GUI Programming Using The WIN32 API

Shuquan Li

February 24, 2014

ECT362 Homework #2

Introduction:

This project is desired to develop a GUI to read context information from a file automatically; add new contact to the viewable list and local array; delete the context information which the user selected from both the viewable list and the local array; updates all the changes to the original file. It focuses on enhancing the programming syntax with Windows GUI programming.

Design and Analysis:

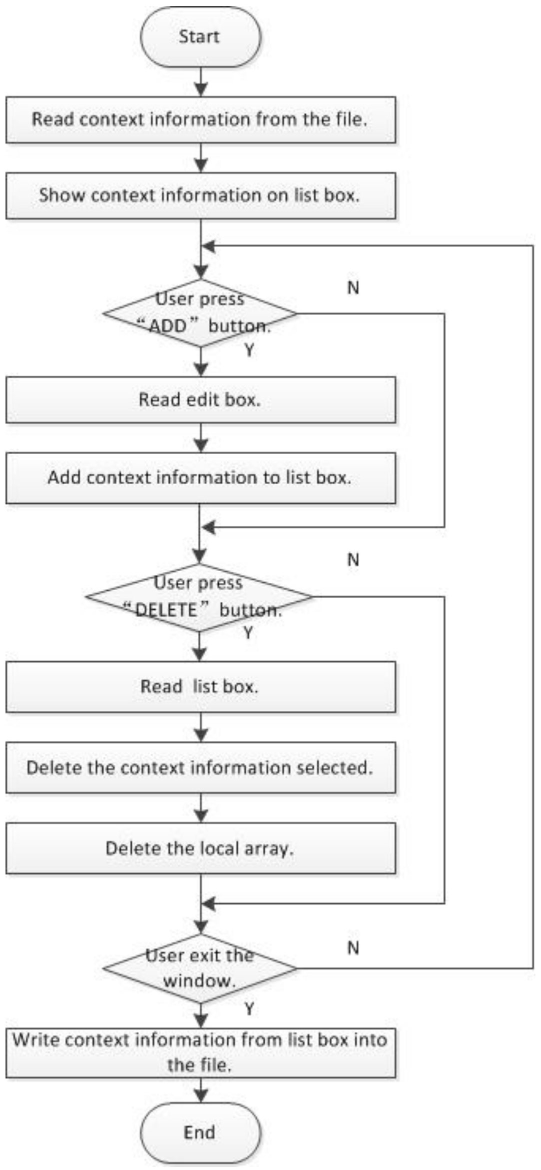
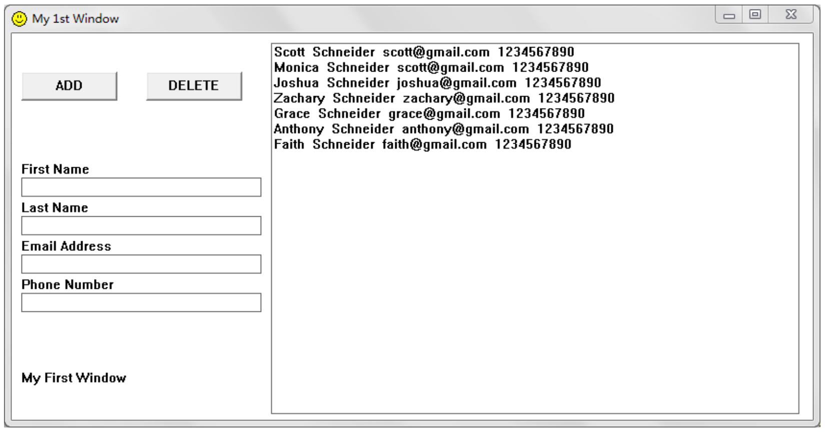

Start
Read context information from the file.
Show context information on list box.
User press "ADD" button.
N
Y
Read edit box.
Add context information to list box.
User press "DELETE" button.
N
Y
Read list box.
Delete the context information selected.
Delete the local array.
User exit the window.
N
Y
Write context information from list box into the file.
End

Figure 1: Flow char of the program design.

The flow chart, shown in Figure 1, indicates what the program do when the user run the program. First, the program reads the context information from the file and shows them on the list box. Then the program reads command, whether the user presses the “ADD” button; if yes, it will read what user typed into the edit box and then show them on the list box; if no, it will run the next step. The program reads whether the user presses the “DELETE” button next step; if yes, it will read what context information the user selected from the list box, then it will delete it both from the list box and the local array, if no, it will run the next step. Next step, the program reads whether the user exits the window or not. If the user exits the window, it will write the context information from the list box into the file then end the program; if no, it will read the “ADD” command and do the following step again.

