CSCI 200: Foundational Programming Concepts & Design Lecture 14

Input Streams

Reading Data Using Files

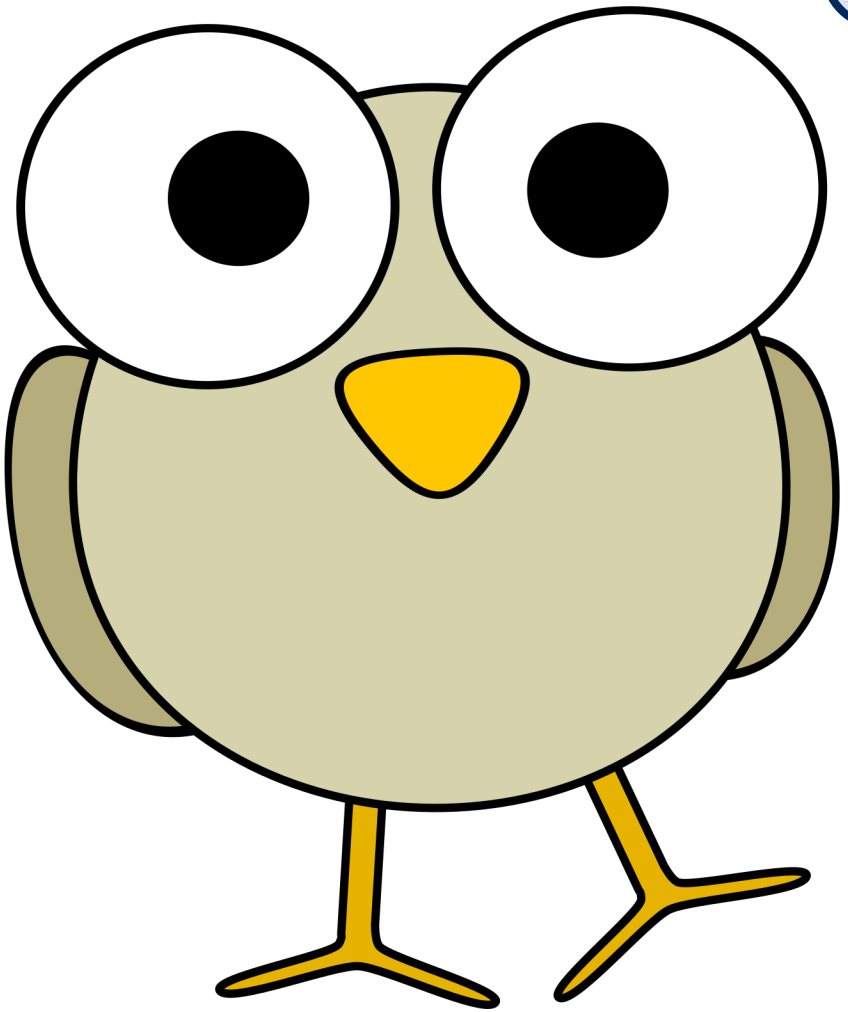
Input Paradigms & Validation

Download Lecture 14 starter code

Previously in CSCI 200

- Object-Oriented Programming: state encapsulated in an object and only object can modify its state
- Create UML diagram for pseudocode of a Class with data members to store state and state is modified via methods
- Objects are an instance of a Class

Questions?



Learning Outcomes For Today

- Recite the six steps to properly use a file stream for reading or writing.
- Write a program that implements the corresponding pseudocode using file streams.
- Define REPL and perform read operations conforming to common input paradigms.
- Create a program that validates user input and removes the need for a cooperative smart user.

