

快速指引 APICS/C++

- 循序執行 (循序結構)
- 指定
- 輸入/輸出
- 邏輯、條件判斷 (if, 選擇結構)
- if/while 比較
- while/for 迴圈 (重複結構)
- 一維陣列
- 多重迴圈/二維陣列
- 二分搜尋
- 字元型別/字元陣列
- 初學常見錯誤
- 其他考試注意重點
- 常用函式
- 常用數值與技巧

程式循序執行

- 我們寫程式告訴電腦要做什麼：
一句話接著另一句話，一個指令接著另一個指令
有時間上的順序 (和數學演算不一樣)
- 分號或左右大括弧包起來的區塊作為一句話的結束
- 可以寫在同一列但是便於閱讀分列寫

利用縮排 (開頭空四個往後縮) 表示語意的結構



X = y;

將 = 右邊的變數 y 當時的值
複製設定成左邊變數 x 的值

做完這個動作後 x 和 y 之間
就沒有任何關係了



輸出入

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int x, y;
7     cin >> x >> y;
8     cout << x + y << endl;
9     return 0;
10 }
```

note1

操作者輸入

```
1 2
3
Process returned 0 (0x0)    execution time : 2.328 s
Press ENTER to continue.
```



輸出入

這兩段程式有同樣的作用和結果

```
int x, y;  
  
cin >> x;      // 使用者輸入測資會用空格或換行分開，  
                // 但程式無須特別處理空格或換行  
  
cin >> y;  
  
cout << x;  
  
cout << " ";    // 輸出用空格分開  
  
cout << y;  
  
cout << endl;   // 用換行結束最後一項輸出
```

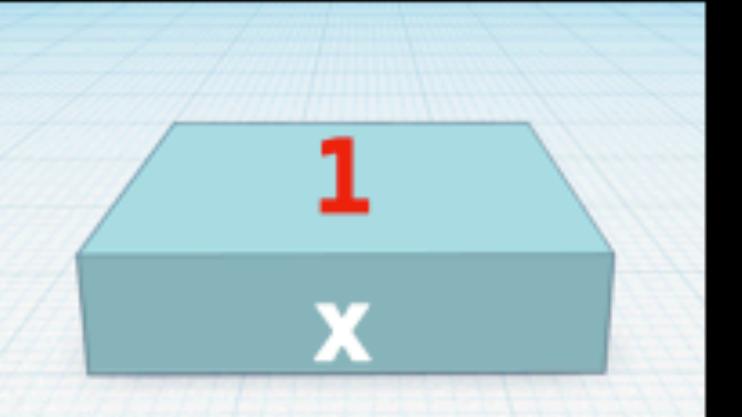
2019 (C) Elton Huang

```
int x, y;  
  
cin >> x >> y; // 使用者輸入測資會用空格或換行分開，  
                  // 但程式無須特別處理空個活換行  
  
cout << x << " " << y << endl;  
                // 輸出用空格分開  
                // 用換行結束最後一項輸出
```

2019 (C) Elton Huang

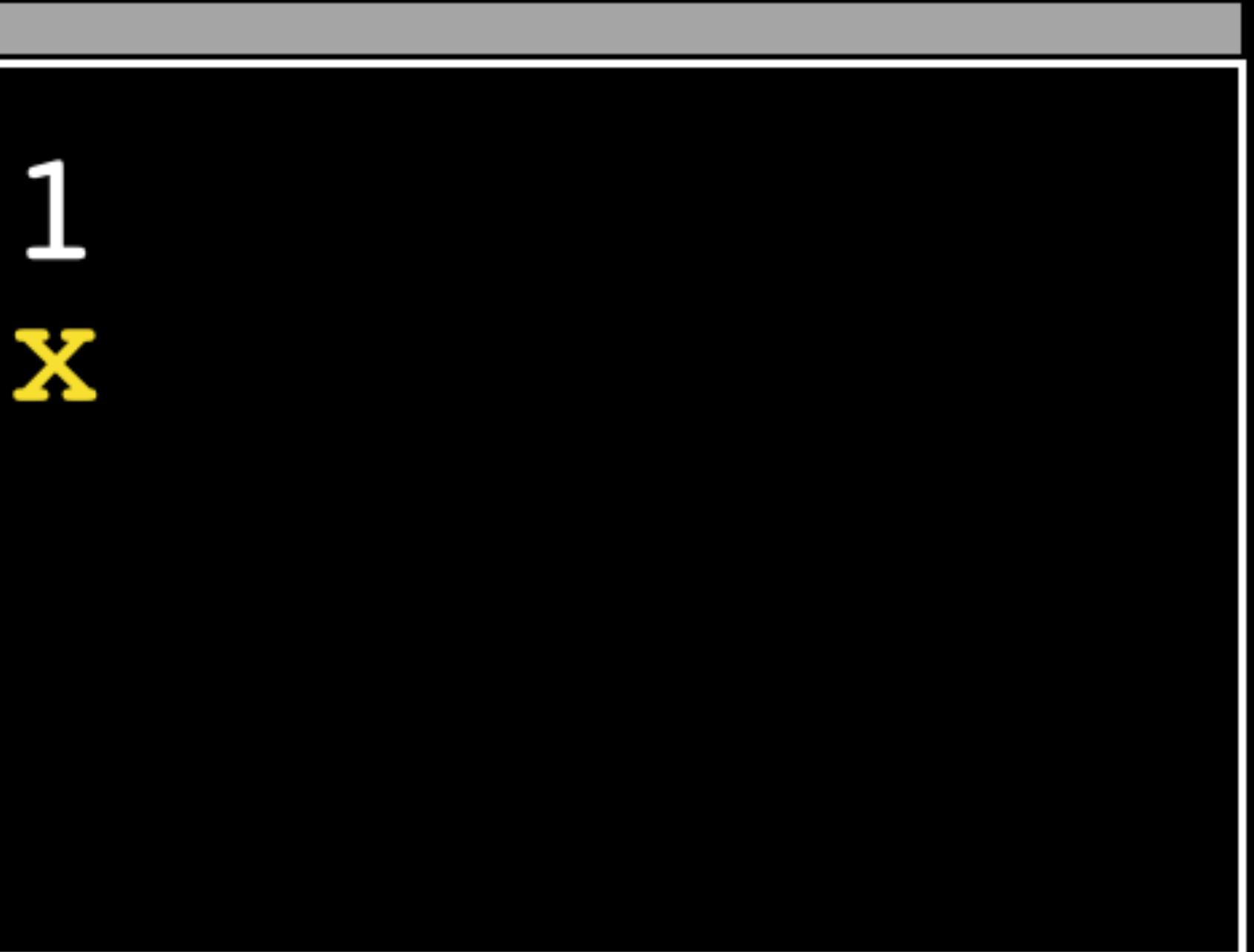


```
int x = 1;
```



```
cout << x << endl;
```

```
cout << "x" << endl;
```



```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int x;
7     cin >> x;
8     if (5 < x && x < 9) {
9         cout << x << " is wthin (5..9)" << endl;
10    } else {
11        cout << x << " is out of range." << endl;
12    }
13    return 0;
14 }
```



邏輯 · 條件判斷 · 選擇結構

Elton Huang (C) 2020

操作者輸入
6

6 is wthin (5..9)

Process returned 0 (0x0) execution time : 6.154 s

Press ENTER to continue.

操作者輸入
12

12 is out of range.

Process returned 0 (0x0) execution time : 5.059 s

Press ENTER to continue.

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int x;
7     cin >> x;
8     if (x < 5 || 9 < x) {
9         cout << x << " is less than 5 or greater than 9." << endl;
10    } else {
11        cout << x << " is not less than 5 nor greater than 9." << endl;
12    }
13    return 0;
14 }
15
```

X 操作者輸入
3
3 is less than 5 or greater than 9.
Process returned 0 (0x0) execution time : 51,908 s
Press ENTER to continue.

