Experiment Report 2

1. Experimental requirements and objective

a) Be able to code, assemble, and execute a program with Visual C++ and MASM.
b) Know how to link your programs to an external code library.
c) Know how to create conditional and looping structures using assembly language.

2. Experimental environment

a) Hardware environment

The microcomputer CPU more than Pentium, more than 120GB capacity hard drive, more than 1GB of memory.

b) Software environment

Visual Studio 2008 and above versions of applications.

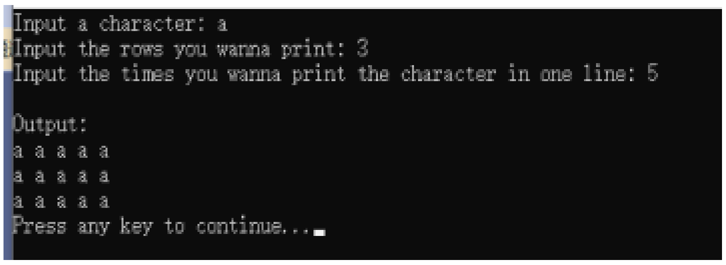
3. Experimental contents

1) Write a procedure that takes three arguments: a character and two integers. The character is to be printed. The first integer specifies the number of times that the character is to be printed on a line, and the second integer specifies the number of lines that are to be printed. Write a program that makes use of this procedure.
2) Write a procedure that sets each element in an array to the sum of the corresponding elements in two other arrays. (That is, if array 1 has the values 2 , 4 , 5 , and 8, and array 2 has the values 1 , 0 , 4 , and 6, the function assigns array 3 the values 3 , 4 , 9 , and 14 .) The procedure should take each address of the three arrays and the array size as arguments. Test the procedure in a simple program.
3) (OPTIONAL) Write a program that prompts the user to enter three sets of five integer numbers each. (You may assume the user responds correctly and doesn’t enter non-numeric data.) The program should accomplish all of the following:
   a. Store the integers in a 3×5 array.
   b. Compute the average of each set of five values, and display the results.
   c. Compute the average of all the values, and display the results.
   d. Determine the largest value of the 15 values, and display the results.
   Each major task should be handled by a separate procedure.

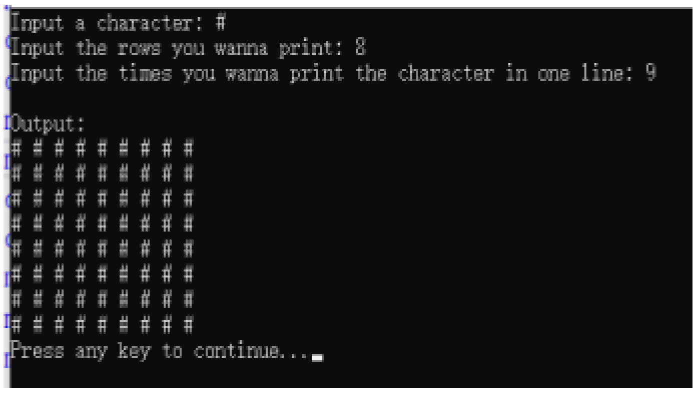
4. Experiment Result

Screenshots of program execution:

a) Content one:

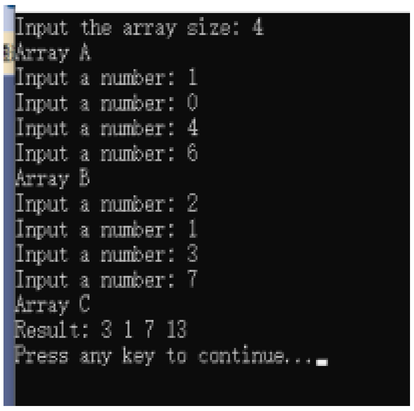


```
Input a character: a
Input the rows you wanna print: 3
Input the times you wanna print the character in one line: 5

Output:
a a a a a
a a a a a
a a a a a
Press any key to continue...
```