In order to delete the context information which the user wants, “LB_GETCURSEL” is used to get the index which the user selected from the list box, and then “LB_DELETESTRING” will delete the line of the delete index. As “in_cnt” is desired to be used into all the functions and wished to be found next time, it is put into “Global variables”. The local array needs to be deleted when deleting the context information from the list box, all the context information after the deleting address need to be overwritten. The file open code is put into “case WM_CREATE” because it is desired to read context information from the file as soon as the program is run. And the code of write to file is put into “case WM_DESTORY” in order to update all the changes into the file when the program finished.

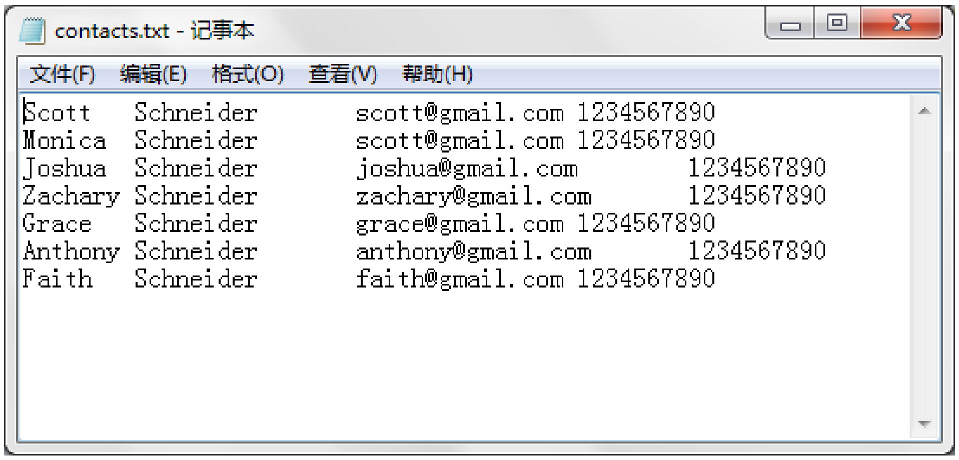
The front panel of this project is shown in Figure 2. Because the context information has first name, last name, email address, and phone number, four edit boxes are needed. If the width of the edit box is not large enough, the whole email address can’t be typed in, so they also need to be enlarged.



```
Scott  Schneider  scott@gmail.com  1234567890
Monica  Schneider  scott@gmail.com  1234567890
Joshua  Schneider  joshua@gmail.com  1234567890
Zachary  Schneider  zachary@gmail.com  1234567890
Grace  Schneider  grace@gmail.com  1234567890
Anthony  Schneider  anthony@gmail.com  1234567890
Faith  Schneider  faith@gmail.com  1234567890
```

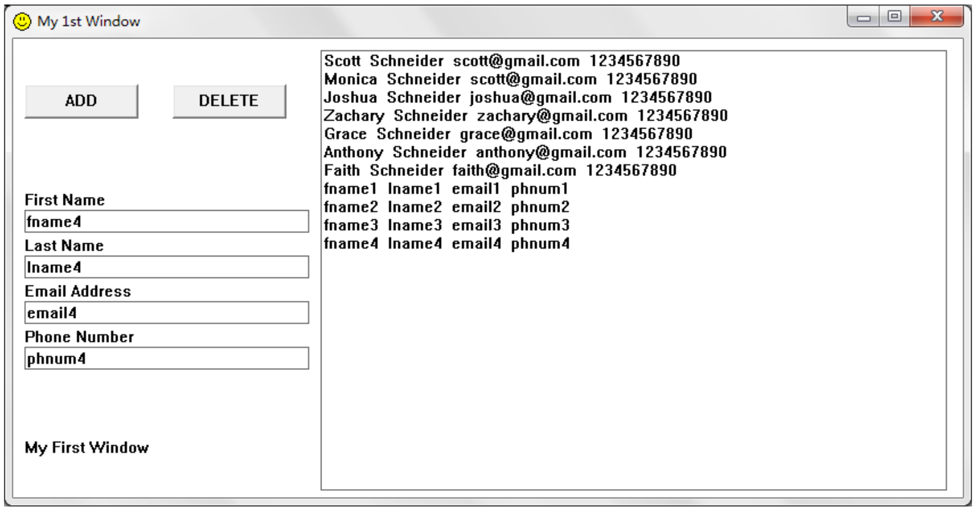
Figure 2: The front panel of the project.

The original file is shown in Figure 3. When “ADD” button is pressed, the information in the edit box will be read and then added to the list box, which is shown in Figure 4. When the window is destroyed, it is also added into file, which is shown in Figure 5.



```
Scott   Schneider       scott@gmail.com 1234567890
Monica  Schneider       scott@gmail.com 1234567890
Joshua  Schneider       joshua@gmail.com        1234567890
Zachary Schneider       zachary@gmail.com       1234567890
Grace   Schneider       grace@gmail.com 1234567890
Anthony Schneider       anthony@gmail.com       1234567890
Faith   Schneider       faith@gmail.com 1234567890
```

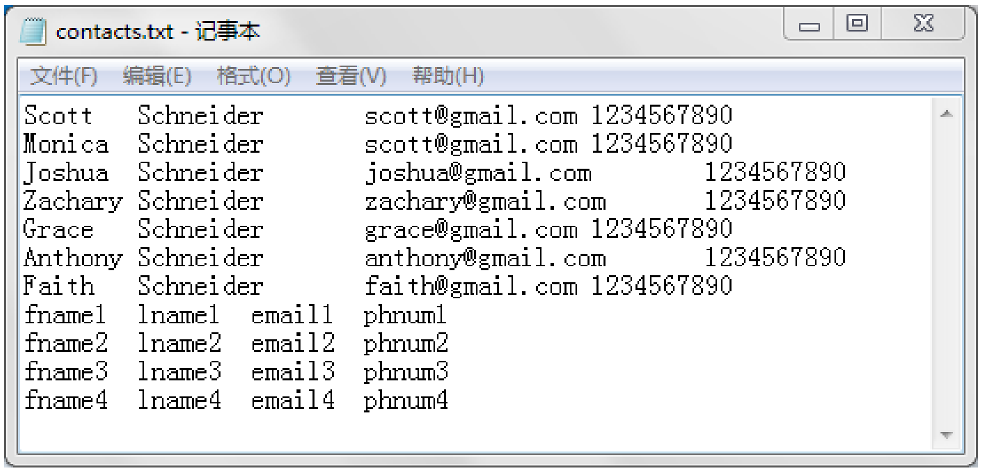
Figure 3: The original file.



First Name: fname4
Last Name: lname4
Email Address: email4
Phone Number: phnum4

```
Scott  Schneider  scott@gmail.com  1234567890
Monica  Schneider  scott@gmail.com  1234567890
Joshua  Schneider  joshua@gmail.com  1234567890
Zachary  Schneider  zachary@gmail.com  1234567890
Grace  Schneider  grace@gmail.com  1234567890
Anthony  Schneider  anthony@gmail.com  1234567890
Faith  Schneider  faith@gmail.com  1234567890
fname1  lname1  email1  phnum1
fname2  lname2  email2  phnum2
fname3  lname3  email3  phnum3
fname4  lname4  email4  phnum4
```

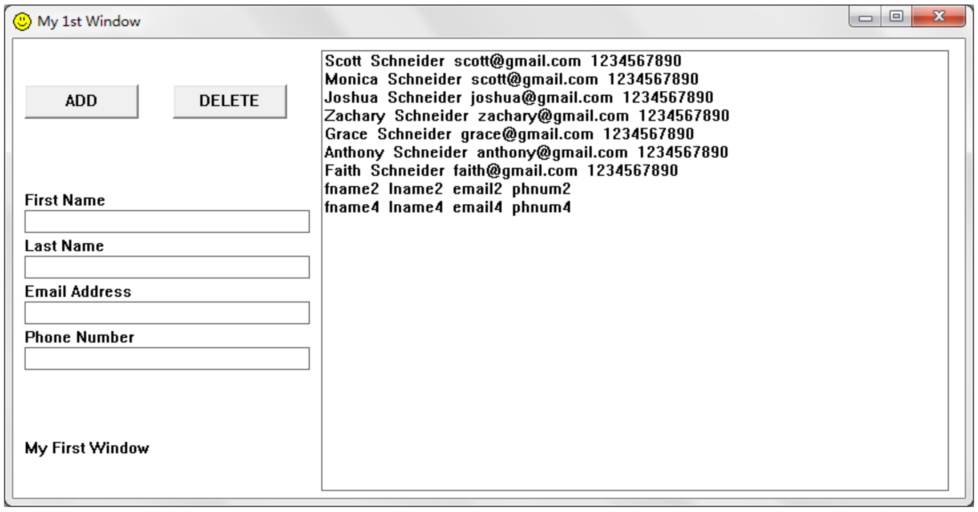
Figure 4: The front panel when “ADD” button is pressed.



```
Scott   Schneider       scott@gmail.com 1234567890
Monica  Schneider       scott@gmail.com 1234567890
Joshua  Schneider       joshua@gmail.com        1234567890
Zachary Schneider       zachary@gmail.com       1234567890
Grace   Schneider       grace@gmail.com 1234567890
Anthony Schneider       anthony@gmail.com       1234567890
Faith   Schneider       faith@gmail.com 1234567890
fname1  lname1  email1  phnum1
fname2  lname2  email2  phnum2
fname3  lname3  email3  phnum3
fname4  lname4  email4  phnum4
```

