

**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**实验名称： 实验八** WIN32编程

**实验时间： 2016-5-10，14：30-17：30 实验地点： 南一楼804室102号实验台**

**指导教师： 张勇**

**专业班级：计算机科学与技术201409班**

**学 号： U2014414808 姓 名： 王林**

**同组学生： 无 报告日期： 2016年 5 月 13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： 王林

日期： 2016.5.13

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**

[实验八 WIN32编程 1](#_Toc450907935)

[一、实验目的与要求 1](#_Toc450907936)

[二、实验内容 1](#_Toc450907937)

[三、实验过程 2](#_Toc450907938)

[8.1任务1 2](#_Toc450907939)

[8.1.1设计思想及存储单元分配 2](#_Toc450907940)

[8.1.2流程图 3](#_Toc450907941)

[8.1.3源程序 3](#_Toc450907942)

[8.1.5实验记录 11](#_Toc450907943)

[四、体会 12](#_Toc450907944)

[五、参考文献 12](#_Toc450907945)

**实验八** WIN32编程

一、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。

二、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：

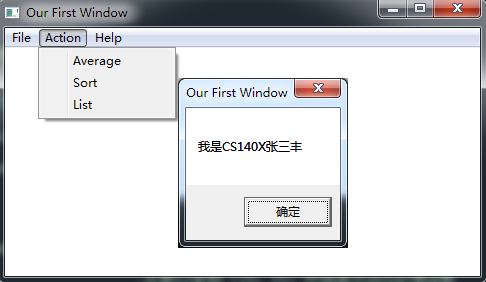
File Action Help

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）。

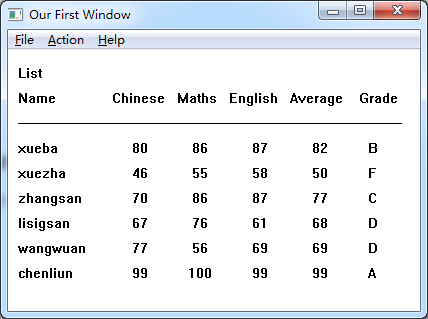


**图8-1 菜单示例**

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。

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。

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。**

三、实验过程

8.1任务1

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）。

8.1.1设计思想及存储单元分配

计思思想：进行win32编程，画出窗口界面，定义如题要求的的菜单，定义好资源文件夹，将实验2实验4的逻辑写过来。

首先在内存分配空间存储学生姓名以及各科成绩和等级，然后运行程序后，屏幕打印出菜单，并且可以通过鼠标点击和快捷键进行操作。点击【list】时打印部分文本信息，平均值和等级部分输出为空白；点击【average】后打印完整文本信息，包括平均成绩和等级。

存储单元分配：以buf为首地址的内存空间存储学生信息，定义一个studengt结构，结构成员包括10字节的姓名、4字节的成绩、1字节的成绩等级。buf是一个结构数组，大小为5，储存5个学生的信息；然后分别定义4个数据存储区，分别储存对应的成绩准话后的字符串，每个空间占10个字节，每个成绩占2个字节的空间。

8.1.2流程图



**图 8-3 基于窗口的应用程序**



**图8-4 average函数流程图**

8.1.3源程序

【ex8.asm】

.386

.model flat,stdcall

option casemap:none

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

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

Display proto :DWORD

Clear proto :DWORD

Averagef proto

Radix proto

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 'I am strawberylin',0

hInstance dd 0

CommandLine dd 0

buf student <'Zhangsan\_a',40,50,60,0,' '>

student <'Zhangsan\_b',96,98,90,0,' '>

student <'Zhangsan\_c',50,60,70,0,' '>

student <'Zhangsan\_d',60,70,80,0,' '>

student <'WangLin ',90,80,70,0,' '>

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 '40','96','50','60','90'

math db '50','98','60','70','80'

english db '60','90','70','80','70'

average db ' ',' ',' ',' ',' '

