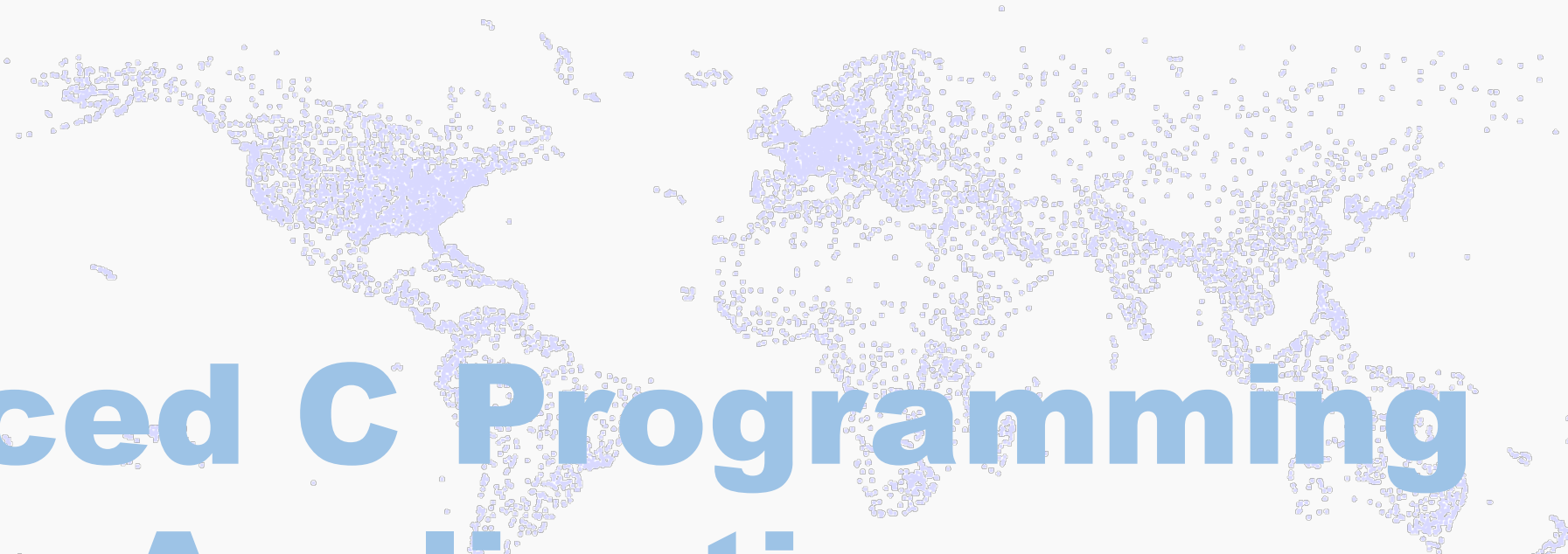


<Adv C & App/>



Advanced C Programming And It's Application

Function Basic

Assistant Prof. Chan, Chun-Hsiang

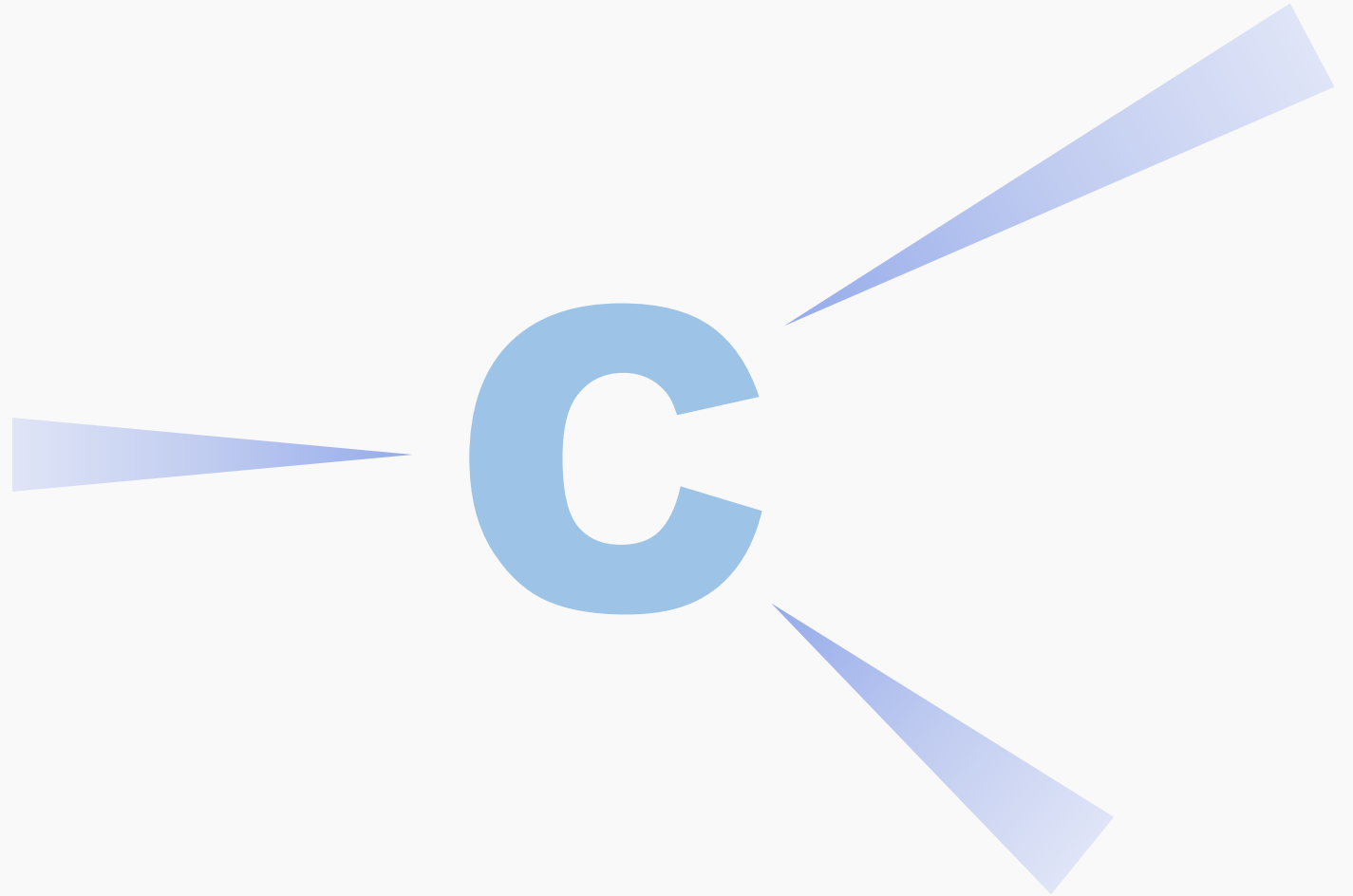
Department of Artificial Intelligence, Tamkang University

Oct. 13, 2021

</ Adv C & App >

大綱

- [1] Define function
- [2] Arguments and return
- [3] Inline function
- [4] Assignments



<Define function/>

宣告函數

