CHAPTER 12
FILE I/O

CMPS 148

More classes: IO

- Usage of cout and cin has always has a *peculiar* syntax
  - They aren't functions…
  - << and >> are defined as *operators*
  - Typically we think of *operators* as taking some sort of an action on *things* (objects)

Examining <<

- << is defined as an **<u>insertion</u>** operator
  - It inserts the right hand side (can be many different types) *into* an *ostream* object
  - The result is the same *ostream* object

```
cout << "hello";
```

cout

cout is an instance of the **ostream** class

Streams (**ostream** is an output stream) are connected to *something*

Examining <<

```
cout << "hello" << " " << "world" << endl;
```

```
cout << " " << "world" << endl;
```

```
cout << "world" << endl;
```

```
cout << endl;
```

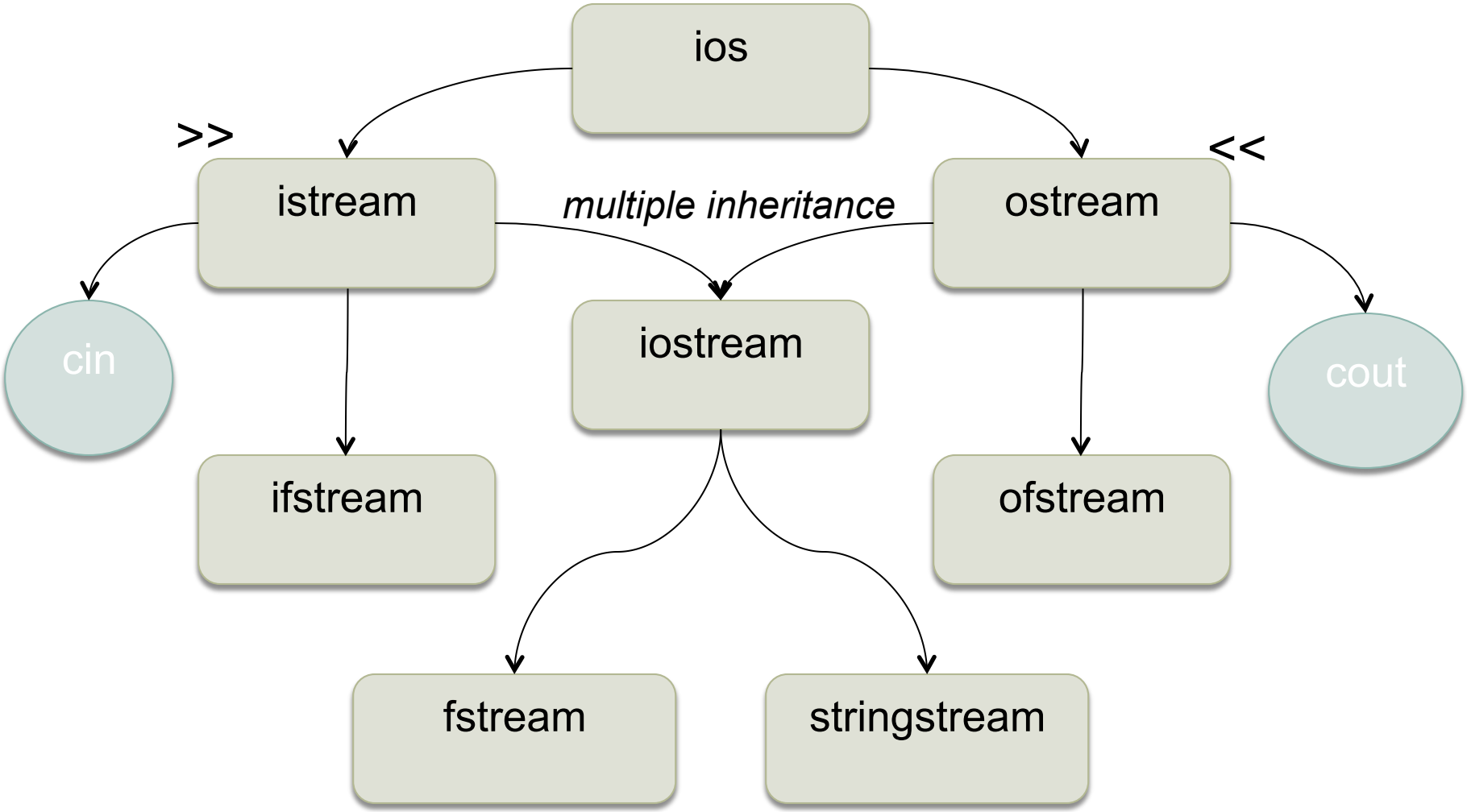
Chaining the << operators is a lot like writing
Int x = 6 + 8 + 9 + 11;
The operators are applied one at a time

Other ostreams

- There are many types of output streams - cout is just the most familiar
- `stringstream`: Output stream attached to a string instead of the console
- `ofstream`: Output stream attached to a file

The << operator works the same way, for **all** ostream types

Inheritance Model



Using File Streams

- Think of a file stream as a variable - you must first *declare* it:

```
ofstream fout;
```

- To associate it with a file, you must *open the file:*

```
fout.open("output.txt");
```

- You may then use it like cout

```
fout << "hello" ; // writes hello to the output.txt
```

- Finally, you must **<u>close</u>** the file:

```
fout.close();
```

Using File Streams

- Must declare (remember, `ifstream` and `ofstream` are NOT THE SAME)

```
ifstream fin;
```

- To associate it with a file, you must *open the file:*

```
fin.open("input.txt");
```

- Make sure the operation was successful
- `if ( !fin ) // failed, file doesn't exist`
- Now you can use fin just like cin
- Finally, you must **close** the file:

```
fin.close();
```

Simple Example

- Read 10 numbers from input.txt
- Print the square of each number to output.txt

Appending Files

- You don't always have to overwrite…

```
fout.open("test.txt", ios::out | ios ::
app);
```

- Lets create a program that asks the user for a filename, and then **appends** all number typed to that file.

Whitespace…

- You can run into trouble when mixing cin >> and getline(cin, str)
- getline reads all characters **up to and including** the newline character.
- cin >> number will read characters up to **but not including** the newline, and parse those characters as necessary.

Whitespace

- Reading with getline before cin >> number works fine.
- But – if you ever need to read numbers **first,** you will find that getline call coming after cin >> calls don't work correctly!
- This is because getline reads the newline character that cin left behind – so it returns an empty string!

- Solution: **cin.ignore()**

Streams and Functions

- ostream and istream are **base** classes for the familiar cout and cin, and the new ofstream and ifstream
  - You can write functions that accept parameters of ostream and istream
  - Can read/write from **any** stream.

Exercise

- Ask user for the name of an input file (containing numbers)
  1. Read the numbers from the input file
  2. Display the numbers
  3. Ask the user to add more numbers
  4. Write the numbers back to the same file (new + old)
- *Use the same function to perform 1 and 3 and 2 and 4 – pass cout/cin* ***<u>or</u>*** *the file stream as input.*