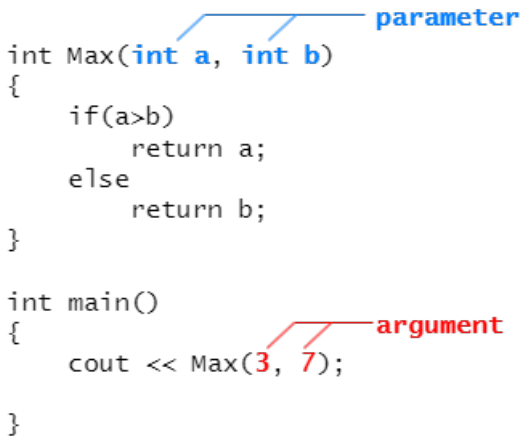
6-3 傳值呼叫 與 傳參考呼叫

參數與引數

在提到函數與呼叫使用函數時，我們會用到 **參數(parameter)** 和 **引數(argument)** 這兩個名詞。

我們可以簡單的用這張圖來區分他們。

- **參數(parameter)** 是在定義函數時，用來承接傳入資料的變數。
- **引數(argument)** 是在呼叫使用函數時，傳入的資料。



然而在大多數的情況下，大家並不會區分的那麼清楚，很多時候我們都會用 **參數** 來意指兩者。在後續的內容裡除非特別需要指出其不同，否則我們會使用 **參數** 這個詞。

傳值呼叫(call by value)

在叫用函數時，我們通常都會傳入數個參數給該函數，例如底下這個求等差數列第 n 項的函數 An( )。

```
int An(int a, int d, int n)
{
  return a+(n-1)*d;
}

int main()
{
  cout << An(1, 2, 10) << endl;  // 19
  cout << An(2, 3, 5) << endl;   // 14

  return 0;
}
```

你可以這樣想像，在第 9 行叫用 An(1, 2, 10) 的時候

1. An 函數產生了 a, d, n 這三個變數，用來承接傳入的引數
2. a 接收到了 1, d 接收到了2, n 接收到了 10
3. 回傳 a+(n-1)*d 的運算結果
4. An 函數之前產生的 a, d, n 三個變數消滅不再存在
5. 返回叫用函數的地方(第9行)，繼續執行下去。

當第 10 行叫用 An(2, 3, 5) 的時候，以上流程會再發生一次。請注意 2 個重點：

1. a, d, n 都是區域變數，當 An() 被叫用時會產生一份區域變數，返回時這些區域變數就會消滅。
2. 叫用 An( ) 時，參數的「值」被複製一份給 a, d, n。所以我們叫它 **傳「值」呼叫 (call by value)**。

接下來這個 exchange 函數會讓你把這個機制的第2個重點看得更清楚。

```
void exchange(int a, int b)
{
  int t = a;
  a = b;
  b = t;
}

int main()
{
  int a = 3;
  int b = 5;

  exchange(a, b);

  cout << "a = " << a << endl;  // a = 3
  cout << "b = " << b << endl;  // b = 5

  return 0;
}
```

第 13 行叫用 exchange(a, b)時，在 main( ) 裡的 a, b 和 exchange( ) 裡的 a, b 是互不相關的。

外面的(main的) a, b 只是把它當下的值複製一份傳給**裡面的**(exchange的) a, b。

在函數裡的 a, b 在 t 的協助下互相交換其值，並且在離開函數回到 main 裡繼續執行前，函數裡的 a, b, t 都消滅了。

函數結束回到 main 裡，接著用 cout 輸出 a, b，這個被輸出的是 main 的 a, b。由於剛才互相交換值的是 exchange 函數內的 a, b，和現在 main 的 a, b 一點關係都沒有，所以輸出的 a 還是 3，b 還是 5。

傳參考呼叫(call by referance)

如果我們真的需要一個函數，能夠幫我們把外面的兩個變數值交換，必須使用 **「傳參考呼叫(call by reference)」**。

唯一不同的地方是在函數的參數列裡，把要被傳入的變數前面加上 `&`。

```
void exchange(int &a, int &b)
{
  int t = a;
  a = b;
  b = t;
}

int main()
{
  int a = 3;
  int b = 5;

  exchange(a, b);

  cout << "a = " << a << endl;  // a = 5
  cout << "b = " << b << endl;  // b = 3

  return 0;
}
```

使用傳參考時，你可以想像外面的變數真的被傳進去了，你在函數裡對它做什麼，實際上真的會作用在外面的變數上。

你也會看到有人會這麼描述傳參考呼叫 **「參考就是別名(alias)」**。用下面這個例子比較容易理解這個別名的概念。

我們把傳入的`引數 a` 取個`別名叫 c`，把傳入的 `引數b` 取個 `別名叫 d`。於是在函數裡提到的 c 實際上就是外面的 a，在函數裡提到的 d 實際上就是外面的 b。

```
void exchange(int &c, int &d)
{
```

```
  int t = c;
  c = d;
  d = t;
}

int main()
{
  int a = 3;
  int b = 5;

  exchange(a, b);

  cout << "a = " << a << endl;  // a = 5
  cout << "b = " << b << endl;  // b = 3

  return 0;
}
```

Revision #17
★Created 3 June 2024 02:43:47 by huihui
Updated 1 August 2024 13:00:41 by huihui