雖然我們在前幾週教過許多不同的內建函數 (= 標準函數庫 = **standard library**)，但是不可能我們所需要用到的函數都有對應的內建函數。因此練習做出一個自定義函數是一個很重的事情。

可能會有人會有一個疑問，為甚麼我要用函數？我也可以多寫幾遍，或是寫長一點的程式碼一樣也可以完成這件事情，那做這件事情的意義為何？

Lab 3-1:

列出幾個你覺得自定義函數使用的時機或是好處？

函數的好處

基本上我覺得會有以下三個好處：

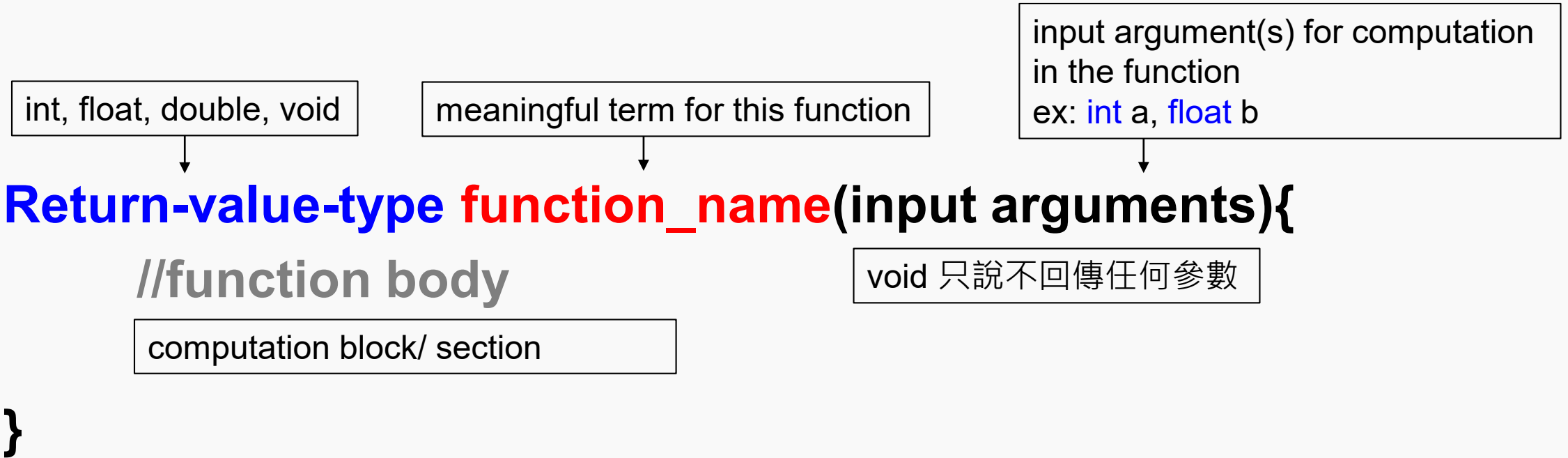
Higher reusability

Higher readability

Higher maintainability

<Define function/>

函數的結構



<Define function/>

宣告一個簡單的函數

```
int hello(void){  
    printf("Hello world!");  
    return 0;  
}
```

