD | WS_BORDER |
WS_TABSTOP | ES_LEFT,
                275,
                10,
                200,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndYieldLabel = CreateWindow(
                "Static",
```

```
                " Annual Yield (percent)",
                WS_VISIBLE | WS_CHILD | SS_SUNKEN ,
                10,
                35,
                250,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndYieldText = CreateWindow(
                "EDIT",
                "0",
                WS_VISIBLE | WS_CHILD | WS_BORDER |
WS_TABSTOP | ES_LEFT ,
                275,
                35,
                200,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndCurrentAgeLabel = CreateWindow(
                "Static",
                " Current Age",
                WS_VISIBLE | WS_CHILD | SS_SUNKEN ,
                10,
                60,
                250,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndCurrentAgeText = CreateWindow(
                "EDIT",
                "0",
                WS_VISIBLE | WS_CHILD | WS_BORDER |
WS_TABSTOP | ES_LEFT ,
                275,
                60,
                200,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndRetAgeLabel = CreateWindow(
                "Static",
                " Retirement Age",
```

```
                WS_VISIBLE | WS_CHILD | SS_SUNKEN ,
                10,
                85,
                250,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndRetAgeText = CreateWindow(
                "EDIT",
                "0",
                WS_VISIBLE | WS_CHILD | WS_BORDER |
WS_TABSTOP | ES_LEFT ,
                275,
                85,
                200,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndInflationLabel = CreateWindow(
                "Static",
                " Inflation (percent)",
                WS_VISIBLE | WS_CHILD | SS_SUNKEN ,
                10,
                110,
                250,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndInflationText = CreateWindow(
                "EDIT",
                "0",
                WS_VISIBLE | WS_CHILD | WS_BORDER |
WS_TABSTOP | ES_LEFT ,
                275,
                110,
                200,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndFutureValueButton = CreateWindow(
                "Button",
                " Total Future Value",
                WS_VISIBLE | WS_CHILD  | WS_TABSTOP,
```

```
                10,
                135,
                250,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndFutureValueText = CreateWindow(
                "STATIC",
                " CALCULATED",
                WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT
,
                275,
                135,
                200,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndPresentValueButton = CreateWindow(
                "Button",
                " Total Present Value",
                WS_VISIBLE | WS_CHILD  | WS_TABSTOP,
                10,
                160,
                250,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndPresentValueText = CreateWindow(
                "STATIC",
                " CALCULATED",
                WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT
,
                275,
                160,
                200,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);


        hwndTotalInterestButton = CreateWindow(
                "Button",
                " Total Interest Earned",
                WS_VISIBLE | WS_CHILD  | WS_TABSTOP,
```

```
                10,
                185,
                250,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        hwndTotalInterestText = CreateWindow(
                "STATIC",
                " CALCULATED",
                WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT
,
                275,
                185,
                200,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);

        ShowWindow(hWnd, nCmdShow);

        MSG msg;
        while(GetMessage(&msg, NULL, 0, 0 )) {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }

        return (int) msg.wParam;
}


LRESULT CALLBACK MainWndProc(HWND hWnd, UINT msg,
        WPARAM wParam, LPARAM lParam) {
        HWND hwndCtl = (HWND) lParam;
        char szContributeText[10] = "";
        char szYield[10] = "";
        char szCurrentAgeText[5] = "";
        char szRetAgeText[5] = "";
        char szInflationText[5] = "";
        double dContribute = 0;
        double dYield = 0;
        double dInflation = 0;
        int iCurrentAge = 0;
        int iRetAge = 0;
        double dReturnValue;
        char szResult[25];
        switch (msg) {
                case WM_COMMAND:
                        switch (wParam) {
                                case BN_CLICKED:
```

```
                                        if (hwndCtl ==
hwndFutureValueButton ||
                          hwndCtl == hwndPresentValueButton)
                    {

        GetWindowText(hwndAnnualText, szContributeText,
10);
                                                dContribute =
atof(szContributeText);

        GetWindowText(hwndYieldText, szYield, 10);
                                                dYield =
atof(szYield);

        GetWindowText(hwndCurrentAgeText, szCurrentAgeText,
5);
                                                iCurrentAge =
atoi(szCurrentAgeText);

        GetWindowText(hwndRetAgeText, szRetAgeText, 5);
                                                iRetAge =
atoi(szRetAgeText);

        GetWindowText(hwndInflationText, szInflationText,
10);
                                                dInflation =
atof(szInflationText);

        savings.setContribution(dContribute);

        savings.setInterestRate(dYield);

        savings.setCurAge(iCurrentAge);

        savings.setRetireAge(iRetAge);

        savings.setInflation(dInflation);
                                                if (hwndCtl
== hwndFutureValueButton) {

        dReturnValue = savings.calcFutureValue();

        _gcvt(dReturnValue, 10, szResult);

        SetWindowText(hwndFutureValueText, szResult);
                                                }
                                                else if
(hwndCtl == hwndPresentValueButton) {

        GetWindowText(hwndInflationText, szInflationText,
                                              5);

        dInflation = atof(szInflationText);

        dReturnValue = savings.calcPresentValue();
```

```
        _gcvt(dReturnValue, 10, szResult);

        SetWindowText(hwndPresentValueText, szResult);
                                                }
                                        }
                                        else if (hwndCtl ==
hwndTotalInterestButton) {
                                                dReturnValue
= savings.getInterestEarned();

        _gcvt(dReturnValue, 10, szResult);

        SetWindowText(hwndTotalInterestText, szResult);
                                        }
                        }
                        break;
                case WM_DESTROY:
                        PostQuitMessage(0);
                        return 0;
                default:
                        return DefWindowProc(hWnd, msg,
wParam, lParam);
        }
        return 0;
}
```

File 2 of 3

```
#include "retirementplanner.h"

RetirementPlanner::RetirementPlanner(void)
{
     dContribution = 0;
     dInterestRate = 0;
     dInterestEarned = 0;
     iYearsOfSaving = 0;
     dFutureValue = 0;
     dPresentValue = 0;
     dInflation = 0;
     iCurAge = 0;
     iRetireAge = 0;
}

RetirementPlanner::~RetirementPlanner(void)
{
}

void RetirementPlanner::setCurAge(int iAgeNow)
{
     iCurAge = iAgeNow;
}
```

```
void RetirementPlanner::setRetireAge(int iAgeThen)
{
     iRetireAge = iAgeThen;
}

void RetirementPlanner::setContribution(double dContribute)
{
     dContribution = dContribute;
}

void RetirementPlanner::setInterestRate(double dInterest)
{
     dInterestRate = dInterest;
}

void RetirementPlanner::setInflation(double dInflate)
{
        dInflation = dInflate;
}

double RetirementPlanner::calcFutureValue(void)
{
     iYearsOfSaving = iRetireAge - iCurAge;
     dFutureValue = 0;
     for (int i=0; i < iYearsOfSaving; i++) {
          dFutureValue += 1;
          dFutureValue *= (1+(dInterestRate/100));
     }
     dFutureValue *= dContribution;
     dInterestEarned = dFutureValue -
          (dContribution * iYearsOfSaving);
     return dFutureValue;
}

double RetirementPlanner::calcPresentValue(void) {
     double dFutureValue = calcFutureValue();
     for(int i = 0; i < iYearsOfSaving; i++) {
          dFutureValue /= (1 + (dInflation/100));
     }
     dPresentValue = dFutureValue;
     return dPresentValue;
}
```

File 3 of 3

```
#if !defined(RETIREMENT_H)
#define RETIREMENT_H
class RetirementPlanner
{
public:
        RetirementPlanner(void);
        ~RetirementPlanner(void);
private:
     double dContribution;
```

```
     double dInterestRate;
     double dInterestEarned;
     int iYearsOfSaving;
     double dFutureValue;
     double dPresentValue;
     double dInflation;
     int iCurAge;
     int iRetireAge;
public:
        void setCurAge(int iAgeNow);
        void setRetireAge(int iAgeThen);
        void setContribution(double dContribution);
        void setInterestRate(double dInterest);
        void setInflation(double dInflate);
        double getInterestEarned(void);
        double calcFutureValue(void);
        double calcPresentValue(void);
};
inline double RetirementPlanner::getInterestEarned(void)
{
        return dInterestEarned;
}
#endif
```

C++ Win 32 Project: Windows API Project - Expanded Calculator 3

Expanded Calculator 3- Example 1

```
#include <windows.h>
#include <cstring>
#include <math.h>
using namespace std;
void setNumbers(char szCurNum[10]);
void runCalculation();
char cOperation = '0';
char szFirstNum[10];
double dFirstNum = 0;
char szSecondNum[10];
double dSecondNum = 0;
char szCurNum[10];
double dCurNum = 0;
LRESULT CALLBACK MainWndProc(HWND, UINT, WPARAM, LPARAM);
HWND hWnd; HWND hwndEdit; HWND hwndButtonPlus;
HWND hwndButtonMinus; HWND hwndButtonMultiply;
HWND hwndButtonDivide; HWND hwndButton9;
HWND hwndButton8; HWND hwndButton7;
HWND hwndButton6; HWND hwndButton5;
HWND hwndButton4; HWND hwndButton3;
HWND hwndButton2; HWND hwndButton1;
HWND hwndButton0; HWND hwndButtonPoint;
```

