汇编语言实验报告

1. 实验目的

1. 大数相乘。要求实现两个十进制大整数的相乘（100位以上），输出乘法运算的结果。
2. 结合Windows界面编程和浮点数编程，实现完善的计算器功能，支持浮点运算和三角函数等功能。
3. Windows界面风格实现两个文本文件内容的比对。若两文件内容一样，输出相应提示；若两文件不一样，输出对应的行号。

2. 实验过程

2.1 大数相乘

2.1.1 数据结构

该程序需要在命令行界面获取输入的两个十进制大整数，并输出相乘结果，故需要设置数组数据结构，保存大整数的每一位。相关代码如下：

```
        .386
        .model flat, stdcall
        option casemap:none

        includelib msvcrt.lib

printf  PROTO C :ptr sbyte, :VARARG
scanf   PROTO C :ptr sbyte, :VARARG
strlen  PROTO C :ptr sbyte, :VARARG

        .data
        format      byte "%s",0
        input_str1  byte 200 dup(0)
        input_str2  byte 200 dup(0)
        output_str  byte 200 dup(0)
        input_num1  dword 200 dup(0)
        input_num2  dword 200 dup(0)
        output_num  dword 200 dup(0)

        input_len1  dword 0
        input_len2  dword 0
```

```
        output_len  dword 0

        radix       dword 10
```

2.1.2 字符串与整数的转换

由于得到的大整数是字符串形式，则需要将整数的每一位转换为数字。该功能由stoi函数实现。该函数会将str_in数组中的每个元素减去48并压入栈，然后将结果出栈，逆序保存在num_out数组中，以方便计算。相关代码如下：

```
stoi proc stdcall str_in :ptr byte, num_out :ptr dword, len
:dword
        mov     ecx,len
        mov     esi,str_in

@@:
        movzx   eax,byte ptr [esi]
        sub     eax,30h
        push    eax
        inc     esi
        loop    @B

        mov     ecx,len
        mov     esi,num_out
@@:
        pop     eax
        mov     dword ptr [esi],eax
        add     esi,4
        loop    @B

        ret
stoi endp
```

为了方便输出，在完成计算后将大整数逆序转换为正确顺序的字符串格式。该功能由类似的itos函数实现。相关代码如下：

```
itos proc stdcall str_out :ptr byte, num_in :ptr dword, len
:dword
        mov     ecx,len
        mov     esi,num_in
@@:
        mov     eax,dword ptr [esi]
        add     esi,4
        push    eax
        loop    @B
```

```
        mov     ecx,len
        mov     esi,str_out
@@:
        pop     eax
        add     eax,30h
        mov     byte ptr [esi],al
        inc     esi
        loop    @B

        ret
itos endp
```

2.1.3 乘法计算

该功能由函数Multiply实现，可以实现两个输入大整数数组结构的相乘并保存到输出数组中。

该函数的主要结构为二重循环，即将乘数的每一位乘以被乘数的每一位。这一过程从数字的低位遍历到高位，并将每一次的乘法运算结果加上上一位保留的累计进位结果，然后除以10，将余数保存到对应的输出位中，并将商累计到下一位的进位结果中。

最后还需要计算结果的长度，以便输出。通常情况下长度为$len_1 + len_2$或$len_1 + len_2 - 1$。如果$output_num[len_1 + len_2 - 1] = 0$，则$output_len = len_1 + len_2 - 1$。如果$output_num[len_1 + len_2 - 1] = 0$且$output_num[0] = 0$，表明结果为0，则$output_len = 1$。

相关代码如下

```
Multiply proc far C
        mov     ecx,input_len1
        xor     edi,edi
L1:
        xor     esi,esi

L2:
        mov     eax,dword ptr input_num1[edi*4]
        mul     input_num2[esi*4]

        mov     ebx,esi
        add     ebx,edi
        add     eax,output_num[ebx*4]

        mov     ebx,10
        div     ebx

        mov     ebx,esi
        add     ebx,edi
```

