Contents

1 hw 0x5 c++

1.1 selection sort

```cpp
#include<bits/stdc++.h>
using namespace std;

int num[1000];
int n;

void output(){
    for(int i = 0; i < n; i++){
        cout << num[i] << " ";
    }
    cout << endl;
}

void selection_sort(){
    // selection sort
    // 每次都抓從 i 以後的最小值換到前面
    for(int i = 0; i < n-1; i++){
        int min = i;
        for(int j = i+1; j < n; j++){
            if(num[j] < num[min]){
                min = j;
            }
        }
        // 並且將最小值與當前那陣列位址的值做交換
        swap(num[i], num[min]);
        output();
    }
}

void input(){
    for(int i = 0; i < n; i++){
        cin >> num[i];
    }
}

int main(){

    bool flag = false;

    while(cin >> n && n){

        input();
        selection_sort();

        if(flag){
            cout << endl;
        }

        output();

        // 輸出中位數
        if(n % 2){
            cout << num[n/2];
        }
        else{
            cout << num[n/2] << " " << num[n/2-1];
        }

        cout << endl;
    }
}
```

2 hw 0x5 assembly language

2.1 header inc

```
INCLUDE Irvine32.inc

input PROTO,
  n: DWORD

sort PROTO,
  n: DWORD,
  arr: DWORD

output PROTO,
  n: DWORD,
  arr: DWORD
```

2.2 main asm

```
.686
.model flat, stdcall
.stack 4096

INCLUDE header.inc

.data
  flag DWORD ?
  N DWORD ?

.code

main PROC

  mov eax, 0
  mov flag, eax

begin:
  push ebp                ; 把 ebp 的 old address 存起來
  mov ebp, esp              ; 要把 ebp 拉到 esp 的位址

  call ReadInt
  cmp eax, 0
  jle fin                 ; 如果輸入等於 0 就判斷結束

  push eax                ; 把 n 存進 stack
      為了傳給其他 .asm
  mov N, eax

  call input

  cmp flag, 0
  je noendl

  call Crlf

noendl:
  push edi                ; push 存 head
  push N                  ; push N
  call sort

  mov eax, 1
  mov flag, eax

  mov eax, N
  mov ebx, 2
  cdq
  div ebx
```

```
  cmp edx, 1
  je odd

  mov esi,eax

  mov eax, [edi+esi*4]
  cmp eax, 0
  jg positive1

  call WriteInt
  jmp space

positive1:
  call WriteDec

space:
  mov al, ' '
  call WriteChar

  dec esi
  mov eax, [edi+esi*4]
  cmp eax, 0
  jg positive2

  call WriteInt
  jmp endl

positive2:
  call WriteDec

endl:
  call Crlf
  jmp begin

odd:
  mov esi,eax

  mov eax, [edi+esi*4]
  cmp eax, 0
  jg positive3

  call WriteInt
  jmp endl

positive3:
  call WriteDec
  jmp endl            ; 回來後繼續新的一趟吃輸入

fin:                  ; 函式結束
  mov esp, ebp          ; 最後也要記得清空
  pop ebp
  INVOKE ExitProcess, 0       ; 結束

main ENDP

END main
```

2.3 input asm

```
INCLUDE header.inc

.data
  array DWORD 1000 DUP(?)

.code

input PROC,             ; 在 PROTO 有宣告參數
    這裡也要宣告
  n: DWORD

  ;push ebp           ; 有 PROC, n:DWORD assembly
      就會自動 push ebp
  mov ebp, esp

  mov ecx, DWORD PTR[ebp+8]   ; ecx 存 N
      (記憶體位址在現在的 ebp 往上 +8)
  mov ebx, ecx          ; 因為往後陣列頭是固定的
      所以要先存進別的暫存器

  lea edi, array          ; 用 lea 不用 OFFSET
      array，一樣是直接傳 array 頭的記憶體"位址"
  mov esi, 0            ; 先將 esi 設為 0 (現在 esi
      相當於 c++ 程式中的 i)

L1:
  call ReadInt          ; 建構 array
  mov [edi+esi], eax        ; 在陣列頭 edi +
      esi(i*4格) 的位址存每一格的數值
  add esi, 4            ; 手動把 esi 每次都 +4
      開新空間

  dec ebx
  cmp ebx, 0
  jne L1

fin:                ; 函式結束
  ;mov esp, ebp
  ;pop ebp            ; 不用 pop 因為前面也沒有 push
  ret 8             ; 要還 PROC, n:DWORD assembly
      借用的空間

input ENDP

END                 ; 這邊不用接 input
    因為只是副程式的結束而已
```

2.4 sort asm

```
INCLUDE header.inc
.data
  min DWORD ?

.code
sort PROC,            ; 在 PROTO 有宣告參數
    這裡也要宣告
  n: DWORD,
  arr: DWORD

  ;push ebp
  mov ebp, esp

  mov ecx, DWORD PTR[ebp+8]   ; ecx 存 N
  mov edi, DWORD PTR[ebp+12]    ; edi 存 head

  mov esi, 0            ; esi = i (as c++)

L1:
  mov eax, ecx
  dec eax
  cmp eax, esi          ; if i == n, jmp
  je fin

  mov eax, [edi+esi*4]
  mov min, esi          ; c++ 裡的 min = i

  push esi            ; 先把 esi = i 存起來
  add esi, 1
L2:
                  ; j = i + 1 = esi + 4
  cmp ecx, esi          ; if j == n , jmp
  je swap

  mov eax, [edi+esi*4]
  mov edx, min
  cmp eax, [edi+edx*4]      ; 比較 array[j] 和
      array[min]
  jge conti           ; 大於等於就跳轉
```

```
38
39    mov min, esi          ; min = j
40
41  conti:
42
43    add esi, 1        ; j++
44    jmp L2
45
46  swap:
47    pop esi
48    mov eax, [edi+esi*4]      ; tmp = array[i]
49    mov edx, min
50    mov ebx, [edi+edx*4]      ; tmp = array[min]
51    xchg eax, ebx         ; swap array[i], array[min]
52    mov [edi+esi*4], eax
53    mov [edi+edx*4], ebx
54
55    mov ebx, esi
56    push edi
57    push ecx            ; 重新 push N
58    call output
59
60    mov esi, ebx
61    add esi, 1
62    jmp L1
63
64  fin:                ; 函式結束
65    mov esp, ebp          ; 最後也要記得清空
66    ;pop ebp
67    ret 8             ; 結束
68
69  sort ENDP
70
71  END
```

2.5 output asm

```
1   INCLUDE header.inc
2
3   .code
4
5   output PROC,              ; 在 PROTO 有宣告參數
        這裡也要宣告
6     n: DWORD,
7     arr: DWORD
8
9     ;push ebp
10    mov ebp, esp
11
12    mov edi, DWORD PTR[ebp+12]      ; edi 存 head
13    mov ecx, DWORD PTR[ebp+8]     ; 在 input 已經存過 N
          了
14
15    mov esi, 0              ; esi = i (as c++)
16
17  L1:
18    cmp ecx, esi
19    je fin
20
21    mov eax, [edi+esi*4]
22    cmp eax, 0
23    jge positive
24
25    call WriteInt
26    jmp space
27
28  positive:
29    call WriteDec
30
31  space:
32    mov al, ' '
33    call WriteChar
34
35    inc esi
36    jmp L1
37
38
39  fin:                  ; 函式結束
40    call Crlf
41    mov esp, ebp            ; 最後也要記得清空
42    ;pop ebp
43    ret 8
44
45  output ENDP
46
47  END
```