**暨南大学本科实验报告专用纸**

课程名称 Windows编程实验 指导教师 刘晓翔 成绩

实验项目名称 MFC单文档界面框架程序 实验项目编号

实验项目类验证实验地点三楼机房1学院智能科学与工程学院 专业 人工智能

学生姓名 巫扬智 学号 2021102258 实验时间 2023 年 4 月 13 日

1. **实验目的**

学习和掌握单文档界面框架程序（SDI）的基本结构与初始化过程。

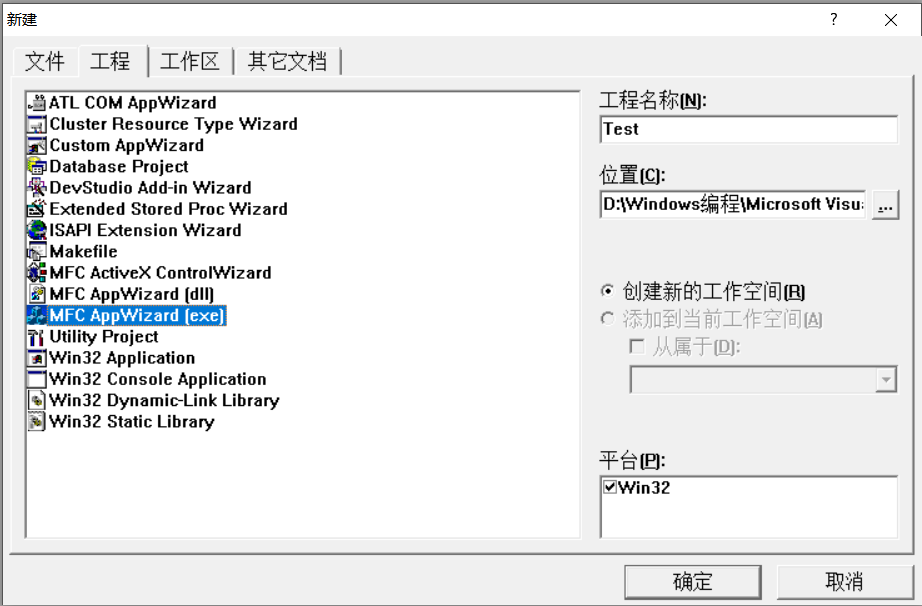
1. **实验要求**

利用Visual C++6.MFC APP Wizard生成一个单文档界面框架程序（SDI）。通过设置断点、单步跟踪调试的方式，剖析MFC框架程序的初始化过程。