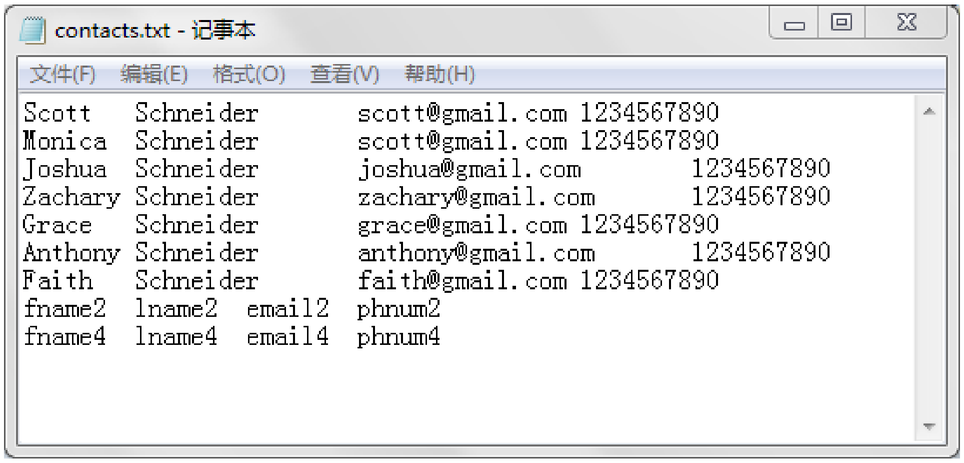
Figure 5: The file when "ADD" button is pressed.

When "DELETE" button is pressed, the information in the edit box will be deleted and then removed from the list box, which is shown in Figure 6. When the window is destroyed, it is also removed from the file, which is shown in Figure 7.



```
Scott  Schneider  scott@gmail.com  1234567890
Monica  Schneider  scott@gmail.com  1234567890
Joshua  Schneider  joshua@gmail.com  1234567890
Zachary  Schneider  zachary@gmail.com  1234567890
Grace  Schneider  grace@gmail.com  1234567890
Anthony  Schneider  anthony@gmail.com  1234567890
Faith  Schneider  faith@gmail.com  1234567890
fname2  lname2  email2  phnum2
fname4  lname4  email4  phnum4
```

