

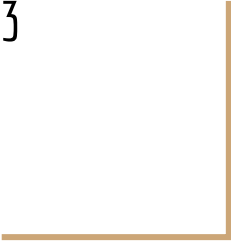
Reminders/Announcements

- Lab 8 is due this Friday by 8:30pm
- Homework 3 is due next Monday November 21 by 8:30pm
- Makeup Submission for Lab 5 is now open. Will be due next Wednesday, 23 by 8:30pm.
- See Canvas Announcement for information on a summer internship opportunity with the Department of Defense.



Chapter 9: String Streams

CSE 2010 - Week 13



Review: Reading From Files

- Make sure to #include <fstream>
- Step 1: Constructing a stream to read files.

```
ifstream streamName;
```

- Step 2: Connect the stream to the source.
Always double check that it opened properly

```
streamName.open(filename);  
if(streamName.fail())
```

- Step 3: Read from the stream;

```
datatype var;  
streamName >> var;
```

- Step 4: Disconnect the source

```
streamName.close();
```

- Step 5: Destroy stream....is done automatically :)

```
1 /*  
2  * Filename: read_file.cpp  
3  * Example program to see how to open a file and read from it  
4  */  
5 #include <iostream>  
6 #include <fstream>//needed to read from file  
7  
8 using namespace std;  
9  
10 int main()  
11 {  
12     //create input file stream  
13     ifstream input_data;  
14     //connect to a file  
15     input_data.open("input1.txt");  
16     //check that it opened properly  
17     if(input_data.fail())  
18         cout << "Error opening \"input1.txt\"\n";  
19     else{  
20         //prepare to read from the stream  
21         //since I will be reading in ints, need an int  
22         int n;  
23         //now I can start reading  
24         //I want to keep reading to the end of the file, so I will  
25         //use a while loop  
26         while(input_data >> n){  
27             cout << "Just read: " << n << "\n";  
28         }  
29         cout << "All done reading file.\n";  
30     }  
31     //disconnect file  
32     input_data.close();  
33  
34     return 0;  
35 }
```

Review: Writing to Files

- Make sure to #include <fstream>
- Step 1: Constructing a stream to write to files.

```
ofstream streamName;
```

- Step 2: Connect the stream to the sink. Always double check that it opened properly
- If the file doesn't exist, the program will simply create a new file with the name you provide.

```
streamName.open(filename);  
if(streamName.fail())
```

- Step 3: Write to the stream;

```
datatype var;  
//give variable some value  
streamName << var;
```

- Step 4: Disconnect the sink

```
streamName.close();
```

- Step 5: Destroy stream....is done automatically :)

```
1 /*  
2 * Example program writing 20 random numbers to a file  
3 */  
4 #include <iostream>  
5 #include <fstream>  
6 #include <cstdlib>  
7 #include <ctime>  
8  
9 using namespace std;  
10  
11 int main()  
12 {  
13     //declare output file object  
14     ofstream out;  
15     //this file doesn't exist, so the program will create it  
16     out.open("unsorted.txt");  
17     if(out.fail()) cout << "Error opening file.\n";  
18     else{  
19         cout << "Writing 20 random numbers to the file...\n";  
20         //seed for random # generator  
21         srand(time(NULL));  
22  
23         int random;  
24         for(int i = 0; i < 20; i++){  
25             random = rand() % 100;  
26             out << random << "\n";  
27         }  
28         cout << "Writing complete\n";  
29     }  
30     out.close();  
31  
32     return 0;  
33 }
```

Review: Reading from and writing to the same file

- Make sure to `#include <fstream>`
- Step 1: Constructing a stream to read & write files.

```
fstream streamName;
```

- Step 2: Connect the stream to the source. Always double check that it opened properly

```
streamName.open(filename, ios::in|ios::out);
```

```
if(streamName.fail())
```

```
    Cout << "Error opening file with name.\n";
```

- Step 3: Depending on what you want to do, use the appropriate operator;

```
streamName >> var; //will read
```

```
streamName << var; // will write
```

```
//white space or newline is reached
```

