CSI33 Data Structures

Department of Mathematics and Computer Science
Bronx Community College

Outline

Class Syntax

Basics

- C++ is Object-Based: classes can be implemented.
- Classes have the same components as in Python
- Data Members = Attributes = (Instance or Class) Variables
- Member Functions = (Instance or Class) Methods
- C++ is compiled, so classes (with their components) must be declared before they can be used by any code.
- A class declaration is usually written in a header file (`<classname>.h`), so it can be included by any program file using or implementing the class. It declares data members and member functions.
- The definitions of member functions of a class are usually in an implementation file (`<classname>.cpp`). Once it is compiled, it can be linked with any application file.

Class Syntax

Form of a Class Declaration (Definition)

- Begins with `class <classname> {`
- Data Member declarations are similar to those for local variables: type, name.
- Member Functions declarations are similar to those for normal functions: return type; function name; parameter list; default values.
- Does not (usually!) give the implementation of member functions.
- Ends with `};`

Class Syntax

Member Functions

- No `self` parameter!
- Like function declarations: return type; function name; parameter list; default values.
- `const` member functions do not change any data member (attribute) of the object.
- `inline` member function declarations have implementation code.
- Constructors have the same name as the class. Different constructors for one class must have different signatures (formal parameter lists).
- (later: destructors)

Class Syntax

Access Specifiers

Set access levels for member functions and data members:

- `public:` available outside the scope of the class.
- `private:` available only within the scope of the class.
- `protected:` available within the scope of the class or within subclasses derived from the class.

Class Syntax

Class Header File Example: Rational.h

```
#ifndef _RATIONAL_H
#define _RATIONAL_H

class Rational {
public:
   Rational(int n = 0, int d = 1) { set(n, d); } //constructor
   bool set(int n, int d); //sets to n/d
   int num() const { return num_; } // access function
   int den() const { return den_; } // access function
   double decimal() const {return num_/double(den_);}
private:
   int num_, den_;
};
#endif
```

Class Syntax

Class Implementation Files (.cpp)

- Include the header file for that class
- Class Implementation
- Member Function Implementations
- Scope resolution operator (::): (classes and namespaces)

Class Syntax

Class Implementation File Example: Rational.cpp

```
#include "Rational.h"
bool Rational::set(int n, int d)
{
   if (d != 0) {
      num_ = n;
      den_ = d;
      return true;
   }
   else
      return false;
}
```

See programs Rational.h, Rational.cpp, MainProg.cpp; RationalOldStyle.cpp

The C++ string Class

Usage

- C++ strings are implemented as a class that has an array of `char` as an instance variable.
- `#include <string>` to use this class.
- `<<`, `>>` are overloaded to work with `cin` and `cout`.
- `getline(cin, name)`: all text typed before the end-of-line delimiter goes into `name`.
- `<`, `<=`, `>`, `>=`, `==`, `!=`, `+`, `+=` are overloaded.
- The `string` class is also defined within the standard namespace so you must have `using namespace std` at the top of the file, or refer to the class as `std::string`.

The C++ string Class

Examples

See program stringex.cpp

ifstream, ofstream Classes

Usage

- `#include <fstream>` to use these classes.
- Use `infile.open(filename.c_str())`, `outfile.open(filename.c_str())` to access.
- <<, >> are overloaded to work with `ifstream` and `ofstream`.
- `getline(infile, name)` all text typed before the end-of-line delimiter goes into `name`.
- Use `infile.close()`, `outfile.close()` to end access.

See programs `getline.cpp`, `readfile.cpp`