使用者函數

6

CHAPTER

6.1 函數定義

函數（function）是執行一項特殊工作之敘述的集合。在前幾章我們已經使用過二種函數：（1）每個程式都必須包含的 main() 函數，（2）如第三章介紹的數學函數 sin()、pow()、rand() 等函數。本章將介紹如何建立使用者自定的函數，事實上 sin()、pow()、rand() 等函數也都是先建立好存在資料庫供開發程式使用。

函數的主要功能有二：一是將程式中重複執行的敘述區塊定義成函數，然後以重複呼叫函數來執行函數中的敘述，如此可使程式更簡潔；二是在大程式中可將程式依功能分成許多程式片斷，然後再將各程式片斷定義成函數，如此可使程式結構化且更方便管理。

上一章討論的迴圈是有規則的重複執行某些敘述，而本章討論的函數則提供主程式隨時呼叫。例如，利用迴圈可以設計計算 1+2+3+ . . . +10 的程式。可是程式中第一次要計算 1+2+3+ . . . +10，第二次要計算 1+2+3+ . . . +100，則可以將計算總和的迴圈定義成函數。當呼叫總和函數並傳遞參數 10，則函數計算 1+2+3+ . . . +10 並傳回 55 給呼叫敘述，當呼叫總和函數並傳遞參數 100，則函數計算 1+2+3+ . . . +100 並傳回 5050 給呼叫敘述。

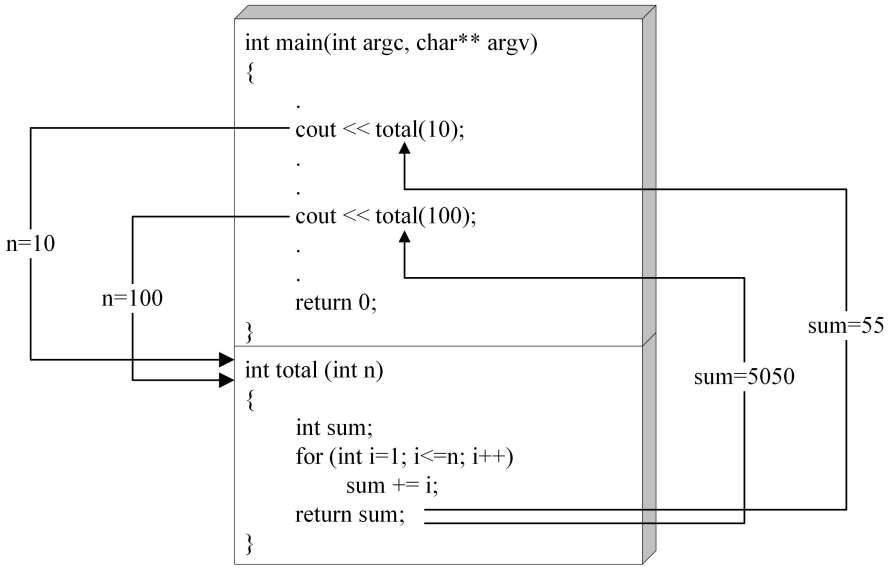


圖 6.1 函數呼叫與返回

6.1.1 宣告函數

```
傳回型態　函數名稱 (參數列)
{
    //函數本體
    return 傳回值;
}
```

- **函數表頭（header）**是指函數的第一行宣告包括傳回型態、函數名稱、與參數列。
- **函數名稱**的命名方式必須符合變數命名的原則。
- **傳回型態**表示函數傳回值的資料型態，若傳回型態為 void 表示該函數不傳回任何值，且可省略區塊中的 return 敘述。
- **參數列**可包括 0 個或多個參數，而宣告必須包含參數型態與參數名稱，例如(int a, float b, char c)。

- **參數**則是由呼叫敘述傳遞到函數的值，可以傳遞數值、變數、指標、陣列等參數。
- **return** 是返回呼叫敘述，傳回值則是要傳回給呼叫敘述的數值，若傳回型態為 void 表示不須傳回任何值，所以可以省略 return 敘述。
- **大括號（{ }）**表示函數敘述區的起始點與結束點。

下面範例是一個使用者自定函數 womain，函數內只使用 cout 敘述輸出一個字串與跳行後結束。void womain() 表示此函數結束時將不傳回任何值給呼叫敘述，而 womain(void)則表示呼叫 womain 函數不需要傳遞任何參數給 womain。

```
void womain(void)                                    //使用者函數
{
   cout << "Woman: I'm doing good. How about you?";
   cout << endl;
}
```

6.1.2 呼叫函數

函數名稱 (參數 0, 參數 1, 參數 2, ...);

- 直接使用函數名稱來呼叫函數，也可將函數名稱置於其他敘述中。
- 呼叫時傳遞的參數個數與型態，必須與定義函數時的參數個數與型態相符。

下面範例是呼叫上一個範例的 womain 函數，因為 womain 函數被定義為不需要傳遞任何參數，所以呼叫 womain 時小括號內是空的。

```
womain();                                    //呼叫使用者函數 womain
```

下圖顯示 main 函數直接呼叫 womain 函數的方塊圖，此圖只是簡單的表示出 main 函數與 womain 函數間的關係。程式 6-1 與說明流程圖將顯示程式流程的轉換過程。

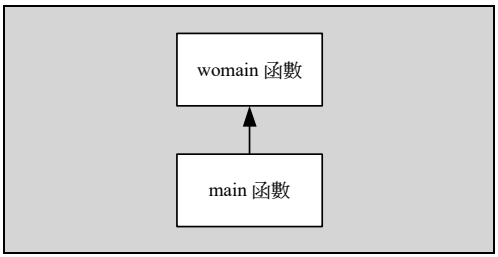


圖 6.2 呼叫函數方塊圖

程式 6-01：宣告與呼叫函數

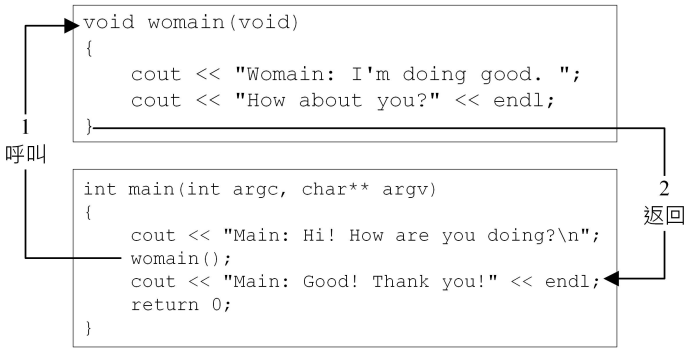
```
1.   //儲存檔名：d:\C++06\C0601.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   void womain(void)                              //使用者函數
6.   {
7.      cout << "Womain: I'm doing good. How about you?";
8.      cout << endl;
9.   }
10.
11.  int main(int argc, char** argv)
12.  {
13.     cout << "Main: Hi! How are you doing?" << endl;
14.     womain();                                   //呼叫使用者函數 womain
15.     cout << "Main: Very well! Thank you!" << endl;
16.     return 0;
17.  }
```

程式輸出

```
Main: Hi! How are you doing?
Womain: I'm doing good. How about you?
Main: Very well! Thank you!
```

下圖顯示程式 6-1 的流程轉移過程：（1）程式首先起始 main 函數並顯示 “Main: Hi! How are you doing?” 字串，（2）main 函數呼叫 womain 函數，將程式執行的流程轉移給 womain 函數，womain 函數則顯示 “Womain: I'm doing good. How about you?”，（3）執行完 womain 函數後

再將程式流程還給 main 函數，並繼續執行呼叫敘述的下一個敘述，顯示 “Main: Very well! Thank you!” 字串。



6.1.3 呼叫多個函數

下圖顯示 main 函數直接呼叫 sub1 函數與 sub2 函數的方塊圖，此圖簡單的表示出 main 函數與 sub1 函數或 sub2 函數間的關係，而 sub1 函數與 sub2 函數並無直接關係。程式 6-2 與說明流程圖將顯示程式流程的轉換過程。

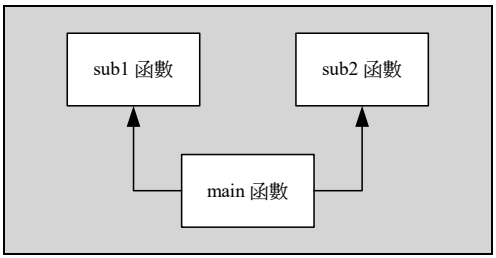


圖 6.3 呼叫多個函數方塊圖

程式 6-02：呼叫多個函數練習

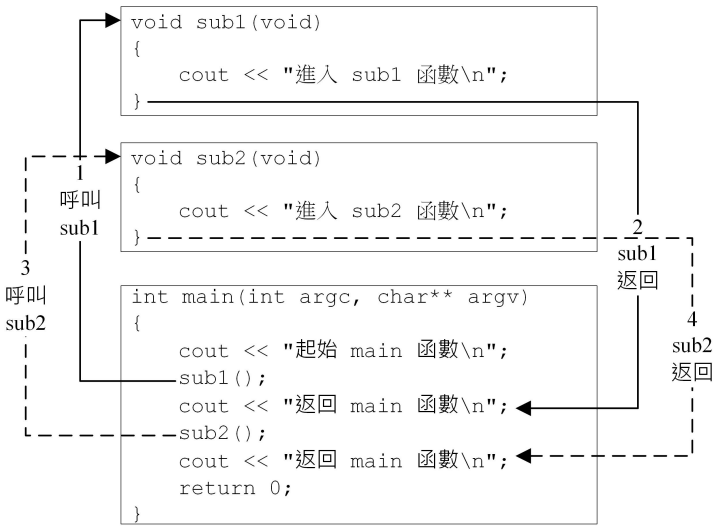
```
1.   //儲存檔名：d:\C++06\C0602.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   void sub1(void)                              //使用者函數 1
6.   {
7.      cout << "進入 sub1 函數\n";
8.   }
```

```
9.
10.  void sub2(void)                                  //使用者函數 2
11.  {
12.     cout << "進入 sub2 函數\n";
13.  }
14.
15.  int main(int argc, char** argv)
16.  {
17.     cout << "起始 main 函數\n";
18.     sub1();                                       //呼叫 sub1
19.     cout << "返回 main 函數\n";
20.     sub2();                                       //呼叫 sub2
21.     cout << "返回 main 函數\n";
22.     return 0;
23.  }
```

程式輸出

```
起始 main 函數
進入 sub1 函數
返回 main 函數
進入 sub2 函數
返回 main 函數
```

下圖顯示程式 6-2 的流程轉移過程，（0）程式首先起始 main 函數並顯示 “起始 main 函數” 字串，（1）main 函數呼叫 sub1 函數，將程式的流程轉移給 sub1 函數，sub1 則顯示 “進入 sub1 函數” 字串，（2）執行完 sub1 函數後將程式流程還給 main 函數，並繼續執行呼叫敘述的下一個敘述，顯示 “返回 main 函數” 字串。（3）main 函數繼續呼叫 sub2 函數，將程式的流程轉移給 sub2 函數，sub2 則顯示 “進入 sub2 函數” 字串，（4）執行完 sub2 函數後將程式流程還給 main 函數，並繼續執行呼叫敘述的下一個敘述，顯示 “返回 main 函數” 字串。



6.1.4 多重呼叫函數

下圖是多重呼叫函數方塊圖，它顯示 main 函數直接呼叫 childUser 函數，而 childUser 函數又呼叫 grandUser 函數的方塊圖。從圖中看出，雖然 main 函數不直接呼叫 grandUser 函數，可是它卻透過 childUser 函數呼叫 grandUser 函數，所以相當於間接呼叫 grandUser 函數。而 grandUser 函數傳回給 childUser 函數的值，也影響到整個程式的執行。程式 6-3 與說明流程圖將顯示程式流程的轉換過程。

