華中科技大學

课 程 实 验 报 告

课程名称： 汇编语言程序设计实验

实验名称： 实验五　WIN32 编程

实验时间： 2019-5-8，8：00-11：50　**实验地点：** 南一楼 803 室

指导教师： 李专

专业班级： CSIE1701 班

学　　号： U201715264　**姓　　名：** 邹子一

同组学生： 无　**报告日期：** 2019 年　5 月 8 日

原创性声明

本人郑重声明：本报告的内容由本人独立完成，有关观点、方法、数据和文献等的引用已经在文中指出。除文中已经注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品或成果，不存在剽窃、抄袭行为。

特此声明！

学生签名：

日期：2019.5.8

成绩评定

实验完成质量得分（70 分）（实验步骤清晰详细深入，实验记录真实完整等）	报告撰写质量得分（30 分）（报告规范、完整、通顺、详实等）	总成绩（100 分）

指导教师签字：

日期：

目录

实验目的与要求

(1) 熟悉 WIN32 程序的设计和调试方法；

(2) 熟悉宏汇编语言中 INVOKE、结构变量、简化段定义等功能；

(3) 进一步理解机器语言、汇编语言、高级语言之间以及实方式、保护方式之间的一些关系。

实验内容

编写一个基于窗口的 WIN32 程序，实现网店商品信息管理程序的推荐度计算及商品信息显示的功能.

实验过程

1 任务 1

1.1 设计思路

在现有框架上修改，加入自己的功能，实现菜单显示功能。

1.2 任务 1 源程序

Menu.rc:

```
#define IDM_FILE_EXIT   10001
#define IDM_ACTION_REC   10010
#define IDM_ACTION_LIST   10011
#define IDM_ACTION_CLEAR  10012
#define IDM_HELP_ABOUT  10101

MyMenu  MENU
BEGIN
    POPUP "&File"
    BEGIN
            MENUITEM "E&xit", IDM_FILE_EXIT
    END
    POPUP "&Action"
    BEGIN
            MENUITEM "R&ecommendation", IDM_ACTION_REC
            MENUITEM "L&ist", IDM_ACTION_LIST
            MENUITEM "C&lear",IDM_ACTION_CLEAR
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&About",IDM_HELP_ABOUT
    END
END
```

窗口消息处理程序：

```
.IF     uMsg == WM_DESTROY
        invoke PostQuitMessage,NULL
    .ELSEIF uMsg == WM_KEYDOWN
       .IF     wParam == VK_F1
           ;;your code
       .ENDIF
    .ELSEIF uMsg == WM_COMMAND
```

```
        .IF     wParam == IDM_FILE_EXIT
            invoke SendMessage,hWnd,WM_CLOSE,0,0
        .ELSEIF wParam == IDM_ACTION_LIST
            mov menuItem, 1
            invoke InvalidateRect,hWnd,0,1
            invoke UpdateWindow, hWnd
        .ELSEIF wParam == IDM_ACTION_CLEAR
            mov menuItem, 0
            invoke InvalidateRect,hWnd,0,1
            invoke UpdateWindow, hWnd
        .ELSEIF wParam == IDM_HELP_ABOUT
            invoke MessageBox,hWnd,addr AboutMsg,addr AppName,0
        .ELSEIF wParam == IDM_ACTION_REC
            mov gi,0
            mov ecx,N
            RECLOOP:
                movzx edx,gi
                mov ebx,offset buf.GoodName
                add ebx,edx
                invoke CALCREC,ebx
                add gi,21
                loop RECLOOP
            invoke MessageBox,hWnd,addr SuccessMsg,addr AppName,0
        .ENDIF
```

