```c
/*------------------------------------------
   POPPAD.c - Popup Editor
------------------------------------------*/

//Header
#include <windows.h>
#include <commdlg.h>
#include <stdio.h>
#include "resource.h"

//lib file
#pragma comment(lib, "comdlg32.lib")    //for common dialog box

//macro
#define EDITID      1
#define UNTITLED    TEXT("(untitled)")

//global function declarations
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
BOOL    CALLBACK AboutDlgProc(HWND, UINT, WPARAM, LPARAM);

        //Functions in PopFile.c
void PopFileInitialize(HWND);
BOOL PopFileOpenDlg   (HWND, PTSTR, PTSTR);
BOOL PopFileSaveDlg   (HWND, PTSTR, PTSTR);
BOOL PopFileRead      (HWND, PTSTR);
BOOL PopFileWrite     (HWND, PTSTR);

        //Function in PopFind.c
HWND PopFindFindDlg    (HWND);
HWND PopFindReplaceDlg (HWND);
BOOL PopFindFindText   (HWND, int*, LPFINDREPLACE);
BOOL PopFindReplaceText(HWND, int*, LPFINDREPLACE);
BOOL PopFindNextText   (HWND, int*);
BOOL PopFindValidFind  (void);

        //Function in PopFont.c
void PopFontInitialize  (HWND);
BOOL PopFontChooseFont  (HWND);
void PopFontSetFont     (HWND);
void PopFontDeinitialize(void);

        //Function in PopPrnt.c
BOOL PopPrntPrintFile(HINSTANCE, HWND, HWND, PTSTR);


//global variable declarations
static HWND  ghDlgModeless = NULL;
static TCHAR szAppName[] = TEXT("PopPad");

FILE *gpFile = NULL;


//WinMain()
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR szCmdLine, int iCmdShow)
{
    //variable declarations
    MSG      msg;
    HWND     hwnd;
    HACCEL   hAccel;
    WNDCLASS wndclass;

    //code
    if(fopen_s(&gpFile, "VJDLog.txt", "w") != 0)
    {
        MessageBox(NULL, TEXT("Error while creating log file."), szAppName, MB_OK | MB_ICONERROR);
        exit(0);
    }
    fprintf(gpFile, "PopPad.c Program Started*******\n");

    wndclass.style         = CS_HREDRAW | CS_VREDRAW;
    wndclass.lpfnWndProc   = WndProc;
    wndclass.cbClsExtra    = 0;
    wndclass.cbWndExtra    = 0;
    wndclass.hInstance     = hInstance;
    wndclass.hIcon         = LoadIcon(hInstance, szAppName);
    wndclass.hCursor       = LoadCursor(NULL, IDC_ARROW);
    wndclass.hbrBackground = (HBRUSH) GetStockObject(WHITE_BRUSH);
    wndclass.lpszMenuName  = szAppName;
    wndclass.lpszClassName = szAppName;

    if(!RegisterClass(&wndclass))
    {
        MessageBox(NULL, TEXT("Error while registering class."), szAppName, MB_OK | MB_ICONERROR);
        exit(0);
    }

    hwnd = CreateWindow(
            szAppName,
            NULL,
            WS_OVERLAPPEDWINDOW,
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            NULL,
            NULL,
            hInstance,
            szCmdLine
        );

    ShowWindow(hwnd, iCmdShow);
    UpdateWindow(hwnd);

    hAccel = LoadAccelerators(hInstance, szAppName);

    while(GetMessage(&msg, NULL, 0, 0))
    {
        if(ghDlgModeless == NULL || !IsDialogMessage(ghDlgModeless, &msg))
        {
            if(!TranslateAccelerator(hwnd, hAccel, &msg))
            {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
            }
        }
    }

    if(gpFile)
    {
        fprintf(gpFile, "PopPad.c Program Terminated******************\n");
        fclose(gpFile);
        gpFile = NULL;
    }

    return((int)msg.wParam);
}

//DoCaption()
void DoCaption(HWND hwnd, TCHAR *szTitleName)
{
    //variable declarations
    TCHAR szCaption[64 + MAX_PATH];

    //code
    wsprintf(szCaption, TEXT("%s - %s"), szAppName, szTitleName[0] ? szTitleName : UNTITLED);

    fprintf(gpFile, "PopPad.c DoCaption(): szCaption => %s\n", szCaption);

    SetWindowText(hwnd, szCaption);

    fprintf(gpFile, "PopPad.c DoCaption(): return\n");
}

//OkMessage()
void OkMessage(HWND hwnd, TCHAR *szMessage, TCHAR *szTitleName)
{
    //variable declarations
    TCHAR szBuffer[64 + MAX_PATH];

    //code
    wsprintf(szBuffer, szMessage, szTitleName[0] ? szTitleName : UNTITLED);
    MessageBox(hwnd, szBuffer, szAppName, MB_OK | MB_ICONEXCLAMATION);
}

//AskAboutSave()
short AskAboutSave(HWND hwnd, TCHAR *szTitleName)
{
    //variable declarations
    TCHAR szBuffer[64 + MAX_PATH];
    int   iReturn;

    //code
    wsprintf(szBuffer, TEXT("Save current changes in %s?"), szTitleName[0] ? szTitleName : UNTITLED);

    iReturn = MessageBox(hwnd, szBuffer, szAppName, MB_YESNOCANCEL | MB_ICONQUESTION);

    fprintf(gpFile, "PopPad.c AskAboutSave(): szBuffer => %s\n", szBuffer);
    if(iReturn == IDYES)
    {
        fprintf(gpFile, "PopPad.c AskAboutSave(): Sending Save Command\n");
        if(!SendMessage(hwnd, WM_COMMAND, IDM_FILE_SAVE, 0))
            iReturn = IDCANCEL;
    }

    fprintf(gpFile, "PopPad.c AskAboutSave(): return\n");
    return(iReturn);
}


//WndProc()
LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    //variable declarations
    static BOOL      bNeedSave = FALSE;
    static HINSTANCE hInst;
    static HWND      hwndEdit;
    static int       iOffset;
    static TCHAR     szFileName[MAX_PATH], szTitleName[MAX_PATH];
    static UINT      messageFindReplace;

    int              iSelBeg, iSelEnd, iEnable;
    LPFINDREPLACE    pfr;

    //code
    switch(message)
    {
        case WM_CREATE:
            fprintf(gpFile, "PopPad.c WM_CREATE: Enter\n");
            hInst = ((LPCREATESTRUCT)lParam)->hInstance;

                //Create the edit control child window
            hwndEdit = CreateWindow(
                    TEXT("edit"),
                    NULL,
                    WS_CHILD | WS_VISIBLE | WS_HSCROLL | WS_VSCROLL | WS_BORDER | ES_LEFT | ES_MULTILINE |
                    ES_NOHIDESEL | ES_AUTOHSCROLL | ES_AUTOVSCROLL,
                    0, 0, 0, 0,
                    hwnd, (HMENU)EDITID, hInst, NULL
                );

            SendMessage(hwndEdit, EM_LIMITTEXT, 32000, 0L);

                //Initialize common dialog box stuff
            PopFileInitialize(hwnd);
            PopFontInitialize(hwndEdit);

            messageFindReplace = RegisterWindowMessage(FINDMSGSTRING);

            DoCaption(hwnd, szTitleName);
            fprintf(gpFile, "PopPad.c WM_CREATE return\n");
            return(0);

        case WM_SETFOCUS:
            fprintf(gpFile, "PopPad.c WM_SETFOCUS Enter\n");
            SetFocus(hwndEdit);
            fprintf(gpFile, "PopPad.c WM_SETFOCUS Return\n");
            return(0);

        case WM_SIZE:
            fprintf(gpFile, "PopPad.c WM_SIZE Enter\n");
            MoveWindow(hwndEdit, 0, 0, LOWORD(lParam), HIWORD(lParam), TRUE);
            fprintf(gpFile, "PopPad.c WM_SIZE Return\n");
            return(0);

        case WM_INITMENUPOPUP:
            switch(lParam)
            {
                case 1:     //Edit Menu
                        //Enable Undo if edit control can do it
                    fprintf(gpFile,"PopPad.c WM_INITMENUPOPUP: Edit Menu Enter\n");
                    EnableMenuItem(
                        (HMENU) wParam,
                        IDM_EDIT_UNDO,
                        SendMessage(hwndEdit, EM_CANUNDO, 0, 0L) ? MF_ENABLED : MF_GRAYED
                    );

                        //Enable Paste if text is in the clipboard
                    EnableMenuItem(
                        (HMENU)wParam,
                        IDM_EDIT_PASTE,
                        IsClipboardFormatAvailable(CF_TEXT) ? MF_ENABLED : MF_GRAYED
                    );

                        //Endable Cut, Copy, and Del if text is selected
                    SendMessage(hwndEdit, EM_GETSEL, (WPARAM)&iSelBeg, (LPARAM) &iSelEnd);

                    iEnable = ( (iSelBeg != iSelEnd) ? MF_ENABLED : MF_GRAYED );

                    EnableMenuItem((HMENU)wParam, IDM_EDIT_CUT,   iEnable);
                    EnableMenuItem((HMENU)wParam, IDM_EDIT_COPY,  iEnable);
                    EnableMenuItem((HMENU)wParam, IDM_EDIT_CLEAR, iEnable);

                    fprintf(gpFile, "PopPad.c WM_INIRMENUPOPUP: Edit Menu Return\n");
                    break;

                case 2: //Search Menu
                    //Enable Find, Next, and Replace if modeless dialogs are not already active
                    fprintf(gpFile, "PopPad.c WM_INITMENUPOPUP: Search Menu Enter\n");
                    iEnable = (ghDlgModeless == NULL) ? MF_ENABLED : MF_GRAYED;
                EnableMenuItem((HMENU)wParam, IDM_SEARCH_FIND,    iEnable);
                EnableMenuItem((HMENU)wParam, IDM_SEARCH_NEXT,    iEnable);
                EnableMenuItem((HMENU)wParam, IDM_SEARCH_REPLACE, iEnable);

                fprintf(gpFile, "PopPad.c WM_INIRMENUPOPUP: Search Menu Return\n");
                break;
        }
        return(0);

case WM_COMMAND:
        //messages from edit control
    if(lParam && LOWORD(wParam) == EDITID)
    {
        fprintf(gpFile, "PopPad.c WM_COMMAND: Edit Control Message Enter\n");
        switch(HIWORD(wParam))
        {
            case EN_UPDATE:
                bNeedSave = TRUE;
                return(0);

            case EN_ERRSPACE:
            case EN_MAXTEXT:
                MessageBox(hwnd, TEXT("Edit control out of space."), szAppName, MB_OK | MB_ICONSTOP);
                return(0);
        }
        fprintf(gpFile, "PopPad.c WM_COMMAND: Edit Control Message Return\n");
        break;
    }

    switch(LOWORD(wParam))
    {
        //Message from File Menu
        case IDM_FILE_NEW:

            fprintf(gpFile, "PopPad.c IDM_FILE_NEW: Enter\n");
            if(bNeedSave && IDCANCEL == AskAboutSave(hwnd, szTitleName))
            {
                fprintf(gpFile, "PopPad.c IDM_FILE_NEW: Cancel Button Clicked\n");
                return(0);
            }

            SetWindowText(hwndEdit, TEXT("\0"));
            szFileName[0] = '\0';
