

計算機程式設計

C語言 Function

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

Function

□ 程式設計使用函式的好處

- 將大程式切割由多人撰寫，利於團隊分工，縮短程式開發時間。
- 可縮短程式長度，具結構化可讀性高，利於測試驗證、除錯維護、重複使用
- 當再開發類似功能產品，只需稍微修改即可套用。

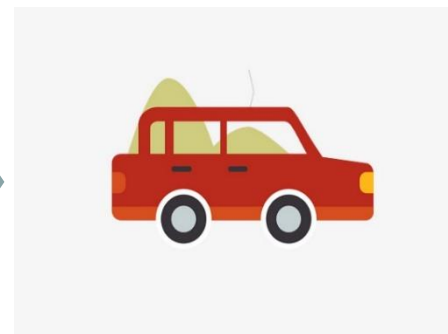
零件:輪子、儀錶板、引擎



工廠組裝汽車



輸出:汽車



Function 呼叫

❑ 呼叫function

- `x = abs(-5);`
- `printf("Hello, world");`

❑ 內建函式可#include後使用

- `scanf(...), printf(...)`

❑ 函式的定義可獨立成一個原始檔，個別編譯後再聯結一起。

運算後回傳的資料型別 命名方法如同變數 引數/參數

```
data_type Function_name (pra1, pra2,... ){  
    函式主體指令 (body statement)  
}
```

必先定義而後使用

任何函式主體指令要縮排

Function 定義與使用

❑ 函式建立後不會執行，須在程式中呼叫函式才會執行

○ [變數 =] 函式名稱 ([參數宣告串列])

❑ parameter / 參數 / **formal parameter**

○ 宣告，被呼叫時必須接收什麼資料

```
data_type Function_name (para1, para2, ... );
```

❑ argument / 引數 / **actual argument**

○ 呼叫函式時，放在括號內的變數/值


```
Function_name (arg1, arg2, ... );
```

傳遞的動作為賦值運算，
pra1 = arg1,
pra2 = arg2
...
praN = argN

Function 定義

□ 自行設計與定義函式

```
1. int CreateCar(int wheels, int dashboard, int engine) {  
2.   car = dashboard + wheels + engine  
3.   return car  
4. }  
5. int main() {  
6.   carA = CreateCar(wheels, dashboard, engine);  
7.   printf("%d", carA);  
8.   return 0;  
9. }
```

