# 6-2 在函數中使用函數

## 相同名稱的函數

原則上函數的名稱不能重覆，但是只要其參數列不同，就可以使用相同的名稱。

以下面的程式為例，我們可以觀察到叫用函數時，編譯器會檢查函數名稱和參數列數量和型別。

```
#include <iostream>

using namespace std;

// 回傳 2 整數中的最小值
int MIN(int a, int b)
{
    cout << "回傳 2 整數中的最小值" << endl;
    if(a<=b)
        return a;
    else
        return b;
}

// 回傳 2 浮點數中的最小值
double MIN(double a, double b)
{
    cout << "回傳 2 浮點數中的最小值" << endl;
    if(a<=b)
        return a;
    else
        return b;
}

// 回傳 3 整數中的最小值
int MIN(int a, int b, int c)
{
    cout << "回傳 3 整數中的最小值" << endl;
    if(a<=b && a<=c)
        return a;
    else if(b<=a && b<=c)
        return b;
    else
        return c;
}
```

```
int main()
{
    int x=2, y=5, z=3;
    double i=5.3, j=2.1, k=4.3;

    cout << MIN(x, z) << endl;        // 回傳 2 整數中的最小值
    cout << MIN(i, j) << endl;        // 回傳 2 浮點數中的最小值
    cout << MIN(x, y, z) << endl;     // 回傳 3 整數中的最小值

    return 0;
}
```

回傳 2 整數中的最小值 2 回傳 2 浮點數中的最小值 2.1 回傳 3 整數中的最小值 2

# 在函數中叫用函數

在前例中我們為了求 3 整數中的最小數，又另外寫了一個 3 參數的 MIN 函數，其內容也是整個重寫。

我們的另一種選擇是利用已寫好的 2 參數 MIN 函數，來實作出 3 參數的 MIN 函數。

```
#include <iostream>

using namespace std;

// 回傳 2 整數中的最小值
int MIN(int a, int b)
{
    cout << "回傳 2 整數中的最小值" << endl;
    if(a<=b)
        return a;
    else
        return b;
}

// 回傳 3 整數中的最小值
int MIN(int a, int b, int c)
{
    cout << "回傳 3 整數中的最小值" << endl;
    return MIN(MIN(a, b), c);  // 利用 MIN(int , int)
}


int main()
{
    int x=2, y=5, z=3;

    cout << "Step 1:" << endl;
```

```
    cout << MIN(x, z) << endl;          // 回傳 2 整數中的最小值

    cout << "Step 2:" << endl;
    cout << MIN(x, y, z) << endl;       // 回傳 3 整數中的最小值

    cout << "Step 3:" << endl;
    cout << MIN(x, MIN(y, z)) << endl; // 回傳 3 整數中的最小值

    return 0;
}
```

由輸出結果我們可以看到，叫用 MIN(int , int, int) 時，MIN(int, int) 被叫用了 2 次。

Step 1: 回傳 2 整數中的最小值 2 Step 2: 回傳 3 整數中的最小值 回傳 2 整數中的最小值 回傳 2 整數中的最小值 2 Step 3: 回傳 2 整數中的最小值 回傳 2 整數中的最小值 2

練習：求 a, b 兩正整數的最大公因數(GCD)

設計一個 GCD 函數，求 2 正整數的最大公因數。

---

**1. 用迴圈慢慢找**

```
int GCD(int a, int b)
{
  if(a>b)
    swap(a, b);
  int ans = 1;
  for(int i=1; i<=a; i++) {
    if(a%i==0 && b%i==0) {
      ans = i;
    }
  }
  return ans;
}
```

**2. 超級快的「輾轉相除法」**

```
int GCD(int a, int b)
{
  int r;
  while(a%b>0) {
    r = a%b;
    a = b;
    b = r;
  }
  return b;
}
```

練習：求 a, b 兩正整數的最小公倍數(LCM)

設計一個 LCM 函數，求 2 正整數的最小公倍數。

### 利用之前的 GCD 函數

```
int LCM(int a, int b)
{
  return a/GCD(a, b)*b;
}
```

我們不使用 `a*b/GCD(a,b)` 的原因是，若先把 a*b ，其相乘後數值溢位的可能性更大，先把 a 除以兩數的公因數，再乘上 b，可以減低溢位的風險。

---

Revision #5
★Created 2024-05-05 04:23:20 UTC by huihui
Updated 2026-03-02 00:04:45 UTC by huihui