On Tap For Today

- Streams
- Reading Files
  - Reading Paradigms
  - Stream Errors
- Practice

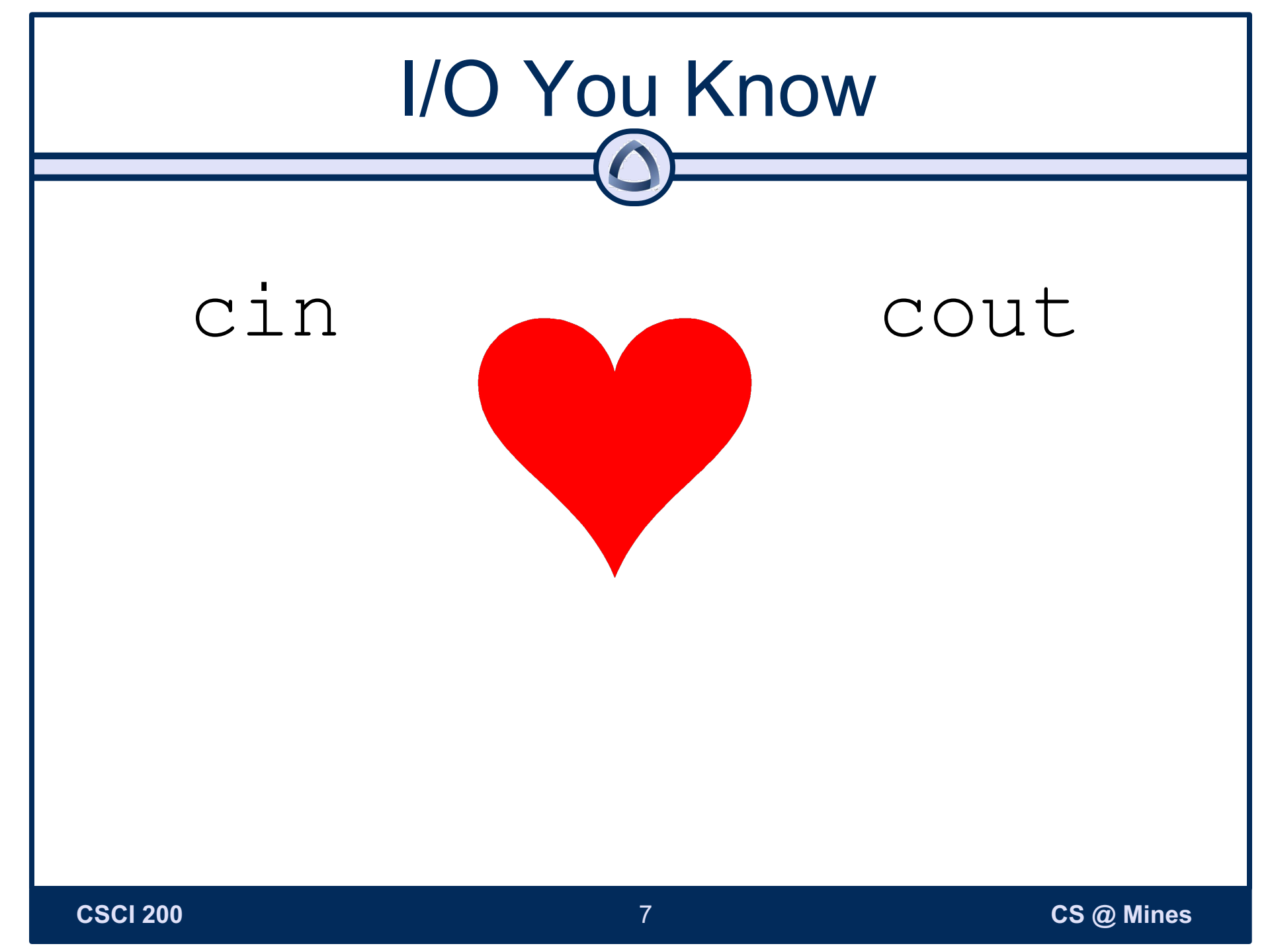
On Tap For Today

- Streams

- Reading Files
  - Reading Paradigms
  - Stream Errors

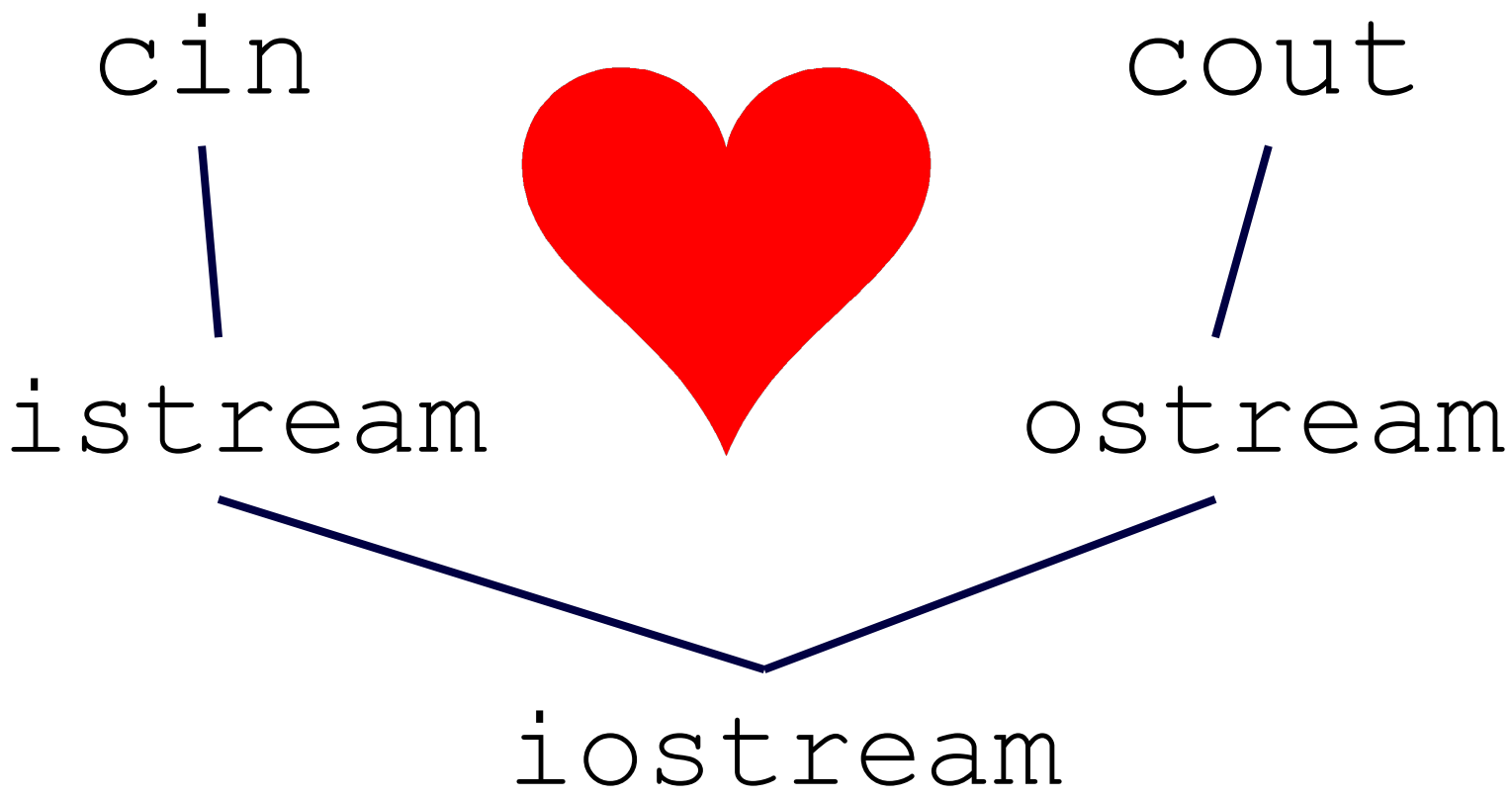
- Practice

I/O You Know



`cin` `cout`

I/O You Know



`#include <iostream>`

- `cin`
- `cout`

- `cin` and `cout` are “streams”

Streams come in

Streams go out

What is a file?

- A logical, coherent stream of bits on some persistent medium.