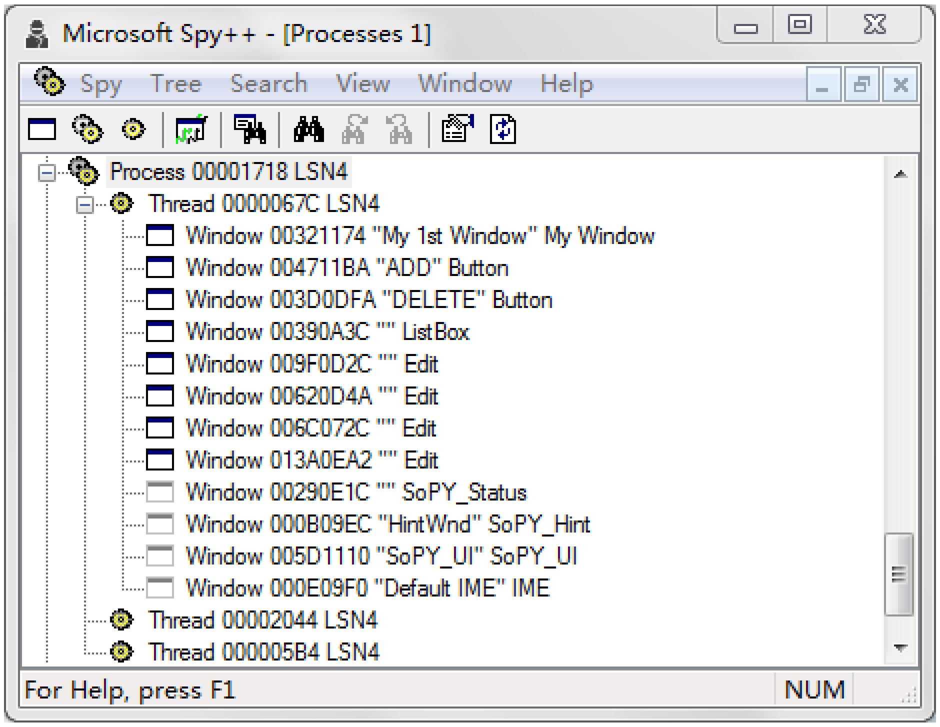
Figure 6: The front panel when "DELETE" button is pressed.



```
Scott   Schneider       scott@gmail.com 1234567890
Monica  Schneider       scott@gmail.com 1234567890
Joshua  Schneider       joshua@gmail.com        1234567890
Zachary Schneider       zachary@gmail.com       1234567890
Grace   Schneider       grace@gmail.com 1234567890
Anthony Schneider       anthony@gmail.com       1234567890
Faith   Schneider       faith@gmail.com 1234567890
fname2  lname2  email2  phnum2
fname4  lname4  email4  phnum4
```

Figure 7: The file when “DELETE” button is pressed.

Figure 8 shows this program has a main window named “My 1st window”, two button “ADD” and “DELETE”, a list box, four edit box.


Microsoft Spy++ - [Processes 1]
Spy Tree Search View Window Help
For Help, press F1
NUM

```
Process 00001718 LSN4
  Thread 0000067C LSN4
    Window 00321174 "My 1st Window" My Window
    Window 004711BA "ADD" Button
    Window 003D0DFA "DELETE" Button
    Window 00390A3C "" ListBox
    Window 009F0D2C "" Edit
    Window 00620D4A "" Edit
    Window 006C072C "" Edit
    Window 013A0EA2 "" Edit
    Window 00290E1C "" SoPY_Status
    Window 000B09EC "HintWnd" SoPY_Hint
    Window 005D1110 "SoPY_UI" SoPY_UI
    Window 000E09F0 "Default IME" IME
  Thread 00002044 LSN4
  Thread 000005B4 LSN4
```

Figure 8: The screen capture of spy++.

Conclusion

It is desired to use a GUI to meet several requirements, such as add and delete the list box, local array and the file; read the file and write to file.

In doing this project, the biggest difficulty is the delete commend. It needs to get the delete index and overwrite the context information after the index. The language is not familiar; the programming has many problems during this time. After finishing this project, many problems were not understood clearly from class have been solved.

References

Windows Messages. Microsoft Developer Network. nd. np. Web. Feb.24, 2014

SendDlgItemMessage function. Microsoft Windows Dev Center - Desktop. nd. np. Web. Feb.24, 2014

List Box Messages. Microsoft Windows Dev Center - Desktop. nd. np. Web. Feb.24, 2014

LB_GETCURSLE Message. Microsoft Windows Dev Center - Desktop. nd. np. Web. Feb.24, 2014