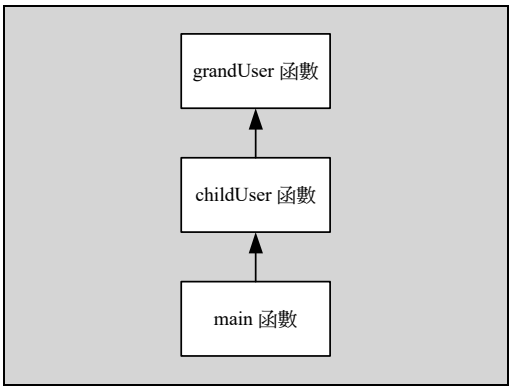


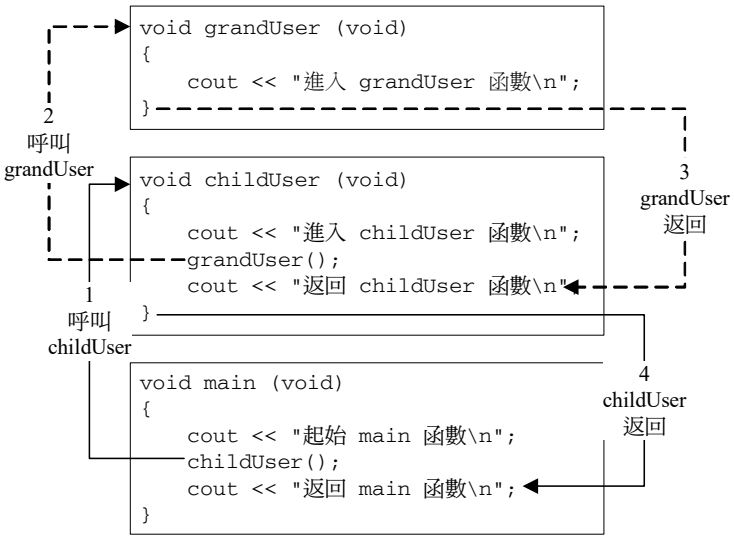
圖 6.4 多重呼叫函數方塊圖

程式 6-03：多重呼叫函數練習

```
1.  //儲存檔名：d:\C++06\C0603.cpp
2.  #include <iostream>
3.  using namespace std;
4.
5.  void grandUser(void)                          //使用者函數 2
6.  {
7.     cout << "進入 grandUser 函數\n";
8.  }
9.
10. void childUser(void)                          //使用者函數 1
11. {
12.    cout << "進入 childUser 函數\n";
13.    grandUser();                               //呼叫 grandUser 函數
14.    cout << "返回 childUser 函數\n";
15. }
16.
17. int main(int argc, char** argv)
18. {
19.    cout << "起始 main 函數\n";
20.    childUser();                               //呼叫 childUser
21.    cout << "返回 main 函數\n";
22.    return 0;
23. }
```

程式輸出

```
起始 main 函數
進入 childUser 函數
進入 grandUser 函數
返回 childUser 函數
返回 main 函數
```



上圖顯示程式 6-3 的流程轉移過程：（0）程式首先起始 main 函數並顯示 “起始 main 函數” 字串，（1）main 函數呼叫 childUser 函數，將程式的流程轉移給 childUser 函數，childUser 則顯示 “進入 childUser 函數” 字串，（2）接著 childUser 函數呼叫 grandUser 函數，將程式的流程轉移給 grandUser 函數，grandUser 函數則顯示 “進入 grandUser 函數” 字串，（3）執行完 grandUser 函數後將程式流程還給 childUser 函數，並繼續執行呼叫敘述的下一個敘述，顯示 “返回 childUser 函數” 字串。（4）執行完 childUser 函數後將程式流程還給 main 函數，並繼續執行呼叫敘述的下一個敘述，顯示 “返回 main 函數” 字串。

6.1.5 宣告函數原型

傳回型態　函數名稱 (參數型態 0, 參數型態 1, 參數型態 2, …);

- **宣告函數原型（function prototype）**敘述與函數標題敘述相同，它提供函數的基本資訊給編譯程式，包括傳遞給函數的參數型態與函數傳回的資料型態。