            szTitleName[0] = '\0';
            DoCaption(hwnd, szTitleName);
            bNeedSave = FALSE;

            fprintf(gpFile, "PopPad.c IDM_FILE_NEW: Save Button Clicked\n");
            return(0);

        case IDM_FILE_OPEN:
            fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Enter\n");
            if(bNeedSave && IDCANCEL == AskAboutSave(hwnd, szTitleName))
            {
                fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: AskAboutSave() => cancel Button clicked\n");
                return(0);
            }

            if(PopFileOpenDlg(hwnd, szFileName, szTitleName))
            {
                fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: File Opened\n");
                fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Reading File\n");
                if(!PopFileRead(hwndEdit, szFileName))
                {
                    fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Error while reading\n");
                    OkMessage(hwnd, TEXT("Could not read file %s!"), szTitleName);
                    szFileName[0] = '\0';
                    szTitleName[0] = '\0';
                }
            }

            DoCaption(hwnd, szTitleName);
            bNeedSave = FALSE;

            fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Returned\n");
            return(0);

        case IDM_FILE_SAVE:
            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Enter\n");
            if(szFileName[0])
            {
                if(PopFileWrite(hwndEdit, szFileName))
                {
                    bNeedSave = FALSE;
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Data written to file \"%s\"\n", szFileName);
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Return\n");
                    return(1);
                }
                else
                {
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Error while writing data in file \"%s\"\n", szFileName);
                    OkMessage(hwnd, TEXT("1Could not write file %s"), szTitleName);
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Return\n");
                    return(0);
                }
            }
            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Fall Through to IDM_FILE_SAVE_AS\n");
            //fall through

        case IDM_FILE_SAVE_AS:
            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Enter\n");
            if(PopFileSaveDlg(hwnd, szFileName, szTitleName))
            {
                DoCaption(hwnd, szTitleName);

                if(PopFileWrite(hwndEdit, szFileName))
                {
                    bNeedSave = FALSE;
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: File save\n");
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Return\n");
                    return(1);
                }
                else
                {
                    OkMessage(hwnd, TEXT("2Could not write file %s"), szTitleName);
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Error while saving\n");
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Return\n");
                    return(0);
                }
            }
            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Return\n");
            return(0);

        case IDM_FILE_PRINT:
            fprintf(gpFile, "PopPad.c IDM_FILE_PRINT: Enter\n");
            if(!PopPrntPrintFile(hInst, hwnd, hwndEdit, szTitleName))
                OkMessage(hwnd, TEXT("Could not print file %s"), szTitleName);

            fprintf(gpFile, "PopPad.c IDM_FILE_PRINT: Return\n");
            return(0);

        case IDM_APP_EXIT:
            fprintf(gpFile, "PopPad.c IDM_APP_EXIT\n");
            SendMessage(hwnd, WM_CLOSE, 0, 0);
            return(0);
            //Messages from Edit Menu
        case IDM_EDIT_UNDO:
            fprintf(gpFile, "PopPad.c IDM_EDIT_UNDO\n");
            SendMessage(hwndEdit, WM_UNDO, 0, 0);
            return(0);

        case IDM_EDIT_CUT:
            fprintf(gpFile, "PopPad.c IDM_EDIT_CUT\n");
            SendMessage(hwndEdit, WM_CUT, 0, 0);
            return(0);

        case IDM_EDIT_COPY:
            fprintf(gpFile, "PopPad.c IDM_EDIT_COPY\n");
            SendMessage(hwndEdit, WM_COPY, 0, 0);
            return(0);

        case IDM_EDIT_PASTE:
            fprintf(gpFile, "PopPad.c IDM_EDIT_PASTE\n");
            SendMessage(hwndEdit, WM_PASTE, 0, 0);
            return(0);

        case IDM_EDIT_CLEAR:
            fprintf(gpFile, "PopPad.c IDM_EDIT_CLEAR\n");
            SendMessage(hwndEdit, WM_CLEAR, 0, 0);
            return(0);

        case IDM_EDIT_SELECT_ALL:
            fprintf(gpFile, "PopPad.c IDM_EDIT_SELECT_ALL\n");
            SendMessage(hwndEdit, EM_SETSEL, 0, -1);
            return(0);


                //Messages from Search Menu
        case IDM_SEARCH_FIND:
            fprintf(gpFile, "PopPad.c IDM_SEARCH_FIND: Enter\n");
            SendMessage(hwndEdit, EM_GETSEL, 0, (LPARAM) &iOffset);
            ghDlgModeless = PopFindFindDlg(hwnd);
            fprintf(gpFile, "PopPad.c IDM_SEARCH_FIND: Return\n");
            return(0);

        case IDM_SEARCH_NEXT:
            fprintf(gpFile, "PopPad.c IDM_SEARCH_NEXT: Enter\n");
            SendMessage(hwndEdit, EM_GETSEL, 0, (LPARAM) &iOffset);

            if(PopFindValidFind())
                PopFindNextText(hwnd, &iOffset);
            else
                ghDlgModeless = PopFindFindDlg(hwnd);

            fprintf(gpFile, "PopPad.c IDM_SEARCH_NEXT: Return\n");

            return(0);

        case IDM_SEARCH_REPLACE:
            fprintf(gpFile, "PopPad.c IDM_SEARCH_REPLACE: Enter\n");
            SendMessage(hwndEdit, EM_GETSEL, 0, (LPARAM) &iOffset);
            ghDlgModeless = PopFindReplaceDlg(hwnd);
            fprintf(gpFile, "PopPad.c IDM_SEARCH_REPLACE: Return\n");
            return(0);

        case IDM_FORMAT_FONT:
            fprintf(gpFile, "PopPad.c IDM_FORMAT_FONT: Enter\n");
            if(PopFontChooseFont(hwnd))
                PopFontSetFont(hwndEdit);
            fprintf(gpFile, "PopPad.c IDM_FORMAT_FONT: Return\n");

            return(0);
```

```
                //Messages from Help Menu
            case IDM_HELP:
                fprintf(gpFile, "PopPad.c IDM_HELP\n");
                OkMessage(hwnd, TEXT("Help not yet implemented!"), TEXT("\0"));
                return(0);

            case IDM_APP_ABOUT:
                fprintf(gpFile, "PopPad.c IDM_APP_ABOUT: Enter\n");

                DialogBox(
                    hInst,
                    TEXT("ABOUTDLGBOX"),
                    hwnd,
                    AboutDlgProc
                );

                fprintf(gpFile, "PopPad.c IDM_APP_ABOUT: Return\n");
                return(0);
        }
        break;

    case WM_CLOSE:
        fprintf(gpFile, "PopPad.c WM_CLOSE: Enter\n");
        if(!bNeedSave || IDCANCEL != AskAboutSave(hwnd, szTitleName))
            DestroyWindow(hwnd);
        fprintf(gpFile, "PopPad.c WM_CLOSE: Return\n");

        return(0);

    case WM_QUERYENDSESSION:
        fprintf(gpFile, "PopPad.c WM_QUERYENDSESSION\n");
        if(!bNeedSave || IDCANCEL != AskAboutSave(hwnd, szTitleName))
            return(1);
        return(0);

    case WM_DESTROY:
        fprintf(gpFile, "PopPad.c WM_DESTROY\n");
        PopFontDeinitialize();
        PostQuitMessage(0);
        return(0);


    default:
        //Process "Find-Replace message
        if(message == messageFindReplace)
        {
            fprintf(gpFile, "PopPad.c messageFindReplace: Enter\n");
            pfr = (LPFINDREPLACE)lParam;

            if(pfr->Flags & FR_DIALOGTERM)
                ghDlgModeless = NULL;

            if(pfr->Flags & FR_FINDNEXT)
                if(!PopFindFindText(hwndEdit, &iOffset, pfr))
                    OkMessage(hwnd, TEXT("Text not found!"), TEXT("\0"));

            if(pfr->Flags & FR_REPLACE || pfr->Flags & FR_REPLACEALL)
                if(!PopFindReplaceText(hwndEdit, &iOffset, pfr))
                    OkMessage(hwnd, TEXT("Text not found!"), TEXT("\0"));

            if(pfr->Flags & FR_REPLACEALL)
                while(PopFindReplaceText(hwndEdit, &iOffset, pfr));

            fprintf(gpFile, "PopPad.c messageFindReplace: return\n");
            return(0);
        }
        break;
    }

    return(DefWindowProc(hwnd, message, wParam, lParam));
}


//AboutDlgProc()
BOOL CALLBACK AboutDlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    //code
    switch(message)
    {
        case WM_INITDIALOG:
            fprintf(gpFile, "PopPad.c WM_INITDIALOG\n");
            return(TRUE);

        case WM_COMMAND:
            switch(LOWORD(wParam))
            {
                case IDOK:
                    fprintf(gpFile, "PopPad.c WM_COMMAND: IDOK\n");
                    EndDialog(hDlg, 0);
                    return(TRUE);
            }
            break;
    }
    return(FALSE);
}
```

```c
/*------------------------------------------
   POPPAD.c - Popup Editor
------------------------------------------*/

//Header
#include <windows.h>
#include <commdlg.h>
#include <stdio.h>
#include "resource.h"

//lib file
#pragma comment(lib, "comdlg32.lib")    //for common dialog box

//macro
#define EDITID      1
#define UNTITLED    TEXT("(untitled)")

//global function declarations
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
BOOL    CALLBACK AboutDlgProc(HWND, UINT, WPARAM, LPARAM);

        //Functions in PopFile.c
void PopFileInitialize(HWND);
BOOL PopFileOpenDlg   (HWND, PTSTR, PTSTR);
BOOL PopFileSaveDlg   (HWND, PTSTR, PTSTR);
BOOL PopFileRead      (HWND, PTSTR);
BOOL PopFileWrite     (HWND, PTSTR);

        //Function in PopFind.c
HWND PopFindFindDlg    (HWND);
