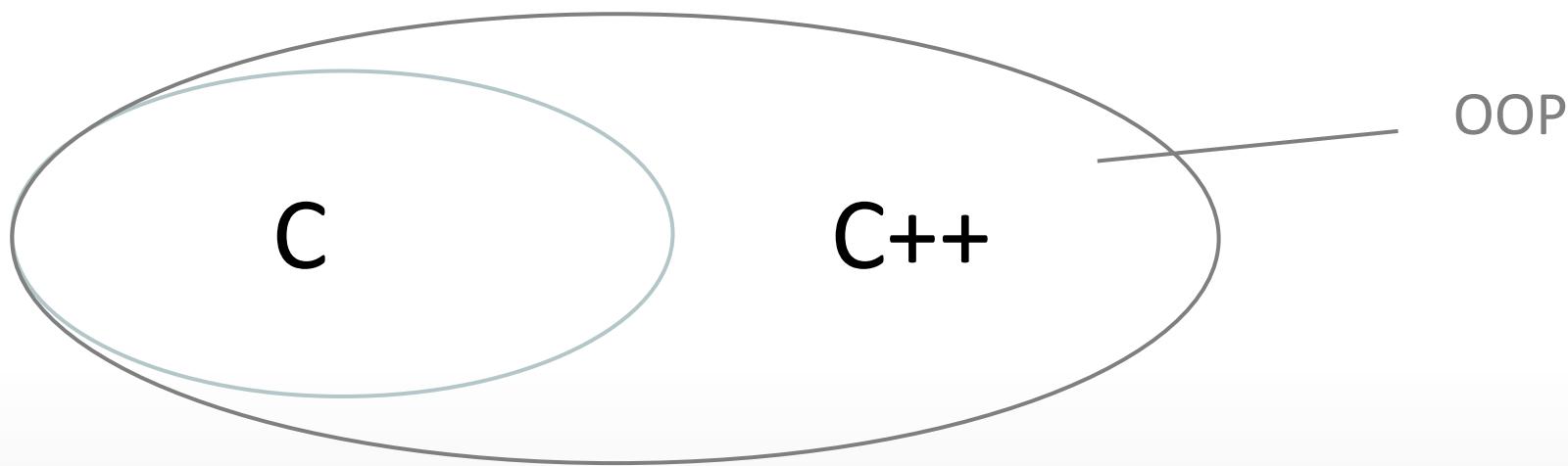


C++

The relationship between C and C++ is like milk tea and milk
tea with tapioca balls



Advantage

- lay the groundwork
- have idea of programming
- Object Oriented Programming (OOP)
- Efficacy

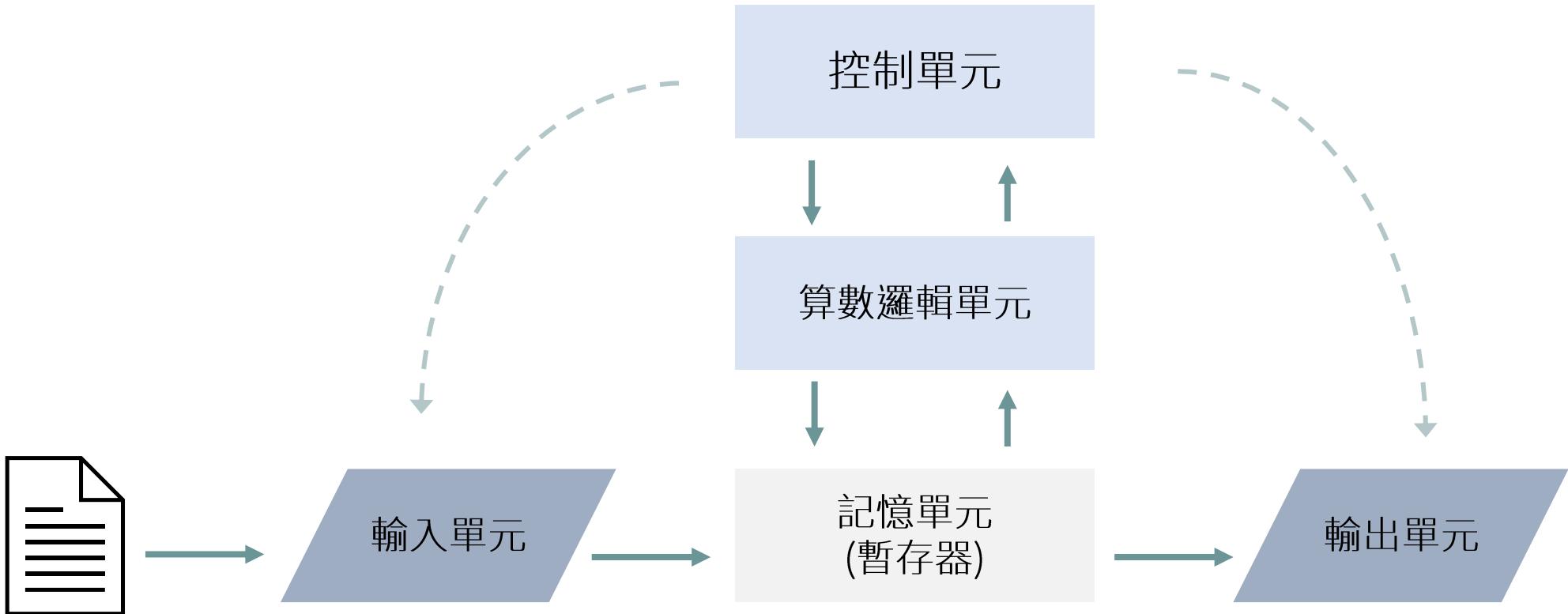
Disadvantage

- harder than others
- Garbage Collection (new \ delete)

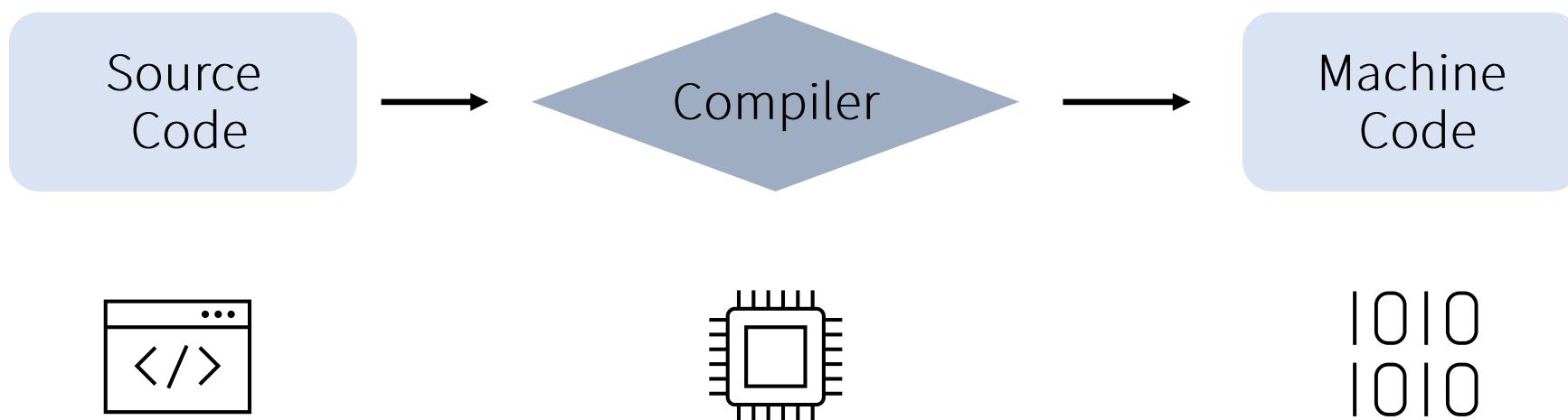
Process

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

Unit



Compiler



Syntax

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

Rule

- 由左至右，由上到下，循序執行
- 「;」表示一個指令的結束
- 執行到return 0 時程式結束(在main裡頭)
- 利用Tab鍵縮排
- {}包起來的內容代表在結構之中

Comment

- Single-line Comments //
 - C++ Multi-line Comments /* */

Base

1. include<iostream> : 引入函式庫
2. using namespace std : 設定命名空間
3. int main(){}
: 任何一個程式的entry point

```
#include<iostream>
using namespace std;

int main(){
    statement...
}
```

I/O

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

I/O Library Header Files

<iostream> : Input Output Stream

- Cout
- Cin
- Cerr

Cout

- 利用「<<」 (stream insertion operator) 來連接需要輸出的內容，然後交由給cout執行，顯示到螢幕上

```
#include<iostream>
using namespace std;

int main(){
    cout<<"Hello, "<<"My name is Shark";
}
```

End Line

- 「endl」 (end line) 插入一個換行符號，使得程式在螢幕上顯示時會執行跳行輸出

```
#include<iostream>
using namespace std;

int main(){

    cout<<"Hello, "<<endl<<"My name is Shark";

}
```

Escape

- 「\」：跳脫字元
- 「\0」：空字元(NULL)
- 「\t」：定位字元(Tab)
- 「\n」：換行字元(ENTER)

Cin

- 利用「>>」 (stream extraction operator) 來連接需要賦值的容器，再交由cin執行，讀取輸入的資料並放入相對應的容器中

```
#include<iostream>
using namespace std;

int main(){

    int number;
    cin>>number;

}
```

Cerr

- 專門被用來顯示錯誤代碼及資訊，使用起來和cout並無很大的不同

```
#include<iostream>
using namespace std;

int main(){
    cerr<<"404 Not Found";
}
```

Variable

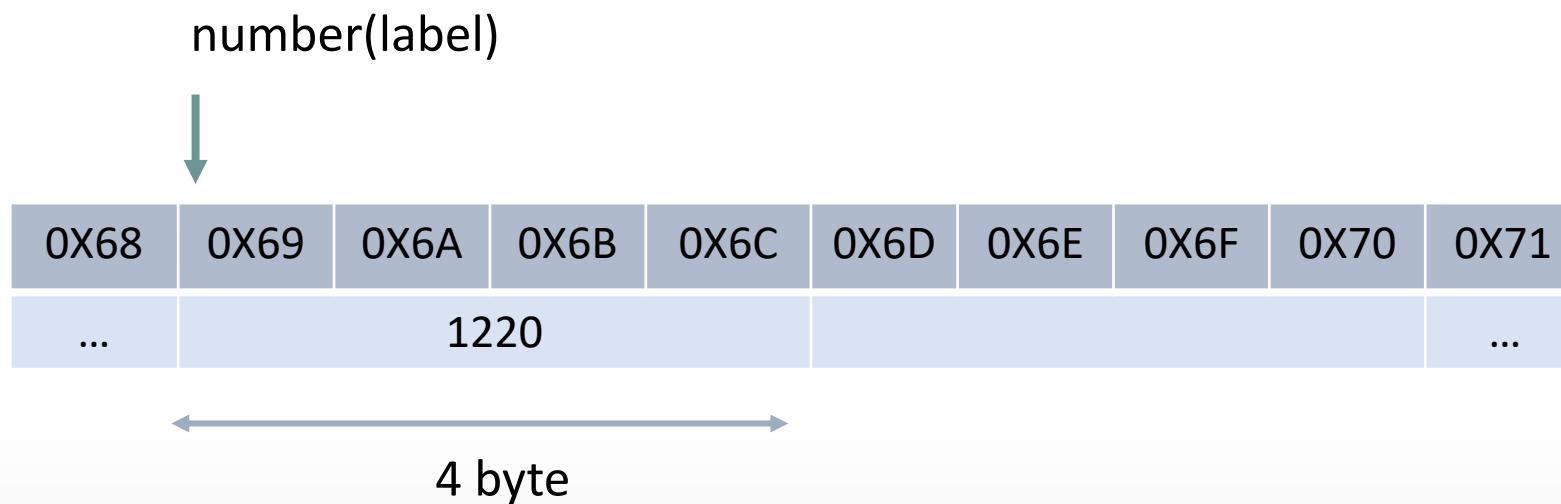
/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

Variable

- Type 型態
- Label 標籤 名字
- Data 資料

Declare

```
int number = 1220 ;
```



Data Type

- Basic Data Type int, char, float, double
- Derived Data Type array, pointer
- Enumeration Data Type enum
- User Defined Data Type structure

Basic Data Type

• int	整數	4byte	-32,768 to 32,767 (2147483647)
• float	單浮點數	4byte	
• double	雙浮點數	8byte	
• char	字元	1byte	-128 to 127
• string	字串		
• bool	布林	1byte	False(0) or True(1)