X 操作者輸入
7
7 is not less than 5 nor greater than 9.
Process returned 0 (0x0) execution time : 2,049 s
Press ENTER to continue.



```

1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int x;
7     cin >> x;
8     if (x != 1) { ←----- 8 -----→
9         cout << "x is not 1." << endl;
10    } else
11        cout << "x is 1." << endl;
12    return 0;
13 }
14

```

```

1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int x;
7     cin >> x;
8     if (!(x == 1)) { →----- 8 -----←
9         cout << "x is not 1." << endl;
10    } else
11        cout << "x is 1." << endl;
12    return 0;
13 }
14

```

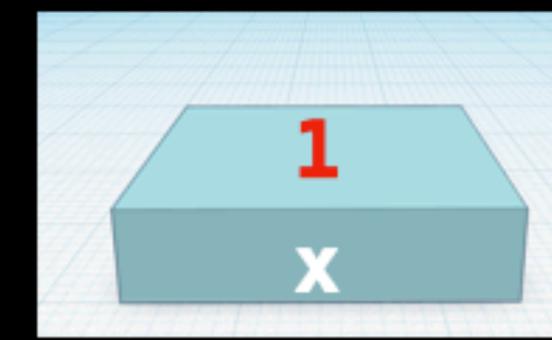
上面兩組程式有同樣的作用

X 操作者 ifnot
2
x is not 1.
Process returned 0 (0x0) execution time : 1.405 s
Press ENTER to continue.

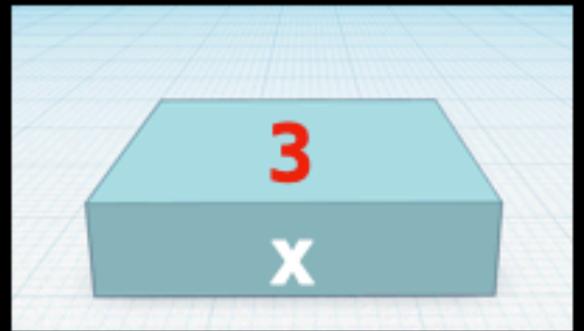
X 操作者 ifnot
1
x is 1.
Process returned 0 (0x0) execution time : 1.373 s
Press ENTER to continue.



```
int x;  
cin >> x;  
if (x < 3) {  
    cout << "x < 3." << endl;  
}  
cout << "end." << endl;
```



```
int x = 0;  
while (x < 3) {  
    cout << x << endl;  
    x = x + 1;  
}  
cout << "end." << endl;
```

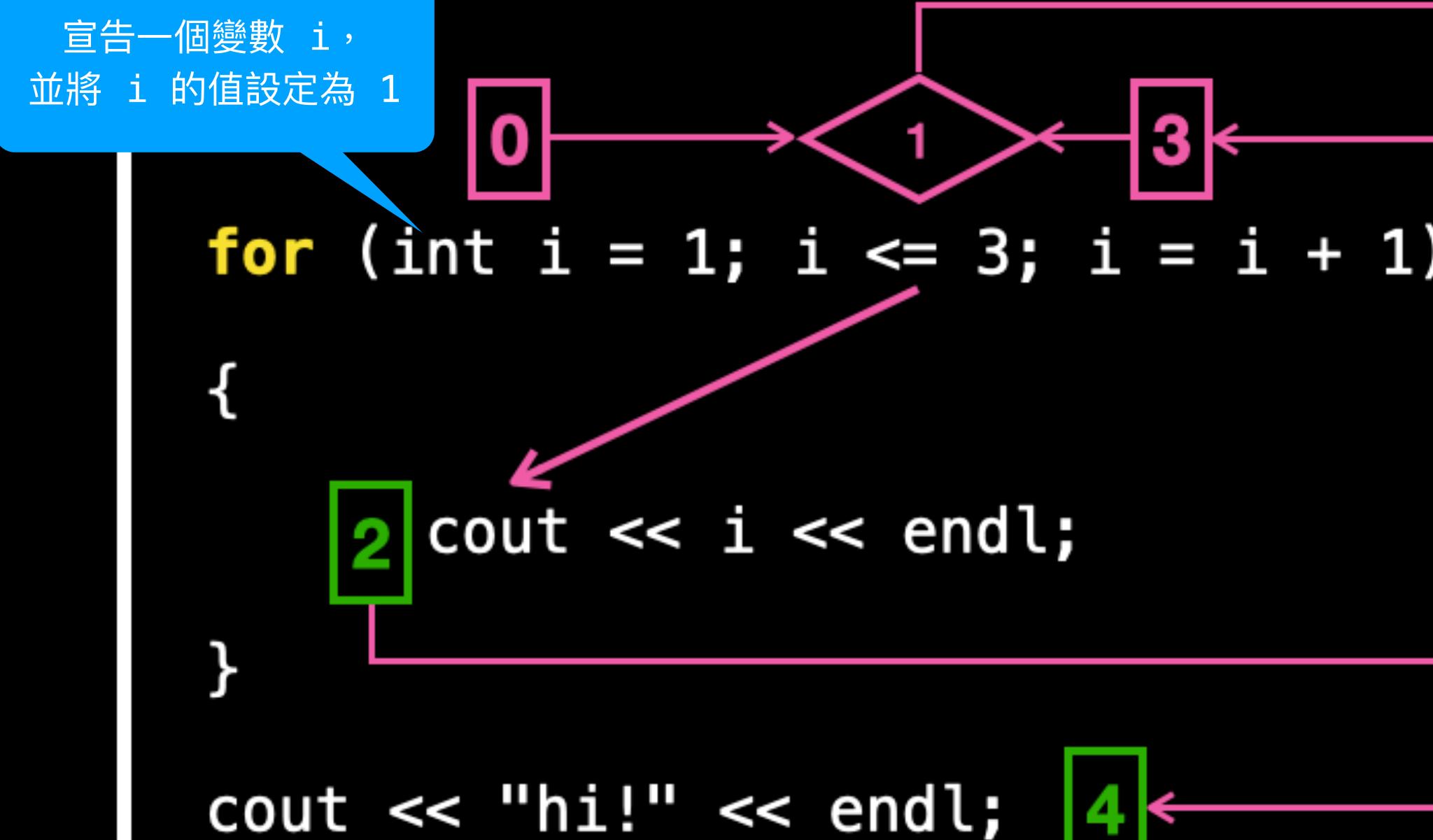
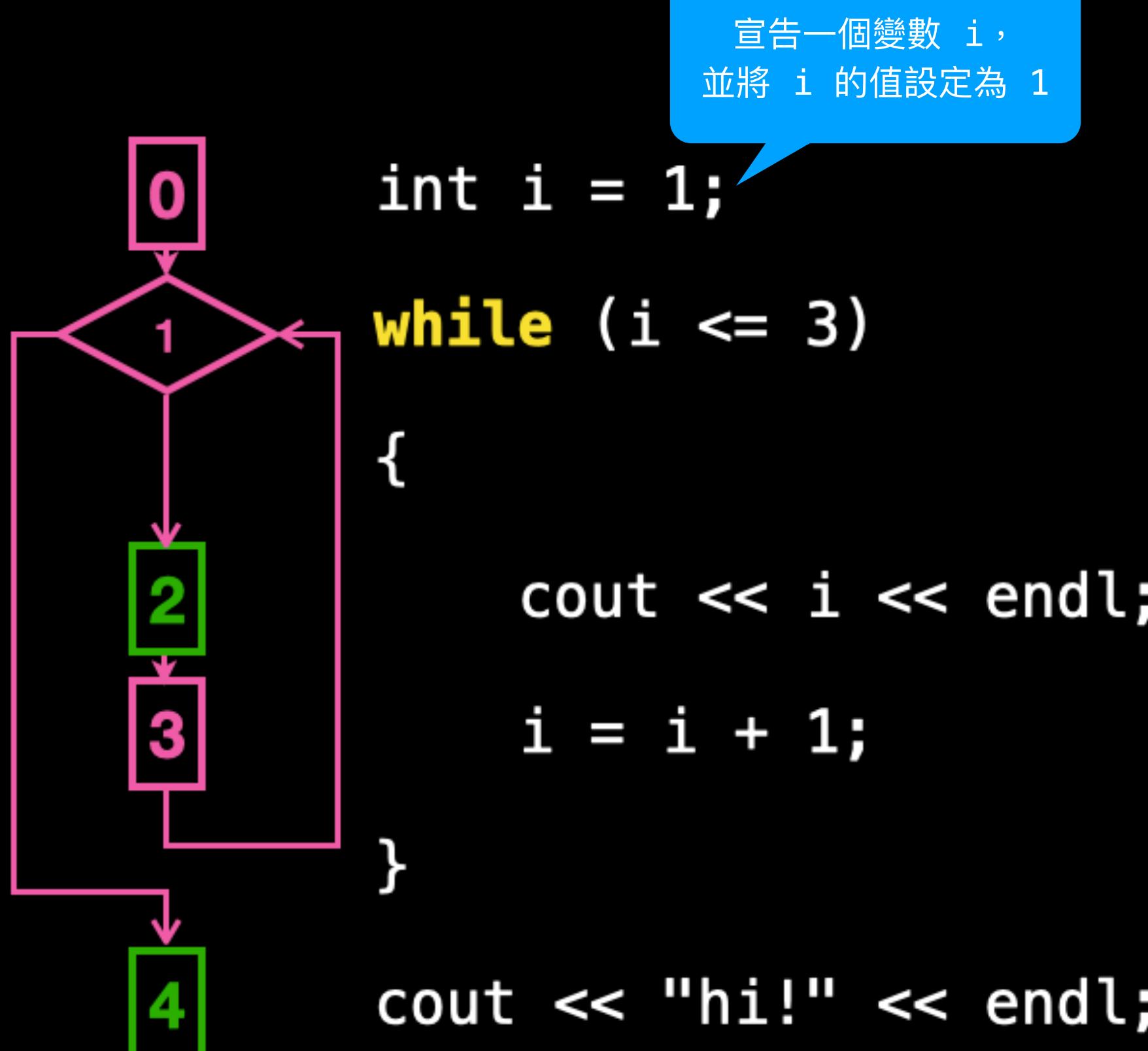


