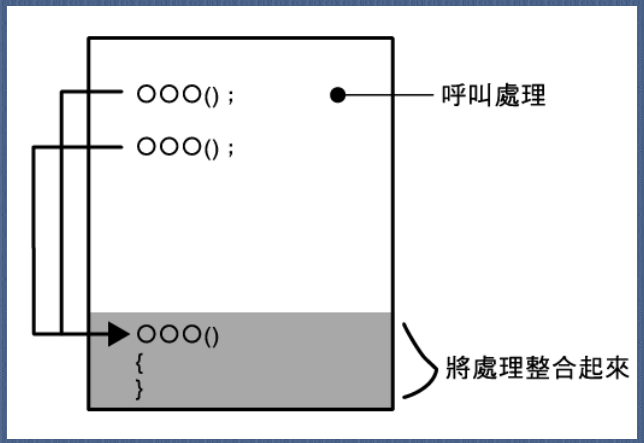
函數

函數

- 函數（function）
  - 可以整合某些特定的處理
  - 整合好的處理可以隨時呼叫使用
  - C++的程式本身也是一個函數，也就是main()函數。
  - 使用函數可簡化程式。

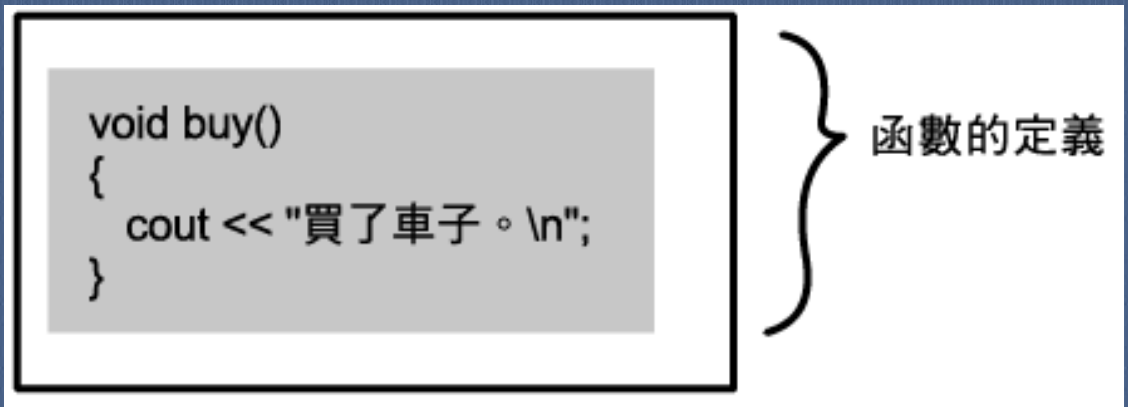


7-2 函數的定義與呼叫

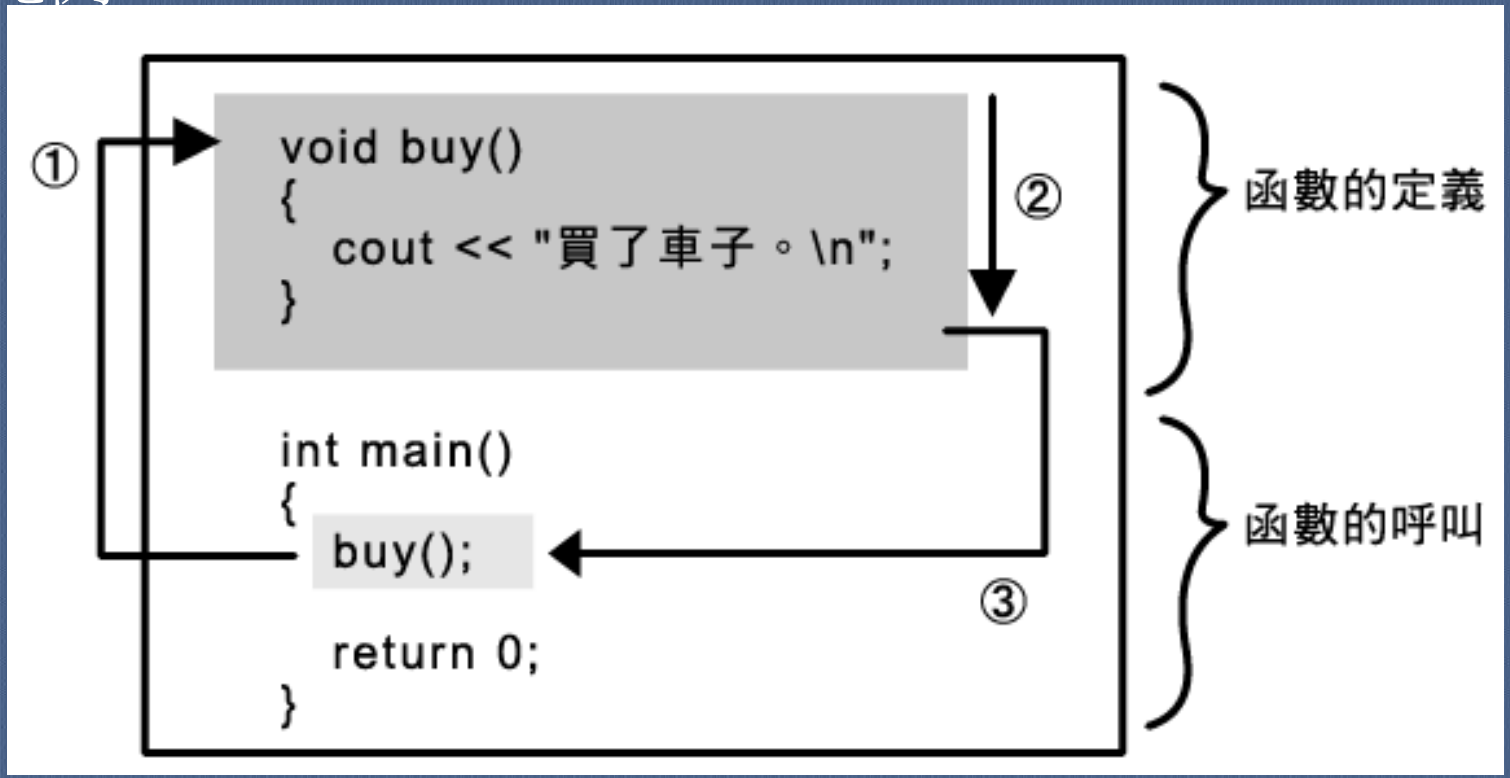
- 定義函數的語法：

```
傳回值的型態 函數名稱（參數列表）
{
  敘述;
  …
return 運算式;
}
```