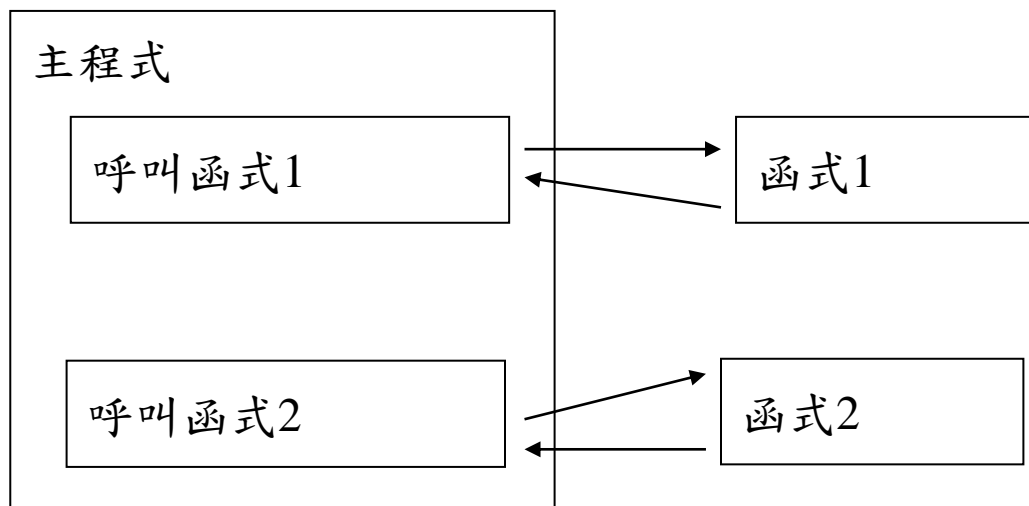


執行順序: 5, 6 (=右邊), 1, 2, 3, 4, 6 (=左邊), 7, 8, 9

Function定義與使用

□ 函式執行結束

- 執行到return指令、或最後}時，
- 會離開結束函式，
- 回傳結果(值)，也可不回傳。



Function

□ 計算三個數a, b, c 的標準差 sd

- 計算三個數的平均值 average
- 三個數分別與平均值的差之平方，三個加總後平均
- 再開根號

➤ `#include <math.h>`

➤ `x = sqrt(y)`

$$SD = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}$$

數學函式的使用

math

□ 在程式前加：`#include <math.h>`

○ `exp` 函式是指數函式 `ex`，使用格式：`exp(x)`

➤ 使用前將 `x` 宣告成 `double` 或 `float`。

○ `log` 函式，自然對數函式，使用格式：`log(x)`

loge

➤ 使用前將 `x` 宣告成 `double` 或 `float`。

○ `log10` 函式，以 10 為底的對數函式，使用格式：`log10(x)`

➤ 使用前將 `x` 宣告成 `double` 或 `float`。

○ `sqrt` 函式，求某數平方根值，使用格式：`sqrt(x)`

➤ 使用前將 `x` 宣告成 `double` 或 `float`。

數學函式的使用


- ❑ floor函式，傳回不大於x的最大整數，使用格式：floor(x)
- ❑ ceil函式，傳回不小於x的最小整數，使用格式：ceil(x)
- ❑ fabs函式，傳回x的絕對值，使用方式：fabs(x)
- ❑ pow(double x, double y)
- ❑ sin(x), cos(x), tan(x)

Function

□ 計算三個數a, b, c 的標準差 sd

○ 不使用 function，算三次

```
#include <math.h>
#include <stdio.h>
int main() {
    int a=3, b=4, c=5, sum = 0;
    double average=0.0, sd= 0.0;
    average = (a + b + c)/3.0;
    sum = (average -a)*(average -a) + (average -b)*(average -b) + (average -c)*(average -c);
    sd = sqrt(sum/3);
    a = 7; b = 8; c = 9;
    average = (a + b + c)/3.0;
    sum = (average -a)*(average -a) + (average -b)*(average -b) + (average -c)*(average -c);
    sd = sqrt(sum/3);
    a = 17; b = 18; c = 19;
    average = (a + b + c)/3.0;
    sum = (average -a)*(average -a) + (average -b)*(average -b) + (average -c)*(average -c);
    sd = sqrt(sum/3);
    return 0;
}
```



Function

□ 計算三個數a, b, c 的標準差 sd

○ 使用 function，算三次

```
#include <math.h>
#include <stdio.h>
double getSD(int a, int b, int c) {
    double average = (a + b + c)/3.0;
    double sum = (average -a)*(average -a) + (average -b)*(average -b) + (average -c)*(average -c);
    return (sqrt(sum/3));
}
int main() {
    int a=3, b=4, c=5, sum = 0;
    double average=0.0, sd= 0.0;
    sd = getSD(a, b, c);
    a = 6; b = 7; c = 8;
    sd = getSD(a, b, c);
    a = 17; b = 18; c = 19;
    sd = getSD(a, b, c);
    return 0;
}
```

good 重複

Function

□ 計算三個數a, b, c 的樣本標準差 s

○ 使用 function，改一個地方

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$$

```
#include <math.h>
#include <stdio.h>
double getSD(int a, int b, int c) {
    average = (a + b + c)/3.0;
    sum = (average -a)*(average -a) + (average -b)*(average -b) + (average -c)*(average -c);
    return (sqrt(sum/(3-1)));
}
int main() {
    int a=3, b=4, c=5, sum = 0;
    double average=0.0, sd= 0.0;
    sd = getSD(a, b, c);
    a = 6; b = 7; c = 8;
    sd = getSD(a, b, c);
    a = 17; b = 18; c = 19;
    sd = getSD(a, b, c);
    return 0;
}
```

命名規則

- ❑ 函數之命名應有意義，指出函數目的或回傳資料
 - 以駝峰式(Camel-Case)命名、或以小寫或輔以底線命名
 - 避免非慣用縮寫
- ❑ 若由多個單字組成，第一個單字要小寫，從第二個單字開始，每個單字的第一個字母大寫
 - studentName, bianryData, dataLength。
- ❑ 名稱不使用縮寫，以清楚、明確為原則。
 - 例如 distance 比 d 清楚
 - dataLength 比 length 更明確易理解。
- ❑ Boolean 命名 用 is開頭
 - isPrime
- ❑ 函式表示一種操作，使用動詞作為第一個單字。
 - computeAverage()