操作者輸入

```
1  
x < 3.  
end.
```



迴圈執行順序



```
int i = 1;  
當 while (i <= 3)  
{  
    cout << i << endl;  
    i = i + 1;  
}  
cout << "hi!" << endl;
```

讓 **i** 從 **i=1** 到 **i=3**, 每次加 **1**, 在這個過程中, 做

```
{  
    cout << i << endl;  
}  
cout << "hi!" << endl;
```



in.cpp

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int i = 0;
7     while (i < 3) {
8         cout << "count to " << i << endl;
9         i = i + 1;
10    }
11    return 0;
12 }
13
```

note2

```
count to 0
count to 1
count to 2

Process returned 0 (0x0) execution time : 0.002 s
Press ENTER to continue.
```

ain.cpp

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     for (int i = 0; i < 3; i = i + 1) {
7         cout << "count to " << i << endl;
8     }
9     return 0;
10 }
```

note2

```
count to 0
count to 1
count to 2

Process returned 0 (0x0) execution time : 0.003 s
Press ENTER to continue.
```

in.cpp

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int i = 0;
7     int x;
8     cin >> x;
9     while (i < x) {
10         cout << "count to " << i << endl;
11         i = i + 1;
12     }
13     return 0;
14 }
15
```

note2

操作者輸入

```
5
count to 0
count to 1
count to 2
count to 3
count to 4

Process returned 0 (0x0)    execution time : 1.781 s
Press ENTER to continue.
```

縮排很重要！！！

縮排很重要！！！

縮排很重要！！！

縮排很重要！！！

程式結構清楚

邏輯才會清楚

用左上方的 tab 鍵

迴圈

Elton Huang (C) 2020

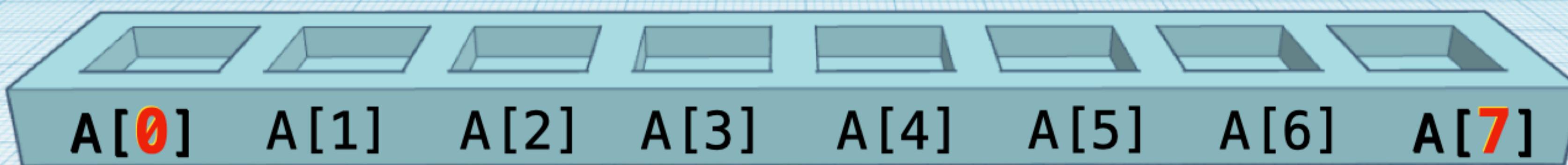


```
int p, q, x, y, s, t, u, v;
```

這是請電腦給我們 8 個可以裝整數的盒子，
每個盒子取 1 個名字來指稱它，8 個盒子，8 個名字

這是請電腦給我們 1 排有 8 個抽屜的櫃子，
每個抽屜可以裝 1 個整數，
這個櫃子用 1 個名字

A 要區分每個抽屜（陣列的元素），就加上序號索引



```
int A [8];
```



```
int births [8];
for (int i = 0; i < 8; i = i + 1)
{
    cin >> births [i];
}

// 回答詢問
int ask;
cin >> ask;
cout << birth [ask - 1] << endl;
```

範例：

輸入：

5

7

1

1

0

3

8

4

2

1

1

輸出

6

A [一個數字]

```
int main ()  
{  
    int A[8]; ← // 這是宣告  
    A[2] = 16;  
}
```

// 是請電腦給我們 1 排有 8 個抽屜的櫃子，
// 每個抽屜可以裝 1 個整數，
// 這個櫃子的名字是 A



```
int main ()  
{  
    int A[8]; ← // 這是宣告  
    A[2] = 16; // 前面會有型別關鍵字  
}
```



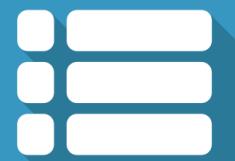
一維陣列

```
int main ()  
{  
    int A[8]; ← // 這是宣告  
    // 這是數字是陣列的大小、  
    // 總共有幾個元素  
    A[2] = 16; // 型別關鍵字  
}
```

```
int main ()
{
    int A[8];
    A[2] = 16;
}
```

// 是告訴電腦我現在用的是
// 序號 2 的抽屜

**// 這是使用陣列裡
// 的某1個元素**



```
int main ()
{
    int A[8];
    A[2] = 16;
}
```

**// 在這裡是
// 把序號 2 的元素指定為 16**

```
int main ()
{
    int A[8];
    A[2] = 16;
}
```

// 這個數字是這個元素的序號



```
int main ()
{
    int A[8];
    cin >> A[2];
    // 這也是使用陣列裡
    // 的某1個元素
}
```

```
int main ()
{
    int A[8];
    cin >> A[2];
    // 把使用者輸入的數字
    // 指定為序號 2 的元素的值
}
```

```
int main ()
{
    int A[8];
    A[2] = 16;
    A[5] = A[2];
    // 這也是使用陣列裡
    // 的某幾個元素
}
```

```
int main ()
{
    int A[8];
    A[2] = 16;
    A[5] = A[2];
    // 把序號 2 的元素的值
    // 指定為序號 5 的元素的值
}
```

重點整理



- 陣列
- 索引從 0 開始
- 陣列以陣列的大小宣告
- 所以索引值從 0 ~ 陣列大小 **-1**

2019 (C) Elton Huang

// 所以 ...

```
int main ()  
{  
    int A[8];  
    A[X] = 16;  
}
```

// 使用陣列元素時，這裡的數字
// 永遠要比宣告的大小來得小



```
int main ()  
{  
    int A[8];  
    for (int i = 0; i <= 8; i = i + 1)  
    {  
        A[8] = ...;  
    }  
}
```



```
for (int i = 0; i < 3; i++) {  
    for (int j = 0; j < 4; j++) {  
        cout << i << "-" << j << " ";  
    }  
}
```

```
0-0 0-1 0-2 0-3 1-0 1-1 1-2 1-3 2-0 2-1 2-2 2-3
```

```
for (int i = 0; i < 3; i++) {  
    cout << i << " ";  
}  
for (int j = 0; j < 4; j++) {  
    cout << j << " ";  
}
```

```
0 1 2 0 1 2 3
```



```
int A[4];
```

A

A[0]	A[1]	A[2]	A[3]
4	7	32	-3

```
A[0] = 4;
```

```
A[3] = -3;
```

```
A[1] = 7;
```

```
A[2] = 32;
```

```
int B[3][4];
```

B

B[0][0]	B[0][1]	B[0][2]	B[0][3]
4	7		
B[1][0]	B[1][1]	B[1][2]	B[1][3]
32			
B[2][0]	B[2][1]	B[2][2]	B[2][3]
		-3	

```
B[0][0] = 4;
```

```
B[2][2] = -3;
```

```
B[0][1] = 7;
```

```
B[1][0] = 32;
```



```
int B [3][4];  
  
for (int i = 0; i < 3; i++)  
{  
    for (int j = 0; j < 4; j++)  
    {  
        cout << B [i] [j] << " ";  
    }  
    cout << endl;  
}
```



	變數	一維陣列	二維陣列
宣告	<code>int x;</code>	<code>int A [4];</code>	<code>int B [3][4];</code>
指定	<code>x = 2;</code>	<code>A [2] = 16;</code>	<code>B [1][2] = 7;</code>
輸入	<code>cin >> x;</code>	<code>cin >> A [2];</code>	<code>cin >> B [1][2];</code>
輸出	<code>cout << x;</code>	<code>cout << A [2];</code>	<code>cout << B [1][2];</code>