```
HWND hwndButtonEquals; HWND hwndButtonClear;
HWND hwndButtonExp; HWND hwndButtonSqrt;
HWND hwndButtonSin; HWND hwndButtonCos;
int WINAPI WinMain(HINSTANCE hInstance,
                                HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine,
                                int nCmdShow) {
        WNDCLASS wc;
        wc.lpszClassName = "CalculatorClass";
        wc.lpfnWndProc = MainWndProc;
        wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
        wc.hInstance = hInstance;
        wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
        wc.hCursor = LoadCursor( NULL, IDC_ARROW );
        wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );
        wc.lpszMenuName = "";
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        RegisterClass(&wc);
        hWnd = CreateWindow(
                "CalculatorClass",
                "Calculator",
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                185,
                300,
                NULL,
                NULL,
                hInstance,
                NULL);
        hwndEdit = CreateWindow(
                "EDIT",
                NULL,
                WS_VISIBLE | WS_CHILD | WS_BORDER | ES_LEFT,
                10,
                10,
                155,
                20,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButtonPlus = CreateWindow(
                "BUTTON",
                "+",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                10,
                40,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButtonMinus = CreateWindow(
```

```
                "BUTTON",
                "-",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                50,
                40,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButtonMultiply = CreateWindow(
                "BUTTON",
                "*",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                90,
                40,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButtonDivide = CreateWindow(
                "BUTTON",
                "/",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                130,
                40,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButton6 = CreateWindow(
                "BUTTON",
                "6",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                10,
                80,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButton7 = CreateWindow(
                "BUTTON",
                "7",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                50,
                80,
                35,
                35,
                hWnd,
```

```
            NULL,
            hInstance,
            NULL);
        hwndButton8 = CreateWindow(
            "BUTTON",
            "8",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            90,
            80,
            35,
            35,
            hWnd,
            NULL,
            hInstance,
            NULL);
        hwndButton9 = CreateWindow(
            "BUTTON",
            "9",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            130,
            80,
            35,
            35,
            hWnd,
            NULL,
            hInstance,
            NULL);
        hwndButton2 = CreateWindow(
            "BUTTON",
            "2",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            10,
            120,
            35,
            35,
            hWnd,
            NULL,
            hInstance,
            NULL);
        hwndButton3 = CreateWindow(
            "BUTTON",
            "3",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            50,
            120,
            35,
            35,
            hWnd,
            NULL,
            hInstance,
            NULL);
        hwndButton4 = CreateWindow(
            "BUTTON",
            "4",
            WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
            90,
```

```
                120,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButton5 = CreateWindow(
                "BUTTON",
                "5",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                130,
                120,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButton0 = CreateWindow(
                "BUTTON",
                "0",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                10,
                160,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButton1 = CreateWindow(
                "BUTTON",
                "1",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                50,
                160,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButtonPoint = CreateWindow(
                "BUTTON",
                ".",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                90,
                160,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButtonEquals = CreateWindow(
```

```
                "BUTTON",
                "=",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                130,
                160,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButtonExp = CreateWindow(
                "BUTTON",
                "Exp",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                10,
                200,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButtonSqrt = CreateWindow(
                "BUTTON",
                "Sqrt",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                50,
                200,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButtonSin = CreateWindow(
                "BUTTON",
                "Sin",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                90,
                200,
                35,
                35,
                hWnd,
                NULL,
                hInstance,
                NULL);
        hwndButtonCos = CreateWindow(
                "BUTTON",
                "Cos",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                130,
                200,
                35,
                35,
                hWnd,
```

```
                NULL,
                hInstance,
                NULL);
        hwndButtonClear = CreateWindow(
                "BUTTON",
                "Clear",
                WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
                10,
                240,
                155,
                25,
                hWnd,
                NULL,
                hInstance,
                NULL);
        ShowWindow(hWnd, nCmdShow);
        MSG msg;
        while(GetMessage(&msg, NULL, 0, 0 )) {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }
        return (int) msg.wParam;
}
LRESULT CALLBACK MainWndProc(HWND hWnd, UINT msg,
        WPARAM wParam, LPARAM lParam) {
        HWND hwndCtl = (HWND) lParam;
        switch (msg) {
                case WM_COMMAND:
                        switch (wParam) {
                                case BN_CLICKED:
                                        if (hwndCtl == hwndButton1)
                                                setNumbers("1");
                                        else if (hwndCtl ==
hwndButton2)
                                                setNumbers("2");
                                        else if (hwndCtl ==
hwndButton3)
                                                setNumbers("3");
                                        else if (hwndCtl ==
hwndButton4)
                                                setNumbers("4");
                                        else if (hwndCtl ==
hwndButton5)
                                                setNumbers("5");
                                        else if (hwndCtl ==
hwndButton6)
                                                setNumbers("6");
                                        else if (hwndCtl ==
hwndButton7)
                                                setNumbers("7");
                                        else if (hwndCtl ==
hwndButton8)
                                                setNumbers("8");
                                        else if (hwndCtl ==
hwndButton9)
                                                setNumbers("9");
```

```
                                        else if (hwndCtl ==
hwndButton0)
                                                setNumbers("0");
                                        else if (hwndCtl ==
hwndButtonPoint)
                                                setNumbers(".");
                                        else if (hwndCtl ==
hwndButtonPlus) {
                                                cOperation = '+';
                                                dFirstNum =
atof(szFirstNum);
                                        }
                                        else if (hwndCtl ==
hwndButtonMinus) {
                                                cOperation = '-';
                                                dFirstNum =
atof(szFirstNum);
                                        }
                                        else if (hwndCtl ==
hwndButtonMultiply) {
                                                cOperation = '*';
                                                dFirstNum =
atof(szFirstNum);
                                        }
                                        else if (hwndCtl ==
hwndButtonDivide) {
                                                cOperation = '/';
                                                dFirstNum =
atof(szFirstNum);
                                        }
                                        else if (hwndCtl ==
hwndButtonExp) {

        GetWindowText(hwndEdit, szCurNum, 10);
                                                dCurNum =
atof(szCurNum);
                                                cOperation = 'e';
                                                runCalculation();
                                        }
                                        else if (hwndCtl ==
hwndButtonSqrt) {

        GetWindowText(hwndEdit, szCurNum, 10);
                                                dCurNum =
atof(szCurNum);
                                                cOperation = 'q';
                                                runCalculation();
                                        }
                                        else if (hwndCtl ==
hwndButtonSin) {

        GetWindowText(hwndEdit, szCurNum, 10);
                                                dCurNum =
atof(szCurNum);
                                                cOperation = 's';
                                                runCalculation();
```

```
                                        }
                                        else if (hwndCtl ==
hwndButtonCos) {

        GetWindowText(hwndEdit, szCurNum, 10);
                                                dCurNum =
atof(szCurNum);
                                                cOperation = 'c';
                                                runCalculation();
                                        }
                                        else if (hwndCtl ==
hwndButtonEquals) {
                                                dSecondNum =
atof(szSecondNum);
                                                runCalculation();
                                        }
                                        else if (hwndCtl ==
hwndButtonClear) {
                                                cOperation = '0';
                                                strcpy(szFirstNum,
"");
                                                dFirstNum = 0;
                                                strcpy(szSecondNum,
"");
                                                dSecondNum = 0;

        SetWindowText(hwndEdit, "");
                                        }
                        }
                        break;
                case WM_DESTROY:
                        PostQuitMessage(0);
                        return 0;
                default:
                        return DefWindowProc(hWnd, msg, wParam,
lParam);
        }
        return 0;
}
void setNumbers(char szCurNum[10]) {
        if (cOperation == '0') {
                strcat(szFirstNum, szCurNum);
                SetWindowText(hwndEdit, szFirstNum);
        }
        else {
                strcat(szSecondNum, szCurNum);
                SetWindowText(hwndEdit, szSecondNum);
        }
}
void runCalculation() {
        double dResult = 0;
        char szResult[25];
        if (cOperation == '+') {
                dResult = dFirstNum + dSecondNum;
        }
        else if (cOperation == '-') {
```

```
                dResult = dFirstNum - dSecondNum;
        }
        else if (cOperation == '*') {
                dResult = dFirstNum * dSecondNum;
        }
        else if (cOperation == '/') {
                dResult = dFirstNum / dSecondNum;
        }
        else if (cOperation == 'e') {
                dResult = exp(dCurNum);
        }
        else if (cOperation == 'q') {
                dResult = sqrt(dCurNum);
        }
        else if (cOperation == 's') {
                dResult = sin(dCurNum);
        }
        else if (cOperation == 'c') {
                dResult = cos(dCurNum);
        }
        _gcvt(dResult, 10, szResult);
        SetWindowText(hwndEdit, szResult);
        cOperation = '0';
        strcpy(szFirstNum, szResult);
        dFirstNum = 0;
        strcpy(szSecondNum, "");
        dSecondNum = 0;
}
```

C++ Win 32 Project: Windows API Project – Menus

Menus

File 1

```
// SimpleApp.cpp : Defines the entry point for the application.
//

#include "stdafx.h"
#include "SimpleApp.h"
#define MAX_LOADSTRING 100

// Global Variables:
HINSTANCE hInst; // current instance
TCHAR szTitle[MAX_LOADSTRING]; // The title bar text
TCHAR szWindowClass[MAX_LOADSTRING]; // the main window class name

// Forward declarations of functions included in this code module:
ATOM MyRegisterClass(HINSTANCE hInstance);
BOOL InitInstance(HINSTANCE, int);
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
```

```
LRESULT CALLBACK About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY _tWinMain(HINSTANCE hInstance,
HINSTANCE hPrevInstance,
LPSTR lpCmdLine,
int nCmdShow)
{
// TODO: Place code here.
MSG msg;
HACCEL hAccelTable;

// Initialize global strings
LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
LoadString(hInstance, IDC_SIMPLEAPP, szWindowClass, MAX_LOADSTRING);
MyRegisterClass(hInstance);

