A STREAM RUNS THROUGH IT

ADAM SWEENEY
CS 211

INTRODUCTION

- We've been using streams for a good while now
- Time to look at some other streams

AGENDA

- Classes and Objects
- File I/O
- Some techniques for file I/O
- A quick look at stringstreams

CLASSES AND OBJECTS

BIG PICTURE OF CLASSES

- Streams are not typical variables
- They are objects, instantiations of a class
- A class is a collection of data and functions under a single name
- An object is an instantiation of a class, holding specific data
- Objects can invoke class member functions, which use the object's data
  - Objects can do work on themselves
- The term "calling object" is used to refer to the object calling a member function

FILE I/O

STREAMS AND BASIC FILE I/O

- "As a leaf is carried by a stream, whether the stream ends in a lake or in the sea, so too is the output of your program carried by a stream not knowing if the stream goes to the screen or to a file"
  -- Washroom wall of a CS department (1995)
- Data flows in to and out of our program
  - When data flows in to the program, it is an input stream
  - When data flows out of the program, it is an output stream
- All streams behave similarly
  - Only the source or destination differ

FILE INPUT/OUTPUT

- When getting input from a file, we are *reading* the file
- When outputting to a file, we are *writing* to the file
- This requires streams made for files
- The biggest differences we'll see here are setup and maintenance

DECLARE A STREAM

- It turns out that `cout` and `cin` are oddballs
- They are objects with the destination/source already configured
  - Their types are `std::ostream` and `std::istream`, respectively
- To declare file streams:

```
#include <fstream>

std::ifstream fin;      // An input file stream
std::ofstream fout;     // An output file stream
std::fstream fstream;   // Can be input || output
```

ATTACH FILE TO STREAM

- Two ways
  - At declaration
    ```
    std::ifstream fin(“FILENAME”);
    ```
  - After declaration
    ```
    std::ifstream fin;
    fin.open(“FILENAME”);
    ```
- One other thing we are required to do after attaching a file
- Check that the stream is “healthy”

HEALTHY STREAM

- ```
  std::ifstream fin(“FILENAME”);
  if (!fin) {
      std::cerr << “Error opening file, exiting.\n”;
      std::exit(1);
  }
  ```
- We are checking the Boolean of the stream
- If everything is fine, the stream returns true
  - We check if the opposite is true
  ```

SOME TECHNIQUES FOR FILE I/O

A DISCLAIMER

- This sections covers a few modes of filestreams
  - Those modes are not used in this course
  - Still good to know

OPEN MODES

- By default, file streams assume that the file is plaintext
- An input file stream is defaulted to read from a file
- An output file stream is defaulted to write to a file
  - Not just that, but to delete the existing contents (if any) and write it as a new file
- We can tell our file streams to open files in binary mode instead of plaintext
- We can write to files without destroying existing contents

STRINGSTREAMS

A STRING AS A STREAM

- Consider a long string containing a row of data
- It is possible to create a `std::stringstream` and use stream syntax to simply place the data into the appropriate variables.