# 第二章：C++的函數與變數

Su - Huang

# OUTLINE

1. 函數介紹
2. 區域變數與全域變數
3. 函數的傳值
4. Practice

# 函數介紹

# 簡單的函數 (1/2)

- 下面的範例計算6的平方值，並在運算結果前後列印星號

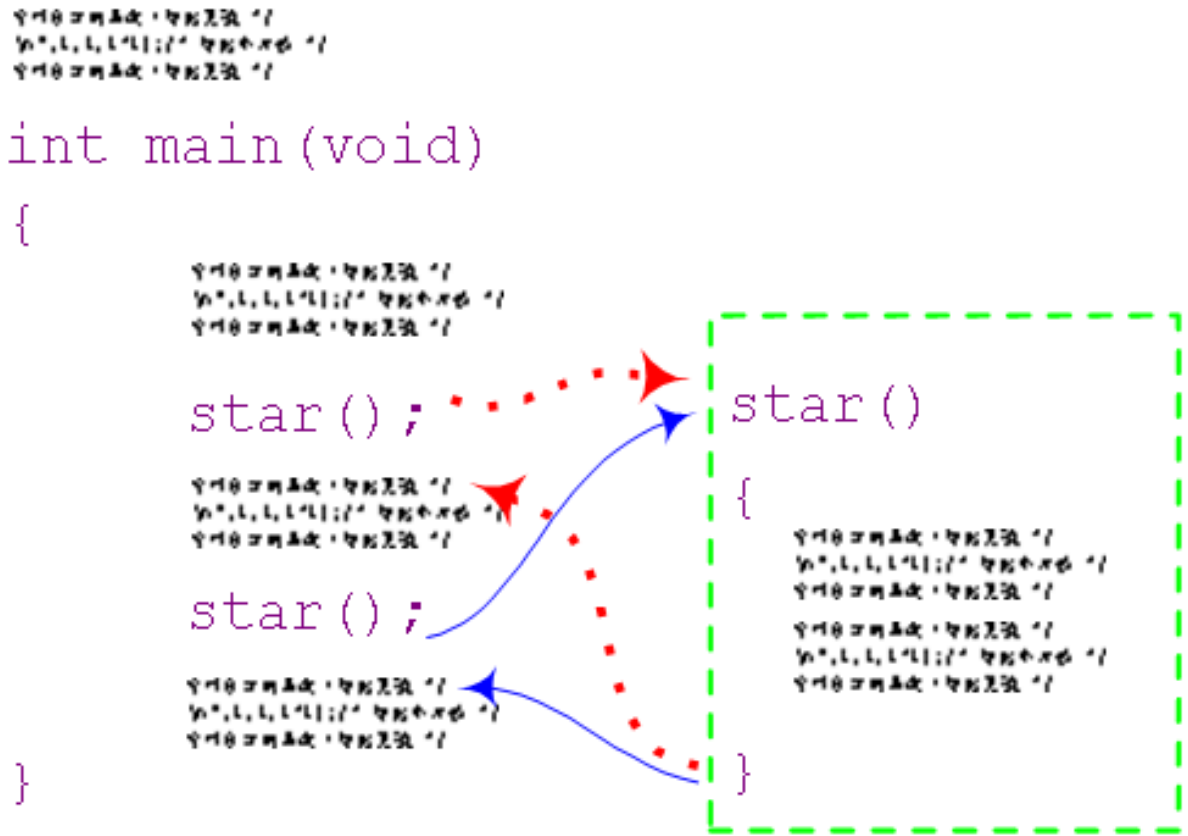
```
01  // prog6_1, 簡單的函數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  void star(void);                          // 函數原型的宣告
06  int main(void)
07  {
08     star();                                // 呼叫自訂的函數，印出星號
09     cout << "6*6=" << 6*6 << endl;          // 印出 6 的平方值
10     star();                                // 呼叫自訂的函數，印出星號
11     system("pause");
12     return 0;
13  }
14
15  void star(void)                           // 自訂的函數 star()
16  {
17     int j;
18     for(j=1;j<=8;j++)
19        cout << "*";                        // 印出*星號
20     cout << endl;
21     return;
22  }
```

```
/* prog6_1 OUTPUT---
********
6*6=36
********

--------------------*/
```

