CS180 - C++ + the command line

Note Title 9/2/2011

Announcements

- g++ should now be ok on turing
- Labs are due Sunday
  prelabs are due **before** class
  generally, you can work in pairs
  (tell me your partner)

- HW1 is due 1 week from Sat.

Command line tips

In general, you'll use 5-6 commands the most

- ls - list files in current directory
- cp sourcefile targetfile
- mkdir name
- rmdir name
- cd directoryname
- mv sourcefile targetfile
- rm

Others

- vi or emacs or pico (nano) ← user friendly
- g++
- man !! ☺

Tricks

- Hitting the up arrow gives the last thing you typed (& then you can edit)
- Hitting tab will autocomplete
- You can use & to get prompt back
  ex: kak file &
- . is current directory, .. is parent
  ex: cd ..
  cp ../file .

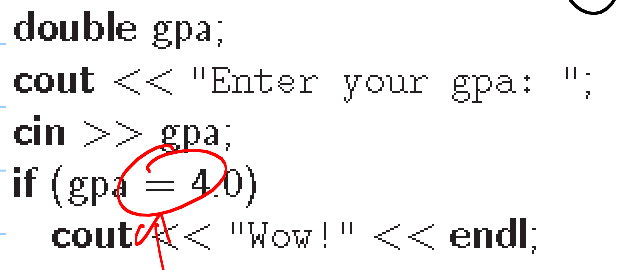
Last time

- More C++:
  - while loops
  - if statements
  - functions

Common error

What is wrong?

$a = b = \underbrace{c = 0}_{0}$;



```
double gpa;
cout << "Enter your gpa: ";
cin >> gpa;
if (gpa = 4.0)
   cout << "Wow!" << endl;
```

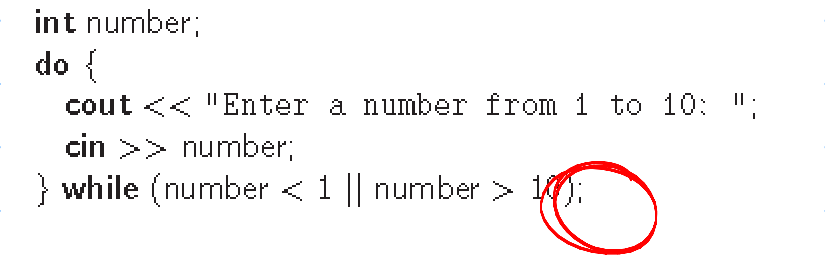
==

In C++, gpa is reset.

(& it is true)

in Python: compiler complains

Do-while loops



```
int number;
do {
   cout << "Enter a number from 1 to 10: ";
   cin >> number;
} while (number < 1 || number > 10);
```

- Executes body before checking the boolean

For loops

In python, iterator based.

Example:

start value

boolean (as in while loop)

update

for (~~int~~ count =10; count >0; count--)
    cout << count << endl;
cout << "Blast off" << endl;

Note: int declaration isn't required

The main function

Every program defaults to running a main.

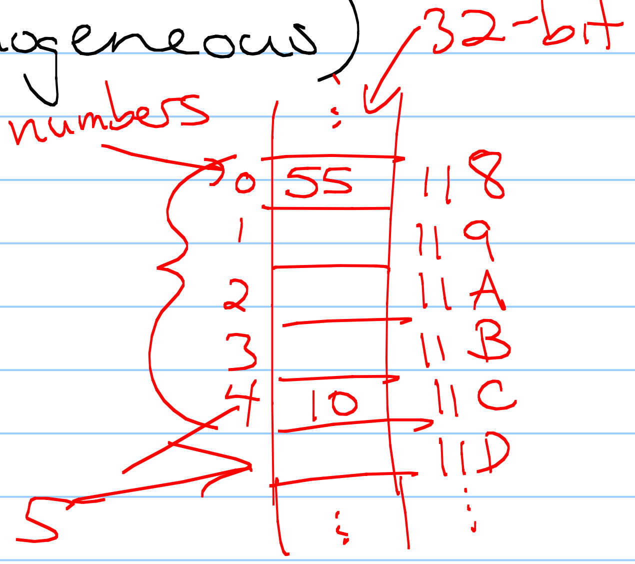
```
int main() {
    body;
→   return 0;
}
```

Arrays

Python has lists, tuples, etc.

In C++, only have arrays.

- Size is fixed at declaration
- type is fixed (& homogeneous)



Ex:

```
int numbers[5];
numbers[0] = 55;
numbers[4] = 10;
```

?
```
numbers[5] = 5;
```

Creating Arrays:

Allowed:

```
int daysInMonth = {31, 28, 31, 30, 31, 30,
                   31, 31, 30, 31, 30, 31};
```

Error:

```
int daysInMonth[];
```

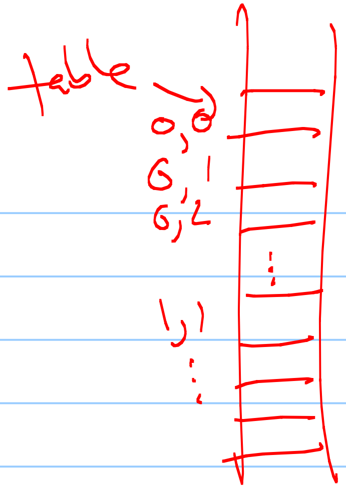
Allowed:

```
char greeting[] = "Hello";
```

Strings are char arrays.