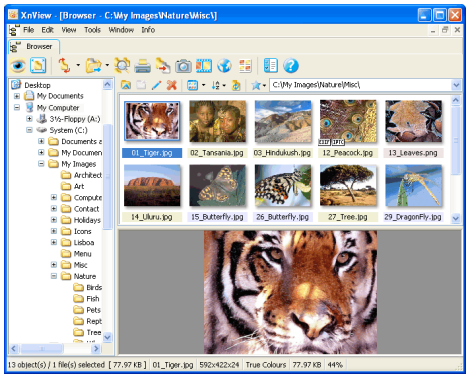
embarrassing photo of you

What is a Digital Photo?

- It's a file, a stream of bits. *Nothing more*.

- But…
  - When a program is used to read that stream of bits...

```
00101010
10101011
10110110
11101011
```



  - …that files becomes a visible photo.

What is an .mp3 file?

- It's a file, a stream of bits. *Nothing more*.

- But…
  - When a program is used to read that stream of bits...

```
00101010
10101011
10110110
11101011
```

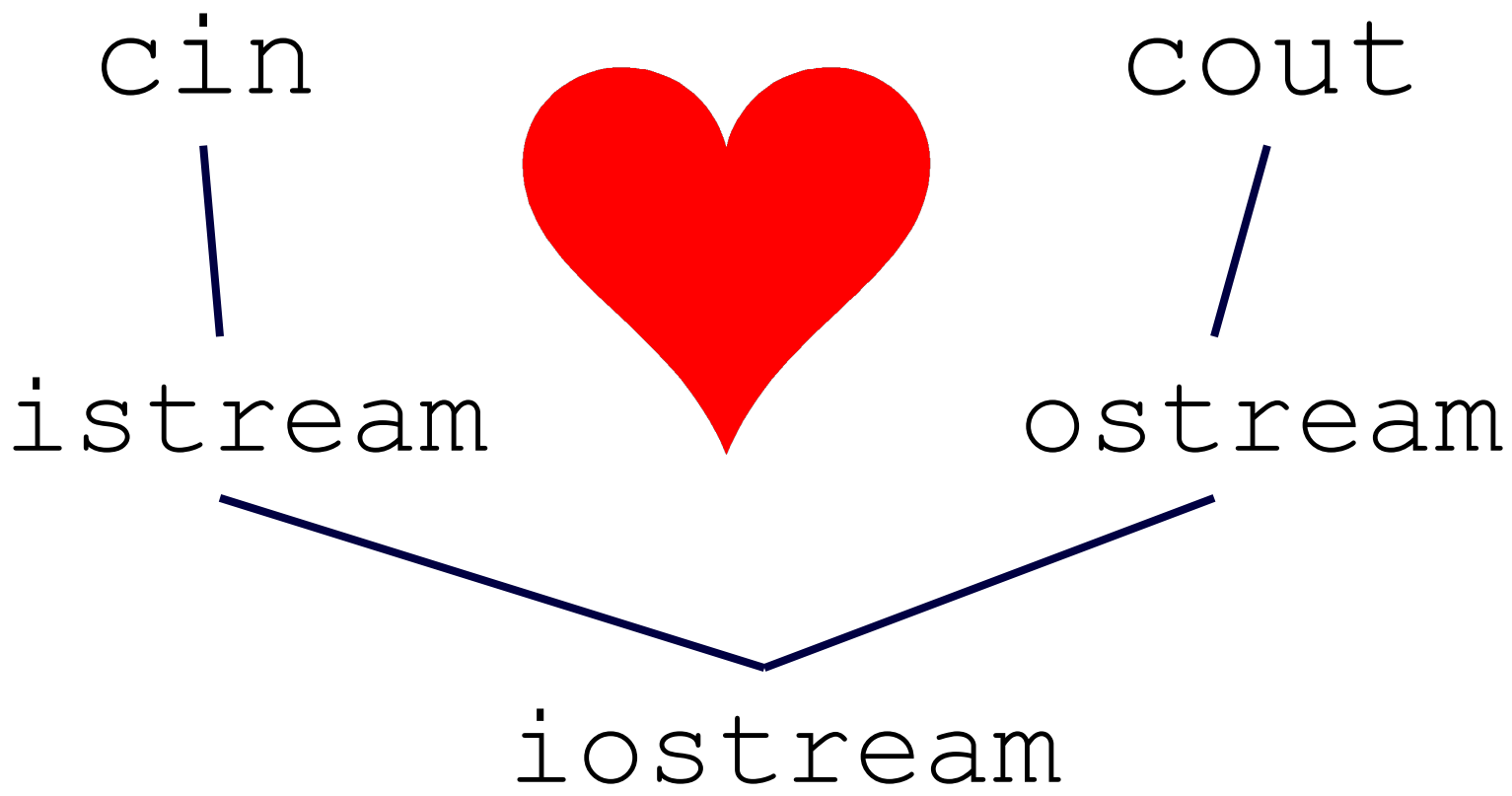
  - …that file becomes something audible.