Identifiers

- h
- height
- sharkheight
- sharkHeight

Challenge

時間轉換

題目說明：將使用者所輸入的秒數轉換成
x小時x分鐘x秒的形式

題目提示：Variable、Operator

輸入：3615

輸出：1hr 0min 15sec

輸入：64

輸出：0hr 1min 4sec

Operator

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

Arithmetic Operators

- + 加法
- - 減法
- * 乘法
- / 商數除法
- % 餘數除法

Assignment Operators

指派運算子會將值儲存在左運算元所指定的容器中。

- = 簡單指派
- += 複合指派

Relational Operators

- `==` 等於
- `!=` 不等於
- `<` 小於
- `<=` 小於等於

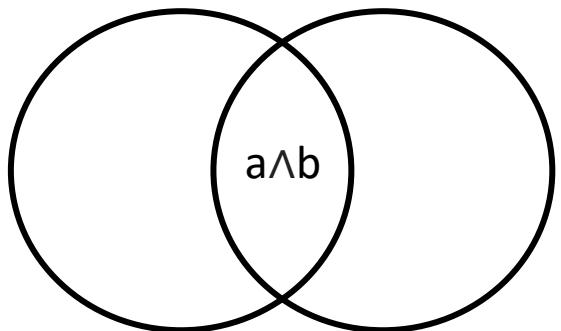
Logical Operators

邏輯運算子會將左右兩邊的運算元作邏輯的判斷

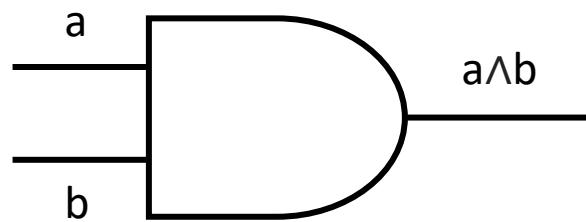
- `&&` and 且
- `||` or 或
- `!` not 相反

Logical Operators

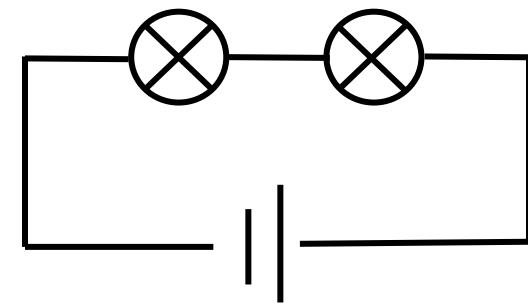
- $\&\&$ and



Venn diagram



logic gate

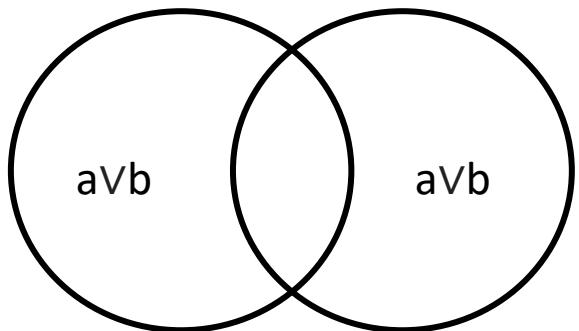


electrical diagram

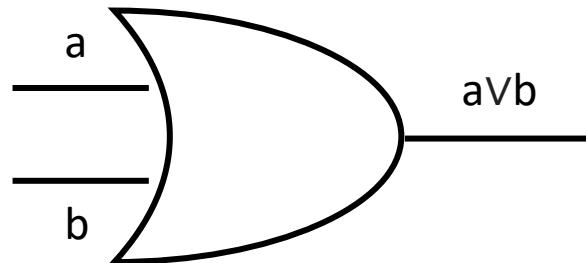
Logical Operators

• ||

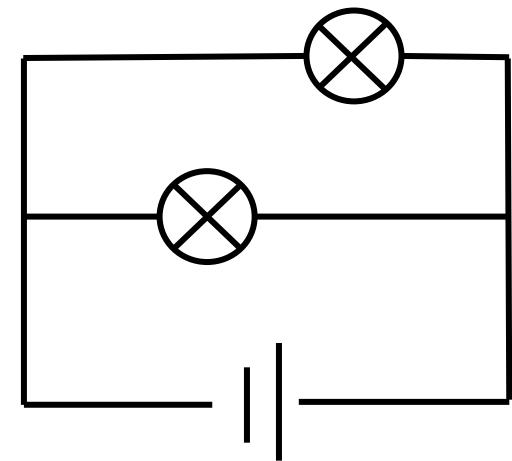
or



Venn diagram



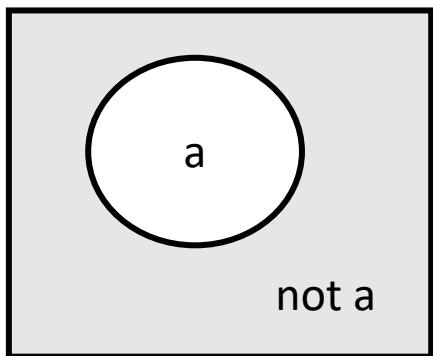
logic gate



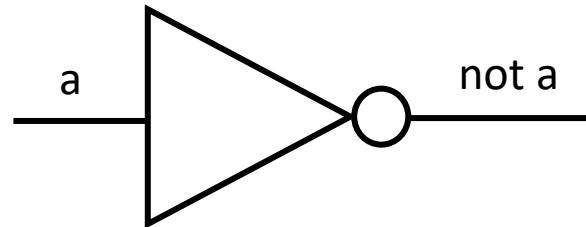
electrical diagram

Logical Operators

- ! not



Venn diagram



logic gate

Unary Operators

- postfix-expression ++
- ++ unary-expression

```
#include<iostream>
using namespace std;

int main(){

    int i=0;
    cout<<++i;      // output: 1
    cout<<i;        // output: 1

    int i=0;
    cout<<i++;     // output: 0
    cout<<i;        // output: 1

}
```

Unary Operators

- postfix-expression ++
- ++ unary-expression

```
#include<iostream>
using namespace std;

int main(){

    int i=0;
    int temp=i++;
    cout<<temp;          // output: 0

    int i=0;
    int temp=++i;
    cout<<temp;          // output: 1

}
```

Encoding

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

Encoding

① a -> A

② IV -> 4

③ Hello -> 哈囉

④ \u4f60\u597d -> 你好

⑤ 豺 -> sharkfoolish.github.io

ASCII

American Standard Code for Information Interchange

Char就是儲存 ASCII 裡的一個整數(0-127)

- ① 控制字元 (‘\t’ , ‘\n’)
- ② 可顯示字元 (‘A’ , ‘a’)

ASCII

```
#include<iostream>
using namespace std;

int main(){

    char(整數)    //將整數轉成ASCII的字元
    int(字元)     //將字元轉成ASCII的編號

}
```

Binary

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

Binary

「世界上只有 10 種人，一種是懂二進位的、一種是不懂二進位的」

- 電線電壓
- 電腦的訊號
- 二進位制
- CPU 的架構

Binary

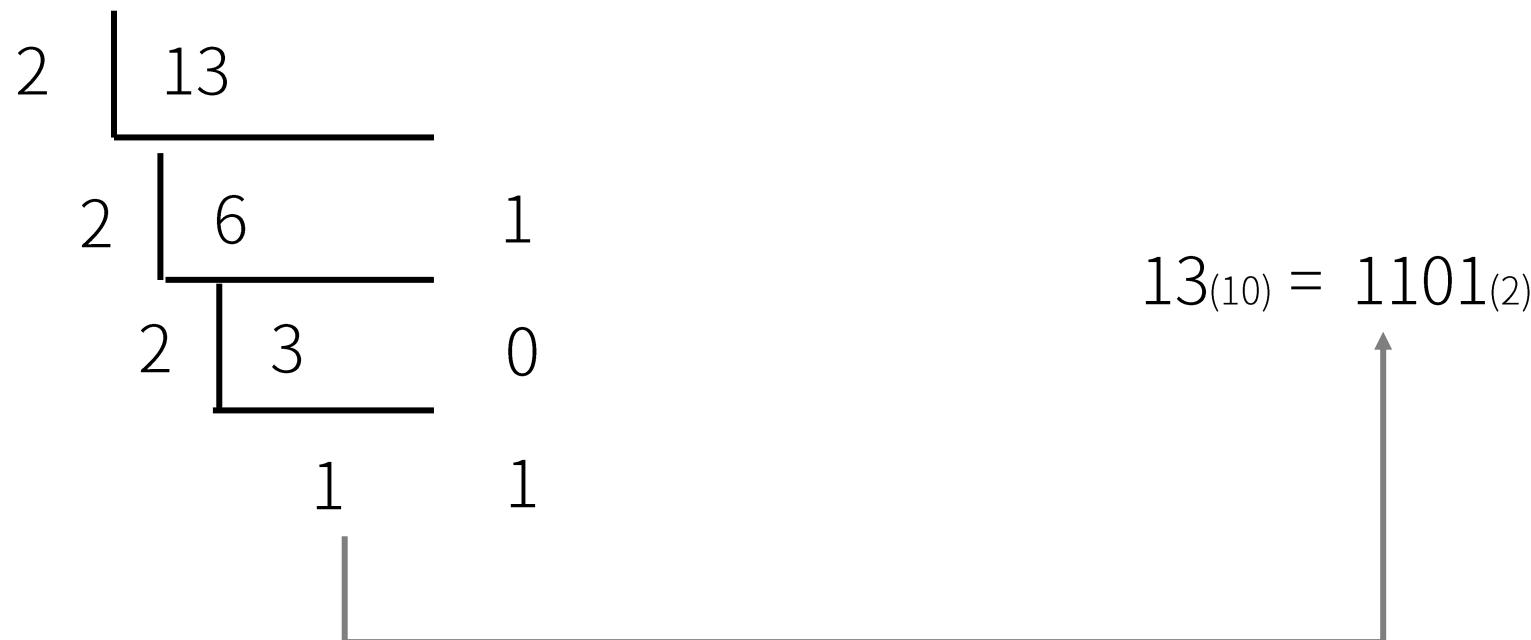
10進位

$$: 426 = 4 * 10^2 + 2 * 10^1 + 6 * 10^0$$

2進位

$$: 1101 = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0$$

Binary



Challenge

小寫轉大寫

題目說明：輸入小寫字母後，轉成大寫字母輸出

題目提示：ASCII、char()、int()

輸入：a

輸出：A

輸入：z

輸出：Z

Challenge

老鼠愛大寫

題目說明：無論你輸入的是大寫的字母還是小寫的字母，最後都還是會顯示大寫字母

題目限制：不能使用if/else

輸入：A

輸出：A

輸入：a

輸出：A

Control Flow

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

Control Flow

- Selection statements 選擇語句
- Iteration statements 反覆語句
- Jump statements 跳轉語句

Selection statements

選擇語句計算單個或多個運算式，並從其結果來確定條件為TRUE或FALSE，決定要執行哪個區塊的程式碼

- If/else
- Switch-Case

Iteration statements

反覆語句重複執行單個或多個運算式，並在每次要重複執行時，去執行條件運算式，除非結果為FALSE或遇到break，否則就會一直運作

- While
- Do-while
- For-loop

Break Statement

break , 休息/中斷

- break 語句會結束執行最接近的封閉迴圈或 switch-case。
- 通常會搭配 if-else 條件使用

Continue Statement

continue , 繼續/持斷

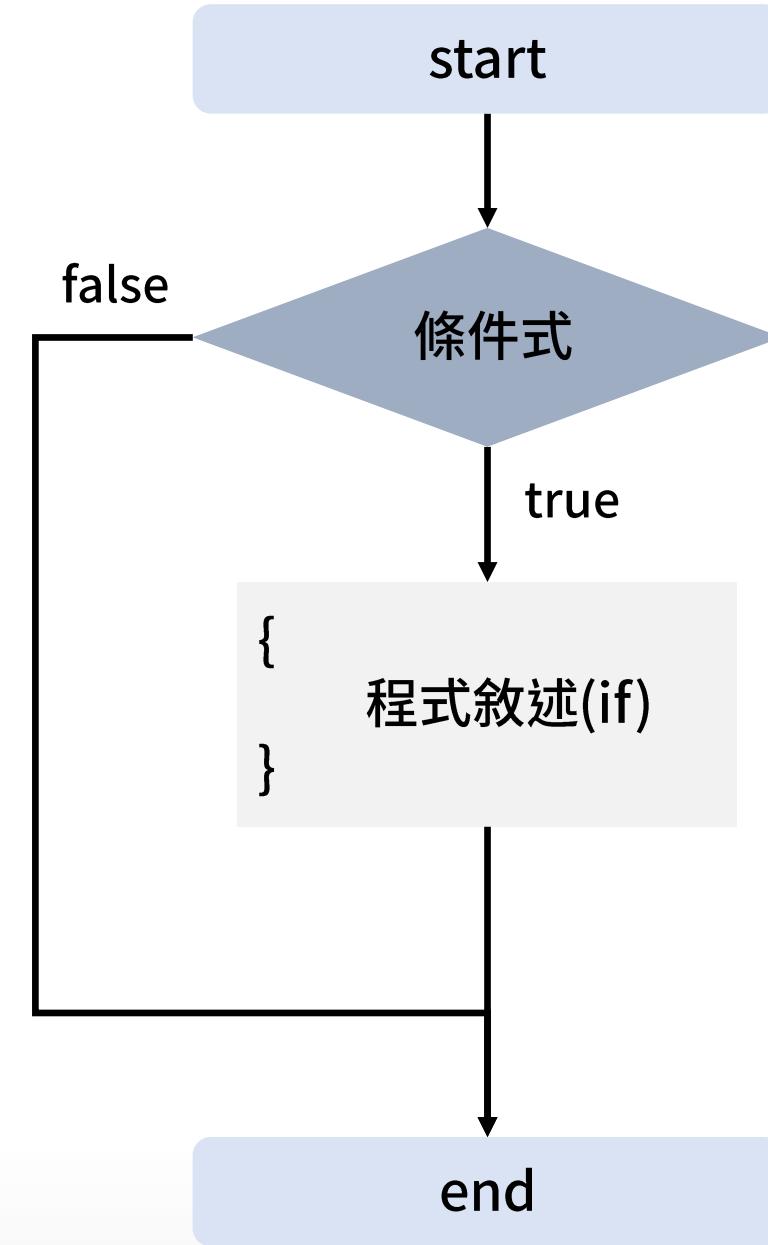
- continue 語句會跳過迴圈當前的反覆運算，並繼續下一輪迴圈
- 在continue底下的程式碼則不會執行

Control Flow: If

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

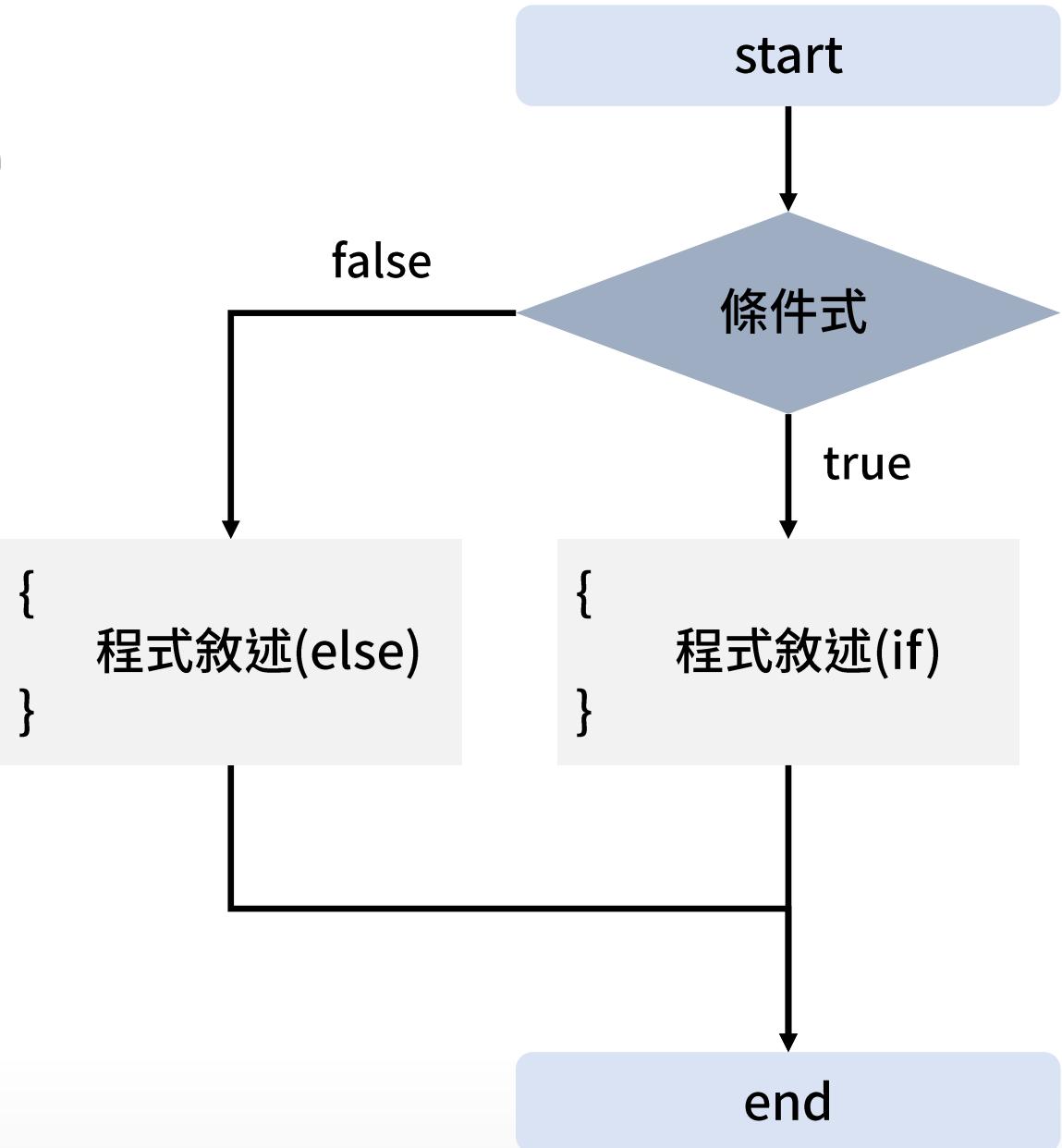
Control Flow: If

```
If ( condition ) {  
    //code to be executed  
}
```