// Perform application initialization:
if (!InitInstance (hInstance, nCmdShow))
{
return FALSE;
}

hAccelTable = LoadAccelerators(hInstance, (LPCTSTR)IDC_SIMPLEAPP);

// Main message loop:
while (GetMessage(&msg, NULL, 0, 0))
{
if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
{
TranslateMessage(&msg);
DispatchMessage(&msg);
}
}

return (int) msg.wParam;
}



//
// FUNCTION: MyRegisterClass()
//
// PURPOSE: Registers the window class.
//
// COMMENTS:
//
// This function and its usage are only necessary if you want this
code
// to be compatible with Win32 systems prior to the
'RegisterClassEx'
// function that was added to Windows 95. It is important to call
this function
// so that the application will get 'well formed' small icons
associated
// with it.
//
```

```
ATOM MyRegisterClass(HINSTANCE hInstance)
{
WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS_HREDRAW | CS_VREDRAW;
wcex.lpfnWndProc = (WNDPROC)WndProc;
wcex.cbClsExtra = 0;
wcex.cbWndExtra = 0;
wcex.hInstance = hInstance;
wcex.hIcon = LoadIcon(hInstance, (LPCTSTR)IDI_SIMPLEAPP);
wcex.hCursor = LoadCursor(NULL, IDC_ARROW);
wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
wcex.lpszMenuName = (LPCTSTR)IDC_SIMPLEAPP;
wcex.lpszClassName = szWindowClass;
wcex.hIconSm = LoadIcon(wcex.hInstance, (LPCTSTR)IDI_SMALL);

return RegisterClassEx(&wcex);
}

//
// FUNCTION: InitInstance(HANDLE, int)
//
// PURPOSE: Saves instance handle and creates main window
//
// COMMENTS:
//
// In this function, we save the instance handle in a global
variable and
// create and display the main program window.
//
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
HWND hWnd;

hInst = hInstance; // Store instance handle in our global variable

hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

if (!hWnd)
{
return FALSE;
}

ShowWindow(hWnd, nCmdShow);
UpdateWindow(hWnd);

return TRUE;
}

//
// FUNCTION: WndProc(HWND, unsigned, WORD, LONG)
//
// PURPOSE: Processes messages for the main window.
```

```
//
// WM_COMMAND - process the application menu
// WM_PAINT - Paint the main window
// WM_DESTROY - post a quit message and return
//
//
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,
LPARAM lParam)
{
int wmId, wmEvent;
PAINTSTRUCT ps;
HDC hdc;

switch (message)
{
case WM_COMMAND:
wmId = LOWORD(wParam);
wmEvent = HIWORD(wParam);
// Parse the menu selections:
switch (wmId)
{
case IDM_ABOUT:
DialogBox(hInst, (LPCTSTR)IDD_ABOUTBOX, hWnd, (DLGPROC)About);
break;
case IDM_EXIT:
DestroyWindow(hWnd);
break;
default:
return DefWindowProc(hWnd, message, wParam, lParam);
}
break;
case WM_PAINT:
hdc = BeginPaint(hWnd, &ps);
// TODO: Add any drawing code here...
EndPaint(hWnd, &ps);
break;
case WM DESTROY:
PostQuitMessage(0);
break;
default:
return DefWindowProc(hWnd, message, wParam, lParam);
}
return 0;
}

// Message handler for about box.
LRESULT CALLBACK About(HWND hDlg, UINT message, WPARAM wParam,
LPARAM lParam)
{
switch (message)
{
case WM_INITDIALOG:
return TRUE;

case WM_COMMAND:
if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
```

```
{
EndDialog(hDlg, LOWORD(wParam));
return TRUE;
}
break;
}
return FALSE;
}
```

File 2

```
// stdafx.cpp : source file that includes just the standard includes
// SimpleApp.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

// TODO: reference any additional headers you need in STDAFX.H
// and not in this file
```

File 3

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by SimpleApp.rc
//

#define IDS_APP_TITLE 103

#define IDR_MAINFRAME 128
#define IDD_SIMPLEAPP_DIALOG 102
#define IDD_ABOUTBOX 103
#define IDM_ABOUT 104
#define IDM_EXIT 105
#define IDI_SIMPLEAPP 107
#define IDI_SMALL 108
#define IDC_SIMPLEAPP 109
#define IDC_MYICON 2
#define IDC_STATIC -1
// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS

#define _APS_NO_MFC 130
#define _APS_NEXT_RESOURCE_VALUE 129
#define _APS_NEXT_COMMAND_VALUE 32771
#define _APS_NEXT_CONTROL_VALUE 1000
#define _APS_NEXT_SYMED_VALUE 110
#endif
#endif
```

File4

```
#pragma once

#include "resource.h"
```

File 5

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

#pragma once

#define WIN32_LEAN_AND_MEAN // Exclude rarely-used stuff from
Windows headers
// Windows Header Files:
#include <windows.h>
// C RunTime Header Files
#include <stdlib.h>
#include <malloc.h>
#include <memory.h>
#include <tchar.h>

// TODO: reference additional headers your program requires here
```

File 6

```
//Microsoft Visual C++ generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
//
//
// Generated from the TEXTINCLUDE 2 resource.
//
#define APSTUDIO_HIDDEN_SYMBOLS
#include "windows.h"
#undef APSTUDIO_HIDDEN_SYMBOLS
////////////////////////////////////////////////////////////////////////////
//
#undef APSTUDIO_READONLY_SYMBOLS

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
LANGUAGE 9, 1
#pragma code_page(1252)
```

```
//////////////////////////////////////////////////////////////////////////////
//
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.

IDI_SIMPLEAPP ICON "SimpleApp.ico"
IDI_SMALL ICON "small.ico"

//////////////////////////////////////////////////////////////////////////////
//
//
// Menu
//

IDC_SIMPLEAPP MENU
BEGIN
POPUP "&File"
BEGIN
MENUITEM "E&xit", IDM_EXIT
END
POPUP "&Help"
BEGIN
MENUITEM "&About ...", IDM_ABOUT
END
END


//////////////////////////////////////////////////////////////////////////////
//
//
// Accelerator
//

IDC_SIMPLEAPP ACCELERATORS
BEGIN
"?", IDM_ABOUT, ASCII, ALT
"/", IDM_ABOUT, ASCII, ALT
END


//////////////////////////////////////////////////////////////////////////////
//
//
// Dialog
//

IDD_ABOUTBOX DIALOG 22, 17, 230, 75
STYLE DS_MODALFRAME | WS_CAPTION | WS_SYSMENU
CAPTION "About"
FONT 8, "System"
BEGIN
ICON IDI_SIMPLEAPP,IDC_MYICON,14,9,16,16
```

```
LTEXT "SimpleApp Version 1.0",IDC_STATIC,49,10,119,8,SS_NOPREFIX
LTEXT "Copyright (C) 2001",IDC_STATIC,49,20,119,8
DEFPUSHBUTTON "OK",IDOK,195,6,30,11,WS_GROUP
END


#ifdef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
//
//
// TEXTINCLUDE
//
1 TEXTINCLUDE
BEGIN
"resource.h\0"
END

2 TEXTINCLUDE
BEGIN
"#define APSTUDIO_HIDDEN_SYMBOLS\r\n"
"#include ""windows.h""\r\n"
"#undef APSTUDIO_HIDDEN_SYMBOLS\r\n"
"\0"
END

3 TEXTINCLUDE
BEGIN
"\r\n"
"\0"
END

#endif // APSTUDIO_INVOKED

////////////////////////////////////////////////////////////////////////////
//
//
// String Table
//

STRINGTABLE
BEGIN
IDC SIMPLEAPP "SIMPLEAPP"
IDS_APP_TITLE "SimpleApp"
END

#endif
////////////////////////////////////////////////////////////////////////////
//



#ifndef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
//
//
// Generated from the TEXTINCLUDE 3 resource.
```

```
//


////////////////////////////////////////////////////////////////////////////
//
#endif // not APSTUDIO_INVOKED
```

C++ Win 32 Project: Windows API Direct3D - Painting a Direct3D Background

Note: To compile Direct3D projects you will need to download the Direct3D SDK from Microsoft and install it on your system.
You will then need to reconfigure your compiler to find the \include and \lib directories.

File 1 of 1

```
//Painting a background with DIrectX 3D
//Note: To compile Direct3D projects you will need to download
//the Direct3D SDK from Microsoft

#include <d3d9.h>
#include <time.h>
//application title
#define APPTITLE "Direct3D_Windowed"
//forward declarations
LRESULT WINAPI WinProc(HWND,UINT,WPARAM,LPARAM);
ATOM MyRegisterClass(HINSTANCE);
int Game_Init(HWND);
void Game_Run(HWND);
void Game_End(HWND);
//Direct3D objects
LPDIRECT3D9 d3d = NULL;
LPDIRECT3DDEVICE9 d3ddev = NULL;
//window event callback function
LRESULT WINAPI WinProc( HWND hWnd, UINT msg, WPARAM wParam, LPARAM
lParam )
{
    switch( msg )
    {
        case WM_DESTROY:
            Game_End(hWnd);
            PostQuitMessage(0);
            return 0;
    }
    return DefWindowProc( hWnd, msg, wParam, lParam );
}
//helper function to set up the window properties
ATOM MyRegisterClass(HINSTANCE hInstance)
{
    //create the window class structure
    WNDCLASSEX wc;
```

