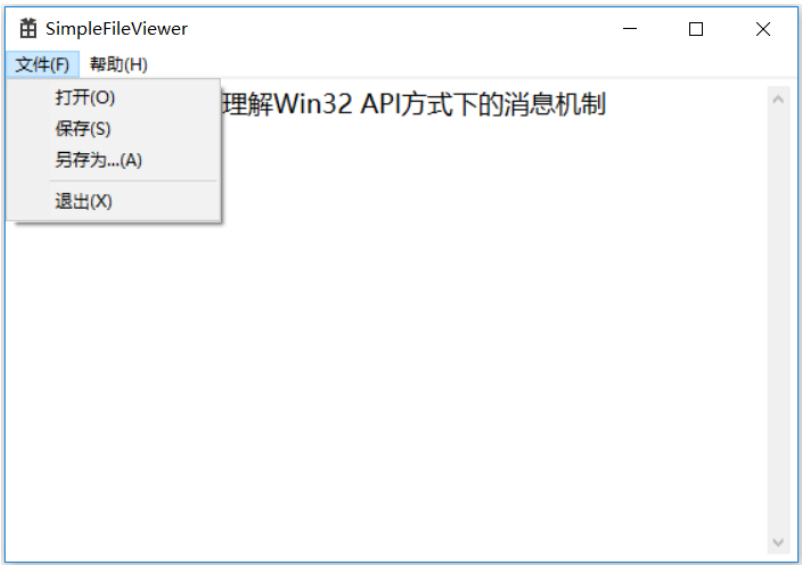
实验一　理解 Windows 窗口及消息机制

实验指导

首先，在 Visual Studio 2015 中的 VC++开发环境中，构建一个“Win32 项目(Win32 Application)”类型的 Windows 窗口程序框架，项目名称和保存路径可以根据需要自定义，对构建的程序框架进行编译并运行；

在“解决方案资源管理器”中找到“资源文件”中的“.rc”文件，并双击打开该文件，进入到“资源视图”后，选择“Menu”下的 IDC_开头的菜单文件，并双击打开该文件，然后在右侧的菜单编辑器中，为“文件”菜单添加“打开”、“保存”和“另存为”三个子菜单项，并在“属性”窗口分别将这三个子菜单项的 ID 命名为：IDM_OPEN、IDM _SAVE 和 IDM _SAVEAS，菜单项如下图所示，最后保存以上结果；



接下来为项目添加一个简单的 C++文件操作类。在“解决方案资源管理器”中选中你的项目名称，并点击鼠标右键打开快捷菜单，选择其中的“添加—>类”，在打开的对话框中选择“C++类”，点击“添加”按钮，在界面中的“类名”文本框中填入你取的类名，系统将自动将类的.h 头文件和.cpp 源文件自动按照你取的类名进行命名，点击“完成”按钮，在“解决方案资源管理器”中的“头文件”和“源文件”中将会列出刚才添加的类文件。分别打开刚才添加的头文件和源文件，添加必要的文件操作代码，主要是文件打开和文件保存的函数代码。

假如你的类为“SimpleFile”，则头文件中的参考代码如下（以下代码非文本，请按照内容自己输入）：

```
#pragma once
#include <Windows.h>
#include <FSTREAM>
#include <VECTOR>
#include <STRING>
#include <sstream>
using namespace std;

typedef vector<wstring> vecLinesContainer;

class SimpleFile
{
public:
   SimpleFile();

   BOOLEAN ReadFile(WCHAR* pathName);
   BOOLEAN WriteFile(WCHAR* pathName, WCHAR* fileLines);

   vecLinesContainer getFileLines(WCHAR* pathName);

private:
   vecLinesContainer FileContent;
};
```

源文件中的函数定义参考代码如下（以下代码非文本，请按照内容自己输入）：

```
BOOLEAN SimpleFile::ReadFile(WCHAR * pathName)
{
    wifstream inFile(pathName, ios::in);
    if (!inFile)
    {
        return FALSE;
    }
    inFile.imbue(locale(locale(), "", LC_CTYPE));
    wstring buffer;
    while (inFile.good())
    {
        std::getline(inFile, buffer);
        FileContent.push_back(buffer);
    }
    inFile.close();

    return TRUE;
}

BOOLEAN SimpleFile::WriteFile(WCHAR * pathName, WCHAR* fileLines)
{
    if (fileLines != NULL)
    {
        wstringstream mySS;
        mySS << fileLines;
        wofstream outFile(pathName, ios::out | ios::trunc);
        if (!outFile)
        {
            return FALSE;
        }
        outFile.imbue(locale(locale(), "", LC_CTYPE));
        wstring myString = mySS.str();
        outFile.write(myString.c_str(), wcslen(fileLines));
        outFile.close();
    }

    return TRUE;
}
```

```
vecLinesContainer SimpleFile::getFileLines(WCHAR* pathName)
{
    ReadFile(pathName);
    return FileContent;
}
```

下面将在自动生成的框架代码中添加多行文本框创建代码和必要的消息处理代码，实现对以上文件操作类的调用，使框架代码具备文本文件打开、保存和另存为的功能。代码只在自动生成的.cpp 源文件文件中添加。