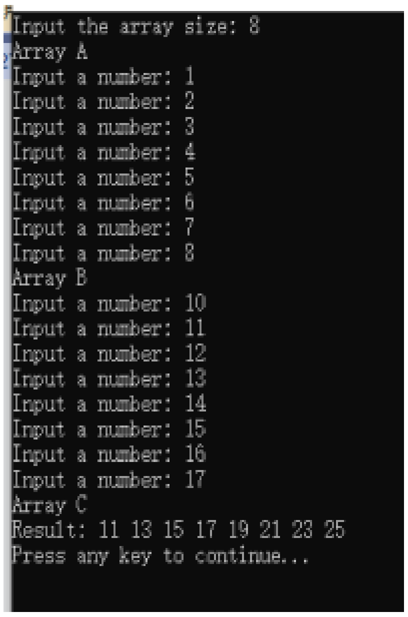


```
Input a character: #
Input the rows you wanna print: 8
Input the times you wanna print the character in one line: 9

Output:
# # # # # # # # #
# # # # # # # # #
# # # # # # # # #
# # # # # # # # #
# # # # # # # # #
# # # # # # # # #
# # # # # # # # #
# # # # # # # # #
Press any key to continue...
```

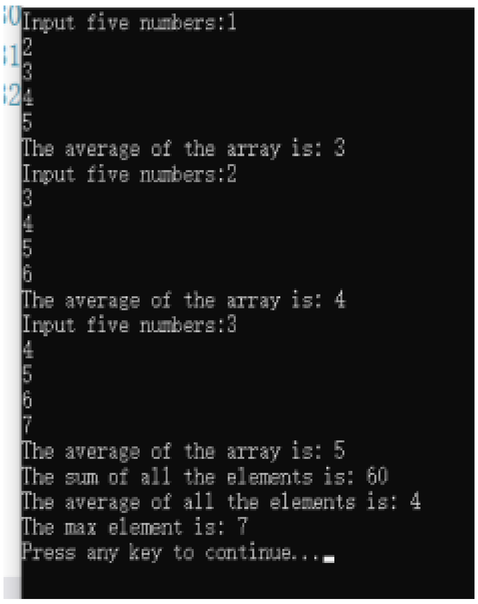
b) Content two:

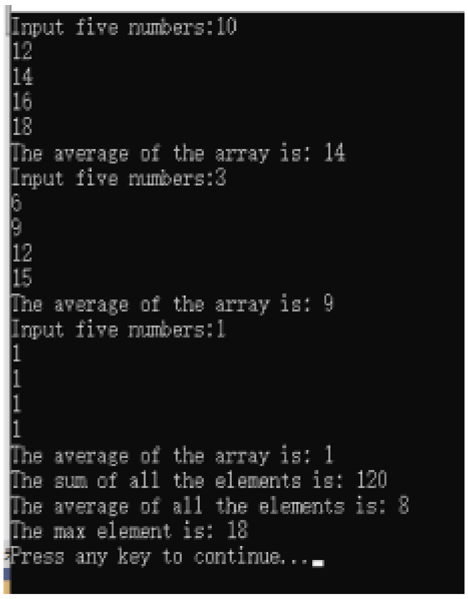


```
Input the array size: 4
Array A
Input a number: 1
Input a number: 0
Input a number: 4
Input a number: 6
Array B
Input a number: 2
Input a number: 1
Input a number: 3
Input a number: 7
Array C
Result: 3 1 7 13
Press any key to continue...
```


```
Input the array size: 8
Array A
Input a number: 1
Input a number: 2
Input a number: 3
Input a number: 4
Input a number: 5
Input a number: 6
Input a number: 7
Input a number: 8
Array B
Input a number: 10
Input a number: 11
Input a number: 12
Input a number: 13
Input a number: 14
Input a number: 15
Input a number: 16
Input a number: 17
Array C
Result: 11 13 15 17 19 21 23 25
Press any key to continue...
```

c) Content three:


```
Input five numbers:1
2
3
4
5
The average of the array is: 3
Input five numbers:2
3
4
5
6
The average of the array is: 4
Input five numbers:3
4
5
6
7
The average of the array is: 5
The sum of all the elements is: 60
The average of all the elements is: 4
The max element is: 7
Press any key to continue...
```


```
Input five numbers:10
12
14
16
18
The average of the array is: 14
Input five numbers:3
6
9
12
15
The average of the array is: 9
Input five numbers:1
1
1
1
1
The average of the array is: 1
The sum of all the elements is: 120
The average of all the elements is: 8
The max element is: 18
Press any key to continue...
```

5. Source Code of Programs

```
title One
INCLUDE Irvine32.inc

.data
    col         dword   ?
    row         dword   ?
    character   byte ?
    hInput      byte "Input a character: ",0
    cInput      byte "Input the times you wanna print the character in one line: ",0
    rInput      byte "Input the rows you wanna print: ",0
    Output      byte "Output: ",0
.code
main proc
    mov         edx,offset hInput
    call    WriteString
    call    ReadChar
    mov         character,al
    call    WriteChar
    call    Crlf
    mov         edx,offset rInput
    call    WriteString
    call    ReadDec
    mov         col,eax
    mov         edx,offset cInput
    call    WriteString
    call    ReadDec
    mov         row,eax
    mov         ecx,0
    mov         edx,0
    call    Crlf
    mov         edx,offset Output
    call    WriteString
    call    Crlf
    mov         edx,0
    .while      ecx < col
        .while      edx < row
            mov         al,character
            call    WriteChar
            mov         al,' '
            call    WriteChar
            inc         edx
        .endw
        mov      edx,0
        call Crlf
        inc      ecx
    .endw
    call     WaitMsg
main endp
end main
```

```
title Two
INCLUDE Irvine32.inc

.data
    ArraySize    dword    0
    ArrayA       dword    20       DUP(0)
    ArrayB       dword    20       DUP(0)
    ArrayC       dword    20       DUP(0)
    ArrA     byte"Array A",0
    ArrB     byte"Array B",0
    ArrC     byte"Array C",0
    InputSize    byte"Input the array size: ",0
    InputNum byte"Input a number: ",0
    Output       byte"Result: ",0

.code
main proc
    mov          edx,offset InputSize
    call     WriteString
    call     ReadDec
    mov          ArraySize,eax
    mov          ecx,0
    mov          esi,0
    mov          edx,offset ArrA
    CALL     WriteString
    call     Crlf
    mov          edx,offset InputNum
    .while       ecx < ArraySize
        call     WriteString
        call     ReadDec
        mov          ArrayA[esi],eax
        inc          ecx
        add          esi,4
    .endw
    mov          ecx,0
    mov          esi,0
    mov          edx,offset ArrB
    CALL     WriteString
    call     Crlf
    mov          edx,offset InputNum
    .while       ecx < ArraySize
        call     WriteString
        call     ReadDec
        mov          ArrayB[esi],eax
        inc          ecx
        add          esi,4
    .endw
    mov          ecx,0
    mov          esi,0
    mov          edx,offset ArrC
    CALL     WriteString
    call     Crlf
    mov          edx,offset InputNum
    .while       ecx < ArraySize
        mov          eax,ArrayA[esi]
        add          eax,ArrayB[esi]
        mov          ArrayC[esi],eax
        inc          ecx
        add          esi,4
    .endw
    mov          ecx,0
    mov          esi,0
    mov          edx,offset Output
    call     WriteString
    .while       ecx < ArraySize
        mov          eax,ArrayC[esi]
        call     WriteDec
        mov          al,’ ’
        call     WriteChar
        inc          ecx
        add          esi,4
    .endw
    call     Crlf
    call     WaitMsg
main endp
end main
```

```
title three

INCLUDE  Irvine32.inc

.data
    ArrayA       dword    5    Dup(0)
    ArrayB       dword    5    Dup(0)
    ArrayC       dword    5    Dup(0)
    avgA     dword    0
    avgB     dword    0
    avgC     dword    0
    Num          dword    5
    hInput       byte"Input five numbers:",0
    hOutput      byte"The average of the array is: ",0
    AllOutput    byte"The average of all the elements is: ",0
    AllSum       byte"The sum of all the elements is: ",0
    max          dword    0
    maxOne       byte"The max element is: ",0
.code
main proc

IA:
    mov          ecx,0
    mov          esi,0
    mov          edx,offset hInput
    call     WriteString
    .while       ecx< 5
        call     ReadDec
        mov          ArrayA[esi],eax
        .if          eax > max
            mov          max,eax
        .endif
        add          avgA,eax
        sar          esi,2
        inc          ecx
    .endw
    jmp          AvgAA
IB:
    mov          ecx,0
    mov          esi,0
    mov          edx,offset hInput
    call     WriteString
    .while       ecx< 5
        call     ReadDec
        mov          ArrayB[esi],eax
        .if          eax > max
            mov          max,eax
        .endif
        add          avgB,eax
        add          esi,4h
        inc          ecx
    .endw
    jmp          AvgAB
IC:
    mov          ecx,0
    mov          esi,0
    mov          edx,offset hInput
    call     WriteString
    .while       ecx< 5
        call     ReadDec
        mov          ArrayC[esi],eax
        .if          eax > max
            mov          max,eax
        .endif
        add          avgC,eax
        add          esi,4h
        inc          ecx
    .endw
    jmp          AvgAC

AvgAA:
    ;call        WriteDec
    mov          edx,0                ;这里必须清零，else会出bug
    mov          eax,avgA
    mov          esi,5
    div          esi
    mov          edx,offset hOutput
    call     WriteString
    call     WriteDec
    call     Crlf
    jmp          IB



AvgAB:
    ;call        WriteDec
    mov          edx,0                ;这里必须清零，else会出bug
    mov          eax,avgB
    mov          esi,5
    div         esi
    mov         edx,offset hOutput
    call    WriteString
    call    WriteDec
    call    Crlf
    jmp         IC

AvgAC:
    ;call        WriteDec
    mov         edx,0               ;这里必须清零，else会出bug
    mov         eax,avgC
    mov         esi,5
    div         esi
    mov         edx,offset hOutput
    call    WriteString
    call    WriteDec
    call    Crlf

SumAvg:
    mov         eax,avgA
    add         eax,avgB
    add         eax,avgC
    mov         edx,offset AllSum
    call    WriteString
    call    WriteDec
    call    Crlf
    mov         edx,0
    mov         esi,15
    div         esi
    mov         edx,offset AllOutput
    call    WriteString
    call    WriteDec
    call    Crlf

TheMax:
    mov         edx,offset maxOne
    call    WriteString
    mov         eax,max
    call    WriteDec
    call    Crlf

    call    WaitMsg
main endp
end main
```

