CSCI 2100: More C++

I/O

Classes

Recap

- HW due Friday
- Lab tommorrow, due Sunday
- Next HW over classes, likely due next Friday

Prelab: due before 2pm tomorrow

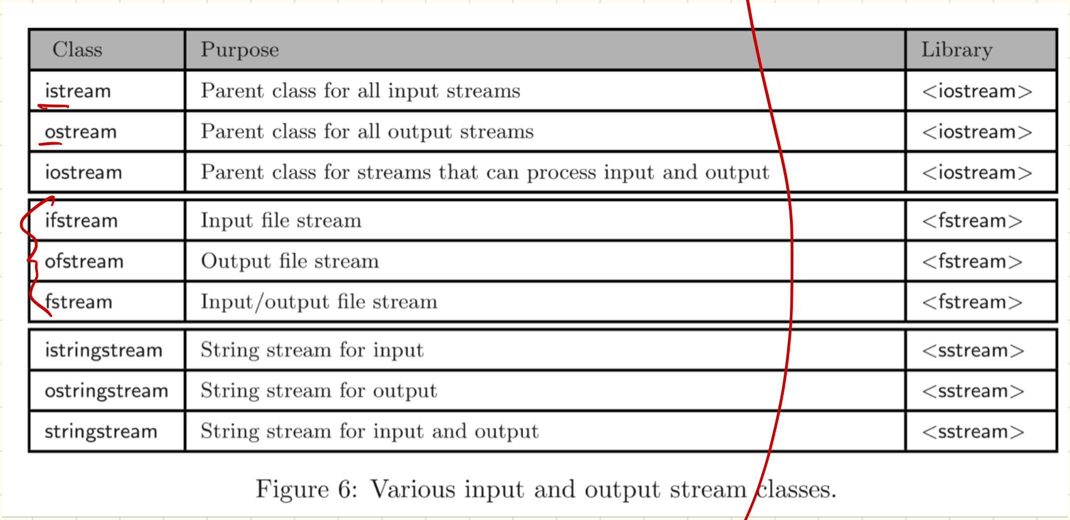
Last time:

- Loops
- If statements
- Functions

Input + Output

cout
cin

Need to include an appropriate class to handle.
(Similar to strings)

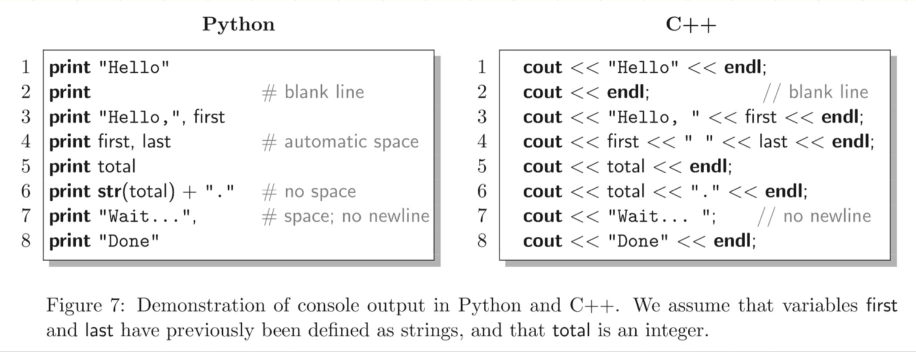


Class	Purpose	Library
istream	Parent class for all input streams	<iostream>
ostream	Parent class for all output streams	<iostream>
iostream	Parent class for streams that can process input and output	<iostream>
ifstream	Input file stream	<fstream>
ofstream	Output file stream	<fstream>
fstream	Input/output file stream	<fstream>
istringstream	String stream for input	<sstream>
ostringstream	String stream for output	<sstream>
stringstream	String stream for input and output	<sstream>

Figure 6: Various input and output stream classes.

Syntax: #include <iostream>
#include <fstream>

Formatting I/o



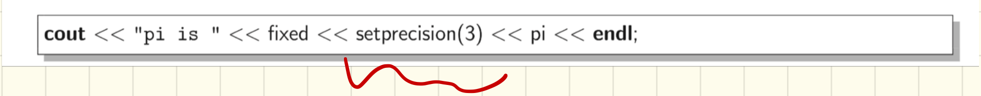
Python

```
print "Hello"
print                      # blank line
print "Hello,", first
print first, last          # automatic space
print total
print str(total) + "."     # no space
print "Wait...",           # space; no newline
print "Done"
```

C++

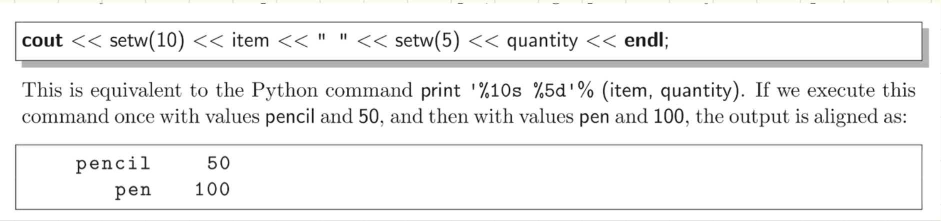
```
cout << "Hello" << endl;
cout << endl;                      // blank line
cout << "Hello, " << first << endl;
cout << first << " " << last << endl;
cout << total << endl;
cout << total << "." << endl;
cout << "Wait... ";        // no newline
cout << "Done" << endl;
```