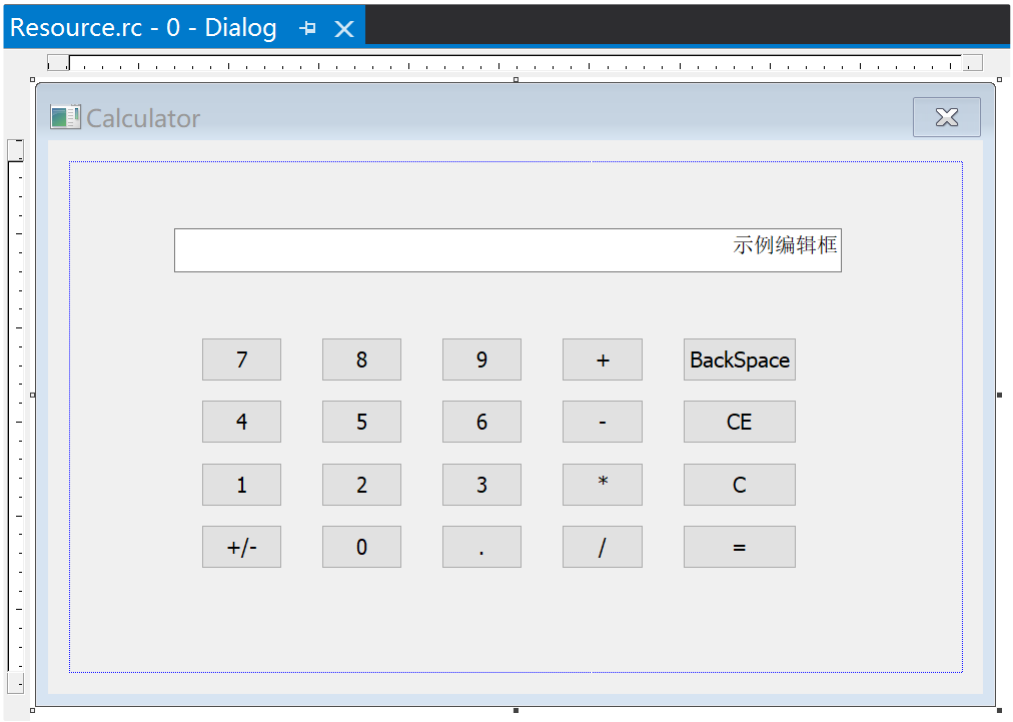
```
        mov     output_num[ebx*4],edx
        add     output_num[ebx*4+4],eax

        inc     esi
        mov     ebx,input_len2
        cmp     esi,ebx
        jb      L2

        inc     edi
        loop    L1

        mov     eax,input_len1
        add     eax,input_len2

        .if     output_num[eax*4-4] == 0
                sub     eax,1
                .if     output_num[0] == 0
                        mov     eax,1
                .endif
        .endif

        mov     output_len,eax
        ret
Multiply endp
```

2.1.4 实验结果

2.2 计算器

2.2.1 图形界面

- 使用资源文件自定义图形界面
- 显示数值的文本框
- 按钮若干



2.2.2 头文件和数据定义

由于内容较多，所以全部放在.inc文件中。

```
include windows.inc
include user32.inc
include kernel32.inc
include comctl32.inc
include masm32.inc
include shell32.inc
include winmm.inc
include comdlg32.inc
includelib user32.lib
includelib comctl32.lib
includelib masm32.lib
includelib winmm.lib
includelib comdlg32.lib
includelib msvcrt.lib

ICON        equ 105
ID_EDIT     equ 1001

ID_NUM0     equ 1002
ID_NUM1     equ 1003
ID_NUM2     equ 1004
ID_NUM3     equ 1005
ID_NUM4     equ 1006
ID_NUM5     equ 1007
ID_NUM6     equ 1008
```

```
        ID_NUM7     equ 1009
        ID_NUM8     equ 1010
        ID_NUM9     equ 1011

        ID_NEG      equ 1012
        ID_DOT      equ 1013

        ID_BACK     equ 1014
        ID_CE       equ 1015
        ID_C        equ 1016

        ID_EQU      equ 1017
        ID_ADD      equ 1018
        ID_SUB      equ 1019
        ID_MUL      equ 1020
        ID_DIV      equ 1021

        WinMain PROTO :DWORD, :DWORD, :DWORD,:DWORD      ;窗
口主程序
        Calculate PROTO :DWORD,:DWORD,:DWORD,:DWORD      ;消
息处理程序
        PackNum PROTO                                    ;数
字分组子程序
        UnpackNum PROTO                                  ;数
字不分组子程序
        BtnNum PROTO :DWORD                              ;数
字按键消息处理程序
        ShowNum PROTO                                    ;显
示数据子程序
        ShowTextM PROTO                                  ;显
示存储信息子程序
        Init PROTO                                       ;初
始化计算器子程序
        GetResult PROTO                                  ;计
算结果子程序
        BtnOperator PROTO                                ;双
目运算符消息处理程序
        BtnEqual PROTO                                   ;等
于消息处理程序

        .data
        hInstance       dd  ?                   ;主程序句柄
        hEdit           dd  ?                   ;输出文本框句
柄
        hIcon           dd  ?                   ;Icon句柄
        DialogName      db  "Calculator",0      ;对话框名称
        IconName        db  "Icon",0            ;Icon名称
```

```
        Output          db  "0.",0,30 dup(0)     ;输出字符串
        IsStart         db  1                    ;判断是否运算
开始
        HasPoint        db  0                    ;判断是否存在
小数点
        HasEqual        db  0                    ;判断是否存在
等号
        Remember        dq  0.0                  ;记忆数据
        Number          dq  0.0                  ;记录临时数据
        Result          dq  0.0                  ;记录结果
        Operand         dq  0.0                  ;记录操作数
        IsPacket        db  0                    ;数字分组
        Operator        db  '.'                  ;记录运算符
        IsError         db  0                    ;记录是否出现
异常
        Div0            db  "除数不能为零。",0
        FunctionError   db  "函数输入无效。",0
        NumLittle       REAL8   1.0E-12
        Num10           REAL8   10.0             ;实数10
        Num100          REAL8   100.0            ;实数100
```

