An Idiot's Guide to C++ Templates - Part 1

Alex Sherbakov
Alex Sherbakov
Joined 27 Jun 2008 Location United States

Posted 5 Nov 2008 Updated 5 Nov 2008 Views 1,000,000 Bookmarked 1,000 times

Creating the Map<Key,Value> class in C++ using templates.

Prologue

Table of Contents
- The Idea Behind Templates
- Function Templates
  - Without Templates and Using a Lot of Functions
  - Function Types and Function Templates
  - Function Template Instantiation
  - Function Template Arguments
  - Function Template Argument Specifications
  - Function Template Argument Deduction
- Class Templates
  - Function Types and Function Templates
  - Class Templates Instantiation
  - Class Template Arguments
  - Class Template Argument Specifications
  - Class Template Argument Deduction

The Idea Behind Templates

Templates are a way of making your classes more flexible by allowing them to work with different types. This saves you a great deal of time.

Let's pretend that I am a programmer who needs to write a function that prints double values to the console:

```
void PrintMyNumber(double);

int main()
{
    PrintMyNumber(1.0);
    PrintMyNumber(2.0);
    PrintMyNumber(3.0);
}
```

Interesting. Although I don't think this is very useful, I guess I could write a function that prints integers to the console:

```
void PrintMyNumber(int);

int main()
{
    PrintMyNumber(1);
    PrintMyNumber(2);
    PrintMyNumber(3);
}
```

Wait a minute! I can do that with both doubles and integers, because I can write two functions with the same name, but with different parameters. Cool! Now I can write a function that prints both doubles and integers:

```
void PrintMyNumber(double);
void PrintMyNumber(int);

int main()
{
    PrintMyNumber(1.0);
    PrintMyNumber(2.0);
    PrintMyNumber(3.0);
    PrintMyNumber(1);
    PrintMyNumber(2);
    PrintMyNumber(3);
}
```

Hmm, I guess I can do that, but what if I want to print a character? Or a string? Or a date? Or a time? Or a complex number? Or a matrix? Or a vector? Or a pointer? Or a function? Or a class? Or a class that I've created myself? Do I have to write a function for each and every type I want to print?

Yes, I do. Or I can use a template.

Templates are a way of making your classes more flexible by allowing them to work with different types. This saves you a great deal of time.

Let's pretend that I am a programmer who needs to write a function that prints double values to the console:

```
template<typename T>
void PrintMyNumber(T);

int main()
{
    PrintMyNumber<double>(1.0);
    PrintMyNumber<double>(2.0);
    PrintMyNumber<double>(3.0);
}
```

Wait a minute! I can do that with both doubles and integers, because I can write one function with a template parameter. Cool! Now I can write a function that prints both doubles and integers:

```
template<typename T>
void PrintMyNumber(T);

int main()
{
    PrintMyNumber<double>(1.0);
    PrintMyNumber<double>(2.0);
    PrintMyNumber<double>(3.0);
    PrintMyNumber<int>(1);
    PrintMyNumber<int>(2);
    PrintMyNumber<int>(3);
}
```

Hmm, I guess I can do that, but what if I want to print a character? Or a string? Or a date? Or a time? Or a complex number? Or a matrix? Or a vector? Or a pointer? Or a function? Or a class? Or a class that I've created myself? Do I have to write a function for each and every type I want to print?

No, I don't. Because I can write one function that prints anything!

Templates are a way of making your classes more flexible by allowing them to work with different types. This saves you a great deal of time.

Let's pretend that I am a programmer who needs to write a function that prints anything to the console:

```
template<typename T>
void PrintMyNumber(T);

int main()
{
    PrintMyNumber<double>(1.0);
    PrintMyNumber<double>(2.0);
    PrintMyNumber<double>(3.0);
    PrintMyNumber<int>(1);
    PrintMyNumber<int>(2);
    PrintMyNumber<int>(3);
    PrintMyNumber<char>('a');
    PrintMyNumber<char>('b');
    PrintMyNumber<char>('c');
    PrintMyNumber<char*>("Hello");
    PrintMyNumber<char*>("Goodbye");
    PrintMyNumber<char*>("Hello again");
}
```

Wait a minute! I can do that with doubles, ints, characters, and strings! Cool! Now I can write a function that prints anything!

Yes, I can. Or I can use a template.

Function Types and Function Templates

Without Templates and Using a Lot of Functions

Let's say I want to write a function that prints the sum of two integers. Easy peasy:

```
int PrintMyNumber(int, int);

int main()
{
    PrintMyNumber(1, 2);
    PrintMyNumber(3, 4);
    PrintMyNumber(5, 6);
}

int PrintMyNumber(int, int)
{
    int sum = 0;
    sum = sum + 1;
    sum = sum + 2;
    return sum;
}
```

Output:

```
3
7
11
```

Hmm, I guess I can do that, but what if I want to print the sum of two doubles? Or two characters? Or two strings? Or two dates? Or two times? Or two complex numbers? Or two matrices? Or two vectors? Or two pointers? Or two functions? Or two classes? Or two classes that I've created myself? Do I have to write a function for each and every type I want to print?

No, I don't. Because I can write one function that prints anything!

Function Types and Function Templates