用户信息处理程序：

```
Display      proc   hdc:HDC
            XX     equ  10
            YY     equ  10
         XX_GAP equ  100
         YY_GAP equ  30
            invoke TextOut,hdc,XX+0*XX_GAP,YY+0*YY_GAP,offset msg_list,4
            invoke TextOut,hdc,XX+0*XX_GAP,YY+1*YY_GAP,offset msg_name,4
            invoke TextOut,hdc,XX+1*XX_GAP,YY+1*YY_GAP,offset
msg_discount,8
            invoke TextOut,hdc,XX+2*XX_GAP,YY+1*YY_GAP,offset
msg_costprice,10
            invoke TextOut,hdc,XX+3*XX_GAP,YY+1*YY_GAP,offset
msg_salesprice,11
            invoke TextOut,hdc,XX+4*XX_GAP,YY+1*YY_GAP,offset
msg_totalvolume,12
            invoke TextOut,hdc,XX+5*XX_GAP,YY+1*YY_GAP,offset
msg_salesvolume,12
            invoke TextOut,hdc,XX+6*XX_GAP,YY+1*YY_GAP,offset
msg_recrate,7
            ;;
            push edx
            mov ecx,N
            mov gi,0
            mov yi,YY_GAP
            add yi,YY
            ShowList:
```

```
            push ecx
            add yi,YY_GAP
            movzx edx,gi
            mov ebx,offset buf.GoodName
            movzx eax,gi
            add ebx,eax
            invoke strlen,ebx
            invoke TextOut,hdc,XX+0*XX_GAP,yi,ebx,eax
            movzx edx,gi
            invoke F2T10,byte ptr buf[edx].Discount,addr
tempdiscount+2,16
            invoke strlen,addr tempdiscount+2
            mov tempdiscount,al
            invoke TextOut,hdc,XX+1*XX_GAP,yi,addr
tempdiscount+2,tempdiscount
            movzx edx,gi
            invoke F2T10,word ptr buf[edx].CostPrice,addr
tempcostprice+2,32
            invoke strlen,addr tempcostprice+2
            mov tempcostprice,al
            invoke TextOut,hdc,XX+2*XX_GAP,yi,addr
tempcostprice+2,tempcostprice
            movzx edx,gi
            invoke F2T10,word ptr buf[edx].SalesPrice,addr
tempsalesprice+2,32
            invoke strlen,addr tempsalesprice+2
            mov tempsalesprice,al
            invoke TextOut,hdc,XX+3*XX_GAP,yi,addr
tempsalesprice+2,tempsalesprice
            movzx edx,gi
            invoke F2T10,word ptr buf[edx].TotalVolume,addr
temptotalvolume+2,32
            invoke strlen,addr temptotalvolume+2
            mov temptotalvolume,al
            invoke TextOut,hdc,XX+4*XX_GAP,yi,addr
temptotalvolume+2,temptotalvolume
            movzx edx,gi
            invoke F2T10,word ptr buf[edx].SalesVolume,addr
tempsalesvolume+2,32
            invoke strlen,addr tempsalesvolume+2
            mov tempsalesvolume,al
            invoke TextOut,hdc,XX+5*XX_GAP,yi,addr
tempsalesvolume+2,tempsalesvolume
            movzx edx,gi
            invoke F2T10,word ptr buf[edx].RecRate,addr temprecrate+2,32
            invoke strlen,addr temprecrate+2
            mov temprecrate,al
            invoke TextOut,hdc,XX+6*XX_GAP,yi,addr
temprecrate+2,temprecrate
            add gi,21
            pop ecx
```

```
            dec ecx
            cmp ecx,0
            jnz ShowList
            pop edx
            ret
Display      endp

F2T10   proc tempNum:dword,tempAddress:dword,tempLength:word
        push ebx
        push esi
        push ecx
        push eax
        mov eax,tempNum
        lea esi,buf2
        cmp dx,tempLength
        jne B
        movsx eax,ax
B:
        or eax,eax
        jns PLUS
        neg eax
        mov byte ptr [esi],'-'
        inc esi
PLUS:
        mov ebx,10
        xor ecx,ecx
LOP1:
        xor edx,edx
        div ebx
        push edx
        inc ecx
        or eax,eax
        jnz LOP1
LOP2:
        pop eax
        cmp al,10
        jb L1
        add al,7
L1:
        add al,30H
        mov [esi],al
        inc esi
        loop LOP2
        mov byte ptr [esi],0
        invoke strcpy,tempAddress,addr buf2
        pop eax
        pop ecx
        pop esi
        pop ebx
        ret
F2T10 endp
```

```
CALCREC proc address:dword
        push eax
        push ebx
        push ecx
        push edx
        mov ecx,address
        add ecx,10
        movzx eax,byte ptr 0[ecx] ; 折扣
        cmp eax,0
        jz SPECIAL
        movzx edx,word ptr 3[ecx] ; 实际售价
        mul dx
        mov ebx,eax               ; ecx=折扣*实际售价
        movzx eax,word ptr 1[ecx]
        mov edx,1280
        mul dx                    ; eax=1280*进货价
        div ebx                   ; eax=rec1
        mov word ptr 9[ecx],ax
        movzx eax,word ptr 7[ecx]
        shl eax,7
        xor edx,edx
        movzx ebx,word ptr 5[ecx]
        div ebx
        add 9[ecx],ax             ; eax=rec2
        jmp CALC_EXIT
SPECIAL: mov word ptr 9[ecx],256
CALC_EXIT:
        pop edx
        pop ecx
        pop ebx
        pop eax
ret
CALCREC endp
            end  Start
```

1.3 实验步骤

1. 首先，根据框架，编写初版程序，观察是否能正常运行。
2. 修改 inc 文件及 rc 文件，添加菜单，观察是否能正确运行。
3. 添加 MessageBox 函数，观察是否能正常弹出。
4. 添加关于商品的结构体，并编写用于输出的函数，观察是否能正常显示。

1.4 实验记录与分析

1. 首先，在 Visual Studio 中新建工程，编写 asm 文件、inc 文件和 rc 文件后，尝试运行，发生错误，如下图所示：

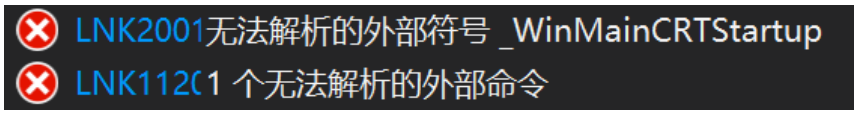

图 5-1 发生的错误截图

通过检查设置发现，没有将 masm 加入生成自定义中。解决方案：将 masm 加入生成自定义，并将 demo.asm 的生成设置设定为 MASM，重新编译，成功运行。

