OntarioTech Science

Kourosh Davoudi
kourosh@uoit.ca

Week 9: Templates

CSCI 1061: Programming Workshop II

Learning Outcomes

In this week, we learn:

- Why templates are useful?
- How to define a function template
- How to define a class template?

Motivation

Overloading

```
void swapValues(int& var1, int& var2)
{
        int temp;
        temp = var1;
        var1 = var2;
        var2 = temp;
}
```

```
void swapValues(char& var1, char& var2)
{
        char temp;
        temp = var1;
        var1 = var2;
        var2 = temp;
}
```

Works for variable type **int**

Works for variable type **char**

Can we do the better job?

Note: the codes of two functions have identical logic!

Function Template

Type parameter

Template prefix

```
template<class T>
void swapValues(T & var1, T & var2)
{
    T temp;
    temp = var1;
    var1 = var2;
    var2 = temp;
}
```

// T used like any other type

Function Template

```
int main()
 {
      int x = 1, y = 2;

      cout << "x = " << x << ", y = " << y <<endl;

      swapValues(x, y); // swapValues(int, int) will be called

      cout << "x = " << x << ", y = " << y <<endl;

      return 0;
 }
```

Function Template

```
int main()
 {
      double x = 1.5, y = 2.3;

      cout << "x = " << x << ", y = " << y <<endl;

      swapValues(x, y); // swapValues(double, double) will be called

      cout << "x = " << x << ", y = " << y <<endl;

      return 0;
 }
```

How to create a class template?

```
// This is written for the integer pairs  !
class Pair
{
      public:
            Pair();
            Pair(int firstVal, int secondVal);
            void setFirst(int newVal);
            void setSecond(int newVal);
            int getFirst() const;
            int getSecond() const;

      private:
            int first; int second;
};
```

Class Template

Template prefix

```
template<class T>
class Pair
{
      public:
            Pair();
            Pair(T firstVal, T secondVal);
            void setFirst(T newVal);
            void setSecond(T newVal);
            T getFirst() const;
            T getSecond() const;
      private:
            T first; T second;
};
```

From: Absolute C++

How to define a member function of a class template?

Type parameter

Template prefix

```
template<class T>
Pair<T>::Pair(): first(0), second(0)
{
}
```

We need to use type parameter here

(Default Constructors)

Class Template

```
template<class T>
Pair<T>::Pair(T firstValue, T secondValue)
            :first(firstValue), second(secondValue)
{
}
```

(Constructor)

Class Template

```
template<class T>
T Pair<T>::accessFirst()
{
    return first;
}
```

(See template-class.cpp for other definitions)

How to create the objects of a class template?

```
int main()
{
  Pair<int> x;
  Pair<int> y(2,3);

  cout << x.accessFirst()<< endl;
  cout << x.accessSecond()<< endl;
  cout << y.accessFirst()<< endl;
  cout << y.accessSecond()<< endl;

  return 0;

}
```

We need the type parameter when creating the objects