Control Flow: If-else

```
If (condition) {  
    //code if condition is true  
} else {  
    //code if condition is false  
}
```



Thinking

重新認識布林

題目說明：請你找出什麼情況下的時候 $\text{if}(x \% 3)$ 會是 `false`，並試著說明其規則

題目提示：`bool`、`mod`、Selection statements

輸入：7%2

輸出：True

輸入：7%4

輸出：True

Challenge

整除判斷

題目說明：輸入一個正整數，判斷是否能同時被
3和11整除，並輸出 0、1(分別代表錯誤及正確)

題目提示：mod、Relational Operators

輸入：12

輸出：false

輸入：99

輸出：true

Challenge

高斯符號

題目說明：當使用者輸入一個「實數」，
你能夠將其進行高斯運算後，得出答案

題目限制：

1. 不能用ceil()、floor()
2. 只能用一個if/else

輸入：12.20

輸出：12

輸入：-1.5

輸出：-2

Ternary Operator

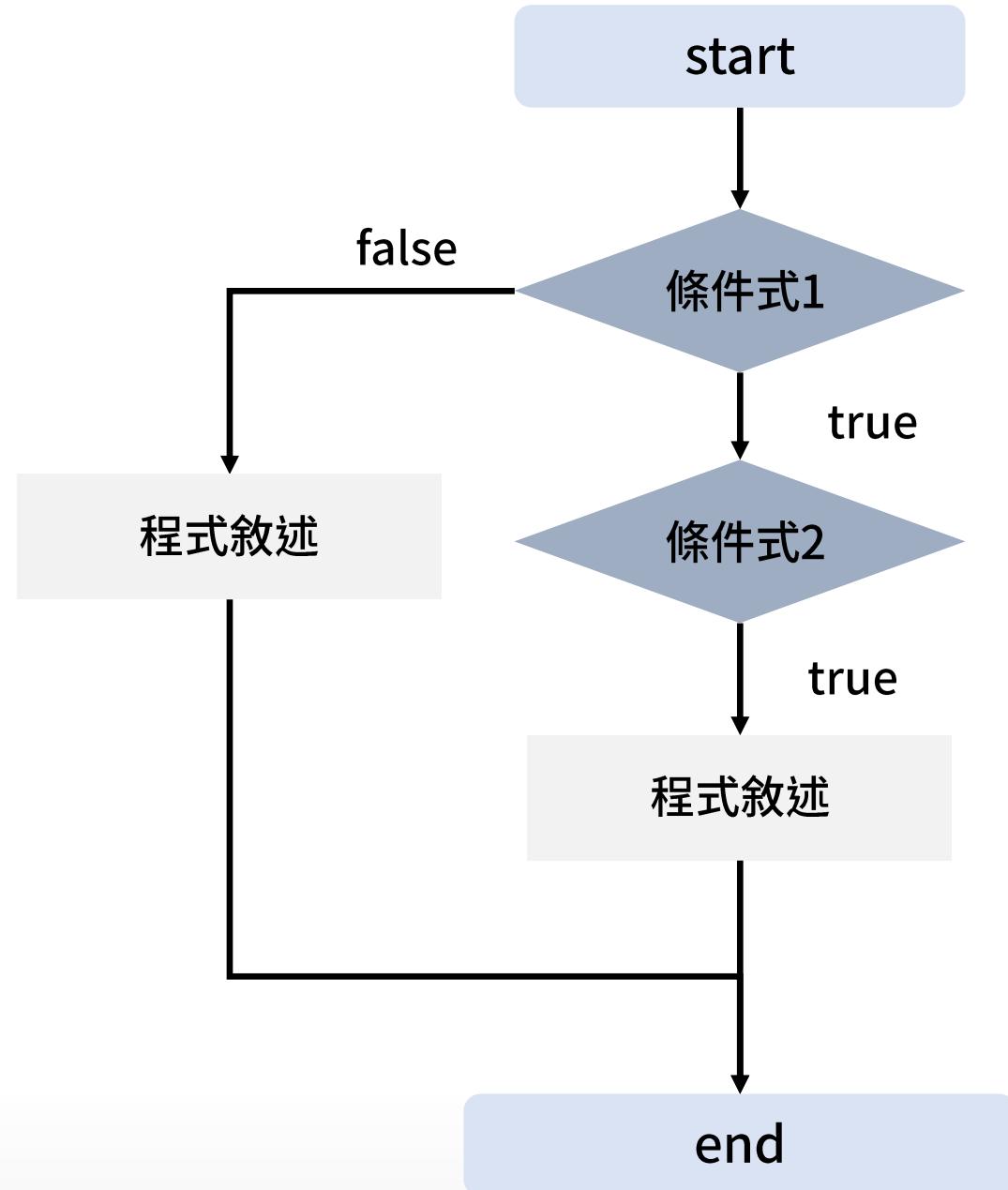
```
variable = condition ? expression1 : expression2 ;
```

- if condition is true, expression1 is executed.
- And, if condition is false, expression2 is executed.

```
score = score >= 100 ? 100 : score ;
```

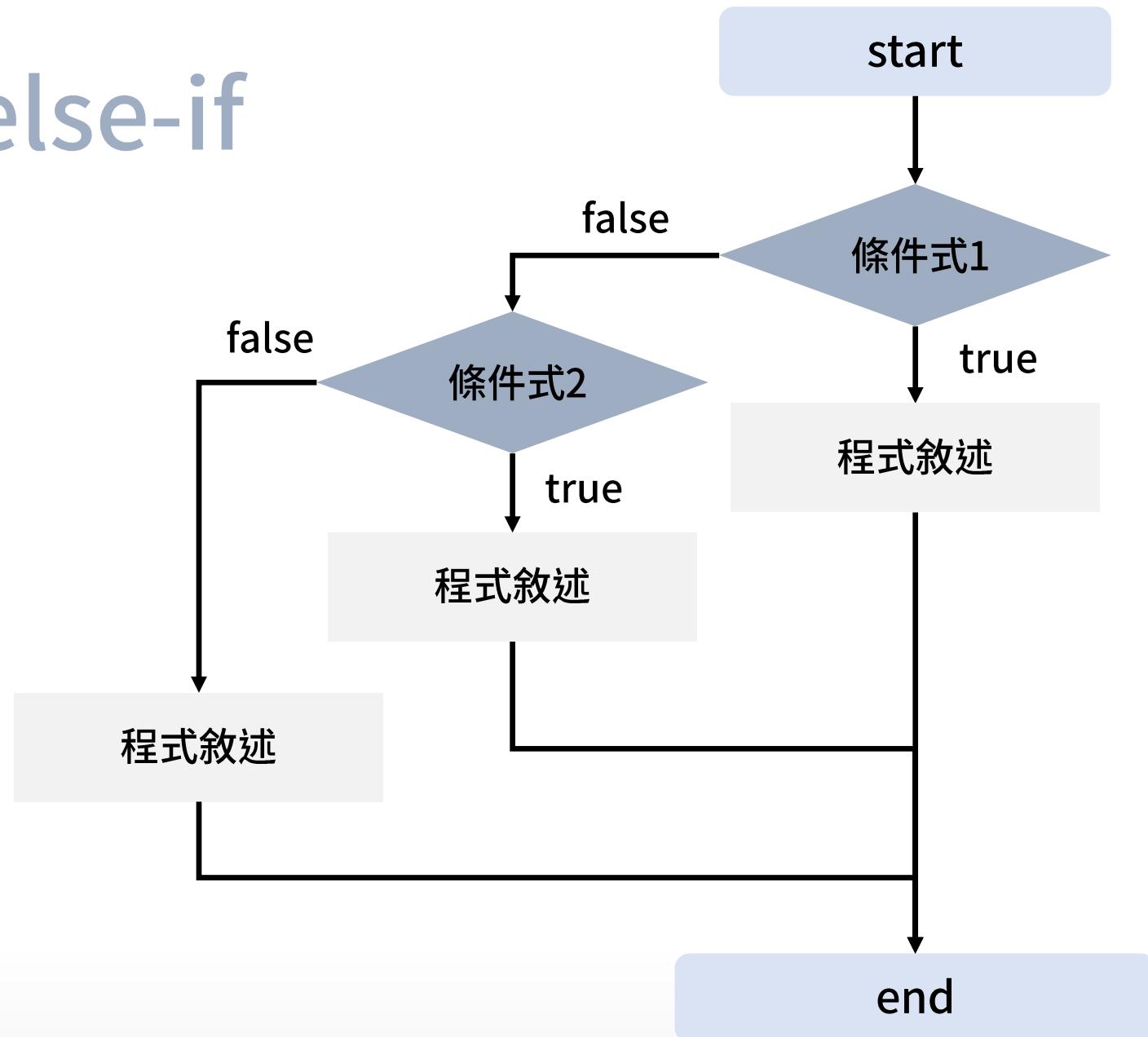
Control Flow: nested-if

```
If (condition1) {  
    If (condition2) {  
        code to be executed if  
        condition1 and if  
        condition2 is true  
    }  
} else {  
    code to be executed if condition1  
are false  
}
```

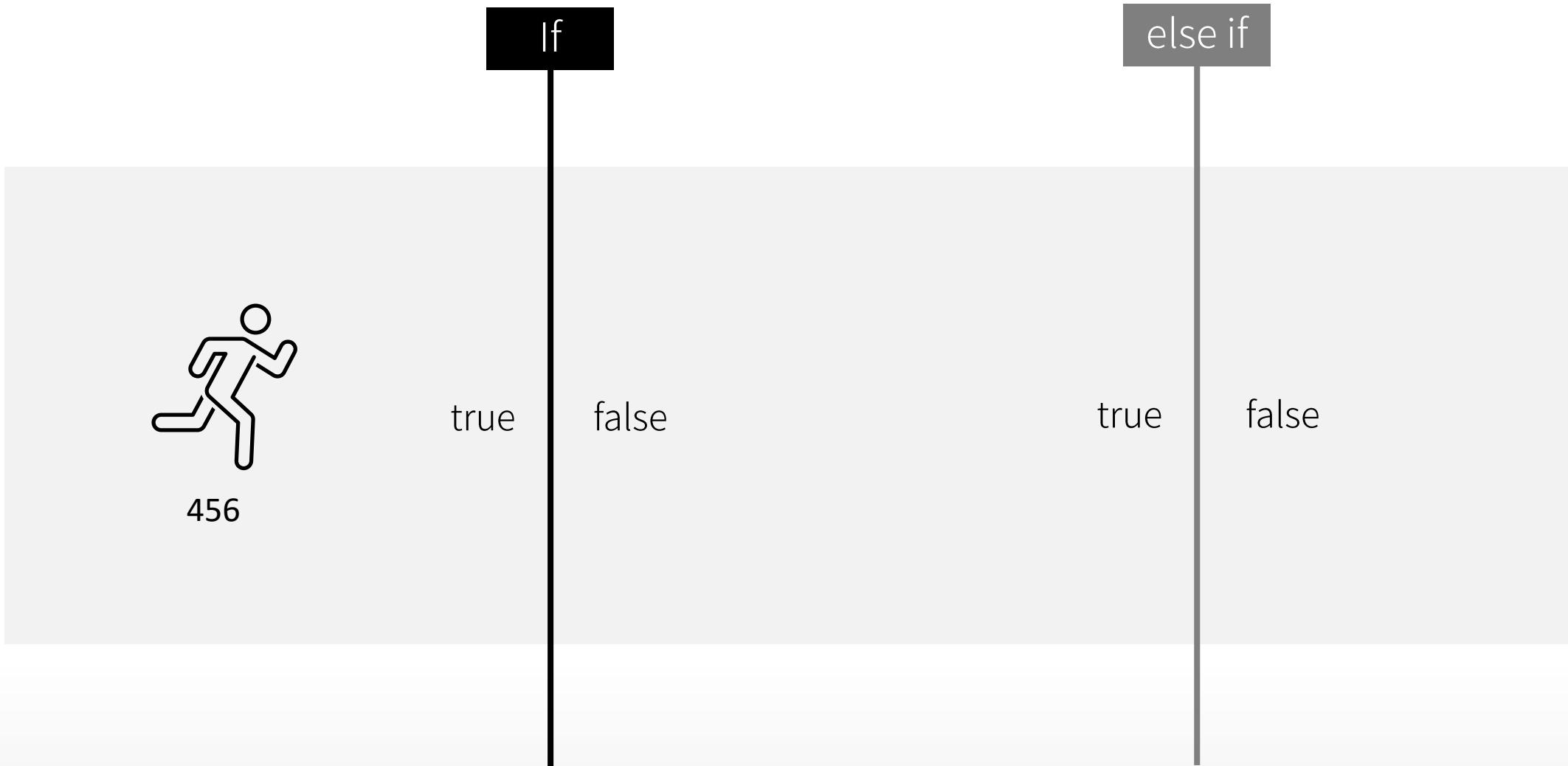


Control Flow: If-else-if

```
If (condition) {  
    code to be executed if  
    condition1 is true  
} else if (condition) {  
    code to be executed if  
    condition2 is true  
} else {  
    code to be executed if all  
    the conditions are false  
}
```



