Can You Figure Them Out?

- 26 L of the A (26 Letters of the Alphabet)
- 7 D of the W
- 13 S in the USF
- 3 W on a T
- 12 M in a Y
- 8 T on an O
- 29 D in F in a L Y
- 13 L in a B D
- 64 S on a C B

Array and functions
File I/O

Luke Sathrum – CSCI 20

Today's Class

- Arrays and Functions
- File I/O

Arrays and Functions

Array Element as an Argument

- We can use an array element much like we use a normal variable in a function call

```
MyClass object;
double i, n, a[10];
object.MyFunction(n);
object.MyFunction(a[3]);
```

Entire Arrays as Function Arguments

- We can pass entire arrays as arguments
- It is called an array parameter
  - Array parameters are denoted by an array with no size

```
double a[];
```

  - In the function declaration it would look like this

```
void MyFunction(double a[]);
```

Function with Array Parameter

- When you call the function it looks like this:

```
double score[5];
object.MyFunction(score);
```

  - Notice **`score`** does not have the **`[]`**
- Note
  - If you modify values in the function the referenced array is changed as well

Sample Code

- Using parameters in arrays
  - `array_parameters.cpp`

File I/O Terms and Syntax

I/O Terms

- Stream
  - A flow of characters (or other data)
- Input Stream
  - Flow of data into your program
  - Can get data from a user or from a file
- Output Stream
  - Data flowing out of your program
  - Output can go to the screen or a file
  - **`cin`** and **`cout`** both are streams
    - Defined in **`iostream`**

Quick Example

- We can create other streams besides **`cin`** and **`cout`**
- **`fin`** could be defined to get input from a file
- **`fout`** could be defined to output to a file
- Input

```
int the_number;
fin >> the_number;
```

- Output

```
fout << "the_number is "
     << the_number;
```

File I/O Terms

- Reading
  - When a program takes input from a file
- Writing
  - When a program outputs to a file

Thinking about Files

- Our examples will simulate what we can do with **`cin`** and **`cout`**
  - Once we start reading a file we can’t go back
  - Once we write to a file we can’t go back and change

File I/O - Syntax

- We include **`fstream`**
  - `#include <fstream>`
- In our program (**`main`**) we need to declare our stream
  - Input

    ```
    std::ifstream fin;
    ```
  - Output

    ```
    std::ofstream fout;
    ```
- Think of `ifstream` and `ofstream` as types

File I/O - Syntax

- Now we need to connect our streams to a file
  - This is called “opening” a file
  - `fin.open("infile.txt");`
    - This connects our stream to a file named infile.txt
    - This file is located in the same directory as our program
- Once connected/opened we can read from the file
  - `int one_number, another_number;`
  - `fin >> one_number >> another_number;`

File I/O - Syntax

- We open an output stream in the same way
- **`fout.open("outfile.txt");`**
  - This either creates or replaces the file outfile.txt
- Once connected we can output to the file

```
fout.open("outfile.txt");
fout << "Some Text" << endl;
```

Closing a File

- When you are done with a file you should close the stream associated with it
- We use **`.close()`**
  - **`fin.close();`**
  - **`fout.close();`**

Sample Code

- Using streams with files
  - `file_io.cpp`

File I/O
-Appending
-Short Syntax

Appending to Files

- So far we can only write new files
- All content is lost when we do this
- We can append instead to the end of a file
- If the file doesn't exist then it will be created

```
std::ofstream fout;
fout.open("data.txt", std::ios::app);
```

More Open File Syntax

- We can combine the syntax for declare and open
- `std::ifstream`

```
std::ifstream fin;
fin.open("infile.txt");
```

Becomes

```
std::ifstream fin("infile.txt");
```

More Open File Syntax

- **std::ofstream**

```
std::ofstream fout;
fout.open("outfile.txt");
```

BECOMES

```
std::ofstream fout("outfile.txt");
```

More Open File Syntax

- **std::ofstream** append

```
std::ofstream fout;
fout.open("outfile.txt", std::ios::app);
```

BECOMES

```
std::ofstream fout("outfile.txt", std::ios::app);
```

Sample Code

- Append and Short syntax with Files
  - `append_short_syntax.cpp`

Checking if File Exists

File Check

- Sometimes the file you want to open doesn't exist
- Or it may not be able to be created
- In C++ we can check to see if the file was opened
- We use **`.fail()`** to check
  - This returns a Boolean value

File Check - Syntax

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

if (fin.fail()) {
   cerr << "Input file opening failed.\n";
}
```

Sample Code

- Check to make sure our files are opened correctly
  - `good_io.cpp`

Review

- Streams are flows of data
- We read data from a file
- We write data to a file
- Use `ifstream` and `ofstream` to connect to a file
- Use **`.close()`** when you are finished
- We can append to files instead of overwriting
- There is a shorter syntax to create a stream and open a file at the same time