# 簡單的函數 (2/2)

- 下圖說明函數呼叫與返回的方式：

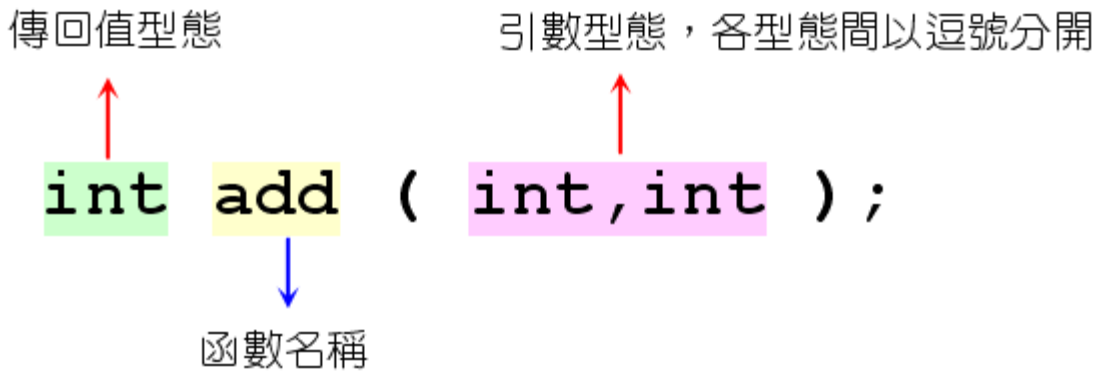


# 函數原型的宣告、撰寫與呼叫(1/3)

- 下面為「函數原型」（prototype）的宣告格式

```
傳回值型態 函數名稱(引數型態 1, 引數型態 2,...);
```

- 下面的格式為合法的函數宣告格式



# 函數原型的宣告、撰寫與呼叫(2/3)

- 自訂函數撰寫的格式如下所示

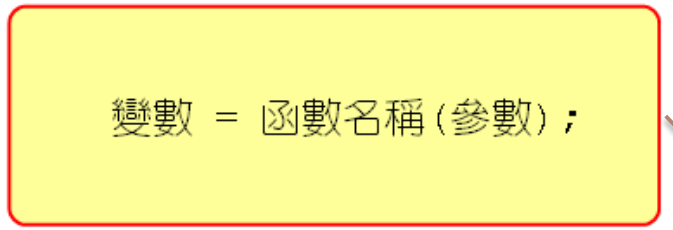
```
傳回值型態 函數名稱(型態 1 引數 1，…，型態 n 引數 n)
{
    變數宣告;
    敘述主體;
    return 運算式;
}
```

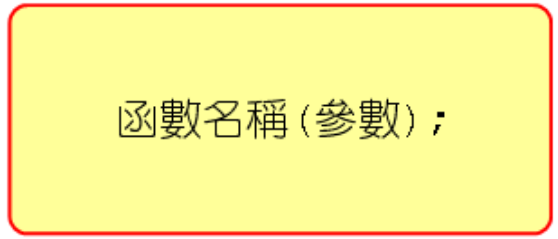
EX:

```
int plus ( int A, int B )        //函數名稱
{
        int C;                   //變數宣告
        C=A+B;                   //敘述主體
        return C;                //回傳值
}
```

# 函數原型的宣告、撰寫與呼叫(2/3)

- 呼叫函數的方式有兩種
  - 一種是將傳回值指定給某個變數接收
  - 另一種則是直接呼叫函數，不需要傳回值




```
int with_return(int i){
        return(i);
}
void without_return(int i){
        std::cout<<i;
}
void main(){
        //有回傳值
        int A=with_return(1);
        std::cout<<A;
        //無回傳值
        without_return(1);
}
```

# 函數原型的宣告、撰寫與呼叫(3/3)

- 下面的敘述為常見的函數呼叫

```
i=func();    // 呼叫 func()函數,並將傳回值給 i 存放
star();      // 直接呼叫 star()函數,沒有傳回值
myfunc(4);   // 呼叫 myfunc()函數,並將引數 4 傳入函數中
```

- 右邊的格式為自訂函數 **square()** 的宣告與呼叫方式
- 宣告於函數內的變數稱為「區域變數」（**local variable**）

```
   ...
int square(int);          → 自訂函數的宣告
   ...
int main(void)            → 主函數
{
   ...
   j=square(i);           → 自訂函數的呼叫
   ...
}
   ...
int square(int i)
{
   int squ;
   squ=i*i;               → 自訂函數的內容
   return squ;
}
```

# 不使用函數原型的方式 (1/2)

- 如果不使用函數原型，可採下面的寫法

```
int square(int i)
{
    int squ;
    squ=i*i;
    return squ;
}
```

自訂函數的定義與宣告

```
    ...
int main(void)
{
    ...
    j=square(i);
    ...
}
    ...
```

主函數

自訂函數的呼叫



# 不使用函數原型的方式 (2/2)

- 下面的程式是不使用函數原型的方式所撰寫而成的

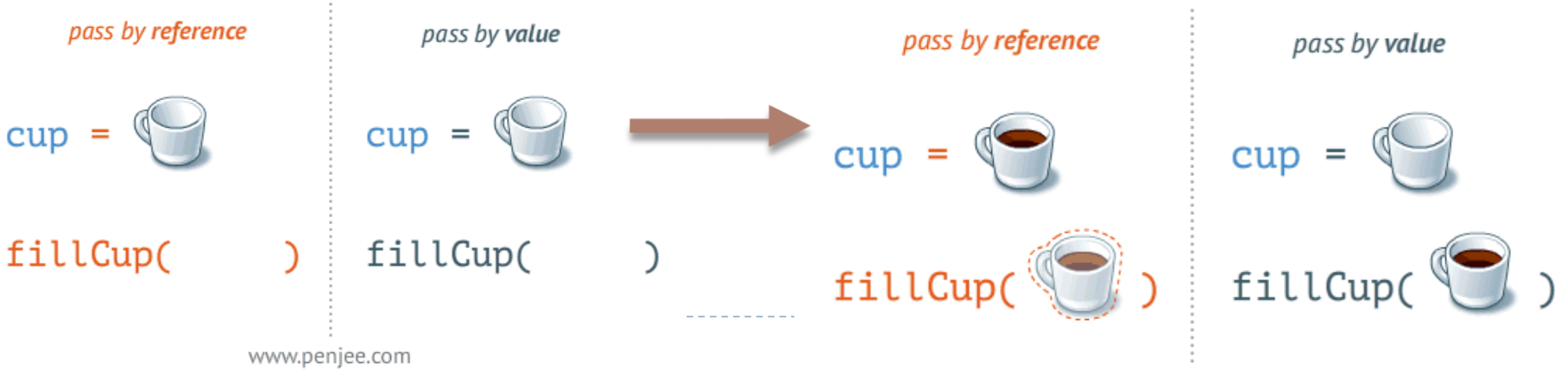
```
01  // prog6_2, 不使用函數原型的方式
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int square(int a)    // 自訂的函數 square(), 計算平方值
06  {
07     int squ;
08     squ=a*a;
09     return squ;
10  }
11
12  int main(void)       // 主程式
13  {
14     cout << "square(6)=" << square(6) << endl; // 印出 square(6)的值
15     system("pause");
16     return 0;
17  }
```

```
/* prog6_2 OUTPUT---
square(6)=36
--------------------*/
```



