Data Structures

I/O Classes
Announcements & Reminders

- Everyone should be able to use Linux lab!
- HW due tomorrow
- Lab tomorrow (prelab due before 9am tomorrow)
- Wed., office hours will be moving (stay tuned)
Last time

- Control structures (loops, if, for, ...)
- Arrays (+ segmentation faults)
- Began I/O

<table>
<thead>
<tr>
<th>Class</th>
<th>Purpose</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>istream</td>
<td>Parent class for all input streams</td>
<td><code>&lt;iostream&gt;</code></td>
</tr>
<tr>
<td>ostream</td>
<td>Parent class for all output streams</td>
<td><code>&lt;iostream&gt;</code></td>
</tr>
<tr>
<td>iostream</td>
<td>Parent class for streams that can process input and output</td>
<td><code>&lt;iostream&gt;</code></td>
</tr>
<tr>
<td>ifstream</td>
<td>Input file stream</td>
<td><code>&lt;fstream&gt;</code></td>
</tr>
<tr>
<td>ofstream</td>
<td>Output file stream</td>
<td><code>&lt;fstream&gt;</code></td>
</tr>
<tr>
<td>ifstream</td>
<td>Input/output file stream</td>
<td><code>&lt;fstream&gt;</code></td>
</tr>
<tr>
<td>istream</td>
<td>String stream for input</td>
<td><code>&lt;sstream&gt;</code></td>
</tr>
<tr>
<td>ostream</td>
<td>String stream for output</td>
<td><code>&lt;sstream&gt;</code></td>
</tr>
<tr>
<td>stringstream</td>
<td>String stream for input and output</td>
<td><code>&lt;sstream&gt;</code></td>
</tr>
</tbody>
</table>

Figure 6: Various input and output stream classes.
### Setting precision:

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

This prints `3.141`. The default precision is unlimited, so this stays.

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

This is equivalent to the Python command `print "%s %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`

```cpp
int number;

cout << "Enter an integer:”;

cin >> number;
```

---

**Notes**

- Inputs are separated by any whitespace!

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

(Careful w/ strings!)

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

Python
Issue

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

Fix: use `getline`:
```
getline (cin, person);
```
Another issue:

```c++
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
```
File streams: `fstream`

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

ifstream mydata("scores.txt");

int val;
mydata >> val;
Adding input:
```
```
Out: 

```cpp
ofstream mystream("scores.txt");
```

or

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

Note:

- Creates a file (overwriting if necessary)
- This appends (Python: "a")
- There is an `fstream` object.
- Complex!
- (We'll avoid in this class)
String streams

Ex: cast between # + string

```cpp
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

Example C++ code:

```cpp
int main() {
    int a = 12;
    int b = 6;
    if (a > b) {
        cout << a << endl;
        cout << b << endl;
        a = 0;
        done = true;
    } else {
        a = 2;
        b = 2;
    }
}
```

The lifetime of a variable in C++ is determined by the scope in which it is declared.

In the given example, `a` is destroyed when the if block completes, while `b` is destroyed when the program finishes executing.

The output will be:

```
12
6
```
Arrays as fun inputs

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

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

if ( sum % 2 == 0 )
    return true;
else
    return false;
Here: int a[] actually makes a (the array) a pointer!

doesn't copy entire array, just has something "pointing" to start of A.

To call:

```cpp
int main () {
    int Ar, i;
    for (i = 0; i < 65; i++)
        Ar[i] = i;

    int sumEven, j;
    for (j = 0; j < 65; j++)
        if (Ar[j] % 2 == 0)
            sumEven += Ar[j];

    cout << "it's even";
    //more code
}
```
What is a class?

"Object" - Stores data + fans that restrict how you interact w/data

Creating one:

```java
String s;
String greeting("Hello");
```

Never: `String s();`

Why? Declared s a function, outputs returning

Never: `String("Hello") s;`
Making our own!

```cpp
class Point {
private:
  double x; // explicit declaration of data members
  double y;

public:
  Point() : x(0), y(0) {} // constructor

  double getX() const {
    return x;
  }

  void setX(double val) {
    x = val;
  }

  double getY() const {
    return y;
  }

  void setY(double val) {
    y = val;
  }
};
```

Figure 9: Implementation of a simple Point class.