

**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： 2017-5-3，14：00-17：30 实验地点： 南一楼803室99号实验台**

**指导教师： 张勇**

**专业班级： 计算机科学与技术201307班**

**学 号： U201314969 姓 名： 王镇宇**

**同组学生： 无 报告日期： 2017年 5 月 4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：

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**

[1 实验目的与要求 3](#_Toc451728091)

[2 实验内容 3](#_Toc451728092)

[3 实验过程 5](#_Toc451728109)

[3.1 任务1 5](#_Toc451728110)

[3.1.1 设计思想及存储单元分配 5](#_Toc451728111)

[3.1.2 流程图 5](#_Toc451728112)

[3.1.3 源程序 5](#_Toc451728113)

[3.1.4 实验步骤 12](#_Toc451728114)

[4 体会 13](#_Toc451728115)

[参考文献 14](#_Toc451728116)

# 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，实现学生成绩表信息的平均值计算、排序及显示等功能（可参考实验二、四），具体要求如下描述。

功能一：编写一个基于窗口的WIN32程序的菜单框架，具有以下的下拉菜单项：

File Action Help

Exit Average About

Sort

List

点菜单File下的Exit选项时结束程序；点菜单Help下的选项About或按F1键，都弹出一个消息框，显示本人信息，类似图8-1所示。点菜单Action下的选项Average、Sort、List将分别实现计算平均值、排序或显示所有成绩的功能（详见功能二的描述）。

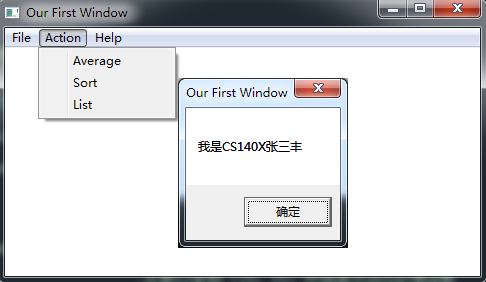


图8-1 菜单示例

**提示：“F1按键消息”的识别方法——**收到的窗口消息是WM\_KEYDOWN时，wParam中键盘虚拟码若为VK\_F1就表示按下了F1键。Windows.inc中定义了VK\_F1 equ 70h。

功能二：每个学生的相关信息包括：姓名（结尾含1个以上的数值0，共占10个字节），语文成绩（1个字节），数学成绩（1个字节），英语成绩（一个字节），平均成绩（1个字节），等级（1个字节）。要求采用结构变量存放学生的相关信息。学生人数至少5人。姓名和各科成绩直接在数据段中给定，不必运行时输入。**成绩表中最后一个学生必须使用自己的姓名**。

1. 点菜单项Average时，计算平均成绩并给出等级（等级的定义见实验二，但这里不用单独显示等级）。平均成绩的计算仍按照实验二的公式进行。平均成绩和等级保存到上述结构变量的相应字段中。用TD32观察计算结果。
2. 点菜单项List时，要求能在窗口中列出所有学生信息，包括姓名、各科成绩、平均成绩、等级等。如图8-2所示。平均成绩尚未计算时，平均成绩及等级显示为空白。

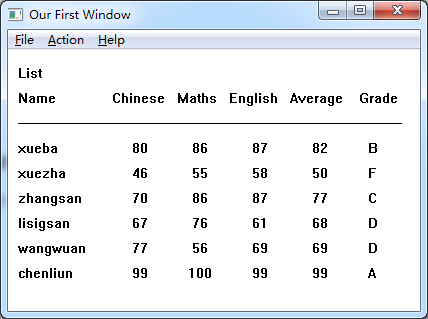


图8-2 成绩单显示示意图

1. （选做）点菜单项Sort后，将按照平均成绩降序排序。再点菜单项List，能显示排序后的所有学生信息。平均成绩尚未计算时，点菜单项Sort将提示要先计算平均成绩。

# 实验过程

## 任务1

### 设计思想及存储单元分配

本次实验主要由两部分构成：完善display子程序和写average子程序，完成List项与Average项的逻辑功能。

### 流程图



### 源程序

1..inc

IDM\_FILE\_EXIT equ 10001

IDM\_Action\_Average equ 10101

IDM\_Action\_Sort equ 10102

IDM\_Action\_List equ 10103

IDM\_HELP\_ABOUT equ 10201

2..rc

#define IDM\_FILE\_EXIT 10001

#define IDM\_Action\_Average 10101

#define IDM\_Action\_Sort 10102

#define IDM\_Action\_List 10103

#define IDM\_HELP\_ABOUT 10201

MyMenu MENU

BEGIN

POPUP "&File"