2.2.3 窗口创建和消息传递

该实验中，窗口的创建和消息的分发都由WinMain函数实现。WinMain函数会声明一个WNDCLASSEX类型的本地变量，并设置该结构体变量的各个成员变量的值，然后用该变量作为参数创建窗口。

在成功创建窗口之后，WinMain函数进入无限循环，以获取窗口传回的消息，并将传递给响应函数。

相关代码如下：

```
WinMain proc
hInst:DWORD,hPrevInst:DWORD,CmdLine:DWORD,CmdShow:DWORD
        LOCAL   wc:WNDCLASSEX                    ;窗口类
        LOCAL   msg:MSG                          ;消息
        LOCAL   hWnd:HWND                        ;对话框
句柄

        mov     wc.cbSize,sizeof WNDCLASSEX
         ;WNDCLASSEX的大小
        mov     wc.style,CS_BYTEALIGNWINDOW or
CS_BYTEALIGNWINDOW   ;窗口风格 or CS_HREDRAW or CS_VREDRAW
        mov     wc.lpfnWndProc,OFFSET Calculate
         ;窗口消息处理函数地址
        mov     wc.cbClsExtra,0
         ;在窗口类结构后的附加字节数，共享内存
```

```
        mov     wc.cbWndExtra,DLGWINDOWEXTRA
         ;在窗口实例后的附加字节数
        mov     eax,hInst
        mov     wc.hInstance,eax
         ;窗口所属程序句柄
        mov     wc.hbrBackground,COLOR_BTNFACE+1
         ;背景画刷句柄
        mov     wc.lpszMenuName,NULL
         ;菜单名称指针
        mov     wc.lpszClassName,OFFSET DialogName
         ;类名称指针
        invoke  LoadIcon,hInst,addr IconName

        mov     wc.hIcon,eax
         ;图标句柄
        invoke  LoadCursor,NULL,IDC_ARROW
        mov     wc.hCursor,eax
         ;光标句柄
        mov     wc.hIconSm,0
         ;窗口小图标句柄

        invoke  RegisterClassEx,addr wc
;注册窗口类
        invoke  CreateDialogParam,hInst,addr
DialogName,0,addr Calculate,0  ;创建对话框窗口
        mov     hWnd,eax
;保存对话框句柄
        invoke  ShowWindow,hWnd,CmdShow
        invoke  UpdateWindow,hWnd
;更新窗口

        .while  TRUE
;消息循环
                invoke  GetMessage,addr msg,0,0,0
;获取消息
                .break  .if eax == 0
                invoke  TranslateMessage,addr msg
;转换键盘消息
                invoke  DispatchMessage,addr msg
;分发消息
        .endw
        mov     eax,msg.wParam
        ret
WinMain endp
```