Appendix

```
/* Preprocessor Directives */
#include <windows.h>  // Win32 API
#include "resource.h"  // resources for your application

#include <iostream>  // cin, cout, endl
#include <string.h>  // strcpy()
#include <stdio.h>  // fgets()
#define ROWS 100
#define COLS 100

using namespace std;  // using the standard namespace

struct contact
{
	char fnames[100];  // array to hold first name
	char lnames[100];  // array to hold last name
	char emails[100];  // array to hold last name
	char phnums[100];  // array to hold last name
};


#define ID_ADDBUTTON 1001
#define ID_LIST                   1010
#define ID_EDIT                   1020
#define ID_DELEBUTTON      1030

/* Function Prototypes */
LRESULT CALLBACK WndProc( HWND, UINT, WPARAM, LPARAM );  //the window procedure
/* A callback function is passed (by reference) to another function */

/* Global variables */
contact contacts[100];
int in_cnt=0;

/* Functions */
```

```
/******************************************************************
 * Function    | WinMain()
 * Description | Entry point for our application, we create and
 *             |  register a window class and then call CreateWindow
 * Inputs      | None
 * Output      | Integer value 0
 ******************************************************************/
int WINAPI WinMain( HINSTANCE hInstance,
                                HINSTANCE hPrevInstance,
                    LPSTR szCmdLine,
                                   int iCmdShow)
{
       static char szAppName[] = "My Window";

       HWND        hwnd, lhWnd;
       WNDCLASSEX  wndclass;  // This is our new windows class
       MSG msg;

       /*  Fill in WNDCLASSEX struct members  */
       wndclass.cbSize         = sizeof(wndclass);
       wndclass.style          = CS_HREDRAW | CS_VREDRAW;
       wndclass.lpfnWndProc    = WndProc;
       wndclass.cbClsExtra     = 0;
       wndclass.cbWndExtra     = 0;
       wndclass.hInstance      = hInstance;
       wndclass.hIcon          = LoadIcon(GetModuleHandle(NULL),
                                 MAKEINTRESOURCE(IDI_MYICON));
       wndclass.hIconSm        = (HICON)LoadImage(GetModuleHandle(NULL),
                                 MAKEINTRESOURCE(IDI_MYICON), IMAGE_ICON, 16, 16, 0);
       wndclass.hCursor        = LoadCursor(NULL, IDC_ARROW);
       wndclass.hbrBackground  = (HBRUSH) GetStockObject(WHITE_BRUSH);
       wndclass.lpszClassName  = szAppName;
       wndclass.lpszMenuName   = NULL;

    /*  Register a new window class with Windows  */
    RegisterClassEx(&wndclass);

    /*  Create a window based on our new class  */
    hwnd = CreateWindow(szAppName,  // class name the window is to be based on
                        "My 1st Window",  // the title of the window that will
                                          //  appear in the bar at the top
                        WS_OVERLAPPEDWINDOW,  // window style (a window that
                                              //  has a caption, a system menu,
                                              //  a thick frame and a minimise
                                              //  and maximise box)
                     /* Use default starting location for window */
                     CW_USEDEFAULT, // initial x position (top left corner)
                     CW_USEDEFAULT, // initial y position (top left corner)
                     850, // initial width
                     440, // initial height
                     NULL, // window parent (NULL for not a child window)
                     NULL, // menu (NULL to use class menu)
                     hInstance, // the program instance passed to us
                     NULL);  // pointer to any parameters wished to be
                            //  passed to the window producer when the window
                            //  is created

       if(hwnd == NULL)  // check to see if an error occurred in creating the window
    {
        MessageBox(NULL, "Window Creation Failed!", "Error!",
            MB_ICONEXCLAMATION | MB_OK);
        return 0;
    }

    /* Show and update our window  */
    ShowWindow(hwnd, iCmdShow);
    UpdateWindow(hwnd);


    /*  Retrieve and process any queued messages until we get WM_QUIT  */
    /* Recall that Windows uses a messaging system to notify window of */
    /*  user actions
          */
    while ( GetMessage(&msg, NULL, 0, 0) )
    {
       TranslateMessage(&msg);  // for certain keyboard messages
       DispatchMessage(&msg);  // send message to WndProc
    }

    /*  Exit with status specified in WM_QUIT message  */
    return msg.wParam;
} // end WinMain()


/*******************************************************************
 * Function    | WinProc()
 * Description | Whenever anything happens to your window, Windows
 *             |  will call this function telling you what has happened.
 *             |  The message parameter contains the message sent
 * Inputs      | None
 * Output      | Integer value 0
 *******************************************************************/
LRESULT CALLBACK WndProc( HWND hwnd,
                          UINT iMsg,
                          WPARAM wParam,
                          LPARAM lParam)
{
    PAINTSTRUCT ps;
       HDC hdc;
       static HWND  addbutton, deletebutton, listbox, editbox_fnames, editbox_lnames,
              editbox_emails, editbox_phnums;

       int len;
       char line[COLS];
       int out_cnt;
       int del_idx;

       FILE *file_ptr;

       /*  Switch according to what type of message we have received  */
       switch ( iMsg )
       {
              case WM_PAINT:
              /* We receive WM_PAINT every time window is updated  */
                     hdc = BeginPaint(hwnd, &ps);
                     TextOut(hdc, 10, 350, "My First Window", 15);