Multi-dimensional arrays

```
int table[8][10];
```



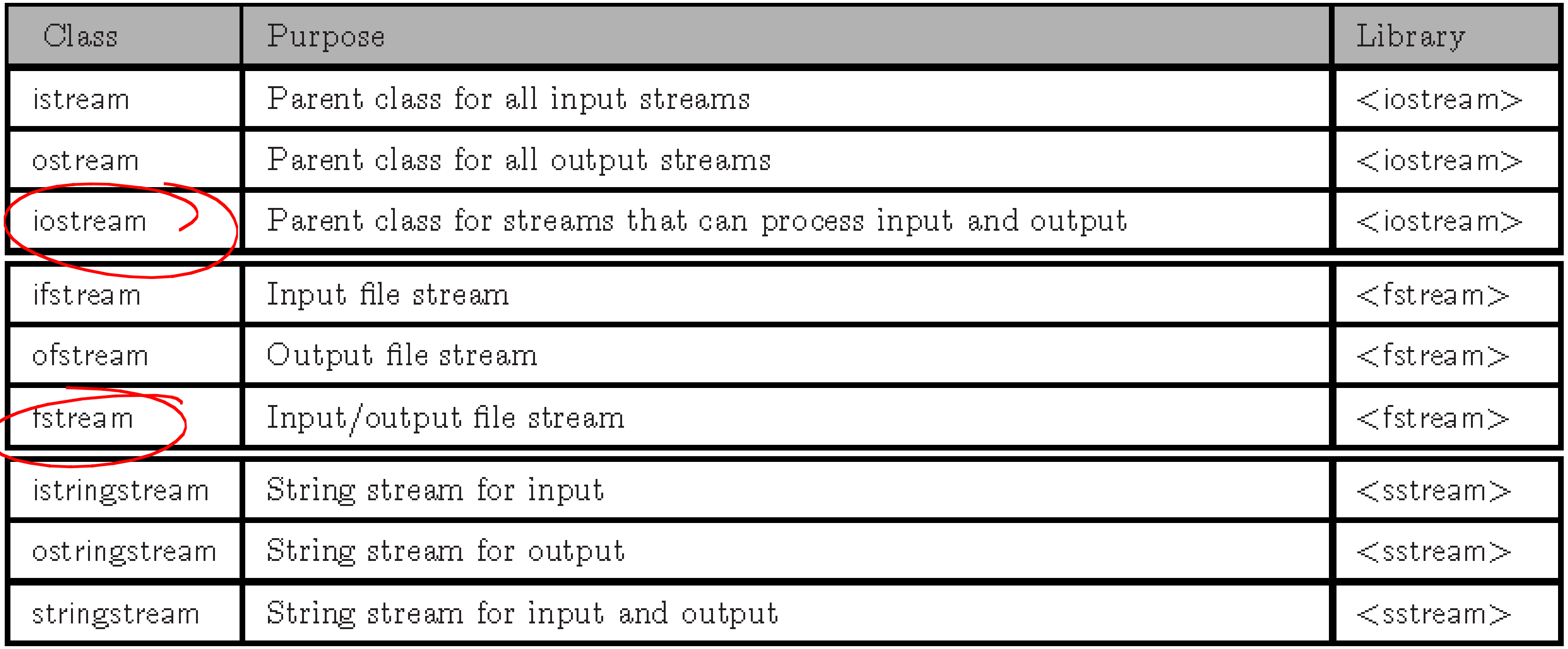
```
for (int i=0; i<8; i++)
  for(int j=0; j<10; j++)
    table[i][j] = 0;
```

$$\sum_{i=1}^{n}\sum_{j=1}^{m} 1$$

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										

Input + Output

C++ has several predefined classes.



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>

Using iostream

top

#include <iostream>

using namespace std; ← omit this

std::cout

Notes: - can now use cin (for input) + cout (for out-put)

- separate distinct variables by >> or <<

cin cout

- use endl for end of a line

'\n'

Example

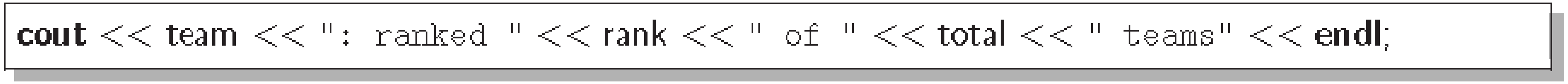
Python

```
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;
```