(1) 先宣告hello函數

```
int main(void){  
    /*Ex 3-1: Define Function */  
    /* call - hello*/  
    printf("Call hello function!\n");  
    hello();  
    return 0;  
}
```

(2) 再呼叫hello函數

Result:

```
Call hello function!  
Hello world![Finished in 148ms]
```

</Define function>

<Argument & return/>

加法函數

```
int add(int a, int b){  
    printf("%d", a+b);  
    return 0;  
}
```

(1) 先宣告add函數

```
int main(void){  
    /*Ex 3-2: Define Function */  
    /* call - add*/  
    printf("Call add function!\n");  
    add(3, 4);  
    return 0;  
}
```

(2) 再呼叫add函數

Result:

```
Call add function!  
7[Finished in 148ms]
```

<Argument & return/>

Lab 3-2

```
int add(int a, int b){
    printf("%d", a+b);
    return 0;
}

int main(void){
    /*Ex 3-2: Define Function */
    /* call - add*/
    printf("Call add function!\n");
    add(3, 4);
    return 0;
}
```

利用這樣的概念將加減乘除的函數分別宣告出來，並計算以下的問題：

- (1) $199 + 315 = ?$
- (2) $623 - 253 = ?$
- (3) $365 \times 124 = ?$
- (4) $85 \div 17 = ?$
- (5) $65 \div 30 = ?$

回傳數值

剛剛我們練習的function，結果都是在子function直接輸出。但事實上，我們再做計算的時候，大多數時候需要分階段運算，這樣才能得到我們要的結果。既然需要分階段運算，就需要回傳運算的結果，才能下個階段的運算，例如: **BMI Calculator**。

1. 分母的身高先做平方。
2. 分子再除以分母。

<Argument & return/>

回傳數值

```
float sqr(float a){  
    return a*a;  
}
```

```
float div(float a, float b){  
    return a/b;  
}
```

```
BMI Calculator!  
26.892324  
[Finished in 148ms]
```

```
int main(void){  
    /*Ex 3-3: Define Function */  
    /* call - BMI Calculator*/  
    float height = 1.67;  
    float weight = 75;  
    float BMI;  
  
    printf("BMI Calculator\n");  
    float height2 = sqr(height);  
    BMI = div(weight, height2);  
  
    printf("%f\n", BMI);  
    return 0;  
}
```

<Argument & return/>

回傳數值

Lab 3-3:

將Ex 3-3的範例修改成使用者可以自行輸入身高與體重，再計算出使用者的BMI出來，另外輸出的精度要到小數點後兩位。

範例:

```
BMI Calculator
Plz enter your height in meter?
1.67
Plz enter your weight in kg?
75
BMI: 26.89
```

Debugging Time

Lab 3-4:

請嘗試回答下列六個程式碼的輸出結果。

(a)

```
1  #include <stdio.h>
2
3  void f(){
4      printf("%d", a);
5  }
6
7  int main(){
8      int a = 25;
9      f();
10 }
```

(b)

```
1  #include <stdio.h>
2
3  void f(float a){
4      printf("%f", c);
5  }
6
7  int main(){
8      int a = 25;
9      f();
10 }
```

(c)

```
1  #include <stdio.h>
2
3  void f(){
4      printf("%d", a);
5  }
6
7  int main(){
8      int a = 25;
9      f(a);
10 }
```

Debugging Time

Lab 3-4:

請嘗試回答下列六個程式碼的輸出結果。

(d)

```
1  #include <stdio.h>
2
3  void f(int a){
4      printf("%d", a);
5  }
6
7  int main(){
8      int a = 25;
9      f(a);
10 }
```

(e)

```
1  #include <stdio.h>
2
3  void f(int a){
4      printf("%d", a+10);
5  }
6
7  int main(){
8      int a = 25;
9      f(a);
10 }
```

(f)

```
1  #include <stdio.h>
2
3  void f(int a){
4      printf("%d", a+10);
5  }
6
7  int main(){
8      int a = 25;
9      f(a*10);
10 }
```