# 函數的引數與參數 (1/2)

- 傳遞給函數的資料稱為函數的「引數」（argument）。
- 函數所收到的資料稱為「參數」（parameter）。
- 傳址呼叫，call by reference
- 是指呼叫函數時，所傳遞的資料是某個變數的位址。
- 傳值呼叫，call by value
- 將資料的值當做引數來傳遞給函數



# 函數的引數與參數 (2/2)

- 下面是傳入兩個引數的例子

```
01  // prog6_3, 呼叫自訂函數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  void func(int,int);              // 函數原型的宣告
06  int main(void)
07  {
08     int a=3,b=6;
09     cout << "In main(),a=" << a << ",b=" << b << endl;  // 印出 a,b 的值
10     func(a,b);
11     cout << "After func(),a=" << a << ",b=" << b << endl;
12
13     system("pause");
14     return 0;
15  }
16
17  void func(int a,int b)           // 自訂的函數 func()，印出 a,b 的值
18  {
19     a+=10;
20     b+=10;
21     cout << "In func(),a=" << a << ",b=" << b << endl;
22     return;
23  }
```

```
/* prog6_3 OUTPUT----

In main(),a=3,b=6
In func(),a=13,b=16
After func(),a=3,b=6

--------------------*/
```

# 函數的傳回值 (1/3)

- return敘述的格式如下所示

```
return 運算式;
```

- 函數的傳回值可以是變數、常數或是運算式
- 函數沒有傳回值時，可以在函數結束的地方加上分號

```
return;
```

# 函數的傳回值 (2/3)

- 下面的程式可以利用函數傳回兩個整數的較大值

```
01  // prog6_4, 傳回較大值
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int max(int,int);          // 函數原型的宣告
06  int main(void)
07  {
08     int a=12,b=35;
09     cout << "a=" << a << ", b=" << b << endl;     // 印出 a,b 的值
10     cout << "The larger number is " << max(a,b) << endl; // 印出較大值
11     system("pause");
12     return 0;
13  }
14
15  int max(int i,int j)       // 自訂的函數 max(),傳回較大值
16  {
17     if (i>j)
18       return i;
19     else
20       return j;
21  }
```

```
/* prog6_4 OUTPUT------

a=12, b=35
The larger number is 35

----------------------*/
```

# 函數的傳回值 (3/3)

- prog6_5是沒有傳回值的函數之範例

```
01  // prog6_5, 沒有傳回值的函數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  void myprint(int,char);          // 函數原型的宣告
06  int main(void)
07  {
08     int a=6;
09     char ch='%';
10     myprint(a,ch);                // 呼叫自訂的函數，印出 a 個字元
11     cout << "Printed!!" << endl;
12     system("pause");
13     return 0;
14  }
15
16  void myprint(int n,char c)       // 自訂的函數 myprint()
17  {
18     int i;
19     for(i=1;i<=n;i++)
20        cout << c;                 // 印出字元
21     cout << endl;
22     return;
23  }
```

```
/* prog6_5 OUTPUT---

%%%%%%
Printed!!

--------------------*/
```

# 函數之間的相互呼叫 (1/2)

- 下面的程式碼是在函數間呼叫其它函數的例子

```
01  // prog6_13, 相互呼叫函數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  void sum(int),fact(int);
06  int main(void)
07  {
08     int a=5;
09     fact(a);
10     sum(a+5);
11     system("pause");
12     return 0;
13  }
14
```

```
/* prog6_13 OUTPUT-----

1*2*...*5=120
1+2+...+5=15      → 由 fact()函數呼叫的 sum()函數
1+2+...+10=55     → 由 main()函數呼叫的 sum()函數

----------------------*/
```

# 函數之間的相互呼叫 (2/2)

```
15  void fact(int a)            // 自訂函數 fact()，計算 a!
16  {
17     int i,total=1;
18     for(i=1;i<=a;i++)
19        total*=i;
20     cout << "1*2*...*" << a << "=" << total << endl;//印出 a!的結果
21     sum(a);
22     return;
23  }
24
25  void sum(int a)             // 自訂函數 sum()，計算 1+2+...+a 的結果
26  {
27     int i,sum=0;
28     for(i=1;i<=a;i++)
29        sum+=i;
30     cout << "1+2+...+" << a << "=" << sum << endl; //印出計算結果
31     return;
32  }
```

# OUTLINE

1. 函數介紹
2. 區域變數與全域變數
3. 函數的傳值
4. Practice

# 區域變數與全域變數

# 變數的等級

- C++提供auto、static auto、extern、static extern及register等五種變數等級
- 宣告變數時，可以一起將變數名稱及其等級同時宣告，如下面的敘述：

```
auto int i;              // 宣告區域整數變數 i
extern char ch;          // 宣告外部字元變數 ch
static float f;          // 宣告靜態浮點數變數 f
```

