2017

程式設計

加強班

程式設計與實習(一)

BY 孫茂勛

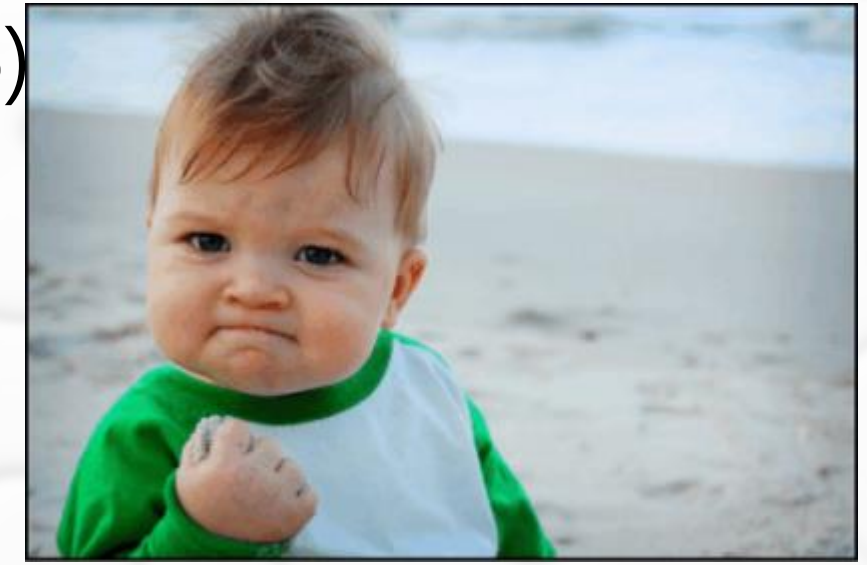
Email:JOHN85051232@GMAIL.COM

複習

Q：使用一個function，輸入一個數字，回傳一個數字的3次方？

檢討一下大家小考常犯的錯誤

- scanf
- if的判斷
- 變數的初始化
- #pragma warning(disable:4996)



- 最後就是...不要放棄...

視域、生命週期

視域(Scope)：變數可被存取的程式碼區段

生命週期(Life Cycle)：該變數何時會消滅

變數類型	生存空間	生命週期
全域變數	整個程式	整個程式
區域變數	只在該函式、方法內	只在該函式、方法內
Static靜態變數	函式、方法內	整個程式

視域、生命週期

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

int global_variable = 20;//寫在最外面的是區域變數
void test()
{
    printf("%d",local_variable);//無法判斷這個是什麼?
}
int main()
{
    int local_variable = 10;//位於main內的區域變數
    test();
    system("PAUSE");
    return 0;
}
```

視域、生命週期

比較看看兩者的差別?

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

void test()
{
    int local_variable = 10;
    printf("%d\n",local_variable);
    local_variable = local_variable + 1;
}
int main()
{
    for(int i = 0 ; i < 10 ; ++i)
    {
        test();
    }
    system("PAUSE");
    return 0;
}
```

區域變數當該區塊執行結束後便會消失。
所以資料不會保存。

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

void test()
{
    static int local_variable = 10;
    printf("%d\n",local_variable);
    local_variable = local_variable + 1;
}
int main()
{
    for(int i = 0 ; i < 10 ; ++i)
    {
        test();
    }
    system("PAUSE");
    return 0;
}
```

Static的變數只會生成一次，之後直到程式結束前都會一直存在。
所以資料可以一直保存。

視域、生命週期

當變數名稱重複時，視域小的優先。

```
#include <stdio.h>
#include <stdlib.h>
int c = 10;
int main()
{
    int c = 20;
    printf("%d\n",c);
    system("PAUSE");
    return 0;
}
```

```
20
請按任意鍵繼續 . . .
```

參考資料

視域與生存空間

- http://notepad.yehyeh.net/Content/CPP/CH01/02Variable/2.php

Define

符號常數(Define)

- 為了讀寫及修改方便經常將常數以符號常數代替。
- 通常放置在開頭。
- 用法：#define 常數名稱 常數

Define

```c
#define PI 3.14159

float area(float);
int main(){

    float radius = 2.3;

    printf("半徑為%.1f 面積為 : %.2f\n" , radius , area(radius));

    system("pause");
    return 0;
}

float area(float radius){

        float out = 0;

        out = radius*radius*PI;

        return out;

}
```

Random

rand()

- #include <stdlib.h>
- 傳回一個介於 0 到 RAND_MAX 之間的一個整數值

Q：如何產生300 – 500間的亂數？

srand()

- #include <time.h>

Random

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



int main(){


    int i = 0;
    i = rand() % 6;

    printf("i = %d\n" , i);

    system("pause");
    return 0;
}
```