Files are “Streams”

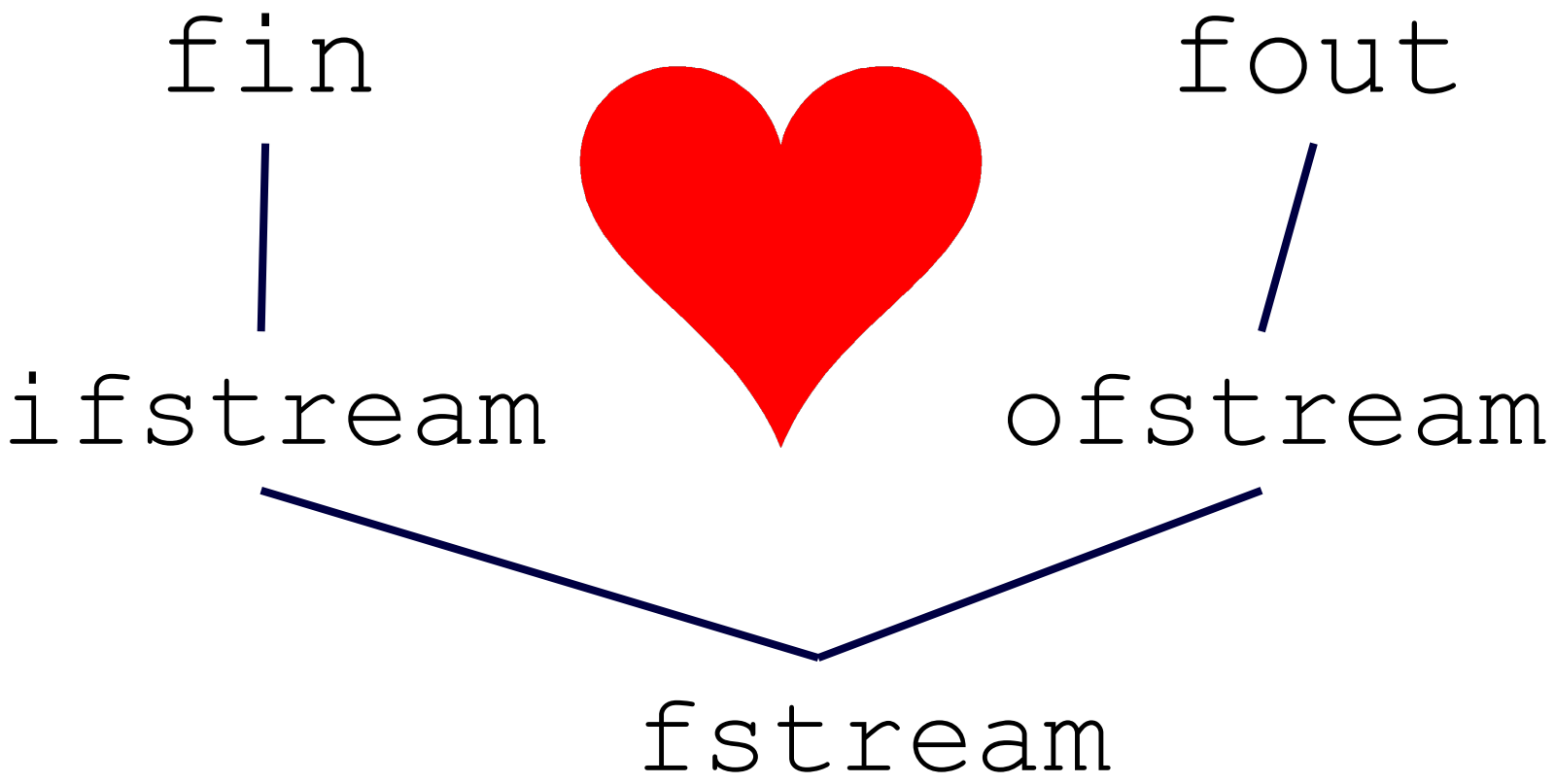
- Files can be read by your program.