1. **添加头文件引用，代码如下：**

```
#include <commdlg.h>
#include "SimpleFile.h"
```

说明：

SimpleFile.h 你上面定义的文件操作类的头文件，如果不是这个名字，请改成你定义的名字。

2. **在框架代码的变量和函数声明的最后添加如下三个变量声明：**

```
vecLinesContainer fileLines;    // 读取的文件内容
HWND hWndEdit;                  // 文本框句柄
WCHAR currentFile[260];         //  文件名
```

3. **在框架代码的“BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)”函数中，添加如下创建多行文本框、设置字体字号以及消息发送等代码：**

```
RECT rect;
GetClientRect(hWnd, &rect);
hWndEdit = CreateWindowW(
    L"EDIT", L"简单文本编辑器：理解Win32 API方式下的消息机制",
    WS_VISIBLE | WS_CHILD | ES_LEFT| ES_MULTILINE |
    ES_WANTRETURN| WS_VSCROLL | ES_AUTOVSCROLL,
    rect.left + 5, rect.top + 5,
    rect.right - 10, rect.bottom-10,
    hWnd, nullptr, hInstance, nullptr);
```

```
HFONT hFont = CreateFont(
        24, 0, 0, 0, 300, false, false, false,
        DEFAULT_CHARSET, OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH,
        L"微软雅黑"
    );
    SendMessage(hWndEdit, WM_SETFONT, (WPARAM)hFont, TRUE);
```

4. **为框架代码的消息处理函数“LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)”添加新的消息处理函数和变量声明。**

a) 在函数的最前面，也就是其中的 switch 语句的前面，添加如下变量声明：

```
OPENFILENAME ofn;            // OpenFile通用结构体
WCHAR szFile[260] = { 0 };   // 文件名
int cTxtLen;                 // 文本长度
WCHAR* allFileLines;         // 文本内容
```

b) 在 switch (message)中的 case WM_COMMAND 消息处理（即菜单消息处理）中添加三个 case 语句，分别处理文件打开、保存和另存为，这三个 case 语句如下：

```
case IDM_OPEN:
  break;
case IDM_SAVE:
  break;
case IDM_SAVEAS:
  break;
```

说明：

以上三个语句必须添加在 default 语句之前，每个 case 中的 IDM_开头的标识是在添加菜单项时，在属性的 ID 中取的标识名

c) 分别为这三个 case 语句添加处理代码，参考代码如下（以下代码非

文本，请按照内容自己输入）：

```
case IDM_OPEN:
   // 初始化OPENFILENAME结构体
   ZeroMemory(&ofn, sizeof(ofn));
   ofn.lStructSize = sizeof(ofn);
   ofn.hwndOwner = hWnd;
   ofn.lpstrFile = currentFile;
   ofn.nMaxFile = sizeof(currentFile);
   ofn.lpstrFilter = _T("All\0*.*\0Text\0*.TXT\0");
   ofn.nFilterIndex = 1;
   ofn.lpstrFileTitle = NULL;
   ofn.nMaxFileTitle = 0;
   ofn.lpstrInitialDir = NULL;
   ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;

   if (GetOpenFileName(&ofn) == TRUE)
   {
      SetWindowText(hWnd, ofn.lpstrFile);
      SimpleFile simplefile;
      fileLines = simplefile.getFileLines(ofn.lpstrFile);

      vecLinesContainer::iterator it;
      std::wstring allLines = L"";
      for (it = fileLines.begin(); it != fileLines.end(); it++)
      {
         allLines.append(*it);
         allLines.append(L"\r\n");
      }

      SetWindowText(hWndEdit, allLines.c_str());

   }
   break;
```

```
case ID_SAVE:
    SetWindowText(hWnd, currentFile);
    cTxtLen = GetWindowTextLength(hWndEdit);
    allFileLines =
        (LPWSTR)VirtualAlloc((LPVOID)NULL,
                             (DWORD)(cTxtLen + 1),
                             MEM_COMMIT,
                             PAGE_READWRITE);
    GetWindowText(hWndEdit, allFileLines, cTxtLen + 1);
    if (allFileLines != NULL)
    {
        SimpleFile simplefile;
        simplefile.WriteFile(currentFile, allFileLines);
    }
    break;
```

```
case IDM_SAVEAS:
   // 初始化OPENFILENAME结构体
   ZeroMemory(&ofn, sizeof(ofn));
   ofn.lStructSize = sizeof(ofn);
   ofn.hwndOwner = hWnd;
   ofn.lpstrFile = szFile;
   ofn.nMaxFile = sizeof(szFile);
   ofn.lpstrFilter = _T("All\0*.*\0Text\0*.TXT\0");
   ofn.nFilterIndex = 1;
   ofn.lpstrFileTitle = NULL;
   ofn.nMaxFileTitle = 0;
   ofn.lpstrInitialDir = NULL;
   ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;
   if (GetSaveFileName(&ofn) == TRUE)
   {

      cTxtLen = GetWindowTextLength(hWndEdit);
      allFileLines =
         (LPWSTR)VirtualAlloc((LPVOID)NULL,
                              (DWORD)(cTxtLen + 1),
                              MEM_COMMIT,
                              PAGE_READWRITE);
      GetWindowText(hWndEdit, allFileLines, cTxtLen + 1);
      if (allFileLines != NULL)
      {
         SimpleFile simplefile;
         simplefile.WriteFile(ofn.lpstrFile, allFileLines);
      }
   }
   break;
```