HWND PopFindReplaceDlg (HWND);
BOOL PopFindFindText   (HWND, int*, LPFINDREPLACE);
BOOL PopFindReplaceText(HWND, int*, LPFINDREPLACE);
BOOL PopFindNextText   (HWND, int*);
BOOL PopFindValidFind  (void);

        //Function in PopFont.c
void PopFontInitialize  (HWND);
BOOL PopFontChooseFont  (HWND);
void PopFontSetFont     (HWND);
void PopFontDeinitialize(void);

        //Function in PopPrnt.c
BOOL PopPrntPrintFile(HINSTANCE, HWND, HWND, PTSTR);


//global variable declarations
static HWND  ghDlgModeless = NULL;
static TCHAR szAppName[] = TEXT("PopPad");

FILE *gpFile = NULL;


//WinMain()
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR szCmdLine, int iCmdShow)
{
    //variable declarations
    MSG      msg;
    HWND     hwnd;
    HACCEL   hAccel;
    WNDCLASS wndclass;

    //code
    if(fopen_s(&gpFile, "VJDLog.txt", "w") != 0)
    {
        MessageBox(NULL, TEXT("Error while creating log file."), szAppName, MB_OK | MB_ICONERROR);
        exit(0);
    }
    fprintf(gpFile, "PopPad.c Program Started*******\n");

    wndclass.style         = CS_HREDRAW | CS_VREDRAW;
    wndclass.lpfnWndProc   = WndProc;
    wndclass.cbClsExtra    = 0;
    wndclass.cbWndExtra    = 0;
    wndclass.hInstance     = hInstance;
    wndclass.hIcon         = LoadIcon(hInstance, szAppName);
    wndclass.hCursor       = LoadCursor(NULL, IDC_ARROW);
    wndclass.hbrBackground = (HBRUSH) GetStockObject(WHITE_BRUSH);
    wndclass.lpszMenuName  = szAppName;
    wndclass.lpszClassName = szAppName;

    if(!RegisterClass(&wndclass))
    {
        MessageBox(NULL, TEXT("Error while registering class."), szAppName, MB_OK | MB_ICONERROR);
        exit(0);
    }

    hwnd = CreateWindow(
            szAppName,
            NULL,
            WS_OVERLAPPEDWINDOW,
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            NULL,
            NULL,
            hInstance,
            szCmdLine
        );

    ShowWindow(hwnd, iCmdShow);
    UpdateWindow(hwnd);

    hAccel = LoadAccelerators(hInstance, szAppName);

    while(GetMessage(&msg, NULL, 0, 0))
    {
        if(ghDlgModeless == NULL || !IsDialogMessage(ghDlgModeless, &msg))
        {
            if(!TranslateAccelerator(hwnd, hAccel, &msg))
            {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
            }
        }
    }

    if(gpFile)
    {
        fprintf(gpFile, "PopPad.c Program Terminated******************\n");
        fclose(gpFile);
        gpFile = NULL;
    }

    return((int)msg.wParam);
}

//DoCaption()
void DoCaption(HWND hwnd, TCHAR *szTitleName)
{
    //variable declarations
    TCHAR szCaption[64 + MAX_PATH];

    //code
    wsprintf(szCaption, TEXT("%s - %s"), szAppName, szTitleName[0] ? szTitleName : UNTITLED);

    fprintf(gpFile, "PopPad.c DoCaption(): szCaption => %s\n", szCaption);

    SetWindowText(hwnd, szCaption);

    fprintf(gpFile, "PopPad.c DoCaption(): return\n");
}

//OkMessage()
void OkMessage(HWND hwnd, TCHAR *szMessage, TCHAR *szTitleName)
{
    //variable declarations
    TCHAR szBuffer[64 + MAX_PATH];

    //code
    wsprintf(szBuffer, szMessage, szTitleName[0] ? szTitleName : UNTITLED);
    MessageBox(hwnd, szBuffer, szAppName, MB_OK | MB_ICONEXCLAMATION);
}

//AskAboutSave()
short AskAboutSave(HWND hwnd, TCHAR *szTitleName)
{
    //variable declarations
    TCHAR szBuffer[64 + MAX_PATH];
    int   iReturn;

    //code
    wsprintf(szBuffer, TEXT("Save current changes in %s?"), szTitleName[0] ? szTitleName : UNTITLED);

    iReturn = MessageBox(hwnd, szBuffer, szAppName, MB_YESNOCANCEL | MB_ICONQUESTION);

    fprintf(gpFile, "PopPad.c AskAboutSave(): szBuffer => %s\n", szBuffer);
    if(iReturn == IDYES)
    {
        fprintf(gpFile, "PopPad.c AskAboutSave(): Sending Save Command\n");
        if(!SendMessage(hwnd, WM_COMMAND, IDM_FILE_SAVE, 0))
            iReturn = IDCANCEL;
    }

    fprintf(gpFile, "PopPad.c AskAboutSave(): return\n");
    return(iReturn);
}


//WndProc()
LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    //variable declarations
    static BOOL      bNeedSave = FALSE;
    static HINSTANCE hInst;
    static HWND      hwndEdit;
    static int       iOffset;
    static TCHAR     szFileName[MAX_PATH], szTitleName[MAX_PATH];
    static UINT      messageFindReplace;

    int              iSelBeg, iSelEnd, iEnable;
    LPFINDREPLACE    pfr;

    //code
    switch(message)
    {
        case WM_CREATE:
            fprintf(gpFile, "PopPad.c WM_CREATE: Enter\n");
            hInst = ((LPCREATESTRUCT)lParam)->hInstance;

                //Create the edit control child window
            hwndEdit = CreateWindow(
                    TEXT("edit"),
                    NULL,
                    WS_CHILD | WS_VISIBLE | WS_HSCROLL | WS_VSCROLL | WS_BORDER | ES_LEFT | ES_MULTILINE |
                    ES_NOHIDESEL | ES_AUTOHSCROLL | ES_AUTOVSCROLL,
                    0, 0, 0, 0,
                    hwnd, (HMENU)EDITID, hInst, NULL
                );

            SendMessage(hwndEdit, EM_LIMITTEXT, 32000, 0L);

                //Initialize common dialog box stuff
            PopFileInitialize(hwnd);
            PopFontInitialize(hwndEdit);

            messageFindReplace = RegisterWindowMessage(FINDMSGSTRING);

            DoCaption(hwnd, szTitleName);
            fprintf(gpFile, "PopPad.c WM_CREATE return\n");
            return(0);

        case WM_SETFOCUS:
            fprintf(gpFile, "PopPad.c WM_SETFOCUS Enter\n");
            SetFocus(hwndEdit);
            fprintf(gpFile, "PopPad.c WM_SETFOCUS Return\n");
            return(0);

        case WM_SIZE:
            fprintf(gpFile, "PopPad.c WM_SIZE Enter\n");
            MoveWindow(hwndEdit, 0, 0, LOWORD(lParam), HIWORD(lParam), TRUE);
            fprintf(gpFile, "PopPad.c WM_SIZE Return\n");
            return(0);

        case WM_INITMENUPOPUP:
            switch(lParam)
            {
                case 1:     //Edit Menu
                        //Enable Undo if edit control can do it
                    fprintf(gpFile,"PopPad.c WM_INITMENUPOPUP: Edit Menu Enter\n");
                    EnableMenuItem(
                        (HMENU) wParam,
                        IDM_EDIT_UNDO,
                        SendMessage(hwndEdit, EM_CANUNDO, 0, 0L) ? MF_ENABLED : MF_GRAYED
                    );

                        //Enable Paste if text is in the clipboard
                    EnableMenuItem(
                        (HMENU)wParam,
                        IDM_EDIT_PASTE,
                        IsClipboardFormatAvailable(CF_TEXT) ? MF_ENABLED : MF_GRAYED
                    );

                        //Endable Cut, Copy, and Del if text is selected
                    SendMessage(hwndEdit, EM_GETSEL, (WPARAM)&iSelBeg, (LPARAM) &iSelEnd);

                    iEnable = ( (iSelBeg != iSelEnd) ? MF_ENABLED : MF_GRAYED );

                    EnableMenuItem((HMENU)wParam, IDM_EDIT_CUT,   iEnable);
                    EnableMenuItem((HMENU)wParam, IDM_EDIT_COPY,  iEnable);
                    EnableMenuItem((HMENU)wParam, IDM_EDIT_CLEAR, iEnable);

                    fprintf(gpFile, "PopPad.c WM_INIRMENUPOPUP: Edit Menu Return\n");
                    break;

                case 2: //Search Menu
                    //Enable Find, Next, and Replace if modeless dialogs are not already active
                    fprintf(gpFile, "PopPad.c WM_INITMENUPOPUP: Search Menu Enter\n");
                    iEnable = (ghDlgModeless == NULL) ? MF_ENABLED : MF_GRAYED;
                    EnableMenuItem((HMENU)wParam, IDM_SEARCH_FIND,    iEnable);
                    EnableMenuItem((HMENU)wParam, IDM_SEARCH_NEXT,    iEnable);
                    EnableMenuItem((HMENU)wParam, IDM_SEARCH_REPLACE, iEnable);

                    fprintf(gpFile, "PopPad.c WM_INIRMENUPOPUP: Search Menu Return\n");
                    break;
            }
            return(0);

        case WM_COMMAND:
                //messages from edit control
            if(lParam && LOWORD(wParam) == EDITID)
            {
                fprintf(gpFile, "PopPad.c WM_COMMAND: Edit Control Message Enter\n");
                switch(HIWORD(wParam))
                {
                    case EN_UPDATE:
                        bNeedSave = TRUE;
                        return(0);

                    case EN_ERRSPACE:
                    case EN_MAXTEXT:
                        MessageBox(hwnd, TEXT("Edit control out of space."), szAppName, MB_OK | MB_ICONSTOP);
                        return(0);
                }
                fprintf(gpFile, "PopPad.c WM_COMMAND: Edit Control Message Return\n");
                break;
            }

            switch(LOWORD(wParam))
            {
                //Message from File Menu
                case IDM_FILE_NEW:

                    fprintf(gpFile, "PopPad.c IDM_FILE_NEW: Enter\n");
                    if(bNeedSave && IDCANCEL == AskAboutSave(hwnd, szTitleName))
                    {
                        fprintf(gpFile, "PopPad.c IDM_FILE_NEW: Cancel Button Clicked\n");
                        return(0);
                    }

                    SetWindowText(hwndEdit, TEXT("\0"));
                    szFileName[0] = '\0';
                    szTitleName[0] = '\0';
                    DoCaption(hwnd, szTitleName);
                    bNeedSave = FALSE;

                    fprintf(gpFile, "PopPad.c IDM_FILE_NEW: Save Button Clicked\n");
                    return(0);

                case IDM_FILE_OPEN:
                    fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Enter\n");
                    if(bNeedSave && IDCANCEL == AskAboutSave(hwnd, szTitleName))
                    {