boolHasPaint db 0

.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

LOCAL ps:PAINTSTRUCT

.IF uMsg == WM\_DESTROY

invoke PostQuitMessage,NULL

.ELSEIF uMsg == WM\_KEYDOWN

.IF wParam == VK\_F1

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

.ENDIF

.ELSEIF uMsg == WM\_COMMAND

.IF wParam == IDM\_FILE\_EXIT

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

.ELSEIF wParam == IDM\_FILE\_LIST

invoke Display,hWnd

mov boolHasPaint, 1

.ELSEIF wParam == IDM\_ACTION\_AVERAGE

invoke Averagef

invoke Display,hWnd

mov boolHasPaint, 1

.ELSEIF wParam == IDM\_ACTION\_SORT

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

.ELSEIF wParam == IDM\_HELP\_ABOUT

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

.ENDIF

.ELSEIF uMsg == WM\_PAINT

invoke BeginPaint,hWnd,addr ps

.IF boolHasPaint == 1

invoke Display,hWnd

.ENDIF

invoke EndPaint,hWnd,addr ps

.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,10

invoke TextOut,hdc,XX+1\*XX\_GAP,YY+1\*YY\_GAP,offset chinese+0,2

invoke TextOut,hdc,XX+2\*XX\_GAP,YY+1\*YY\_GAP,offset math+0, 2

invoke TextOut,hdc,XX+3\*XX\_GAP,YY+1\*YY\_GAP,offset english+0,2

invoke TextOut,hdc,XX+4\*XX\_GAP,YY+1\*YY\_GAP,offset average+0,2

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+2\*YY\_GAP,offset buf[1\*15].myname,10

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

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

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

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

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

invoke TextOut,hdc,XX+0\*XX\_GAP,YY+3\*YY\_GAP,offset buf[2\*15].myname,10

invoke TextOut,hdc,XX+1\*XX\_GAP,YY+3\*YY\_GAP,offset chinese+4,2

invoke TextOut,hdc,XX+2\*XX\_GAP,YY+3\*YY\_GAP,offset math+4, 2

invoke TextOut,hdc,XX+3\*XX\_GAP,YY+3\*YY\_GAP,offset english+4,2

invoke TextOut,hdc,XX+4\*XX\_GAP,YY+3\*YY\_GAP,offset average+4,2

invoke TextOut,hdc,XX+5\*XX\_GAP,YY+3\*YY\_GAP,offset buf[2\*15].grade,1

invoke TextOut,hdc,XX+0\*XX\_GAP,YY+4\*YY\_GAP,offset buf[3\*15].myname,10

invoke TextOut,hdc,XX+1\*XX\_GAP,YY+4\*YY\_GAP,offset chinese+6,2

invoke TextOut,hdc,XX+2\*XX\_GAP,YY+4\*YY\_GAP,offset math+6, 2

invoke TextOut,hdc,XX+3\*XX\_GAP,YY+4\*YY\_GAP,offset english+6,2

invoke TextOut,hdc,XX+4\*XX\_GAP,YY+4\*YY\_GAP,offset average+6,2

invoke TextOut,hdc,XX+5\*XX\_GAP,YY+4\*YY\_GAP,offset buf[3\*15].grade,1

invoke TextOut,hdc,XX+0\*XX\_GAP,YY+5\*YY\_GAP,offset buf[4\*15].myname,10

invoke TextOut,hdc,XX+1\*XX\_GAP,YY+5\*YY\_GAP,offset chinese+8,2

invoke TextOut,hdc,XX+2\*XX\_GAP,YY+5\*YY\_GAP,offset math+8, 2

invoke TextOut,hdc,XX+3\*XX\_GAP,YY+5\*YY\_GAP,offset english+8,2

invoke TextOut,hdc,XX+4\*XX\_GAP,YY+5\*YY\_GAP,offset average+8,2

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

ret

Display endp

Averagef proc

push edi

push esi

push ecx

push edx

push eax

push ebx

mov ecx, 5

lea esi, average

lea edi, buf

figure:

mov eax, 0

mov edx, 0

mov dl, [edi+10]

add edx, edx

add edx, edx

add eax, edx

mov edx, 0

mov dl, [edi+11]

add edx, edx

add eax, edx

mov edx, 0

mov dl, [edi+12]

add eax, edx

xor edx, edx

mov ebx, 7

div ebx

mov [edi+13], al

.IF al >= 90

mov bl, 41h

mov [edi+14], bl

.ELSEIF al >=80

mov bl, 42h

mov [edi+14], bl

.ELSEIF al >=70

mov bl, 43h

mov [edi+14], bl

.ELSEIF al >=60

mov bl, 44h

mov [edi+14], bl

.ELSE

mov bl, 45h

mov [edi+14], bl

.ENDIF

call Radix

add edi, 15

dec ecx

jnz figure

pop ebx

pop eax

pop edx

pop ecx

pop esi

pop edi

ret

Averagef endp

Radix proc

push ecx

push ebx

push edx

mov ebx, 10

xor cx, cx

radix\_loop1:

