

Contents

1	exams	1
1.1	star pyramid	1
2	homeworks	1
2.1	print prime asm	1
2.2	print prime cpp	2
2.3	max and min	2

1 exams

1.1 star pyramid

```

1 .686                      ; 409262449 吳家萱 10/27 P1
2 .model flat, stdcall
3 .stack 4096
4
5 INCLUDE Irvine32.inc
6 ExitProcess PROTO, dwExitCode: DWORD
7
8 .data
9 n SDWORD ?                ; 變數宣告
10 space SDWORD ?
11 b SDWORD ?
12 format1 BYTE " ", 0
13 format2 BYTE "x", 0
14
15 .code
16 main PROC
17
18 conti:
19     call ReadDec           ; 用 Dec 讀取
20     mov n, eax             ; 讀取n
21     mov eax, n
22     mov space, eax
23     cmp n, 0               ; 如果 n == 0 結束程式
24     je finish
25     mov b, 1               ; 初始化 b
26
27     mov eax, n              ; 要輸出的星星層數
28     add eax, n
29     mov n, eax
30     mov ecx, n
31     call printree
32
33     jmp conti
34
35 finish:
36     invoke ExitProcess, 0   ; 離開程式
37     main ENDP
38
39 printree PROC
40
41 L1:
42     push ecx
43     mov eax, space
44     sub eax, b
45     cmp eax, 0              ; 如果 eax < 0
46     ; 停止迴圈繼續往下
47     jbe cont
48     mov ecx, eax
49
50 L2:                          ; 迴圈空白
51     mov edx, OFFSET format1
52     call WriteString
53     loop L2
54
55 cont:
56     mov eax, b
57     add eax, b
58     sub eax, 1
59     mov ecx, eax
60
61 L3:                          ; 迴圈*

```

```

59     mov edx, OFFSET format2
60     call WriteString
61     loop L3
62
63     call Crlf
64
65     pop ecx                  ; 把 L1 的 ecx 拿出來 - 1
66     mov eax, ecx
67     sub eax, 1
68     mov ecx, eax
69     mov eax, b
70     add eax, 1
71     mov b, eax
72     loop L1
73
74     ret                      ; 要記得 ret QQ
75     ; 不然會超出組建範圍
76     printree ENDP
77 END main

```

2 homeworks

2.1 print prime asm

```

1 .686p
2 .model flat, C
3 .stack 4096
4
5 includelib legacy_stdio_definitions.lib
6
7 ExitProcess PROTO, dwExitCode: DWORD
8
9 printf PROTO C,            ; Std C library function
10    format: PTR BYTE, args: VARARG ; (in libc.lib)
11
12 scanf PROTO C,             ; Std C library function
13    format: PTR BYTE, args: VARARG ; (in libc.lib)
14
15 AllPrimeNumbers PROTO C,
16    dwInt: DWORD            ; in main.cpp
17
18 TAB = 9
19
20 .data
21     ; define your variables here
22     format1 BYTE "Give me an interger owo : ", 0
23
24     format2 BYTE "%d", 0
25
26     format3 BYTE "OK, so the integer N is... %d !", 0dh,
27     0ah, 0
28
29     format4 BYTE "%d", 0dh, 0ah, 0
30
31     format5 BYTE "And all prime numbers less than N is : ", 0
32
33     newLine BYTE 0dh, 0ah, 0
34
35     space BYTE " ", 0
36
37     Number DWORD ?
38
39 .code
40 asmMain PROC C
41     push ebp
42     mov ebp, esp
43
44     ; write your assembly code here
45     INVOKE printf, ADDR format1
46
47     INVOKE printf, ADDR newLine ; 換行

```

```

48
49 INVOKE scanf, ADDR format2, ADDR Number ;
    scanf("%d", &Number)
50
51 INVOKE printf, ADDR format3, Number ; printf("N
    : %d\n", Number)
52
53 INVOKE printf, ADDR format5
54
55 INVOKE printf, ADDR newLine
56
57 INVOKE AllPrimeNumbers, Number ; return value
    in EAX
58
59 INVOKE printf, ADDR Format4, eax
60
61 mov eax, 0 ; return 0
62 pop ebp
63
64 ret
65
66 asmMain ENDP
67
68
69 END

```

2.2 print prime cpp

```

1 #include <cstdio>
2 #include <cstdlib>
3 #include <string.h>
4
5 using namespace std;
6
7 extern "C" int asmMain();
8
9 extern "C" int AllPrimeNumbers(int);
10
11 int main()
12 {
13     int num = asmMain();
14
15     system("pause");
16
17     return 0;
18 }
19
20 int AllPrimeNumbers(int num)
21 {
22     int cnt = 0;
23     bool prime[1000000];
24
25     memset(prime, 0, sizeof(prime));
26
27     for (int i = 2; i < num; i++) {
28
29         if (!prime[i]) {
30
31             printf("%d ", i);
32             cnt++;
33
34             for (int j = i + i; j < num; j += i) {
35
36                 prime[j] = true;
37
38             }
39         }
40     }
41     printf(" \nCnt is : ");
42
43     return cnt;
44 }