                        fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: AskAboutSave() => cancel Button clicked\n");
                        return(0);
                    }

                    if(PopFileOpenDlg(hwnd, szFileName, szTitleName))
                    {
                        fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: File Opened\n");
                        fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Reading File\n");
                        if(!PopFileRead(hwndEdit, szFileName))
                        {
                            fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Error while reading\n");
                            OkMessage(hwnd, TEXT("Could not read file %s!"), szTitleName);
                            szFileName[0] = '\0';
                            szTitleName[0] = '\0';
                        }
                    }

                    DoCaption(hwnd, szTitleName);
                    bNeedSave = FALSE;

                    fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Returned\n");
                    return(0);

                case IDM_FILE_SAVE:
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Enter\n");
                    if(szFileName[0])
                    {
                        if(PopFileWrite(hwndEdit, szFileName))
                        {
                            bNeedSave = FALSE;
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Data written to file \"%s\"\n", szFileName);
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Return\n");
                            return(1);
                        }
                        else
                        {
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Error while writing data in file \"%s\"\n", szFileName);
                            OkMessage(hwnd, TEXT("1Could not write file %s"), szTitleName);
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Return\n");
                            return(0);
                        }
                    }
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Fall Through to IDM_FILE_SAVE_AS\n");
                    //fall through

                case IDM_FILE_SAVE_AS:
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Enter\n");
                    if(PopFileSaveDlg(hwnd, szFileName, szTitleName))
                    {
                        DoCaption(hwnd, szTitleName);

                        if(PopFileWrite(hwndEdit, szFileName))
                        {
                            bNeedSave = FALSE;
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: File save\n");
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Return\n");
                            return(1);
                        }
                        else
                        {
                            OkMessage(hwnd, TEXT("2Could not write file %s"), szTitleName);
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Error while saving\n");
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Return\n");
                            return(0);
                        }
                    }
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Return\n");
                    return(0);

                case IDM_FILE_PRINT:
                    fprintf(gpFile, "PopPad.c IDM_FILE_PRINT: Enter\n");
                    if(!PopPrntPrintFile(hInst, hwnd, hwndEdit, szTitleName))
                        OkMessage(hwnd, TEXT("Could not print file %s"), szTitleName);

                    fprintf(gpFile, "PopPad.c IDM_FILE_PRINT: Return\n");
                    return(0);

                case IDM_APP_EXIT:
                    fprintf(gpFile, "PopPad.c IDM_APP_EXIT\n");
                    SendMessage(hwnd, WM_CLOSE, 0, 0);
                    return(0);
                    //Messages from Edit Menu
                case IDM_EDIT_UNDO:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_UNDO\n");
                    SendMessage(hwndEdit, WM_UNDO, 0, 0);
                    return(0);

                case IDM_EDIT_CUT:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_CUT\n");
                    SendMessage(hwndEdit, WM_CUT, 0, 0);
                    return(0);

                case IDM_EDIT_COPY:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_COPY\n");
                    SendMessage(hwndEdit, WM_COPY, 0, 0);
                    return(0);

                case IDM_EDIT_PASTE:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_PASTE\n");
                    SendMessage(hwndEdit, WM_PASTE, 0, 0);
                    return(0);

                case IDM_EDIT_CLEAR:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_CLEAR\n");
                    SendMessage(hwndEdit, WM_CLEAR, 0, 0);
                    return(0);

                case IDM_EDIT_SELECT_ALL:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_SELECT_ALL\n");
                    SendMessage(hwndEdit, EM_SETSEL, 0, -1);
                    return(0);


                        //Messages from Search Menu
                case IDM_SEARCH_FIND:
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_FIND: Enter\n");
                    SendMessage(hwndEdit, EM_GETSEL, 0, (LPARAM) &iOffset);
                    ghDlgModeless = PopFindFindDlg(hwnd);
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_FIND: Return\n");
                    return(0);

                case IDM_SEARCH_NEXT:
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_NEXT: Enter\n");
                    SendMessage(hwndEdit, EM_GETSEL, 0, (LPARAM) &iOffset);

                    if(PopFindValidFind())
                        PopFindNextText(hwnd, &iOffset);
                    else
                        ghDlgModeless = PopFindFindDlg(hwnd);

                    fprintf(gpFile, "PopPad.c IDM_SEARCH_NEXT: Return\n");

                    return(0);

                case IDM_SEARCH_REPLACE:
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_REPLACE: Enter\n");
                    SendMessage(hwndEdit, EM_GETSEL, 0, (LPARAM) &iOffset);
                    ghDlgModeless = PopFindReplaceDlg(hwnd);
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_REPLACE: Return\n");
                    return(0);

                case IDM_FORMAT_FONT:
                    fprintf(gpFile, "PopPad.c IDM_FORMAT_FONT: Enter\n");
                    if(PopFontChooseFont(hwnd))
                        PopFontSetFont(hwndEdit);
                    fprintf(gpFile, "PopPad.c IDM_FORMAT_FONT: Return\n");

                    return(0);
```

```
                //Messages from Help Menu
        case IDM_HELP:
            fprintf(gpFile, "PopPad.c IDM_HELP\n");
            OkMessage(hwnd, TEXT("Help not yet implemented!"), TEXT("\0"));
            return(0);

        case IDM_APP_ABOUT:
            fprintf(gpFile, "PopPad.c IDM_APP_ABOUT: Enter\n");

            DialogBox(
                hInst,
                TEXT("ABOUTDLGBOX"),
                hwnd,
                AboutDlgProc
            );

            fprintf(gpFile, "PopPad.c IDM_APP_ABOUT: Return\n");
            return(0);
    }
    break;

case WM_CLOSE:
    fprintf(gpFile, "PopPad.c WM_CLOSE: Enter\n");
    if(!bNeedSave || IDCANCEL != AskAboutSave(hwnd, szTitleName))
        DestroyWindow(hwnd);
    fprintf(gpFile, "PopPad.c WM_CLOSE: Return\n");

    return(0);

case WM_QUERYENDSESSION:
    fprintf(gpFile, "PopPad.c WM_QUERYENDSESSION\n");
    if(!bNeedSave || IDCANCEL != AskAboutSave(hwnd, szTitleName))
        return(1);
    return(0);

case WM_DESTROY:
    fprintf(gpFile, "PopPad.c WM_DESTROY\n");
    PopFontDeinitialize();
    PostQuitMessage(0);
    return(0);


default:
    //Process "Find-Replace message
    if(message == messageFindReplace)
    {
        fprintf(gpFile, "PopPad.c messageFindReplace: Enter\n");
        pfr = (LPFINDREPLACE)lParam;

        if(pfr->Flags & FR_DIALOGTERM)
            ghDlgModeless = NULL;

        if(pfr->Flags & FR_FINDNEXT)
            if(!PopFindFindText(hwndEdit, &iOffset, pfr))
                OkMessage(hwnd, TEXT("Text not found!"), TEXT("\0"));

        if(pfr->Flags & FR_REPLACE || pfr->Flags & FR_REPLACEALL)
            if(!PopFindReplaceText(hwndEdit, &iOffset, pfr))
                OkMessage(hwnd, TEXT("Text not found!"), TEXT("\0"));

        if(pfr->Flags & FR_REPLACEALL)
            while(PopFindReplaceText(hwndEdit, &iOffset, pfr));

        fprintf(gpFile, "PopPad.c messageFindReplace: return\n");
        return(0);
    }
    break;
    }

    return(DefWindowProc(hwnd, message, wParam, lParam));
}


//AboutDlgProc()
BOOL CALLBACK AboutDlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    //code
    switch(message)
    {
        case WM_INITDIALOG:
            fprintf(gpFile, "PopPad.c WM_INITDIALOG\n");
            return(TRUE);

        case WM_COMMAND:
            switch(LOWORD(wParam))
            {
                case IDOK:
                    fprintf(gpFile, "PopPad.c WM_COMMAND: IDOK\n");
                    EndDialog(hDlg, 0);
                    return(TRUE);
            }
            break;
    }
    return(FALSE);
}
```

```
/*-----------------------------------------
   POPPAD.c - Popup Editor
-----------------------------------------*/

//Header
#include <windows.h>
#include <commdlg.h>
#include <stdio.h>
#include "resource.h"

//lib file
#pragma comment(lib, "comdlg32.lib")    //for common dialog box

//macro
#define EDITID      1
#define UNTITLED    TEXT("(untitled)")

//global function declarations
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
BOOL    CALLBACK AboutDlgProc(HWND, UINT, WPARAM, LPARAM);

        //Functions in PopFile.c
void PopFileInitialize(HWND);
BOOL PopFileOpenDlg   (HWND, PTSTR, PTSTR);