```
#include <iostream>
using namespace std;
// 函式 srch 在元素個數為 n 的陣列 a[] 中找尋 lookfor
int srch (int a[], int lookfor, int n) { // 如果找到，回傳所序號，否則回傳 -1
    int left = 0;
    int rite = n - 1; // lookfor 在 a [left] ... a [rite] (包含) 內
    while (rite >= left) {
        int mid = (left + rite) / 2;
        if (a [mid] == lookfor)
            return mid;
        else if (a [mid] < lookfor) // lookfor 會在 a [n] 的右邊
            left = mid + 1;
        else // lookfor < a [mid], lookfor 會在 a [mid] 的左邊
            rite = mid - 1;
    }
    return -1;
}
```



二分搜尋 (二元搜尋)

Elton Huang (C) 2020

如果要找的 lookfor == 10

while 次	0	1	2	3
left	0	4	4	4
rite	7	7	4	3
mid	3	5	4	
a[mid]	9	23	11	



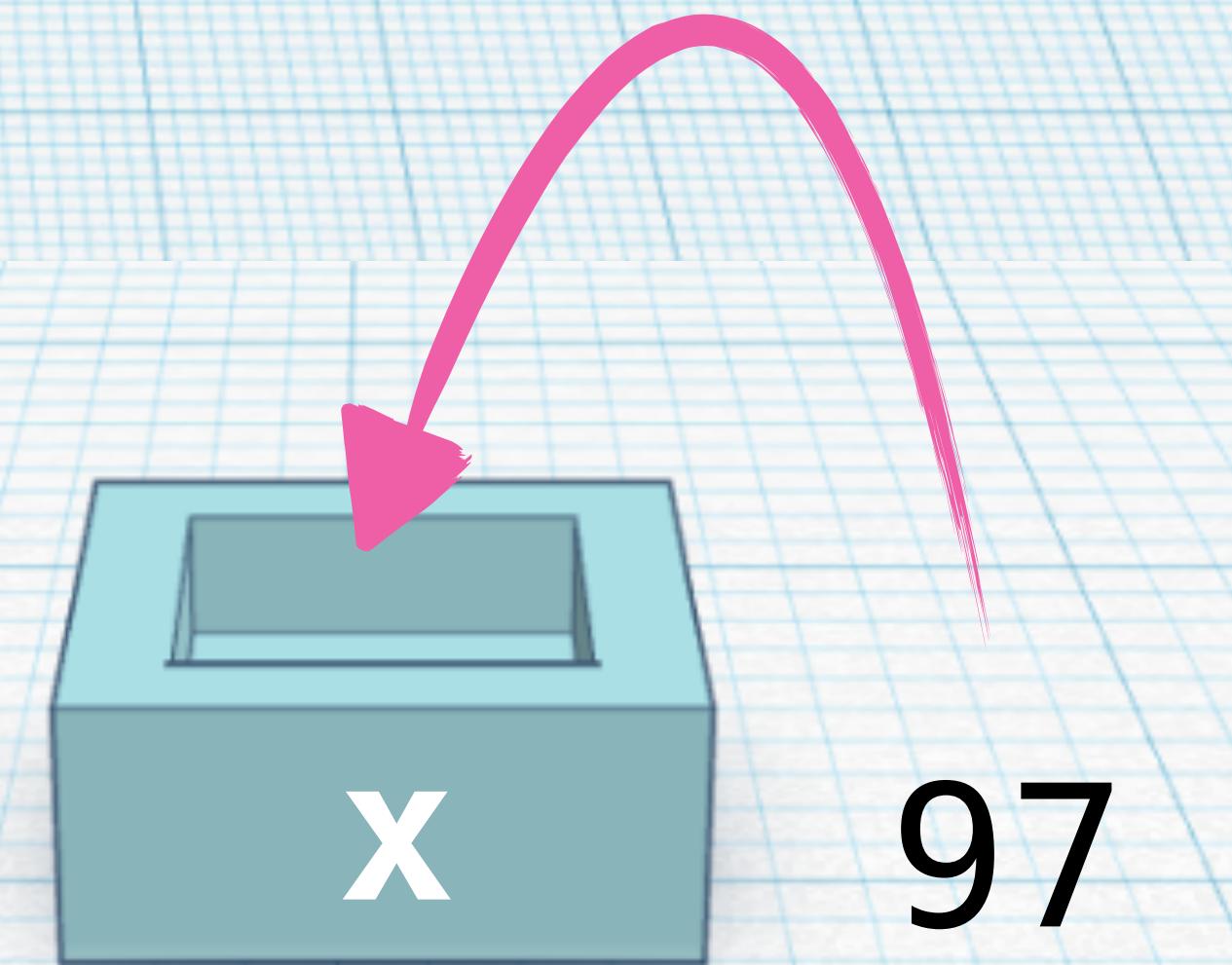
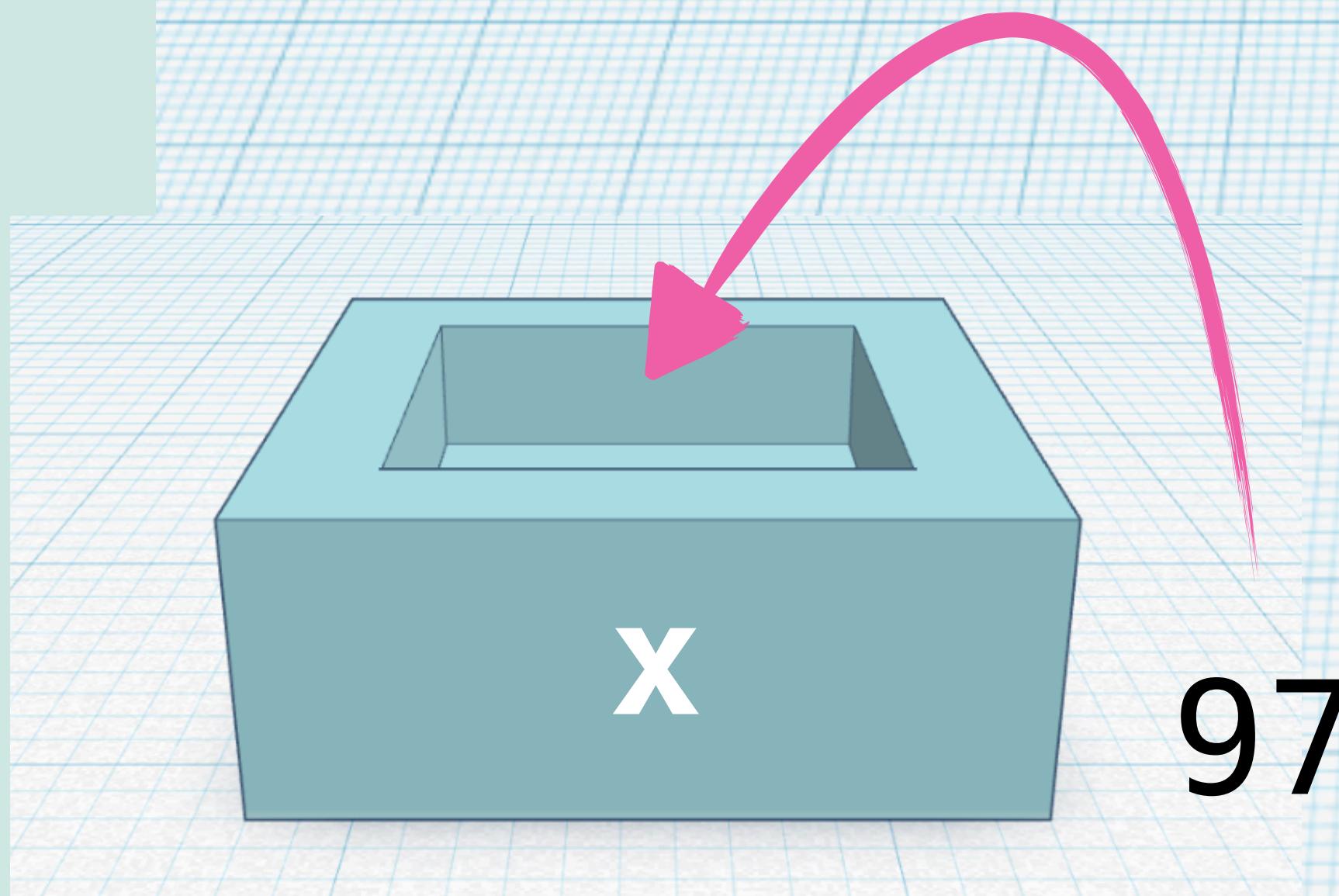
二分搜尋 (二元搜尋)

Elton Huang (C) 2020



