國立屏東大學 資訊工程學系 物件導向程式設計

16. 函式模板

有時我們可以會為不同型態的資料設計不同的函式，例如：

```
void swap2Int(int &a, int &b)
{
   int temp;
   temp=a;
   a=b;
   b=temp;
}

void swap2Double(double &a, double &b)
{
   double temp;
   temp=a;
   a=b;
   b=temp;
}
```

這種做法在使用函式時，必須視所要處理的資料之型態的差異，選擇呼叫對應的版本。因此，我們可以使用Function Overloading的方式，將不同版本(但通常功能相同)的函式改以「同名異式」的方式設計如下：

```
void swap2Num(int &a, int &b)
{
   int temp;
   temp=a;
   a=b;
   b=temp;
}

void swap2Num(double &a, double &b)
{
   double temp;
   temp=a;
   a=b;
   b=temp;
}
```

如此一來，在使用時就不必依據不同的資料型態使用不同的函式，但若是要支援更多的型態，就必須提供

更多版本的實作。

函式模板(Function Template)則可以針對這種情況，設計一個版本來滿足多種不同的資料型態需求！請參考以下的程式碼：

```
template<class T>
void swap2Num(T &a, T &b)
{
   T temp;
   temp=a;
   a=b;
   b=temp;
}
```

Function Template使用template <class T>做為前綴(template prefix)代表將T視為某種型態；未來當swap2Num()函式被呼叫時，將會視所傳入的參數決定T究竟是何種型態，並將函式內容中所出現的T以該型態進行代換。以下是一個完整的範例：

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

template<class T>
void swap2Num(T &a, T &b)
{
   T temp;
   temp=a;
   a=b;
   b=temp;
}

int main()
{
   int x=4, y=6;
   double a=3.123, b=2.342;
   swap2Num(x, y);
   swap2Num(a, b);
}
```