BOOL PopFileSaveDlg   (HWND, PTSTR, PTSTR);
BOOL PopFileRead      (HWND, PTSTR);
BOOL PopFileWrite     (HWND, PTSTR);

        //Function in PopFind.c
HWND PopFindFindDlg    (HWND);
HWND PopFindReplaceDlg (HWND);
BOOL PopFindFindText   (HWND, int*, LPFINDREPLACE);
BOOL PopFindReplaceText(HWND, int*, LPFINDREPLACE);
BOOL PopFindNextText   (HWND, int*);
BOOL PopFindValidFind  (void);

        //Function in PopFont.c
void PopFontInitialize  (HWND);
BOOL PopFontChooseFont  (HWND);
void PopFontSetFont     (HWND);
void PopFontDeinitialize(void);

        //Function in PopPrnt.c
BOOL PopPrntPrintFile(HINSTANCE, HWND, HWND, PTSTR);


//global variable declarations
static HWND  ghDlgModeless = NULL;
static TCHAR szAppName[] = TEXT("PopPad");

FILE *gpFile = NULL;


//WinMain()
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR szCmdLine, int iCmdShow)
{
    //variable declarations
    MSG      msg;
    HWND     hwnd;
    HACCEL   hAccel;
    WNDCLASS wndclass;

    //code
    if(fopen_s(&gpFile, "VJDLog.txt", "w") != 0)
    {
        MessageBox(NULL, TEXT("Error while creating log file."), szAppName, MB_OK | MB_ICONERROR);
        exit(0);
    }
    fprintf(gpFile, "PopPad.c Program Started*******\n");

    wndclass.style          = CS_HREDRAW | CS_VREDRAW;
    wndclass.lpfnWndProc    = WndProc;
    wndclass.cbClsExtra     = 0;
    wndclass.cbWndExtra     = 0;
    wndclass.hInstance      = hInstance;
    wndclass.hIcon          = LoadIcon(hInstance, szAppName);
    wndclass.hCursor        = LoadCursor(NULL, IDC_ARROW);
    wndclass.hbrBackground = (HBRUSH) GetStockObject(WHITE_BRUSH);
    wndclass.lpszMenuName  = szAppName;
    wndclass.lpszClassName = szAppName;

    if(!RegisterClass(&wndclass))
    {
        MessageBox(NULL, TEXT("Error while registering class."), szAppName, MB_OK | MB_ICONERROR);
        exit(0);
    }

    hwnd = CreateWindow(
            szAppName,
            NULL,
            WS_OVERLAPPEDWINDOW,
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            NULL,
            NULL,
            hInstance,
            szCmdLine
        );

    ShowWindow(hwnd, iCmdShow);
    UpdateWindow(hwnd);

    hAccel = LoadAccelerators(hInstance, szAppName);

    while(GetMessage(&msg, NULL, 0, 0))
    {
        if(ghDlgModeless == NULL || !IsDialogMessage(ghDlgModeless, &msg))
        {
            if(!TranslateAccelerator(hwnd, hAccel, &msg))
            {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
            }
        }
    }

    if(gpFile)
    {
        fprintf(gpFile, "PopPad.c Program Terminated******************\n");
        fclose(gpFile);
        gpFile = NULL;
    }

    return((int)msg.wParam);
}

//DoCaption()
void DoCaption(HWND hwnd, TCHAR *szTitleName)
{
    //variable declarations
    TCHAR szCaption[64 + MAX_PATH];

    //code
    wsprintf(szCaption, TEXT("%s - %s"), szAppName, szTitleName[0] ? szTitleName : UNTITLED);

    fprintf(gpFile, "PopPad.c DoCaption(): szCaption => %s\n", szCaption);

    SetWindowText(hwnd, szCaption);

    fprintf(gpFile, "PopPad.c DoCaption(): return\n");
}

//OkMessage()
void OkMessage(HWND hwnd, TCHAR *szMessage, TCHAR *szTitleName)
{
    //variable declarations
    TCHAR szBuffer[64 + MAX_PATH];

    //code
    wsprintf(szBuffer, szMessage, szTitleName[0] ? szTitleName : UNTITLED);
    MessageBox(hwnd, szBuffer, szAppName, MB_OK | MB_ICONEXCLAMATION);
}

//AskAboutSave()
short AskAboutSave(HWND hwnd, TCHAR *szTitleName)
{
    //variable declarations
    TCHAR szBuffer[64 + MAX_PATH];
    int   iReturn;

    //code
    wsprintf(szBuffer, TEXT("Save current changes in %s?"), szTitleName[0] ? szTitleName : UNTITLED);

    iReturn = MessageBox(hwnd, szBuffer, szAppName, MB_YESNOCANCEL | MB_ICONQUESTION);

    fprintf(gpFile, "PopPad.c AskAboutSave(): szBuffer => %s\n", szBuffer);
    if(iReturn == IDYES)
    {
        fprintf(gpFile, "PopPad.c AskAboutSave(): Sending Save Command\n");
        if(!SendMessage(hwnd, WM_COMMAND, IDM_FILE_SAVE, 0))
            iReturn = IDCANCEL;
    }

    fprintf(gpFile, "PopPad.c AskAboutSave(): return\n");
    return(iReturn);
}


//WndProc()
LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    //variable declarations
    static BOOL      bNeedSave = FALSE;
    static HINSTANCE hInst;
    static HWND      hwndEdit;
    static int       iOffset;
    static TCHAR     szFileName[MAX_PATH], szTitleName[MAX_PATH];
    static UINT      messageFindReplace;

    int              iSelBeg, iSelEnd, iEnable;
    LPFINDREPLACE    pfr;

    //code
    switch(message)
    {
        case WM_CREATE:
            fprintf(gpFile, "PopPad.c WM_CREATE: Enter\n");
            hInst = ((LPCREATESTRUCT)lParam)->hInstance;

                //Create the edit control child window
            hwndEdit = CreateWindow(
                    TEXT("edit"),
                    NULL,
                    WS_CHILD | WS_VISIBLE | WS_HSCROLL | WS_VSCROLL | WS_BORDER | ES_LEFT | ES_MULTILINE |
                    ES_NOHIDESEL | ES_AUTOHSCROLL | ES_AUTOVSCROLL,
                    0, 0, 0, 0,
                    hwnd, (HMENU)EDITID, hInst, NULL
                );

            SendMessage(hwndEdit, EM_LIMITTEXT, 32000, 0L);

                //Initialize common dialog box stuff
            PopFileInitialize(hwnd);
            PopFontInitialize(hwndEdit);

            messageFindReplace = RegisterWindowMessage(FINDMSGSTRING);

            DoCaption(hwnd, szTitleName);
            fprintf(gpFile, "PopPad.c WM_CREATE return\n");
            return(0);

        case WM_SETFOCUS:
            fprintf(gpFile, "PopPad.c WM_SETFOCUS Enter\n");
            SetFocus(hwndEdit);
            fprintf(gpFile, "PopPad.c WM_SETFOCUS Return\n");
            return(0);

        case WM_SIZE:
            fprintf(gpFile, "PopPad.c WM_SIZE Enter\n");
            MoveWindow(hwndEdit, 0, 0, LOWORD(lParam), HIWORD(lParam), TRUE);
            fprintf(gpFile, "PopPad.c WM_SIZE Return\n");
            return(0);

        case WM_INITMENUPOPUP:
            switch(lParam)
            {
                case 1:     //Edit Menu
                        //Enable Undo if edit control can do it
                    fprintf(gpFile,"PopPad.c WM_INITMENUPOPUP: Edit Menu Enter\n");
                    EnableMenuItem(
                        (HMENU) wParam,
                        IDM_EDIT_UNDO,
                        SendMessage(hwndEdit, EM_CANUNDO, 0, 0L) ? MF_ENABLED : MF_GRAYED
                    );

                        //Enable Paste if text is in the clipboard
                    EnableMenuItem(
                        (HMENU)wParam,
                        IDM_EDIT_PASTE,
                        IsClipboardFormatAvailable(CF_TEXT) ? MF_ENABLED : MF_GRAYED
                    );

                        //Endable Cut, Copy, and Del if text is selected
                    SendMessage(hwndEdit, EM_GETSEL, (WPARAM)&iSelBeg, (LPARAM) &iSelEnd);

                    iEnable = ( (iSelBeg != iSelEnd) ? MF_ENABLED : MF_GRAYED );

                    EnableMenuItem((HMENU)wParam, IDM_EDIT_CUT,   iEnable);
                    EnableMenuItem((HMENU)wParam, IDM_EDIT_COPY,  iEnable);
                    EnableMenuItem((HMENU)wParam, IDM_EDIT_CLEAR, iEnable);

                    fprintf(gpFile, "PopPad.c WM_INIRMENUPOPUP: Edit Menu Return\n");
                    break;

                case 2: //Search Menu
                    //Enable Find, Next, and Replace if modeless dialogs are not already active
                    fprintf(gpFile, "PopPad.c WM_INITMENUPOPUP: Search Menu Enter\n");
                    iEnable = (ghDlgModeless == NULL) ? MF_ENABLED : MF_GRAYED;
                    EnableMenuItem((HMENU)wParam, IDM_SEARCH_FIND,    iEnable);
                    EnableMenuItem((HMENU)wParam, IDM_SEARCH_NEXT,    iEnable);
                    EnableMenuItem((HMENU)wParam, IDM_SEARCH_REPLACE, iEnable);

                    fprintf(gpFile, "PopPad.c WM_INIRMENUPOPUP: Search Menu Return\n");
                    break;
            }
            return(0);

        case WM_COMMAND:
                //messages from edit control
            if(lParam && LOWORD(wParam) == EDITID)
            {
                fprintf(gpFile, "PopPad.c WM_COMMAND: Edit Control Message Enter\n");
                switch(HIWORD(wParam))
                {
                    case EN_UPDATE:
                        bNeedSave = TRUE;
                        return(0);

                    case EN_ERRSPACE:
                    case EN_MAXTEXT:
                        MessageBox(hwnd, TEXT("Edit control out of space."), szAppName, MB_OK | MB_ICONSTOP);
                        return(0);
                }
                fprintf(gpFile, "PopPad.c WM_COMMAND: Edit Control Message Return\n");
                break;
            }

            switch(LOWORD(wParam))
            {
                //Message from File Menu
                case IDM_FILE_NEW:

                    fprintf(gpFile, "PopPad.c IDM_FILE_NEW: Enter\n");
                    if(bNeedSave && IDCANCEL == AskAboutSave(hwnd, szTitleName))
                    {
                        fprintf(gpFile, "PopPad.c IDM_FILE_NEW: Cancel Button Clicked\n");
                        return(0);
                    }

                    SetWindowText(hwndEdit, TEXT("\0"));
                    szFileName[0] = '\0';
                    szTitleName[0] = '\0';
                    DoCaption(hwnd, szTitleName);
                    bNeedSave = FALSE;

                    fprintf(gpFile, "PopPad.c IDM_FILE_NEW: Save Button Clicked\n");
                    return(0);

                case IDM_FILE_OPEN:
                    fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Enter\n");