```
    wc.cbSize = sizeof(WNDCLASSEX);
    //fill the struct with info
    wc.style         = CS_HREDRAW | CS_VREDRAW;
    wc.lpfnWndProc   = (WNDPROC)WinProc;
    wc.cbClsExtra       = 0;
    wc.cbWndExtra       = 0;
    wc.hInstance     = hInstance;
    wc.hIcon         = NULL;
    wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
    wc.lpszMenuName  = NULL;
    wc.lpszClassName = APPTITLE;
    wc.hIconSm       = NULL;
    //set up the window with the class info
    return RegisterClassEx(&wc);
}
//entry point for a Windows program
int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPSTR     lpCmdLine,
                   int       nCmdShow)
{
    // declare variables
        MSG msg;
        // register the class
        MyRegisterClass(hInstance);
    // initialize application
    //note--got rid of initinstance
    HWND hWnd;
    //create a new window
    hWnd = CreateWindow(
       APPTITLE,              //window class
       APPTITLE,              //title bar
       WS_OVERLAPPEDWINDOW,   //window style
       CW_USEDEFAULT,         //x position of window
       CW_USEDEFAULT,         //y position of window
       500,                   //width of the window
       400,                   //height of the window
       NULL,                  //parent window
       NULL,                  //menu
       hInstance,             //application instance
       NULL);                 //window parameters
    //was there an error creating the window?
    if (!hWnd)
      return FALSE;
    //display the window
    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

        //initialize the game
    if (!Game_Init(hWnd))
        return 0;
    // main message loop
    int done = 0;
        while (!done)
    {
```

```
        if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
            {
            //look for quit message
            if (msg.message == WM_QUIT)
            {
                MessageBox(hWnd, "Received WM_QUIT message",
"WinMain", MB_OK);
                done = 1;
            }
            //decode and pass messages on to WndProc
                   TranslateMessage(&msg);
                   DispatchMessage(&msg);
            }
        else
            //process game loop (else prevents running after window
is closed)
            Game_Run(hWnd);
    }
        return msg.wParam;
}
int Game_Init(HWND hwnd)
{
    //display init message
    MessageBox(hwnd, "Program is about to start", "Game_Init",
MB_OK);
    //initialize Direct3D
    d3d = Direct3DCreate9(D3D_SDK_VERSION);
    if (d3d == NULL)
    {
        MessageBox(hwnd, "Error initializing Direct3D", "Error",
MB_OK);
        return 0;
    }
    //set Direct3D presentation parameters
    D3DPRESENT_PARAMETERS d3dpp;
    ZeroMemory(&d3dpp, sizeof(d3dpp));
    d3dpp.Windowed = TRUE;
    d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
    d3dpp.BackBufferFormat = D3DFMT_UNKNOWN;
    //create Direct3D device
    d3d->CreateDevice(
        D3DADAPTER_DEFAULT,
        D3DDEVTYPE_HAL,
        hwnd,
        D3DCREATE_SOFTWARE_VERTEXPROCESSING,
        &d3dpp,
        &d3ddev);
    if (d3ddev == NULL)
    {
        MessageBox(hwnd, "Error creating Direct3D device", "Error",
MB_OK);
        return 0;
    }
        //set random number seed
        srand(time(NULL));
    //return okay
```

```
    return 1;
}
void Game_Run(HWND hwnd)
{
    //make sure the Direct3D device is valid
    if (d3ddev == NULL)
        return;
    //clear the backbuffer to black
    d3ddev->Clear(0, NULL, D3DCLEAR_TARGET, D3DCOLOR_XRGB(0,255,0),
1.0f, 0);

    //start rendering
    if (d3ddev->BeginScene())
    {


        //stop rendering
        d3ddev->EndScene();
    }
    //display the back buffer on the screen
    d3ddev->Present(NULL, NULL, NULL, NULL);
}
void Game_End(HWND hwnd)
{
    //display close message
    MessageBox(hwnd, "Program is about to end", "Game_End", MB_OK);
    //release the Direct3D device
    if (d3ddev != NULL)
        d3ddev->Release();
    //release the Direct3D object
    if (d3d != NULL)
        d3d->Release();
}
```

C++ Win 32 Project: Windows API Direct3D - Flashing Background

Note: To compile Direct3D projects you will need to download the Direct3D SDK from Microsoft and install it on your system.
You will then need to reconfigure your compiler to find the \include and \lib directories.

File 1 of 1

```
// Flashing Background
//header files to include
#include <d3d9.h>
#include <time.h>
//application title
#define APPTITLE "Create_Surface"
//macros to read the keyboard asynchronously
#define KEY_DOWN(vk_code) ((GetAsyncKeyState(vk_code) & 0x8000) ? 1
: 0)
```

```
#define KEY_UP(vk_code)((GetAsyncKeyState(vk_code) & 0x8000) ? 1 :
0)
//screen resolution
#define SCREEN_WIDTH 640
#define SCREEN_HEIGHT 480
//forward declarations
LRESULT WINAPI WinProc(HWND,UINT,WPARAM,LPARAM);
ATOM MyRegisterClass(HINSTANCE);
int Game_Init(HWND);
void Game_Run(HWND);
void Game_End(HWND);
//Direct3D objects
LPDIRECT3D9 d3d = NULL;
LPDIRECT3DDEVICE9 d3ddev = NULL;
LPDIRECT3DSURFACE9 backbuffer = NULL;
LPDIRECT3DSURFACE9 surface = NULL;
//window event callback function
LRESULT WINAPI WinProc( HWND hWnd, UINT msg, WPARAM wParam, LPARAM
lParam )
{
    switch( msg )
    {
        case WM_DESTROY:
            Game_End(hWnd);
            PostQuitMessage(0);
            return 0;
    }
    return DefWindowProc( hWnd, msg, wParam, lParam );
}
//helper function to set up the window properties
ATOM MyRegisterClass(HINSTANCE hInstance)
{
    //create the window class structure
    WNDCLASSEX wc;
    wc.cbSize = sizeof(WNDCLASSEX);
    //fill the struct with info
    wc.style         = CS_HREDRAW | CS_VREDRAW;
    wc.lpfnWndProc   = (WNDPROC)WinProc;
    wc.cbClsExtra       = 0;
    wc.cbWndExtra       = 0;
    wc.hInstance     = hInstance;
    wc.hIcon         = NULL;
    wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
    wc.lpszMenuName  = NULL;
    wc.lpszClassName = APPTITLE;
    wc.hIconSm       = NULL;
    //set up the window with the class info
    return RegisterClassEx(&wc);
}
//entry point for a Windows program
int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPSTR     lpCmdLine,
                   int       nCmdShow)
{
```

```
    // declare variables
        MSG msg;
        // register the class
        MyRegisterClass(hInstance);
    // initialize application
    //note--got rid of initinstance
    HWND hWnd;
    //create a new window
    hWnd = CreateWindow(
       APPTITLE,              //window class
       APPTITLE,              //title bar
       WS_EX_TOPMOST | WS_VISIBLE | WS_POPUP,   //window style
       CW_USEDEFAULT,         //x position of window
       CW_USEDEFAULT,         //y position of window
       SCREEN_WIDTH,          //width of the window
       SCREEN_HEIGHT,         //height of the window
       NULL,                  //parent window
       NULL,                  //menu
       hInstance,             //application instance
       NULL);                 //window parameters
    //was there an error creating the window?
    if (!hWnd)
      return FALSE;
    //display the window
    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

        //initialize the game
    if (!Game_Init(hWnd))
        return 0;
    // main message loop
    int done = 0;
        while (!done)
    {
        if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
            {
            //look for quit message
            if (msg.message == WM_QUIT)
                done = 1;
            //decode and pass messages on to WndProc
                   TranslateMessage(&msg);
                   DispatchMessage(&msg);
            }
        else
            //process game loop (else prevents running after window
is closed)
            Game_Run(hWnd);
    }
        return msg.wParam;
}
int Game_Init(HWND hwnd)
{
    HRESULT result;
    //initialize Direct3D
    d3d = Direct3DCreate9(D3D_SDK_VERSION);
    if (d3d == NULL)
```

```
    {
        MessageBox(hwnd, "Error initializing Direct3D", "Error",
MB_OK);
        return 0;
    }
    //set Direct3D presentation parameters
    D3DPRESENT_PARAMETERS d3dpp;
    ZeroMemory(&d3dpp, sizeof(d3dpp));
    d3dpp.Windowed = FALSE;
    d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
    d3dpp.BackBufferFormat = D3DFMT_X8R8G8B8;
    d3dpp.BackBufferCount = 1;
    d3dpp.BackBufferWidth = SCREEN_WIDTH;
    d3dpp.BackBufferHeight = SCREEN_HEIGHT;
    d3dpp.hDeviceWindow = hwnd;
    //create Direct3D device
    d3d->CreateDevice(
        D3DADAPTER_DEFAULT,
        D3DDEVTYPE_HAL,
        hwnd,
        D3DCREATE_SOFTWARE_VERTEXPROCESSING,
        &d3dpp,
        &d3ddev);
    if (d3ddev == NULL)
    {
        MessageBox(hwnd, "Error creating Direct3D device", "Error",
MB_OK);
        return 0;
    }
        //set random number seed
        srand(time(NULL));
    //clear the backbuffer to black
    d3ddev->Clear(0, NULL, D3DCLEAR_TARGET, D3DCOLOR_XRGB(0,0,0),
1.0f, 0);

    //create pointer to the back buffer
    d3ddev->GetBackBuffer(0, 0, D3DBACKBUFFER_TYPE_MONO,
&backbuffer);
    //create surface
    result = d3ddev->CreateOffscreenPlainSurface(
        100,                //width of the surface
        100,                //height of the surface
        D3DFMT_X8R8G8B8,    //surface format
        D3DPOOL_DEFAULT,    //memory pool to use
        &surface,           //pointer to the surface
        NULL);              //reserved (always NULL)
    if (!result)
        return 1;
    //return okay
    return 1;
}
void Game_Run(HWND hwnd)
{
    RECT rect;
    int r,g,b;
    //make sure the Direct3D device is valid
```

