Advanced Programming

Templates

Topics

- Templates
  - Function Templates
  - Class Templates
  - Inheritance in Classes with Template Definition
- Examples

Templates

- Templates are a feature of C++ that allow to write a single code segment for a set of related functions, called a function template, and for a set of related classes, called a class template.
- These functions or classes are also referred to generic functions/classes

Example

- Assume a function is needed to find the maximum value in a list.

- A class is needed to represent bounded arrays

Syntax

- Using ***template<class TypeName>*** we define a place-holder for a type
- E.g.
  - template <class T>
- A function can be defined with input parameters, or return type of type T.
- Similarly, a class with member variables of type T can be defined

Function Templates

- The expression template<class TypeName> should precede the function definition.
- Function return type, parameters, or local variables can be defined in terms of TypeName

Calling Generic Functions

- The C++ compiler creates multiple versions of the generic function/class
- You can call the function in the normal way if the parameters are not ambiguous.
- It is also possible to call the function with specifying the type

Example

- Write a function to get two numbers and return the larger one.

```
template<class T1>
T1 Max(T1 a, T1 b)
{
	if( a > b )
		return a;
	else
		return b;
}
```

Example (cont.)

```
cout << Max( 1, 4);
cout << Max( 2.5, 1.7) << endl;
cout << Max( 1.0, 3);   // Ambiguous case
cout << Max<double>(1.0, 3) << endl;
```

Class Templates

- Similarly it is possible to create generic classes with template statement.

```
template<class T2>
class List
{
      int size;
      T2  *data;
       public:
       List( int s);
       T2& operator[](int index);
}
```

Example

- Define a class to store numeric values in a List. Assume the List accepts duplicate values.

Inheritance in Generic Classes

- Generic classes can be used both as base and derived classes.
- When instantiating from a generic class, the type should be specified.

Example

- Derive a class named SortedList from List class to store numeric values.
- Remember to repeat template<class TypeName> before the implementation of each member class