Like Game



Challenge

判斷象限

題目說明：輸入兩數，分別代表 x 跟 y ，判斷此座標是位在哪個象限，並輸出其結果

題目提示：logical operators、if-else-if

輸入：1 2

輸出：First

輸入：-3 2

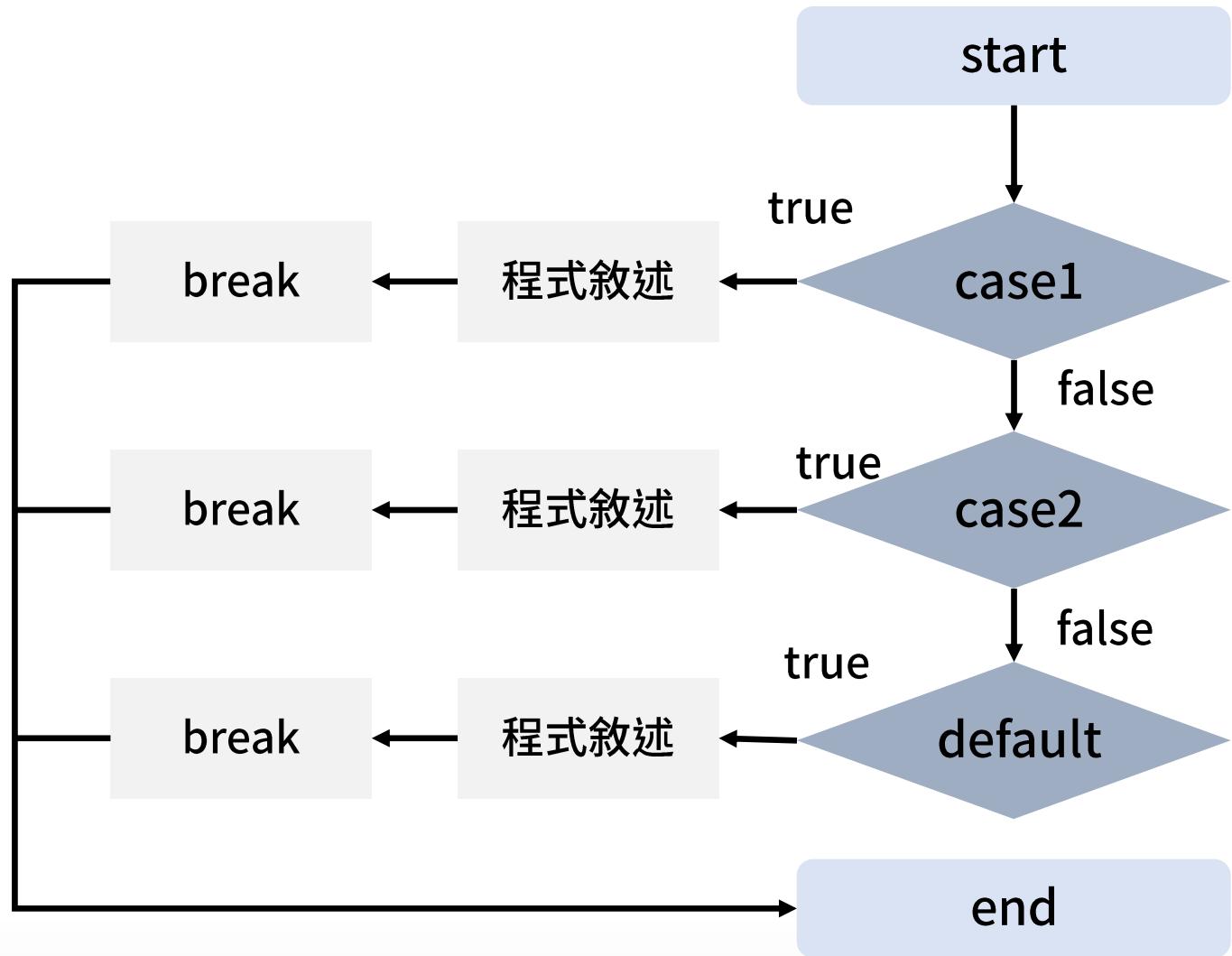
輸出：Second

Control Flow: Switch-Case

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

Control Flow: Switch-case

```
switch (exp) {  
    case value1:  
        code to be executed;  
        break;  
    case value2:  
        code to be executed;  
        break;  
    .....  
    default:  
        code to be executed if  
        all cases are not matched;  
        break;  
}
```



Control Flow: Switch-case

中止條件

- 一直執行到無case為止
- break 強制跳離流程

Thinking

重新認識 switch-case

題目說明：如請問右邊的程式碼會輸出甚麼結果？

題目提示：switch-case、break

```
#include <iostream>
using namespace std;

int main() {
    int number = 2;
    switch (number) {
        case 1:
            cout << "1";
        case 2:
            cout << "2";
        case 3:
            cout << "3";
            break;
    }
}
```

Challenge

成績等第

題目說明：輸入一數代表成績，判斷此成績的等第是多少，並輸出其結果
(90~100 : A ; 80~89 : B ; 70~79 : C ; 0~69 : F)

題目提示：switch-case、Arithmetic Operator

輸入：90

輸出：A

輸入：85

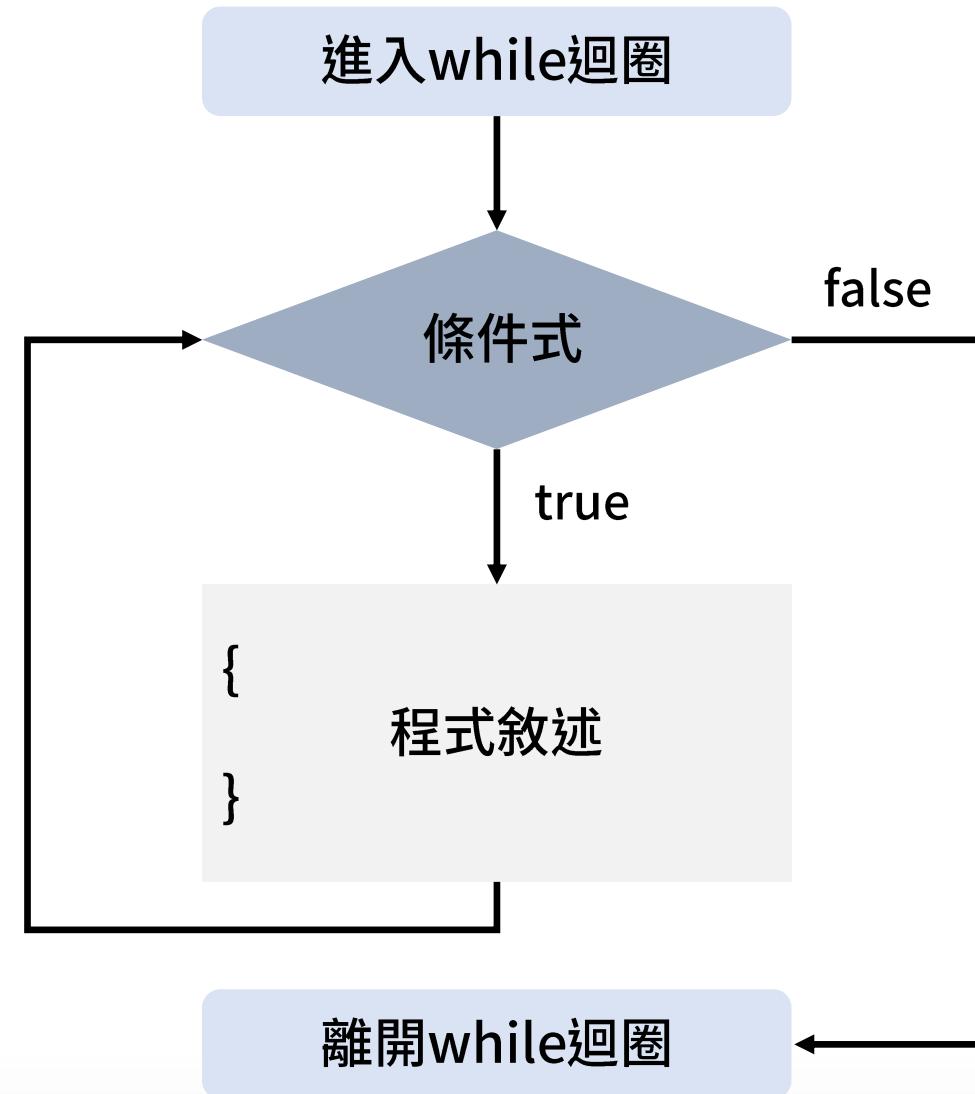
輸出：B

Control Flow: While

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

Control Flow: While

```
while (condition) {  
    code to be executed if  
    condition is true  
}
```

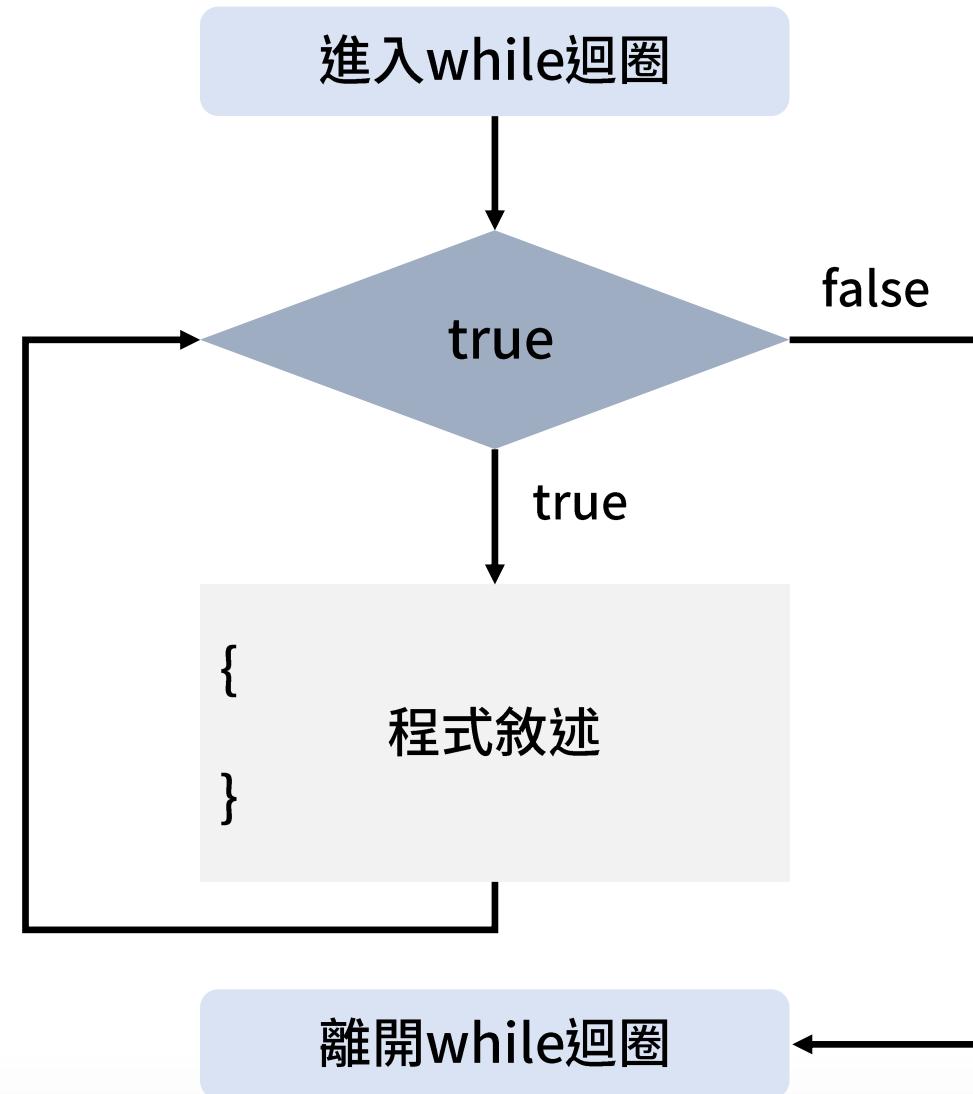


Control Flow: Infinite loop

```
while (true) {
```

code to be executed if
condition is true

```
}
```



Challenge

3n+1猜想

題目說明：輸入一個數字，如果它是奇數，則對它乘3再加1，如果它是偶數，則對它除以2，如此循環。請利用程式把運算的過程顯示出來最終得出答案1

題目提示：while、if-else

輸入：40

輸出：

40
20
10
5
16
8
4
2
1

Challenge

找出因數

題目說明：輸入一個正整數n，找出其所有因數

題目提示：n的所有因數可能為1~n

輸入：30

輸出：

1
2
3
5
6
10
15
30

Challenge

判斷質數

題目說明：輸入一個正整數n，判斷是否為質數

題目提示：如果 n 除了1和自己本身以外，並無其他因數，那就可以稱做為質數

輸入：20

輸出：否

輸入：23

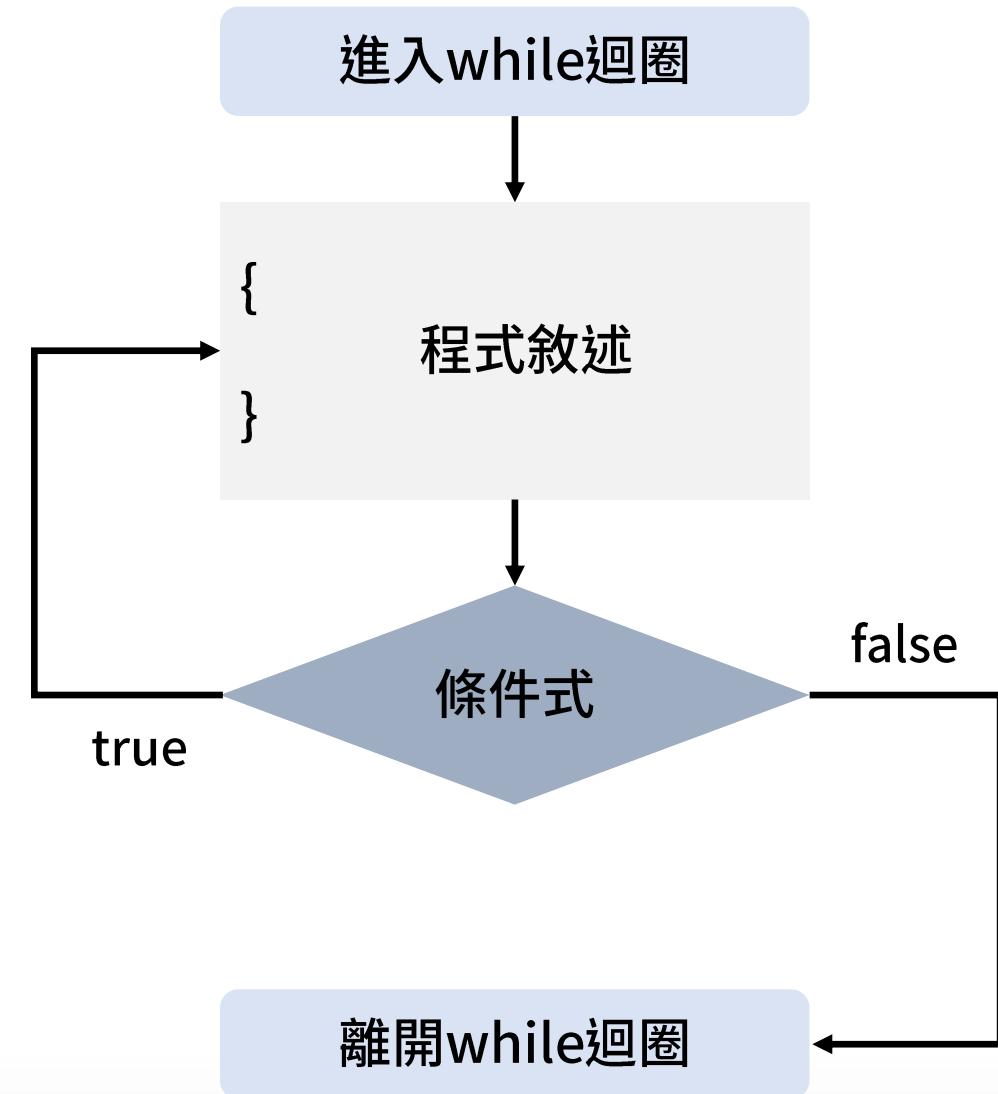
輸出：是

Control Flow: Do-While

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

Control Flow: Do-While

```
do {  
    code to be executed if  
    first to be executed or  
    condition is true  
} while ( condition );
```