```
    if (d3ddev == NULL)
        return;
    //start rendering
    if (d3ddev->BeginScene())
    {
        //fill the surface with random color
        r = rand() % 255;
        g = rand() % 255;
        b = rand() % 255;
        d3ddev->ColorFill(surface, NULL, D3DCOLOR_XRGB(r,g,b));
        //copy the surface to the backbuffer
        rect.left = rand() % SCREEN_WIDTH/2;
        rect.right = rect.left + rand() % SCREEN_WIDTH/2;
        rect.top = rand() % SCREEN_HEIGHT;
        rect.bottom = rect.top + rand() % SCREEN_HEIGHT/2;
        d3ddev->StretchRect(surface, NULL, backbuffer, &rect,
D3DTEXF_NONE);

        //stop rendering
        d3ddev->EndScene();
    }
    //display the back buffer on the screen
    d3ddev->Present(NULL, NULL, NULL, NULL);
    //check for escape key (to exit program)
    if (KEY_DOWN(VK_ESCAPE))
        PostMessage(hwnd, WM_DESTROY, 0, 0);
}
void Game_End(HWND hwnd)
{
    //free the surface
    surface->Release();
    //release the Direct3D device
    if (d3ddev != NULL)
        d3ddev->Release();
    //release the Direct3D object
    if (d3d != NULL)
        d3d->Release();
}
```

C++ Win 32 Project: Windows API Direct3D - Loading a Texture and Bitmap

Note: To compile Direct3D projects you will need to download the Direct3D SDK from Microsoft and install it on your system.

You will then need to reconfigure your compiler to find the \include and \lib directories.

File 1 of 1

```
// Beginning Game Programming
// Chapter 6
// Load_Bitmap program
```

```
//header files to include
#include <d3d9.h>
#include <d3dx9.h>
#include <time.h>
//application title
#define APPTITLE "Create_Surface"
//macros to read the keyboard asynchronously
#define KEY_DOWN(vk_code) ((GetAsyncKeyState(vk_code) & 0x8000) ? 1
: 0)
#define KEY_UP(vk_code)((GetAsyncKeyState(vk_code) & 0x8000) ? 1 :
0)
//screen resolution
#define SCREEN_WIDTH 640
#define SCREEN_HEIGHT 480
//forward declarations
LRESULT WINAPI WinProc(HWND,UINT,WPARAM,LPARAM);
ATOM MyRegisterClass(HINSTANCE);
int Game_Init(HWND);
void Game_Run(HWND);
void Game_End(HWND);
//Direct3D objects
LPDIRECT3D9 d3d = NULL;
LPDIRECT3DDEVICE9 d3ddev = NULL;
LPDIRECT3DSURFACE9 backbuffer = NULL;
LPDIRECT3DSURFACE9 surface = NULL;
//window event callback function
LRESULT WINAPI WinProc( HWND hWnd, UINT msg, WPARAM wParam, LPARAM
lParam )
{
    switch( msg )
    {
        case WM_DESTROY:
            Game_End(hWnd);
            PostQuitMessage(0);
            return 0;
    }
    return DefWindowProc( hWnd, msg, wParam, lParam );
}
//helper function to set up the window properties
ATOM MyRegisterClass(HINSTANCE hInstance)
{
    //create the window class structure
    WNDCLASSEX wc;
    wc.cbSize = sizeof(WNDCLASSEX);
    //fill the struct with info
    wc.style         = CS_HREDRAW | CS_VREDRAW;
    wc.lpfnWndProc   = (WNDPROC)WinProc;
    wc.cbClsExtra       = 0;
    wc.cbWndExtra       = 0;
    wc.hInstance     = hInstance;
    wc.hIcon         = NULL;
    wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
    wc.lpszMenuName  = NULL;
    wc.lpszClassName = APPTITLE;
    wc.hIconSm       = NULL;
```

```
    //set up the window with the class info
    return RegisterClassEx(&wc);
}
//entry point for a Windows program
int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPSTR     lpCmdLine,
                   int       nCmdShow)
{
    // declare variables
        MSG msg;
        // register the class
        MyRegisterClass(hInstance);
    // initialize application
    //note--got rid of initinstance
    HWND hWnd;
    //create a new window
    hWnd = CreateWindow(
       APPTITLE,              //window class
       APPTITLE,              //title bar
       WS_EX_TOPMOST | WS_VISIBLE | WS_POPUP,   //window style
       CW_USEDEFAULT,         //x position of window
       CW_USEDEFAULT,         //y position of window
       SCREEN_WIDTH,          //width of the window
       SCREEN_HEIGHT,         //height of the window
       NULL,                  //parent window
       NULL,                  //menu
       hInstance,             //application instance
       NULL);                 //window parameters
    //was there an error creating the window?
    if (!hWnd)
      return FALSE;
    //display the window
    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

        //initialize the game
    if (!Game_Init(hWnd))
        return 0;
    // main message loop
    int done = 0;
        while (!done)
    {
        if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
            {
            //look for quit message
            if (msg.message == WM_QUIT)
                done = 1;
            //decode and pass messages on to WndProc
                   TranslateMessage(&msg);
                   DispatchMessage(&msg);
            }
        else
            //process game loop (else prevents running after window
is closed)
            Game_Run(hWnd);
```

```
    }
        return msg.wParam;
}
int Game_Init(HWND hwnd)
{
    HRESULT result;
    //initialize Direct3D
    d3d = Direct3DCreate9(D3D_SDK_VERSION);
    if (d3d == NULL)
    {
        MessageBox(hwnd, "Error initializing Direct3D", "Error",
MB_OK);
        return 0;
    }
    //set Direct3D presentation parameters
    D3DPRESENT_PARAMETERS d3dpp;
    ZeroMemory(&d3dpp, sizeof(d3dpp));
    d3dpp.Windowed = FALSE;
    d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
    d3dpp.BackBufferFormat = D3DFMT_X8R8G8B8;
    d3dpp.BackBufferCount = 1;
    d3dpp.BackBufferWidth = SCREEN_WIDTH;
    d3dpp.BackBufferHeight = SCREEN_HEIGHT;
    d3dpp.hDeviceWindow = hwnd;
    //create Direct3D device
    d3d->CreateDevice(
        D3DADAPTER_DEFAULT,
        D3DDEVTYPE_HAL,
        hwnd,
        D3DCREATE_SOFTWARE_VERTEXPROCESSING,
        &d3dpp,
        &d3ddev);
    if (d3ddev == NULL)
    {
        MessageBox(hwnd, "Error creating Direct3D device", "Error",
MB_OK);
        return 0;
    }
        //set random number seed
        srand(time(NULL));
    //clear the backbuffer to black
    d3ddev->Clear(0, NULL, D3DCLEAR TARGET, D3DCOLOR XRGB(0,0,0),
1.0f, 0);

    //create surface
    result = d3ddev->CreateOffscreenPlainSurface(
        640,                //width of the surface
        480,                //height of the surface
        D3DFMT_X8R8G8B8,    //surface format
        D3DPOOL_DEFAULT,    //memory pool to use
        &surface,           //pointer to the surface
        NULL);              //reserved (always NULL)
    if (result != D3D_OK)
        return 1;
    //load surface from file into newly created surface
    result = D3DXLoadSurfaceFromFile(
```

```
        surface,            //destination surface
        NULL,               //destination palette
        NULL,               //destination rectangle
        "legotron.bmp",     //source filename
        NULL,               //source rectangle
        D3DX_DEFAULT,       //controls how image is filtered
        0,                  //for transparency (0 for none)
        NULL);              //source image info (usually NULL)
    //make sure file was loaded okay
    if (result != D3D_OK)
        return 1;
    //return okay
    return 1;
}
void Game_Run(HWND hwnd)
{
    //make sure the Direct3D device is valid
    if (d3ddev == NULL)
        return;
    //start rendering
    if (d3ddev->BeginScene())
    {
        //create pointer to the back buffer
        d3ddev->GetBackBuffer(0, 0, D3DBACKBUFFER_TYPE_MONO,
&backbuffer);
        //draw surface to the backbuffer
        d3ddev->StretchRect(surface, NULL, backbuffer, NULL,
D3DTEXF_NONE);

        //stop rendering
        d3ddev->EndScene();
    }
    //display the back buffer on the screen
    d3ddev->Present(NULL, NULL, NULL, NULL);
    //check for escape key (to exit program)
    if (KEY_DOWN(VK_ESCAPE))
        PostMessage(hwnd, WM_DESTROY, 0, 0);
}
void Game_End(HWND hwnd)
{
    //free the surface
    surface->Release();
    //release the Direct3D device
    if (d3ddev != NULL)
        d3ddev->Release();
    //release the Direct3D object
    if (d3d != NULL)
        d3d->Release();
}
```

Old C Code Projects

C Code Projects - 2004 C. Germany

C is the predecessor of C++, and as such we owe much of C++'s power and finesse to this efficient and beautiful language!

Currency Converter 3.0

Download: CurrencyConverter.exe

```
// Curreny Calculator 3.0 - 2004 C. Germany - Team A Software
// Flow Chart At Bottom

#include <stdio.h>

// This function encapsulates the error checking process.
// If we take input as a float or integer, when the user enters
// a character/string, it will throw the program into an infinite
// loop and crash it.  If, on the other hand, we take input as
// a string, we can parse the string for ASCII values that relate
// to numbers and then we can handle both kinds of input.