- Files can be written by your program.

I/O You Know



I/O To Learn



On Tap For Today

- Streams
- Reading Files
  - Reading Paradigms
  - Stream Errors
- Practice

Reading From A File

- “Computer, create an input filestream called myInput that’ll let me read (stream) content from filename.ext”

```
ifstream myInput( "filename.ext" );
```

Use it like `cin`

- "Computer, read this string from the keyboard"

```
cin >> userString;
```

- "Computer, read this string from my file"

```
ifstream myInput( "filename.ext" );
myInput >> fileString;
```

Extraction operator

How >> and Streams Work

- Think of this data as “streaming” to your program

datafile.txt

```
12 34 56
78 90 01
```

How >> and Streams Work

- Think of this data as “streaming” to your program

datafile.txt

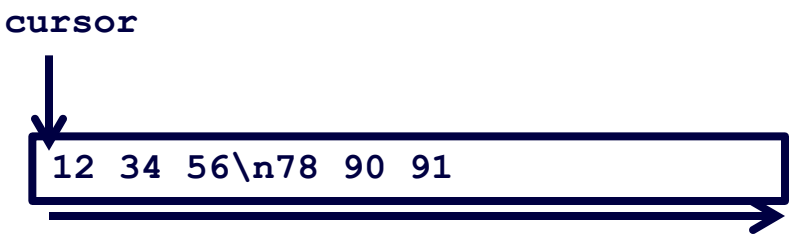
```
12 34 56
78 90 01
```

```
ifstream fileIn( "datafile.txt" );
```

```
12 34 56\n78 90 91
```

How >> and Streams Work

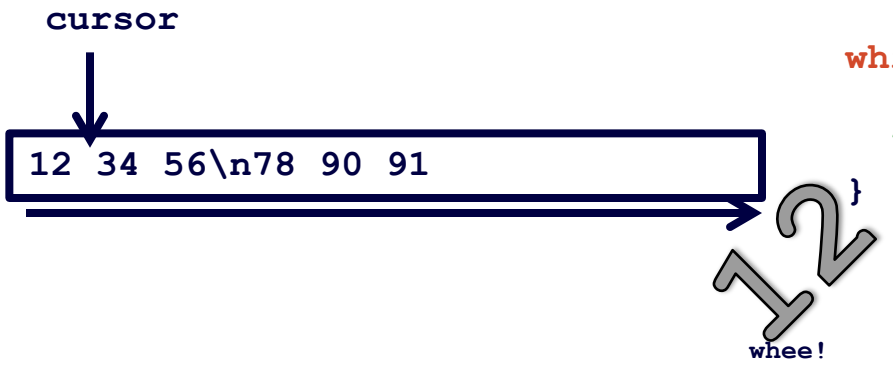
- "Computer, read the next value after the cursor up until a whitespace character"




```
int x;
while( fileIn >> x ) {
  // do something
}
```