- 範例：



- 呼叫函數的語法：
  函數名稱(引數列表);
- 範例：



Sample1.cpp ▶ 製作基本的函數

```
#include <iostream>
using namespace std;

//buy 函數的定義
void buy()
{
   cout <<   "買了車子。\n"
}

//buy 函數的利用
int main()
{
   buy();

   return 0;
}
```

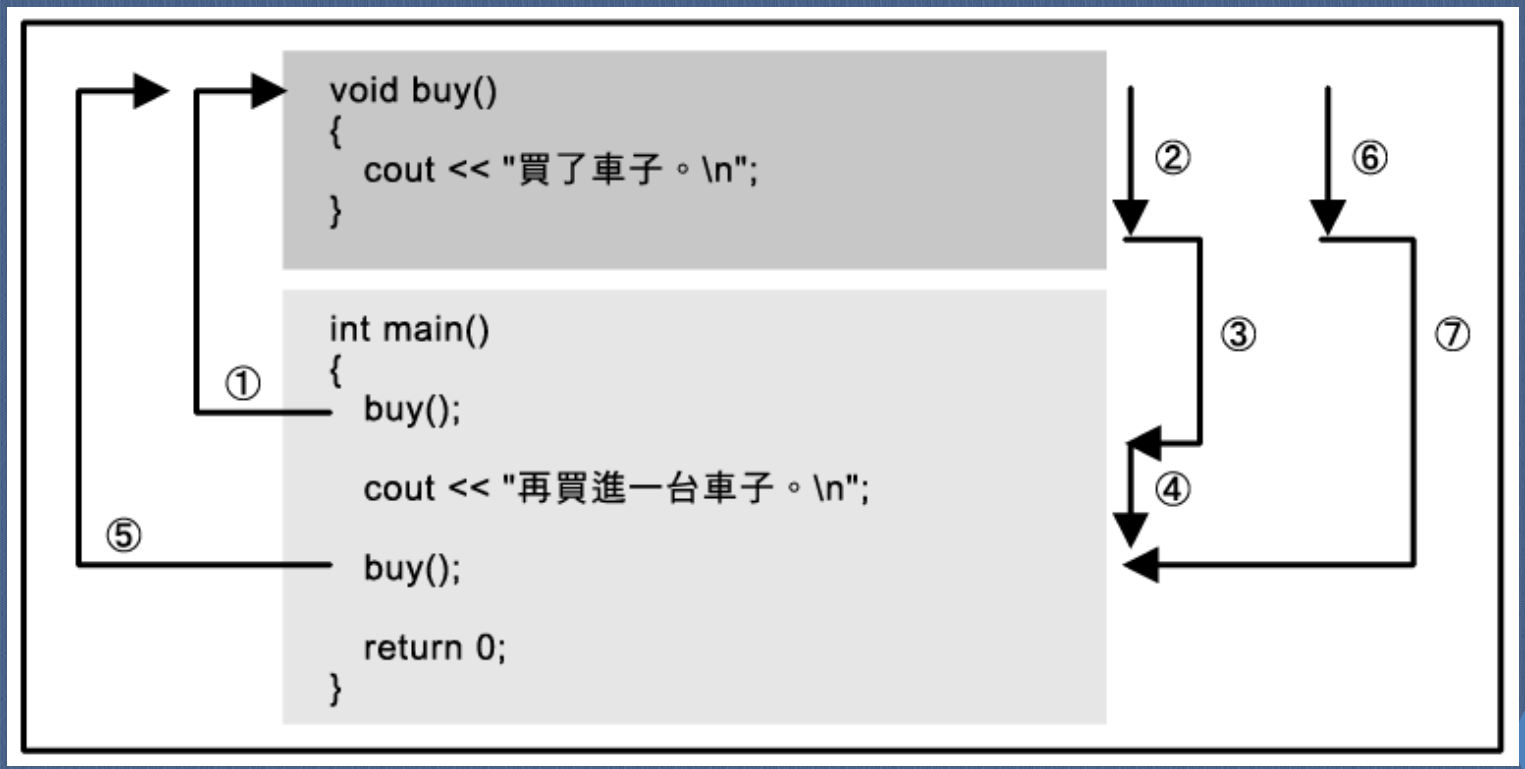
這是 buy()函數的處理內容

這裡執行 buy()函數的處理

- 使用函數的處理流程為：

  1.呼叫函數；2.執行函數中的處理；3.回到呼叫處。

- 也可以多次呼叫函數，範例如下：

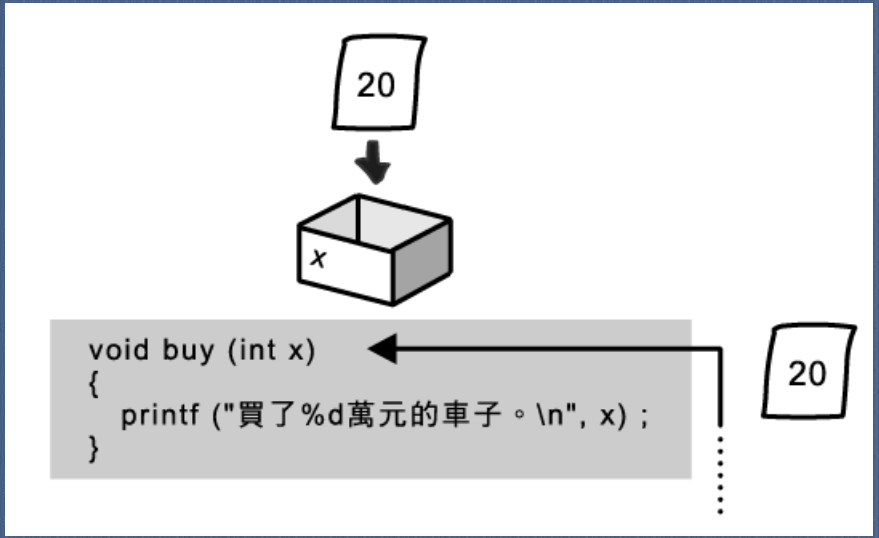


Sample2.cpp ▶ 不斷呼叫函數

```
#include <iostream>
using namespace std;

//buy 函數的定義
void buy()
{