BEGIN

MENUITEM "E&xit",IDM\_FILE\_EXIT

END

POPUP "&Action"

BEGIN

MENUITEM "A&verage",IDM\_Action\_Average

MENUITEM "S&ort",IDM\_Action\_Sort

MENUITEM "L&ist",IDM\_Action\_List

END

POPUP "&Help"

BEGIN

MENUITEM "&About",IDM\_HELP\_ABOUT

END

END

3..asm

.386

.model flat,stdcall

option casemap:none

WinMain proto :DWORD,:DWORD,:DWORD,:DWORD

WndProc proto :DWORD,:DWORD,:DWORD,:DWORD

Display proto :DWORD

include ex8.INC

include windows.inc

include user32.inc

include kernel32.inc

include gdi32.inc

include shell32.inc

includelib user32.lib

includelib kernel32.lib

includelib gdi32.lib

includelib shell32.lib

student struct

myname db 10 dup(0)

chinese db 0

math db 0

english db 0

average db 0

grade db 0

student ends

.data

ClassName db 'TryWinClass',0

AppName db 'Our First Window',0

MenuName db 'MyMenu',0

DlgName db 'MyDialog',0

AboutMsg db '我是CS1409唐礼威',0

hInstance dd 0

CommandLine dd 0

buf student <'Jin',97,98,99,98,'A'>

student <'zhangsan',80,85,90,85,'A'>

student <'lisi',95,90,95,93,'A'>

student <'wangwu',80,75,85,80,'A'>

student <'xiao',85,90,90,88,'A'>

msg\_name db 'name',0

msg\_chinese db 'chinese',0

msg\_math db 'math',0

msg\_english db 'english',0

msg\_average db 'average',0

msg\_grade db 'grade',0

chinese db 2,'97','80','95','80', '85'

math db 2,'98','85','90','75', '90'

english db 2,'99','90','95','85', '90'

average db 2,'00','00','00','00', '00'

.code

Start: invoke GetModuleHandle,NULL

mov hInstance,eax

invoke GetCommandLine

mov CommandLine,eax

invoke WinMain,hInstance,NULL,CommandLine,SW\_SHOWDEFAULT

invoke ExitProcess,eax

;;

WinMain proc hInst:DWORD,hPrevInst:DWORD,CmdLine:DWORD,CmdShow:DWORD

LOCAL wc:WNDCLASSEX

LOCAL msg:MSG

LOCAL hWnd:HWND

invoke RtlZeroMemory,addr wc,sizeof wc

mov wc.cbSize,SIZEOF WNDCLASSEX

mov wc.style, CS\_HREDRAW or CS\_VREDRAW

mov wc.lpfnWndProc, offset WndProc

mov wc.cbClsExtra,NULL

mov wc.cbWndExtra,NULL

push hInst

pop wc.hInstance

mov wc.hbrBackground,COLOR\_WINDOW+1

mov wc.lpszMenuName, offset MenuName

mov wc.lpszClassName,offset ClassName

invoke LoadIcon,NULL,IDI\_APPLICATION

mov wc.hIcon,eax

mov wc.hIconSm,0

invoke LoadCursor,NULL,IDC\_ARROW

mov wc.hCursor,eax

invoke RegisterClassEx, addr wc

INVOKE CreateWindowEx,NULL,addr ClassName,addr AppName,\

WS\_OVERLAPPEDWINDOW,CW\_USEDEFAULT,\

CW\_USEDEFAULT,CW\_USEDEFAULT,CW\_USEDEFAULT,NULL,NULL,\

hInst,NULL

mov hWnd,eax

INVOKE ShowWindow,hWnd,SW\_SHOWNORMAL

INVOKE UpdateWindow,hWnd

;;