```

2.3 max and min

```

1 .686 ; 指定微處理器模式
2 .model flat, stdcall ; model ->
    定義記憶體模式, stdcall -> windows API
    的標準呼叫約定
3 .stack 4096 ; 堆疊 ->
    用來保存傳遞的參數 & 保存調用函數的程式位址
4
5 INCLUDE Irvine32.inc
6 ExitProcess PROTO, dwExitCode: DWORD ; windos 服務,
    原型包含函數名, PROTO 關鍵字, 輸入參數列表
7
8 .data ; 組譯時配置
9 arr SDWORD 200 DUP(0) ; 陣列宣告 SDWORD
10 n SDWORD ? ; 變數宣告
11 small SDWORD ?
12 large SDWORD ?
13 format1 BYTE "Min = ", 0 ; 輸出字串宣告
14 format2 BYTE "Max = ", 0
15
16 .code
17
18 main PROC
19     call ReadInt ; Irvine32.lib 內建函式之一
    可用來讀取 Int
20     mov esi, OFFSET arr ; 把 arr 的位址放進 esi
    暫存器
21     mov n, eax ; ReadInt
    讀取的輸入會先暫存在 eax 中 所以要 mov 進 n 裡
22     mov ecx, n ; 迴圈寫法, ecx 是 計數器,
    原理: ecx != 0 繼續跑 -> 總共跑 n 次
23
24 L1: ; 這邊在做讀取 input 的單層 for
    迴圈的工作
25     call ReadInt
26     mov [esi], eax ; 把 eax 讀取輸入的值放入
    arr, esi 指向陣列位址的頭
27     add esi, TYPE arr ; 往上指下一位 ( TYPE
    array 大小 ) 的 esi
28     loop L1 ; 迴圈循環
29
30     sub n, 1 ; n - 1 因為外層迴圈只要跑 n
    - 1 次就好
31     mov ecx, n ; 在 call mini proc 前先把
    ecx 準備好
32     mov esi, OFFSET arr ; 把 esi 歸 0
    回到記憶體位址的頭
33     call mini
34
35     mov edx, OFFSET format1 ; 因為 WriteString 吃
    edx 暫存器
36     call WriteString
37     cmp eax, 0 ; 如果 small 是正整數 or 0
38     jge positive1
39     call WriteInt ; WriteInt 吃 eax,
    輸出負值的 small
40     call Crlf ; 內建換行寫法
41     jmp conti ; 如果這邊不跳過
    程式就會繼續往下執行 positive1
42
43 positive1: ; 如果 small 是正數
44     call WriteDec ; 就輸出 decimal ( 沒有負數
    )
45     call Crlf
46
47 conti:
48     mov ecx, n
49     mov esi, OFFSET arr
50     call maxi
51
52     mov edx, OFFSET format2
53     call WriteString

```

```
54     cmp eax, 0
55     jge positive2
56     call WriteInt
57     call Crlf
58     jmp finish
59
60 positive2:
61     call WriteDec
62     call Crlf
63
64 finish:
65     invoke ExitProcess, 0      ; 離開程式
66     main ENDP                 ; main 結束
67
68 mini PROC
69     push esi                  ; 先把 esi 目前位址指向處
70     push ecx                  ; 同理，放入堆疊是因為等等
71     push ecx                  ; 出去後暫存器會回到 push 進去的狀態，
72     pop esi                   ; 所以內部就算有改動也沒關係
73     mov eax, [esi]
74     mov small, eax            ; 先預設 small 初始值是
75     arr[0]
76     add esi, TYPE arr
77     mov ecx, n
78 L2:
79     mov eax, [esi]
80     add esi, TYPE arr
81     cmp eax, small            ;
82     ; 因為不能兩邊都是記憶體位址，所以要先 mov 進 eax
83     ; 才去做比較
84     jge skip1                 ; 如果大於等於就跳到 skip1
85     mov small, eax            ; 否則，就重新紀錄當前
86     small 是誰
87 skip1:
88     loop L2
89
90     mov eax, small            ; 等等回到 main proc 後
91     WriteInt 要用到 eax，現在先存好
92     pop ecx                   ; 後進去 ecx 的要先出來
93     pop esi                   ; 要 pop 是為了讓他們回到進來
94     ; mini proc 前的狀態
95     ret                       ; return 回去
96
97 mini ENDP
98
99 maxi PROC                    ; 所有理論皆與 mini 同理
100     push esi
101     push ecx
102     mov eax, [esi]
103     mov large, eax
104     add esi, TYPE arr
105     mov ecx, n
106 L3:
107     mov eax, [esi]
108     add esi, TYPE arr
109     cmp eax, large
110     jle skip2
111     mov large, eax
112 skip2:
113     loop L3
114
115     mov eax, large
116     pop ecx
117     pop esi
118     ret
119
120 maxi ENDP
121
122 END main
```