    cout << "買了車子。\n";
}

//buy 函數的利用
int main()
{
    buy();●——呼叫出 buy()函數

    cout << "再買進一台車子。\n";

    buy();●——再呼叫一次 buy()函數

    return 0;
}
```

引數

- 使用引數傳遞資料
  - 當要呼叫出函數時，可以進行以下的處理：
    從呼叫處將某項資料（值）傳遞到函數內，
    並依照該值來執行相對應之處理。
  - 要傳遞給函數的資料稱為引數（argument）。



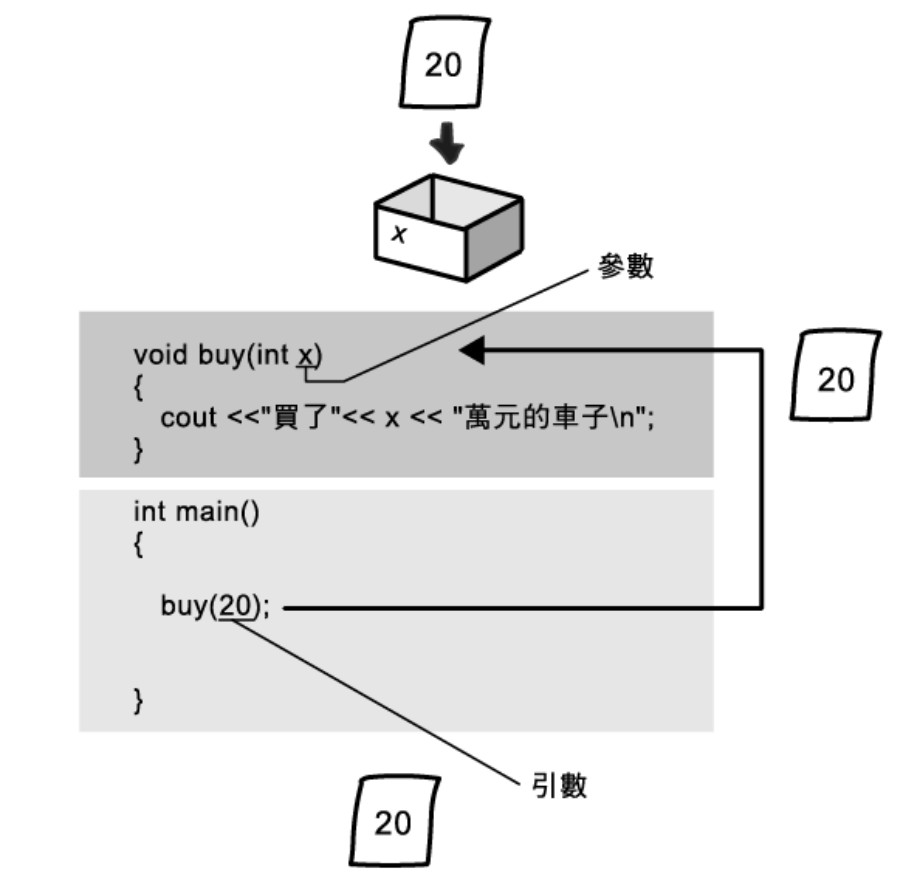
- 傳遞引數給函數的範例如下：

```
#include <iostream>
using namespace std;

//buy函數的定義
void buy(int x)
{
   cout <<"買了"<<x <<"萬元的車子。\n";
}

//buy函數的呼叫
int main()
{
   buy(20);
   buy(50);
   return 0;
}
```

