File Streams

Due this week

- **HW 5**
  - Write solutions in VSCode and paste in **CodeRunner**.
  - Extra-credit
  - Zip your .cpp files and submit on canvas. Check the due date! **No late submissions!!**
- **Mandatory Grading Interview -** Oct 16th – 20th!
- **Quiz 5.** Check the due date! **No late submissions!!**

Today

- Streams
  - Reading from files
  - Writing to files
- Code example

Streams

Streams

Oil pipelines, like the one in the picture is used to transport oil between cities and even countries

Think of a stream as a pipe, which is used to carry something

Streams

So, based on our understanding of an oil pipeline, we know

- It’s used to carry oil (or any liquid in general)
- Takes oil from place A (for ex: Texas)
- Provides oil to place B (for ex: Colorado)

Well, is an oil pipeline used to store oil ??

Streams

Producer / Source

Carrier / Medium

Consumer / Target

Streams

Some other examples you've seen

- Conveyor Belt
- A queue for a breakfast buffet!
- Roads in between cities

They are all examples of streams, because they transport/carry something from a source to a destination.

Well that's probably how the name **stream** came by!

Streams - In Programming

- The C++ input/output library is based on the concept of *streams*.
- An *input stream* is a source of data. (*Think of keyboard inputs*)
- An *output stream* is a destination for data. (*Think of outputs on your monitor*)

- The most common sources and destinations for data are the files on your hard disk.
  - You need to know how to read/write disk files to work with large amounts of data that are common in business, administrative, graphics, audio, and science/math programs

Streams

- You’ve seen and used streams before! Think of what they are…

Streams

- You've seen and used streams before! Think of what they are…

CIN and COUT - they are streams

- Cin - An input stream from your *keyboard* (source) to a *variable in your program* (destination)
- Cout - An output stream from *a variable in your program* (source) to your *monitor* (destination)

Remember the syntax?
<< : Used with cout (known as insertion operator)
>> : Used with cin (known as extraction operator)

Streams - Files on your system

You can also read and write files stored on your hard disk:

- plain text files
- binary information (a binary file)
  - Such as images or audio recording

To read/write files, you use *variables* of the stream types:

`ifstream` for input from plain text files.
`ofstream` for output to plain text files.
`fstream` for input and output from binary files.

You must `#include <fstream>`

Opening a Stream

- To read/write anything from a file stream, you need to *open* the stream.
- *Opening a stream* means enabling either a read/write from/to a file.

```
// to open a stream for reading a file
ifstream is;
is.open("read.txt");

// to open another stream for writing to a file
ofstream os;
os.open("write.txt");
```

Opening a Stream

- That easy?? Think of what could go wrong when you create a stream.

Opening a Stream

- That easy?? Think of what could go wrong when you create a stream.

Things to keep in mind

- Maybe file doesn’t exist, or maybe it’s in another directory. Provide the correct path to the file.
- Maybe you don’t have the permissions to read or write to it. (Little advanced, but read on it!)

The open method sets a flag to indicate if a file could be open or not!

Opening a Stream

Keeping the conditions that could fail,


```
// to open a stream for reading a file
ifstream is;
is.open(<correct path to file>);

if (is.fail()) {
   cout << “Could not open the file to read” << endl;
}
```

File Path Names

File names can contain directory path information, such as:

MAC

```
in_file.open("~/nicework/input.dat");
```

Windows

```
in_file.open("c:\\nicework\\input.dat");
```

When you specify the file name as a string literal, and the name contains backslash characters (as in Windows), you must **supply each backslash *twice*** to avoid having unintended *escape characters* in the string.

\\ becomes a single \ when processed by the compiler.

Closing a Stream

- When the program ends, all streams that you have opened will be automatically closed.
- You *can* manually close a stream with the **close** member function:

```
is.close();
```

Review of some functions with streams

- open - open a stream for either reading/writing or both
- close - close a stream, after use
- fail - check if open operation succeeded or not

Check your understanding

- Order the sequence correctly

1. Check if opening a stream succeeded
2. Close the stream
3. Create a variable for the stream
4. Read some data from the stream
5. Open the stream

Check your understanding

- Order the sequence correctly

1. Check if opening a stream succeeded
2. Close the stream
3. Create a variable for the stream
4. Read some data from the stream
5. Open the stream

3 -> 5 -> 1 -> 4 -> 2

Reading from a stream

Functions that helps us read from a stream

- The extraction operator >>
- Getline

Reading from a stream

```
string name;
int number;
is >> name >> number;
```

The example reads a name (string) and a number (int) from the stream

Note how in_file has replaced cin

No difference when it comes to reading using **>>**.

Reading from a stream

- The **>>** operator returns a “not failed” condition, allowing you to combine an input statement and a test.
- A “failed” read yields a **`false`** and a “not failed” read yields a **`true`**.

```
if (is >> name >> number)
{
   // Process input
}
```

Reading from a stream

- You can even read ALL the data from a file because running out of things to read causes that same “failed state” test to be returned:

```
while (is >> name >> number)
{
   // Do anything with what you read
}
```

Writing to a stream

Functions that helps us write to a stream

- The insertion operator <<

```
string name = “CSCI”;
int number = 1300;
os << name << ” “ << number;
```

Test your understanding

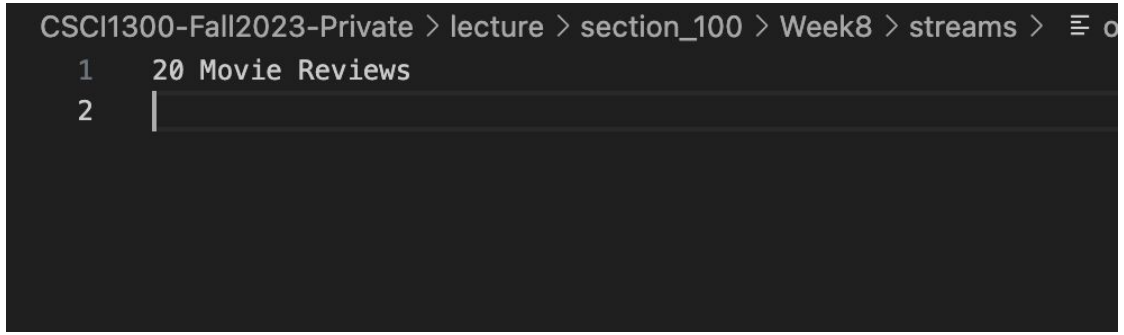
Pick the correct options

1. You can read from an ofstream variable
2. You don’t need to have the file created when you use ofstream.open
3. You can’t open the stream after you’ve closed it

More Streams

Test your understanding

```
ifstream is;
is.open("file.txt");
string str;
is >> str;
```

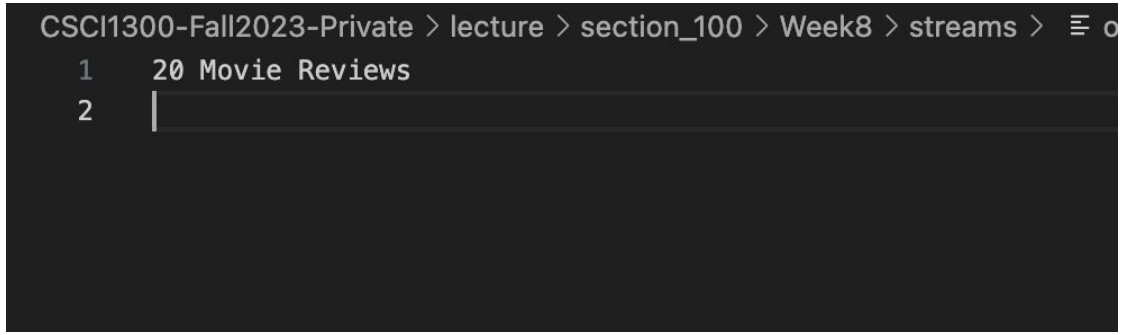


What does the string contain?

1. 20 Movie Reviews
2. 20
3. Error, because 20 is an integer

Test your understanding

```
ifstream is;
is.open("file.txt");
string str;
is >> str;
```



What does the string contain?

1. 20 Movie Reviews
2. **20 -> Correct answer**
3. Error, because 20 is an integer

Getline vs >>

- >> Reads only until the first space or the first newline character
- How do we read a line entirely? **Getline**

Syntax

```
ifstream is;
is.open("file.txt");
string str;
getline(is, str); ->
```

This reads an entire line from the text file.