2.2.4 消息处理

WinMain函数在接收到消息后，会转发给Calculate函数进行处理。

Calculate函数会对消息进行分类。

若uMsg == WM_INITDIALOG，表明该消息是初始化对话框消息，则进行初始化。

若uMsg == WM_CHAR，表明该消息是键盘输入消息，则调用相应的按钮响应过程。

若uMsg == WM_COMMAND，表明该消息是按钮单击消息，则根据参数aParam的值调用按钮响应过程。

相关代码如下：

```
Calculate proc
hWin:DWORD,uMsg:UINT,aParam:DWORD,bParam:DWORD
        .if uMsg == WM_INITDIALOG
                invoke  GetDlgItem,hWin,ID_EDIT
;获取输出文本框句柄
                mov     hEdit,eax
;保存文本框句柄
                invoke  LoadIcon,hInstance,addr IconName
;载入Icon
                mov     hIcon,eax
;保存Icon句柄
                invoke
SendMessage,hWin,WM_SETICON,ICON_SMALL ,eax
                invoke  SendMessage,hEdit,WM_SETTEXT,0,addr
Output  ;显示"0."
        .elseif uMsg == WM_CHAR
;键盘输入
                mov eax,aParam
                sub eax,'0'
                add eax,ID_NUM0
                .if     (eax >= ID_NUM0) && (eax <=
ID_NUM9)        ;数字
                        invoke
Calculate,hWin,WM_COMMAND,eax,0
                .elseif eax == 961
;ID_BACK
                        invoke
Calculate,hWin,WM_COMMAND,ID_BACK,0
                .elseif eax == 1014
;ID_EQU
                        invoke
Calculate,hWin,WM_COMMAND,ID_EQU,0
                .elseif eax == 999
;ID_DOT
```

```
                        invoke
Calculate,hWin,WM_COMMAND,ID_DOT,0
                .elseif eax == 996
;ID_ADD
                        invoke
Calculate,hWin,WM_COMMAND,ID_ADD,0
                .elseif eax == 998
;ID_SUB
                        invoke
Calculate,hWin,WM_COMMAND,ID_SUB,0
                .elseif eax == 995
;ID_MUL
                        invoke
Calculate,hWin,WM_COMMAND,ID_MUL,0
                .elseif eax == 1000
;ID_DIV
                        invoke
Calculate,hWin,WM_COMMAND,ID_DIV,0
                .endif
        .elseif uMsg == WM_COMMAND
                mov     eax,aParam
                .if     eax == ID_CE
;清零按钮CE
                        lea     esi,Output
                        mov     BYTE PTR[esi],'0'
                        mov     BYTE PTR[esi+1],'.'
                        mov     BYTE PTR[esi+2],0
                        .if     IsError==1
                                invoke  Init
                        .endif
                        invoke
SendMessage,hEdit,WM_SETTEXT,0,addr Output
                .elseif eax == ID_C
;初始化按钮C
                        invoke
Calculate,hWin,WM_COMMAND,ID_CE,bParam
                        invoke  Init
                .elseif IsError == 1
                        ret
                .elseif eax == ID_BACK
;退格按钮Backspace
                        invoke  UnpackNum
                        .if     IsStart == 0
                                lea     esi,Output
                                .while  BYTE PTR[esi] != 0
                                        inc     esi
                                .endw
```

```
                                .if     BYTE PTR[esi-
1]=='.'
                                        .if     HasPoint ==
1
                                                mov
HasPoint,0
                                        .else
                                                .if
BYTE PTR[esi-3] == '-'
                                                        lea
esi,Output
                                                        mov
BYTE PTR[esi],'0'
                                                        mov
BYTE PTR[esi+1],'.'
                                                        mov
BYTE PTR[esi+2],0
                                                .else
                                                        mov
BYTE PTR[esi-2],'.'
                                                        mov
BYTE PTR[esi-1],0
                                                .endif
                                        .endif
                                .else
                                        mov     BYTE
PTR[esi-1],0
                                .endif
                                lea     esi,Output
                                .if     BYTE PTR[esi] ==
'.'
                                        mov     BYTE
PTR[esi],'0'
                                        mov     BYTE
PTR[esi+1],'.'
                                        mov     BYTE
PTR[esi+2],0
                                .endif
                                invoke  ShowNum
                        .endif
                .elseif (eax >= ID_NUM0) && (eax <=
ID_NUM9)    ;数字按钮
                        .if     HasEqual == 1
                                invoke  Init
                        .endif
                        invoke  BtnNum,eax
```

```
                .elseif eax == ID_NEG
;正负号按钮
                        invoke  UnpackNum
                        invoke  StrToFloat,addr Output,
addr Number
                        finit
                        fldz
                        fld     Number
                        fsub
                        fstp    Number
                        invoke  FloatToStr2,Number,addr
Output
                        invoke  ShowNum
                .elseif eax == ID_DOT
;小数点按钮
                        mov     BYTE PTR HasPoint,1
                        mov     BYTE PTR IsStart,0
                .elseif (eax >= ID_ADD) && (eax <= ID_DIV)
;双目运算符按钮
                        invoke  BtnOperator
                .elseif eax == ID_EQU
;等于按钮
                        invoke  BtnEqual
                .endif
        .elseif
                invoke
DefWindowProc,hWin,uMsg,aParam,bParam
        .endif
        ret
Calculate endp
```