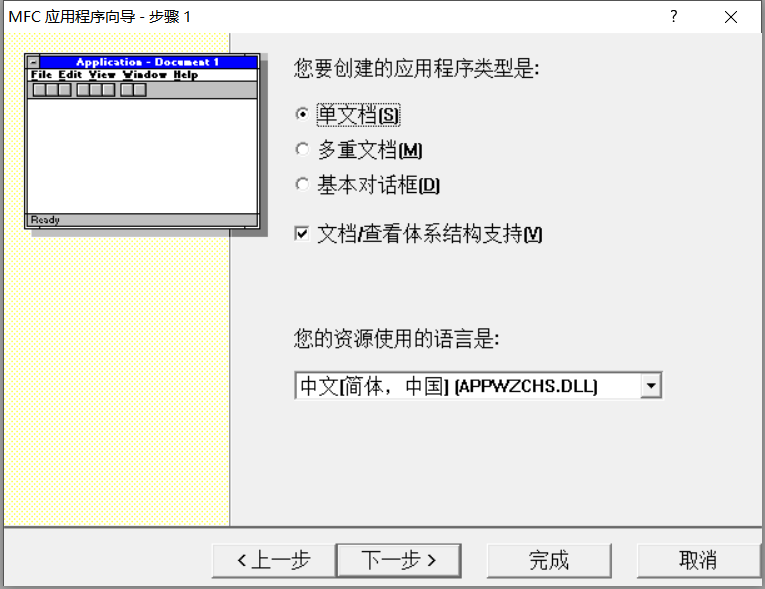
1. **实验过程**

3.1建立一个基于MFC的单文档界面（SDI)应用程序

3.1.1新建文件，在projects选项下选择MFC AppWizard(exe)，在右侧输入工程名Test



3.1.2 选择单文档界面，接下来一直默认点击next即可

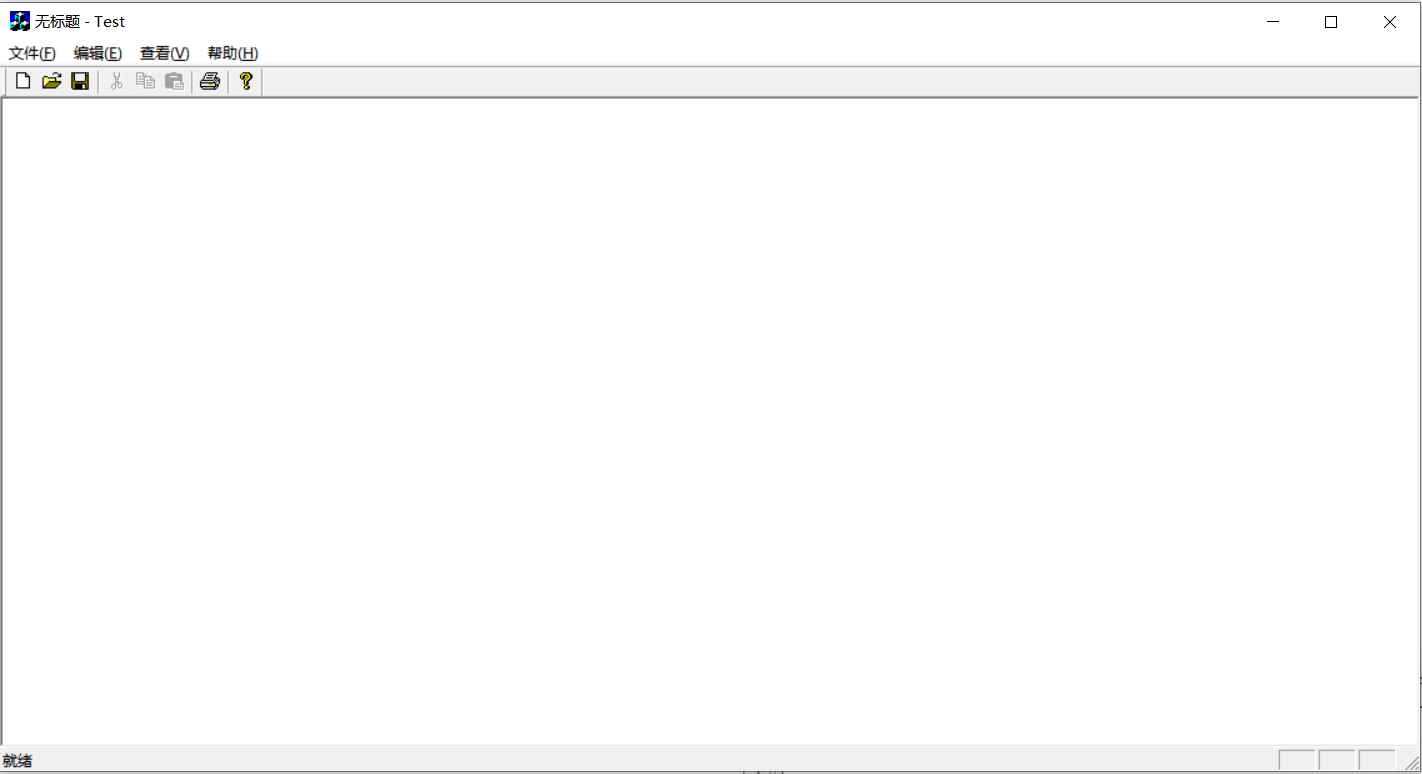


3.1.3 单文档应用程序(SDI)和多文档应用程序(MDI)的区别

SDI:用户在其中只能打开一个文档。如果用户尝试打开另一个文档，当前文档将被关闭。

MDI:用户可以同时打开多个文档，每个文档都在应用程序的主窗口中显示为一个单独的子窗口。