6. Summary

Preparations:

For the whole experiment, since we have already prepared the environment, we can start to programming with the knowledge below:

- How the data is stored in the array using assembly language
- Branch (.if & .endif) and loop (.while & .endw)
- Multiplication and division (with one register or memory, cannot use the instant number)

Writing the program:

a) For the content one, we can turn the requirements into the following contents:

- Input a character, it needs to be printed in the screen by calling procedure "WriteChar"
- Input two character, one is representing how many rows of the line will the procedure print, the other is representing how many characters will be printed in one single line.
- Using while loop to print the array of the character

Register inside:

- Edx → Output the string & Inner loop iterator
- Eax → Transformation of the number for the input and output
- Ecx → Outer loop iterator
- Al → Store the character and output the character

Variables inside:

- Col → Store the column number
- Row → Store the row number
- Character → Store the character which will be repeatedly printed

In order to give a friendly interface, we add some strings:

- hInput → "Input a character"
- cInput → "Input the times you wanna print the character in one line"
- rInput → "Input the rows you wanna print"
- Output → "Output"

b) For content two, we can turn the requirements into the following content:

- Declare three arrays with same sizes, two for storing the input numbers, another one for sum of the corresponding elements of the arrays.
- Declare a variable for storing the size of the arrays.
- Loops for number inputs and sum the corresponding elements.

Register inside:

- Edx → Output the string

- Eax → Transformation of the number for the input and output
- Ecx → Loop iterator
- Esi → Index for the arrays
- Al → Store the character and output the character

Variables inside:
- ArrayA & ArrayB → Storing two sets of the numbers
- ArrayC → Storing the sum of the corresponding elements
- ArraySize → Storing the size of the arrays

In order to give a friendly interface, we add some strings:
- ArrA → "Array A"
- ArrB → "Array B"
- ArrC → "Array C"
- InputSize → "Input the array size"
- InputNum → "Input a number"
- Output → "Result"

c) For content three, we can turn the requirements into the following content:
- Declare three arrays to store the data and the size is five
- Calculate the average of each array and the average of total elements by using div instruction
- Find the maximal number by using branch.

Registers inside have the same function as content two.

Variables inside:
- ArrayA & ArrayB & ArrayC → Store the data
- avgA & avgB & avgC → Store the average number of the array respectively
- Num → Divisor
- Max → Maximal number

In order to give a friendly interface, we add some strings:
- hInput → "Input five numbers"
- hOutput → "The average of the array is"
- AllOutput → "The average of all the elements is"
- AllSum → "The sum of all the elements is"
- maxOne → "The max element is"

In conclusion:
- Since we use the "dword" type variable, when we do the iteration, the index need to be increased by four.
- Carefully using the mul/div instruction: They are the unary instruction, and the operand cannot be the instant number. In addition, we need to initialize the edx

register with the number zero when we do the division since we use the “dword” type variable. If we do not initialize the edx register, the compiler will pass code but the code will fail to run correctly since integer overflow occurs.