2.2.5 BtnNum函数

BtnNum函数响应数字按钮消息，向文本框中添加字符。

```
BtnNum proc USES eax,Num:DWORD
        lea     esi,Output
        mov     eax,Num
        sub     eax,954
        .if     IsStart == 1
                mov     [esi],eax
                inc     esi
                mov     BYTE PTR[esi],'.'
                inc     esi
                mov     BYTE PTR[esi],0
                mov     IsStart,0
```

```
        .else
                .while  BYTE PTR[esi] != '.'
                        inc esi
                .endw
                .if     HasPoint == 1
                        .while BYTE PTR[esi]!=0
                            inc esi
                        .endw
                        mov [esi],ax
                        inc esi
                        mov BYTE PTR[esi],0
                .else
                        .if BYTE PTR[Output]=='0'
                            lea esi,Output
                            mov [esi],eax
                            mov BYTE PTR[esi+1],'.'
                            mov BYTE PTR[esi+2],0
                        .else
                            mov [esi],eax
                            inc esi
                            mov BYTE PTR[esi],'.'
                            inc esi
                            mov BYTE PTR[esi],0
                        .endif
                .endif
        .endif
        invoke  ShowNum
        ret
BtnNum endp
```

2.2.6 BtnOperator函数

BtnOperator函数响应运算符按钮消息，进行运算并输出结果。首先判断是否为等号，如果不是则调用GetResult函数先进行一次运算，然后将当前操作符存入Operator变量中。

```
BtnOperator proc USES eax
        .if     HasEqual != 1
                invoke GetResult
        .endif
        .if     eax ==  ID_MUL
                mov     Operator,'*'
        .elseif eax == ID_DIV
                mov     Operator,'/'
        .elseif eax == ID_SUB
                mov     Operator,'-'
```

```
        .elseif eax == ID_ADD
                mov     Operator,'+'
        .endif
        mov HasEqual,0
        ret
BtnOperator endp
```

2.2.7 BtnEqual函数

BtnEqual函数响应等号按钮消息。首先判断是否为起始状态，如果不是则调用GetResult函数，并将HasEqual变量置1。

```
BtnEqual proc
        .if (IsStart==1) && (HasEqual==0)
            fstp    Number
            fst     Number
            fld     Number
        .endif
        invoke  GetResult
        mov     HasEqual,1
        ret
BtnEqual endp
```

2.2.8 GetResult函数

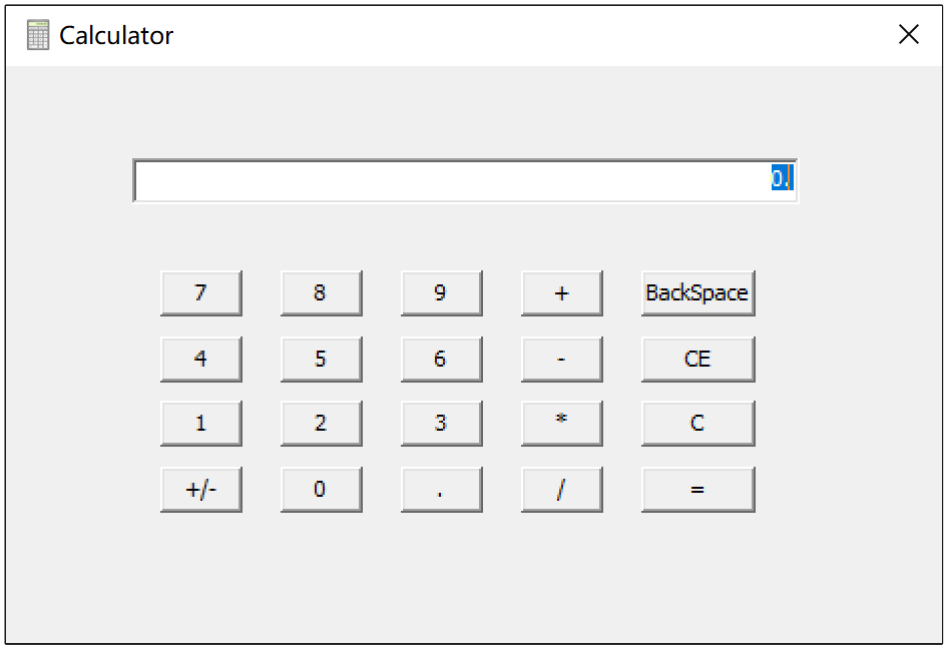
GetResult函数进行运算并输出结果。

```
GetResult proc USES eax
        invoke  UnpackNum
        finit
        .if (IsStart==1) && (HasEqual==0)
        .else
            .if HasEqual != 1
                    invoke  StrToFloat,addr Output, addr
Operand
            .endif
            fld     Result
            fld     Operand
            .if Operator == '.'
                fst     Result
                jmp     Show
            .elseif Operator=='+'
                fadd    ST(1),ST(0)
            .elseif Operator=='-'
                fsub    ST(1),ST(0)
```

