

Contents

1	hw 0x5 c++	1
1.1	selection sort	1
2	hw 0x5 assembly language	1
2.1	header inc	1
2.2	main asm	1
2.3	input asm	2
2.4	sort asm	2
2.5	output asm	3

1 hw 0x5 c++

1.1 selection sort

```

1 #include<bits/stdc++.h>
2 using namespace std;
3
4 int num[1000];
5 int n;
6
7 void output(){
8     for(int i = 0; i < n; i++){
9         cout << num[i] << " ";
10    }
11    cout << endl;
12 }
13
14 void selection_sort(){
15     // selection sort
16     // 每次都抓從 i 以後的最小值換到前面
17     for(int i = 0; i < n-1; i++){
18         int min = i;
19         for(int j = i+1; j < n; j++){
20             if(num[j] < num[min]){
21                 min = j;
22             }
23         }
24         // 並且將最小值與當前那陣列位址的值做交換
25         swap(num[i], num[min]);
26         output();
27     }
28 }
29
30 void input(){
31     for(int i = 0; i < n; i++){
32         cin >> num[i];
33     }
34 }
35
36 int main(){
37
38     bool flag = false;
39
40     while(cin >> n && n){
41
42         input();
43         selection_sort();
44
45         if(flag){
46             cout << endl;
47         }
48
49         output();
50
51         // 輸出中位數
52         if(n % 2){
53             cout << num[n/2];
54         }
55         else{
56             cout << num[n/2] << " " << num[n/2-1];
57         }
58     }

```

```

59     cout << endl;
60 }
61 }

```

2 hw 0x5 assembly language

2.1 header inc

```

1 INCLUDE Irvine32.inc
2
3 input PROTO,
4     n: DWORD
5
6 sort PROTO,
7     n: DWORD,
8     arr: DWORD
9
10 output PROTO,
11     n: DWORD,
12     arr: DWORD

```

2.2 main asm

```

1 .686
2 .model flat, stdcall
3 .stack 4096
4
5 INCLUDE header.inc
6
7 .data
8     flag DWORD ?
9     N DWORD ?
10
11 .code
12
13 main PROC
14
15     mov eax, 0
16     mov flag, eax
17
18 begin:
19     push ebp                ; 把 ebp 的 old address 存起來
20     mov ebp, esp            ; 要把 ebp 拉到 esp 的位址
21
22     call ReadInt
23     cmp eax, 0
24     jle fin                  ; 如果輸入等於 0 就判斷結束
25
26     push eax                 ; 把 n 存進 stack
27                             ; 為了傳給其他 .asm
28     mov N, eax
29
30     call input
31
32     cmp flag, 0
33     je noendl
34
35     call Crlf
36
37 noendl:
38     push edi                 ; push 存 head
39     push N                   ; push N
40     call sort
41
42     mov eax, 1
43     mov flag, eax
44
45     mov eax, N
46     mov ebx, 2
47     cdq
48     div ebx

```

```

48  cmp edx, 1
49  je odd
50
51  mov esi, eax
52
53  mov eax, [edi+esi*4]
54  cmp eax, 0
55  jg positive1
56
57  call WriteInt
58  jmp space
59
60 positive1:
61  call WriteDec
62
63 space:
64  mov al, ' '
65  call WriteChar
66
67  dec esi
68  mov eax, [edi+esi*4]
69  cmp eax, 0
70  jg positive2
71
72  call WriteInt
73  jmp endl
74
75 positive2:
76  call WriteDec
77
78 endl:
79  call Crlf
80  jmp begin
81
82 odd:
83  mov esi, eax
84
85  mov eax, [edi+esi*4]
86  cmp eax, 0
87  jg positive3
88
89  call WriteInt
90  jmp endl
91
92 positive3:
93  call WriteDec
94  jmp endl          ; 回來後繼續新的一趟吃輸入
95
96 fin:              ; 函式結束
97  mov esp, ebp      ; 最後也要記得清空
98  pop ebp
99  INVOKE ExitProcess, 0 ; 結束
100
101 main ENDP
102
103 END main

```

2.3 input asm

```

1  INCLUDE header.inc
2
3  .data
4  array DWORD 1000 DUP(?)
5
6  .code
7
8  input PROC,          ; 在 PROTO 有宣告參數
   這裡也要宣告
9  n: DWORD
10
11 ;push ebp            ; 有 PROC, n:DWORD assembly
   就會自動 push ebp
12 mov ebp, esp
13

```

```

14  mov ecx, DWORD PTR[ebp+8] ; ecx 存 N
   (記憶體位址在現在的 ebp 往上 +8)
15  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
16
17  lea edi, array          ; 用 lea 不用 OFFSET
   array, 一樣是直接傳 array 頭的記憶體"位址"
18  mov esi, 0              ; 先將 esi 設為 0 (現在 esi
   相當於 c++ 程式中的 i)
19
20 L1:
21  call ReadInt            ; 建構 array
22  mov [edi+esi], eax      ; 在陣列頭 edi +
   esi(i*4格) 的位址存每一格的數值
23  add esi, 4              ; 手動把 esi 每次都 +4
   開新空間
24
25  dec ebx
26  cmp ebx, 0
27  jne L1
28
29 fin:                    ; 函式結束
30 ;mov esp, ebp
31 ;pop ebp                ; 不用 pop 因為前面也沒有 push
32 ret 8                   ; 要還 PROC, n:DWORD assembly
   借用的空間
33
34 input ENDP
35
36 END                      ; 這邊不用接 input
   因為只是副程式的結束而已

```

2.4 sort asm

```

1  INCLUDE header.inc
2  .data
3  min DWORD ?
4
5  .code
6  sort PROC,            ; 在 PROTO 有宣告參數
   這裡也要宣告
7  n: DWORD,
8  arr: DWORD
9
10 ;push ebp
11 mov ebp, esp
12
13 mov ecx, DWORD PTR[ebp+8] ; ecx 存 N
14 mov edi, DWORD PTR[ebp+12] ; edi 存 head
15
16 mov esi, 0              ; esi = i (as c++)
17
18 L1:
19  mov eax, ecx
20  dec eax
21  cmp eax, esi            ; if i == n, jmp
22  je fin
23
24  mov eax, [edi+esi*4]
25  mov min, esi            ; c++ 裡的 min = i
26
27  push esi                ; 先把 esi = i 存起來
28  add esi, 1
29 L2:
30 ; j = i + 1 = esi + 4
31  cmp ecx, esi            ; if j == n, jmp
32  je swap
33
34  mov eax, [edi+esi*4]
35  mov edx, min
36  cmp eax, [edi+edx*4]    ; 比較 array[j] 和
   array[min]
37  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

```

```

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

```

2.5 output asm

```

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

```