MsgLoop: INVOKE GetMessage,addr msg,NULL,0,0

cmp EAX,0

je ExitLoop

INVOKE TranslateMessage,addr msg

INVOKE DispatchMessage,addr msg

jmp MsgLoop

ExitLoop: mov eax,msg.wParam

ret

WinMain endp

WndProc proc hWnd:DWORD,uMsg:DWORD,wParam:DWORD,lParam:DWORD

LOCAL hdc:HDC

.IF uMsg == WM\_DESTROY

invoke PostQuitMessage,NULL

.ELSEIF uMsg == WM\_KEYDOWN

.IF wParam == VK\_F1

invoke MessageBox,hWnd,addr AboutMsg,addr AppName,0

;;your code

.ENDIF

.ELSEIF uMsg == WM\_COMMAND

.IF wParam == IDM\_FILE\_EXIT

invoke SendMessage,hWnd,WM\_CLOSE,0,0

.ELSEIF wParam == IDM\_Action\_Average

invoke Display,hWnd

.ELSEIF wParam == IDM\_Action\_Sort

invoke Display,hWnd

.ELSEIF wParam == IDM\_Action\_List

invoke Display,hWnd

.ELSEIF wParam == IDM\_HELP\_ABOUT

invoke MessageBox,hWnd,addr AboutMsg,addr AppName,0

.ENDIF

.ELSEIF uMsg == WM\_PAINT

;;redraw window again

.ELSE

invoke DefWindowProc,hWnd,uMsg,wParam,lParam

ret

.ENDIF

xor eax,eax

ret

WndProc endp

Display proc hWnd:DWORD

XX equ 10

YY equ 10

XX\_GAP equ 100

YY\_GAP equ 30

LOCAL hdc:HDC

invoke GetDC,hWnd

mov hdc,eax

invoke TextOut,hdc,XX+0\*XX\_GAP,YY+0\*YY\_GAP,offset msg\_name,4

invoke TextOut,hdc,XX+1\*XX\_GAP,YY+0\*YY\_GAP,offset msg\_chinese,7

invoke TextOut,hdc,XX+2\*XX\_GAP,YY+0\*YY\_GAP,offset msg\_math,4

invoke TextOut,hdc,XX+3\*XX\_GAP,YY+0\*YY\_GAP,offset msg\_english,7

invoke TextOut,hdc,XX+4\*XX\_GAP,YY+0\*YY\_GAP,offset msg\_average,7

invoke TextOut,hdc,XX+5\*XX\_GAP,YY+0\*YY\_GAP,offset msg\_grade,5

;;

invoke TextOut,hdc,XX+0\*XX\_GAP,YY+1\*YY\_GAP,offset buf[0\*15].myname,3

invoke TextOut,hdc,XX+1\*XX\_GAP,YY+1\*YY\_GAP,offset chinese+1,chinese

invoke TextOut,hdc,XX+2\*XX\_GAP,YY+1\*YY\_GAP,offset math+1, math

invoke TextOut,hdc,XX+3\*XX\_GAP,YY+1\*YY\_GAP,offset english+1,english

invoke TextOut,hdc,XX+4\*XX\_GAP,YY+1\*YY\_GAP,offset average+1,average

invoke TextOut,hdc,XX+5\*XX\_GAP,YY+1\*YY\_GAP,offset buf[0\*15].grade,1

;;

invoke TextOut,hdc,XX+0\*XX\_GAP,YY+1\*YY\_GAP,offset buf[1\*15].myname,8

invoke TextOut,hdc,XX+1\*XX\_GAP,YY+1\*YY\_GAP,offset chinese+2,chinese

invoke TextOut,hdc,XX+2\*XX\_GAP,YY+1\*YY\_GAP,offset math+2, math

invoke TextOut,hdc,XX+3\*XX\_GAP,YY+1\*YY\_GAP,offset english+2,english

invoke TextOut,hdc,XX+4\*XX\_GAP,YY+1\*YY\_GAP,offset average+2,average

invoke TextOut,hdc,XX+5\*XX\_GAP,YY+1\*YY\_GAP,offset buf[1\*15].grade,1

;;

invoke TextOut,hdc,XX+0\*XX\_GAP,YY+1\*YY\_GAP,offset buf[2\*15].myname,4

invoke TextOut,hdc,XX+1\*XX\_GAP,YY+1\*YY\_GAP,offset chinese+3,chinese

invoke TextOut,hdc,XX+2\*XX\_GAP,YY+1\*YY\_GAP,offset math+3, math

invoke TextOut,hdc,XX+3\*XX\_GAP,YY+1\*YY\_GAP,offset english+3,english

invoke TextOut,hdc,XX+4\*XX\_GAP,YY+1\*YY\_GAP,offset average+3,average

invoke TextOut,hdc,XX+5\*XX\_GAP,YY+1\*YY\_GAP,offset buf[2\*15].grade,1

;;