```
int x;  
x = 97;
```

```
char x;  
x = 'a';
```

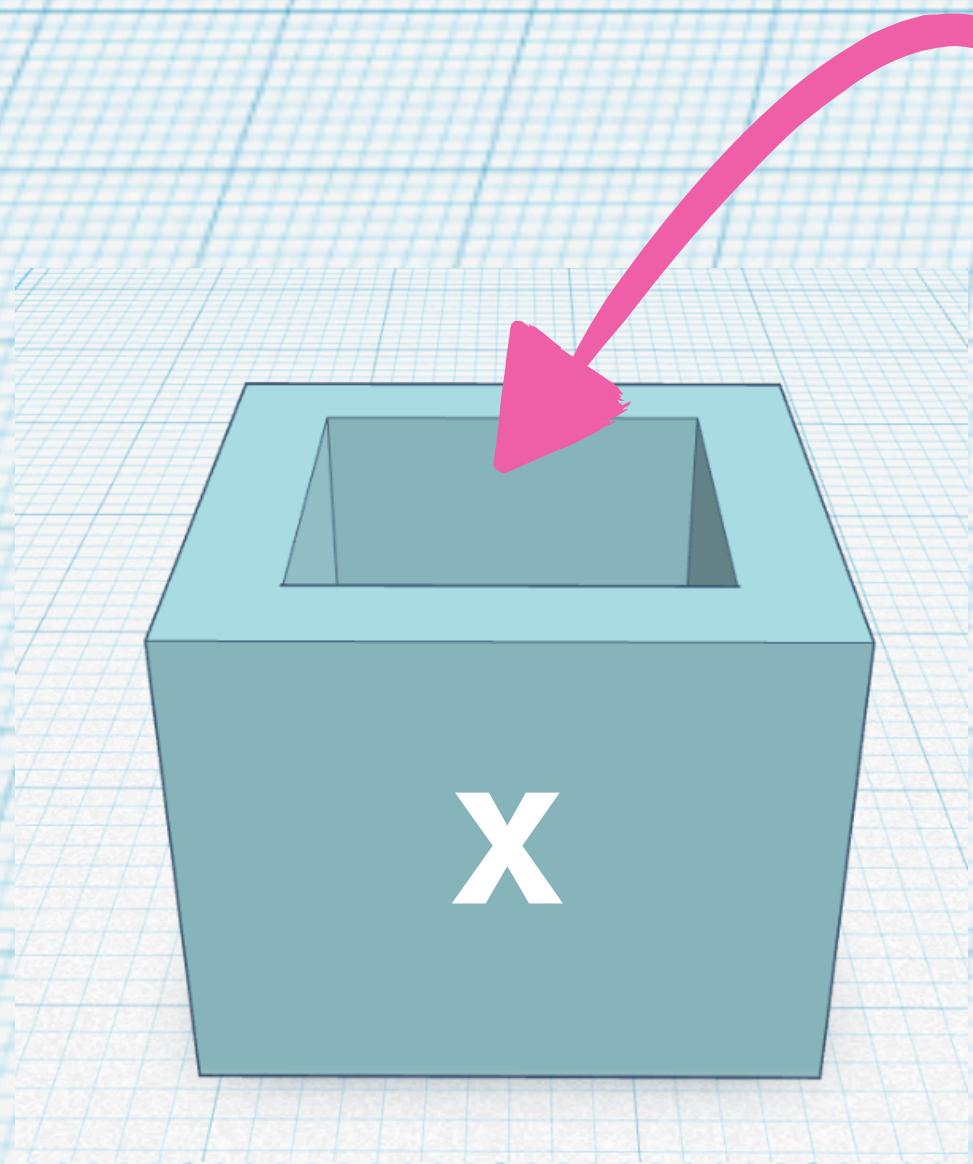


字母 | 十進位 | 二進位
a | 097 | 01100001

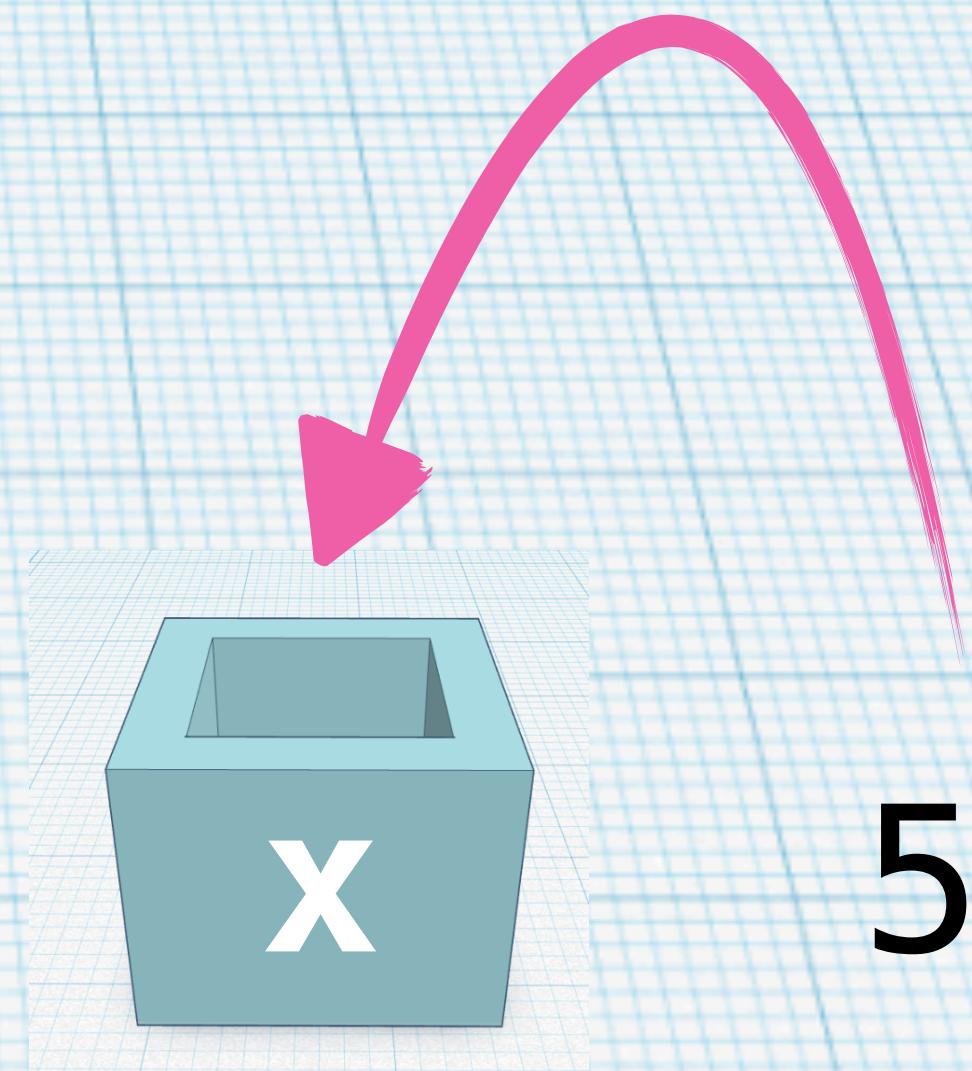