                    if(bNeedSave && IDCANCEL == AskAboutSave(hwnd, szTitleName))
                    {
                        fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: AskAboutSave() => cancel Button clicked\n");
                        return(0);
                    }

                    if(PopFileOpenDlg(hwnd, szFileName, szTitleName))
                    {
                        fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: File Opened\n");
                        fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Reading File\n");
                        if(!PopFileRead(hwndEdit, szFileName))
                        {
                            fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Error while reading\n");
                            OkMessage(hwnd, TEXT("Could not read file %s!"), szTitleName);
                            szFileName[0] = '\0';
                            szTitleName[0] = '\0';
                        }
                    }

                    DoCaption(hwnd, szTitleName);
                    bNeedSave = FALSE;

                    fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Returned\n");
                    return(0);

                case IDM_FILE_SAVE:
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Enter\n");
                    if(szFileName[0])
                    {
                        if(PopFileWrite(hwndEdit, szFileName))
                        {
                            bNeedSave = FALSE;
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Data written to file \"%s\"\n", szFileName);
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Return\n");
                            return(1);
                        }
                        else
                        {
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Error while writing data in file \"%s\"\n", szFileName);
                            OkMessage(hwnd, TEXT("1Could not write file %s"), szTitleName);
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Return\n");
                            return(0);
                        }
                    }
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Fall Through to IDM_FILE_SAVE_AS\n");
                    //fall through

                case IDM_FILE_SAVE_AS:
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Enter\n");
                    if(PopFileSaveDlg(hwnd, szFileName, szTitleName))
                    {
                        DoCaption(hwnd, szTitleName);

                        if(PopFileWrite(hwndEdit, szFileName))
                        {
                            bNeedSave = FALSE;
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: File save\n");
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Return\n");
                            return(1);
                        }
                        else
                        {
                            OkMessage(hwnd, TEXT("2Could not write file %s"), szTitleName);
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Error while saving\n");
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Return\n");
                            return(0);
                        }
                    }
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Return\n");
                    return(0);

                case IDM_FILE_PRINT:
                    fprintf(gpFile, "PopPad.c IDM_FILE_PRINT: Enter\n");
                    if(!PopPrntPrintFile(hInst, hwnd, hwndEdit, szTitleName))
                        OkMessage(hwnd, TEXT("Could not print file %s"), szTitleName);

                    fprintf(gpFile, "PopPad.c IDM_FILE_PRINT: Return\n");
                    return(0);

                case IDM_APP_EXIT:
                    fprintf(gpFile, "PopPad.c IDM_APP_EXIT\n");
                    SendMessage(hwnd, WM_CLOSE, 0, 0);
                    return(0);
                    //Messages from Edit Menu
                case IDM_EDIT_UNDO:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_UNDO\n");
                    SendMessage(hwndEdit, WM_UNDO, 0, 0);
                    return(0);

                case IDM_EDIT_CUT:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_CUT\n");
                    SendMessage(hwndEdit, WM_CUT, 0, 0);
                    return(0);

                case IDM_EDIT_COPY:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_COPY\n");
                    SendMessage(hwndEdit, WM_COPY, 0, 0);
                    return(0);

                case IDM_EDIT_PASTE:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_PASTE\n");
                    SendMessage(hwndEdit, WM_PASTE, 0, 0);
                    return(0);

                case IDM_EDIT_CLEAR:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_CLEAR\n");
                    SendMessage(hwndEdit, WM_CLEAR, 0, 0);
                    return(0);

                case IDM_EDIT_SELECT_ALL:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_SELECT_ALL\n");
                    SendMessage(hwndEdit, EM_SETSEL, 0, -1);
                    return(0);


                        //Messages from Search Menu
                case IDM_SEARCH_FIND:
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_FIND: Enter\n");
                    SendMessage(hwndEdit, EM_GETSEL, 0, (LPARAM) &iOffset);
                    ghDlgModeless = PopFindFindDlg(hwnd);
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_FIND: Return\n");
                    return(0);

                case IDM_SEARCH_NEXT:
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_NEXT: Enter\n");
                    SendMessage(hwndEdit, EM_GETSEL, 0, (LPARAM) &iOffset);

                    if(PopFindValidFind())
                        PopFindNextText(hwnd, &iOffset);
                    else
                        ghDlgModeless = PopFindFindDlg(hwnd);

                    fprintf(gpFile, "PopPad.c IDM_SEARCH_NEXT: Return\n");

                    return(0);

                case IDM_SEARCH_REPLACE:
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_REPLACE: Enter\n");
                    SendMessage(hwndEdit, EM_GETSEL, 0, (LPARAM) &iOffset);
                    ghDlgModeless = PopFindReplaceDlg(hwnd);
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_REPLACE: Return\n");
                    return(0);

                case IDM_FORMAT_FONT:
                    fprintf(gpFile, "PopPad.c IDM_FORMAT_FONT: Enter\n");
                    if(PopFontChooseFont(hwnd))
                        PopFontSetFont(hwndEdit);
                    fprintf(gpFile, "PopPad.c IDM_FORMAT_FONT: Return\n");

                    return(0);
```

```
                //Messages from Help Menu
        case IDM_HELP:
            fprintf(gpFile, "PopPad.c IDM_HELP\n");
            OkMessage(hwnd, TEXT("Help not yet implemented!"), TEXT("\0"));
            return(0);

        case IDM_APP_ABOUT:
            fprintf(gpFile, "PopPad.c IDM_APP_ABOUT: Enter\n");

            DialogBox(
                hInst,
                TEXT("ABOUTDLGBOX"),
                hwnd,
                AboutDlgProc
            );

            fprintf(gpFile, "PopPad.c IDM_APP_ABOUT: Return\n");
            return(0);
    }
    break;

case WM_CLOSE:
    fprintf(gpFile, "PopPad.c WM_CLOSE: Enter\n");
    if(!bNeedSave || IDCANCEL != AskAboutSave(hwnd, szTitleName))
        DestroyWindow(hwnd);
    fprintf(gpFile, "PopPad.c WM_CLOSE: Return\n");

    return(0);

case WM_QUERYENDSESSION:
    fprintf(gpFile, "PopPad.c WM_QUERYENDSESSION\n");
    if(!bNeedSave || IDCANCEL != AskAboutSave(hwnd, szTitleName))
        return(1);
    return(0);

case WM_DESTROY:
    fprintf(gpFile, "PopPad.c WM_DESTROY\n");
    PopFontDeinitialize();
    PostQuitMessage(0);
    return(0);


default:
    //Process "Find-Replace message
    if(message == messageFindReplace)
    {
        fprintf(gpFile, "PopPad.c messageFindReplace: Enter\n");
        pfr = (LPFINDREPLACE)lParam;

        if(pfr->Flags & FR_DIALOGTERM)
            ghDlgModeless = NULL;

        if(pfr->Flags & FR_FINDNEXT)
            if(!PopFindFindText(hwndEdit, &iOffset, pfr))
                OkMessage(hwnd, TEXT("Text not found!"), TEXT("\0"));

        if(pfr->Flags & FR_REPLACE || pfr->Flags & FR_REPLACEALL)
            if(!PopFindReplaceText(hwndEdit, &iOffset, pfr))
                OkMessage(hwnd, TEXT("Text not found!"), TEXT("\0"));

        if(pfr->Flags & FR_REPLACEALL)
            while(PopFindReplaceText(hwndEdit, &iOffset, pfr));

        fprintf(gpFile, "PopPad.c messageFindReplace: return\n");
        return(0);
    }
    break;
    }

    return(DefWindowProc(hwnd, message, wParam, lParam));
}


//AboutDlgProc()
BOOL CALLBACK AboutDlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    //code
    switch(message)
    {
        case WM_INITDIALOG:
            fprintf(gpFile, "PopPad.c WM_INITDIALOG\n");
            return(TRUE);

        case WM_COMMAND:
            switch(LOWORD(wParam))
            {
                case IDOK:
                    fprintf(gpFile, "PopPad.c WM_COMMAND: IDOK\n");
                    EndDialog(hDlg, 0);
                    return(TRUE);
            }
            break;
    }
    return(FALSE);
}
```

```
/*-----------------------------------------
   POPPAD.c - Popup Editor
----------------------------------------*/

//Header
#include <windows.h>
#include <commdlg.h>
#include <stdio.h>
#include "resource.h"

//lib file
#pragma comment(lib, "comdlg32.lib")    //for common dialog box

//macro
#define EDITID      1
#define UNTITLED    TEXT("(untitled)")

//global function declarations
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