Challenge

公車刷卡

題目說明：輸入一個整數表示悠遊卡中的餘額，然後設計一個程式，會持續刷卡扣錢10元，直到沒錢為止，但是如果第一次刷的時候，不夠錢的話，還是可以先賒帳上車

題目提示：do-while

輸入：30

輸出：

悠遊上車，餘額: 20

悠遊上車，餘額: 10

悠遊上車，餘額: 0

輸入：5

輸出：

悠遊上車，餘額: -5

Control Flow: For-loop

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

Control Flow: For-loop

```
for ( init-expression ; cond-expression ; loop-expression ) {  
    code to be executed if cond-expression is true  
}
```

- init-expression

通常用來初始迴圈索引，可能包含運算式或宣告。

- cond-expression

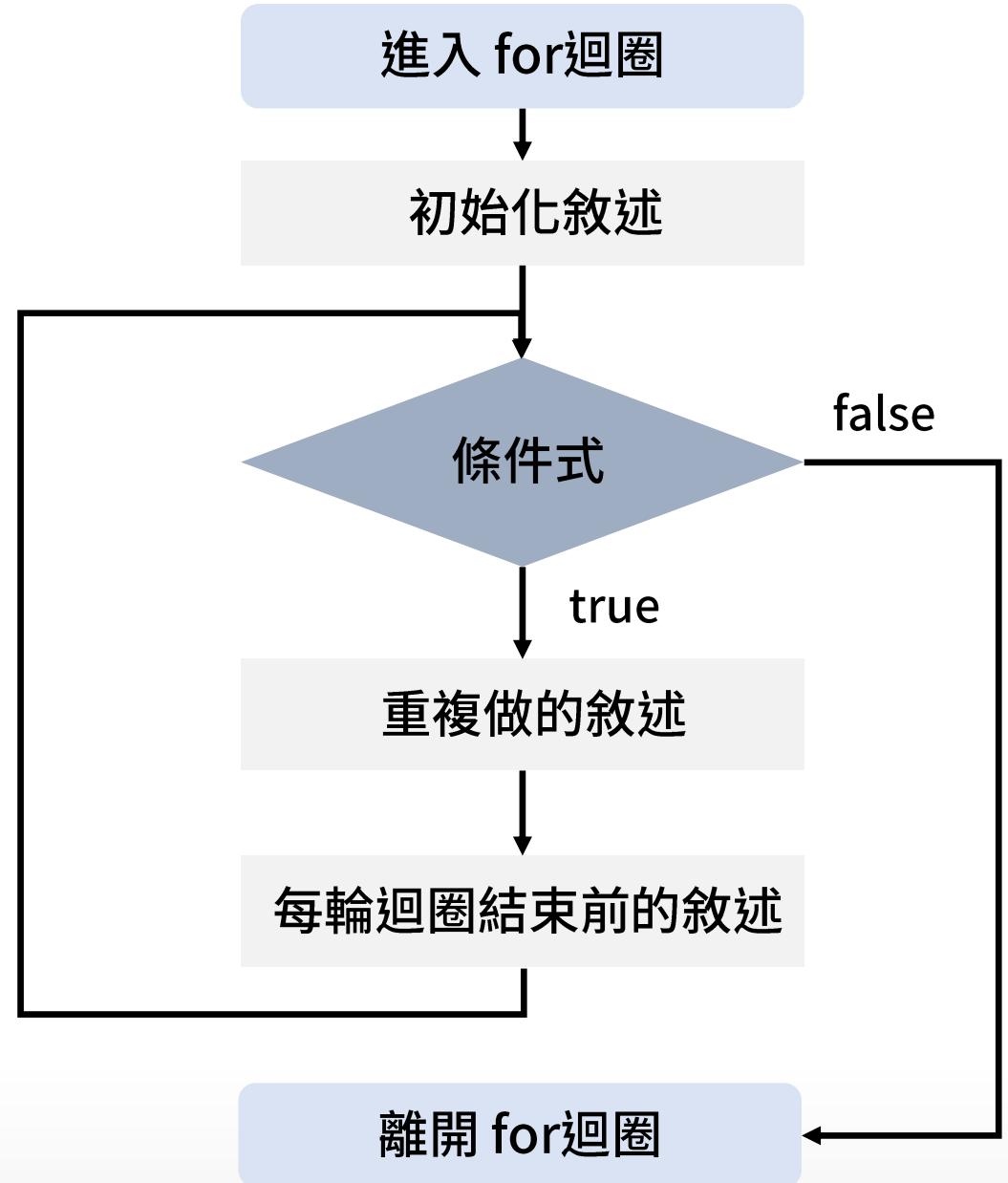
通常用來充當迴圈終止準則，判斷其運算式結果為true或false。

- loop-expression

通常用來遞增迴圈索引。

Control Flow: For-loop

```
Int i=0;  
while ( i<10 ) {  
    cout << i << endl;  
    i += 1;  
}  
  
for ( int i=0; i<10 ; i+= 1 ) {  
    cout << i << endl;  
}
```



Challenge

Sigma

題目說明：輸入一個 n，把 $1+2+3+\dots+n$ 的過程顯示出來，並將結果輸出

題目提示：for-loop 、sigma

輸入：5

輸出：

$1+2+3+4+5$
15

輸入：10

輸出：

$1+2+3+4+5+6+7+8+9+10$
55

Challenge

明明愛數數

題目說明：明明的媽媽叫他從 n 開始數，下一個數字是 $n+1$ ，以此類推。媽媽想知道，明明數了幾個數字之後，他數過的這些數字的總和會超過 m

題目提示：for-loop

輸入：5 30

輸出：

5

輸入：1 20

輸出：

6

Challenge

逆轉人生

題目說明：小陞的帳戶餘額只剩下少少的1999

元，請問你能否寫個程式去將餘額反向輸出，讓他看到帳戶時可以開心一點

題目提示：for-loop

輸入：1999

輸出：

9991

輸入：1022

輸出：

2201

Control Flow: nested-loop

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

Control Flow: nested-loop

字面意義

巢狀指的是像鳥巢一般，一層包一層包下去，然而在C++中指的是
while/for 裡面還有 while/for 的情況。

生活舉例

用時鐘來比喻，分針就是內層迴圈，時針就是外層迴圈。當內層迴圈執行一輪之後，外層迴圈才會進到下一項。

Control Flow: nested-loop

```
for (int i = 0 ; i < 3 ; i ++ ) {  
    for (int j = 0 ; j < 2 ; j ++ ) {  
        cout<<i<<j;  
    }  
}
```

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

Challenge

質因數分解

題目說明：輸入一個正整數，將其寫成幾個質因數的乘積輸出(例如：60 -> 2*2*3*5)

輸入：60

輸出：2 2 3 5

Challenge

台灣人愛排隊

前情提要：只要有名的品牌進駐，就會有很多人半夜去排隊，只為了搶頭香。身為國家科技未來的你，請用程式模擬出台灣特有的排隊熱潮!!

題目說明：分別輸入兩個數表示排隊人數、商品數，並再運算後將排隊狀況用程式輸出來，記得在商品數不夠時，跳出提醒

輸入：5 3

輸出：

1 2 3 4 5
2 3 4 5
3 4 5
4 5

商品銷售一空

Array

/ 如何儲存學生的各項成績？

Syntax

type label [size] ;

type label [size] = {...} ;

- int a[3] ;
- char a[2] = { 'A', 'B' } ;

- int t;
- int a[t];

In memories

Int A[3] = {96 , 45 , 24 }



A[0]					A[1]				A[2]				
0X68	0X69	0X70	0X7A	0X7B	0X7C	0X7D	0X7E	0X80	0X81	0X82	0X83	0X84	0X85
...	96				45				24				...



4 byte

index

A[3] = {96 , 45 , 24 }



Challenge

剪刀石頭布

前情提要：小陞和小莉玩剪刀石頭布，勝負規則是七戰四勝，請幫他們計算各贏幾局，並宣判誰勝誰負。

題目說明：請輸入兩個長度為7的陣列，分別代表小陞和小莉的出拳，陣列內容為1,2,3 其中之一。1=石頭, 2=剪刀 , 3=布

輸入：

1 2 3 2 1 2 3
2 1 3 2 1 1 2

輸出：

小陞贏了: 1 局
小莉贏了: 2 局
平手

輸入：

1 2 2 1 1 2 1
2 1 3 2 2 1 3

輸出：

小陞贏了: 5 局
小莉贏了: 2 局
勝者是小陞

Challenge

迴文

根據迴文的定義，輸入一個句子後去判斷是否為迴文

Challenge

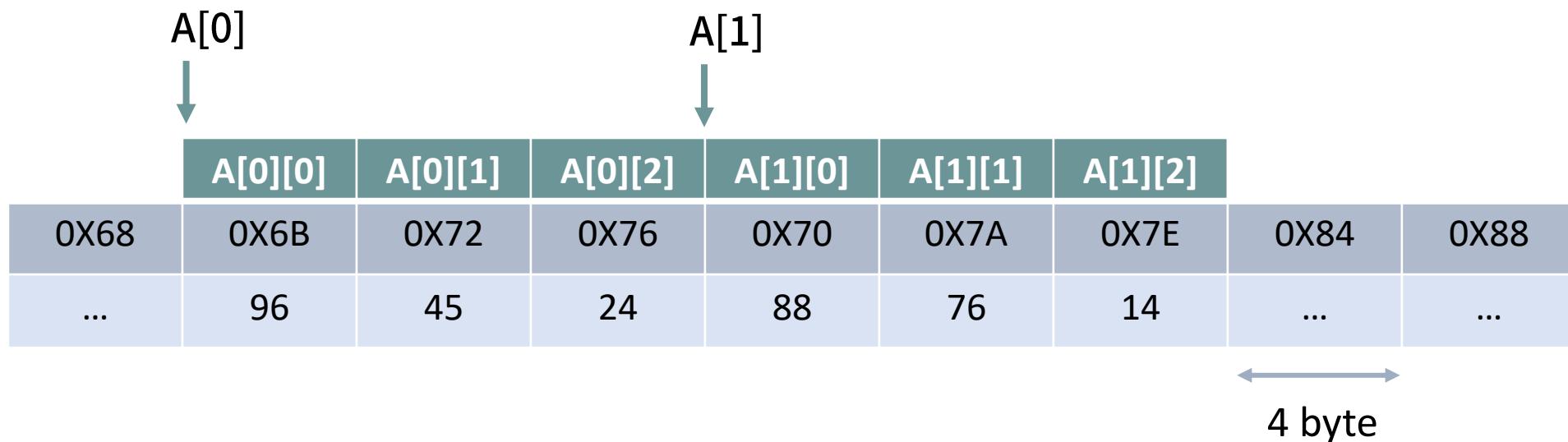
找出出現次數最多的字母

給定一個句子，去判斷句子中的出現最多次數的字母，並將答案輸出。

```
char sentence[13] = {'a', 'r', 'i', 'g',
'n', 'v', 'i', 'f', 'w', 'j', 'l', 'e', 'f', 'i'};
```

Dimension Array

$$A[2][3] = \{ \{ 96, 45, 24 \}, \{ 88, 76, 14 \} \} = \{ 96, 45, 24, 88, 76, 14 \}$$



Challenge

巴斯卡三角形

前情提要：信陞在上數學課時，老師跟他說
 $(a+b)^n$ 展開後每一項的係數和巴斯卡三角
形有關，可以的話就記住。但是愛搞怪的信
陞怎麼可能乖乖記住呢！

題目說明：請輸出巴斯卡三角形(5層)

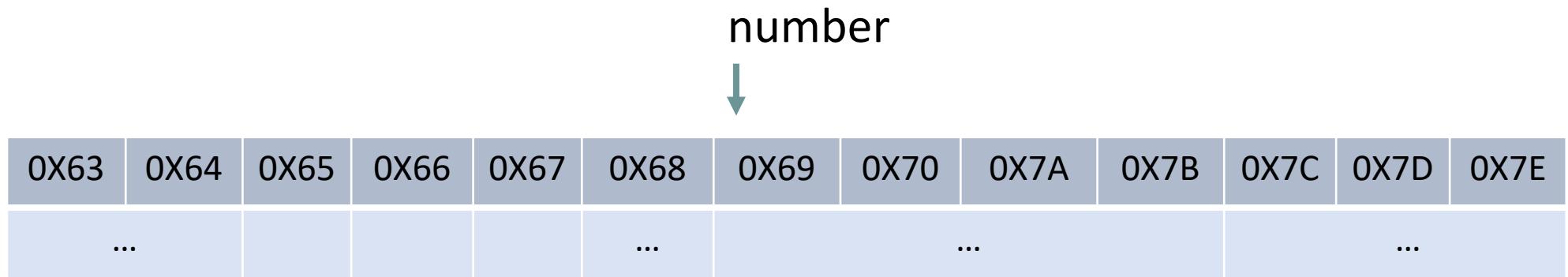
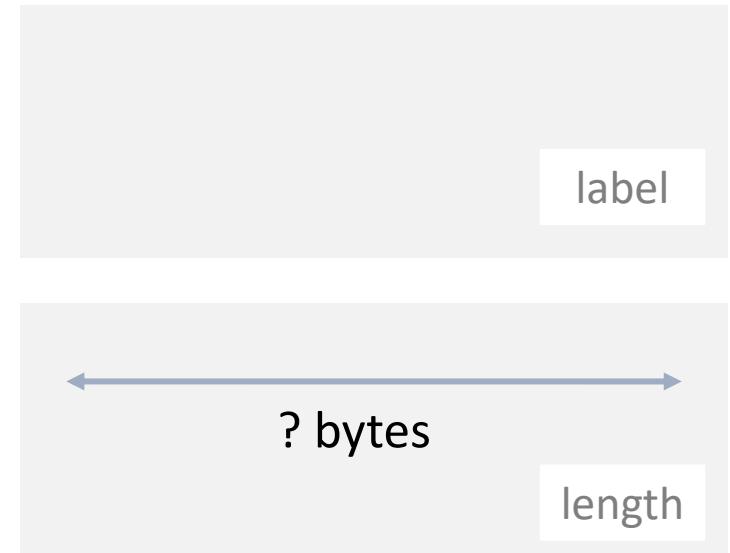
1				
1	1			
1	2	1		
1	3	3	1	
1	4	6	4	1

