Classes

• *Classes* can be used to implement types representing complex objects
• An object is an *instance* of a class, that
  – Stores data that define its characteristics
  – Is associated with a set of *operations* that work with its own data
• *string, vector<>* are classes
• Classes support *encapsulation*, which allows implementation details to be hidden from outside code
Declaring Classes

• Form:
  class class-name {
    access-specifier_1: decl-list_1
    access-specifier_2: decl-list_2
    ...
  }

  access-specifier is either public, private or protected
• decl-list is a list of variable or function declarations of the class' members

public and private

• Members declared public: are accessible to functions outside the class
• Example: size() is a public member of the string class
• Operations on objects usually public
• private members are hidden from outside functions
• Member variables, helper functions used to implement operations are usually private
Implementing Classes

- To define member functions outside declaration:
  \[\text{ret-type class-name: : func-name (type}_1 \ arg_1, \ type}_2 \ arg_2, \ ... \text{)} \ compound-stmt\]
- Member variables are already declared and available, and store the data of the instance on which the function operates
- Members that do not modify the underlying object should be declared \textbf{const} (use after argument list)

Overloaded Operators

- In classes that are value types, it is common practice to overload certain operators to support "expected" operations
- \(\gg\) to convert text into instance data
- \(\ll\) to output a string representation
- \(=\) to copy from one instance to another
- \(==, \ !=\) to compare instances
- \(<, \ >, \ <=, \ =>\) if values can be ordered
Testing Classes

- When you implement a class, you should write a suite of functions that will test each member function
- Use a `main()` function that will run each test, and produce output that reports success or failure
- Often, implementation changes, and you should repeat tests to ensure nothing is broken (regression testing)
- Things *will* be broken!

Next Time

More about simple classes
- Special member functions
  - Constructors
  - Destructors
- Copy constructors, overloading =
- Implementation, header and application files