Lab of Object-Oriented Programming: Template

黃威、陳岳紘、邱彥翔

2022

使用 moodle 點名

請登入實習課的 moodle 課程

點擊出缺席並完成今日的點名

Outline

- Function Template
- Class Template
- Exercise8

Int 的加法

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

int addInt(int x, int y){
    return (x + y);
}

int main() {
    cout << "3 + 5 = " << addInt(3,5) << endl;
    return 0;
}

```

Float 的加法

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

float addFloat(float x, float y){
    return (x + y);
}

int main() {
    cout << "3.8 + 5.1 = " << addFloat(3.8,5.1) << endl;
    return 0;
}
```

Int & Float的加法

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

int addInt(int x, int y){
    return (x + y);
}

float addFloat(float x, float y){
    return (x + y);
}

int main() {
    cout << "3 + 5 = " << addInt(3,5) << endl;
    cout << "3.8 + 5.1 = " << addFloat(3.8,5.1) << endl;

    return 0;
}
```

Int & Float的sort

```cpp
void sortInt(int* arr, int len) {
    for (int i = 0; i < len-1; ++i) {
        for (int j = 0; j < len-i-1; ++j)
            if (arr[j] > arr[j+1]){
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
    }
    // print_array
    for (int i = 0; i < len; i++){
        cout<< arr[i] <<" ";
    }
    cout<<endl;
}
```

```cpp
void sortFloat(float* arr, int len) {
    for (int i = 0; i < len-1; ++i) {
        for (int j = 0; j < len-i-1; ++j)
            if (arr[j] > arr[j+1]){
                float temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
    }
    // print_array
    for (int i = 0; i < len; i++){
        cout<< arr[i] <<" ";
    }
    cout<<endl;
}
```

Template

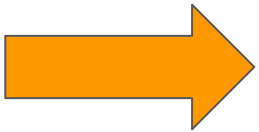
- 將資料型態參數化(Parameterized Types)
- 資料型態用簡化的名稱代替 (ex: T 或任何你想命名的名稱), 在編譯過程會將該名稱 (T) 替換成原來的資料型態
- Template 種類
  - Function Template
  - Class Template

Function Template

- 適用於不同資料型態的參數，但實作內容相同的函式
- 避免相同過程、資料型態不同的函式需宣告並定義好幾次

Function Template (add)

```
4  int addInt(int x, int y){
5      return (x + y);
6  }
7
8  float addFloat(float x, float y){
9      return (x + y);
10 }
```

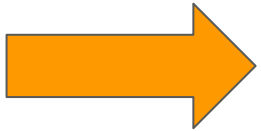


```
20 template <typename T>
21 T add (T x, T y){
22     return (x + y);
23 }
```

https://onlinegdb.com/ot8ylmBqL

Function Template (sort)

```
25 void sortInt(int* arr, int len) {
26     for (int i = 0; i < len-1; ++i) {
27         for (int j = 0; j < len-i-1; ++j)
28             if (arr[j] > arr[j+1]){
29                 int temp = arr[j];
30                 arr[j] = arr[j+1];
31                 arr[j+1] = temp;
32             }
33     }
34     // print_array
35     for (int i = 0; i < len; i++){
36         cout<< arr[i] <<" ";
37     }
38     cout<<endl;
39 }
40
41 void sortFloat(float* arr, int len) {
42     for (int i = 0; i < len-1; ++i) {
43         for (int j = 0; j < len-i-1; ++j)
44             if (arr[j] > arr[j+1]){
45                 float temp = arr[j];
46                 arr[j] = arr[j+1];
47                 arr[j+1] = temp;
48             }
49     }
50     // print_array
51     for (int i = 0; i < len; i++){
52         cout<< arr[i] <<" ";
53     }
54     cout<<endl;
55 }
```



```
57 template <typename T>
58 void sort(T* arr, int len) {
59     for (int i = 0; i < len-1; ++i) {
60         for (int j = 0; j < len-i-1; ++j)
61             if (arr[j] > arr[j+1]){
62                 T temp = arr[j];
63                 arr[j] = arr[j+1];
64                 arr[j+1] = temp;
65             }
66     }
67     // print_array
68     for (int i = 0; i < len; i++){
69         cout<< arr[i] <<" ";
70     }
71     cout<<endl;
72 }
```

Function Template

- 以型態定義函式，將型態當作參數傳給template

  template <typename T>

  template <class T>

- 以關鍵字 class / typename 表示泛用型態，宣告多個型態：template <typename T1, typename T2, typename T3>

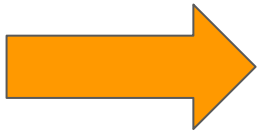
Function Template

當函式宣告及定義分開時，兩邊都要各寫一次template

```
17  template <typename T>
18  T add (T x, T y);
19
20  // template <typename T>
21  T add (T x, T y){
22      return (x + y);
23  }
```

```
main.cpp:21:1: error: 'T' does not name a type
   21 | T add (T x, T y){
      | ^
```

ERROR !