// The first part parses numbers to the left of the decimal,
// then the second part parses numbers to the right.

// If we get a non-number ASCII value, an error message is displayed
// and the the return value is set to 0.0.

//For the "Miracle C" Compiler, these had to be globally declared
outside the function.
int x;
float THEamount = 0.0;
float factor;         //Factor for decimal places

//-------------------------------------------------------------------
----------------------------

float CheckErrors(char * amountString)
{
        x = 0;
        THEamount = 0.0;
        printf("String value passed is %s.", amountString);

        while( (amountString[x] - 48) < 10 && (amountString[x] - 48) > 0
)
        {
                //Multiply by 10 to move the decimal place to the
left.
                THEamount = 10 * THEamount + (amountString[x] - 48);
                 x++;   //proceed to next character in the array
        }

        //If last character is a decimal point, we need to move numbers
to
```

```
       //the right instead of the left.

       if(amountString[x] == '.')
       {
                printf("\nI see a decimal point!\n");
                x++;             //Need to increment x to the next
character FIRST

               factor = 1.0;    //factor needs to be reinitialized each
time!

               while( (amountString[x] - 48) < 10 && (amountString[x] -
48) > 0 )
              {
                           factor = factor * 0.1;  //Decrease by factor
of 10
                                                          //Add a
decimal place to the right
                           THEamount = THEamount + (amountString[x] - 48)
* factor;
                           x++;
              }

              //If they type a non-number ASCII value after the decimal
flag
              //it as an error and return 0.0 as the value.
               if(!(amountString[x] - 48) < 10 && (amountString[x] - 48)
> 0)
              {
                      printf("\n\nSorry, but that was not a whole
number.\n");
                      printf("The number you have entered will be set to
0 (NULL).\n");
                      THEamount = 0.0;
                      return THEamount;
               }

          }  //close block that executes if char was a '.'

          else
          {
                    //If they type a non-number ASCII value before the
decimal flag
                    //it as an error and return 0.0 as the value.
                    if(!(amountString[x] - 48) < 10 && (amountString[x] -
48) > 0)
                    {
                         printf("\n\nSorry, but that was not a normal
number!\n");
                         printf("The strange string you have entered will
be set to 0 NULL.\n");
                         THEamount = 0.0;
                         return THEamount;
                    }
          }
```

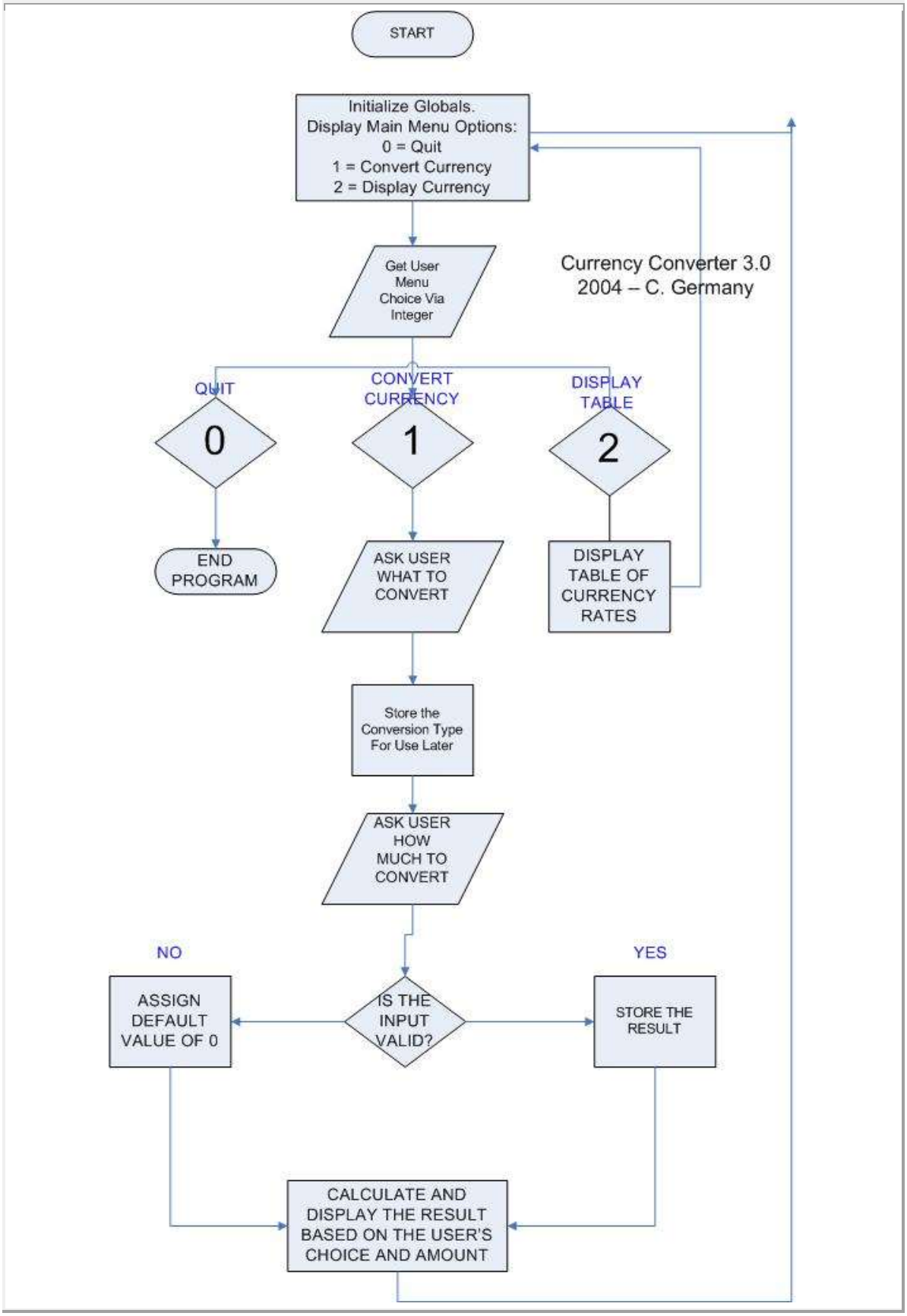
```
            return THEamount;

} //close function

//-------------------------------------------------------------------------
-----------------------------

void ConvertCurrency()
{
      //Note: "to" = units to 1 US Dollar
      //      "from" = 1 US Dollar to units

     int selection = 100;
     float amount = 0.0, result = 0.0;
     char amt[10];

     //Variable for converting to and from the US dollar. Self
explanatory.
     float EUROSto, RUBLESto, PESOSto, RUPEESto, YENto, POUNDSto;
     float EUROSfrom, RUBLESfrom, PESOSfrom, RUPEESfrom, YENfrom,
POUNDSfrom;

     EUROSto = .8040;
     RUBLESto = 24.00884;
     PESOSto = 11.4155;
     RUPEESto = 45.9137;
     YENto = 108.778;
     POUNDSto = .5343;

     EUROSfrom = 1.2438;
     RUBLESfrom = .0344;
     PESOSfrom = .0876;
     RUPEESfrom = .0218;
     YENfrom = .0092;
     POUNDSfrom = 1.8715;

     //Print a menu of conversion choices
     printf("\n\t*************** Main Menu ***************");
     printf("\n\t*                                       *");
     printf("\n\t*     Choose an option below:           *");
     printf("\n\t*                                       *");
     printf("\n\t*     1. US Dollars to EUROS.           *");
     printf("\n\t*     2. US Dollars to RUBLES.          *");
     printf("\n\t*     3. US Dollars to PESOS.           *");
     printf("\n\t*     4. US Dollars to RUPEES.          *");
     printf("\n\t*     5. US Dollars to YEN.             *");
     printf("\n\t*     6. US Dollars to POUNDS.          *");
     printf("\n\t*     7. EUROS to US Dollars.           *");
     printf("\n\t*     8. RUBLES to US Dollars.          *");
     printf("\n\t*     9. PESOS to US Dollars.           *");
     printf("\n\t*     10. RUPEES to US Dollars.         *");
     printf("\n\t*     11. YEN to US Dollars.            *");
     printf("\n\t*     12. POUNDS to US Dollars.         *");
     printf("\n\t*                                       *");
     printf("\n\t*****************************************\n\n");
```

```
     scanf("%d", &selection); // Get their choice

     printf("\n\nNow enter the amount to convert (as whole or decimal
value): ");
     scanf("%s", amt);   // Take the input as a string (char array).

     amount = CheckErrors(amt); // Pass to error-checking function.

     printf("\n\nThe string converted to a float number is: %f .\n\n",
amount);

     //Perform the calculations based on user's choice.
     switch(selection)
     {
               case 1 : result = amount * EUROSto;
                        printf("\n%f US dollars is %f EUROS.", amount,
result);
                        break;

               case 2 : result = amount * RUBLESto;
                        printf("\n%f US dollars is %f RUBLES.",
amount, result);
                        break;

               case 3 : result = amount * PESOSto;
                        printf("\n%f US dollars is %f PESOS.", amount,
result);
                        break;

               case 4 : result = amount * RUPEESto;
                        printf("\n%f US dollars is %f RUPEES.",
amount, result);
                        break;

               case 5 : result = amount * YENto;
                        printf("\n%f US dollars is %f YEN.", amount,
result);
                        break;

               case 6 : result = amount * POUNDSto;
                        printf("\n%f US dollars is %f POUNDS.",
amount, result);
                        break;

               case 7 : result = amount * EUROSfrom;
                        printf("\n%f EUROS is %f US dollars.", amount,
result);
                        break;

               case 8 : result = amount * RUBLESfrom;
                        printf("\n%f RUBLES is %f US dollars.",
amount, result);
                        break;

               case 9 : result = amount * PESOSfrom;
```

```
                            printf("\n%f PESOS is %f US dollars.", amount,
result);
                            break;

                case 10 : result = amount * RUPEESfrom;
                            printf("\n%f RUPEES is %f US dollars.",
amount, result);
                            break;

                case 11 : result = amount * YENfrom;
                            printf("\n%f YEN is %f US dollars.", amount,
result);
                            break;

                case 12 : result = amount * POUNDSfrom;
                            printf("\n%f POUNDS is %f US dollars.",
amount, result);
                            break;

                default : printf("Invalid choice.");
                            break;

        } //close switch

} //close function

//--------------------------------------------------------------------------
-----------------------------

void DisplayChart()
{
        //Just display a simple chart with today's exchange rates.
        printf("\n\t**************** Currency Rates *************");
        printf("\n\t*                                           *");
        printf("\n\t*     1.  1 US Dollar = .8040 EUROS.        *");
        printf("\n\t*     2.  1 US Dollar = 24.00884 RUBLES.    *");
        printf("\n\t*     3.  1 US Dollar = 11.4155 PESOS.      *");
        printf("\n\t*     4.  1 US Dollar = 45.9137 RUPEES.     *");
        printf("\n\t*     5.  1 US Dollar = 108.778 YEN.        *");
        printf("\n\t*     6.  1 US Dollar = .5343 POUNDS.       *");
        printf("\n\t*                                           *");
        printf("\n\t*     7.  EUROS = 1.2438 US Dollars.        *");
        printf("\n\t*     8.  RUBLES = .0344 US Dollars.        *");
        printf("\n\t*     9.  PESOS = .0876 US Dollars.         *");
        printf("\n\t*     10. RUPEES = .0218 US Dollars.        *");
        printf("\n\t*     11. YEN to = .0092 Dollars.           *");
        printf("\n\t*     12. POUNDS = 1.8715 US Dollars.       *");
        printf("\n\t*                                           *");
        printf("\n\t*********************************************\n\n");

}

//--------------------------------------------------------------------------
-----------------------------

void main()
```

```
{
     int RunProgram = 100;
     char choice = 'z';

     printf("\nCurrency Converter v. 3.0 - 2004 C. Germany - Team A
Software!!");

     //Lock program into a loop so it will keep running until user
selects quit.
     while(RunProgram != 0)
     {
                printf("\n\nMENU - Select an option\n\n");
                printf("\n\tQ = Quit");
                printf("\n\tC = Convert Curency");
                printf("\n\tD = Display Chart");
                printf("\n\n\tEnter your choice: ");

                scanf("%s", &choice);

               //Main menu choices
                switch(tolower(choice))
                {

                      case 'q'   :  RunProgram = 0;
                                        break;
                      case 'c'   :  ConvertCurrency(); //Call conversion
menu
                                        break;
                      case 'd'   :  DisplayChart();
                                        break;
                      default :     printf("Sorry, invalid choice.");
                                        break;

            } //close switch statement

        } //close while true loop on choice

         //User has chosen to quit - display exit message.
        printf("\nYou have choosen to quit.\nEnding calculator program.
Exiting...\n\n");

} // close main
```

Flowchart for Program:


START
Initialize Globals.
Display Main Menu Options:
0 = Quit
1 = Convert Currency
2 = Display Currency
Get User Menu Choice Via Integer
Currency Converter 3.0
2004 – C. Germany
QUIT
CONVERT CURRENCY
DISPLAY TABLE
0
1
2
END PROGRAM
ASK USER WHAT TO CONVERT
DISPLAY TABLE OF CURRENCY RATES
Store the Conversion Type For Use Later
ASK USER HOW MUCH TO CONVERT
NO
YES
ASSIGN DEFAULT VALUE OF 0
IS THE INPUT VALID?
STORE THE RESULT
CALCULATE AND DISPLAY THE RESULT BASED ON THE USER'S CHOICE AND AMOUNT

Number Guessing Game

Download: NumGame.exe

```
#include <stdio.h>
//--------------------------------------------------------------------------
----------------------------------

int RollEmBaby()
{
    //Yeah, I know it's pseudo random, but good enough for our purposes
...
    int HighNumber;
    int LowNumber;
    int RandomResult;

    HighNumber = 12;
    LowNumber = 2;

    srand(time(NULL));
    RandomResult = (rand()%HighNumber) + LowNumber;
    return RandomResult;
}
//--------------------------------------------------------------------------
----------------------------------

void main(void)
 {
       int ComputersNumber;
       int UsersChoice;
       ComputersNumber = RollEmBaby();
      printf("There are two ways to win, and both depend\n");
      printf("on whether or not lady luck shines.\n\n");
      printf("1. If you guess the number, you win.  For this,\n");
      printf("you have a snow ball\'s chance in Hell! :) \n\n");
      printf("2. At slightly better odds, your second chance\n");
      printf("is if you roll snake eyes or double sixes!\n\n");
      printf("Now that\'s a pretty fair deal, right?");
      printf("\n\n What number do you guess? ");

      scanf("%d", &UsersChoice);
      printf("\n\nThe number rolled was: %d .", ComputersNumber);
      if(UsersChoice == ComputersNumber)
      {
               printf("\n\nYou did it!  Unbelieveable luck!  You
win!\n\n");
       }
     if(ComputersNumber == 2)
     {  printf("\n\nYou win!  Snake Eyes!  Yeah!");  }

   if(ComputersNumber >2 && ComputersNumber < 12)
   {
        if(ComputersNumber != UsersChoice)
        {
           printf("\n\nSorry . . . You loose.\n\n");
        }
```

```
   }
   if(ComputersNumber == 12)
   {  printf("\n\nYou win!  Double sixes!!!"); }

 } // close main()
```

```
//Version 2.0
#include <stdio.h>
void Greet()
 {
   char TheirName[20];
   printf("Hi!  What\'s your name?\t");
   scanf("%s", TheirName);
   printf("\n\nHello, %s, I hope you are having a nice day!\n",
TheirName);
 }

int LuckyNumber(int low, int high)
{
    int PlayerGuess;
    int Lucky;

    srand(time(NULL));
    Lucky = (rand()%low) + high;
    return Lucky;
}


void main(void)
 {
     int MagicNumber;
     int PlayerGuess;
     Greet();
     MagicNumber = LuckyNumber(1, 6);
     printf("\nWhat\'s your best guess? ");
     scanf("%i", &PlayerGuess);
     printf("\n\nReceived %d from player.\n\n");
         if(PlayerGuess == MagicNumber)
         {
            printf("\n\nYou did it! You guessed my lucky number!\n\n");
         }
         else
         {
            printf("\n\nSorry, you did not guess it.\n\n");
            printf("The number was %d", MagicNumber);
         }
 }
```

Multidimensional Arrays of Type Char and Command Line Arguments

```
Download:  CommandLine.exe
#include <stdio.h>
#include <ctype.h>
```

```
int main(int argc, char * argv[])
{
     //2004 - C. Germany. The string arguments passed to the command
line are
     //actually multi-dimensional arrays, so we need nested for loops.

   char * tempstring;
   printf("I received %d arguments!\n", argc);
   for(int i=0; i<argc; i++)
   {
          printf("\nArgument %d:", i);
          tempstring = argv[i];
  printf("You typed: \"");
  printf(tempstring);
  printf("\". In capital letters that\'s \"");
  for(int z = 0; tempstring[z]; z++)
      putchar(toupper(tempstring[z]));
  printf("\".");
   }
    printf("\n\n");
    char * t1;
    t1 = "the end";
    for(int x=0; t1[x]; x++)
        putchar(toupper(t1[x]));
    printf("\n\n\n");
    return 0;
}
```

Multidimensional Arrays to Save Unnecessary Lines of Code

Download: MultiArray.exe

```
//2004 - C. Germany
// Parsing a string for integer errors without isdigit() and cctype

#include <iostream>

void main(void)
{
   //Take three numbers as a multi-dimensional array of strings
   char FavNumber[3][10];
   int Num[3] = {0, 0, 0};  //initialize to NULL
   int x = 0;
   int count;

   printf("Enter your three favorite numbers: \n\n");
for(count = 0; count<3; count++)
{

   //scanf("%d%d%d", &a, &b, &c);
   printf("Enter favorite number %d: ", count+1); //add 1 for fence post
   scanf("%s", FavNumber[count]);
   //Need to convert the ASCII value character array to integer values
```

```
    x = 0; //reset x to 0 for each loop
    while(  (FavNumber[count][x] - 48) < 10 && (FavNumber[count][x] -
48) > 0  )
{
    Num[count] = 10 * Num[count] + (FavNumber[count][x]
- 48);
        x++;
}

    if(!(FavNumber[count][x] - 48) < 10 && (FavNumber[count][x] - 48) >
0)
{   //offset for fencepost - x was incremented 1 past
    printf("\n\nHey, not fair! That was not a plain old number!\n\n");
        printf("I will set this garbledygunk number to 0 (NULL)!\n\n");
        Num[count] = 0;
}
} //close the for loop


for(count = 0; count<3; count++)
{
   printf("\nFavorite number %d as a string values is: %s .",
(count+1),FavNumber[count]);
   printf("\nFavorite number %d as an integer value is: %d \n",
(count+1), Num[count]);
} //close for loop
} //close main()
```

```
//2004 C. Germany - Accounting for User Error
#include <iostream>
#include <cctype>
using namespace std;

void main(void)
{
   //Take three numbers as a multi-dimensional array of stirngs
   char FavNumber[3][10];
   int Num[3] = {0, 0, 0};  //initialize to NULL
   int x = 0;
   int count;

   printf("Enter your three favorite numbers: \n\n");
for(count = 0; count<3; count++)
{

