CS 368-1 :: C++ for Java Programmers :: Lecture 10 Please sign the attendance sheet!

A. Preview

Last Time (Apr 1)	**Today (Apr 8)**	**Next Time(Apr 15)**
• Polynomial.h, .cpp • Review Big 3 • overloading operators • overloading += and *= • overloading + and * • overloading ++	• review of operator overloading • overloading == • friends • input/output streams • console input/output • overloading output <<	Chapter 9 • reading console /IO • Point class • file I/O • FileIO.cpp • strings

B. Announcements:

1. **Program p3** is due Thursday April 10th. **Student Pairs who complete Program p3 using exemplary Pair Programming will each be given 5 bonus points, but may not receive a score above 100 points.** The points may not be carried forward.

2. In the copyFiles method of SortedList.cpp, you will need to use the following function header:

This is due to the fact that Listnode is declared private in SortedList.h
The scope operator SortedList:: allows the struct to be used as a return type

3. Starting with assignment p3, all assignments will lose **10 points per day late**, regardless of the reason. This applies to all situations and circumstances. The purpose of this change is to make late work possible but to discourage it. With 3 programs to finish in 4+ weeks, you will not have time to get behind. In order to have your late work graded you must put your work in the handin directory, and email the instructor as soon as you have completed your late assignment.

C. Last time: The Polynomial class. This class models a polynomial with an array of doubles. Because we declared the array as a pointer, and because it is good practice in C++, we keep track of the size of the array. In our last lecture we overloaded the following operators. For details, see last week's notes.

overloaded as member function	**+= , *= , p++, ++p**
overloaded as non-member function	**+, ***

D. What is really happening when we overload an operator? (Polynomial class)

overloaded operator	**What the compiler is thinking…..**
`p3 = p2`	`p3.operator=(p2)`
`p3 += p2`	`p3.operator+=(p2)`
`p3 *= 7.0`	`p3.operator*=(7.0)`
`p3 = 7.0 * p2`	`p3.operator=(operator*(7.0, p2))`
`p3 = p2 * 7.0`	`p3.operator=(operator*(p2, 7.0))`
`p3 = p1 + p2`	`p3.operator=(operator+(p1, p2))`
`p3 = p2 + 7.0`	`p3.operator=(operator+(p2, Polynomial(7.0)))`
`p3 = p0+p1+p2`	`p3.operator=(operator+(operator+(p0, p1), p2));`
`p3++`	`p3.operator++(0)`
`++p3`	`p3.operator++()`

On Your Own: Write a main function that demonstrates the above table. Run it and verify that the output is correct.

E. Overloading == and other relational operators

main.cpp	`if (p0 == p1)`
Polynomial.h	`friend bool operator==(const Polynomial &lhs, const Polynomial &rhs);`

Polynomial.cpp	`bool operator==(const Polynomial &lhs, const Polynomial &rhs){`
	`  if (lhs.size != rhs.size) {`
	`  return false;`
	`  }`
	
	`  for (int i=0; i<lhs.size; i++) {`
	`  if (lhs.coefs[i] != rhs.coefs[i]){`
	`  return false;`
	`  }`
	`  }`
	`  return true;`
	`}`

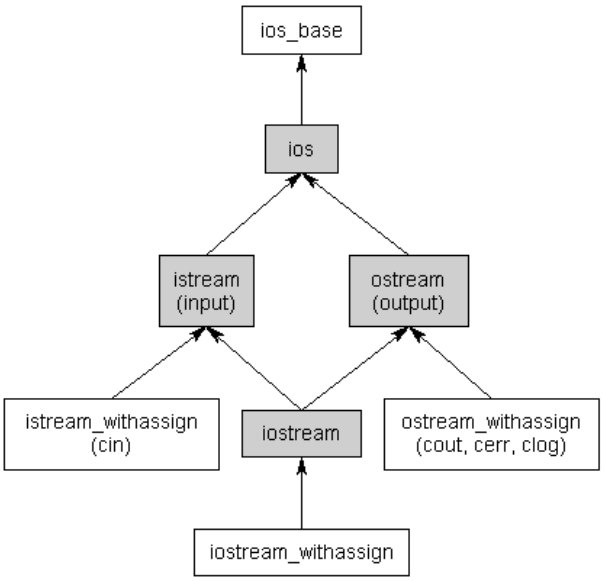
Why we need friend to make this work:

How friend works in general:

F. Basic Ideas about Console Input and Output:

The Inheritance Hierarchy of Stream I/O

(Because C++ allows multiple inheritance, you see classes that have multiple superclasses.)