Random

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>



int main(){

    srand(time(NULL));

    printf("亂數1 = %d\n" , rand());
    printf("亂數2 = %d\n" , rand());

    int a = rand()%6+1;
    printf("a = %d\n" , a);

    system("pause");
    return 0;
}

```

Recursion

遞迴(Recursion)

- 在函式中不斷地呼叫自己，直到達成終止條件才會結束。
- 需要額外堆疊空間儲存。
- 程式碼通常都很短，但通常都不好寫。
- 常看到的：階乘、費式數列、GCD、河內塔...

Recursion

階乘

- 1! = 1
- 2! = 2 * 1!
- 3! = 3 * 2!
- 4! = 4 * 3!
- …
- n! = ?

Recursion

```c
# include <stdio.h>
# include <stdlib.h>
int factirial(int num);//階乘

int main(){
    int input = 0;
    scanf("%d",&input);
    for(int i = 0 ; i < input ; i++)
    {
        printf("%d\n",factirial(i));
    }
    return 0;
}
int factirial(int num)
{
    if(num <= 1)
    {
        return 1;
    }
    else
    {
        return num * factirial(num-1);
    }
}
```

Recursion

```
5  long factirial(int number);//階層
6  int main(void) {
7
8
9     int num = 0;
10    scanf("%d" , &num);
11
12    printf("%d! = %ld\n" ,num ,  factirial(num));
13   system("pause");
14   return 0;
15 }
16
17 long factirial(int number){
18
19       if(number <= 1){
20                 return 1;
21       }
22       else{
23            return (number * factirial(number-1));
24       }
25
26 }
```

Recursion

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

int fib(int num)
{
    if(num <= 1)
    {
        return num;
    }
    else
    {
        return fib(num-1) + fib(num-2);
    }
}
int main()
{
    for(int i = 1 ; i <= 10 ; ++i)
    {
        printf("%d ",fib(i));
    }
    return 0;
}
```

Recursion

```
3
4  int gcd(int, int); //最大公因數
5  int main(void) {
6      int m = 0;
7      int n = 0;
8
9      printf("輸入兩數(num1 num2)：");
10     scanf("%d %d", &m, &n);
11
12     printf("GCD: %d\n", gcd(m, n));
13
14     system("pause");
15     return 0;
16 }
17 int gcd(int m, int n) {
18     if(m % n == 0) {
19         return n;
20     }
21     else {
22         return gcd(n, m % n);
23     }
24 }
25
```

函式與電腦記憶體之間的關係(CH5-7)