BOOL    CALLBACK AboutDlgProc(HWND, UINT, WPARAM, LPARAM);

        //Functions in PopFile.c
void PopFileInitialize(HWND);
BOOL PopFileOpenDlg   (HWND, PTSTR, PTSTR);
BOOL PopFileSaveDlg   (HWND, PTSTR, PTSTR);
BOOL PopFileRead      (HWND, PTSTR);
BOOL PopFileWrite     (HWND, PTSTR);

        //Function in PopFind.c
HWND PopFindFindDlg    (HWND);
HWND PopFindReplaceDlg (HWND);
BOOL PopFindFindText   (HWND, int*, LPFINDREPLACE);
BOOL PopFindReplaceText(HWND, int*, LPFINDREPLACE);
BOOL PopFindNextText   (HWND, int*);
BOOL PopFindValidFind  (void);

        //Function in PopFont.c
void PopFontInitialize  (HWND);
BOOL PopFontChooseFont  (HWND);
void PopFontSetFont     (HWND);
void PopFontDeinitialize(void);

        //Function in PopPrnt.c
BOOL PopPrntPrintFile(HINSTANCE, HWND, HWND, PTSTR);


//global variable declarations
static HWND  ghDlgModeless = NULL;
static TCHAR szAppName[] = TEXT("PopPad");

FILE *gpFile = NULL;


//WinMain()
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR szCmdLine, int iCmdShow)
{
    //variable declarations
    MSG      msg;
    HWND     hwnd;
    HACCEL   hAccel;
    WNDCLASS wndclass;

    //code
    if(fopen_s(&gpFile, "VJDLog.txt", "w") != 0)
    {
        MessageBox(NULL, TEXT("Error while creating log file."), szAppName, MB_OK | MB_ICONERROR);
        exit(0);
    }
    fprintf(gpFile, "PopPad.c Program Started*******\n");

    wndclass.style          = CS_HREDRAW | CS_VREDRAW;
    wndclass.lpfnWndProc    = WndProc;
    wndclass.cbClsExtra     = 0;
    wndclass.cbWndExtra     = 0;
    wndclass.hInstance      = hInstance;
    wndclass.hIcon          = LoadIcon(hInstance, szAppName);
    wndclass.hCursor        = LoadCursor(NULL, IDC_ARROW);
    wndclass.hbrBackground = (HBRUSH) GetStockObject(WHITE_BRUSH);
    wndclass.lpszMenuName  = szAppName;
    wndclass.lpszClassName = szAppName;

    if(!RegisterClass(&wndclass))
    {
        MessageBox(NULL, TEXT("Error while registering class."), szAppName, MB_OK | MB_ICONERROR);
        exit(0);
    }

    hwnd = CreateWindow(
            szAppName,
            NULL,
            WS_OVERLAPPEDWINDOW,
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            NULL,
            NULL,
            hInstance,
            szCmdLine
        );

    ShowWindow(hwnd, iCmdShow);
    UpdateWindow(hwnd);

    hAccel = LoadAccelerators(hInstance, szAppName);

    while(GetMessage(&msg, NULL, 0, 0))
    {
        if(ghDlgModeless == NULL || !IsDialogMessage(ghDlgModeless, &msg))
        {
            if(!TranslateAccelerator(hwnd, hAccel, &msg))
            {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
            }
        }
    }

    if(gpFile)
    {
        fprintf(gpFile, "PopPad.c Program Terminated******************\n");
        fclose(gpFile);
        gpFile = NULL;
    }

    return((int)msg.wParam);
}

//DoCaption()
void DoCaption(HWND hwnd, TCHAR *szTitleName)
{
    //variable declarations
    TCHAR szCaption[64 + MAX_PATH];

    //code
    wsprintf(szCaption, TEXT("%s - %s"), szAppName, szTitleName[0] ? szTitleName : UNTITLED);

    fprintf(gpFile, "PopPad.c DoCaption(): szCaption => %s\n", szCaption);

    SetWindowText(hwnd, szCaption);

    fprintf(gpFile, "PopPad.c DoCaption(): return\n");
}

//OkMessage()
void OkMessage(HWND hwnd, TCHAR *szMessage, TCHAR *szTitleName)
{
    //variable declarations
    TCHAR szBuffer[64 + MAX_PATH];

    //code
    wsprintf(szBuffer, szMessage, szTitleName[0] ? szTitleName : UNTITLED);
    MessageBox(hwnd, szBuffer, szAppName, MB_OK | MB_ICONEXCLAMATION);
}

//AskAboutSave()
short AskAboutSave(HWND hwnd, TCHAR *szTitleName)
{
    //variable declarations
    TCHAR szBuffer[64 + MAX_PATH];
    int   iReturn;

    //code
    wsprintf(szBuffer, TEXT("Save current changes in %s?"), szTitleName[0] ? szTitleName : UNTITLED);

    iReturn = MessageBox(hwnd, szBuffer, szAppName, MB_YESNOCANCEL | MB_ICONQUESTION);

    fprintf(gpFile, "PopPad.c AskAboutSave(): szBuffer => %s\n", szBuffer);
    if(iReturn == IDYES)
    {
        fprintf(gpFile, "PopPad.c AskAboutSave(): Sending Save Command\n");
        if(!SendMessage(hwnd, WM_COMMAND, IDM_FILE_SAVE, 0))
            iReturn = IDCANCEL;
    }

    fprintf(gpFile, "PopPad.c AskAboutSave(): return\n");
    return(iReturn);
}


//WndProc()
LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    //variable declarations
    static BOOL      bNeedSave = FALSE;
    static HINSTANCE hInst;
    static HWND      hwndEdit;
    static int       iOffset;
    static TCHAR     szFileName[MAX_PATH], szTitleName[MAX_PATH];
    static UINT      messageFindReplace;

    int              iSelBeg, iSelEnd, iEnable;
    LPFINDREPLACE    pfr;

    //code
    switch(message)
    {
        case WM_CREATE:
            fprintf(gpFile, "PopPad.c WM_CREATE: Enter\n");
            hInst = ((LPCREATESTRUCT)lParam)->hInstance;

                //Create the edit control child window
            hwndEdit = CreateWindow(
                    TEXT("edit"),
                    NULL,
                    WS_CHILD | WS_VISIBLE | WS_HSCROLL | WS_VSCROLL | WS_BORDER | ES_LEFT | ES_MULTILINE |
                    ES_NOHIDESEL | ES_AUTOHSCROLL | ES_AUTOVSCROLL,
                    0, 0, 0, 0,
                    hwnd, (HMENU)EDITID, hInst, NULL
                );

            SendMessage(hwndEdit, EM_LIMITTEXT, 32000, 0L);

                //Initialize common dialog box stuff
            PopFileInitialize(hwnd);
            PopFontInitialize(hwndEdit);

            messageFindReplace = RegisterWindowMessage(FINDMSGSTRING);

            DoCaption(hwnd, szTitleName);
            fprintf(gpFile, "PopPad.c WM_CREATE return\n");
            return(0);

        case WM_SETFOCUS:
            fprintf(gpFile, "PopPad.c WM_SETFOCUS Enter\n");
            SetFocus(hwndEdit);
            fprintf(gpFile, "PopPad.c WM_SETFOCUS Return\n");
            return(0);

        case WM_SIZE:
            fprintf(gpFile, "PopPad.c WM_SIZE Enter\n");
            MoveWindow(hwndEdit, 0, 0, LOWORD(lParam), HIWORD(lParam), TRUE);
            fprintf(gpFile, "PopPad.c WM_SIZE Return\n");
            return(0);

        case WM_INITMENUPOPUP:
            switch(lParam)
            {
                case 1:     //Edit Menu
                        //Enable Undo if edit control can do it
                    fprintf(gpFile,"PopPad.c WM_INITMENUPOPUP: Edit Menu Enter\n");
                    EnableMenuItem(
                        (HMENU) wParam,
                        IDM_EDIT_UNDO,
                        SendMessage(hwndEdit, EM_CANUNDO, 0, 0L) ? MF_ENABLED : MF_GRAYED
                    );

                        //Enable Paste if text is in the clipboard
                    EnableMenuItem(
                        (HMENU)wParam,
                        IDM_EDIT_PASTE,
                        IsClipboardFormatAvailable(CF_TEXT) ? MF_ENABLED : MF_GRAYED
                    );

                        //Endable Cut, Copy, and Del if text is selected
                    SendMessage(hwndEdit, EM_GETSEL, (WPARAM)&iSelBeg, (LPARAM) &iSelEnd);

                    iEnable = ( (iSelBeg != iSelEnd) ? MF_ENABLED : MF_GRAYED );

                    EnableMenuItem((HMENU)wParam, IDM_EDIT_CUT,   iEnable);
                    EnableMenuItem((HMENU)wParam, IDM_EDIT_COPY,  iEnable);
                    EnableMenuItem((HMENU)wParam, IDM_EDIT_CLEAR, iEnable);

                    fprintf(gpFile, "PopPad.c WM_INIRMENUPOPUP: Edit Menu Return\n");
                    break;

                case 2: //Search Menu
                    //Enable Find, Next, and Replace if modeless dialogs are not already active
                    fprintf(gpFile, "PopPad.c WM_INITMENUPOPUP: Search Menu Enter\n");
                    iEnable = (ghDlgModeless == NULL) ? MF_ENABLED : MF_GRAYED;
                    EnableMenuItem((HMENU)wParam, IDM_SEARCH_FIND,    iEnable);
                    EnableMenuItem((HMENU)wParam, IDM_SEARCH_NEXT,    iEnable);
                    EnableMenuItem((HMENU)wParam, IDM_SEARCH_REPLACE, iEnable);

                    fprintf(gpFile, "PopPad.c WM_INIRMENUPOPUP: Search Menu Return\n");
                    break;
            }
            return(0);

        case WM_COMMAND:
                //messages from edit control
            if(lParam && LOWORD(wParam) == EDITID)
            {
                fprintf(gpFile, "PopPad.c WM_COMMAND: Edit Control Message Enter\n");
                switch(HIWORD(wParam))
                {
                    case EN_UPDATE:
                        bNeedSave = TRUE;
                        return(0);

                    case EN_ERRSPACE:
                    case EN_MAXTEXT:
                        MessageBox(hwnd, TEXT("Edit control out of space."), szAppName, MB_OK | MB_ICONSTOP);
                        return(0);
                }
                fprintf(gpFile, "PopPad.c WM_COMMAND: Edit Control Message Return\n");
                break;
            }

            switch(LOWORD(wParam))
            {
                //Message from File Menu
                case IDM_FILE_NEW:

                    fprintf(gpFile, "PopPad.c IDM_FILE_NEW: Enter\n");
                    if(bNeedSave && IDCANCEL == AskAboutSave(hwnd, szTitleName))
                    {
                        fprintf(gpFile, "PopPad.c IDM_FILE_NEW: Cancel Button Clicked\n");
                        return(0);
                    }

                    SetWindowText(hwndEdit, TEXT("\0"));
                    szFileName[0] = '\0';
                    szTitleName[0] = '\0';
                    DoCaption(hwnd, szTitleName);
                    bNeedSave = FALSE;

