Template Classes
Lecture 9
Section 9.3

Robb T. Koether
Hampden-Sydney College
Thu, Feb 5, 2009
Outline

1. Template Classes
2. Template Member Functions
3. A Matrix Class
4. Assignment
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 heading.
- The constructor names.
- The destructor name.
Template Classes

Example (Template Classes)

```cpp
template <class T>
class Vectr
{
    Vectr();
    ~Vectr();
    Vectr<T> add(const Vectr<T>& v) const;

    int mSize;
    T* element;
};
```
Template Classes

Example (Instantiating Template Objects)

```cpp
Vectr<int> v1(2, 123);
Vectr<double> v2(3, 4.56);
Vectr<Point2D> v3(4, Point2D(7, 8));
```

- Objects of the class are instantiated using
  
  \( \text{Name<Type> Object;} \)
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.
Example (A “Freebie” Matrix Class)

- Download `vectr.h`
- Download and run `VectrVectrTest.cpp`
Assignment

Homework

- Read Section 9.3, pages 457 - 474.