# Function and Class Templates

Introduction to Programming

1/04/2016



## Template and Generic Programming

- Template is a way for C++ to enable generic programming
- With a single code segment, an entire range of related (overloaded) functions (or classes) are specified.
- Templates and template specializations
- Function Templates and Class Templates

# Function Templates

- Overloaded functions normally perform similar or identical operations on different types of data.
- Function Templates can be used to express identical operations performed on different types of data.
  - Based on the argument types, the compiler generates object code functions to handle each type.
  - In C, this can be done by using macro (`#define`)
    - However, macro does not perform type checking on arguments
    - Function Templates enable type checking
- Example: exr1.cpp



# Template Function Specializations

- The template function in the last example

```
template <typename T>
T SumArray(const T *Array,const int N)
```

- Specialized by the following expressions

```
SumArray(A,Ni)
SumArray(B,Nd)
SumArray(C,Nc)
```

- Compiler generates the following overload functions

```
int    SumArray(int *   Array,const int N)
double SumArray(double *Array,const int N)
char   SumArray(char *  Array,const int N)
```

# Function Template Definition

- Function template definition

```
template< typename T>  // followed by regular function
                       // definition assuming T is a
                       // defined type
template<class C>      // typename, class are interchangeable
template< typename T1, typename T2 > // multiple
                                     // template parameters
```

- Template parameters are used to specify
  - The types of the argument of the function
  - The return type of the function
  - The type of the local variables in the function



# Overloading Function Templates

- The function-template specializations generated from a function template all have the same name, so the compiler uses overloading resolution to invoke the proper function.
- A function template can be further overloaded by another function template with the same name but different number of parameters
- A function template can also be overloaded by providing nontemplate functions with the same name but different function arguments.

# Class Templates

- Class templates are called parameterized types
- The programmer writes a set of simple concise code for each parameter type, the compiler writes the source code for the specialization the programmer requires
- Creating Class Template

```
template <class T > // proceeds normal class declaration
                    // T is assumed to be valid type
```

- Member function definition outside of class body

```
template <class T>
return_type class_name<T> function_name( ) {} ;
```

- Nonmember function definition

```
template <class T>
return_type function_name(class_name<T> arg ...)
```

- Example: exr2.cpp



# Class Template Instantiation and Specialization

- The template class

```
template <class T>
class BArray {
    // ....
};
```

  - is instantiated by the declarations:

```
BArray<int>    iArray(10);
BArray<double> dArray(5);
BArray<char>   cArray(9);
```

  - After that, three classes exist

```
BArray<int>, BArray<double>, BArray<char>
```

# Parameterized Classes

- Class template is also known as parameterized class
- Actual class is defined with the template parameter
- Class template can take more than one parameter

```
Template <class T, class U>
class newClass {
    T data1;
    U data2;
    // ....
}
```

- And it should be specialized by multiple parameters

```
newClass<int,int>    A;
newClass<double,int> B;
```



# Nontype parameters and Default Types

- Class template can have nontype template parameters
- Nontype parameters can have default arguments and are treated as consts
- Default parameters must be trailing parameters
- Explicit class template

```
template <>
Class BArray<Complex> { ... };
```

- This is an explicit specialization and is a complete replacement for the BArray class.
- Note: Friendship can be established between a class template and a global function, a member function of another class, or even a entire class.
- Examples: exr3.cpp , exr4.cpp

# Summary

- Template and generic programming
- Function templates
  - Function template specialization
  - Function template definition
  - Overloading function templates
- Class templates
  - Class template instantiation and specialization
  - Parameterized classes
  - Nontype parameters