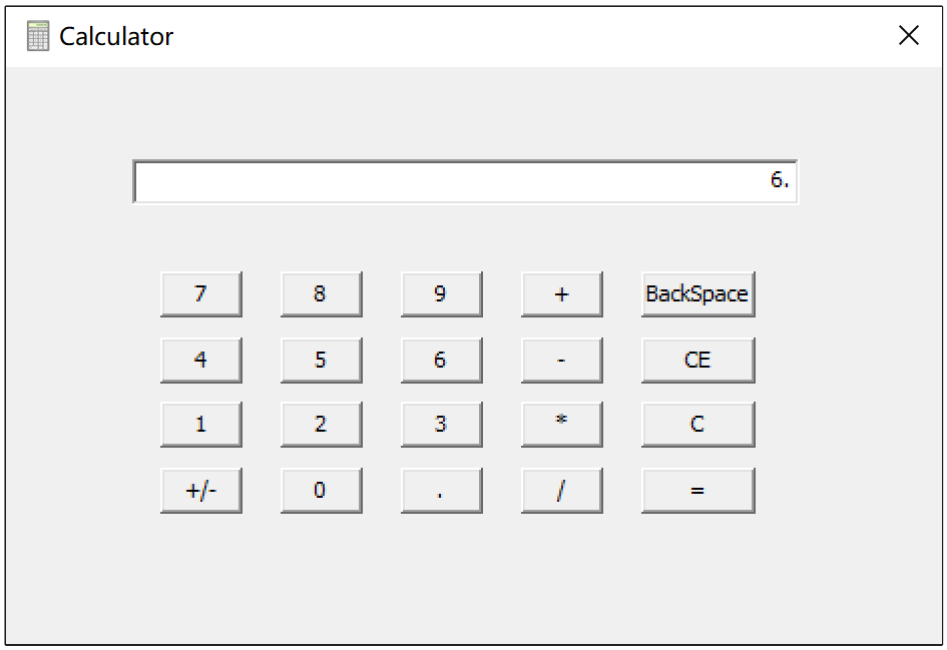
```
            .elseif Operator=='*'
                fmul    ST(1),ST(0)
            .elseif Operator=='/'
                fldz
                fcomi   ST(0),ST(1)
                jnz     NotZero
                mov     IsError,1
                invoke  SendMessage,hEdit,WM_SETTEXT,0,addr
Div0
                ret
NotZero:        fstp    Operand
                fdiv    ST(1),ST(0)
            .endif
            fstp    Operand
            fst     Result
Show:       mov     IsStart,1
            mov     HasPoint,0
            invoke  FloatToStr2,Result,addr Output
            invoke  ShowNum
        .endif
        ret
GetResult endp
```