                     TextOut(hdc, 10, 133, "First Name", 10);
                     TextOut(hdc, 10, 173, "Last Name", 9);
                     TextOut(hdc, 10, 213, "Email Address", 13);
                     TextOut(hdc, 10, 253, "Phone Number", 12);

                     EndPaint(hwnd, &ps);
                     break;
              case WM_CREATE:
              /* Operations to be performed whenthis window is created */
                     /* Create a child window for a pushbutton */
                     addbutton = CreateWindow(  "BUTTON",  // predefined class
                                  "ADD",  // button text
                                  WS_VISIBLE | WS_CHILD | BS_PUSHBUTTON,
                                  // Size and position values are given
                                  //  explicitly, because the CW_USEDEFAULT
                                  //  constant gives zero values for buttons.
                                  10,  // starting x position
                                  40,  // starting y position
                                  100, // button width
                                  30,  // button height
                                  hwnd, // parent window
                                  (HMENU)ID_ADDBUTTON, // no menu
                                  (HINSTANCE) 0,  // ignored for Windows XP
                                  NULL );  // pointer not needed

                     deletebutton = CreateWindow(  "BUTTON",  // predefined class
                                  "DELETE",
                                  WS_VISIBLE | WS_CHILD | BS_PUSHBUTTON,
                                  140,
                                  40,
                                  100,
                                  30,
                                  hwnd,
                                  (HMENU)ID_DELEBUTTON,
                                  (HINSTANCE) 0,
                                  NULL );  // pointer not needed


                     /* Create a list box to display values within the window */
                     listbox = CreateWindow( "LISTBOX",  // predefined class
                                  "",  // initial lst box text (empty)
                                  // automatically sort items in list box
                                  //  and include a vertical scroll bar
                                  //  with a border
                                  WS_CHILD | WS_VISIBLE | LBS_NOTIFY |
                                  WS_VSCROLL | WS_BORDER,
                                  270,  // starting x position
                                  10,  // starting y position
                                  550,  // list box width
                                  400,  // list box height
                                  hwnd,  // parent window
                                  (HMENU)ID_LIST,  // no menu
                                  (HINSTANCE) 0,  // ignored for Windows XP
                                  NULL ); // pointer not needed

                     /* Create edit box for adding and deleting items from list */
                     editbox_fnames = CreateWindow( "EDIT",  // predefined class
                                           "",  // initial lst box text (empty)
```

```
                                    // create an edit box with a border
                                    WS_CHILD | WS_VISIBLE | WS_BORDER,
                                    10,  // starting x position
                                    150,  // starting y position
                                    250,  // list box width
                                    20,  // list box height
                                    hwnd,  // parent window
                                    (HMENU)ID_EDIT,  // no menu
                                    (HINSTANCE) 0,  // ignored for Windows XP
                                    NULL); // pointer not needed

        editbox_lnames = CreateWindow( "EDIT",  // predefined class
                                   "",  // initial lst box text (empty)
                                    // create an edit box with a border
                                    WS_CHILD | WS_VISIBLE | WS_BORDER,
                                    10,  // starting x position
                                    190,  // starting y position
                                    250,  // list box width
                                    20,  // list box height
                                    hwnd,  // parent window
                                    (HMENU)ID_EDIT,  // no menu
                                    (HINSTANCE) 0,  // ignored for Windows XP
                                    NULL); // pointer not needed

        editbox_emails = CreateWindow( "EDIT",  // predefined class
                                    "",  // initial lst box text (empty)
                                    // create an edit box with a border
                                    WS_CHILD | WS_VISIBLE | WS_BORDER,
                                    10,  // starting x position
                                    230,  // starting y position
                                    250,  // list box width
                                    20,  // list box height
                                    hwnd,  // parent window
                                    (HMENU)ID_EDIT,  // no menu
                                    (HINSTANCE) 0,  // ignored for Windows XP
                                    NULL); // pointer not needed

        editbox_phnums = CreateWindow( "EDIT",  // predefined class
                                    "",  // initial lst box text (empty)
                                    // create an edit box with a border