<Inline function/>

Inline Function

```
#include <stdio.h>
```

```
int main(void){  
    /*Ex 3-4: Inline Function */  
    /* inline - hello*/  
    printf("Call hello function!\n");  
    inline int hello(void){ printf("Hello world!\n");}  
    hello();  
}
```

Inline Function

```
int main(void){  
    /*Ex 3-5: Define Function */  
    /* call - BMI Calculator*/  
    float height = 1.67;  
    float weight = 75;  
    float BMI;  
    printf("BMI Calculator\n");  
    inline float sqr(float a){return a*a;}  
    inline float div(float a, float b){return a/b;}  
    float height2 = sqr(height);  
    BMI = div(weight, height2);  
    printf("%2.2f\n", BMI);
```

}

<Inline function/>

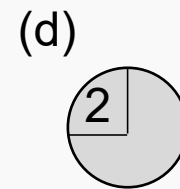
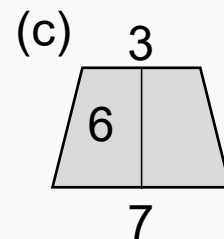
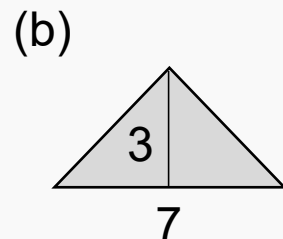
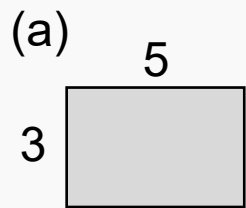
Inline Function

Lab 3-5:

請嘗試利用inline function的方式，撰寫四個函數：

- (1) 長方形計算公式
- (2) 三角形計算公式
- (3) 梯形計算公式
- (4) 圓形計算公式 ($\pi = 3.14$)。

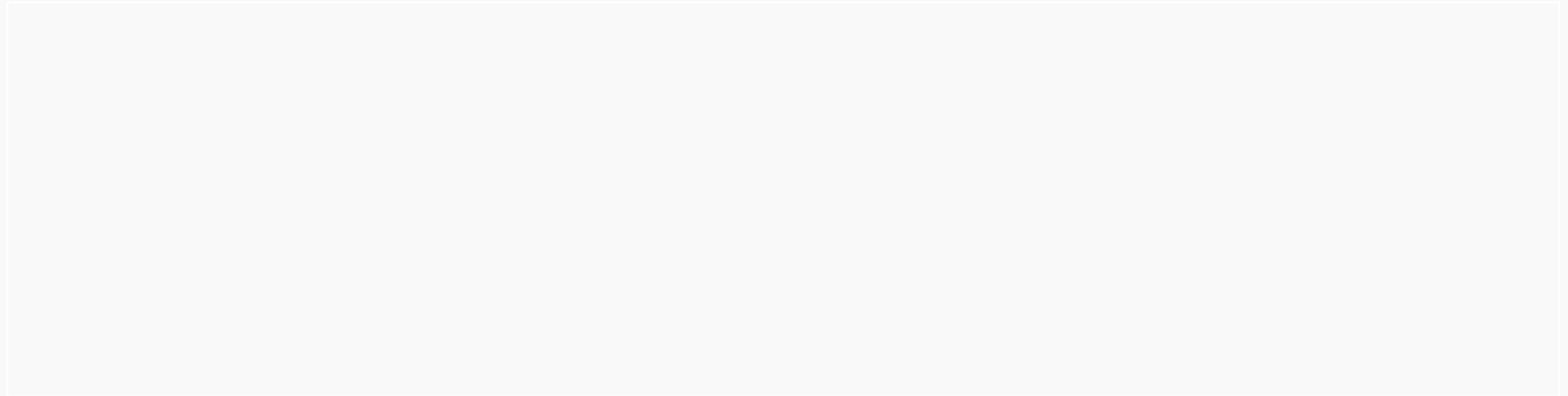
請計算出下列四個形狀的面積，並印出答案。



Inline Function

Lab 3-6:

經過以上的練習，你覺得inline function有甚麼好處 and 壞處？



作業一 Fibonacci number

相信大家對於 **Fibonacci number** 不會太陌生，但是單看這個名字可能會想不起來你/妳曾經在國高中對他恨之入骨的回憶。接下來，我們就用實例看一下這個數列長甚麼樣子吧!

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233 ...

有恢復一點記憶了嗎?

接下來用函數的方法表示:

$$F(0) = 0; F(1) = 1;$$

$$F(n) = F(n-1) + F(n-2), \text{ where } n > 1.$$

作業一 Fibonacci number

請利用你設計的兩種my_fibon()計算出下列五個fibon級數的答案。

(1) $F(1) = ?$

(2) $F(2) = ?$

(3) $F(10) = ?$

(4) $F(25) = ?$

(5) $F(40) = ?$

* For loop version and While loop version