買了20萬元的車子。
買了50萬元的車子。

在這個main()函數當中，會執行以下處理：
第一次呼叫出buy()函數時，傳遞「20」這個值之後呼叫
第二次呼叫出buy()函數時，傳遞「50」這個值之後呼叫

- 函數本身所定義的引數（變數）稱為**參數**（parameter）。另一方面，從函數的呼叫來源所傳遞的引數（值）則稱為**引數**（argument）。



- 所以前面範例的變數x是參數，「20」和「50」則是引數。

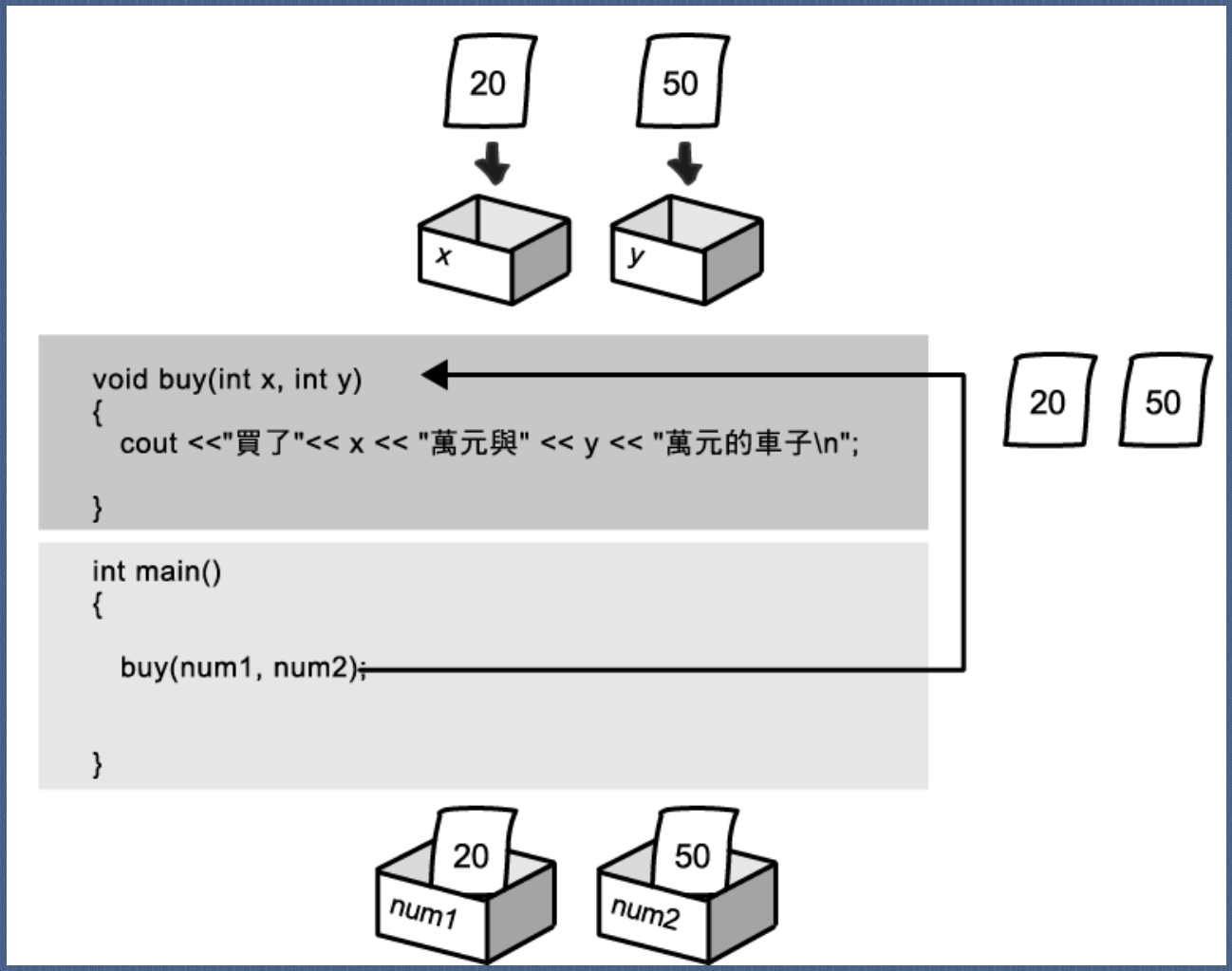
- 將從鍵盤輸入的值傳到函數中：

```
#include <iostream>
using namespace std;
//buy函數的定義
void buy(int x)
{
   cout << "買了"<<x <<"萬元的車子。\";
}

//buy函數的呼叫
int main()
{
   int num;
   cout << "第1台要買多少錢的車子？ \";
   cin >> num;
   buy(num);
   cout << "第2台要買多少錢的車子？ \";
   cin >> num;
   buy(num);
   return 0;
}
```

在C++中，傳遞到函數的並不是引數本身，而是儲存在其中的「值」。這種引數的傳遞方法有時候稱之為傳值（pass by value）。

這裡使用main()函數當中所準備的變數num（的值），作為由呼叫來源傳遞到函數的引數。所以會將由鍵盤讀取到num的值傳遞到函數中。

- 函數可以帶有多個引數，但在呼叫時要用逗號(，)來區隔所指定的多個引數，這些引數又叫做「引數列表」，它們會依指定的順序傳遞給參數。
- 範例：



Sample5.cpp ▶ 使用帶有多個引數的函數

```
#include <iostream>
using namespace std;

//buy 函數的定義
void buy(int x, int y)
{
    cout << "買了" << x << "萬元與" << y << "萬元的車子。\n";
}

//buy 函數的呼叫
int main()
{
    int num1, num2;

    cout << "要買多少錢的車子？ \n";
    cin >> num1;

    cout << "要買多少錢的車子？ \n";
    cin >> num2;

    buy(num1, num2);

    return 0;
}
```