命名規則

命名

```
double f() {  
    double x1;  
    int x2, x3, x4;  
    scanf("%d", &x2);  
    scanf("%d", &x3);  
    scanf("%d", &x4);  
    x1=(x2+x3+x4)/3.0;  
    return x1;  
}
```

```
double ComputeAverage() {  
    double average = 0.0;  
    int englishGrade, mathGrage, chineseGrade;  
    scanf("%d", &englishGrade);  
    scanf("%d", &mathGrage);  
    scanf("%d", &chineseGrade);  
    average=(englishGrade+mathGrage+chineseGrade)/3.0;  
    return average;  
}
```



Function

□ function種類

- 1 ○有參數、有回傳值 (傳入資料，處理完後回傳結果)
- 2 ○無參數、有回傳值
- 3 ○無參數、無回傳值
- 4 ○有參數、無回傳值

MVC

Function

□ 有參數、有回傳值

○ 輸入參數為何?回傳值為何?

```
#include <math.h>
double getSD(int a, int b, int c) {
    average = (a + b + c)/3.0;
    sum = (average -a)*(average -a) + (average -b)*(average -b) + (average -c)*(average -c);
    return (sqrt(sum/2));
}
```

□ 無參數、有回傳值: 輸入資料

```
double ComputeAverage() {
    double average = 0.0;
    int englishGrade, mathGrage, chineseGrade;
    scanf("%d", &englishGrade);
    scanf("%d", &mathGrage);
    scanf("%d", &chineseGrade);
    average=(englishGrade+mathGrage+chineseGrade)/3.0;
    return average;
}
```


Function

□ 有參數、無回傳值: 輸出報表、結果

```
void myFunction(char name) {  
    printf("Hello~ %c", name);  
}  
int main() {  
    myFunction('K');  
    myFunction('O');  
    return 0;  
}
```

□ 無參數、無回傳值

```
void myFunction() {  
    printf("Hello~\n");  
    printf("Hi~\n");  
}  
int main() {  
    myFunction();  
    myFunction();  
    return 0;  
}
```

可維護程式

□ 程式品質

*Any fool can write code that a computer can understand.
Good programmers write code that humans can understand.*

- Martin Fowler.

任何傻瓜都能寫出電腦能懂的程式碼。而只有好的程式設計師才能寫出人能懂的程式碼

- Martin Fowler.

Quality is not an act. It is a habit.

- Aristotle

品質不是動作，是一種習慣

- Aristotle

Exercise

- ❑ 程式要切割模組
 - 好的設計
 - MVC架構
- ❑ 將計算三人成績總和、與標準差，練習寫成至少三個function
 - input
 - compute
 - output

Exercise

□ 計算BMI公式: $BMI = \text{體重(公斤)} / \text{身高}^2(\text{公尺平方})$

○ 測試案例: 52公斤的人, 身高155公分, BMI為: $52(\text{公斤}) / (1.55 * 1.55)(\text{公尺平方}) = 21.64412$ 。兩位小數 21.64

```
double computeBMI(int kg, int M):
```

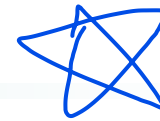
```
    #BMI = round(kg/(M*M),2)          #四捨五入取兩位小數
```

```
    BMI = ((100*kg/(M*M))/1.0)/100;  #乘 100取整數, 再除100取兩位小數
```

```
    return BMI;
```

```
}
```

Function 不定參數



□ 不定長度引數關鍵字

○ va_list: 一個特殊的型別 (type) , 在 ¹va_start、²va_arg 與 ³va_end 三個巨集 (macro) 中當作參數使用。

¹ ○ va_start: 啟始不定長度引數的巨集，第一個引數是 va_list，第二個引數是最後一個具名參數。

² ○ va_arg: 讀取不定長度引數的巨集。

³ ○ va_end: 終止不定長度引數的巨集。

□ 宣告不定長度引數時，函式定義參數 ... 前至少要有一個具名參數，之後使用 ... 表示將使用不定長度引數，例如：

○ void foo(int, ...);

□ 使用 va_arg 巨集取出引數內容時，須指定以何種資料型別取出，例如：

○ va_arg(num_list, double);

#define

Function 不定參數

□ 不定長度引數使用

```
#include <stdio.h>
#include <stdarg.h>
void foo(int len, ...) {
    va_list args;
    va_start(args, len);
    for(int j = 0; j < len; j++) {
        printf("%.1f\n", va_arg(args, double));
    }
    va_end(args);
}
int main() {
    double x = 1.1, y = 2.1, z = 3.9;
    double a = 0.1, b = 0.2, c = 0.3;
    foo(3, x, y, z);
    foo(6, x, y, z, a, b, c);
    return 0;
}
```