```
17  template <typename T>
18  T add (T x, T y);
19
20  template <typename T>
21  T add (T x, T y){
22      return (x + y);
23  }
```

Function Template

宣告不同函式，也都要各寫一次template

```
17  template <typename T>
18  T add (T x, T y);
19
20  template <typename T>
21  T add (T x, T y){⟺}
24
25  // template <typename T>
26  void sort(T* arr, int len) {⟺}
```

```
main.cpp:26:11: error: 'T' was not declared in this scope
```

Function Template

宣告不同函式，也都要各寫一次template

```
17  template <typename T>
18  T add (T x, T y);
19
20  template <typename T>
21  T add (T x, T y){⟺}
24
25  template <typename T>
26  void sort(T* arr, int len) {
27      for (int i = 0; i < len-1; ++i) {
28          for (int j = 0; j < len-i-1; ++j)
29              if (arr[j] > arr[j+1]){
30                  T temp = arr[j];
31                  arr[j] = arr[j+1];
32                  arr[j+1] = temp;
33              }
34      }
35      // print_array
36      for (int i = 0; i < len; i++){
37          cout<< arr[i] <<" ";
38      }
39      cout<<endl;
40  }
```

Function Template

使用 T 做函示宣告及定義時，T 是由自己命名，本身表示相同的資料型態。(不同資料型態會error)

```
    cout << "3 + 5 = " << add(3,5) << endl;
    cout << "3.8 + 5.1 = " << add(3.8,5.1) << endl;
    cout << "3 + 5.1 = " << add(3,5.1) << endl; //error
```

Class Template

語法和function template 相同

```
4  template <typename T>
5  class Tclass{
6  public:
7      Tclass(int);
8      ~Tclass();
9
10     T get(int);
11     void set(int, T);
12 private:
13     int size;
14     T* array;
15 };
```

```
17 template <typename T>
18 Tclass<T>::Tclass(int s){
19     size = s;
20     array = new T[size];
21 }
22
23 template <typename T>
24 Tclass<T>::~Tclass(){
25     delete[] array;
26 }
27
28 template <typename T>
29 T Tclass<T>::get(int i){
30     return array[i];
31 }
32
33 template <typename T>
34 void Tclass<T>::set(int i, T v){
35     array[i] = v;
36 }
```

Class Template

語法和function template 相同

```
4  template <typename T>
5  class Tclass{
6  public:
7      Tclass(int);
8      ~Tclass();
9
10     T get(int);
11     void set(int, T);
12 private:
13     int size;
14     T* array;
15 };
```

```
38 int main(){
39     Tclass<int> intArr(10);
40     Tclass<float> floatArr(5);
```

https://onlinegdb.com/n2XmY_Tvc

Class Template

多型態的class template

```cpp
#include <iostream>
#include <string>

using namespace std;

template <typename T, int N>
class Array
{
private:
    T m_Array[N];
public:
    int GetSize() const { return N;}
};

int main(){
    Array<int, 5> array;
    cout << array.GetSize() << endl;

    return 0;
}
```

Exercise 8

回『基礎題庫』

a310: 模板加法

內容 ：
使用template實作不同型態的資料加法

請使用Template實作，沒有使用的本次作業成績*0.5

輸入說明

測資會有int, float, char三種不同的資料型態，並回傳兩個值的和

測資形式如下：

data_type x data_type y

第一、三個為資料型態，第二、四個為欲相加的數字

當data_type為-1時結束程式

輸出說明

輸出兩個值之和，無條件進位至小數點下第一位

0則輸出0，不需輸出小數點

範例輸入

```
int 4 float 6.1
char 3 float 4.7
-1
```

範例輸出

```
Sum: 10.1
Sum: 7.7
```

Any questions?