C++程式設計基礎
week 3

陳毅

上週回顧

- 條件控制
- 迴圈控制

條件控制

- 關係運算符（>, <, >=, <=, ==, != ）
- 邏輯運算符（&& (AND), || (OR), ! (NOT)）
- if-else if-else
- switch

```
if (條件式1){
    程式區塊1;
    ...
} else if (條件式2){
    程式區塊2;
    ...
    ...
} else if (條件式N){
    程式區塊N;
    ...
} else {
    程式區塊N+1;
}
```

```
switch (變數/運算式){
    case 值1:
        程式敘述1;
        ...
        break;
    case 值2:
        程式敘述2;
        ...
        break;

        ...

    case 值N:
        程式敘述N;
        ...
        break;
    default:
        程式敘述N+1;
        ...
        break;
}
```

迴圈控制

- for迴圈
- while迴圈
- do-while迴圈
- continue敘述
- break敘述

```
for (起始式; 判斷式; 運算式){
    程式敘述;
    ...
}
```

```
while (判斷式){
    程式敘述;
    ...
}
```

```
do {
    程式敘述;
    ...
} while (判斷式);
```

練習

- 以亂數擲10000次骰子，並分別列出出現1、2、3、4、5、6點的次數。(Dice.cpp)
- 輸入兩個數字，求最大公因數，與最小公倍數。
- 列出1~100中的所有質數。
- 列出九九乘法表。

本週概要

- 使用者函數（自定義函數）
  - 函數的架構
  - 引數（參數）的傳遞

使用者函數（自定義函數）

好處

- 將重複功能的部分，使用函數替代，增加程式碼的**可利用性**。
- 將複雜的程式切分成數個較小且簡單的問題，在維護和修改上會更為方便。
- 程式語言中的函數並**不單單只是數學函數**，它真正有用的是，可以**實作並包裝一個功能**，讓程式設計師能有效率地去拼組出一個複雜的程式。
- **黑盒子**
  - 與他人合作開發時，可將自己負責的部分進行**封裝**，別人只需要知道輸入與輸出，不需要知道實作方式，即可使用。

3-1.cpp

函數的架構

函數雛型(Declaration)

用來告訴編譯器，這個程式會有哪些函數。

```
型態 函數名稱(引數1型態, 引數2型態, ...);
```

函數宣告(Definition)

用來定義一個函數實際的執行內容。

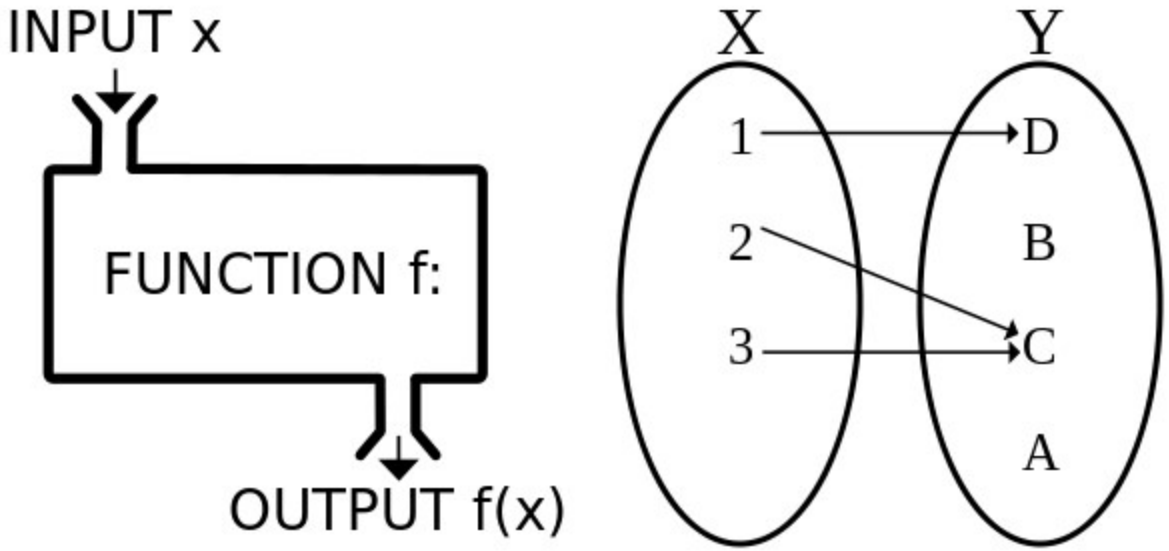
```
型態 函數名稱(引數1, 引數2, ...){
    程式敘述;
    ...
}
```

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

void I_AM_A_FUNCTION(int, int);

int main(){

    I_AM_A_FUNCTION(1, 2);

    return 0;
}

void I_AM_A_FUNCTION(int a, int b){
    cout << a << " " << b << endl;
    return;
}
```

3-2.cpp

函數的架構

- return敘述
  - 可以將變數傳回呼叫它的函數內。
  - 傳回的值必須與函數宣告時的型態相同。
  - 在函數中，一旦執行到return敘述，程式將直接結束這個函數的執行。



圖片來源: https://ithelp.ithome.com.tw/articles/10192017

練習

- 請寫一個函數，用以計算整數的次方。
  - 輸入：int a, int n
  - 輸出：a^n
- 請寫一個函數，找出四個整數中的最大值。
  - 輸入：int a, int b, int c, int d
  - 輸出：max(a,b,c,d)
- 請寫一個函數，在螢幕上輸出10行「Hello, world!」。
  - 輸入：無
  - 輸出：無

下週放假！春假愉快！