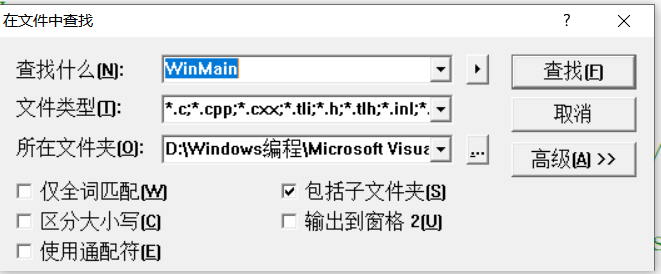
3.1.4 编译和运行程序



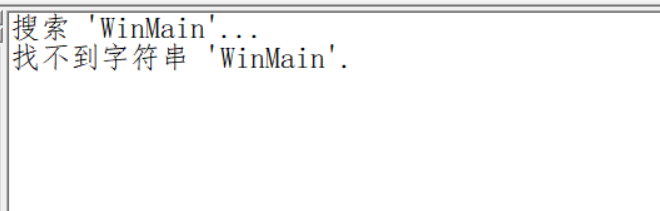
至此，一个基于MFC的单文档界面应用程序被创建出来

3.2 寻找WinMain函数

3.2.1在工程文件中查找WinMain函数



无法查找到

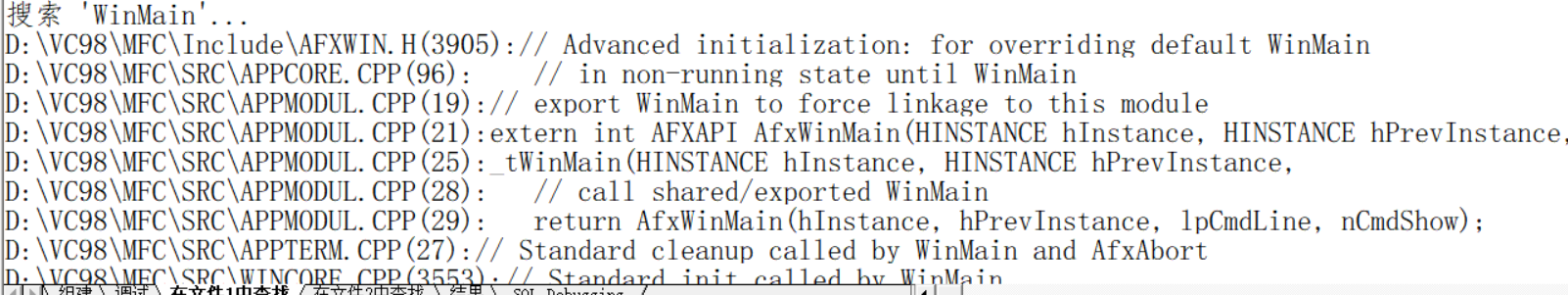


原因：MFC的底层框架封装了WinMain函数，CreateWindow函数等每一个窗口程序都需要的步骤，它们被隐藏在MFC的安装目录中。这些函数是在程序编译链接时，由编译器将这些函数链接到程序中。

