Object-Oriented Programming Using C++

Template

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

C++ Template

- Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type
- A template is a blueprint or formula for creating a generic class or a function
- The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept
- There is a single definition of each container, such as vector, but we can define many different kinds of vectors for example, vector <int> or vector <string>

Function Template

```
template <class type> ret-type func-name(parameter list) {
   // body of function
}
```

Example: Function Template

Template for function Maximum

```
template <typename T>
T Max (T a, T b) {
   return a < b ? b:a;
}

int main () {
   int i = 39;
   int j = 20;
   cout << "Max(i, j): " << Max(i, j) << endl;

   double f1 = 13.5;
   double f2 = 20.7;
   cout << "Max(f1, f2): " << Max(f1, f2) << endl;

   string s1 = "Hello";
   string s2 = "World";
   cout << "Max(s1, s2): " << Max(s1, s2) << endl;

   return 0;
}
```

Template for function Maximum

```
Max(i, j): 39
Max(f1, f2): 20.7
Max(s1, s2): World
```

Class Template

```
template <class type> class class-name {
    .
    .
    .
}
```

Example: Class Template

Stack Class Template

```
template <class T>
class Stack {
   private:
      vector<T> elems;     // elements

   public:
      void push(T const&);  // push element
      void pop();               // pop element
      T top() const;            // return top element

      bool empty() const {      // return true if empty.
         return elems.empty();
      }
};
```

Example: Class Template

Stack Class Template

```
template <class T>
void Stack<T>::push (T const& elem) {
   // append copy of passed element
   elems.push_back(elem);
}

template <class T>
void Stack<T>::pop () {
   if (elems.empty()) {
      throw out_of_range("Stack<>::pop(): empty stack");
   }

   // remove last element
   elems.pop_back();
}

template <class T>
T Stack<T>::top () const {
   if (elems.empty()) {
      throw out_of_range("Stack<>::top(): empty stack");
   }

   // return copy of last element
   return elems.back();
}
```

Example: Class Template

Stack Class Template

```
int main() {
   try {
      Stack<int>         intStack;  // stack of ints
      Stack<string> stringStack;    // stack of strings

      // manipulate int stack
      intStack.push(7);
      cout << intStack.top() <<endl;

      // manipulate string stack
      stringStack.push("hello");
      cout << stringStack.top() << std::endl;
      stringStack.pop();
      stringStack.pop();
   } catch (exception const& ex) {
      cerr << "Exception: " << ex.what() <<endl;
      return -1;
   }
}
```

Example: Class Template

Template for function Maximum

```
7
hello
Exception: Stack<>::pop(): empty stack
```

Object-Oriented Programming Using C++

Template

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur