CST209

Object-oriented Programming C++

(Week 11)

Dr. Teo Bee Guan

Content

- Function Templates
- Class Templates

Function Templates

- Overloaded functions make programming convenient because only one function name must be remembered for a set of functions that perform similar operations.

```
int square(int number)
{
     return number * number;
}
double square(double number)
{
     return number * number;
}
```

Function Templates

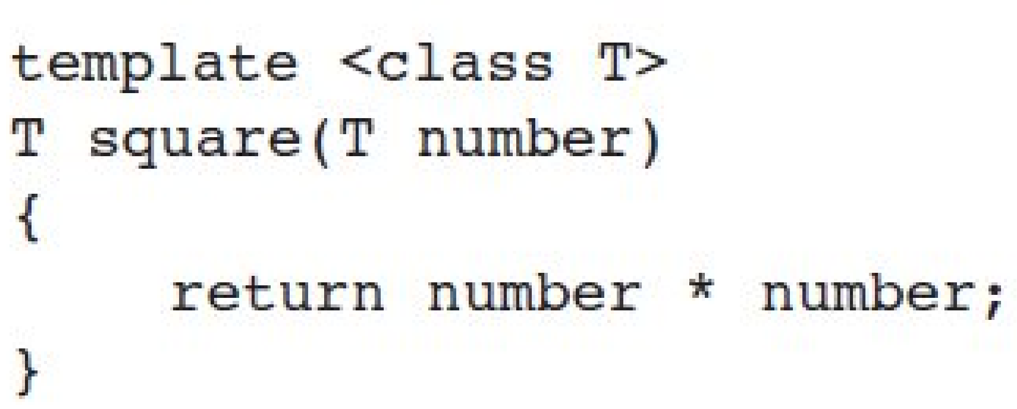
- Each of the functions, however, must still be written individually, even if they perform the same operation.

- The only differences between these two functions are the data types of their return values and their parameters. In situations like this, it is more convenient to write a **function template** than an overloaded function.

Function Templates

- Function templates allow us to write a single function definition that works with many different data types, instead of having to write a separate function for each data type used.

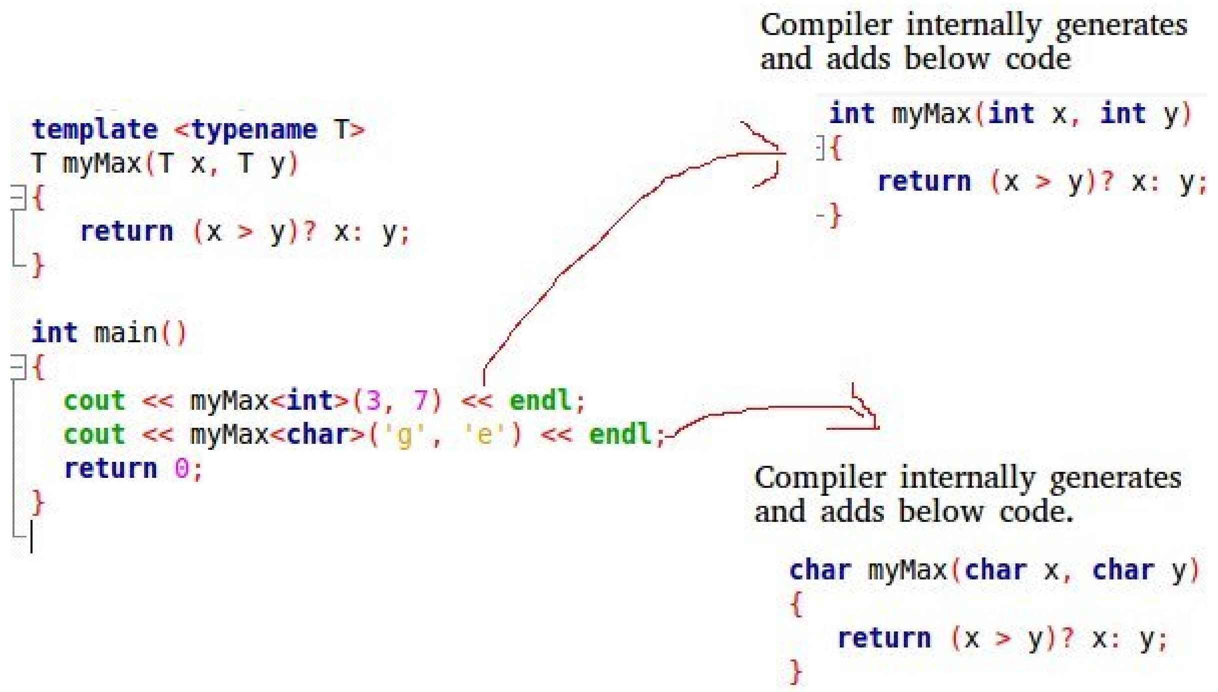
- A function template is not an actual function, but a “mold” the compiler uses to generate one or more functions.



```
template <class T>
T square(T number)
{
    return number * number;
}
```

Function Templates

- When writing a function template, we do not have to specify actual types for the parameters, return value, or local variables.
- Instead, we use a type parameter to specify a generic data type.
- When the compiler encounters a call to the function, it examines the data types of its arguments and generates the function code that will work with those data types.



```cpp
template <typename T>
T myMax(T x, T y)
{
    return (x > y)? x: y;
}

int main()
{
   cout << myMax<int>(3, 7) << endl;
   cout << myMax<char>('g', 'e') << endl;
   return 0;
}
```

Compiler internally generates and adds below code

```cpp
int myMax(int x, int y)
{
    return (x > y)? x: y;
}
```

Compiler internally generates and adds below code.

```cpp
char myMax(char x, char y)
{
    return (x > y)? x: y;
}
```

In-Class Practice: Example 1

In-Class Exercise 1

Write a template function named “square” to calculate and return the square of a number.

Function Templates with Multiple Types

- More than one generic type may be used in a function template.
- Each type must have its own parameter

In-Class Practice: Example 2

Function Templates with Multiple Types

In-Class Exercise 2

Write a template function named “addition” that can work with two numbers from different data types (e.g. int & float) and return the sum of the number.

Overloading with Function Templates

- Function templates may be overloaded.

- As with regular functions, function templates are overloaded by having different parameter lists.

In-Class Practice: Example 3

Class Template

- Templates may also be used to create generic classes and abstract data types.
- Class templates allow us to create one general version of a class without having to duplicate code to handle multiple data types.
- Declaring a class template is very similar to declaring a function template. First, a template prefix, such as template<class T> , is placed before the class declaration.

In-Class Practice: Example 4, 5

Class Template

In-Class Exercise 3

Write a template class named “Calculator” that have two private member variables, number1 & number 2. The class should have four member functions to perform the addition, subtraction, multiplication, and division. These four member functions should return a result of calculation.

(Presume that the number1 and number2 share the same data type)

Test your class in the main program by creating a Calculator instance with int type and another with float type.

Class Template

In-Class Exercise 4

Write a template class named “Calculator” that have two private member variables, number1 & number 2. The class should have four member functions to perform the addition, subtraction, multiplication, and division. These four member functions should return a result of calculation.

(Presume that the number1 and number2 share the different data type (e.g. int, float))

Test your class in the main program by creating a Calculator instance with int type and another with float type.

See you next class