2. 修改 rc 文件，将菜单调整为与实验预期相符。重新编译运行，发现菜单没有发生变化，并且点击菜单无效。观察 asm 文件的语法，发现此处引用了 inc 的定义；修改 inc 文件，并使其与 rc 文件的定义相同，再次编译运行，发现菜单显示正常。

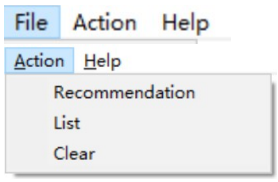

图 5-2 File 菜单截图
图 5-3 Action 菜单截图
图 5-4 Help 菜单截图

3. 修改 About 的逻辑，使用 invoke 调用 MessageBox 函数，输出个人信息。编译运行，成功显示：

图 5-5 About 显示的相关信息截图

点击确定后，能成功返回主界面。

4. 编写计算推荐度的子程序。将之前实验中编写的子程序复制后，进行修改，将参数的传递方法由寄存器法修改为 invoke 所默示的堆栈法，入口参数设置为结构体的首地址。
5. 编写循环计算推荐度的子程序。运行后，发生闪退；通过断点追踪，发现在访问结构体的成员时，发生了错误。通过查阅书本相关信息，修改了代码后，成功完成循环计算功能。为 Recommendation 功能编写了一个 MessageBox，在运行之后显示成功计算，截图如下：

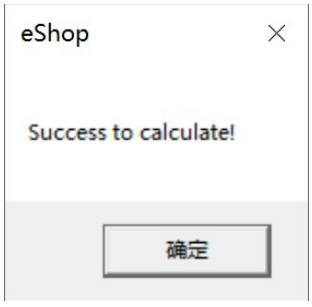

图 5-6 Recommendation 计算成功后显示的提示截图

尝试点击确定，能够返回主界面。

6. 编写显示结构体成员的函数。先尝试输出一个成员的信息，使用之前实验中使用过的 F2T10 来完成数字到字符串的转换。编译并运行后，发生错误，点击 List 后会发生闪退。通过观察代码，发现 F2T10 仍调用的是 Dos 系统下的输出语句。修改输出部分，将其生成的字符串复制到指定位置，并在 Display 子程序中使用 TextOut 函数输出该字符串，成功输出商品的名字及折扣信息。

7. 完善 List 的界面，加入第一行的提示文本 List，重新编译运行，发现 List 没能显示，原有的位置被第一个商品的信息覆盖了。观察代码，发现没有修改 y 轴显示位置。修改后，成功显示部分商品信息如下：

List

Name	Discount	Cost Price	Sales Price	Total Volume	Sales Volume	RecRate
PEN	10	35	56	70	25	0

图 5-7 部分商品信息截图

8. 添加循环部分，循环显示商品信息。重新编译运行，成功显示所有（5 个）商品的信息：

List

Name	Discount	Cost Price	Sales Price	Total Volume	Sales Volume	RecRate
PEN	10	35	56	70	25	0
BOOK	0	12	30	125	5	0
RULER	8	12	30	1250	0	0
GLUE	5	12	30	1250	0	0
BOX	7	30	60	90	5	0

图 5-8 所有商品信息截图

9. 运行推荐度计算功能，发现计算出来的推荐度严重偏大，并且导致其他商品的信息被部分覆盖，变成乱码。推测推荐度计算时，偏移地址计算错误，导致推荐度结果被放在了不合理的内存。通过断点与单步调试功能，追踪推荐度计算的过程，发现对偏移地址的计算错误。重新计算结构体的存放方式，并观察内存中数据的实际存放规律，根据以上结果修改了代码，重新编译运行，发现能正确计算推荐度：

List

Name	Discount	Cost Price	Sales Price	Total Volume	Sales Volume	RecRate
PEN	10	35	56	70	25	125
BOOK	0	12	30	125	5	256
RULER	8	12	30	1250	0	64
GLUE	5	12	30	1250	0	102
BOX	7	30	60	90	5	98

图 5-9 所有商品推荐度计算结果

10. 点击 Clear，发现所有信息均被消去：

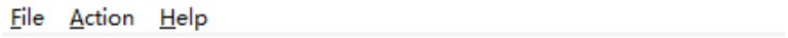

图 5-10 运行 Clear 后截图

与实验预期相符。

总结与体会

本次实验是在 Win32 环境下编程，而不是使用 Dosbox 在虚拟环境中运行；并且，本次实验的运行方式也不是命令行控制台，而是窗口程序。尽管只是写了简单的代码，但我深深的感受到了这两者的区别。Win32 下，使用了简化的段定义，使用了 invoke 的简化函数调用方式，使用了结构体；另外，win32 程序的 4 个组成部分之间的关系，也让我有了新的体验。从 Dos 到 win32 编程，我也感受到了计算机系统的发展与变革。

参考文献

[1]《80X86 汇编语言程序设计》，王元珍，曹忠升，韩宗芳编著，华中科技大学出版社，2005 年 4 月第一版