Getline vs >>

```
4
5   int main() {
6       string first_string;
7       cin >> first_string;
8       cout << "You entered : " << first_string << endl;
9       return 0;
10  }
```

- What's the value of `first_string` when the input is
  - hello world
  - 22 Jump Street
  - A-long-string-with-hyphens-in-middle
  - F.R.I.E.N.D.S

Getline vs >>

```
4
5   int main() {
6       string first_string;
7       getline(cin, first_string);
8       cout << "You entered : " << first_string << endl;
9       return 0;
10  }
```

- What's the value of `first_string` when the input is
  - hello world
  - 22 Jump Street
  - A-long-string-with-hyphens-in-middle
  - F.R.I.E.N.D.S

Getline vs >>

Getline

Fine. I'll take in the rest until newline

>>

I'm gonna stop if a see a space or newline

Some text to read and process

Getline

- Okay, getline reads an entire line (reads until it sees a newline character).
- How do I read a bunch of lines from a file??

Getline

- Okay, getline reads an entire line (reads until it sees a newline character).
- How do I read a bunch of lines from a file?? **WHILE LOOP**

Syntax

```
ifstream is;
is.open("file.txt");
string str;
while (getline(is, str)) {
   // At each iteration, str will be a line from the file
}
```

Movie Reviews: Example

- Suppose we have a file with contents like this..

You're given the task of reading this file

What's your thought process before attempting to read this file?

```
Movie Names,Year,Rating,Votes
Guardians of the Galaxy,2014,8.1,757074
Prometheus,2012,7,485820
Split,2016,7.3,157606
Sing,2016,7.2,60545
Suicide Squad,2016,6.2,393727
The Great Wall,2016,6.1,56036
La La Land,2016,8.3,258682
Mindhorn,2016,6.4,2490
The Lost City of Z,2016,7.1,7188
Passengers,2016,7,192177
Fantastic Beasts and Where to Find Them,2016,7.5,232072
Hidden Figures,2016,7.8,93103
Rogue One,2016,7.9,323118
Moana,2016,7.7,118151
Colossal,2016,6.4,8612
The Secret Life of Pets,2016,6.6,120259
Hacksaw Ridge,2016,8.2,211760
Jason Bourne,2016,6.7,150823
Lion,2016,8.1,102061
Arrival,2016,8,340798
Gold,2016,6.7,19053
Manchester by the Sea,2016,7.9,134213
Hounds of Love,2016,6.7,1115
Trolls,2016,6.5,38552
Independence Day: Resurgence,2016,5.3,127553
```

Movie Reviews: Example

- Some questions you need to answer before attempting

  How is it structured?

  What are the contents of each line?

  What about their types?

```
Movie Names,Year,Rating,Votes
Guardians of the Galaxy,2014,8.1,757074
Prometheus,2012,7,485820
Split,2016,7.3,157606
Sing,2016,7.2,60545
Suicide Squad,2016,6.2,393727
The Great Wall,2016,6.1,56036
La La Land,2016,8.3,258682
Mindhorn,2016,6.4,2490
The Lost City of Z,2016,7.1,7188
Passengers,2016,7,192177
Fantastic Beasts and Where to Find Them,2016,7.5,232072
Hidden Figures,2016,7.8,93103
Rogue One,2016,7.9,323118
Moana,2016,7.7,118151
Colossal,2016,6.4,8612
The Secret Life of Pets,2016,6.6,120259
Hacksaw Ridge,2016,8.2,211760
Jason Bourne,2016,6.7,150823
Lion,2016,8.1,102061
Arrival,2016,8,340798
Gold,2016,6.7,19053
Manchester by the Sea,2016,7.9,134213
Hounds of Love,2016,6.7,1115
Trolls,2016,6.5,38552
Independence Day: Resurgence,2016,5.3,127553
```

We can read the lines, but how to break them down into parts?

Stringstream

- Similar to file streams, C++ has a string stream, to read and manipulate strings.
- What it allows you to do is
  - Work with strings as though they were streams
  - Concatenate strings
  - Tokenize strings
  - And much more…

Stringstream - Tokenize

- Tokenize refers to breaking a string into smaller parts that are separated by a delimiter
- When we tokenize this string with the delimiter `comma`
  `“Guardians of the Galaxy,2014,8.1,757074”`

  We get 4 parts
  ```
  Guardians of the Galaxy
  2014
  8.1
  757074
  ```

Stringstream - Tokenize

- `string str = “Guardians of the Galaxy,2014,8.1,757074”`
  `stringstream ss(str);`
  `string token;`

Contents of the string stream	Tokenize	Token
Guardians of the Galaxy,2014,8.1,757074	getline(ss, token, ‘,’)	Guardians of the Galaxy
2014,8.1,757074	getline(ss, token, ‘,’)	2014
8.1,757074	getline(ss, token, ‘,’)	8.1
757074	getline(ss, token, ‘,’)	757074

Movie Reviews: Complete Example