                    fprintf(gpFile, "PopPad.c IDM_FILE_NEW: Save Button Clicked\n");
                    return(0);

                case IDM_FILE_OPEN:
                    fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Enter\n");
                    if(bNeedSave && IDCANCEL == AskAboutSave(hwnd, szTitleName))
                    {
                        fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: AskAboutSave() => cancel Button clicked\n");
                        return(0);
                    }

                    if(PopFileOpenDlg(hwnd, szFileName, szTitleName))
                    {
                        fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: File Opened\n");
                        fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Reading File\n");
                        if(!PopFileRead(hwndEdit, szFileName))
                        {
                            fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Error while reading\n");
                            OkMessage(hwnd, TEXT("Could not read file %s!"), szTitleName);
                            szFileName[0] = '\0';
                            szTitleName[0] = '\0';
                        }
                    }

                    DoCaption(hwnd, szTitleName);
                    bNeedSave = FALSE;

                    fprintf(gpFile, "PopPad.c IDM_FILE_OPEN: Returned\n");
                    return(0);

                case IDM_FILE_SAVE:
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Enter\n");
                    if(szFileName[0])
                    {
                        if(PopFileWrite(hwndEdit, szFileName))
                        {
                            bNeedSave = FALSE;
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Data written to file \"%s\"\n", szFileName);
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Return\n");
                            return(1);
                        }
                        else
                        {
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Error while writing data in file \"%s\"\n", szFileName);
                            OkMessage(hwnd, TEXT("1Could not write file %s"), szTitleName);
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Return\n");
                            return(0);
                        }
                    }
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE: Fall Through to IDM_FILE_SAVE_AS\n");
                    //fall through

                case IDM_FILE_SAVE_AS:
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Enter\n");
                    if(PopFileSaveDlg(hwnd, szFileName, szTitleName))
                    {
                        DoCaption(hwnd, szTitleName);

                        if(PopFileWrite(hwndEdit, szFileName))
                        {
                            bNeedSave = FALSE;
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: File save\n");
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Return\n");
                            return(1);
                        }
                        else
                        {
                            OkMessage(hwnd, TEXT("2Could not write file %s"), szTitleName);
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Error while saving\n");
                            fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Return\n");
                            return(0);
                        }
                    }
                    fprintf(gpFile, "PopPad.c IDM_FILE_SAVE_AS: Return\n");
                    return(0);

                case IDM_FILE_PRINT:
                    fprintf(gpFile, "PopPad.c IDM_FILE_PRINT: Enter\n");
                    if(!PopPrntPrintFile(hInst, hwnd, hwndEdit, szTitleName))
                        OkMessage(hwnd, TEXT("Could not print file %s"), szTitleName);

                    fprintf(gpFile, "PopPad.c IDM_FILE_PRINT: Return\n");
                    return(0);

                case IDM_APP_EXIT:
                    fprintf(gpFile, "PopPad.c IDM_APP_EXIT\n");
                    SendMessage(hwnd, WM_CLOSE, 0, 0);
                    return(0);
                    //Messages from Edit Menu
                case IDM_EDIT_UNDO:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_UNDO\n");
                    SendMessage(hwndEdit, WM_UNDO, 0, 0);
                    return(0);

                case IDM_EDIT_CUT:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_CUT\n");
                    SendMessage(hwndEdit, WM_CUT, 0, 0);
                    return(0);

                case IDM_EDIT_COPY:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_COPY\n");
                    SendMessage(hwndEdit, WM_COPY, 0, 0);
                    return(0);

                case IDM_EDIT_PASTE:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_PASTE\n");
                    SendMessage(hwndEdit, WM_PASTE, 0, 0);
                    return(0);

                case IDM_EDIT_CLEAR:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_CLEAR\n");
                    SendMessage(hwndEdit, WM_CLEAR, 0, 0);
                    return(0);

                case IDM_EDIT_SELECT_ALL:
                    fprintf(gpFile, "PopPad.c IDM_EDIT_SELECT_ALL\n");
                    SendMessage(hwndEdit, EM_SETSEL, 0, -1);
                    return(0);


                        //Messages from Search Menu
                case IDM_SEARCH_FIND:
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_FIND: Enter\n");
                    SendMessage(hwndEdit, EM_GETSEL, 0, (LPARAM) &iOffset);
                    ghDlgModeless = PopFindFindDlg(hwnd);
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_FIND: Return\n");
                    return(0);

                case IDM_SEARCH_NEXT:
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_NEXT: Enter\n");
                    SendMessage(hwndEdit, EM_GETSEL, 0, (LPARAM) &iOffset);

                    if(PopFindValidFind())
                        PopFindNextText(hwnd, &iOffset);
                    else
                        ghDlgModeless = PopFindFindDlg(hwnd);

                    fprintf(gpFile, "PopPad.c IDM_SEARCH_NEXT: Return\n");

                    return(0);

                case IDM_SEARCH_REPLACE:
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_REPLACE: Enter\n");
                    SendMessage(hwndEdit, EM_GETSEL, 0, (LPARAM) &iOffset);
                    ghDlgModeless = PopFindReplaceDlg(hwnd);
                    fprintf(gpFile, "PopPad.c IDM_SEARCH_REPLACE: Return\n");
                    return(0);

                case IDM_FORMAT_FONT:
                    fprintf(gpFile, "PopPad.c IDM_FORMAT_FONT: Enter\n");
                    if(PopFontChooseFont(hwnd))
                        PopFontSetFont(hwndEdit);
                    fprintf(gpFile, "PopPad.c IDM_FORMAT_FONT: Return\n");

                    return(0);
```

```
                //Messages from Help Menu
            case IDM_HELP:
                fprintf(gpFile, "PopPad.c IDM_HELP\n");
                OkMessage(hwnd, TEXT("Help not yet implemented!"), TEXT("\0"));
                return(0);

            case IDM_APP_ABOUT:
                fprintf(gpFile, "PopPad.c IDM_APP_ABOUT: Enter\n");

                DialogBox(
                    hInst,
                    TEXT("ABOUTDLGBOX"),
                    hwnd,
                    AboutDlgProc
                );

                fprintf(gpFile, "PopPad.c IDM_APP_ABOUT: Return\n");
                return(0);
        }
        break;

    case WM_CLOSE:
        fprintf(gpFile, "PopPad.c WM_CLOSE: Enter\n");
        if(!bNeedSave || IDCANCEL != AskAboutSave(hwnd, szTitleName))
            DestroyWindow(hwnd);
        fprintf(gpFile, "PopPad.c WM_CLOSE: Return\n");

        return(0);

    case WM_QUERYENDSESSION:
        fprintf(gpFile, "PopPad.c WM_QUERYENDSESSION\n");
        if(!bNeedSave || IDCANCEL != AskAboutSave(hwnd, szTitleName))
            return(1);
        return(0);

    case WM_DESTROY:
        fprintf(gpFile, "PopPad.c WM_DESTROY\n");
        PopFontDeinitialize();
        PostQuitMessage(0);
        return(0);


    default:
        //Process "Find-Replace message
        if(message == messageFindReplace)
        {
            fprintf(gpFile, "PopPad.c messageFindReplace: Enter\n");
            pfr = (LPFINDREPLACE)lParam;

            if(pfr->Flags & FR_DIALOGTERM)
                ghDlgModeless = NULL;

            if(pfr->Flags & FR_FINDNEXT)
                if(!PopFindFindText(hwndEdit, &iOffset, pfr))
                    OkMessage(hwnd, TEXT("Text not found!"), TEXT("\0"));

            if(pfr->Flags & FR_REPLACE || pfr->Flags & FR_REPLACEALL)
                if(!PopFindReplaceText(hwndEdit, &iOffset, pfr))
                    OkMessage(hwnd, TEXT("Text not found!"), TEXT("\0"));

            if(pfr->Flags & FR_REPLACEALL)
                while(PopFindReplaceText(hwndEdit, &iOffset, pfr));

            fprintf(gpFile, "PopPad.c messageFindReplace: return\n");
            return(0);
        }
        break;
    }

    return(DefWindowProc(hwnd, message, wParam, lParam));
}


//AboutDlgProc()
BOOL CALLBACK AboutDlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    //code
    switch(message)
    {
        case WM_INITDIALOG:
            fprintf(gpFile, "PopPad.c WM_INITDIALOG\n");
            return(TRUE);

        case WM_COMMAND:
            switch(LOWORD(wParam))
            {
                case IDOK:
                    fprintf(gpFile, "PopPad.c WM_COMMAND: IDOK\n");
                    EndDialog(hDlg, 0);
                    return(TRUE);
            }
            break;
    }
    return(FALSE);
}
```