                                    WS_CHILD | WS_VISIBLE | WS_BORDER,
                                    10,  // starting x position
                                    270,  // starting y position
                                    250,  // list box width
                                    20,  // list box height
                                    hwnd,  // parent window
                                    (HMENU)ID_EDIT,  // no menu
                                    (HINSTANCE) 0,  // ignored for Windows XP
                                    NULL); // pointer not needed

/* Read command line information */
file_ptr = fopen("contacts.txt", "r");  // open the contacts.txt file for
                               //  reading only
if( file_ptr == NULL ) // failed to open file
{
       MessageBox(NULL, "Error Opening File", "ERROR",
                        MB_ICONINFORMATION | MB_OK);
}
```

```
        /* Read file line-by-line, storing each line into a seperate row in */
        /*  the names[][] string array                                       */
        in_cnt;
        while( fgets( line, COLS, file_ptr) != NULL ) //no infromation i got from
                                                        file.
        {
               sscanf(line, "%s\t%s\t%s\t%s", contacts[in_cnt].fnames,
                     contacts[in_cnt].lnames, contacts[in_cnt].emails,
                     contacts[in_cnt].phnums);
               sprintf(line, "%s  %s  %s  %s", contacts[in_cnt].fnames,
                     contacts[in_cnt].lnames, contacts[in_cnt].emails,
                     contacts[in_cnt].phnums);
               SendDlgItemMessage( hwnd, ID_LIST,
                                           LB_ADDSTRING, 0,
                                          (LPARAM)line);
               in_cnt++;
        }
        fclose(file_ptr);  //close file since all data has been extracted
        break;
case WM_DESTROY:
/* Window has been destroyed, so exit cleanly  */

        /* Read command line information */
        file_ptr = fopen("contacts.txt", "w");  // open the contacts.txt file for
                                                   //  writing only
        if( file_ptr == NULL ) // failed to open file
        {
               MessageBox(NULL, "Error Opening File", "ERROR",
                                 MB_ICONINFORMATION | MB_OK);
        }

        /* create new contact line for file */
        for( out_cnt = 0; out_cnt < in_cnt; out_cnt++ )
        {
               strcpy(line, contacts[out_cnt].fnames);
               strcat(line, "\t");
               strcat(line, contacts[out_cnt].lnames);
               strcat(line, "\t");
               strcat(line, contacts[out_cnt].emails);
               strcat(line, "\t");
               strcat(line, contacts[out_cnt].phnums);
               strcat(line, "\n");

               /* Update file with added contacts */
               fputs(line, file_ptr);  // write new contact information to file
        }

        PostQuitMessage(0);

        break;
case WM_COMMAND:
/* User selected a command from a menu or a control sent a message */
     if (HIWORD(wParam) == BN_CLICKED)
     {
         switch (LOWORD(wParam))
         {
            case ID_ADDBUTTON:
```

```
                        MessageBox(NULL, "Add Button Pressed", "ADD",
                                   MB_ICONINFORMATION | MB_OK);
                        /* Read length of string entered into the text box */
                        len = GetWindowTextLength(editbox_fnames);
                        len = GetWindowTextLength(editbox_lnames);
                        len = GetWindowTextLength(editbox_emails);
                        len = GetWindowTextLength(editbox_phnums);
                        /* Read string from edit box and store to name[] */
                        GetWindowText(editbox_fnames, contacts[in_cnt].fnames, len + 20);
                        GetWindowText(editbox_lnames, contacts[in_cnt].lnames, len + 20);
                        GetWindowText(editbox_emails, contacts[in_cnt].emails, len + 20);
                        GetWindowText(editbox_phnums, contacts[in_cnt].phnums, len + 20);
                        sprintf(line, "%s  %s  %s  %s", contacts[in_cnt].fnames,
                                contacts[in_cnt].lnames, contacts[in_cnt].emails,
                                contacts[in_cnt].phnums);
                        in_cnt++;
                        SendDlgItemMessage( hwnd, ID_LIST,
                                            LB_ADDSTRING, 0,
                                             (LPARAM)line);
                        break;
                     case ID_DELEBUTTON:
                        MessageBox(NULL, "Delete Button Pressed", "DELETE",
                                   MB_ICONINFORMATION | MB_OK);

                        del_idx = SendDlgItemMessage( hwnd, ID_LIST,
                                                       LB_GETCURSEL,
                                                      (WPARAM) 0, 0);
                        SendDlgItemMessage( hwnd, ID_LIST,
                                           LB_DELETESTRING,
                                           (WPARAM) del_idx, 0);
                        // update local array of structures
                        for( out_cnt = del_idx; out_cnt < in_cnt; out_cnt++ )
                        {
                               contacts[out_cnt] = contacts[out_cnt+1];
                        }
                        in_cnt--;
                        break;
                  }
            }
            break;
   default:
   /* We do not want to handle this message so pass back to Windows */
   /*  to handle it in a default way                                */
      return DefWindowProc(hwnd, iMsg, wParam, lParam);
   }
   return 0;

} // end WndProc
```