Figure 7: Demonstration of console output in Python and C++. We assume that variables first and last have previously been defined as strings, and that total is an integer.



```
cout << "pi is " << fixed << setprecision(3) << pi << endl;
```

pi is 3.141 ← this stays set



```
cout << setw(10) << item << " " << setw(5) << quantity << endl;
```

This is equivalent to the Python command print '%10s %5d'% (item, quantity). If we execute this command once with values pencil and 50, and then with values pen and 100, the output is aligned as:

```
    pencil     50
       pen    100
```

Using cin

```
int number;
cout << "Enter an integer:" ;
                          ↑
cin >> number;
```

```
Enter a number: 13
```

Notes

- Inputs are separated by *any* whitespace!

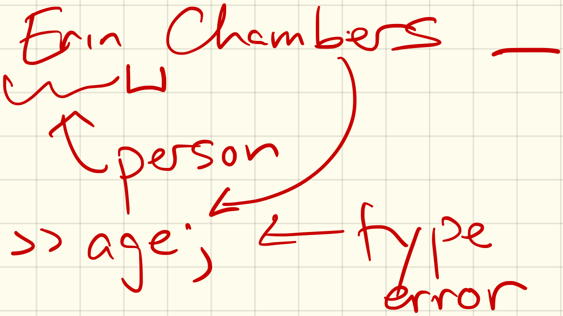
```
cin >> a >> b;      10 15
                    10
                    15
```

(Careful w/ strings!)

- Type of input must match type of variable (*not* all strings)

Issue

```
string person;
cout << "Enter your name: ";
cin >> person;
```

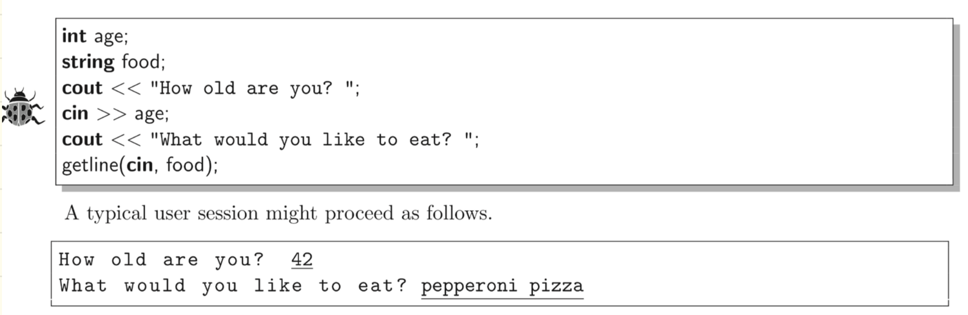


```
cin >> age;
```

Fix: use getline:

```
getline(cin, person);
```

Another Issue:



```
int age;
string food;
cout << "How old are you? ";
cin >> age;
cout << "What would you like to eat? ";
getline(cin, food);
```

A typical user session might proceed as follows.

```
How old are you?  42
What would you like to eat? pepperoni pizza
```

input stream:

$\underbrace{42}_{\text{age}}\ \underbrace{/n}_{\text{food}}$ pepperoni_pizza/n

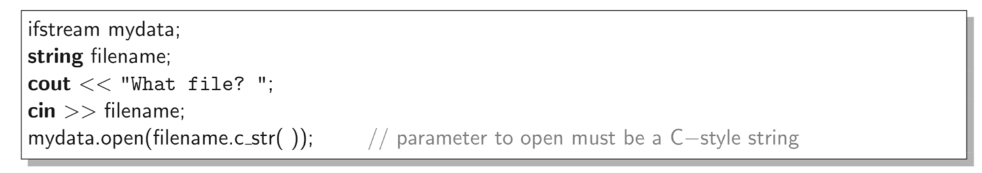
File streams : fstream

```
#include <fstream>
using namespace std;
ifstream mydata("scores.txt");
```

// ↑ creates input stream object

```
mydata >> firstscore;
```

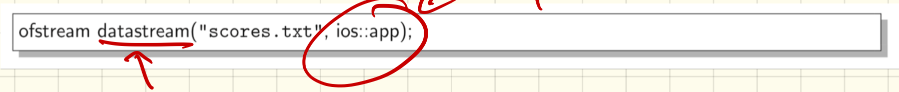
Adding input:



```
ifstream mydata;
string filename;
cout << "What file? ";
cin >> filename;
mydata.open(filename.c_str( ));        // parameter to open must be a C−style string
```