How >> and Streams Work

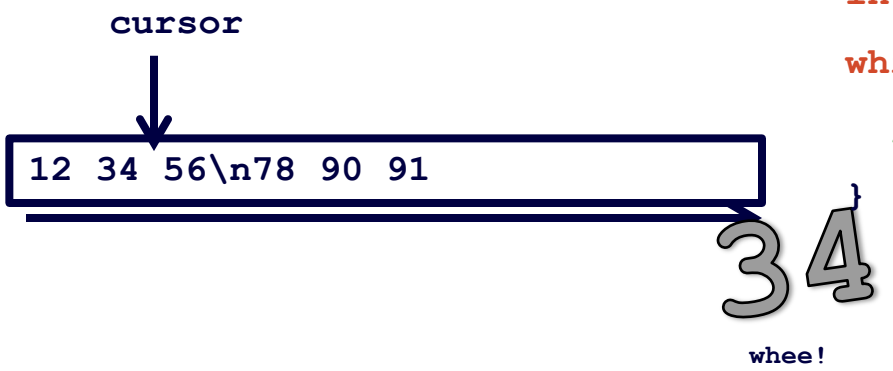
- "Computer, read the next value after the cursor up until a whitespace character"




```
int x;
while( fileIn >> x ) {
  // do something
}
```

How >> and Streams Work

- "Computer, read the next value after the cursor up until a whitespace character"



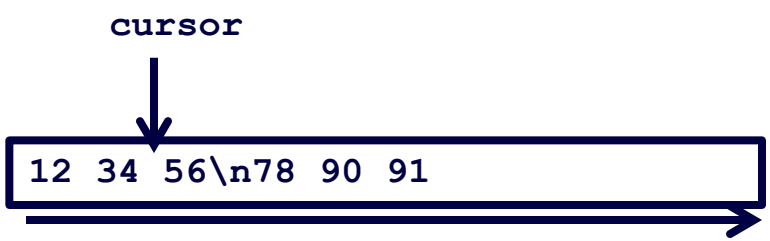

```
int x;
while( fileIn >> x ) {
  // do something
}
```

34

whee!

How get() and Streams Work

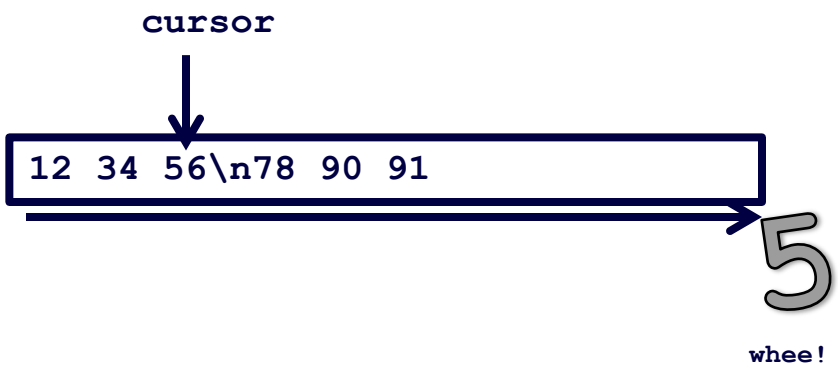
- "Computer, read ONE character after the cursor"




```
char x;
while( !fileIn.eof() ) {
  x = fileIn.get();
  // do something
}
```

How get() and Streams Work

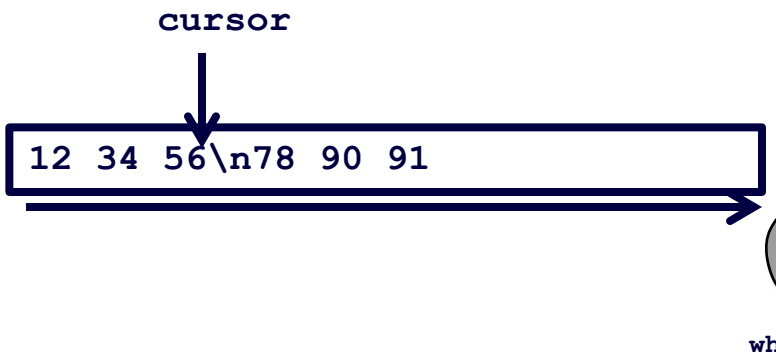
- "Computer, read ONE character after the cursor"




```
char x;
while( !fileIn.eof() ) {
  x = fileIn.get();
  // do something
}
```