帶有 2 個引數的函數

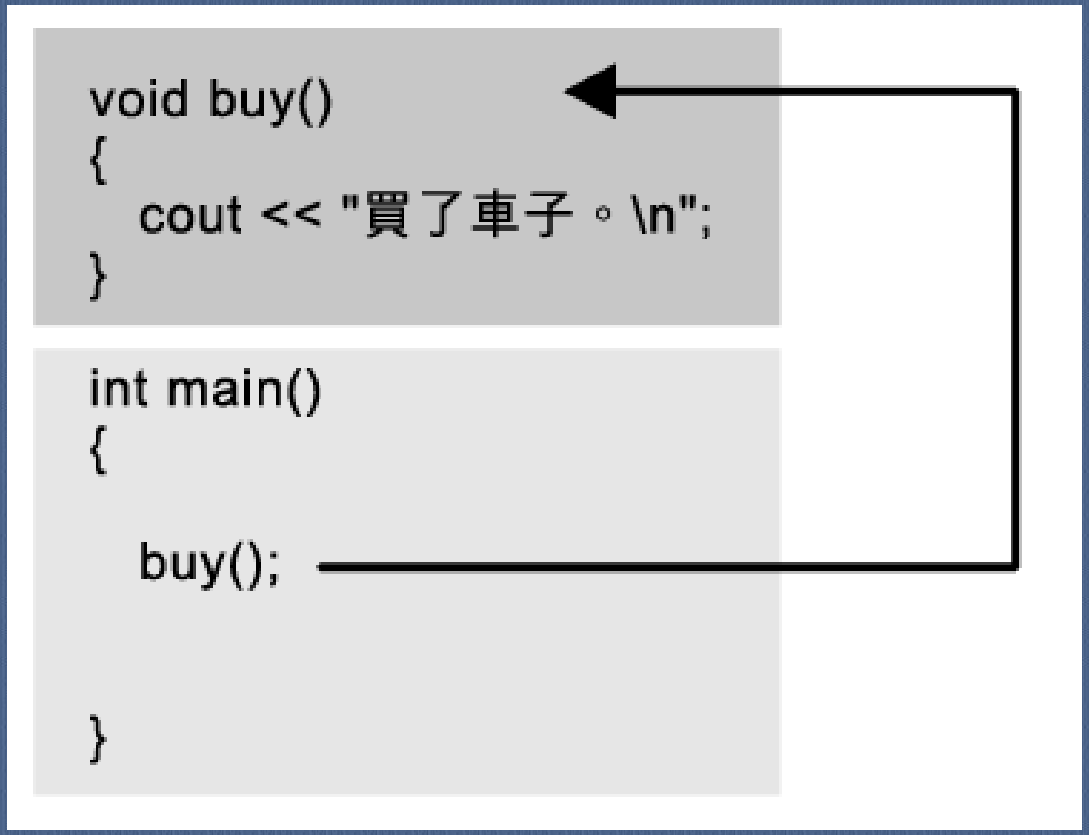
輸出第 2 個引數

輸出第 1 個引數

傳遞 2 個引數

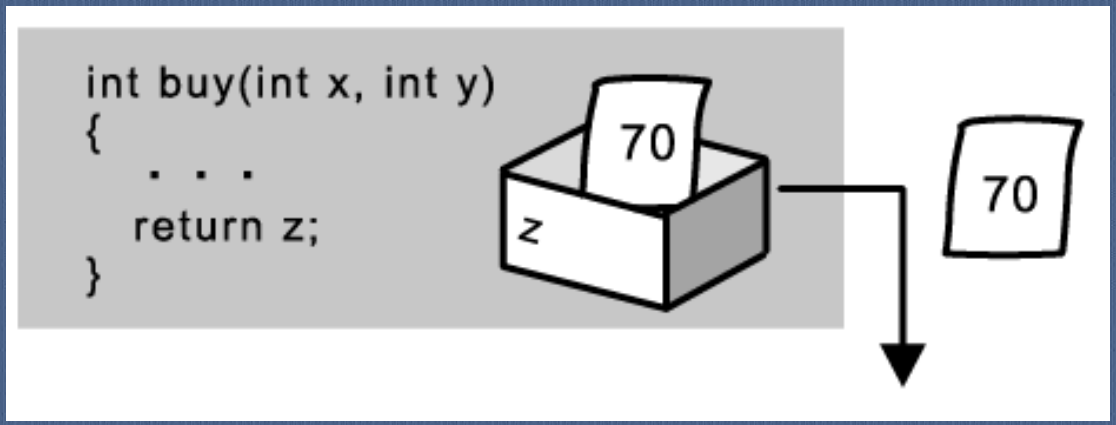
- 定義沒有引數的函數時，可以省略引數型態的指定，或指定為void型態。

- 範例：

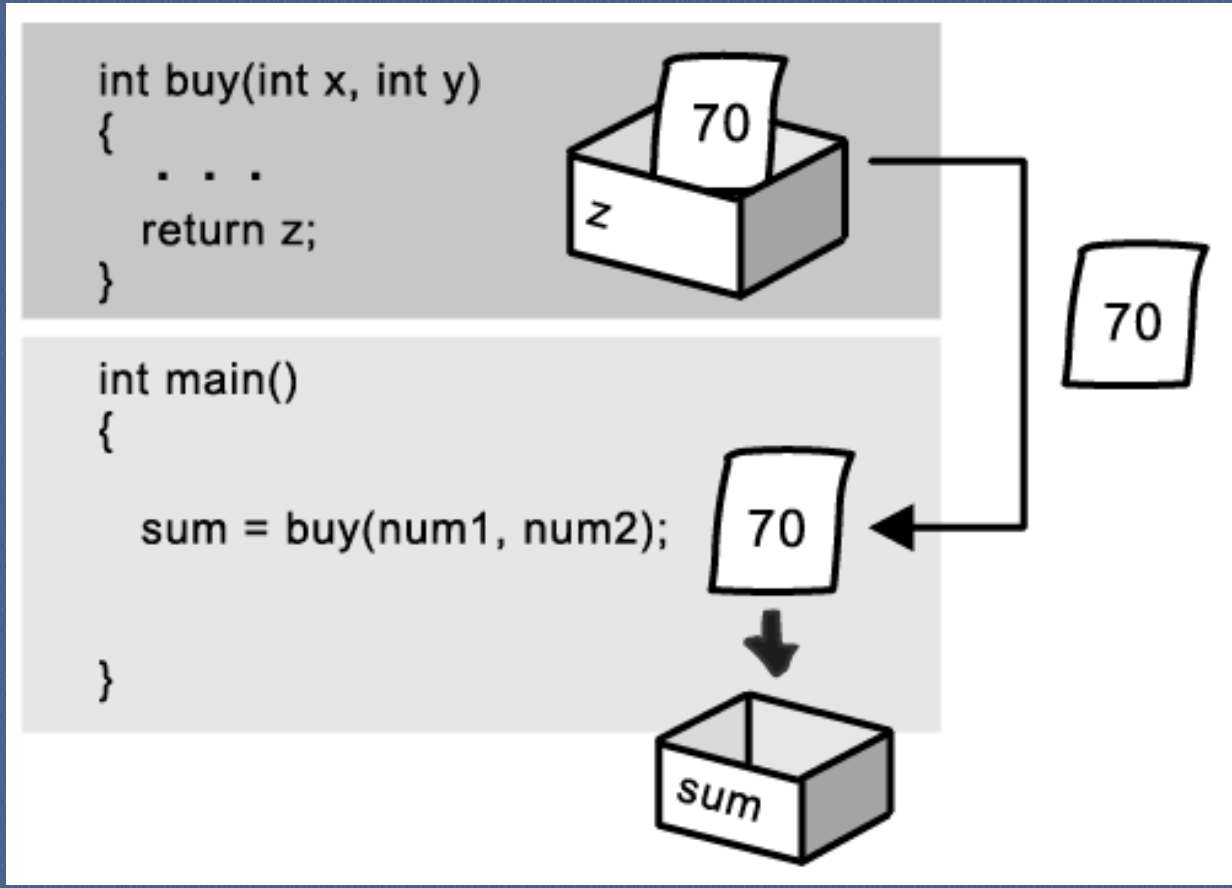


傳回值

- 理解傳回值的機制
  - 從函數本身傳回特定資訊到函數的呼叫處
  - 從函數所傳回的資訊稱為傳回值（return value）。它與可以多重指派的引數不同，傳回值只能有一個。
- 在定義函數時會先指定傳回值的型態，而return敘述後之運算式的值將會傳回呼叫處。



- 可以在呼叫來源處使用傳回值來進行處理。
- 範例：



Sample6.cpp ▶ 帶有傳回值的函數

```
#include <iostream>
using namespace std;

//buy 函數的定義
int buy(int x, int y)
{
    int z;

    cout << "買了"<<x << "萬元與" << y <<"萬元的車子。\";

    z = x+y;

    return z;
}

//buy 函數的呼叫
int main()
{
    int num1, num2, sum;

    cout << "要買多少錢的車子？ \n";
    cin >> num1;

    cout << "要買多少錢的車子？\";
    cin >> num2;

    sum = buy(num1, num2);

    cout << "合計為" << sum << "萬元。\";

    return 0;
}
```

帶有傳回值的函數

將傳回值傳回

呼叫函數，將該傳回值指派給變數 sum

輸出傳回值的值

- 沒有傳回值的函數
  - 範例：