- Step 3.1: If you expect to write after reading, you will have to also use the clear function to clear the stream and set the eof bit to false, allowing for input at the end of the file. (`streamName.clear()`)
- Step 4: Disconnect the source

```
streamName.close();
```

- Step 5: Destroy stream....is done automatically :)

Getting Whole Lines of Text

- When we use the >> operator to get input from a file, it will implicitly stop at the first white space or new line.
- How can we get a whole line of text, including spaces?
 - getline() function we have been using.
- Note the difference in the following methods:

input.txt

```
The quick brown fox jumps (over) the lazy dog.  
Sally sells seashells by the seashore.
```

```
ifstream in;  
in.open("input.txt");  
string input;  
in >> input;  
cout << input << "\n";
```

Output: The

```
ifstream in;  
in.open("input.txt");  
string input;  
getline(in, input);  
cout << input << "\n";
```

Output: The quick brown fox jumps (over) the lazy dog.

String Streams

- With **console streams (iostream)**, the contents of the keyboard and monitor are the stream of data.
- With **file streams (fstream)**, the contents of a file are the stream of data.
- Now let's look at **string streams**, where the contents of a string are the stream of data.
- The `<sstream>` library will allow us to treat C++ string objects like they were streams.
- Why would you ever want to do such a thing?
 - Buffer up output for later display
 - Parse out the pieces of a string
 - Data type conversions

Using String Streams

- `#include <sstream> & <iostream>`
- Syntax:

```
sstream streamName;  
streamName << //will add/output to the stream  
streamName >> //will use stream as input
```


String Streams - Convert numbers to strings

- With string streams, we can convert numbers into strings using << and >>

```
stringstream ss;  
int num = 12345;  
ss << num; // Will store value of num into ss as a string  
string strNum; //empty string variable  
ss >> strNum; //will store the string "12345" into strNum
```

- We can convert strings into numbers using << and >>

```
stringstream ss;  
string strNum = "12345";  
ss << strNum; //stores string value into stream  
int num; //int variable  
ss >> num; //will store 12345 into num as an integer
```

- Note: Trying to convert "cat" into an integer or double would result in the value of 0.

String Streams - Convert numbers to strings

- With string streams, we can convert numbers into strings using << and >>

```
stringstream ss;  
int num = 12345;  
ss << num; // Will store value of num into ss as a string  
string strNum; //empty string variable  
ss >> strNum; //will store the string "12345" into strNum
```

- We can also convert strings into numbers using << and >>

```
stringstream ss;  
string strNum = "12345";  
ss << strNum; //stores string value into stream  
int num; //int variable  
ss >> num; //will store 12345 into num as an integer
```

- Note: Trying to convert "cat" into an integer or double would result in the value of 0.

String Streams - Parsing

- We can parse a string of many values into different variables of different datatypes.

```
string info = "Obi 4 03-15-2018";
stringstream ss;
ss << info; // ss = "Obi 4 03-15-2018"
string name, DOB;
int age;
ss >> name >> age >> DOB; //stores strings into variables
//name = "Obi"
//age = 4
//DOB = "03-15-2018"
```

- Note: Storing the values into the variables does not clear the stream. So if we did

```
ss << "Hello World!";
//ss = "Obi 4 03-15-2018 Hello World!"
```

String Streams - Outputting streams

- Say we want to output the entire contents of a stream

```
string info = "Obi 4 03-15-2018";
stringstream ss;
ss << info; // ss = "Obi 4 03-15-2018"
cout << ss; //will give us an error. ss is a STREAM

//we need to convert to a string
cout << ss.str();
//str() is a member functions to convert to a string
```



Let's look at an example!
(Employee Example on Canvas)



Streams Review

- Streams in C++ are objects that represent a device on which input and output operations can be performed.
- Streams have states made up of bits (eof,badbit,failbit), and all bits must not be set for it to be in a goodstate and usable.
- We can use file streams to read from and write to files.
 - `#include <fstream>`
 - `ifstream` - read from files
 - `ofstream` - write to files
 - `fstream` - read and write to files
- String Streams:
 - Allow us to treat strings as streams.
 - We can use them to convert data types.
 - We can also use them to parse through a string and place different values in different variables.