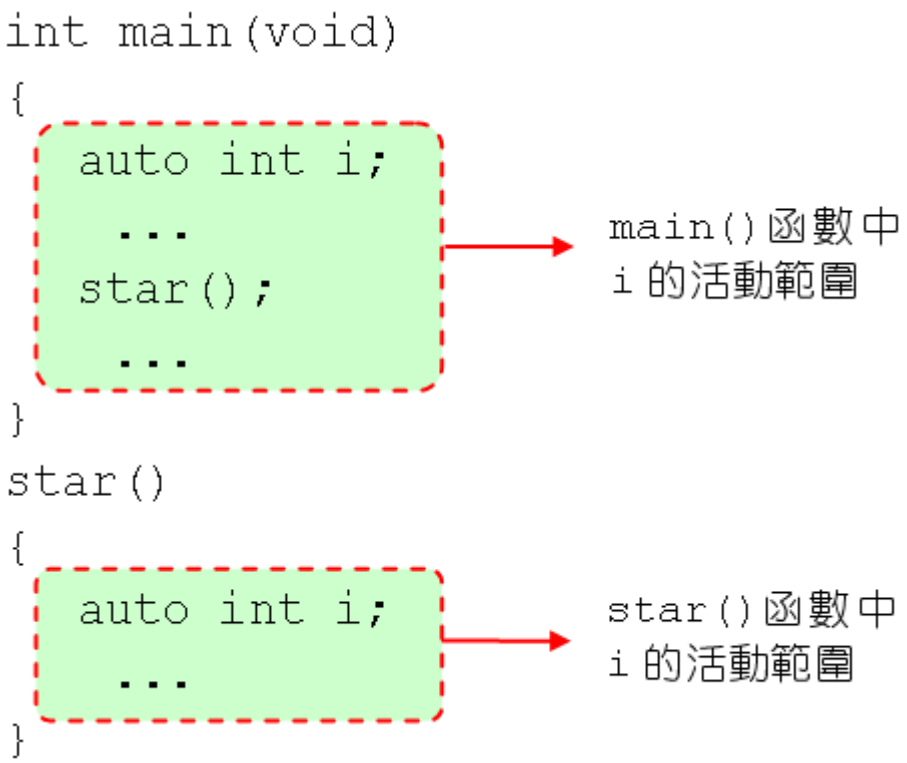
# 區域變數 (1/3)

- 區域變數又稱為「自動變數」（automatic variable）
  - 我們平常所宣告的就是 auto 只是通常不會去打。
- 離開包含區域變數的程式碼區塊後，區域變數的值自動消失
- 區域變數在程式執行時會以堆疊（stack）的方式存放，屬於動態的變數
- 下面的宣告皆是屬於區域變數的一種：

```
auto int i;     // 宣告區域整數變數 i
char ch;        // 宣告區域字元變數 ch（省略關鍵字 auto）
```

# 區域變數 (2/3)

- 下圖是區域變數i在所屬區段中的活動範圍之示意圖



# 區域變數 (3/3)

- 由下面的程式裡可以看到區域變數的使用

```
01  // prog6_7, 區域變數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  void func(void);                              // 函數原型的宣告
06  int main(void)
07  {
08     auto int a=10;
09     cout << "In Main(),a=" << a << endl;       // 印出 main() 中 a 的值
10     func();                                    // 呼叫自訂的函數
11     cout << "In Main(),a=" << a << endl;       // 印出 a 的值
12     system("pause");
13     return 0;
14  }
15
16  void func(void)                               // 自訂的函數 func()
17  {
18     int a=30;
19     cout << "In func(),a=" << a << endl;       // 印出 func() 中 a 的值
20     return;
21  }
```

```
/* prog6_7 OUTPUT---
In Main(),a=10
In func(),a=30
In Main(),a=10
-------------------*/
```

# 靜態區域變數 (1/2)

- 靜態區域變數是在編譯時就已配置固定的記憶體空間
- 包含靜態區域變數的程式碼區塊執行完後，靜態區域變數的值不會自動消失
- 下面的敘述為靜態區域變數的範例

```
static float f;     // 定義靜態區域浮點數變數 f
```

# 靜態區域變數 (2/2)

- 由下面的程式裡可以看到靜態區域變數a的變化

```
01  // prog6_8, 靜態區域變數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  void func(void);      // 函數原型的宣告
06  int main(void)
07  {
08     func();            // 呼叫自訂的函數
09     func();
10     func();
11     system("pause");
12     return 0;
13  }
14
15  void func(void)       // 自訂的函數 func()
16  {
17     static int a=10;
18     cout << "In func(),a=" << a << endl;   // 印出 func()中 a 的值
19     a+=20;
20     return;
21  }
```

此行僅在初始化時進行

```
/* prog6_8 OUTPUT---

In func(),a=10
In func(),a=30
In func(),a=50

--------------------*/
```

# 外部變數 (1/4)

- 外部變數（external variable）是在函數外面所宣告的變數
- 外部變數又稱為「總體變數」或「全域變數」(global Variable)
- 下面的程式片段是外部變數的宣告範例

