Template Classes
Lecture 11
Section 16.3

Robb T. Koether

Hampden-Sydney College

Fri, Feb 10, 2017

Outline

Template Classes

- To make a class a template class, write

  template <**class** T>

  before the class definition.
- The class name *Name* becomes *Name<T>* in all places except:
  - The class name in the heading.
  - The constructor names in the prototypes.
  - The destructor name in the prototype.

Template Classes

Example (Template Classes)

```cpp
template <class T>
class Vectr
{
    public:
        Vectr();
        Vectr(const Vectr<T>& v);
        ~Vectr();
        Vectr<T> add(const Vectr<T>& v) const;
            ⋮
    private:
        int mSize;
        T* element;
};
template <class T>
operator+(const Vectr<T>& v1, const Vectr<T>& v2);
```

Template Classes

Example (Instantiating Template Objects)

```
Vectr<int> v1(2, 123);
Vectr<double> v2(3, 4.56);
Vectr<Point> v3(3, Point(1, 2))
Vectr<Date> v4(4, Date("Feb", 8, 2016));
```

- Objects of the class are instantiated using

```
Name<Type> Object;
```

Outline

Implementing Member Functions

- Each member function of a template class is a template function.
- The member functions are instantiated and compiled individually *only as necessary*.
- The class definition is stored as uncompiled text until the compiler determines that it is needed.
- That determination cannot be made until the file that uses the class is compiled.

Implementing Member Functions

- Therefore, we write the member functions in the header file, to be included by the program.
- There is no separate implementation file for a template class.
- Why not?

Testing Member Functions

- To test a template, just to see if the member functions compile, each member function must be invoked.
- Further testing may be necessary to see if the member functions are correct.

A Vector of Vectors

- Is it possible to create a "vector of vectors?"

Outline

Assignment

Assignment

- Read Section 16.3.