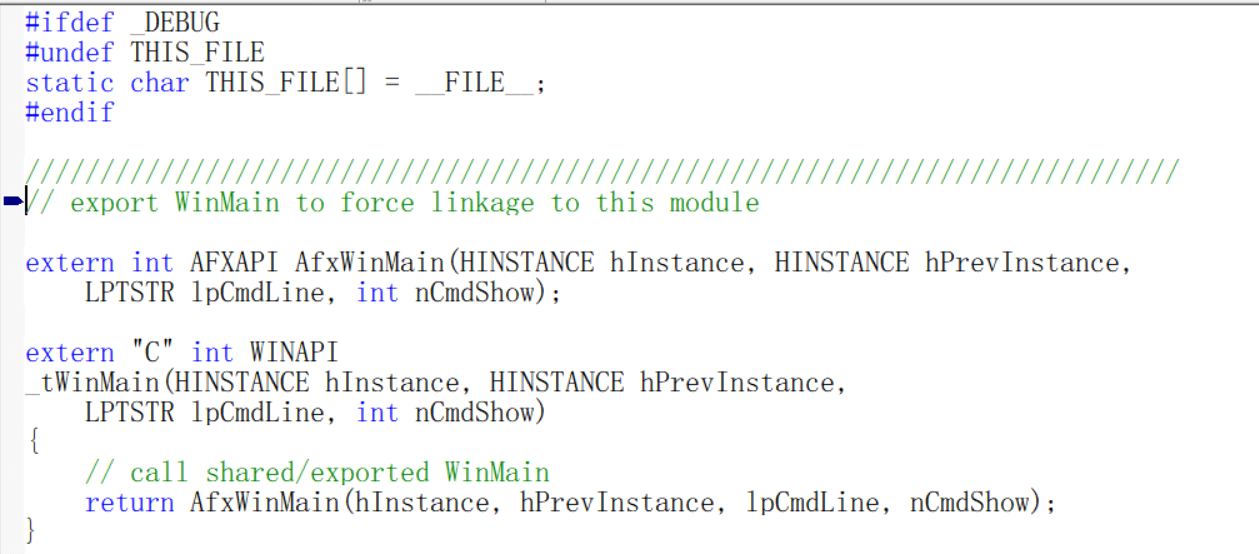
3.2.2 在MFC的安装目录中查找WinMain函数



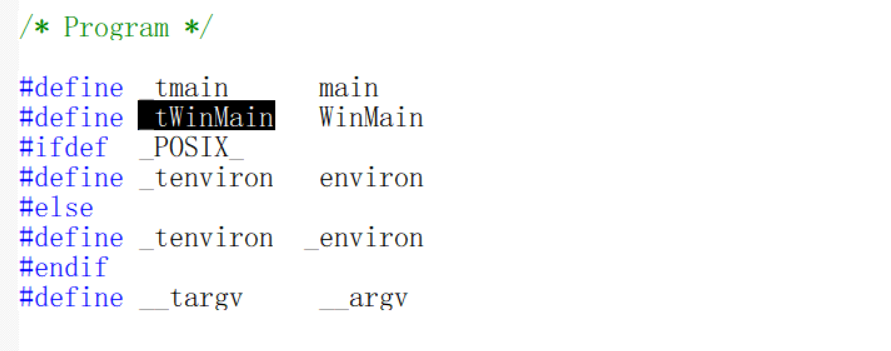
成功查找到



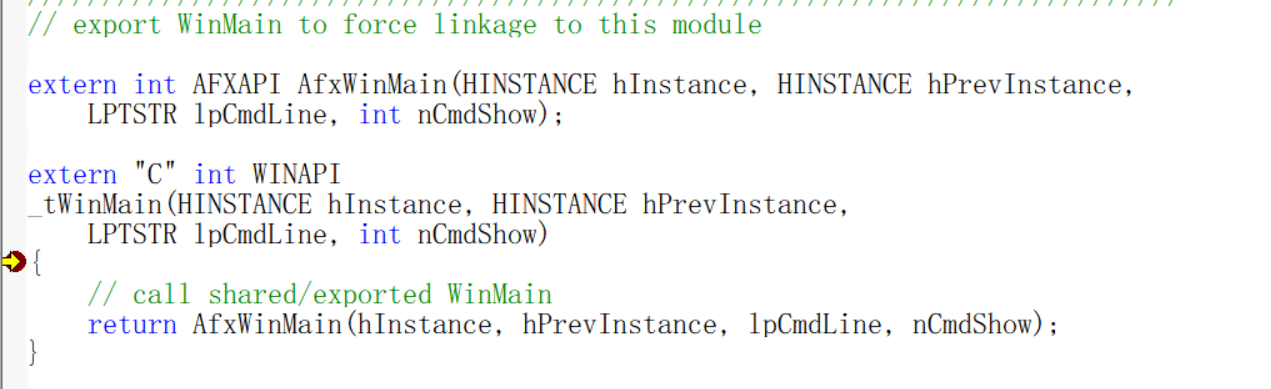
点击APPMODULAPP.CPP，打开该文件



但是这里的WinMain函数的名字是\_tWinMain，点击右键转到宏定义



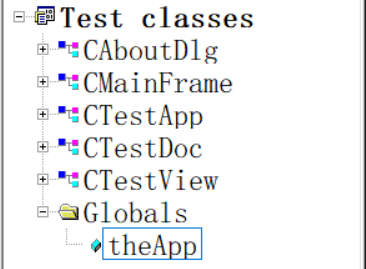
发现这就是WinMain函数，设置断点运行程序



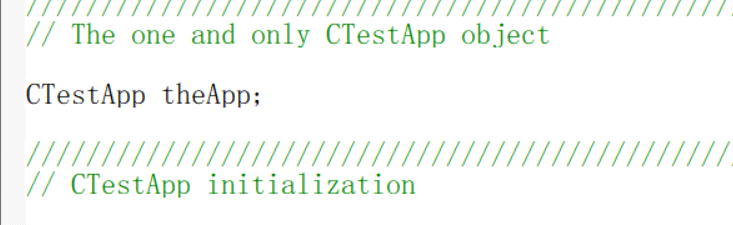
程序从这个地方暂停，说明这个WinMain函数是此程序的一部分

3.3 分析WinMain函数如何与MFC程序联系

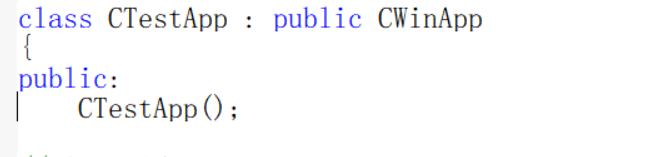
3.3.1 theApp全局对象