- 宣告函數原型必須出現在呼叫敘述之前，通常是放在程式的前置處理區。

習慣上，總是將 main 函數放在前面，而將被呼叫函數放在後面。可是從程式 6.1 至程式 6.3 可看出被呼叫函數必須定義在 main 函數之前，如果將被呼叫函數 womain 定義在 main 函數之後如下，則 main 函數呼叫 womain 函數時將產生 “Call to undefined function 'womain' ” 的錯誤。也就是說，當 C++ 編譯器編譯到呼叫 womain() 函數時，將不認識 womain() 函數，因為在呼叫之前並沒有宣告。

```
int main(int argc, char** argv)                 //main 函數
{
   cout << "Main: Hi! How are you doing?" << endl;
   womain();                                    //錯誤, 呼叫未定義函數
   cout << "Main: Very well! Thank you!" << endl;
   return 0;
}

void womain(void)                               //使用者函數
{
   cout << "Womain: I'm doing good. How about you?";
   cout << endl;
}
```

在上面範例的 main 函數之前，宣告 womain 函數原型如下面範例，則編譯呼叫 womain 函數時就不會產生錯誤。宣告 womain 函數原型只是告訴 C++ 編譯器，本程式稍後會定義與使用 womain 函數，則編譯器會按照函數原型的宣告，保留 womain 的符號與相關資訊(如參數與傳回資料等)。

```
void womain(void);                              //宣告使用者函數原型

int main(int argc, char** argv)                 //main 函數
{
   cout << "Main: Hi! How are you doing?" << endl;
   womain();                                    //呼叫使用者函數 womain
   cout << "Main: Very well! Thank you!" << endl;
   return 0;
}

void womain(void)                               //使用者函數
{
   cout << "Womain: I'm doing good. How about you?";
```

```
    cout << endl;
}
```

程式 6-04：宣告函數原型練習

```
1.   //儲存檔名：d:\C++06\C0604.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   void womain(void);                          //宣告使用者函數原型
6.
7.   int main(int argc, char** argv)
8.   {
9.      cout << "Main: Hi! How are you doing?" << endl;
10.     womain();                                //呼叫使用者函數 womain
11.     cout << "Main: Very well! Thank you!" << endl;
12.     return 0;
13.  }
14.
15.  void womain(void)                          //使用者函數
16.  {
17.     cout << "Womain: I'm doing good. How about you?";
18.     cout << endl;
19.  }
```

程式輸出

```
Main: Hi! How are you doing?
Womain: I'm doing good. How about you?
Main: Very well! Thank you!
```

6.2 傳遞參數

參數（arguments）就是某敘述呼叫函數時，同時傳遞給該函數的數值。例如呼叫正弦函數（sin）必須傳遞要計算的角度，或呼叫冪次函數（pow）必須傳遞要計算的底數與冪次。如下範例：

```
x = sin(30 / 180 * 3.14159);                 //求 sin(30°)值
p = pow(2, 5);                               //求 2^5 值
```

下圖的參數加 s，表示呼叫敘述可以傳遞多個參數給被呼叫的函數。但是傳遞參數的個數必須與函數宣告的參數個數相等，且被傳遞的參數型態也必須與函數宣告的參數型態相符。不過 6.1 節也曾使用無參數的呼

叫，也就是不傳遞參數給被呼叫的函數，因此該函數只能使用公用變數或函數本身的區域變數。

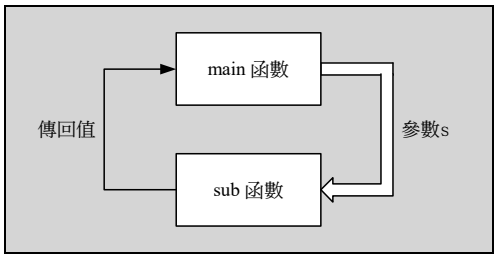


圖 6.5 傳遞參數流程

6.2.1 傳遞單一參數

首先介紹**傳遞單一數值（pass-by-value）參數**，也就是被呼叫的函數只宣告一個參數，而呼叫敘述也只傳遞一個參數。

程式 6-05 是將 main 函數放在被呼叫的函數之前，所以在 main 函數的呼叫敘述之前必須宣告函數原型。

```
void number(int);                                    //宣告函數原型
```

然後可以將函數定義在 main 函數之後。void number(int n) 宣告此函數名稱為 number，它必須接受一個整數型態的參數（n），void 表示此函數沒有傳回值。Number 函數是判斷接收的參數是否是 5 的倍數，因 number 沒有傳回值，所以如果要知道函數執行結果，則可以在函數中使用 cout 函數輸出訊息。

```
void number(int n)                                   //判斷 5 的倍數函數
{
   if (n % 5 == 0)                                   //若參數 n 除以 5 餘數為 0
      cout << n << " 是 5 的倍數\n";
   else
      cout << n << " 不是 5 的倍數\n";
}
```

定義函數後，可以在 main 函數中使用函數名稱 number 呼叫該函數，呼叫時必須傳遞一個整數型態的參數（如 135）。

```
int main(int argc, char** argv)
{
    number(135);
    return 0;
}
```

```
void number(int n)
{
    if (n % 5 == 0)
       cout << n << " 是 5 的倍數";
    else
       cout << n << " 不是 5 的倍數";
}
```

程式 6-05：判斷是否爲 5 的倍數

```
1.  // 儲存檔名：d:\C++06\C0605.cpp
2.  #include <iostream>
3.  using namespace std;
4.
5.  void number(int n);                          //宣告函數原型
6.
7.  int main(int argc, char** argv)
8.  {
9.     number(5);                                //傳遞 5 給 number 函數
10.    number(58);                               //傳遞 58 給 number 函數
11.    number(135);                              //傳遞 135 給 number 函數
12.    return 0;
13. }
14.
15. void number(int n)                           //判斷 5 的倍數函數
16. {
17.    if (n % 5 == 0)                           //若參數 n 除以 5 餘數為 0
18.       cout << n << " 是 5 的倍數\n";
19.    else
20.       cout << n << " 不是 5 的倍數\n";
21. }
```

程式輸出

```
5 是 5 的倍數
58 不是 5 的倍數
135 是 5 的倍數
```

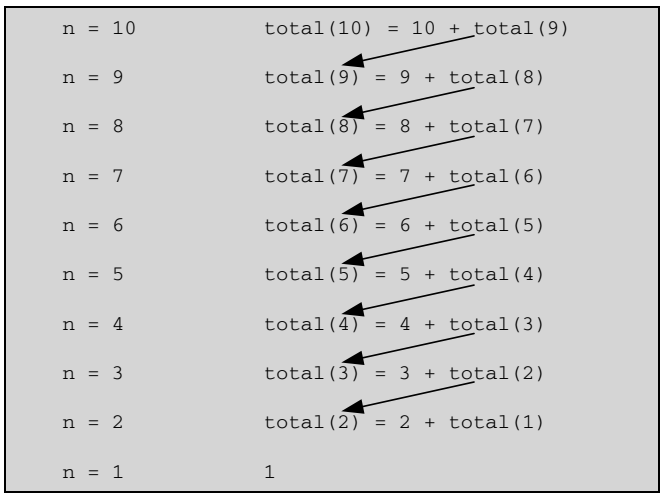
6.5 函數特殊用途

6.5.1 遞迴函數

遞迴函數呼叫（Recursive Function Calls）就是函數中包含一個呼叫自己的敘述。它也可能出現在二函數之間，如函數 1 呼叫函數 2，而函數 2 再呼叫函數 1 而形成遞迴呼叫。

下面範例是遞迴函數呼叫，當 n>0 時傳回 n+total(n-1)，而 total(n-1) 又呼叫 total 函數並傳遞 n-1 值。假設 n=10，所以呼叫 total(10) 傳回 10+total(9)，接著呼叫 total(9) 傳回 9+total(8)，total(8) 傳回 8+total(7)，total(7) 傳回 7+total(6)，total(6) 傳回 6+total(5)，total(5) 傳回 5+total(4)，total(4) 傳回 4+total(3)，total(3) 傳回 3+total(2)，total(2) 傳回 2+total(1)，total(1) 則傳回 1 並結束遞迴呼叫如下圖。

```
int total(int n)                                    //計算總和函數
{
   if (n > 1)                                       //若 n > 1
      return n + total(n-1);                        //呼叫函數自身
   else                                             //若 n<=1
      return 1;                                     //結束遞迴呼叫,並傳回 1
}
```



程式 6-19：遞迴呼叫計算總和

```
1.   //儲存檔名：d:\C++06\C0619.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   int total(int);                                  //宣告函數原型
6.
7.   int main(int argc, char** argv)
8.   {
9.      cout << "1+2+3+...+10=" << total(10);   //顯示函數傳回值
10.     cout << "\n1+2+3+...+100=" << total(100);   //顯示函數傳回值
11.     cout << endl;
12.     return 0;
13.  }
14.
15.  int total(int n)                               //計算總和函數
16.  {
17.     if (n > 1)                                   //若 n > 1
18.        return n + total(n-1);                    //呼叫函數自身
19.     else                                         //若 n<=1
20.        return 1;                                 //結束遞迴呼叫,並傳回 1
21.  }
```

程式輸出

```
1+2+3+...+10=55
1+2+3+...+100=5050
```

6.5.2 定義函數巨集 #define

#define　巨集名稱　數值
#define　巨集名稱　字串
#define　巨集名稱　函數

- **#define** 指令（directive）可以定義一個巨集名稱來取代數值、字串、或函數。
- **巨集名稱**的命名方式與變數名稱相同。但大多數程式設計師喜歡使用大寫符號來區分該符號是使用 #define 宣告的符號。

- **數值、字串、函數**是在程式編譯前，以此對等資料取代程式中所有的巨集名稱。
- #define 是前置處理指令（preprocessor directive），所以不使用分號結尾。

下面範例是定義一個常數巨集，令 PI 等於浮點數值 3.14159。

```
#define  PI 3.14159                                    //定義符號 PI
```

下面範例是定義三個字串或標點符號巨集，令 BEGIN 等於左大括號（{}、END 等於右大括號（ ））、TAB 等於定位字元（'\t'）。

```
範例：定義字串與符號巨集名稱
#define  BEGIN   {                                     //定義起始符號
#define  END     }                                     //定義結束符號
#define  TAB     '\t'                                  //定義定位符號
```

下面範例是定義二個函數巨集，第一式令 ABS(n) 等於 (n<0 ? –n : n)，當參數小於 0 則傳回-n 值，當參數不小於 0 則傳回 n 值。第一式令 EVEN(n) 等於 (n%2==0 ? "偶數" : "奇數")，當參數 n 除以 2 餘數等於 0 則傳回字串 "偶數"，當參數 n 除以 2 餘數不等於 0 則傳回字串 "奇數"。

```
#define  ABS(n)   (n<0 ? -n : n)                       //定義取絕對值函數巨集
#define  EVEN(n)  (n%2==0 ? "偶數" : "奇數")           //定義判斷奇偶數函數巨集
```

程式 6-20：#define 定義常數符號

```
1.   //儲存檔名：d:\C++06\C0620.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   #define  PI 3.14159                               //定義符號 PI
6.
7.   int main(int argc, char** argv)
8.   {
9.      int r = 5;
10.     cout << "圓半徑 = " << r;                       //顯示圓半徑值
11.     cout << "\n 圓周長 = " << 2 * PI * r;          //顯示圓周長值
12.     cout << "\n 圓面積 = " << PI * r * r;          //顯示圓面積值
13.     cout << endl;
```

```
14.     return 0;                                          //告訴 OS 程式正常結束
15.  }
```

程式輸出

```
圓半徑 = 5
圓周長 = 31.4159
圓面積 = 78.5397
```

程式 6-21：#define 定義字串名稱

```
1.   //儲存檔名：d:\C++06\C0621.cpp
2.   #include <iostream>
3.   #include <iomanip>
4.   using namespace std;
5.
6.   #define  BEGIN  {                                   //定義起始符號
7.   #define  END    }                                   //定義結束符號
8.   #define  TAB    '\t'                                //定義定位符號
9.   #define  PI     3.14159                             //定義常數符號
10.
11.  int main(int argc, char** argv)
12.  BEGIN                                                //使用起始符號
13.     cout << "半徑\t 圓周長\t   圓面積\n";
14.     cout.precision(3); cout.setf(ios::fixed);         //設定輸出格式
15.     for (int r = 5; r <= 10; r++)                     //顯示計算值迴圈
16.     BEGIN                                             //使用起始符號
17.        cout << setw(3) << r << TAB;                   //顯示圓半徑值
18.        cout << setw(6) << 2 * PI * r << TAB;          //顯示圓周長值
19.        cout << setw(9) << PI * r * r << endl;         //顯示圓面積值
20.     END                                               //使用結束符號
21.     return 0;
22.  END                                                 /使用結束符號
```

程式輸出

```
 半徑      圓周長      圓面積
  5       31.416      78.540
  6       37.699     113.097
  7       43.982     153.938
  8       50.265     201.062
  9       56.549     254.469
 10       62.832     314.159
```

程式 6-22：#define 定義函數巨集

```
1.   //儲存檔名：d:\C++06\C0622.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   #define MAX(x, y) ((x>y) ? x : y)          //判斷較大值函數巨集
6.   #define MIN(x, y) ((x<y) ? x : y)          //判斷較小值函數巨集
7.
8.   int main(int argc, char** argv)
9.   {
10.     int num1, num2;
11.     cout << "請輸入二個整數：";
12.     cin >> num1 >> num2;
13.     cout << num1 << " 和 " << num2;
14.     cout << " 的較大值是 " << MAX(num1, num2);   //呼叫巨集函數 MAX
15.     cout << endl;
16.     cout << num1 << " 和 " << num2;
17.     cout << " 的較小值是 " << MIN(num1, num2);   //呼叫巨集函數 MIN
18.     cout << endl;
19.     return 0;
20.  }
```

程式輸出

```
請輸入二個整數：234 345
234 和 345 的較大值是 345
234 和 345 的較小值是 234
```

6.6 習題

選擇題

1. 如果函數沒有傳回值，則必須宣告傳回型態為_____。

   a) null　　b) NULL　　c) void　　d) VOID

2. 若函數的表頭是 void show(int number)，則正確呼叫此函數的敘述是_____。

   a) show(5);　　b) show(5.5);　　c) show(int 5);　　d) void show(int 5);

3. 傳遞給函數的數值稱為_____。

   a) 變數　　b) 參數　　c) 傳遞值　　d) 巨集

4. 如果被呼叫函數定義於呼叫敘述後，則必須在呼叫函數前宣告____。

   a) 函數巨集　　b) 函數表頭　　c) 函數原型　　d) 函數名稱

5. 一個函數可以包含_____參數。

   a) 0 或 1 個　　b) 2 個　　c) 多個　　d) 以上皆對

6. 一個函數可以包含_____傳回值。

   a) 0 或 1 個　　b) 2 個　　c) 多個　　d) 以上皆對

7. _____函數可以終止程式執行。

   a) break()　　b) stop()　　c) exit()　　d) end()

8. 如果函數中包含一個呼叫自己的敘述，則此函數稱為_____。

   a) 巨集函數　　b) 呼叫函數　　c) 遞迴函數　　d) 多載函數

實作題

1. 寫一 C++ 程式，計算球面積與體積。

   a) 定義一個 sArea(pi, r) 函數，接收 pi 與 r 參數，傳回球面積給呼叫敘述。

   b) 定義一個 sVolumn(pi, r) 函數，接收 pi 與 r 參數，傳回球體積給呼叫敘述。

   c) 在 main 函數中，呼叫 sArea() 函數與 sVolumn 函數，假設球半徑為 5、6、7、8、9、10。（公式：球表面積 = $4\pi r^2$，球體積 = $(4/3)\pi r^3$）

2. 寫一 C++ 程式，求二個整數的最大公因數（GCD）與最小公倍數（LCM）。

   a) 定義 gcd(x, y) 與 lcm(x, y) 二個函數，分別計算二個整數 x 與 y 的最大公因數（GCD）與最小公倍數（LCM），並傳回 GCD 與 LCM 給呼叫敘述。

   b) 程式從鍵盤輸入二個整數資料 x 與 y，然後顯示二數的 GCD 與 LCM。