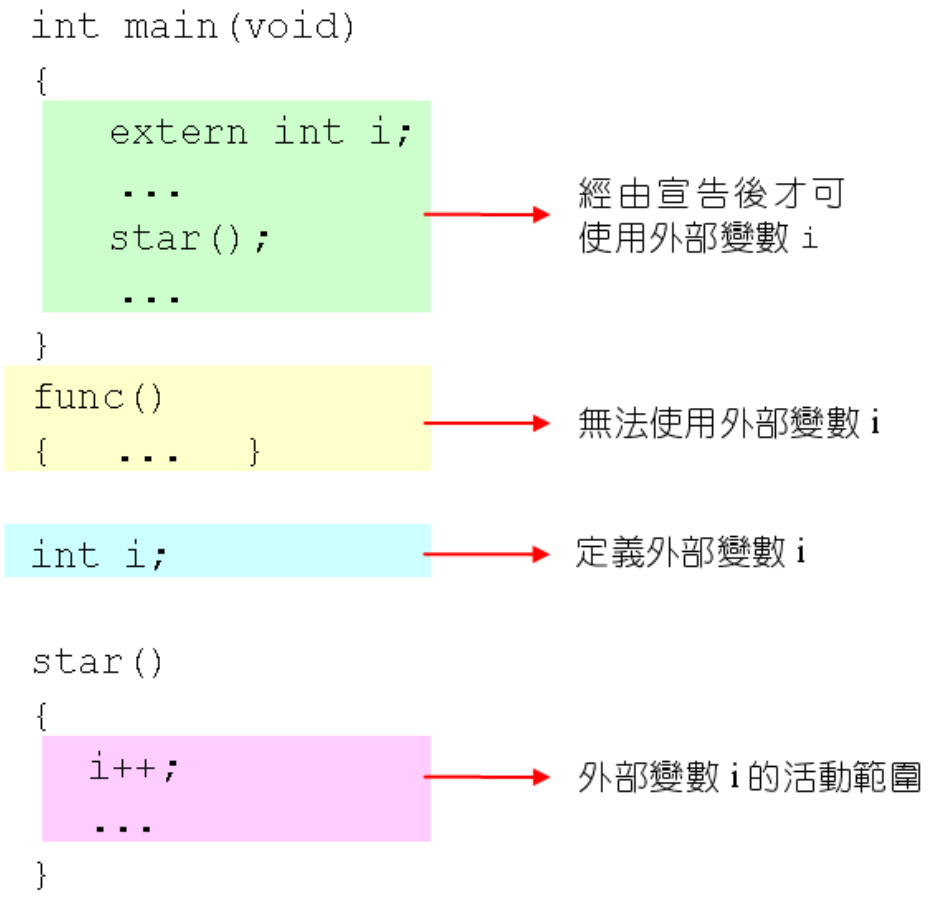
```
int main(void)
{
    extern int a;
    ...
}
 int a;
func()
{   ...   }
```

宣告 a 為外部整數變數，即可拓展外部變數 a 的活動範圍

定義 a 為外部整數變數

# 外部變數 (2/4)

- 從下圖的內容中可以看到外部變數i的活動範圍



# 外部變數 (3/4)

- 下面的程式定義外部變數pi，利用它求取圓周及圓面積

```
01  // prog6_9, 外部變數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  void peri(double),area(double);     // 函數原型的宣告
06  int main(void)
07  {
08     extern double pi;                // 定義外部變數 pi
09     double r=1.0;
10     cout << "pi=" << pi << endl;
11     cout << "radius=" << r << endl;
12     peri(r);                         // 呼叫自訂的函數
13     area(r);
14     system("pause");
15     return 0;
16  }
```

```
/* prog6_9 OUTPUT-----
pi=3.14
radius=1
peripheral length=6.28
area=3.14
----------------------*/
```

# 外部變數 (4/4)

```
17  double pi=3.14;                          // 外部變數 pi 設值為 3.14
18  void peri(double r)                      // 自訂的函數 peri()，印出圓周
19  {
20     cout << "peripheral length=" << 2*pi*r << endl;
21     return;
22  }
23
24  void area(double r)                      // 自訂的函數 area()，印出圓面積
25  {
26     cout << "area=" << pi*r*r << endl;
27     return;
28  }
```

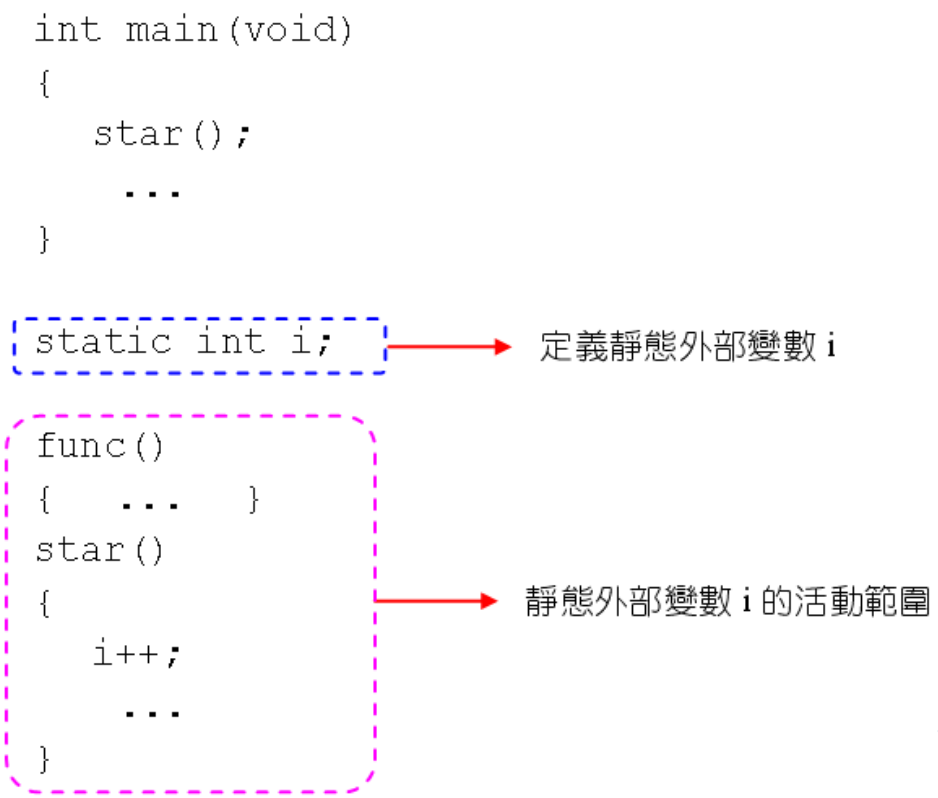
```
/* prog6_9 OUTPUT-----

pi=3.14
radius=1
peripheral length=6.28
area=3.14

----------------------*/
```