```
int x;  
x = 3;
```

```
char x;  
x = '3';
```

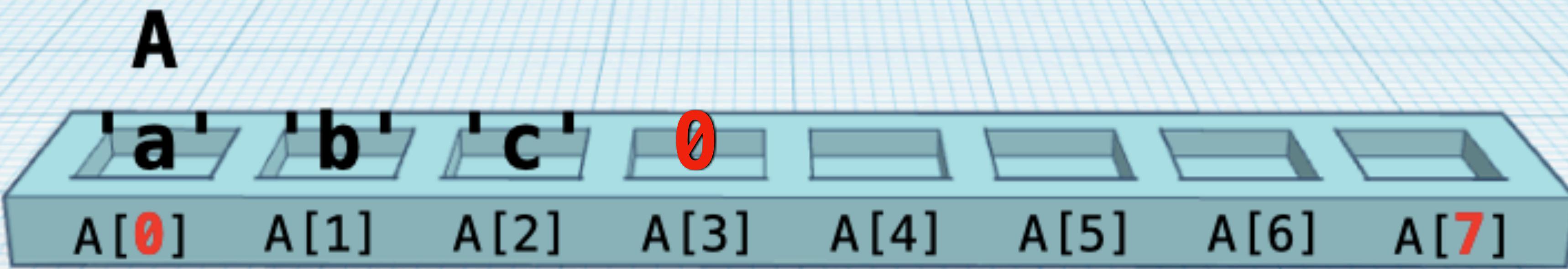


3



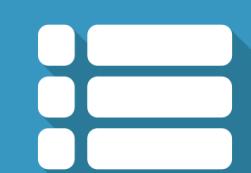
51

十進位	二進位	字元
49	00110001	1
50	00110010	2
51	00110011	3
52	00110100	4



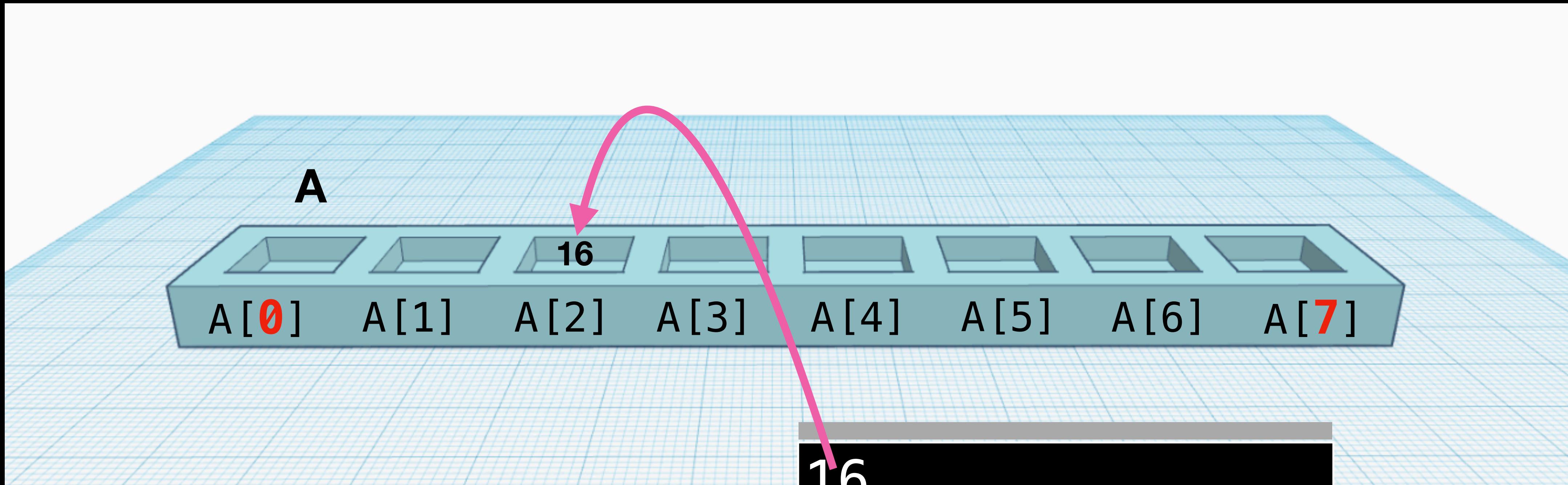
```