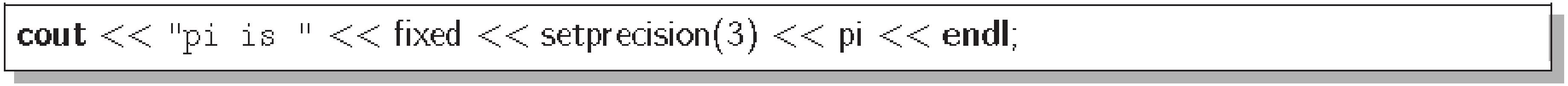
Formatting output



```
cout << team << ": ranked " << rank << " of " << total << " teams" << endl;
```

- No '%d' here to easily format

Can set precision:



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

- Note that precision stays set to 3

Using cin

```
int number;
cout << "Enter a number:";
cin >> number;
```

Note:

- inputs are separated by any white space

```
cin >> a >> b;
```

```
10  20 ↵
10 ↵
20 ↵
```

- type of input must match type of variable (<u>not</u> all strings)

One possible problem:

```
string person;
cout << "What is your name?";
cin >> person;
```

I type "Erin Chambers".

What happens? ↑

person = "Erin"

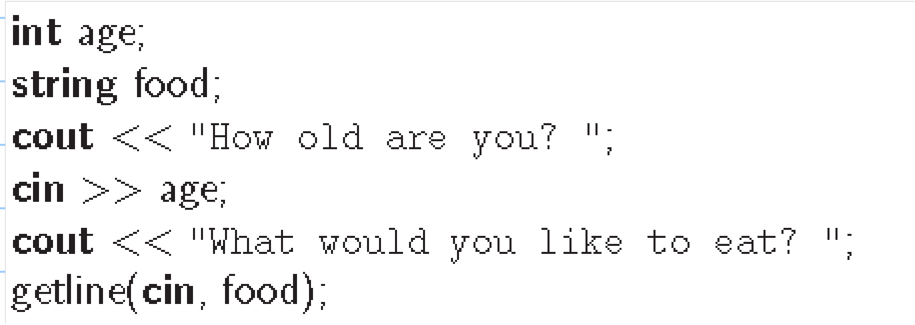
Getline

- getline is a function which saves the string up to (but not including) the next newline

Ex:

```
string person;
cout << "What is your name? ";
getline ( cin, person);
```

Another tricky example



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

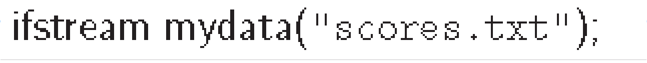
I type : 15 \n
hot dogs

Problem: age = 15
food = ""

Using File Streams - fstream

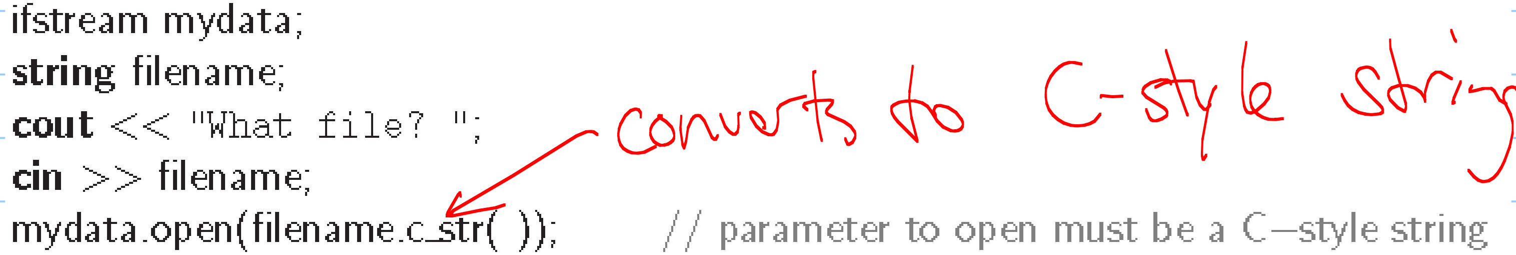
```
#include <fstream>
using namespace std;
```

if file is known:



```
ifstream mydata("scores.txt");
```

if not:



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

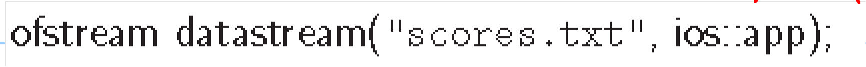
converts to C-style string

```
mydata >> var;
```

ofstream

By default writing to a file overwrites the file'.
(Think 'w' in Python.)

To append:



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

'a' in Python

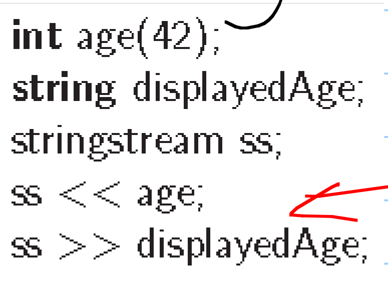
Reading and writing

There is also an fstream object, which allows reading & writing to a single file.

Much more complex.

String Streams

Ex: Casting between numbers & strings.



```
int age(42);
string displayedAge;
stringstream ss;
ss << age;
ss >> displayedAge;
```

writes an int

reads the int back in as a string