点开theApp.cpp,发现创建了一个CTestApp对象

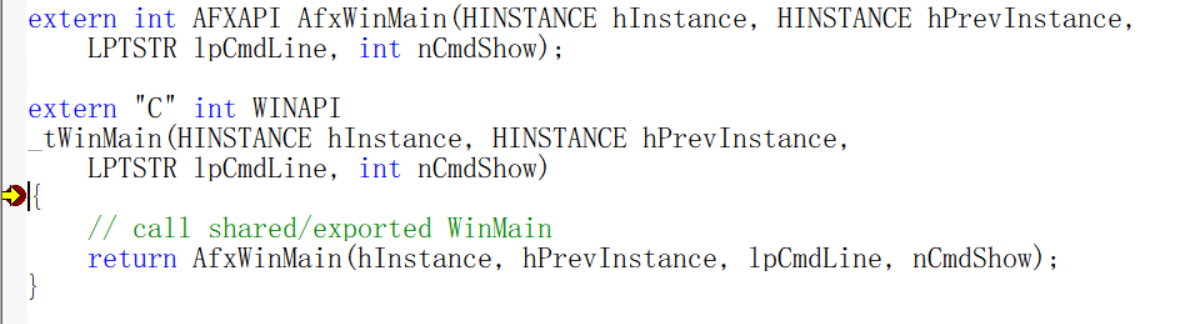


点击CTestApp，跳转到该类的定义文件(Test.h),发现该类继承自CWinApp类



在这个派生类的构造函数中设置一个断点，调试运行，发现程序在此停止（如下图所示）

这说明，这个全局对象的创建比WinMain函数还早（WinMain函数的断点还在）



Ps:在学习Win32 SDK应用程序时，应用程序的实例是由应用程序的实例句柄（WinMain函数的hInstance参数）来表示的。而对于MFC程序来说，通过产生一个应用程序类的对象来唯一标识应用程序的实例。每一个MFC程序有且仅有一个从应用程序（CWinAPP）派生的类，也就是这里的CTestApp类。每一个MFC程序实例有且仅有一个派生类的实例对象，也就是theApp全局对象。theApp对象创建完成后，应用程序也就开始了，theApp生命周期结束后，应用程序也就结束了。该对象代表了应用的程序本生，这个作用很像其他面向对象编程里面的引导类。

3.3.2 类CTestApp的父类CWinApp

查找CWinApp,查找到源文件:appcore.cpp，打开它



里面有这样一句代码

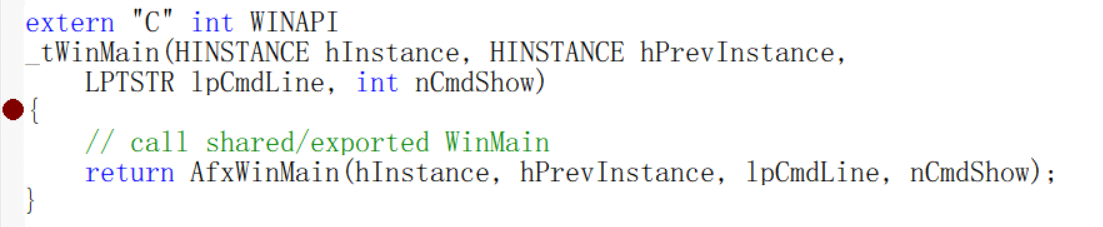


根据C++的继承性原理，这个this对象代表的是子类CTestApp的对象，即theApp，那这有什么用呢，

在后面再讨论

3.3.3 AfxWinMain函数

WinMain函数其实是调用AfxWinMain函数来完成它的功能的



Ps:Afx为前缀的函数都是全局函数，可以在程序中任何地方调用它们。

查找AfxWinMain函数，它在WINMAIN.CPP文件中



打开WINMAIN.CPP文件，找到下面两句代码

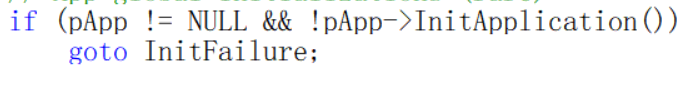


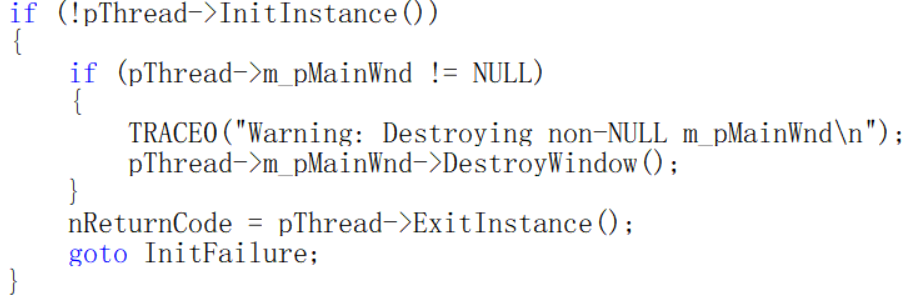
刚刚在CwinApp中，这三句代码



指明了这两个函数返回的都是this指针，而this指针指向的是theApp全局对象则pThread和pApp的值都是this，但是由于CwinApp派生于CWinThread类，根据多态性，它们虽然是同一个值，但是却可以做不同的事。

回到AfxWinMain的代码，可以看到，pThread和pApp先后调用了3个函数，这三个函数完成了Win32程序需要的几个步骤：设计窗口类、注册窗口类、创建窗口、显示窗口、更新窗口、消息循环以及窗口过程函数。



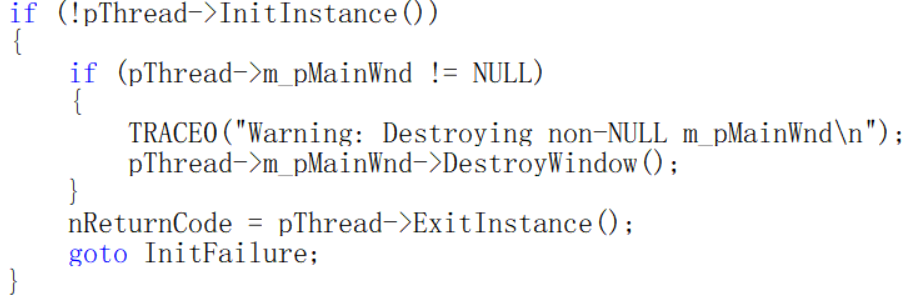




在这里总结一下，WinMain函数调用AfxWinMain函数完成它的功能。AfxWinMain函数通过获得this指针（指向theApp全局对象）以及利用多态性，来调用3个函数来完成它的功能。

3.3.4 寻找设计窗口类和注册窗口类

在AfxWinMain函数中，pThread->InitInstance()调用InitInstance()函数完成窗口的设计和注册



由于这里的pThread指向的是theApp对象，并且CTestApp中有这个声明



这是一个虚函数，根据多态性，pThread调用的是CTestApp里的InitInstance函数

CTestApp里的InitInstance里面的定义里有一句代码



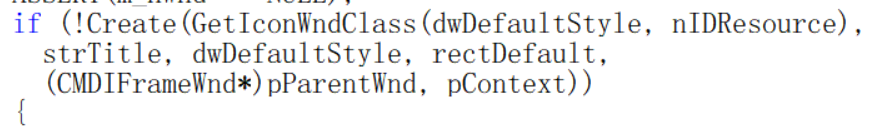
注：ProcessShellCommand(cmdInfo)会new一个主框类对象，从而调用主框类函数CFrameWnd::LoadFrame()