Outstreams:

append



```
ofstream datastream("scores.txt", ios::app);
```

name

Note

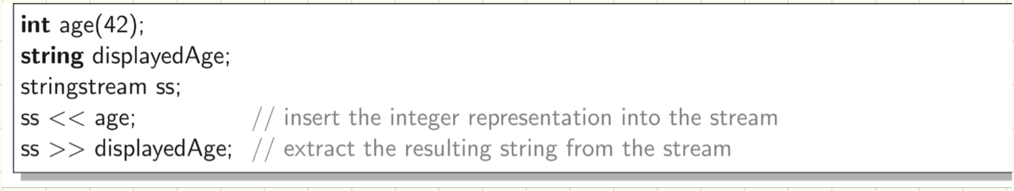
Use: datastream << "My output" << endl;

- more Syntax examples in transition guide

There is an fstream object.
Complex!
(We'll avoid in this class)

String streams

Ex: Cast between # & string



```
int age(42);
string displayedAge;
stringstream ss;
ss << age;           // insert the integer representation into the stream
ss >> displayedAge;  // extract the resulting string from the stream
```

A note on variable scopes

```
int main() {
   int a;          <- put declarations at top!
   cin >> a;
   if (a > 0) {
       int b=12;
   } // b is destroyed
   else {
       int b=16;
   } //destroys b
   cout << "a is " << a << endl;
   cout << "b is " << b << endl;
}
```

syntax error

"b is unknown"

```
int i;
for (~~int~~ i=0; i<size; i++){

} //i is destroyed
```

Arrays as fcn inputs

Ex: write a fcn to specify if sum of values is even

```
                                   empty size
bool sumEven( int anArray[], int size){
    sum = 0;
    for (int i=0; i<size; i++)
        sum += anArray[i];
    if (sum % 2 == 0)
        return true;
    else
        return false;
}
```

Here: int a[] actually makes a (the array) a pointer!

More later...

Doesn't copy entire array, just has something "pointing" to start of it.

To call:

```
if (sumEven(myArray, 10))
    cout << " . . . "
```

Classes

What is a class?

Storing an "object":

- methods
- data (collection)

Creating one:

```
string s;
string greeting("Hello");
```

constructor

initialization value

Never:

```
string s();
```

why?, declares a functions w/ return type of string

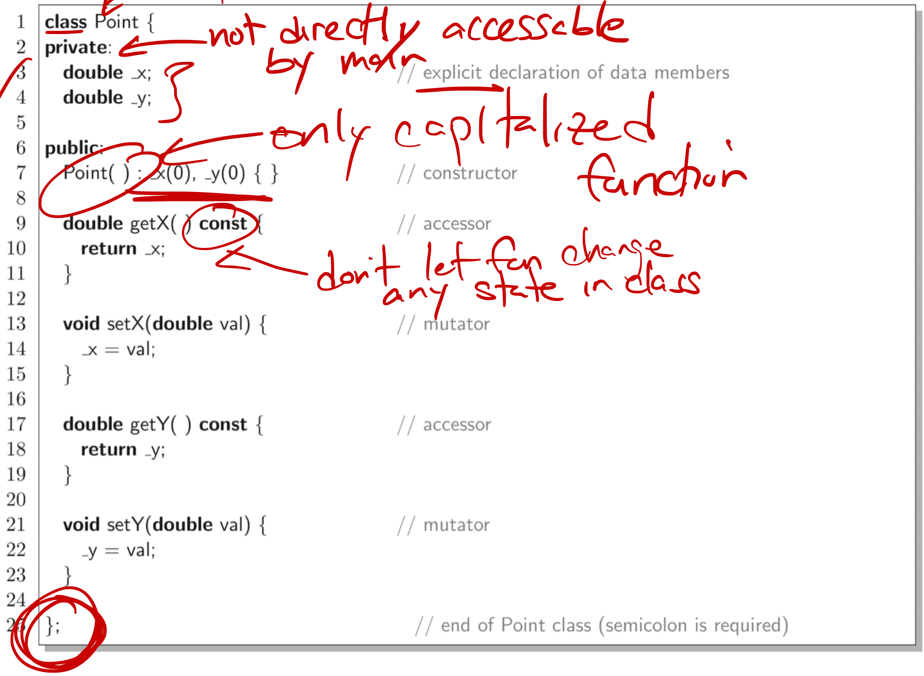
Never:

```
string("Hello") s;
```

(wrong)

Making our own:

capitl letter (only place)



not directly accessable by main

only capitalized function

don't let fcn change any state in class

```
class Point {
private:
  double _x;                        // explicit declaration of data members
  double _y;

public:
  Point( ) : _x(0), _y(0) { }       // constructor

  double getX( ) const {            // accessor
    return _x;
  }

  void setX(double val) {           // mutator
    _x = val;
  }

  double getY( ) const {            // accessor
    return _y;
  }

  void setY(double val) {           // mutator
    _y = val;
  }
};                                   // end of Point class (semicolon is required)
```

Figure 9: Implementation of a simple Point class.

other options: public or protected

save file as Point.h

another file:

```
#include "Point.h" ←
#include <iostream>
using namespace std;

int main() {
   Point myPoint;
   Point other;

    myPoint._x = 12; X compiler
                              error
    myPoint.setX(12);
    other = myPoint; //deepcopy

  return 0;
}
```

On Hopper:

```
>> gcc myfile.cpp -o pgm
       ↳ a.out
>> ./a.out

   >> ./pgm
```