CSI33 Data Structures

Sharon Persinger

Fall 2019

Day 19 November 11

Topics – Classes in C++

Syntax of using classes – strings, filestreams

Syntax of defining classes:

- Header file, implementation file
- Public, private members
- Constructors
- Other member functions
- Instance variables
- Operator overloading

Class use syntax with string class

File of examples stringex.cpp

Familiar dot operator syntax – object.method(   )

Constructor call syntax is different –

- classname varname(parameters); in C++
- varname = classname(parameters) in Python

File input and output - readwrite.cpp

Output to a file example

Input from a file example using getline

getline reads a line from the file into a string variable. The method returns a reference to the input file stream which is interpreted as True if there is more to read, as False if the end of the file has been reached or an error occurs

Writing a class definition

Create a project RationalNum in Visual Studio or your IDE.

Use Visual Studio to add the empty header and implementation files for the Rational class to your project.

Put the code from Rational.h, Rational.cpp, and RationalNum.cpp into the appropriate files.

Writing a class definition

Rational class example

#ifndef again

Rational.h header file has class definition, even though just function prototypes.

Public section. Methods are public.

- Constructor – function with no return type, same name as class. Notice default constructor, no arguments
- Set method
- Accessor methods
- Convert to decimal

Private section: instance variables

- Just variable names, no "self"

What does public mean? What does private mean?

C++ enforces privacy. Private class members can be accessed only by using public member functions of the class.

Member variables generally private. Most member methods are public, though some methods may be private.

const methods

Implementing the methods/functions

Rational.cpp

Include Rational.h

Scope resolution operator :: indicates the method being defined belongs to the class Rational.

Common practice to put _ at end of instance variable name. Way to reuse the obvious name without causing name conflict with the parameters of functions.

Overloading operators

Operators can be overloaded as a method in the class. See Rationalv2. operator+ is overloaded as a member of the Rational class

Rational v2

Operators can be overloaded as stand-alone methods. Some operators must be overloaded this way, for instance, input >> and output << operators.

See Rationalv3. Operator+ is overloaded as a stand-alone function. Operator>>, operator<< are declared in the class as friend, prototypes not in class.

Rational v3

Organizing the class definition

You can put many of the function implementations into the header file.

Look at Rationalv1.h

Examples of functions defined when declared and defined using inline.

Class variables and methods

Class variables are declared using the keyword static in the class definition with the other variables. There is one copy for the class of the member variable, shared with all of the object instances of the class.

See Card.h, Card.cpp

Class variables are declared outside of any functions, are initialized once at first execution of the program. Notice in this example the static variables are also declared const. They are also private, so accessible only through the public member methods of the class.

A class can have a static method also. Static methods can access class variables, but not instance varibles.