CFrameWnd:: LoadFrame的定义在WinFrm.cpp中



CFrameWnd:: LoadFrame中的AfxDeferRegisterClass函数会调用API函数::RegisterClass()来注册主框架窗口。至此，问题二的窗口类注册问题得以解决！（找到RegisterClass）

CFrameWnd:: LoadFrame中的Create函数会调用基类的窗口创建函数CWnd::CreateEx()。



CWnd::CreateEx 的定义在WinCore.cpp中在CreateEx函数内部：首先会调用PreCreateWindow(cs)函数，给设计人员一个更改窗口风格的机会。然后调用API函数::CreateWindowEx创建窗口

InitIstance函数中，下面两行代码负责显示窗口和更新窗口



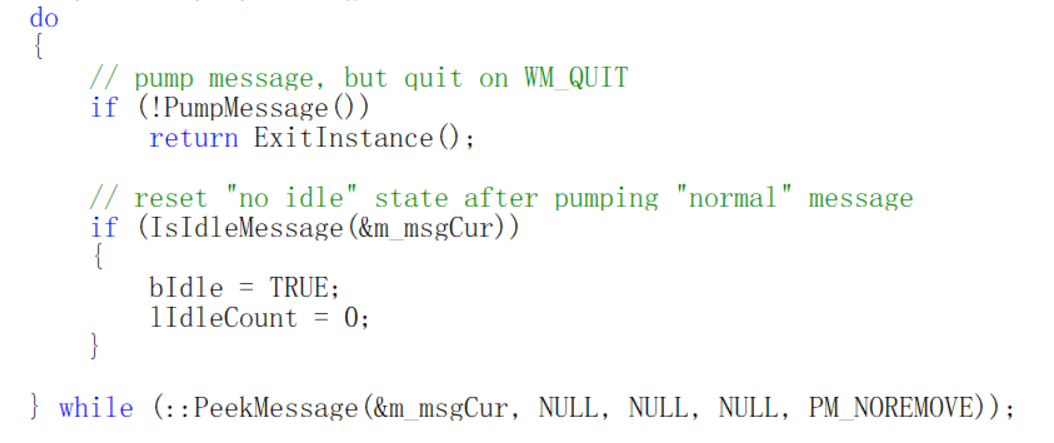
3.3.5 寻找消息循环

在AfxWinMain函数中，有下面一句代码

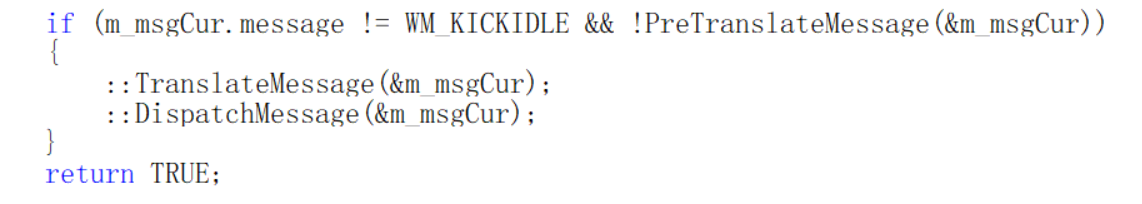


pThread调用了Run函数，Run函数负责创建一个消息循环

下面是Run函数的定义，里面有这样一句代码



Run函数调用了PumpMessage函数，下面是PumpMeaasge函数的部分定义



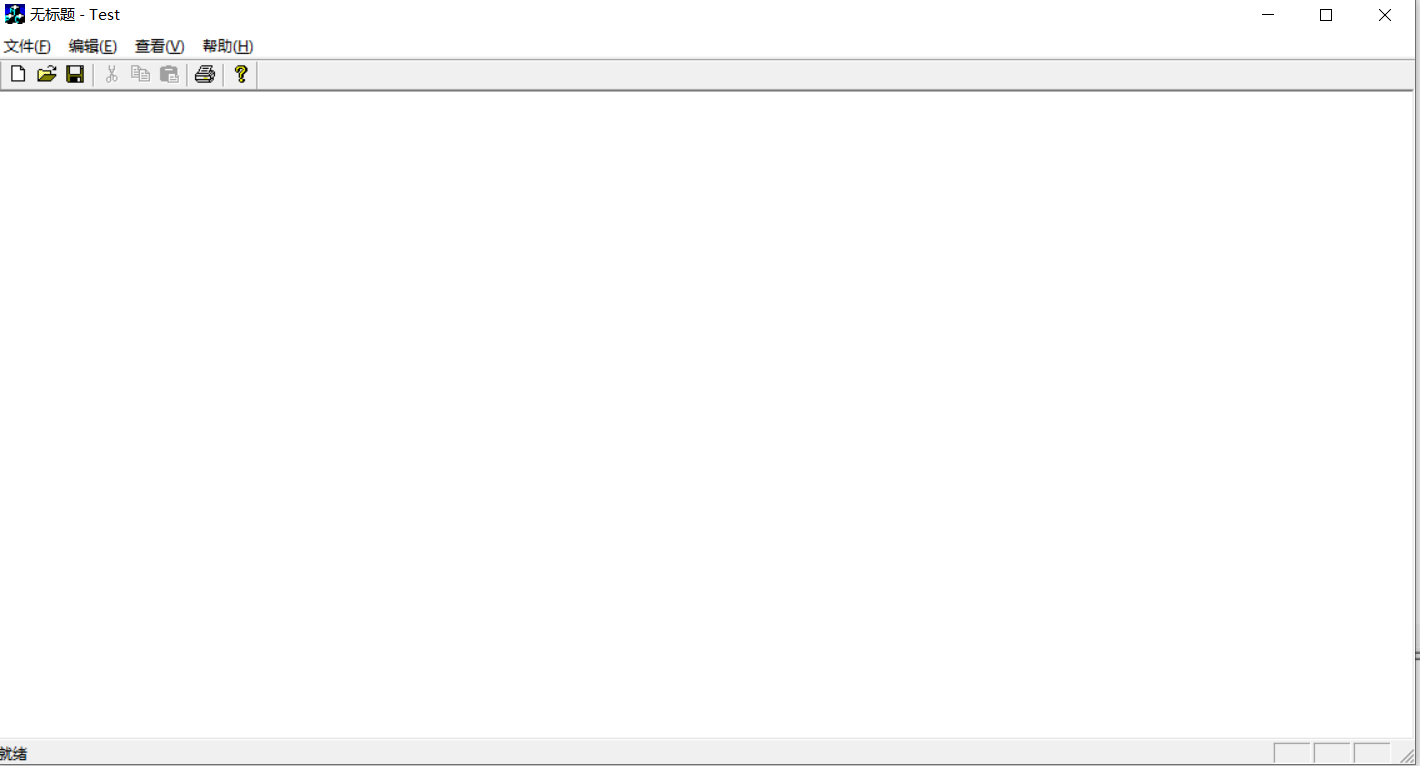