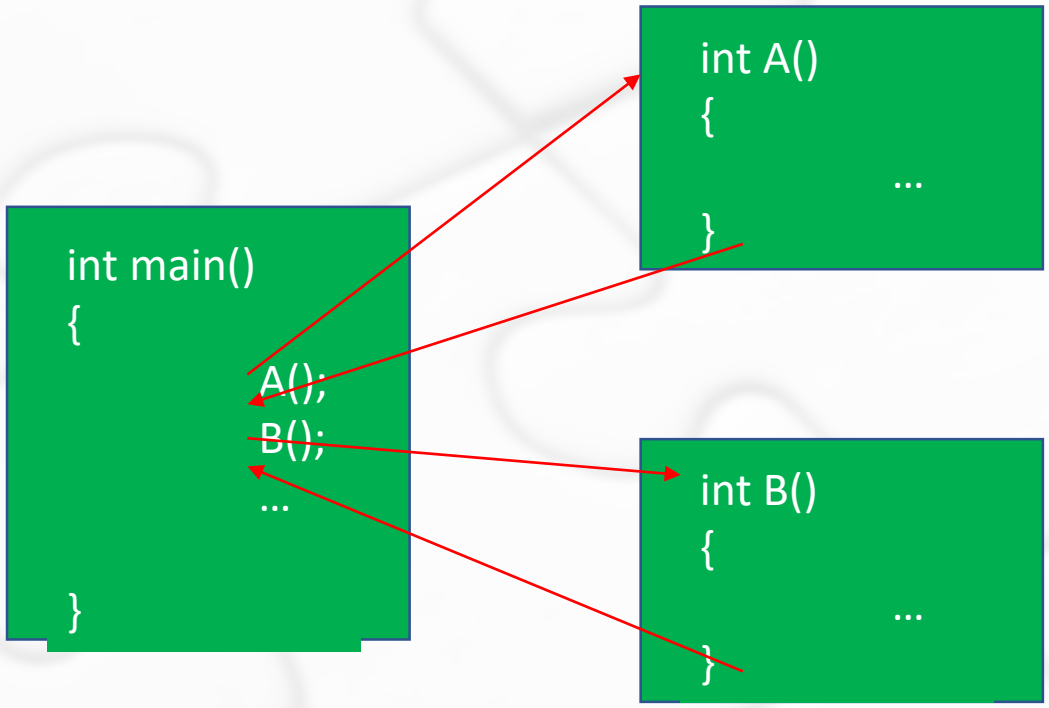
Stack

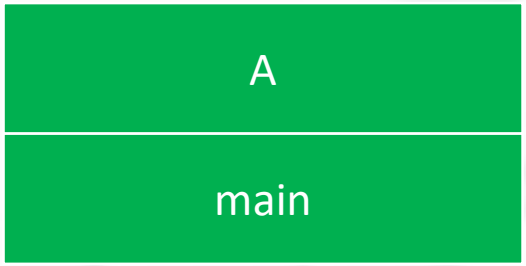
- 一種資料結構
- Last In, First Out(LIFO)
- Ex：疊盤子的時候，一定是從最上層(最後放的)開始拿
- 程式的呼叫順序就是一種Stack
- 遞迴的原理也是Stack

函式與電腦記憶體之間的關係

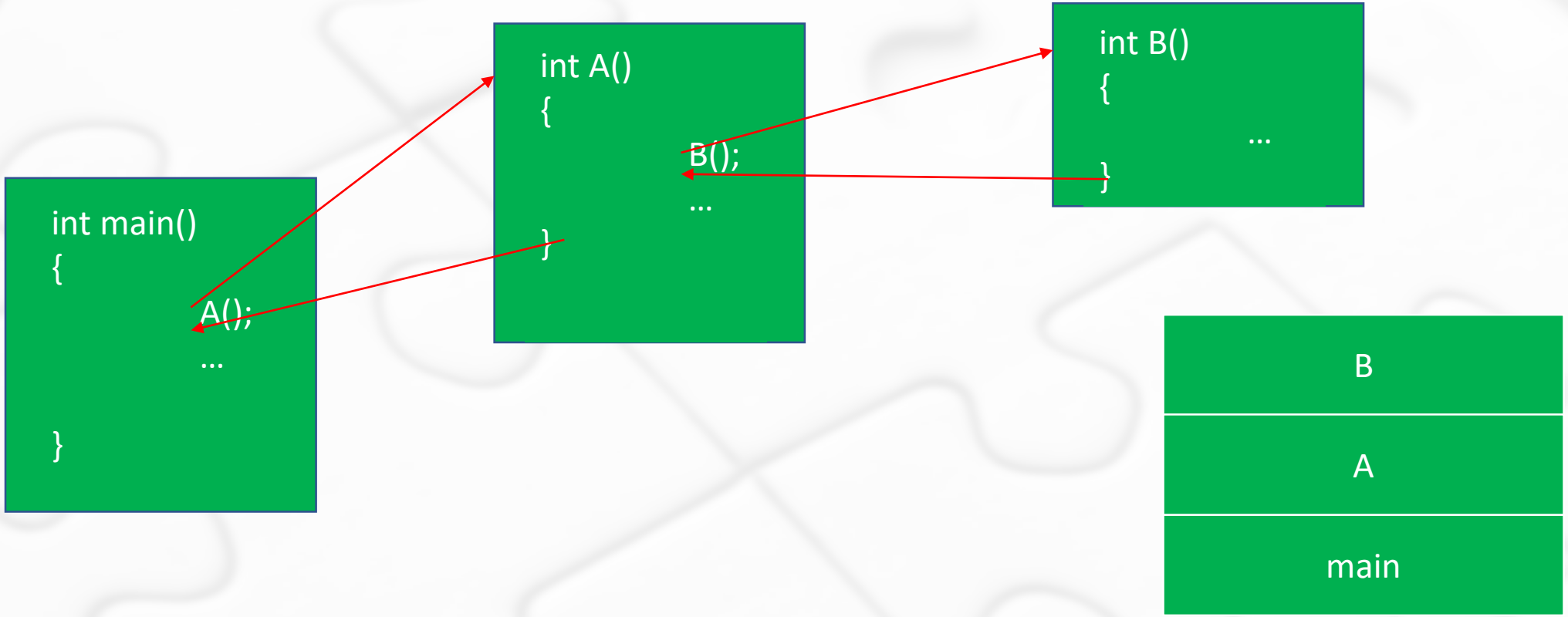
- 程式所佔的記憶體在電腦中是以Stack的形式儲存的
- 比較晚呼叫的會比較晚放在Stack內，但會先被執行
- 執行完後會將Function從Stack中移除

程式的執行順序





程式的執行順序



Passing Arguments By Value and Passing By Reference(CH5-9)

- Function傳參數的兩種方式
- 之後再介紹(如果期中前還沒介紹就不會考)

THANK YOU