char A [8];  
cin >> A;
```

abc

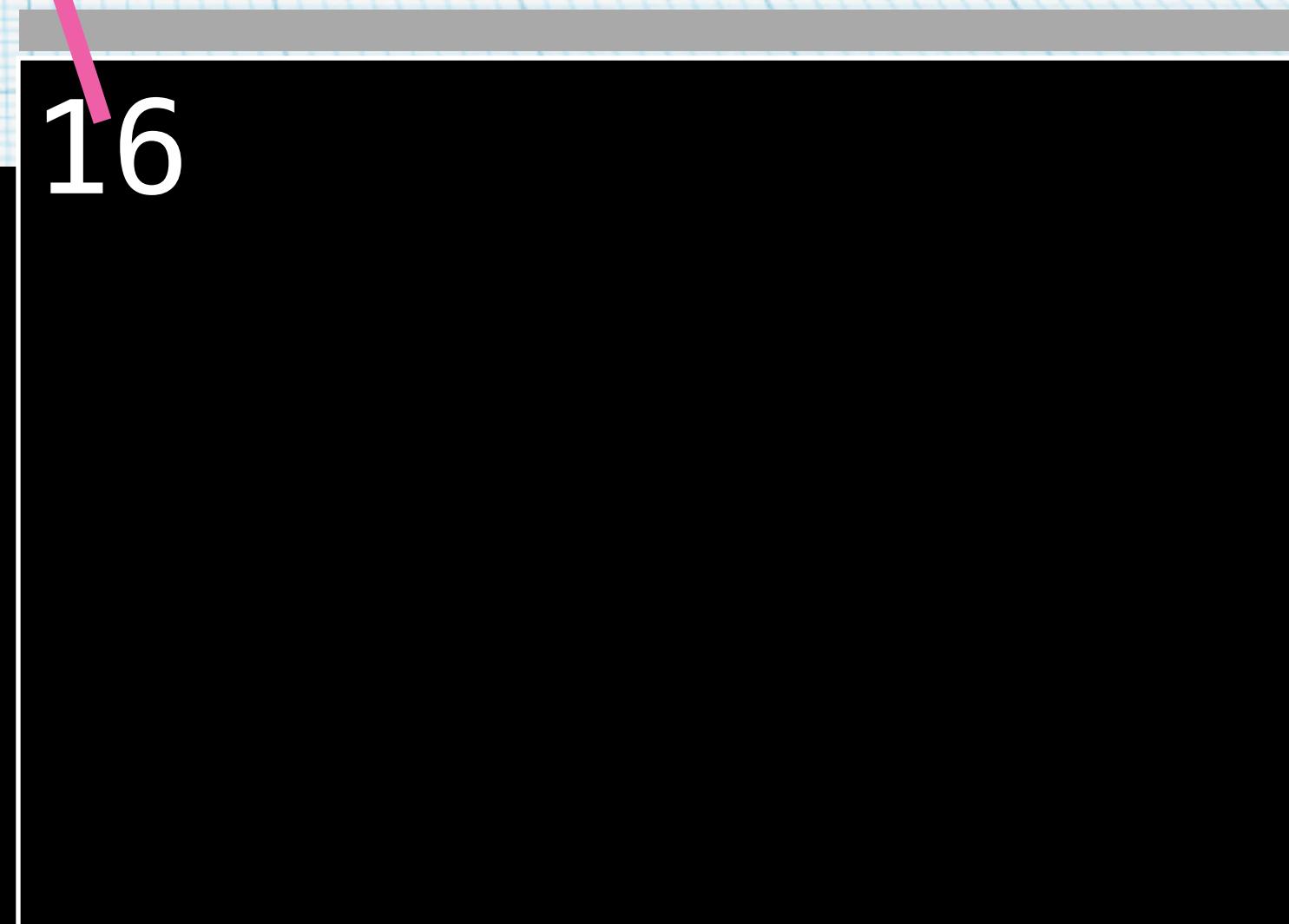


字元陣列

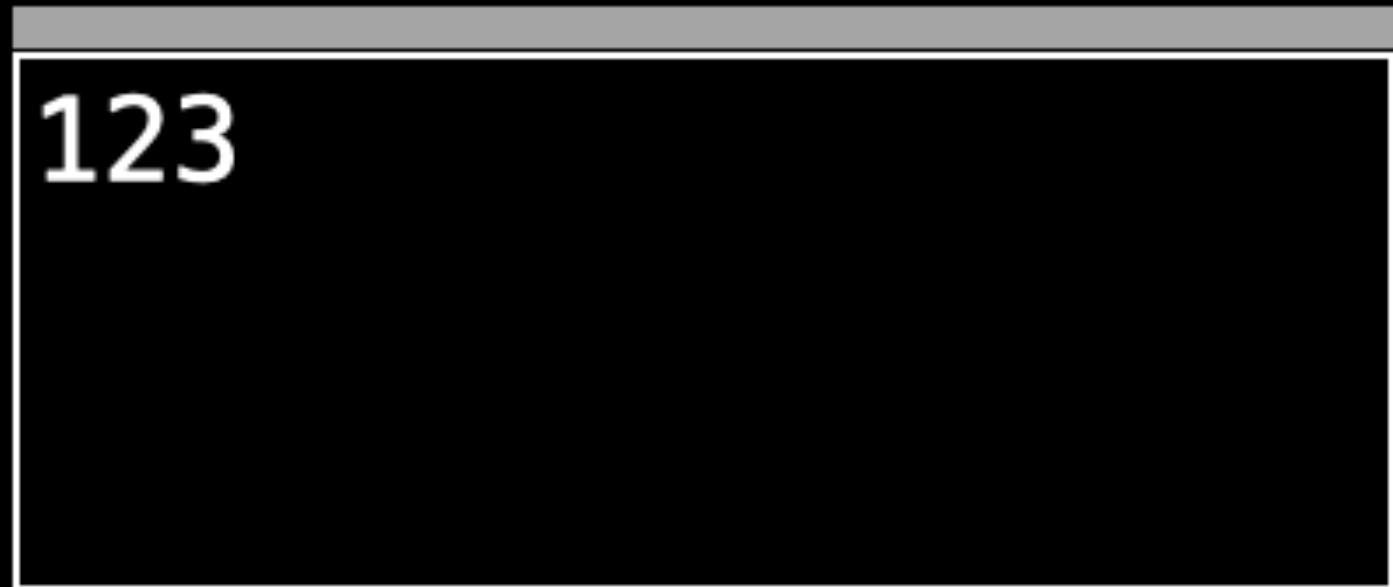
Elton Huang (C) 2020



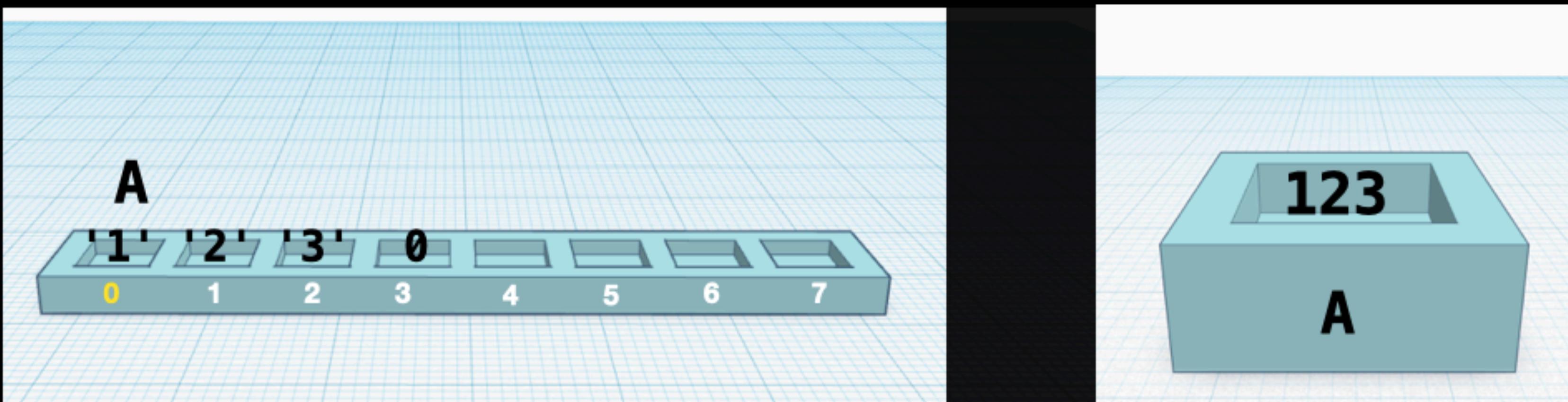
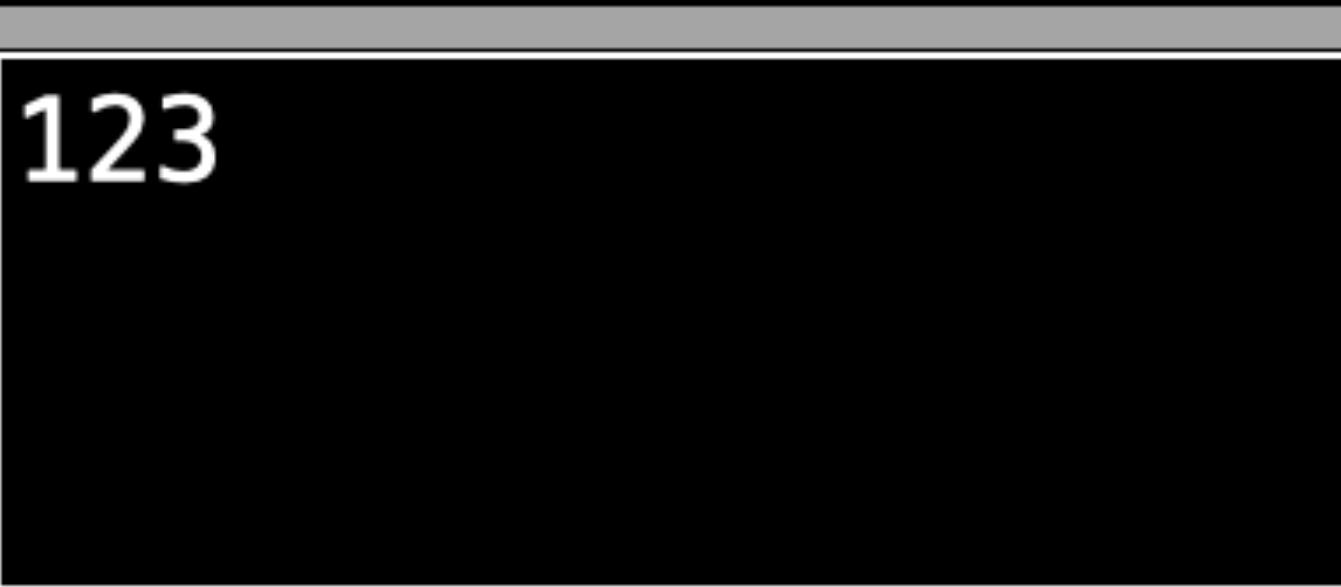
```
int A [8];  
cin >> A [2];
```