1. A stream is an abstract way of thinking of a sequence of bytes that is sent into, or sent out from, a program.

2. cin and cout are pre-defined “stream” objects that have predefined operators >> and << which work with different types of parameters.

3. Extraction operator >>
removes things from the input stream.
cin >> age >> height;

4. Insertion operator <<
places things into the output stream
cout << name << “, “ << age << endl;

5. Both >> and << allow for chaining.

The familiar cin and cout statements that you have been using in your program for keyboard input and display output are really objects of the iostream file class. The cin and cout objects invoke predefined file streams. Thus we say that standard input is read from the cin stream and standard output is written to the cout stream. When you include the <iostream> header file your program the cin and cout file streams are defined automatically. Of course, the only files that you can access conveniently with cin and cout are the keyboard and display files that area “attached” to these file systems.

Different web links to dive deeper:

Input and Output streams this link:

Basic Input/Output this link.

A Gentle Introduction to C++ io streams.

G. Overloading output operator<<

In Java, we would write a toString() method for our class, which would then allow us to write System.out.println("my object is:" + someObject); We are doing the a similar thing in C++. We write a member function and then overload operator<< to call the member function.

main.cpp

```
cout << p1 << " is a polynomial" ;
```

Polynomial.h

```
// public, member function
void print (ostream & out = cout ) const;  // accessor

// non-member function
ostream & operator<< (ostream & out, const Polynomial & p)
```

Polynomial.cpp

```
ostream & operator<< (ostream & out, const Polynomial &p){
    p.print(out);
    return out;
}

void Polynomial::print(ostream & out) const{
    if (size == 0) {
        return;
    }
    for (int i = size - 1; i > 0; i--)
        out << coefs[i] << "x^" << i << " + ";
    out << coefs[0] << endl;
}
```

Recall that cout is a pre-defined instance of the ostream class.
Our function returns a reference to an ostream so that we can chain output commands together.
Check here if you would like more details.

We change the header of the print function to accept a parameter of type ostream, or just default to cout. This allows us to use cout without naming it, or if we want, using an output file.

H. Understanding how cin >> works

Consider the following code:

```
char c;
int x;
double y;
cin >> c >> x >> y;
```

What is required for the input to be entered in successfully?

- a single character, (could be anything)
- zero or more whitespaces
- a series of digits
- one or more whitespaces
- a series of digits, possibly followed by a '.' and more digits
- possible whitespace
- a newline character (return)

In our next lecture we will see how we need to use this idea when overriding the >> operator, and when using files.

Extra Notes about operator<<

The first argument is a reference to an output stream. This operator returns a reference to the same output stream, so that the << operators can be chained together.

Friend:
The keyword friend is used to give a non-member function or operator access to all members (including private data members) of a class. These declarations may appear anywhere within a class definition.

Friendships are not symmetric, so if one class friends another, the relationship is one-way.

File Streams will be discussed in our next lecture.

Extra notes on streams from Computer Science Tapestry, by Owen Astrachan:

The statement

```
cin >> first >> second;
```

is read, or parsed, by the C++ compiler as though it were written as

```
(cin >> first) >> second;
```

because >> is left-associative (see Table A.4 in Howto A.) Think of the input stream, cin, as flowing through the extraction operators, >>. The first word on the stream is extracted and stored in first, and the stream continues to flow so that the second word on the stream can be extracted and stored in second. The result of the first extraction, the value of the expression (cin >> first), is the input stream, cin, without the word that has been stored in the variable first.

The most important point of this explanation is that the expression (cin >> first) not only reads a string from cin but returns the stream so that it can be used again, (e.g., for another extraction operation).