How get() and Streams Work

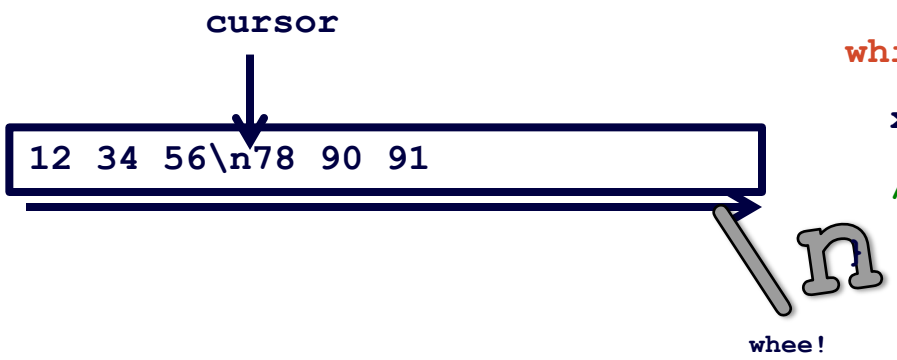
- "Computer, read ONE character after the cursor"




```
char x;
while( !fileIn.eof() ) {
  x = fileIn.get();
  // do something
}
```

How get() and Streams Work

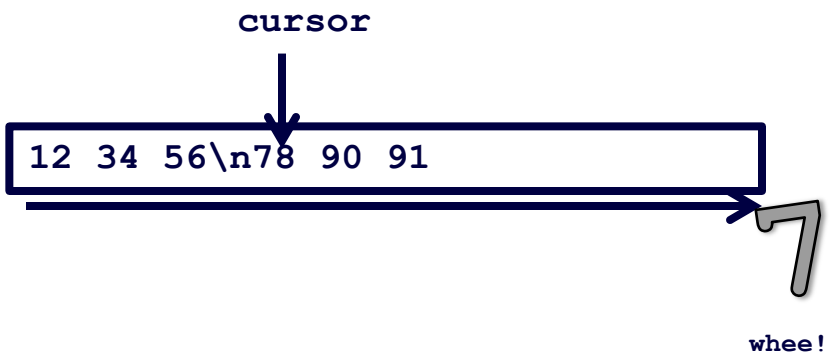
- "Computer, read ONE character after the cursor"




```
char x;
while( !fileIn.eof() ) {
  x = fileIn.get();
  // do something
```

How get() and Streams Work

- "Computer, read ONE character after the cursor"




```
char x;
while( !fileIn.eof() ) {
  x = fileIn.get();
  // do something
}
```

Reading From A File

- Include the fstream library
- Declare/open the input filestream
- Check for an error
- Read data
- Close the file

Reading From A File

- Include the fstream library

```
#include <fstream>

using namespace std;
```

- Declare/open the input filestream
- Check for an error
- Read data
- Close the file

Reading From A File

- Include the fstream library

```
#include <fstream>

using namespace std;
```

- Declare/open the input filestream

```
ifstream fileIn( "myData.txt" );
```

- Check for an error

- Read data

- Close the file

Reading From A File

- Include the fstream library

```
#include <fstream>

using namespace std;
```

- Declare/open the input filestream

```
ifstream fileIn;

fileIn.open( "myData.txt" );
```

- Check for an error
- Read data
- Close the file

Reading From A File

- Include the fstream library

```
#include <fstream>

using namespace std;
```

- Declare/open the input filestream

```
ifstream fileIn( "myData.txt" );
```

- Check for an error

- Read data

- Close the file

Reading From A File

- