Chapter 6 ： 函式的定義及宣告

程式中如需經常對不同的資料重覆相同之計算，我們可將此計算定義成為函式，此計算的內容只需於函式之中定義一次，程式每次需進行此計算時，只需要呼叫此函式，將欲處理之資料傳入即可，而不必於主程式中重覆相同的程式命令。

6.1 函式定義

函式定義就是指以程式命令，依照函式定義的格式，定義出此自訂函式所需執行工作內容，其格式如下。

格式:

```
回傳值資料型態  函式名稱(資料型態 參數 A, 資料型態 參數 B, …..)
{
    程式命令(函式執行工作的內容);
}
```

說明:

一個函式通常有接收參數(parameter)以及回傳值(但非必要)；參數是用來接收函式呼叫時所輸入與其對應的引數(argument)值(現階段而言：傳值法)，其格式為宣告於函式名稱後的( )中未指定值之變數，之後於{}中基於參數以程式命令定義函式的內容；回傳值為函式執行完其所定義之運算後，以 return 指令傳回之計算結果，成為主程式中的函式本身所代表之值。

就目前而言，一個函式可有任意個接收參數，以接收多個對應的呼叫引數值，但只能有一個回傳值。函式定義時，需宣告其接收參數以及回傳值之資料型態，如函式無回傳值，則需宣告其為 void，無輸入參數則( )中為空白即可。

例:

```
double area(double r)
{
    double a;
    a = 3.1415926 * r * r;
    return a;
}
```

```
void print_number(int i)
{
    cout<<"number = "<< i << endl;
}

void error( )
{
    cout<<"Error: Matrix cannot be inversed!"<<endl;
    exit(0);
}
```

注意要點:

使用自訂的函式，函式的定義的位置必需出現於呼叫使用之前，否則就必須於使用之前先做函式原型宣告。

程式範例: cpp_ex27.cpp

6.2 函式原型宣告

程式範例 cpp_ex27.cpp 的做法(函式定義於主程式 main 之前)可被 C/C++的編譯器接受且順利執行，但建議於定義函式之前，最好先做函式原型宣告，其格式如下。使用函式原型宣告，函式定義之程式碼位置可出現於函式呼叫之後，可使函式定義的位置與先後順序不受限制。

程式編排上，如主程式前只有函式原型宣告，可使程式較清楚易讀。函式定義不受使用函式原型宣告與否影響，格式與前完前相同。

格式:

傳回值資料型態　函式名稱(資料型態 參數 A, 資料型態 參數 B, …..);

例:

```
double area(double r);
```

```
void print_number(int i);

void error( );
```

程式範例: cpp_ex28.cpp

注意要點:

1. 函式原型宣告主要是宣告此函式之參數數目及其個別的資料型態，與其定義(程式內容)無關，故函式原型宣告中之參數名稱可與函式定義中的不同，甚至可不指定，只需資料型態的宣告即可，但函式原型與函式定義中相對應參數的資料型態必須相同。例：以下三函式原型宣告均可。

   ```
   double area(double r);
   double area(double rad);
   double area(double );
   ```

2. 使用函式原型宣告促使編譯器去核對函式原型宣告與定義之參數列的各個參數的資料型別是否一致，並於不一致時顯示錯誤訊息，以確保符合函式原型宣告規定引數型別的呼叫必定有對應之函式定義存在(定義可位於呼叫之後)。
3. 有函式原型宣告，必定要有對應的函式定義(同名，同數量與型態參數與傳回值)，函式執行之工作內容必須定義於函式定義。

6.3 inline 函式

inline 函式也是函式的一種，只是其格式是在函式傳回值前，加上 inline 關鍵字，如下。

格式:

```
inline 傳回值資料型態   函式名稱(資料型態 參數 A, 資料型態 參數 B, …)
{
    程式命令(函式實際工作的內容);
}
```

說明:

inline 函式也是函式的一種，其定義與規則與一般函式相同，使用 inline 函式可增加程式執行的速度，但先決條件是 inline 函式的程式碼不能太大。

編譯器編譯到 inline 函式時，會將 inline 函式的內容，插入 inline 函式的呼叫處做編譯，有如於主程式中重複函式內的程式一樣，故較有效率。

程式範例: cpp_ex29.cpp

6.4 函式參數預設值

C/C++允許於函式參數列中設定其接收參數的預設值，如此一來於函式呼叫時有預設值的參數可不需指定其值(引數)，可增加函式使用的彈性。

當某參數有設定預設值時，如呼叫時其值(引數)未指定，則會起用預設值為其值，如有指定值(引數)，則以指定值為其值。

格式:

函式參數預設值的指定位置，可在函式定義或函式原型宣告處，但同時只能有一處有預設值之設定，並於需位於函式被呼叫使用之前，故只可能有以下兩種格式。

例:

(1)
```
int sum(int x = 0, int y = 0)   //函式定義
{
    return x+y;
}

void main()
{
    …..
    result = sum(c);      //error before, OK here
    ……
}
```

(2) int sum(int x = 0, int y = 0); //函式原型宣告

```
void main()
{
      …..
      result = sum( );       //error before, OK here
      ……
}

int sum(int x , int y)    //函式定義
{
      return x+y;
}
```

程式範例: cpp_ex30.cpp

注意要點:

1. 函式參數的預設值，是在函式呼叫時，其參數相對位置的輸入值(引數)未被指定時，系統才會自動引用預設值；如有輸入值(引數)，則使用輸入值(引數)為參數之值。

2. 函式呼叫時參數值之設定方式，為由左至右逐一設定，無論有無預設值。故如有無預設值的參數，必須集中在函式參數列的左側，如此編譯器才能將函式呼叫所輸入值(引數)先指定給那一些無預設值的參數。(編譯器無法分辨並先指定無預設值的參數)

   例： int sum(int x=0, int y=0, int z=0);
   int sum(int y, int x=0, int z=0);
   int sum(int x, int y, int z=0);

   以下不可：

   int sum(int x=0, int y, int z=0);