2.2.9 实验结果

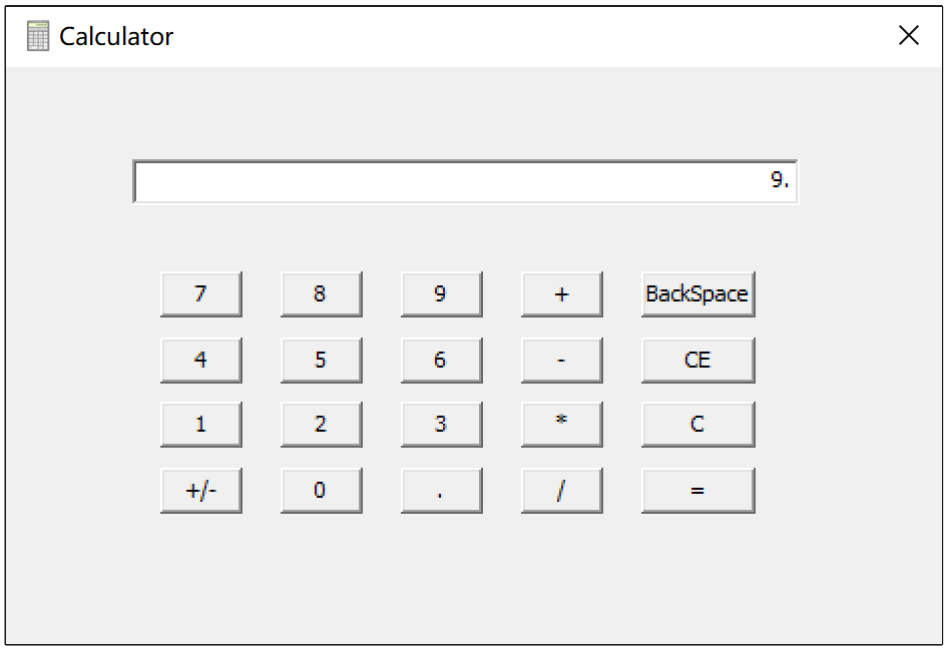
- 初始界面



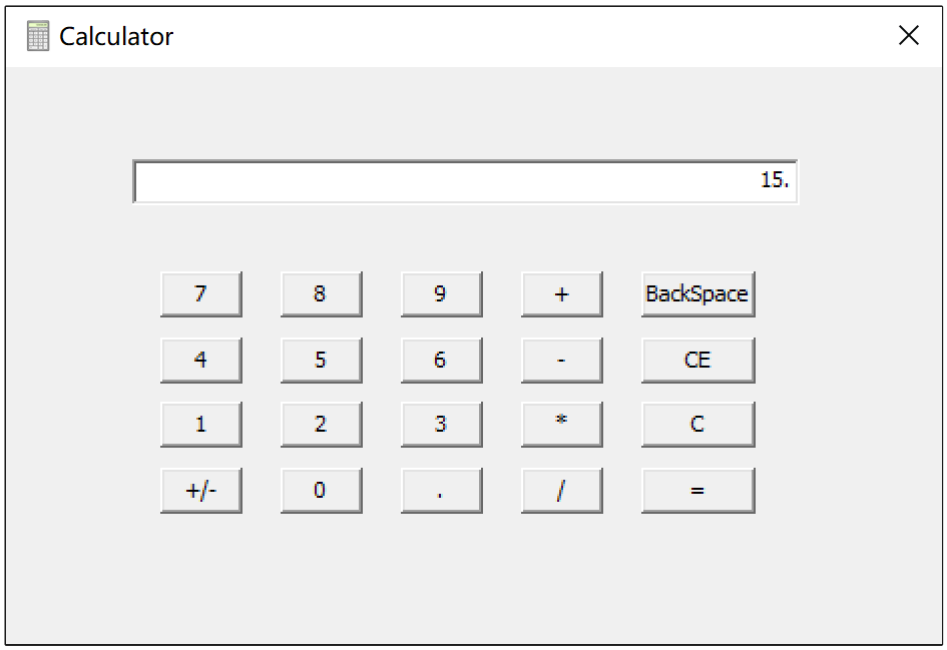
- 点击数字按钮6



- 点击加号按钮后无变化
- 点击数字按钮9



- 点击等号按钮



2.3 文本内容比较

2.3.1 图形界面

- 窗口主要元素
  - 两个输入文件路径的文本框
  - 确认按钮
  - 结果提示窗口
- 窗口初始化函数
  - 创建两个文本框
  - 创建确认按钮，单击事件的触发值设为15

2.3.2 读取文件

- 在点击确认后，文件路径会被分别保存到file1_path和file2_path变量。
- Readline函数实现读取文件某一行的功能
  - 参数为文件指针和对应缓冲区指针
  - 循环调用ReadFile函数，每次读入一个字符
  - 若读取到\n，则该行读取完毕
  - 若读取到空字符，则文件读取完毕

相关代码如下：

```
ReadLine proc fp :HANDLE,buffer :ptr byte
        LOCAL len :dword
        LOCAL chr :byte

        mov     esi,buffer
```

```
        mov     edi,0

L1:
        invoke  ReadFile,fp,addr chr,1,addr len,NULL
        cmp     len,0
        je      L2
        cmp     chr,10
        je      L2
        mov     al, chr
        mov     byte ptr [esi],al
        inc     esi
        inc     edi
        jmp     L1
L2:
        mov     byte ptr [esi],0
        mov     eax,edi
        ret
ReadLine endp
```