xor edx, edx

div ebx

push dx

inc cx

or eax, eax

jnz radix\_loop1

radix\_loop2:

pop ax

cmp al, 10

jb radix\_l1

add al, 7

radix\_l1:

add al, 30h

mov [esi], al

inc esi

loop radix\_loop2

pop edx

pop ebx

pop ecx

ret

Radix endp

end Start

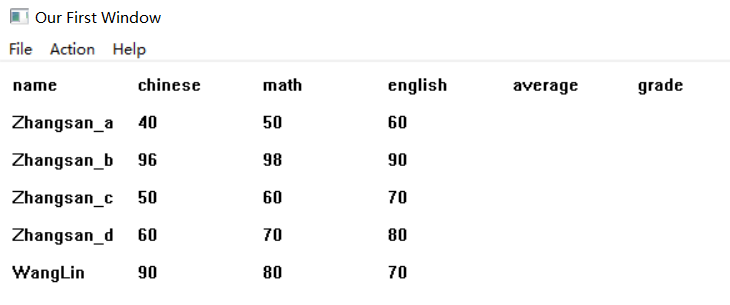
8.1.4实验步骤

1、在汇编网站下载相关资料，阅读任务书和相关说明，学习有关masm32的相关知识

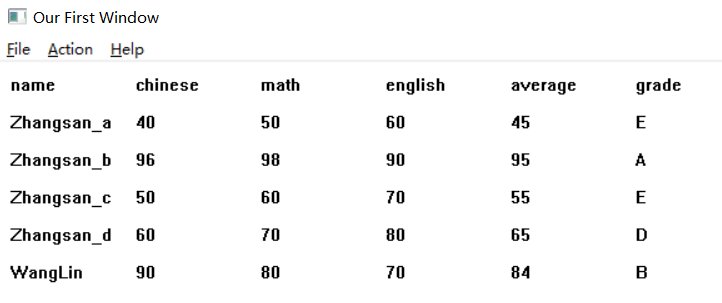
2、阅读实验所给的框架源码。

3、在所给框架的基础上编写源码，编译、链接、生成可执行程序，调试直到没有BUG，并且遗尿症程序正确性。

程序运行验证：



**图8-5 窗口list验证**



**图8-6 窗口average验证**

8.1.5实验记录

1、实验环境条件：P3 1GHz，256M内存；WINDOWS XP命令行窗口；Sublime Text 2；MASM32。

2、基于窗口的程序包含4部分，即主程序、窗口主程序、窗口消息处理程序、用户子程序，主程序需要完成初始化工作，主要是：获取窗口句柄，取命令行参数的地址，调用窗口主程序等。

3、TD32 与16位TD有很多不同，包括寄存器全部变为32位，堆栈段显示发生变化，数据区被保护了，全部显示为‘？’。

4、数据区要查看数据，需要在数据区按下快捷键【ctrl+g】，然后输入ds:[偏移地址]来查看对应的数据区内容。

5、可以通过鼠标点击和按下快捷键来发送消息，消息类型有直接退出- WM\_DESTROY、按下快捷键F1- WM\_KEYDOWN、点击热区- WM\_COMMAND、还有其他改变窗口消息-WM\_SIAING-WM\_MOVE-WM\_SETFOCUS等等。

6、本次实验中用到字符串输出函数 TEXOUT，参数是坐标，字符串地址，输出长度。

7、本次试验中定义了结构变量，可以是数据结构更加清晰，结构更加明显。

四、体会

通过本次上机，熟悉了win32编程的框架，了解了win32编程的相关知识。

其次，本次上机的具体收获有：（1）学习了有关win32编程的框架，了解了主程序、窗口主程序、消息处理程序、用户子程序的关系。（2）学习了有关32位TD和16位TD的区别，以及用TD32调试程序的不同。（3） 学习了在保护方式下32位程序和16位程序的区别。（4）学习了有关masm32的知识。

五、参考文献

[序号] 作者.书名.版本(第×版).译者.出版地:出版者,出版年：起页-止页

[序号] URL: 网络地址，如URL:http://www.cbs.dtu.dk/services