```
char A [8];  
cin >> A;
```



```
int A;  
cin >> A;
```





類型	正確	錯誤	說明	警告或錯誤訊息
運算符號	<code>if (a == b) ...</code>	<code>if (a = b) ...</code>	(或迴圈判斷式處) 兩個等號 是比較，一個是指定 注意：只有警告	warning: suggested parentheses around assignment used as truth value 警告：建議將用作真假值的指定加上括弧
輸入串流運算符號	<code>int m, n; cin >> m >> n;</code>	<code>int m, n; cin >> m, n;</code>	輸入多個變數用 <code>>></code> 分開 注意：只有警告	warning: right operand of comma operator has no effect 警告：逗號運算子的右運算元無作用
輸出串流運算符號	<code>cout << "x:" << x << endl;</code>	<code>cout << "x:" x << endl;</code>	輸出項目用 <code><<</code> 分開	error: expected ";" before '...' 錯誤：在 '...' 前期望有 ";"
輸出入串流運算符號	<code>cin >> x; cout << x;</code>	<code>cin << x; cout >> x;</code>	想像訊息流動的方向	error: no match for operator ... 錯誤：無法匹配運算子 ...
忘記換行	<code>cout << x << endl;</code>	<code>cout << x;</code>	如果題目要求換行，要輸出換行	(無) 注意：沒有任何訊息
忘記分號	<code>cin >> x;</code>	<code>cin >> x</code>	每個指令，說完一句話後要分號。 <code>if (...)</code> 只說了半句	error: expected ";" before '...' 錯誤：在 '...' 前期望有 ";"
忘記大括弧	<code>for (auto i: a) { cout << i << " "; cout << endl; }</code>	<code>for (auto i: a) cout << i << " "; cout << endl; x</code>	超過一個指令，要做的事不只一件，要用大括弧包成一個區塊	(無) 注意：沒有任何訊息
懸掛的 else	<code>if (...) { if (...) ... } else ...</code>	<code>if (...) if (...) ... else ...</code>	<code>else</code> 會找最接近的 <code>if</code> 配對。 根據算法邏輯加上大括號或是每個 <code>if</code> 都配上 <code>else</code> 注意：只有警告	warning: suggest explicit braces to avoid ambiguous 'else' 警告：建議加上括弧避免模稜兩可的 <code>else</code>
使用保留字作變數名稱	<code>int dot;</code>	<code>int do;</code>	C/C++ 語言保留字不得作為變數或函式名稱	error: expected unqualified-id before '...' 錯誤：在 '...' 前期望有 unqualified-id

- while/for 迴圈語法中的條件是迴圈持續的條件，不是中止的條件 (比較迴圈中可以有的 if/break；和自然語言的習慣一致)。
- while/for 迴圈中，while/for 那列陳述包含迴圈每次迭代所要檢查的條件 (以及在 for 迴圈時每次條件的改變)，
所以條件的初始設定在迴圈開始前，不是在迴圈內。
- for 迴圈注意迴圈的遞增的情況較遞減的常用一些，特別注意遞減時的情況。
- 輸出結果如果出現亂七八糟奇怪的數字，或是 Code::Blocks 開始卡住或不正常工作，可能的原因之一是你的程式碼陣列宣告得不夠大，以至於執行時錯誤地使用了沒有分配給你的記憶體區域。適時使用 build clean。
- 到目前為止，實作題最容易的 p1，其次 p2，p3、p4 比較難；p4 至少有兩次沒有比 p3 難，只是敘述很長，試圖把複雜的觀念講清楚，但程式要怎麼寫題目也說明了，所以只要轉換成程式碼就可以了，不用特別去思考設計運算法。觀念題最後幾題也傾向放較難或是敘述較複雜的題組，但每題的配分都一樣，所以挑容易、簡單、閱讀理解和處理所需的時間較少的題目先做。
- 觀念題和實作思考時變數的地方用常數代入可以比較快想清楚。
- 注意考試時要根據題號命名 p1.cpp、p2.cpp、p3.cpp、p4持續條件，變數.cpp，並在提交後複製一份程式碼到 backup 檔案夾中。



- 輸出列尾該換行就要換行，特別注意最後一列。

處理輸入連續測資。注意題目允不許輸出列尾的空格，可以在答題系統驗證看看。

沒有解題系統驗證大測資，所以解題時就儘量做好效能的考量

包括加入

```
ios::sync_with_stdio (false); cin.tie (0); cout.tie (0);
```

但第一題通常不會太複雜（第一題先求有再求好，測資有可能不會有太嚴厲的要求，所以先做出來，有時間再去想說怎麼樣可以更好、更快；

但是注意滿分測資的要求，盡量從滿足滿分的測資想解，例如 <秘密差> 那一題。）

- 大陣列宣告成全域變數。

- 考試可以上廁所，跟監考老師說，他會帶你過去，再帶你回來

- Code::Blocks 管理視窗不見了: View → Manager, Logs, Toolbars → ...

Code::Blocks 編輯視窗內按滑鼠右鍵 → Split view 可以分割編輯視窗，程式長長後要參照前後很好用。

- Code::Blocks 用 int a [200] = { 0 }; 之後寫入超過 a [10] 之後的元素在 debugger 中都會看到寫到 a [1] (但實際上可能正常)。

- Code::Blocks

Settings->Debugger->Default->Debugger initialization commands

set max-value-size unlimited

- 學過 Python 的同學注意在 Python 不需要大括弧、小括弧的地方，C/C++ 需要。

- GreenJudge 版本的 C++ 不接受可變大小的陣列宣告，也沒有萬能標頭 bits/stdc++.h 可用。



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

int main () {
    ios::sync_with_stdio (false); // C++ 的原生 iostream 很吃效能，加上這三行能避免這個問題
    cin.tie (0);
    cout.tie (0);
    . . .

    int m = INT_MAX, n = INT_MIN; // 最大整數 (2,147,483,647) 與最小整數 (-2,147,483,648)
    . . .

    y = abs (x); // y 被指定為 x 的絕對值
    . . .

    int x, y, m, n;
    . . .
    m = max (x, y); // m 被指定為 x 和 y 之中較大的值
    n = min (x, y); // m 被指定為 x 和 y 之中較小的值
    . . .

    int n;
    . . .
    cin >> n;
    int a [n];
    int mn = INT_MAX;
    for (int i = 0; i < n; i++)
        mn = min (a[i], mn);
    // 至此，mn 會是 a [n] 中的最小值
    . . .

    // if (mn > a[i])
    //     mn = a[i];
```

如果用 if 寫

時間複雜度 $O(n)$ 但排序要 $O(n \times \log(n))$



```
int x, y;
x = 2;
y = 3;
swap (x, y); // x 和 y 的值交換 // int z = x; x = y; y = z;
cout << x << " " << y << endl; // 會輸出: 3 2
-----
int n;
. . .
cin >> n;
int a[n];
. . .
sort (a      , a + n      ); // a [0] ~ a [n - 1] 由小到大排序
// 或者
sort (a + i, a + j + 1); // a [i] ~ a [j] 由小到大排序
. . .
-----
double x;
int m;

m = round (x); // 指定 m 為最接近 x 的整數
m = floor (x); // 指定 m 為不大於 x 的整數
m = ceil (x); // 指定 m 為不小於 x 的整數
. . .
}
```

```
int main () {
    int a [] = { 2, 4, 3, 1 };
    sort (a, a + 4);
    // 用迴圈逐一輸出 a []
    . . .
    return 0;
}
```

```
int main () {
    char a [] = "hello";
    sort (a, a + 5);
    cout << a << endl;
    return 0;
}
```



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

int main () {
    char A [] = "1234567";      // char A [] = { '1', '2', '3', '4', '5', '6', '7', 0 };
    cout << strlen (A) << endl; // 回傳 A 內字元串之長度

    if (isdigit (A [2]))        // 若 A [2] 為數字，回傳真
        cout << "A [2] is a digit." << endl;

    if (isalpha (A [2]))        // 若 A [2] 為英文字母，回傳真
        cout << "A [2] is an alphabet." << endl;

    int x = atoi (A);           // 將字元陣列 A 裡的數字順序視為一個十進位數整數，回傳對應的整數
    cout << x << endl;

    return 0;
}
```

若萬能標頭的 `bits/stdc++.h` 找不到：

```
#include <cstring> ... strlen()
#include <climits> ... INT_MAX, INT_MIN
#include <cstdlib> ... abs()
#include <algorithm> ... swap(), max(), min(), sort()
```



● 常用數值與技巧¹

- 1 累加到 $10 = 55$
- 2 的次方倍： $1, 2, 4, 8, 16 (2^4), 32, 64, 128, 256 (2^8), 512, 1024 (2^{10}), 2048, 4096$
- 最大 int: $(2^{31} - 1) = 2,147,483,647 \div 2 \times 10^9$: 10 進位時為 10 位數 / unsigned 數值多一倍, 位數仍為 10 位
最大 long long: $9,223,372,036,854,775,807 \div 9 \times 10^{18}$: 10 進位時為 19 位數 / unsigned 數值多一倍, 位數增加一位為 20 位
- 二元搜尋時間複雜度: $\log_2 n$
- 判斷奇數: $(n \% 2) == 1$ 或 $(n \& 1) == 1$
- 個位數數字: $n \% 10$

- 邏輯運算真(假)值表 truth table:

真		假
!	假	真

&&		真	假
真	真	假	
假	假	假	

		真	假
真	真	真	
假	真	假	

- 笛摩根定理: $!(p \&& q) == (!p) || (!q)$
- 運算優先順序不確定時, 保守一點多加一些括弧
- 在 xterm d 上: 滑鼠中鍵(滾輪往下按), 在其他視窗複製貼上: Ctrl-C, Ctrl-V / 滑鼠右鍵 / 視窗命令列「編輯」下拉
- 先用紙筆操作範例可以幫助並確定對問題的正確了解, 並且從中構思解法
- 實作數字的問題有時必須 (106-3-p1 第 3 子題組) 用字串處理會比較容易
- 費氏數列、輾轉相除常考
- 用 Leafpad 整理測資方便每次測試的複製貼上。
- Lubuntu Linux 計算工具: LXTerminal → bc -l。活用編譯指令 #define/#if/#else/#endif

¹ // 寫程式一定要寫註解。幫助日後看懂程式外, 隨著註解複習自己的程式加深所使用編程技巧的印象

• 好程式標準: 最重要不要錯, 盡量不要有 bug, 易讀好改(容易維護), 其次在記憶體不爆掉的情況下跑得快, 再其次精簡