```
/* buy函數的定義 */
void buy()
{
   cout << "買了車子。\n";
}
```

- 在不帶傳回值的函數中，要先將傳回值的型態設定為void。
- 也可利用以下這個單純的return敘述來結束函數：

```
/* buy函數的定義 */
void buy()
{
   cout << "買了車子。\n";

   return;
}
```

函數的利用

- 求合計值的函數：

```
#include <iostrem>
;
//sum函數的定義
int sum(int x, int y)
{
  return x+y;
}

int main()
{
  int num1, num2, ans;
  cout << "請輸入第1個整數：\";
  cin >> num1;
  cout << "請輸入第2個整數：\";
  cin >> num2;
  ans = sum(num1, num2);
  cout << "合計為" << ans << "。\";
  return 0;
}
```

接收兩個數值

進行傳回合計值的處理

呼叫函數

輸出傳回值

- 讓函數自己呼叫自己的處理機制，就叫做遞迴(recursion)。

- 求最大值的函數：

```
#include <iostream>
using namespace std;
//max函數的定義
int max(int x, int y)
{
  if (x > y)
    return x;
  else
    return y;
}
int main()
{
  int num1, num2, ans;
  cout << "請輸入第1個整數：\";
  cin >> num1;
  cout << "請輸入第2個整數：\";
  cin >> num2;
  ans = max(num1, num2);
  cout << "最大值為" << ans << "。\";
  return 0;
}
```