   //scanf("%d%d%d", &a, &b, &c);
   printf("Enter favorite number %d: ", count+1); //add 1 for fence post
   scanf("%s", FavNumber[count]);
   //Need to convert the ASCII value character array to integer values
    x = 0; //reset x to 0 for each loop
    while(isdigit(FavNumber[count][x]))
{
         Num[count] = 10 * Num[count] + (FavNumber[count][x] - 48);
     x++;
}
```

```
    if(!isdigit(FavNumber[count][x-1]))
{   //offset for fencepost - x was incremented 1 past
    printf("\n\nHey, not fair! That was not a plain old
number!\n\n");
        printf("I will set this garbledygunk number to 0 (NULL)!");
Num[count] = 0;
}
} //close the for loop


for(count = 0; count<3; count++)
{
   printf("\nFavorite number %d as a string values is: %s .",
(count+1),FavNumber[count]);
   printf("\nFavorite number %d as an integer value is: %d \n",
(count+1), Num[count]);
} //close for loop
} //close main()
```

Arrays and Loops

```
Download:  Alphabet.exe
// C Program Example - C. Germany
// Implement the following program using a while loop instead of a for
loop

#include <stdio.h>

void main(void)
 {
   int counter;
   char Alphabet[27] = "abcdefghijklmnopqrstuvwxyz";

   //Alphabet forwards
   printf("This is for my son Jacob who can now get to A-B-C-D-E!\n\n");
   printf("Here\'s the phonetic alphabet forwards, Jacob:\n\n ");

   counter = 0;

   while(counter <= 26)
   {
    printf("%c", Alphabet[counter]);
    counter++;
   }

   printf("\n\n");
   //Alphabet backwards
   printf("\n\nHere\'s the phonetic alphabet backwards:\n\n ");

   counter = 26;

   while(counter >= 0)
   {
```

```
    printf("%c", Alphabet[counter]);
    counter--;
   }

 }
```

```
// Same behavior but using for loops instead of while true
#include <stdio.h>
void main(void)
 {
   int counter;
   char Alphabet[27] = "abcdefghijklmnopqrstuvwxyz";

   //Alphabet forwards
   printf("This is for my son Jacob who can now get to A-B-C-D-E!\n\n");
   printf("Here\'s the phonetic alphabet forwards, Jacob:\n\n ");
   for (counter = 0; counter <= 26; counter++)
    printf("%c", Alphabet[counter]);

   printf("\n\n");
   //Alphabet backwards
   printf("\n\nHere\'s the phonetic alphabet backwards:\n\n ");
   for (counter = 26; counter >= 0; counter--)
    printf("%c", Alphabet[counter]);
 }
```

Reversing For Loops

```
Download:  ReverseLoop.exe
#include <stdio.h>
void main(void)
 {
   int counter;

   for (counter = 1; counter <= 5; counter++)
    printf("%d ", counter);

   printf("\nStarting second loop\n");

   for (counter = 1; counter <= 10; counter++)
    printf("%d ", counter);

   printf("\nStarting third loop\n");

   for (counter = 0; counter <= 5; counter++)
    printf("%d ", counter);
 }
```

Pointers

```
Download:  Pointers.exe
```

```
#include <stdio.h>
void main(void)
 {
   int NumOne = 1;
   int NumTwo = 2;

   int * IAmAPointerToAnInteger;

   // Assign pointer to an address
   IAmAPointerToAnInteger = &NumOne;

   // Change the value pointed to by IAmAPointerToAnInteger to 5
   *IAmAPointerToAnInteger = 5;

   // Display the value
   printf("The value pointed to by IAmAPointerToAnInteger is %d . The
variable NumOne is %d\n",
     *IAmAPointerToAnInteger, NumOne);

 }
```

Temperature Conversion

Download: Temperature.exe

```
//2004 C. Germany - Convert Temperatures

#include <stdio.h>

void ConvertTemp()
{
     //Note: "to" = units to 1 US Dollar
     //       "from" = 1 US Dollar to units

    int selection;
    float temperature, result;


    printf("\nChoose:");
    printf("\n1 = Convert Farenheit to Celcius.");
    printf("\n2 = Convert Celcius to Farenheit.");


    printf("\n\nYour choice: ");
    scanf("%d", &selection);

    printf("\n\nNow enter the temperature to convert: ");
    scanf("%f", &temperature);

       switch(selection)
       {
```

```
        //Convert Farenheit to Celcius
        case 1 : result = ( (temperature - 32) * 5 ) / 9;
                printf("\n %f degrees farenheit is %f degrees celcius.", temperature, result);
                break;


        //Convert Celcius to Farenheit
        case 2 : result = (temperature * 9 / 5) + 32;
                printf("\n %f degrees celcius is %f degrees farenheit.", temperature, result);
                break;


        default : printf("Invalid choice.");
                 break;


       } //close switch

} //close function

//-------------------------------------------------------------------------------------------------------------------
---------

void main()
{
    int RunProgram = 100;
    int choice = 100;

    printf("\nTempertature Converter - 2004 C. Germany - Team A Software!!");

    while(RunProgram != 0)
    {
          printf("\n\nMENU - Select an option\n\n");
          printf("\n0 = Quit");
          printf("\n1 = Convert Temperature");
;
          printf("\n\nEnter your choice: ");

          scanf("%d", &choice);

          switch(choice)
          {

                case 0 : RunProgram = 0;
                        break;
```

```
            case 1 : ConvertTemp();
                     break;


            default : printf("Sorry, invalid choice.");
                      break;

         } //close switch statement

      } //close while true loop on choice

      printf("\nYou have choosen to quit.\nEnding temperature conversion
program.  Exiting...\n\n");

} // close main
```

How Many Lines?

Download: Lines.exe

```
#include <stdio.h>

void main(void)
 {
      int NumTimes;
      int x;
      printf("How many lines woud you like?  ");
      scanf("%d", &NumTimes);
      //NumTimes = 4;
      printf("Received the number %d .", NumTimes);
      printf("\n\n");

   if(NumTimes < 500 && NumTimes >0)
   {
       for(x = 0; x < NumTimes; x++)
       {
           printf("This is line number ");
           //Need to adjust the offset for the fencepost error
           printf("%d", x+1);
           printf(".\n");
       }
   }
   else
   {
      if(NumTimes <= 0)
      {
        printf("Printing 0 or less lines is pretty pointless.");
      }

      else
      {
```

```
           printf("More than 500?!? That is too many lines!  We don\'t
have all day!");
      }
   }
   printf ("\n\n");
   printf ("The End");
   printf ("\n\n");
 } // close main()
```

Download: Lines2.exe

```
#include <stdio.h>
void main()
 {
   int NumTimes;
   int x;
   int StopIt;
   StopIt = 0;  //false

   while(StopIt != 1)
   {
        printf("How many lines woud you like?  ");
        scanf("%d", &NumTimes);
        //NumTimes = 4;
        printf("Received the number %d .", NumTimes);
        printf("\n\n");

        switch(NumTimes)
        {
          case 0 : printf("Too few lines to print!\n");
                   StopIt = 1;  //true
                   break;
          case 1 : printf("A line!\n");
                   StopIt = 1;  //true
                   break;
          case 2 : printf("A line!\nA line!\n");
                   StopIt = 1;  //true
                   break;
          case 3 : printf("A line!\nA line!\nA line!\n");
                   StopIt = 1;  //true
                   break;
          default : printf("That is not an option!\n");
                    break;
        } // close switch

  } // close while true loop
   printf("\n\n");
   printf("The End");
   printf("\n \n");
 } // close main() function
```

Array of Strings

Download: StringArray.exe

```
#include <stdio.h>
char * GetName()
{
     char pName[20];
     printf("Please tell me your name: ");
     scanf("%s", pName);
     printf("Received the string: ");
     printf(pName);
     printf(" while inside the GetName function.");
     printf("\nNow exiting the GetName() function.");
     return pName;
}

void main(void)
{
   int x;
   char * Messages[3] = {"C++ ", "is very close ", "to C.\n\n"};
   char * TheName;

   TheName = GetName();

   printf("\n\n");
   printf(TheName);
   printf(" , the 3 messages are: ");

   for(x = 0; x <3; x++)
   {
      printf(Messages[x]);
   }

   printf("\n\nThe End");

 }
```

Detecting Numbers With IsDigit()

```
#include <stdio.h>
#include <cctype.h>
void main(void)
{
   char FavNumber[10];
   int Num, x;
   int a, b, c;
   x = 0;
   printf("Enter your three favorite numbers: ");
   //scanf("%d%d%d", &a, &b, &c);
   scanf("%s", FavNumber);
   printf("Your three favorite numbers as string values are: %s \n\n",
FavNumber);
//Need to convert the ASCII value character array to integer values
```

```
while(isdigit(FavNumber[x]))
{
     if(!isdigit(FavNumber[x]))
     {
        printf("\n\nHey!  That's not a number! Invalid.\n\n");
        break;
     } //close if
     else
     {
        Num = 10 * Num + (FavNumber[x] - 48);
        x++;
     } //close else

} //close while true loop


   printf("Your favorite numbers as integers are: %d \n", Num);
}
```

Even Numbers

```
include <stdio.h>

void main(void)
 {
   int remainder;
   int result;
   int x;

   printf("Counting even numbers from 0 - 100:\n\n");

   for(x=0; x<=100; x++)
   {
      result = x % 2;

      if(result == 0)
      {
         printf("\t");
         printf("%d", x);
      }
   }
 }
```