里面有一个消息循环的定义，至此，消息循环找到

3.3.5 窗口过程函数

MFC采用了一种称之为**消息映射的机制**来处理各种消息。Mapping（消息映射表）的做法，把消息和其处理程序关联起来。

3.3.6 文档/视类结构

MFC程序中有3个大类，主框架类、视图类和文档类。其中，主框架类和视图类派生于CWnd类，文档类派生于CCmdTarget类，而CCmdTarget派生于CObject类。数据的存储和加载由文档类完成，数据的显示和修改则由视类完成，这种结构就是文档/视类结构。



整个窗口是主框架窗口，而白色的那一部分，则是视图窗口。视图窗口是主框架窗口的子窗口。

1. **实验总结**

将MFC的运行过程再梳理一遍

1. 首先创建全局对象theApp,利用theApp对象启动程序。创建theApp对象时，调用它的构造函数，这会完成一些初始化工作。
2. 然后进入WinMain函数，WinMain函数调用AfxWinMain完成它的功能。AfxWinMain函数会获得theApp对象的指针，并且调用3个函数InitApplication、InitInstance和Run函数。其中InitApplication完成一些初始化工作，InitInstance完成设计窗口类（使用默认窗口），注册窗口类，显示窗口和更新窗口的功能。而Run函数负责消息循环和窗口过程函数的创建。
3. 进入消息循环，程序正式开始。

**暨南大学本科实验报告专用纸(附页)**