Pointer

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

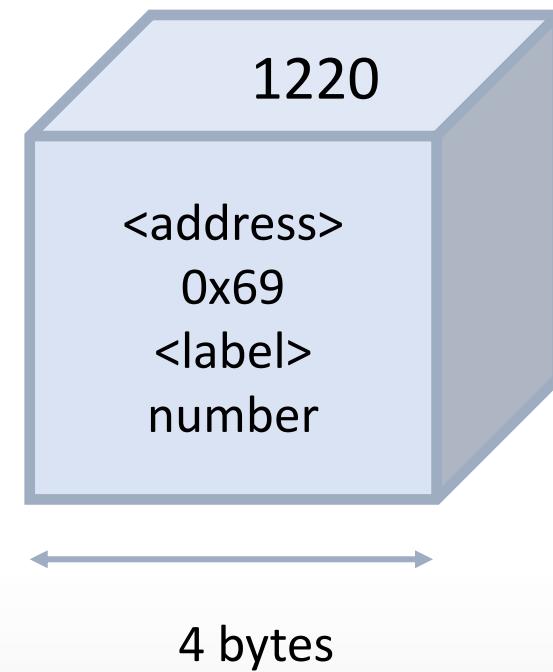
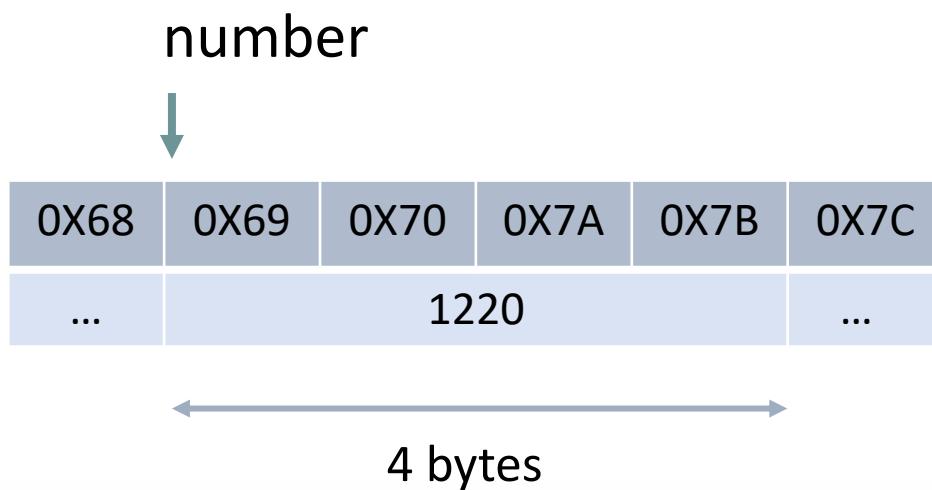
Recall

1. none data ?
2. none type ?
3. none label ?



Recall

```
int number = 1220 ;
```



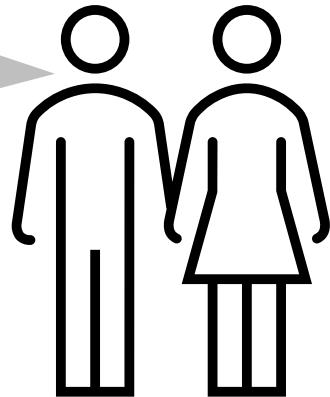
Pointer



外地人

請問這附近很有名的
咖啡廳地址是什麼？

1. 浪浪別哭
2. 台中市南屯區
干城街214巷1號



在地人  

Syntax

`T* aprt`

`aPtr` is pointed a `T` type object/value

`&aPtr`

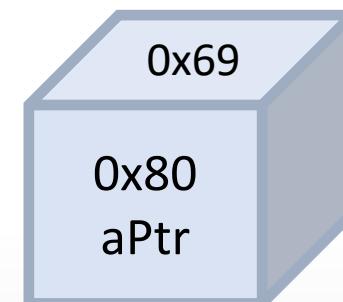
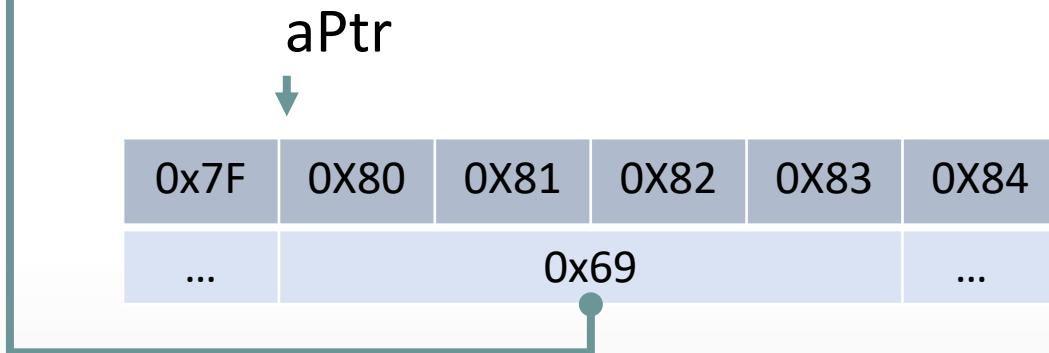
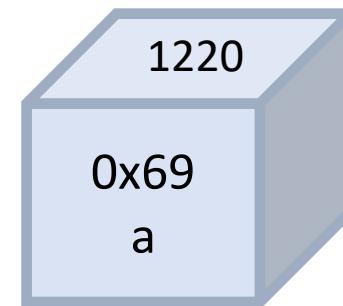
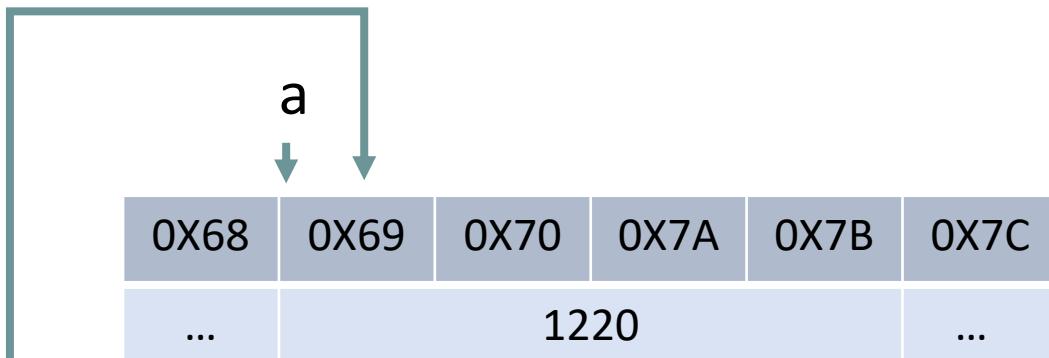
get address of value

`*aPtr`

Access data by an address

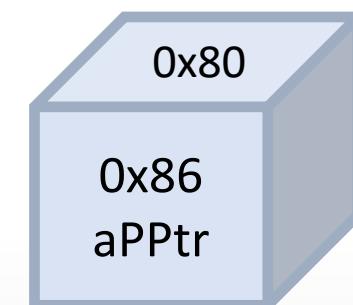
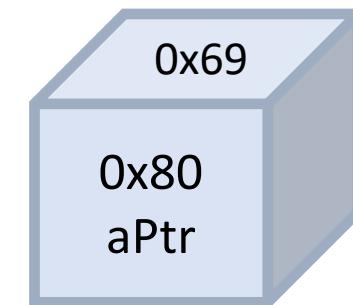
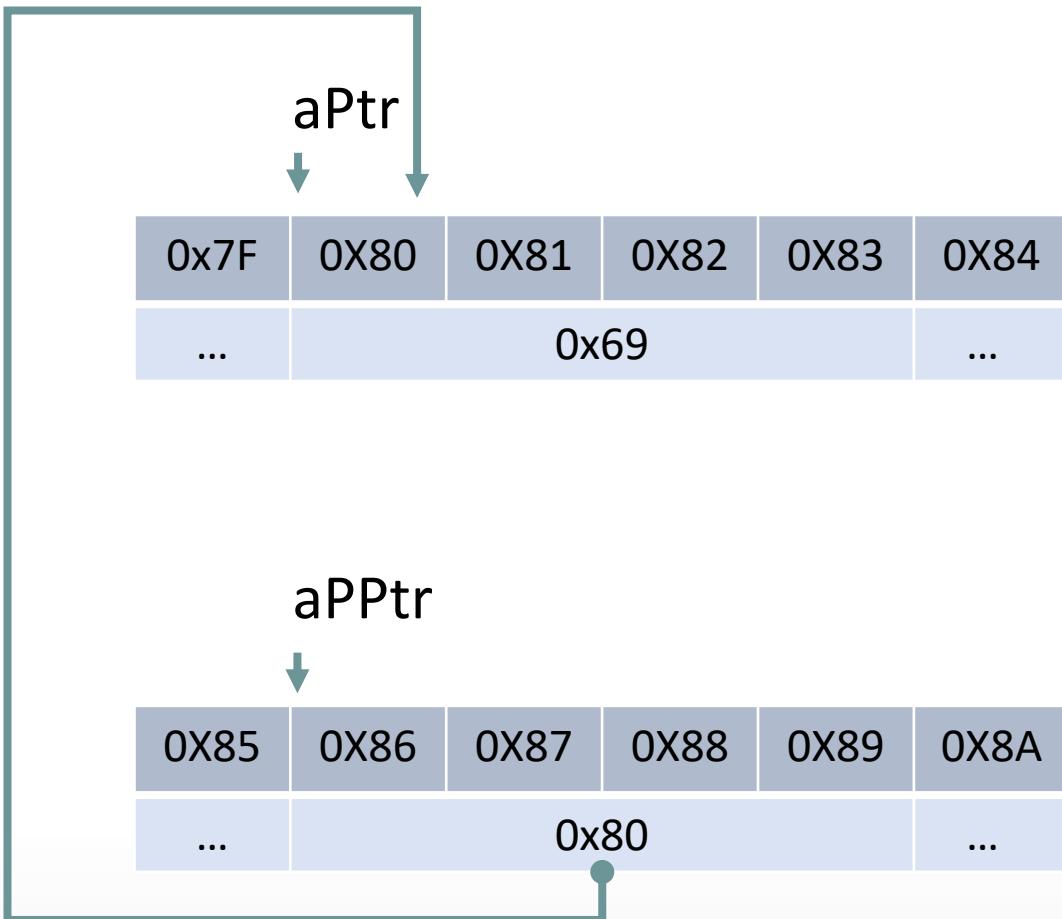
Pointer

```
int a = 1220;  
int* aPtr = &a;
```



Pointer in Pointer

```
int a = 1220;  
int* aPtr = &a;  
int** aPPtr = &aPtr;
```



index

int A[3] = {91 , 12 , 20 }



A[0]					A[1]				A[2]				
0X69 + 0*4byte					0X69 + 1*4byte				0X69 + 2*4byte				
0X68	0X69	0X70	0X7A	0X7B	0X7C	0X7D	0X7E	0X80	0X81	0X82	0X83	0X84	0X85
...	91				12				20				...

 4 byte

Thinking

指標與運算子

題目說明：思考看看為什麼程式碼執行後的結果是會變這樣？

提示：pointer、unary operators、assignment operators

```
int num[3] = {1,2,3};  
  
int* ptr = num;  
  
*ptr++ += 10;  
  
*++ptr += 10;
```

輸出：
num = [11, 2, 13]

Function

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

math.h

- `pow(x, y)` x的y次方
- `sqrt(x)` x的開根號
- `log10(x)` 以10為底的底數x
- `abs()` x的絕對值

Scope

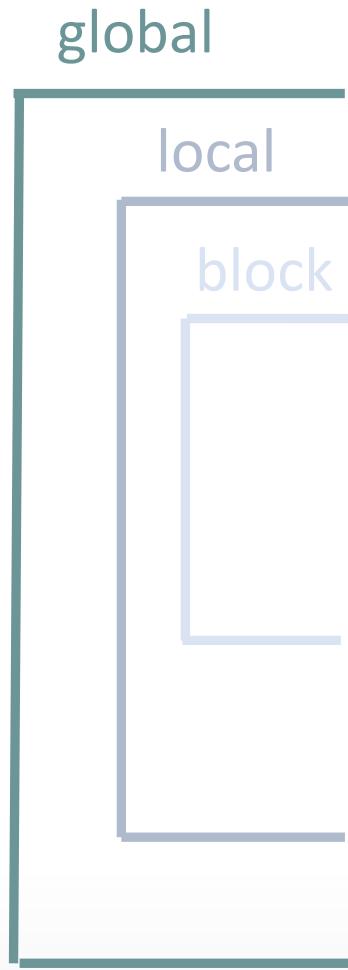
/ 人的生命轉瞬即逝，要好好把握

Basic Scope

The diagram illustrates the scope levels in the C++ code. It features three nested rectangular boxes. The outermost box is dark teal and labeled 'global'. Inside it is a medium teal box labeled 'local'. Inside the 'local' box is a light blue box labeled 'block'. The code itself is positioned to the right of these boxes, with lines of code corresponding to each scope level. The variable 'a' is declared at the global level. The function 'matchZero' is defined at the local level. The variable 'c' is declared within the block level. The output statement 'cout << c;' and the final closing brace are also part of the block level.

```
global
int a = 0 ;
local
void matchZero ( int b ) {
    if ( b == 0 ) {
        int c = 0
    }
    cout << c;
}
//
```

Test



```
int a = 0 ;  
void matchZero ( int a ) {  
    if ( a == 0 ) {  
        int a = 1  
        cout << a;  
    }  
    cout << a;  
}  
cout << a;
```

Calling Stack

```
int main() {  
    int a = 0 ;  
    if ( a == 0 ) {  
        int i = 0 ;  
        for ( ; i < 2 ; i++)  
            int b = 3 ;  
    }  
    cout << a;  
}
```

int main () a=0

Calling Stack

```
int main() {  
    int a = 0 ;  
    if ( a == 0 ) {  
        int i = 0 ;  
        for ( ; i < 2 ; i++)  
            int b = 3 ;  
    }  
    cout << a;  
}
```

If.. i=0
int main () a=0

Calling Stack

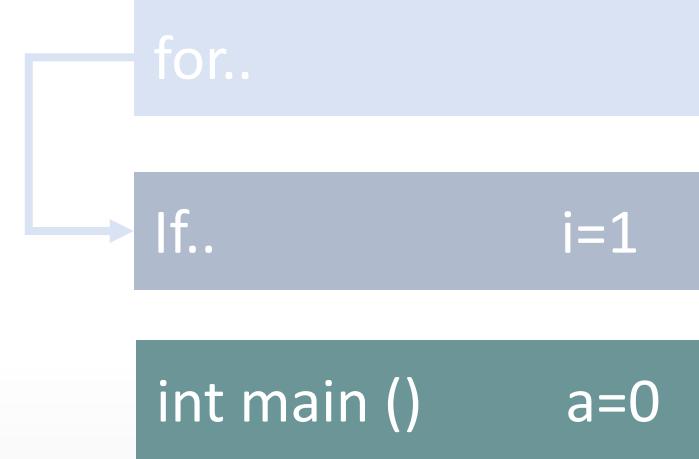
```
int main() {  
    int a = 0 ;  
    if ( a == 0 ) {  
        int i = 0 ;  
        for ( ; i < 2 ; i++)  
            int b = 3 ;  
    }  
    cout << a;  
}
```

If.. i=1

int main () a=0

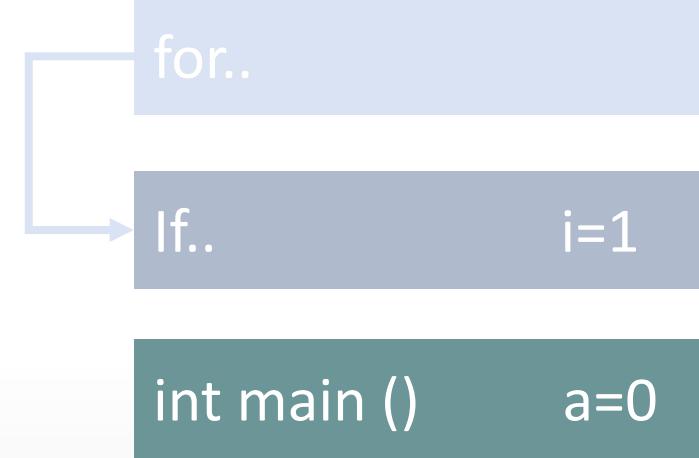
Calling Stack

```
int main() {  
    int a = 0 ;  
    if ( a == 0 ) {  
        int i = 0 ;  
        for ( ; i < 2 ; i++)  
            int b = 3 ;  
    }  
    cout << a;  
}
```



Calling Stack

```
int main() {  
    int a = 0 ;  
    if ( a == 0 ) {  
        int i = 0 ;  
        for ( ; i < 2 ; i++)  
            int b = 3 ;  
    }  
    cout << a;  
}
```



Calling Stack

```
int main() {  
    int a = 0 ;  
    if ( a == 0 ) {  
        int i = 0 ;  
        for ( ; i < 2 ; i++)  
            int b = 3 ;  
    }  
    cout << a;  
}
```

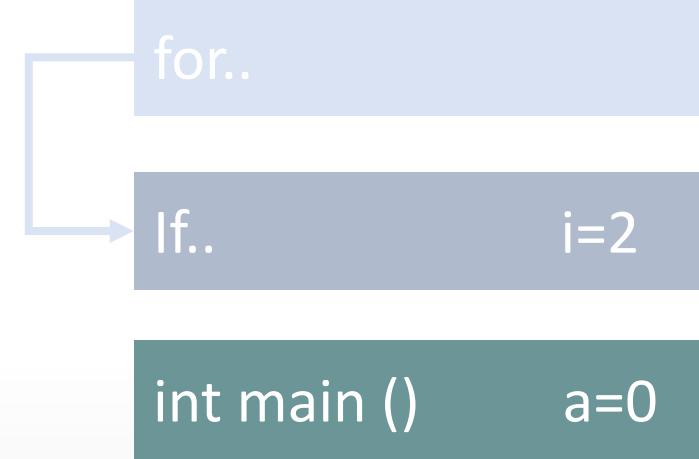
for.. b=3

If.. i=1

int main () a=0

Calling Stack

```
int main() {  
    int a = 0 ;  
    if ( a == 0 ) {  
        int i = 0 ;  
        for ( ; i < 2 ; i++)  
            int b = 3 ;  
    }  
    cout << a;  
}
```



Calling Stack

```
int main() {  
    int a = 0 ;  
    if ( a == 0 ) {  
        int i = 0 ;  
        for ( ; i < 2 ; i++)  
            int b = 3 ;  
    }  
    cout << a;  
}
```

If.. i=2

int main () a=0

Calling Stack

```
int main() {  
    int a = 0 ;  
    if ( a == 0 ) {  
        int i = 0 ;  
        for ( ; i < 2 ; i++)  
            int b = 3 ;  
    }  
    cout << a;  
}
```

If.. i=0

int main () a=0

Recursion

/ 使用相同的方法，解決重複性的問題

呼叫方式

① 呼叫A

```
void A(){  
    ...  
    ...  
    A();  
}
```

void A(){
 ...
 B();
 ...
}

① 呼叫B

void B(){
 ...
 A();
 ...
}② 呼叫A

基本規則

規則

- ① 需要有終點
- ② 每次呼叫都要讓問題變更小
- ③ 不能有循環出現

```
void f(int n){  
    return f(n)+f(n);  
}
```

小試身手

factorial(階乘)

- ① 請用遞迴的方式達成
- ② 可以試著畫圖看看

```
int factorial(int $n) {  
    4! = 4 * 3!  
    3! = 3 * 2!  
    2! = 2 * 1!  
    ....  
}
```

小試身手

factorial(階乘)

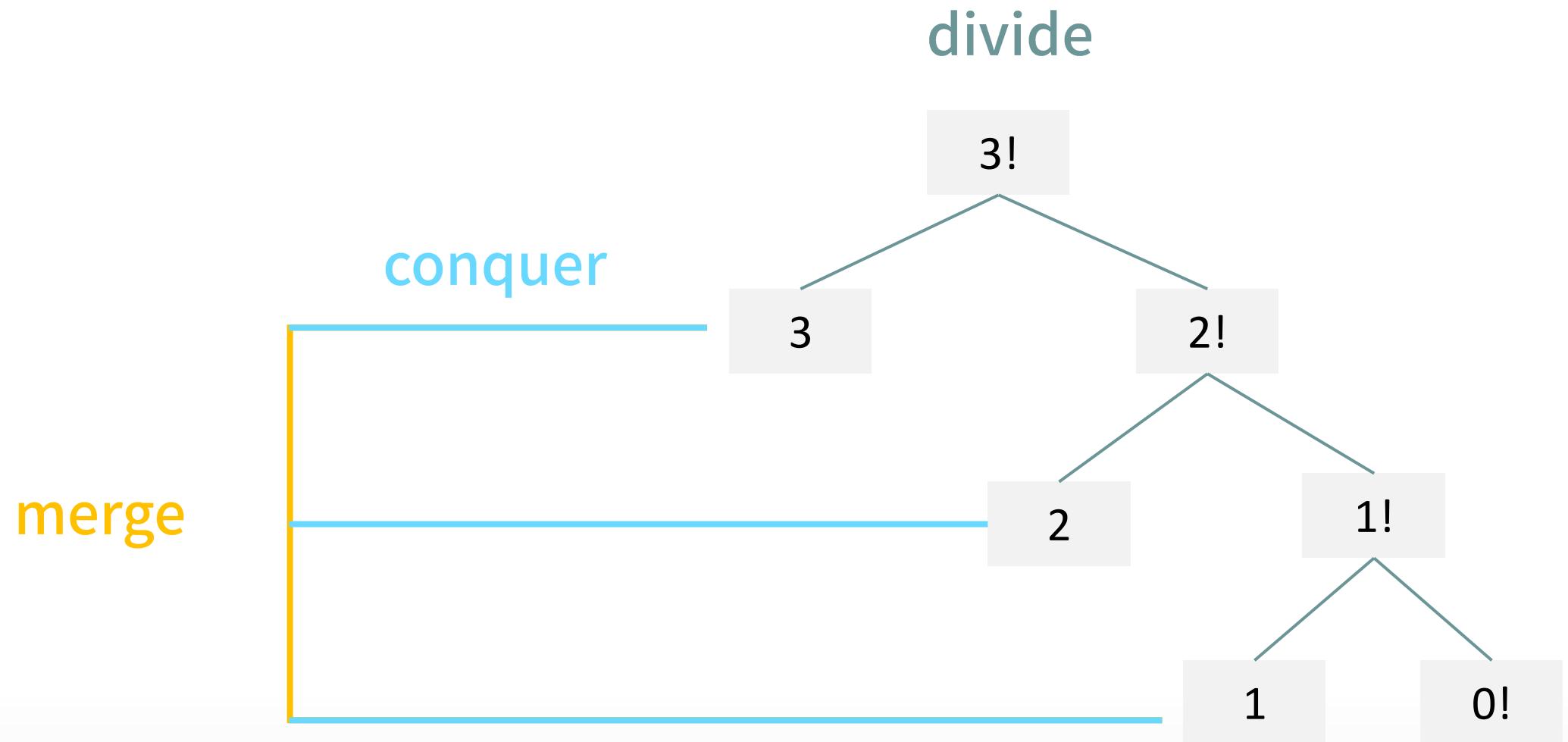
- ① $n! = n * (n-1) !$
- ② $0! = 1$

```
int factorial(int $n) {  
    if ($n == 0)  
        return 1;  
  
    return $n * factorial($n - 1);  
}
```

Divide and Conquer

/ 分而治之，各個擊破

基本步驟

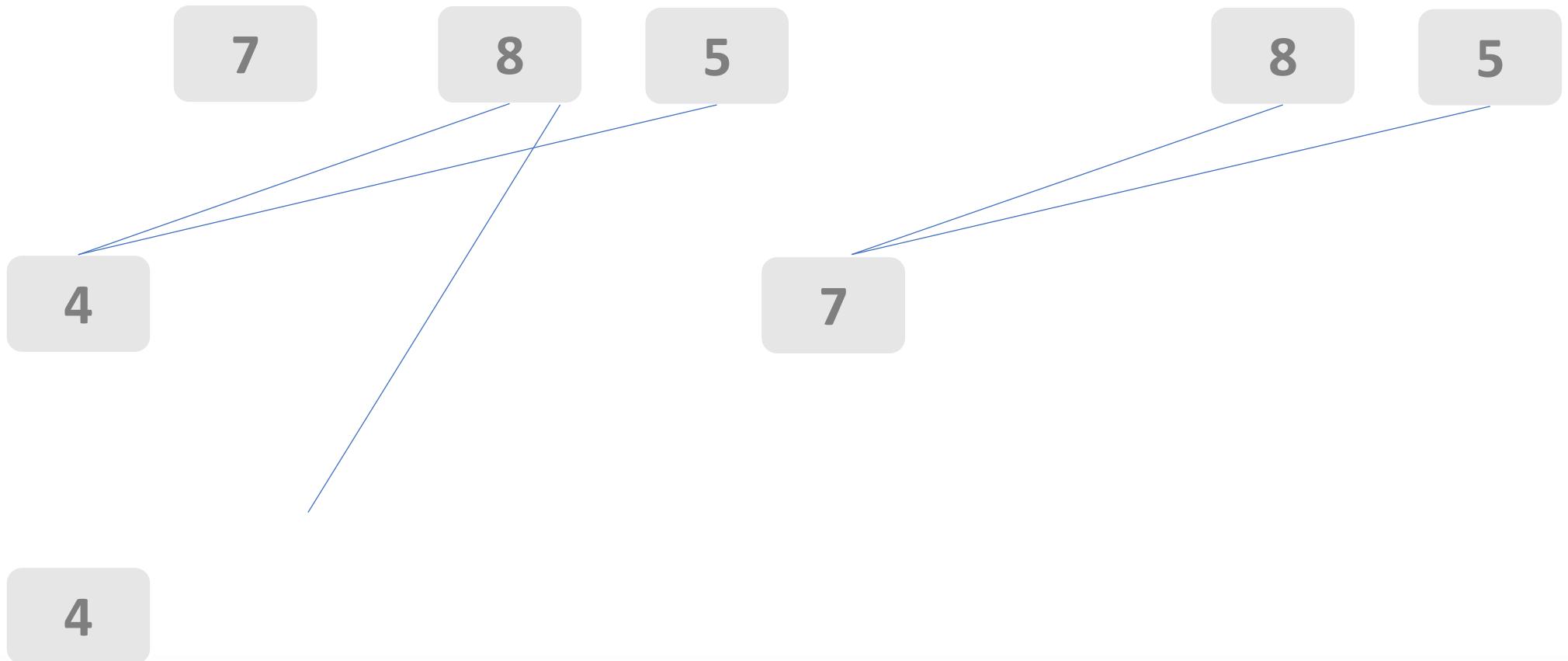


什麼情況

問題特徵

- ① 縮小到一定程度就可以被解決
- ② 可以被分解成相同的子問題
- ③ 子問題的解可以合併成該問題的解
- ④ 子問題之間是互相獨立的

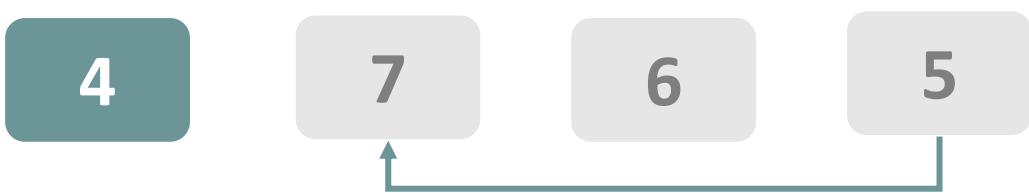
Selection Sort



Selection Sort



- ① 找出未排序陣列中最小的數字
- ② 和未排序陣列首項交換
- ③ 移動陣列的迭代的起點



Selection Sort

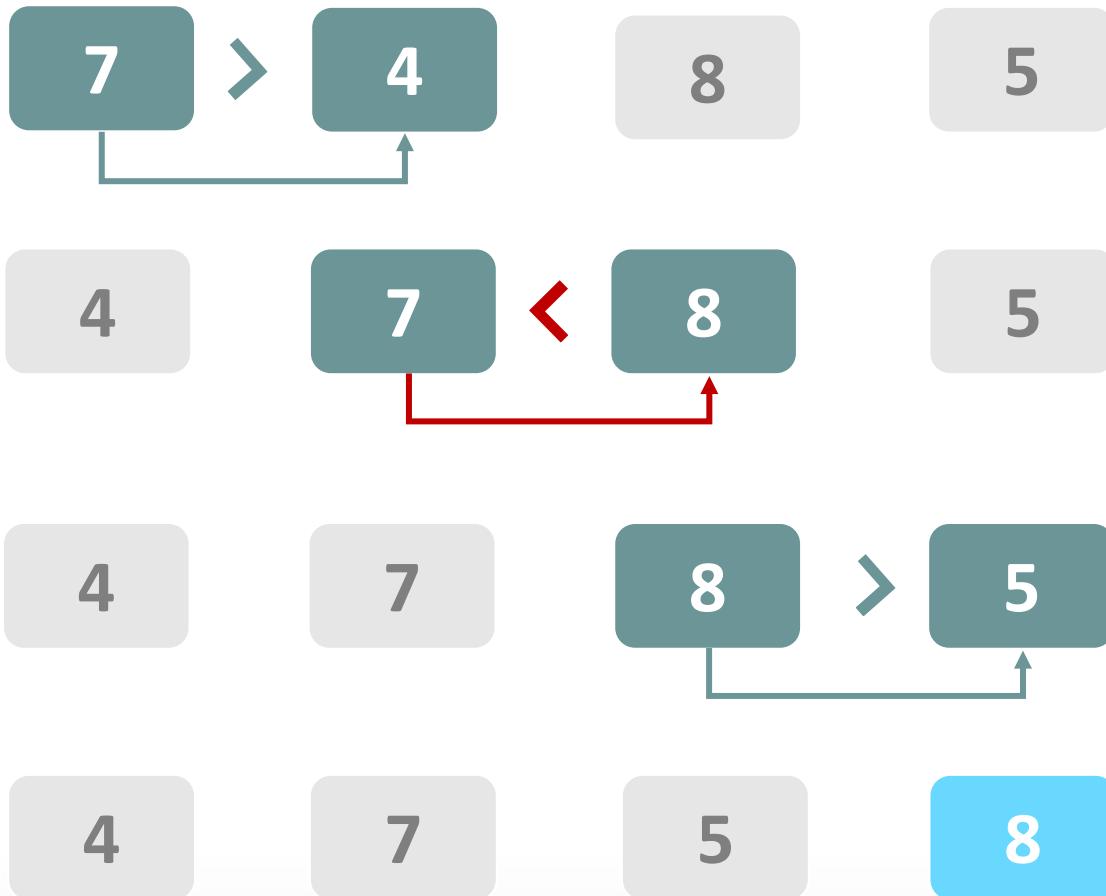
5

7

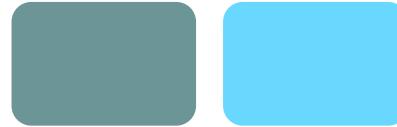
6

5

Bubble Sort



Quick Sort



1

5

8

6

9

10

Preprocessor

/ 程式到底怎麼被執行的？

執行流程

main.cpp

```
#include <iostream>
using namespace std;

int F(int);

int main() {
    cout << F(5);
}
```