invoke TextOut,hdc,XX+0\*XX\_GAP,YY+1\*YY\_GAP,offset buf[3\*15].myname,6

invoke TextOut,hdc,XX+1\*XX\_GAP,YY+1\*YY\_GAP,offset chinese+4,chinese

invoke TextOut,hdc,XX+2\*XX\_GAP,YY+1\*YY\_GAP,offset math+4, math

invoke TextOut,hdc,XX+3\*XX\_GAP,YY+1\*YY\_GAP,offset english+4,english

invoke TextOut,hdc,XX+4\*XX\_GAP,YY+1\*YY\_GAP,offset average+4,average

invoke TextOut,hdc,XX+5\*XX\_GAP,YY+1\*YY\_GAP,offset buf[3\*15].grade,1

;;

invoke TextOut,hdc,XX+0\*XX\_GAP,YY+1\*YY\_GAP,offset buf[4\*15].myname,4

invoke TextOut,hdc,XX+1\*XX\_GAP,YY+1\*YY\_GAP,offset chinese+5,chinese

invoke TextOut,hdc,XX+2\*XX\_GAP,YY+1\*YY\_GAP,offset math+5, math

invoke TextOut,hdc,XX+3\*XX\_GAP,YY+1\*YY\_GAP,offset english+5,english

invoke TextOut,hdc,XX+4\*XX\_GAP,YY+1\*YY\_GAP,offset average+5,average

invoke TextOut,hdc,XX+5\*XX\_GAP,YY+1\*YY\_GAP,offset buf[4\*15].grade,1

ret

Display endp

Average proc number:DWORD

push eax

push ebx

push ecx

push edx

push esi

xor ecx, ecx

avg\_loop:

cmp ecx, number

jge avg\_finish

xor eax, eax

xor edx, edx

imul ebx, ecx, 15

mov al, buf[ebx].chinese

mov dl, buf[ebx].math

lea eax, [edx + eax \* 2]

xor edx, edx

mov dl, buf[ebx].english

lea esi, [edx + eax \*2]

mov eax, 92492493h

imul esi

add edx, esi

sar edx, 2

mov eax, edx

shr eax, 1fh

add edx, eax

mov buf[ebx].average, dl

.if dl > 90

mov buf[ebx].stugrade, 'A'

.elseif dl > 80

mov buf[ebx].stugrade, 'B'

.elseif dl > 70

mov buf[ebx].stugrade, 'C'

.elseif dl > 60

mov buf[ebx].stugrade, 'D'

.else

mov buf[ebx].stugrade, 'E'

.endif

inc ecx

jmp avg\_loop

avg\_finish:

pop esi

pop edx

pop ecx

pop ebx

pop eax

ret

Average endp

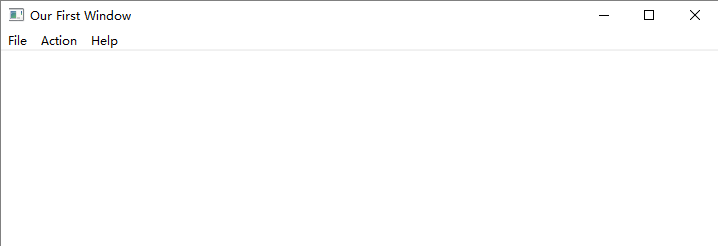
end Start

### 实验步骤

1. 将 masm32/bin 文件夹加入环境变量，建立编译、链接工具环境；

2. 整理思路，画出各模块的执行流程图；

3. 改写 ex8.rc 文件，为窗口界面增加菜单栏。改写后的菜单栏如下图所示



4. 进行资源文件生成、编译、链接；

5. 根据流程图，编写 win32 程序；

# 体会

通过这次上机，进一步熟悉了汇编工具的使用，懂得怎么设置环境变量，初步熟悉了win32编程的基本方法和需要注意的地方，为以后编写图形界面的程序打下坚实的基础，相信对今后的编程实践有着很大的益处。

# 参考文献

[1] 王元珍等.80x86汇编语言程序设计.版本(第1版)

[2] 王晓虹等.汇编语言程序设计教程.版本(第1版)