接收兩個數值

當x大於y的時候

傳回x的值

如果不是的時候，則傳回y的值

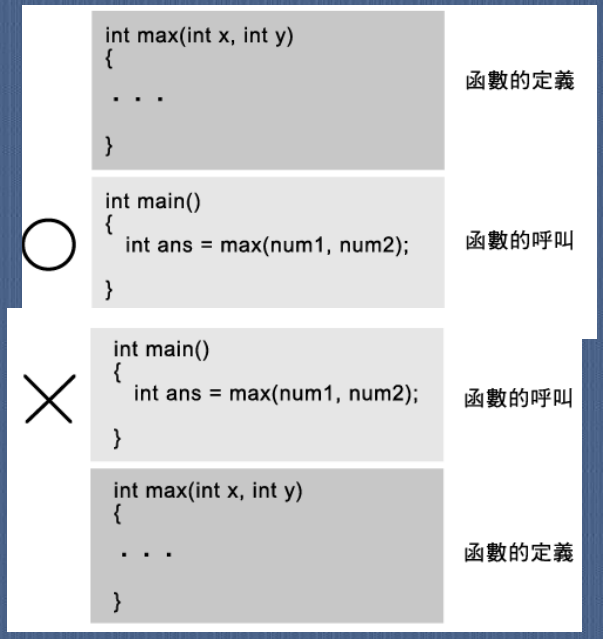
呼叫函數

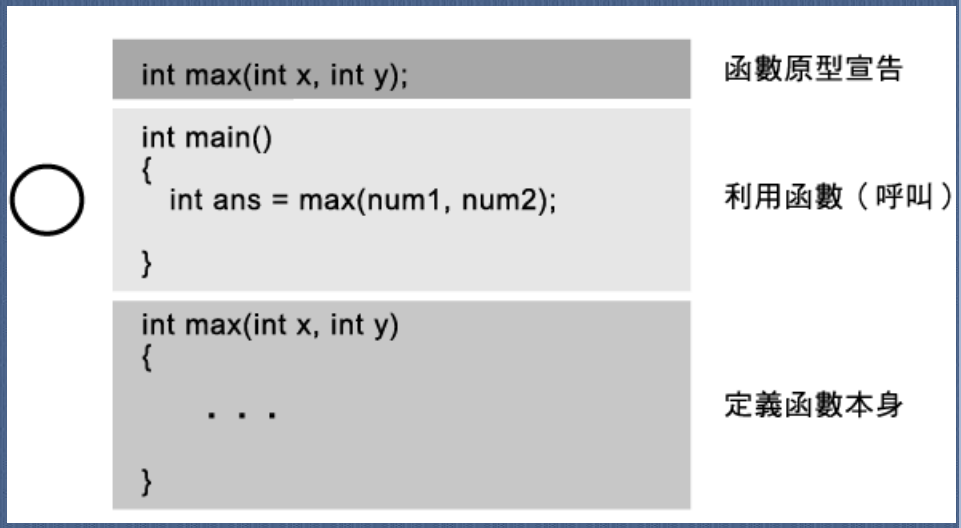
輸出傳回值

函數的宣告

- 通常在呼叫函數前，都必須先定義函數才不會出錯，若想將函數的定義放在呼叫之後的話，就必須進行函數原型宣告。
- 函數原型宣告之語法：

  傳回值的型態　函數名稱（引數列表）；





Sample10.cpp ▶ 使用函數原型宣告

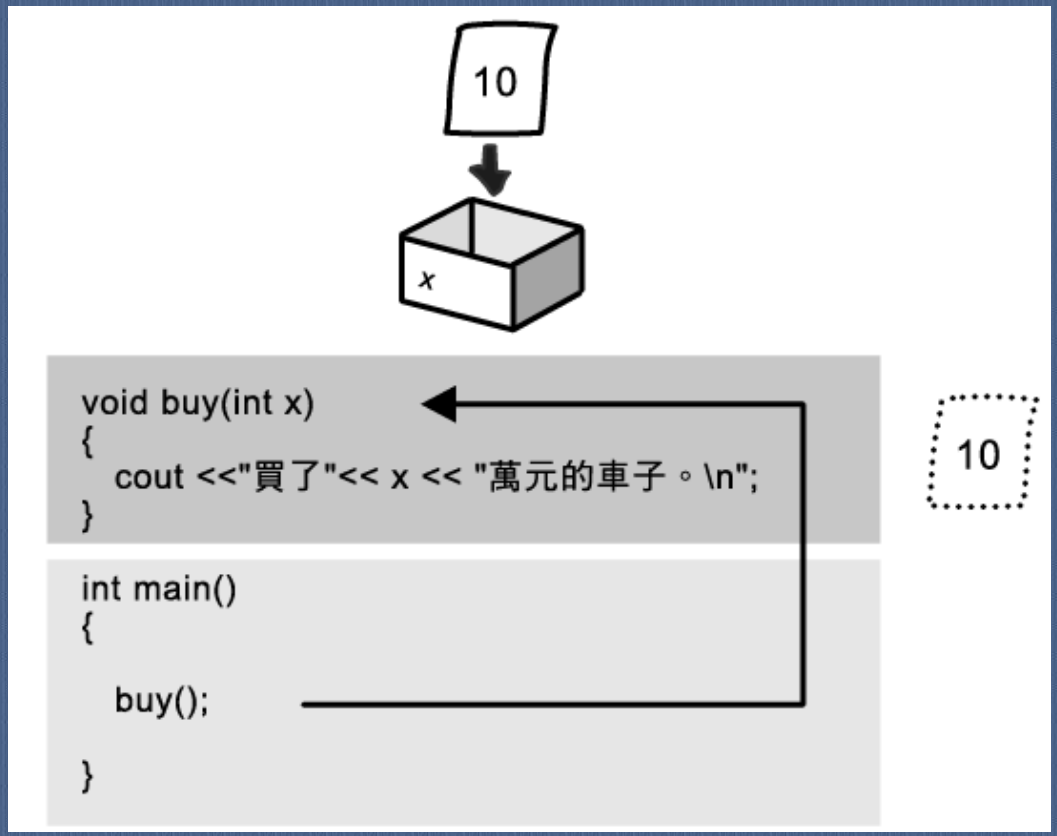
```
#include <iostream>
using namespace std;

//max 函數的宣告
int max(int x, int y);

//max 函數的呼叫
int main()
{
    int num1, num2, ans;

    cout << "請輸入第 1 個整數：\n";
    cin >> num1;
    cout << "請輸入第 2 個整數：\n";
    cin >> num2;

    ans = max(num1, num2);

    cout << "最大值為" << ans << "。\n";

    return 0;
}

//max 函數的定義
int max(int x, int y)
{
    int(x > y)
        return x;
    else
        return y;
}
```

這是函數原型宣告

呼叫函數

可以寫在函數的定義之後

- 使用預設引數
  - 如果在函數中事先指派好預設引數，當不指定引數時，就會自動使用預設值。
  - 預設引數必須由右開始依序設定。



- 範例：

```
//buy函數的宣告
void buy(int x=10);

//buy函數的呼叫
int main()
{
   cout <<"第1次以100萬元買入。\n";
   buy(100);

   cout <<"第2次以預設價格買入。\n";
   buy();

   return 0;
}

//buy函數的定義
void buy(int x)
{
   cout << "買了"<< x <<"萬元的車子。\n";
}
```

指派引數後呼叫

不指派引數呼叫

7-8 函數範本

- 理解函數範本（template）的機制
  - C++中有一項方便的功能，就是可以準備函數的「原型」，而這裡說的原型稱為函數範本（template）。
  - 函數範本可快速建立只有型態不同，但處理相同的函數。
  - 函數範本必須依照下列步驟來使用：

