Template(様板)

Sprout 2019

std :: swap (a , b) ;

- 交換A和B的程式

```
#include<iostream>
int main(){
    int a=5,b=3;
    std::cout<<"a = "<<a<<" b = "<<b<<"\n";
    std::swap(a,b);
    std::cout<<"a = "<<a<<" b = "<<b<<"\
    return 0;
}
```

```
a = 5 b = 3
a = 3 b = 5

--------------------
Process exited after
請按任意鍵繼續 . . .
```

請實作swap

- a, b 都是 int?

```
void swap(int &a,int &b){
    int tmp=a;
    a=b;
    b=tmp;
    return;
}
```

蛋糕一塊~~~

請實作swap

- a, b 都是 double?

```
void swap(double &a,double &b){
    double tmp=a;
    a=b;
    b=tmp;
    return;
}
```

Hello? 認真嗎?

請實作swap

- a, b 可能都是int或都是double?

```
void swap(int &a,int &b){
    int tmp=a;
    a=b;
    b=tmp;
}
void swap(double &a,double &b){
    double tmp=a;
    a=b;
    b=tmp;
}
```

不好說...這樣如何?

請實作swap

- a, b 可能同時是int或float或double或char或string或…?
- a, b 可能同時是任意資料型態?
- a, b 可能不同時是任意資料型態?

template

- 資料型態參數化
- 可分為函式樣板和類別樣板
- 編譯器自動產生需要的程式碼(函式)
- 避免過度重複，程式較小、較簡潔

以 template 實作 swap

```
template <typename T>
void swap(T &a,T &b){
    T tmp=a;
    a=b;
    b=tmp;
    return;
}
```

```
template   < typename  T  >
```

↑樣板宣告啟始關鍵字

```
void  swap  ( T &a , T &b )


{
    T  tmp = a ;
    a = b ;
    b = tmp ;
}
```

```
template    < typename  T  >
                    ↑様板参數
void  swap  ( T &a , T &b )


{
    T  tmp = a ;
    a = b ;
    b = tmp ;
}
```

```
template    < typename  T  >
```

↑泛用型態關鍵字，or class

```
void  swap  ( T &a , T &b )


{
    T  tmp = a ;
    a = b ;
    b = tmp ;
}
```

```
template    < typename  T  >
```

↑型態(代表)名稱

```
void  swap  ( T &a , T &b )


{
    T  tmp = a ;
    a = b ;
    b = tmp ;
}
```

```
template  < typename  T  >

void  swap  ( T &a , T &b )
 ↑函式部分
{
    T  tmp = a ;
    a = b ;
    b = tmp ;
}
```

```
template   < typename  T  >

void  swap  ( T &a , T &b )
{
    T  tmp = a ;
    a = b ;
    b = tmp ;
}
```

↑函式輸入參數，
整個函數都可以使用T

```
template   < typename  T  >

void  swap  ( T &a , T &b )

{
    T  tmp = a ;
    a = b ;
    b = tmp ;
}
```

← 函式內容

另一個例子

```
template <typename T>
T maxValue(T a,T b){
    if(a>b) return a;
    else return b;
}
```

Warning: 如果型態沒有定義 “>” (如 struct)會產生error

也可以宣告多個樣板參數

```
template <typename T1,typename T2,typename T3>
T3 name(T1 a,T2 b){
    //somthing you want to do
}
```

函式宣告與實作分離

```
#include<iostream>
//宣告
template <typename T>
T maxValue(T a,T b);
//實作
template <typename T>//記得加這行
T maxValue(T a,T b){
    if(a>b) return a;
    else return b;
}
```

template + struct

```
#include<iostream>
//宣告
template <typename T>
struct Node{
    T data;
    Node* next;
};
//使用
int main(){
    Node<int>* list=new Node<int>;
    list->data=1;
    list->next=nullptr;
    return 0;
}
```