2.3.3 文本内容比较

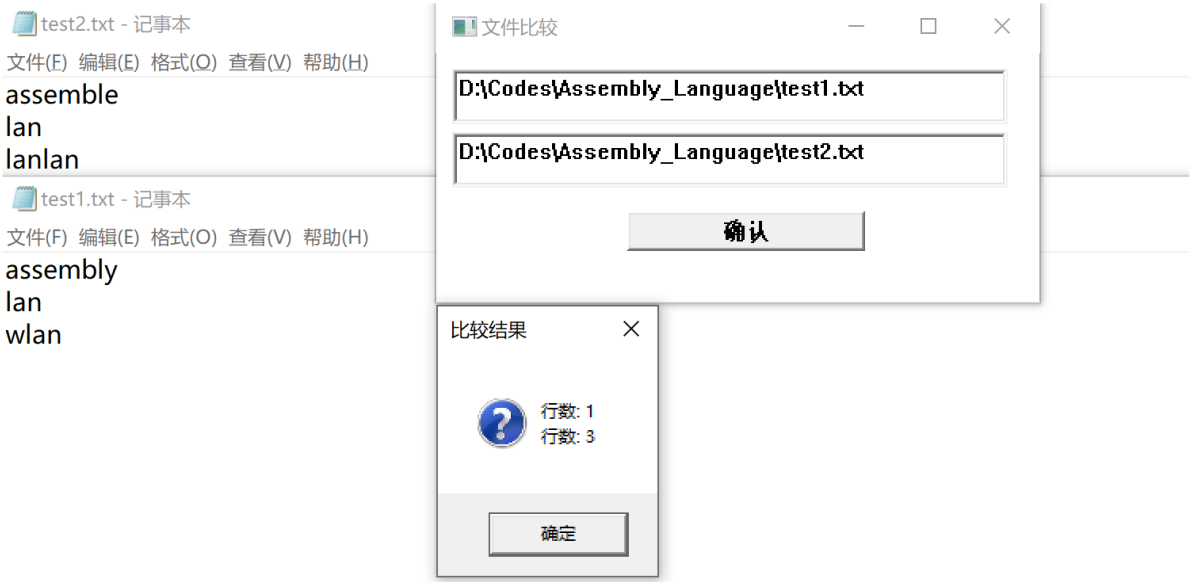
- 该程序需要实现文本文件的逐行比较，对于行号k采用以下比较规则：
    - 调用strcmp函数比对两个文件的第k行。若结果不一致，则记录该行号。
    - 若第k行是任一文件的最后一行，则在比较完成之后输出结果

相关代码如下：

```
CompareFile proc fpath1:ptr byte, fpath2:ptr byte
        LOCAL fp1 :HANDLE
        LOCAL fp2 :HANDLE
        LOCAL lp1 :dword
        LOCAL lp2 :dword
        LOCAL index_line :dword
        LOCAL buffer_differ[1024] :byte

        invoke  CreateFile, fpath1, GENERIC_READ,
FILE_SHARE_READ, NULL, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL, NULL
        mov     fp1,eax
        invoke  CreateFile, fpath2, GENERIC_READ,
FILE_SHARE_READ, NULL, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL, NULL
        mov     fp2,eax
        mov     differ_num,0
        mov     index_line,0
        mov     esi,offset differs
```

```
        mov     byte ptr[esi],0
L0:
        inc     index_line
        invoke  ReadLine,fp1,offset buffer1
        mov     lp1,eax
        invoke  ReadLine,fp2,offset buffer2
        mov     lp2,eax
L1:
        cmp     lp1,0
        jne     L3
        cmp     lp2,0
        jne     L2
        jmp     ENDFUNC
L2:
        invoke  sprintf, addr buffer_differ, offset
DiffContent, index_line
        invoke  strcat, offset differs, addr buffer_differ
        inc     differ_num
        jmp     L0
L3:
        cmp     lp2,0
        jne     L4
        invoke  sprintf, addr buffer_differ, offset
DiffContent, index_line
        invoke  strcat, offset differs, addr buffer_differ
        inc     differ_num
        jmp     L0
L4:
        invoke  strcmp, offset buffer1, offset buffer2
        cmp     eax, 0
        je      L0
        invoke  sprintf, addr buffer_differ, offset
DiffContent, index_line
        invoke  strcat, offset differs, addr buffer_differ
        inc     differ_num
        jmp     L0
ENDFUNC:
        ret
CompareFile endp
```

2.3.4 实验结果


test2.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
assemble
lan
lanlan
test1.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
assembly
lan
wlan
文件比较
D:\Codes\Assembly_Language\test1.txt
D:\Codes\Assembly_Language\test2.txt
确认
比较结果
行数: 1
行数: 3
确定