vice.cpp

```
int F(int x){
    return x+1;
};
```

執行流程

Preprocessor

recognize meta-information

Compiler

translate source code into machine-dependent object code

Linker

link together all object files into an application

執行流程



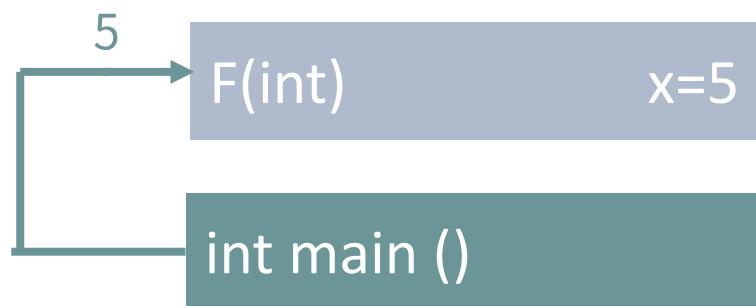
```
g++ -c AirTicket.cpp  
g++ -c main.cpp  
g++ -o prog AirTicket.o main.o  
. /prog.exe
```

Memory

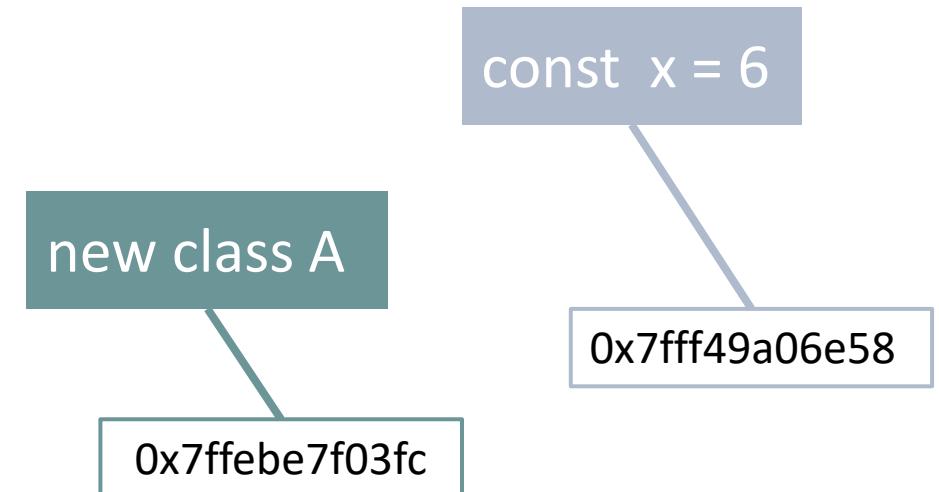
/ 資料存在哪?

基本介紹

Stack - 規律可預測的



Heap- 不規律不可預測的



OOP

/ uses objects to design programs

前言

一個語言如果不改變你的思考方式，就不值得學

- 程式語言的本質究竟是什麼？
- 什麼是「抽象化」？
- 不同系統關注重點不同
 - 在純物件導向程式語言的世界中，一切都是物件！

基本介紹

Object-Oriented Programming

- Object = data fields + methods
- Program = object + object + ⋯ + object

Features

- 封裝 -- 物件傳遞
- 繼承 -- 多人開發
- 多型 -- 程式維護



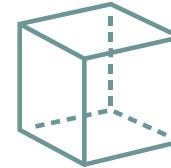
類別

成員變數 (組成)

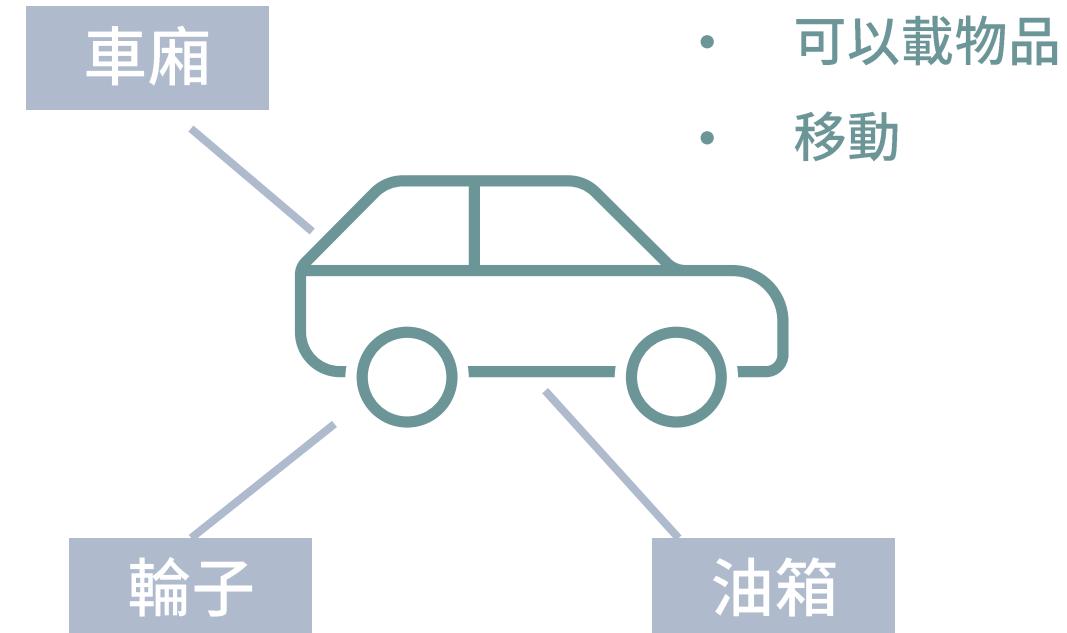
- 車廂
- 輪子
- 油箱

成員函數(功能)

- 載物
- 移動



物件



Instantiation

Constructor

- invoke when an object of your class is created

Destructor

- is invoked automatically when the object goes out of scope

This-Pointer

Pointer

- its own address
- Only member functions have a this-pointer

An implicit parameter

```
name = 'YU,XIN-SHENG';  
  
this->name = 'YU,XIN-SHENG';
```

Encapsulation

Like Function

- hide the actual content , so that the user only needs to know how to use it

Level

- Public
- Protected
- Private

①



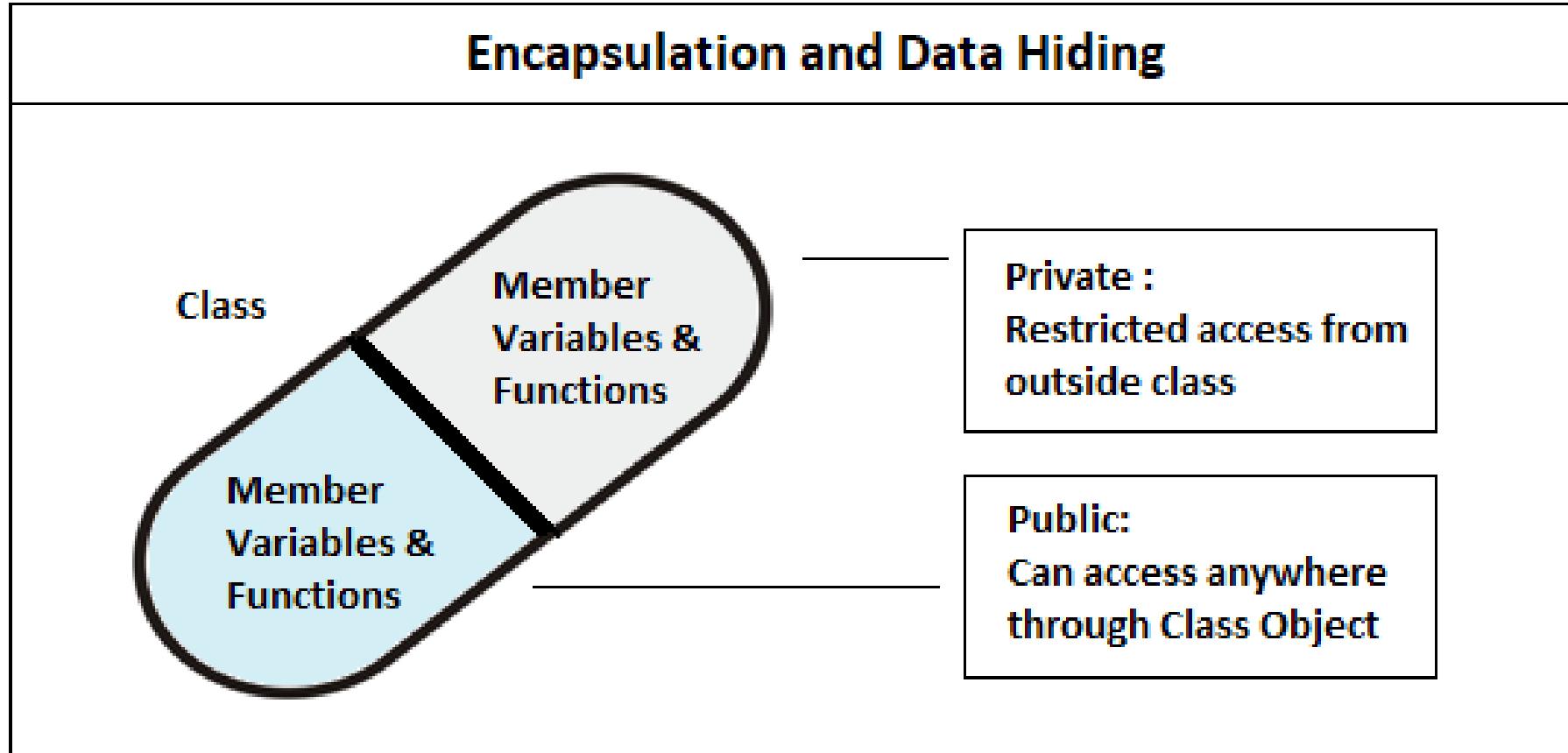
②



3



Encapsulation



- 圖片出自 : <https://www.cpp.thiyagaraaj.com>

Inheritance

Introduction

- This also provides an opportunity to reuse the code
- implement the is a relationship

Shape Problem

- circle is a kind of oval (radius)
- oval is a kind of circle (area overwrite)

Inheritance

Base and Derived Classes

- derived class (~~child~~)
- base class (~~parent~~)
- 繼承享受了取用基底類別內容的好處，卻也必須
背負牽一髮動全身的風險。

Overload

Overload

- In a class, define same method , but parameters are different

```
class A {  
public:  
    print(int x);  
    print(int x, int y);  
    print(int x, string z);  
}
```

Override

Override

- A derived class can override the contents of a method in the base class

```
class Computer {  
    public output(){  
        1、Displayed on the screen  
    }  
};  
  
class ModernComputer extend Computer{  
    public output(){  
        1、Displayed on the screen  
        2、Displayed on the TV  
    }  
};
```

17:50 Start

新興科技遠距教學示範服務計畫之物聯網程式語言教學

Exercises

/ 新興科技遠距教學示範服務計畫之物聯網程式語言教學

APCS concept

14. 假設 x, y, z 為布林(boolean)變數，且 $x=TRUE, y=TRUE, z=FALSE$ 。請問下面各布林運算式的真假值依序為何？(TRUE 表真，FALSE 表假)

- $! (y \quad || \quad z) \quad || \quad x$
- $! y \quad || \quad (z \quad || \quad !x)$
- $z \quad || \quad (x \quad \&\& \quad (y \quad || \quad z))$
- $(x \quad || \quad x) \quad \&\& \quad z$

- (A) TRUE FALSE TRUE FALSE
(B) FALSE FALSE TRUE FALSE
(C) FALSE TRUE TRUE FALSE
(D) TRUE TRUE FALSE TRUE

APCS concept

12. 若函式 `rand()` 的回傳值為一介於 0 和 10000 之間的亂數，下列那個運算式可產生介於 100 和 1000 之間的任意數(包含 100 和 1000)？

- (A) `rand() % 900 + 100`
- (B) `rand() % 1000 + 1`
- (C) `rand() % 899 + 101`
- (D) `rand() % 901 + 100`

APCS concept

22. 如果 X_n 代表 X 這個數字是 n 進位，請問 $D02A_{16} + 5487_{10}$ 等於多少？

(A) 1100 0101 1001 1001₂

(B) 16263₁₈

(C) 58787₁₆

(D) F599₁₆

APCS concept

16. 右側程式執行過後所輸出數值為何？

- (A) 11
- (B) 13
- (C) 15
- (D) 16

```
void main () {
    int count = 10;
    if (count > 0) {
        count = 11;
    }
    if (count > 10) {
        count = 12;
        if (count % 3 == 4) {
            count = 1;
        }
        else {
            count = 0;
        }
    }
    else if (count > 11) {
        count = 13;
    }
    else {
        count = 14;
    }
    if (count) {
        count = 15;
    }
    else {
        count = 16;
    }
    printf ("%d\n", count);
}
```

APCS concept

15. 若以 **f(22)** 呼叫右側 **f()** 函式，總共會印出多少數字？

- (A) 16
- (B) 22
- (C) 11
- (D) 15

```
void f(int n) {
    printf ("%d\n", n);
    while (n != 1) {
        if ((n%2)==1) {
            n = 3*n + 1;
        }
        else {
            n = n / 2;
        }
        printf ("%d\n", n);
    }
}
```

APCS concept

22. 右側 **f()** 函式執行後所回傳的值為何？

- (A) 1023
- (B) 1024
- (C) 2047
- (D) 2048

```
int f() {  
    int p = 2;  
    while (p < 2000) {  
        p = 2 * p;  
    }  
    return p;  
}
```

APCS concept

23. 請問右側程式，執行完後輸出為何？

- (A) 2417851639229258349412352 7
- (B) 68921 43
- (C) 65537 65539
- (D) 134217728 6

```
int i=2, x=3;
int N=65536;

while (i <= N) {
    i = i * i * i;
    x = x + 1;
}
printf ("%d %d \n", i, x);
```