C++ Programming

Function Templates

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

What is the problem here?

```
4  int max(int a, int b) {
5      if (a > b)
6          return a;
7      return b;
8  }
9
10 double max(double a, double b) {
11     if (a > b)
12         return a;
13     return b;
14 }
15
16 int main() {
17     cout << max(2, 5) << "\n";       // 5
18     cout << max(2.5, 5.4) << "\n";   // 5.4
19
```

- These 2 functions do **EXACTLY the same logic**, regardless of data type
- So very tedious copy paste!
- Also if we discovered a bug in one function, we have to solve it in all of the different copies?
- It will be severe if we have to make like e.g. 6 functions, with copy paste and just data type change

Generic Programming using **Templates**

```
 4  // Hey compiler this is template. Use Type
 5  template<typename Type>
 6  Type MyMax(Type a, Type b) {
 7      if (a > b)
 8          return a;
 9      return b;
10  }
11
12  struct Employee {};
13
14  int main() {
15      cout << MyMax(2, 5) << "\n";                // 5: Guessed as int
16      cout << MyMax<int>(2, 5) << "\n";           // 5
17      cout << MyMax<double>(2.5, 5.4) << "\n";    // 5.4
18      cout << MyMax('A', 'X') << "\n";            // X
19
20      //cout << MyMax(2, 5.4) << "\n";            // CE: Can't guess
21      cout << MyMax<int>(2, 5.4) << "\n";         // 5
22      cout << MyMax<double>(2, 5.4) << "\n";      // 5.4
23
24      Employee a, b;
25      //cout << MyMax<Employee>(a, b) << "\n";    // CE: can't compare
26
```

Generation

- Different versions of function MyMax are **generated** on compile time using the template
  - Compiler generates ONLY based on the used cases
    - E.g. in previous code it generates 3 functions:
    - Int MyMax(int, int) / double MyMax(double, double) / char MyMax(char, char)
- I did not name function max, I used MyMax
  - Max is defined internally, it will cause error. Try it
- In practice:
  - Might be complex and hard to apply
  - Error messages: Hard to understand and ugle. Try to generate mistakes :(

More

```
4  template<class Type1, class Type2>
5  Type1 sum(Type1 a, Type2 b) {
6      Type1 r = a + b;
7      return r;
8  }
9
10 int main() {
11     cout << sum(1, 10) << "\n";          // 11
12     cout << sum(1, 10.5) << "\n";        // 11
13     cout << sum(1.2, 10.5) << "\n";      // 11.7
14     cout << sum(1.2, 10) << "\n";                // 11.2
15     cout << sum<int, int>(1.2, 10) << "\n";      // 11
16     cout << sum('A', 1) << "\n";         // B
17     cout << sum(1, 'A') << "\n";         // 66
18
19     //cout << sum("I am", "Mostafa") << "\n";        // CE: char*
20     cout << sum(string("I am "), string("Mostafa")) << "\n";    // I amMostafa
21
```

Static variable

```
 4 int global_var = 0;
 5
 6 template<typename T>
 7 void increment_me(T x) {
 8     static int i = 0;
 9     cout << ++i << " " << ++global_var << "\n";
10     return;
11 }
12
13 int main() {
14     // One static variable for each generated function
15
16     increment_me(5);    // 1 1
17     increment_me(5);    // 2 2
18     increment_me(5);    // 3 3
19
20     increment_me(2.4);  // 1 4
21     increment_me(2.4);  // 2 5
22     increment_me(2.4);  // 3 6
23
```

Function template specialization

- What if a **specific data type** should be handled differently?

```
4  template<class T>
5  T add(T a, T b) {
6      return a + b;
7  }
8
9  template<class T>
10 T multiply(T a, int factor) {
11     return a * factor;
12 }
13
14 template<>
15 string multiply(string str, int factor) {
16     // we can't multiply strings
17     string res = "";
18
19     while (factor--)
20         res += str;
21
22     return res;
```

```
25 int main() {
26     string s = "Magic";
27
28     cout<<add(10, 4)<<"\n";          // 14
29     cout<<add(s, s)<<"\n";           // MagicMagic
30
31     cout<<multiply(10, 4)<<"\n";     // 40
32     cout<<multiply(s, 4)<<"\n";      // MagicMagicMagicMagic
33
34     return 0;
35 }
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."