# 靜態外部變數 (1/2)

- 靜態外部變數只能在一個程式檔內使用
- 下圖為靜態外部變數i的活動範圍



# 靜態外部變數 (2/2)

- 下面的程式可以認識靜態外部變數的生命週期與活動範圍

```
01  // prog6_10, 靜態外部變數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  static int a;                    // 定義靜態外部整數變數 a
06  void odd(void);                  // 函數原型的宣告
07  int main(void)
08  {
09     odd();                        // 呼叫 odd()函數
10     cout << "after odd(), a=" << a << endl;
11     system("pause");
12     return 0;
13  }
14
15  void odd(void)                   // 自訂函數 odd()，判斷 a 為奇數或是偶數
16  {
17     a=10;
18     if(a%2==1)
19        cout << "a=" << a << ", a 是奇數" << endl;      // 印出 a 為奇數
20     else
21        cout << "a=" << a << ", a 是偶數" << endl;      // 印出 a 為偶數
22     return;
23  }
```

```
/* prog6_10 OUTPUT---

a=10, a 是偶數
after odd(), a=10

----------------------*/
```

# OUTLINE

1. 函數介紹
2. 區域變數與全域變數
3. 函數的傳值
4. Practice

# 函數的傳值

# 函數的傳值 (1/2)

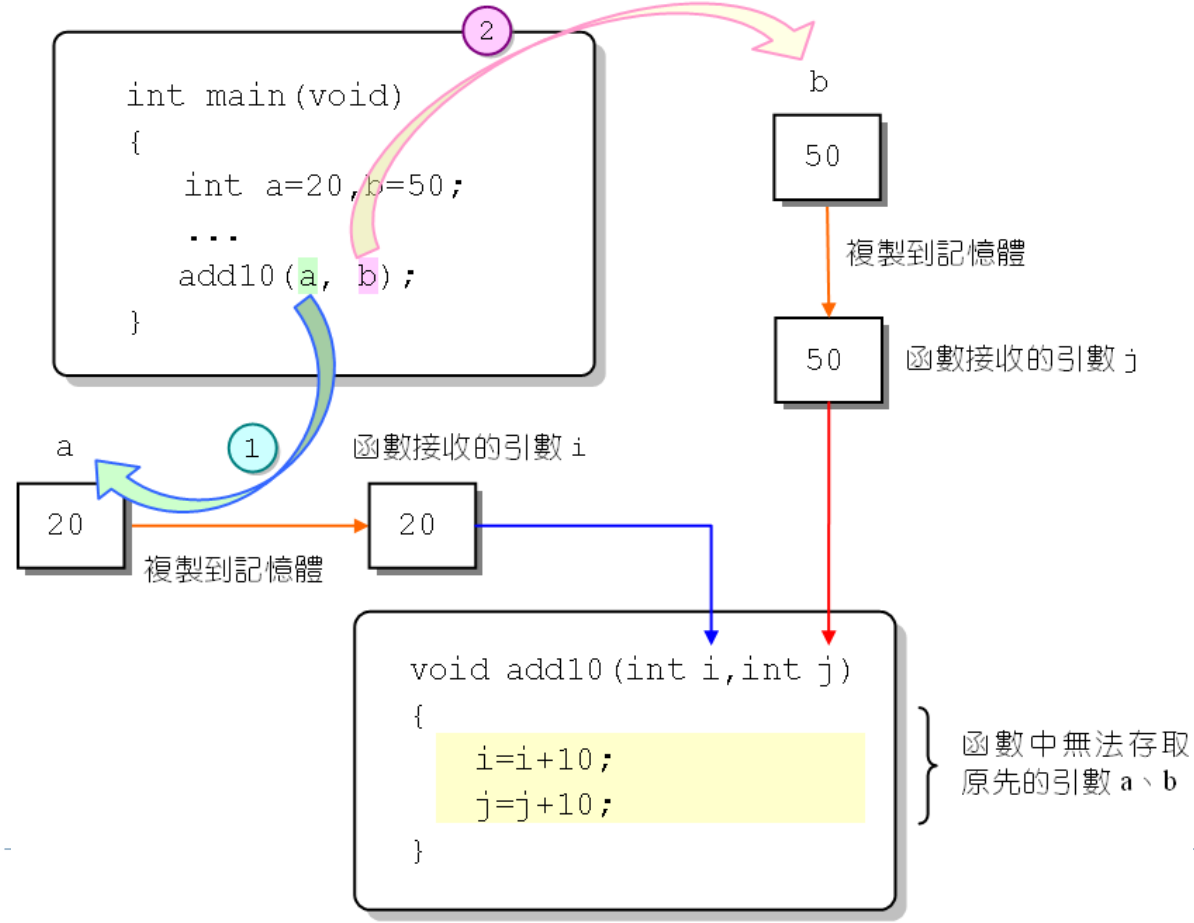
- 下面的程式可用來觀察函數裡，變數值的變化情形

```
01  // prog7_1, 函數的傳值
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  void add10(int,int);
06  int main(void)
07  {
08     int a=20,b=50;
09     cout << "before calling add10(): ";
10     cout << "a=" << a << ", b=" << b << endl;  // 印出 a、b 的值
11     add10(a,b);
12     cout << "after called add10(): ";
13     cout << "a=" << a << ", b=" << b << endl;  // 印出 a、b 的值
14     system("pause");
15     return 0;
16  }
17
18  void add10(int i,int j)
19  {
20     i=i+10;
21     j=j+10;
22  }
```

```
/* prog7_1 OUTPUT----------------
before calling add10(): a=20, b=50
after called add10(): a=20, b=50
--------------------------------*/
```

# 函數的傳值 (2/2)

- 以prog7_1為例，將函數傳值呼叫的方式繪製成圖



# 參照的基本認識 (1/3)

- C++提供參照（reference）來做為資料的別名
- 參照的效果與使用指標一樣，都會更動到原本資料
- 參照與指標的差別，在於參照使用起來與一般資料一樣，較為直覺，且在宣告的時候就要給定初值
- 參照的宣告格式如下

```
資料型態 變數名稱;
資料型態 &參照名稱=變數名稱;
```



# 參照的基本認識 (2/3)

- 想為整數變數a使用參照ref，可以做出如下的宣告：

```
int a;              // 宣告整數變數 a
int &ref=a;         // 宣告變數 a 的參照 ref，並使 ref 指向變數 a
```

- 如果想將ref的值設成10，可以寫出下面的敘述：

```
int& ref=a;         // 宣告 ref 為變數 a 的參照
```

# 參照的基本認識 (3/3)

- 下面的程式碼是參照的使用範例常用的流程圖符號

```
01  // prog7_2, 參照的認識
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int num=5;
08     int &rm=num;                    // 宣告 rm 為 num 的參照
09
10     rm=rm+10;                       // 參照 rm 加 10
11     cout << "num=" << num << endl;  // 印出 num 的值
12     cout << "rm=" << rm << endl;    // 印出 rm 的值
13     system("pause");
14     return 0;
15  }
```

```
/* prog7_2 OUTPUT---
num=15
rm=15
-------------------*/
```

# 傳遞參照到函數 (1/4)

- 下面是將參照當成引數傳入函數的原型宣告

```
int func(int &,char &);        // 將參照當成引數傳入函數的函數原型之宣告
```

- 在定義函數時，於變數名稱前加上參照運算子&即可

```
int func(int &ref1,char &ref2)    // 將參照當成引數傳入函數的函數之定義
{
   ...
}
```

# 傳遞參照到函數 (2/4)

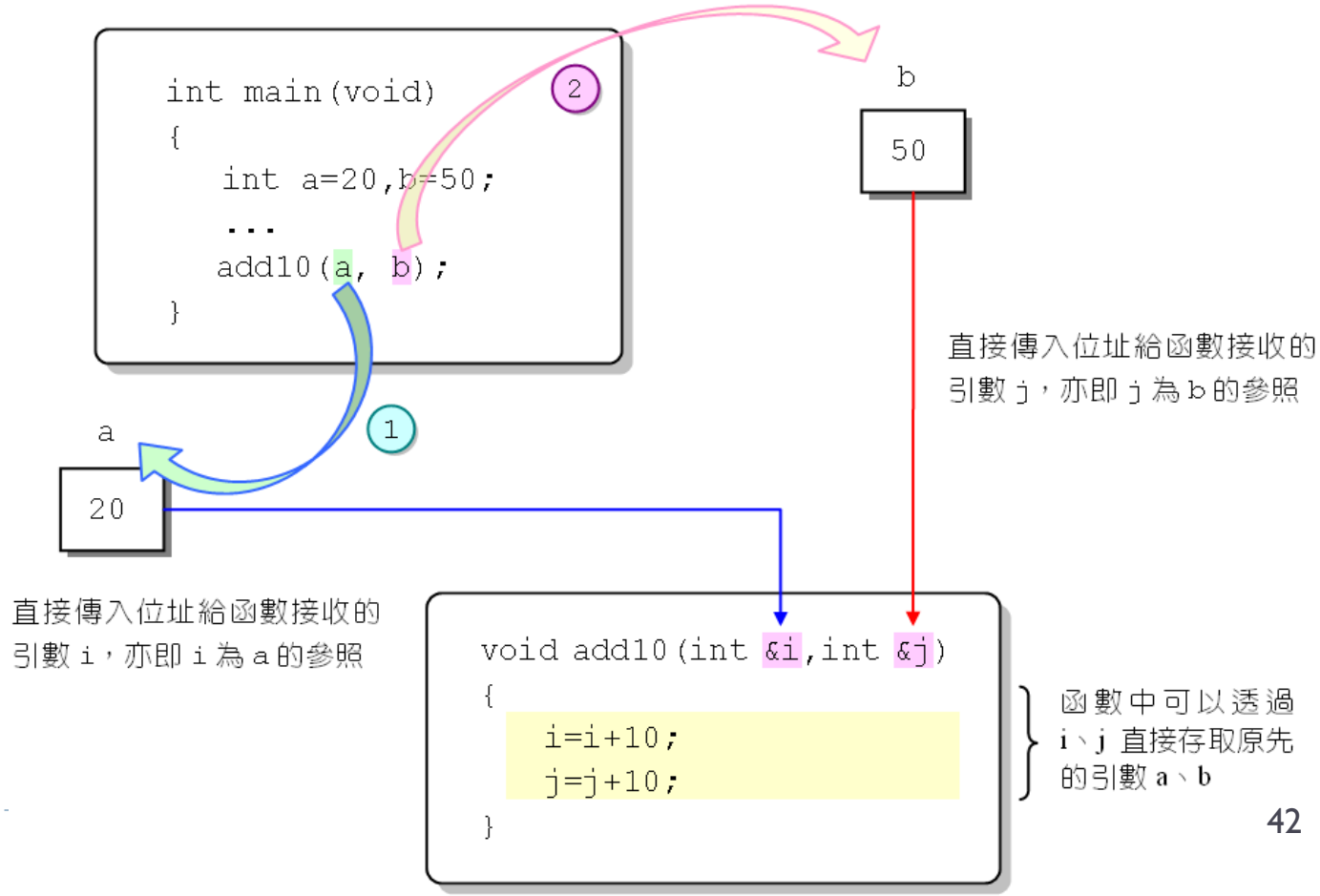
```
01  // prog7_3, 傳參照到函數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  void add10(int &,int &);
06  int main(void)
07  {
08     int a=20,b=50;
09     cout << "before calling add10(): ";
10     cout << "a=" << a << ", b=" << b << endl;  // 印出 a、b 的值
11     add10(a,b);
12     cout << "after called add10(): ";
13     cout << "a=" << a << ", b=" << b << endl;  // 印出 a、b 的值
14     system("pause");
15     return 0;
16  }
17
18  void add10(int &i,int &j)
19  {
20     i=i+10;
21     j=j+10;
22     return;
23  }
```

- prog7_3是以參照的方式傳遞到函數

```
/* prog7_3 OUTPUT----------------
before calling add10(): a=20, b=50
after called add10(): a=30, b=60
---------------------------------*/
```

# 傳遞參照到函數 (3/4)

- 下圖是以prog7_3為例，說明參照呼叫的方式



# 傳遞參照到函數 (4/4)

- 下面的程式是利用print() 函數，印出欲列印的字元

```
01  // prog7_4, 參照的傳遞
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  void print(char,int &);
06  int main(void)
07  {
08     int i,count=0;
09     for(i=0;i<3;i++)
10        print('*',count);
11     cout << endl;
12     for(i=0;i<5;i++)
13        print('$',count);
14     cout << endl;
15     cout << "print() function is called " << count << " times.";
16     cout << endl;
17     system("pause");
18     return 0;
19  }
20
21  void print(char ch, int& cnt)          // 自訂函數print()
22  {
23     cout << ch;
24     cnt++;
25     return;
26  }
```

```
/* prog7_4 OUTPUT------------------
***
$$$$$
print() function is called 8 times.
-----------------------------------*/
```

# 傳回值為參照的函數 (1/2)

- 函數的傳回值也可以是參照
- 舉例來說，於程式中宣告一名為max的函數，可傳回兩個整數中較大值之參照，函數原型為：

- 想將傳回的參照值設為100，即可寫出下面的敘述：

```
int &max(int &,int &);
max(i,j)=100;
```

# 傳回值為參照的函數 (2/2)

- 下面是函數傳回參照的使用範例

```
01  // prog7_5, 傳回值為參照
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int &max(int &,int &);          // 宣告函數原型，其傳回值為參照
06  int main(void)
07  {
08     int i=10,j=20;
09     max(i,j)=100;                // 將 max()函數傳回的參照值重設為 100
10     cout << "i=" << i << ",j=" << j << endl;
11     system("pause");
12     return 0;
13  }
14
15  int &max(int &a,int &b)
16  {
17     if(a>b)
18        return a;
19     else
20        return b;
21  }
```

```
/* prog7_5 OUTPUT---
i=10,j=100
--------------------*/
```

# OUTLINE

1. 函數介紹
2. 區域變數與全域變數
3. 函數的傳值
4. Practice

# Practice

# Practice 1

- 今年 2020 年的每個月的天數如下：

| 一月 | 二月 | 三月 | 四月 | 五月 | 六月 | 七月 | 八月 | 九月 | 十月 | 十一月 | 十二月 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 29 | 31 | 30 | 31 | 30 | 31 | 31 | 30 | 31 | 30 | 31 |

- 請寫一個函數 int fun(int ,int) ，輸入月份及日期，回傳值為今年的第幾天。請於主程式中將答案印出。

(注意!使用 call by value)

INPUT :

日期 (MM DD)

OUTPUT:

第幾天

SAMPLE INPUT :

01 01

02 12

SAMPLE OUTPUT:

1

43

# Practice 2

- 試利用 void sum(int &, int &) 函數，傳入 a, b 兩個整數，於函數中計算 a+b 的值，並將計算結果存入 a 中。a 與 b 的值請自行設定。

(注意! 請使用call by reference 今天上課教的內容)

Hint : long long int (一種data type)

INPUT :

A,B 兩個正數 小於2147483647 大於-2147483647

OUTPUT:

A+B

SAMPLE INPUT :

12 48

-20 5

SAMPLE OUTPUT:

60

-15

# Practice 3

- 試寫一個函數，將引數 a, b 以大小排列。其函數原形為 void sort (int &, int &);

(注意! 請使用call by reference 今天上課教的內容)

INPUT :

A,B 兩個正數 小於2147483648 大於-2147483647

OUTPUT:

比較大的數 比較小的數

SAMPLE INPUT :

12 48

-20 5

SAMPLE OUTPUT:

48 12

5 -20