    1.宣告、定義函數範本

    2.呼叫函數（自動產生函數）

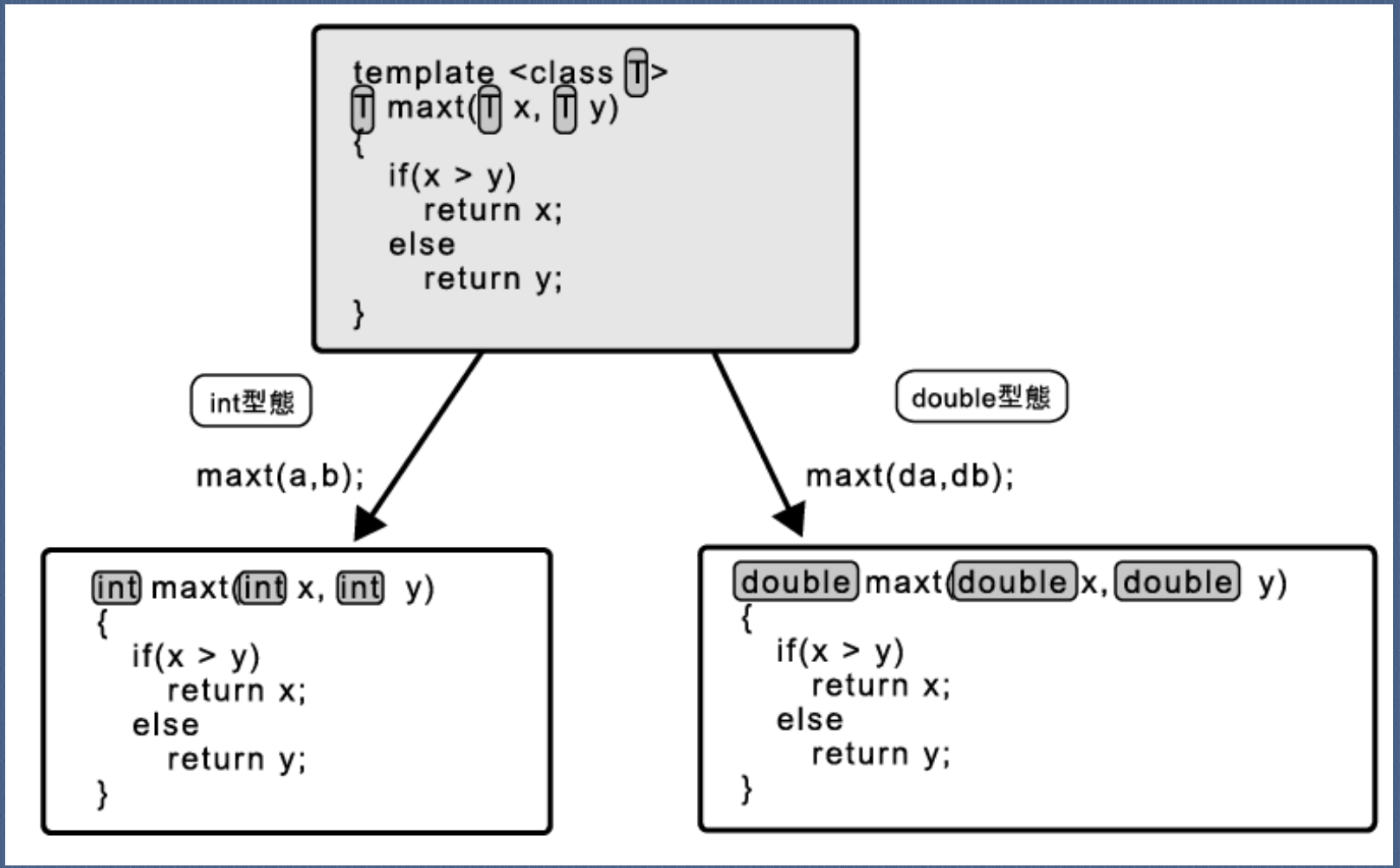
- 函數範本的宣告與定義：

  template <class範本引數的列表>

  函數的宣告或定義
- 範本引數通常是使用T這種假的型態名稱，來取代具體的型態名稱。
- 只要設定了函數範本，在編譯時就會依放入函數中的值的型態，來呼叫該型態的函數。
- 使用函數範本便能將處理的型態不同，但其他部份完全相同的函數，整合在一段程式碼中。

```
template <class T>
T maxt(T x, T y)
{
   if(x > y)
      return x;
   else
      return y;
}
```

- 利用函數範本：


template <class T>
T maxt(T x, T y)
{
if(x > y)
return x;
else
return y;
}
int型態
double型態
maxt(a,b);
maxt(da,db);
int maxt(int x, int y)
{
if(x > y)
return x;
else
return y;
}
double maxt(double x, double y)
{
if(x > y)
return x;
else
return y;
}

Sample13.cpp ▶ 使用函數範本

```
#include <iostream>
using namespace std;

//函數範本
template <class T>
T maxt(T x, T y)
{
    if(x > y)
        return x;
    else
        return y;
}
//函數範本的使用
int main()
{
    int a, b;
    double da, db;

    cout << "請輸入2個整數：\n";
    cin >> a >> b;

    cout << "請輸入2個小數：\n";
    cin >> da >> db;

    int ans1 = max(a, b);
    double ans2 = max(da, db);

    cout << "整數值的最大值為" << ans1 << "。\n";
    cout << "小數值的最大值為" << ans2 << "。\n";

    return 0;
}
```

這是範本引數

呼叫在範本引數中放入int型態的函數

呼叫在範本引數中放入double型態的函數

綜合整理

- 本章學習過的內容與重點
  - 可以將一連串的處理整合成為函數，然後呼叫出來使用。
  - 可以將引數傳遞到函數本身，然後進行處理。
  - 可以從函數本身接收傳回值。
  - 執行簡單處理的函數，可以設為inline函數。
  - 利用函數原型宣告，可以告知編譯器，函數的規格。
  - 引數可以指定預設值。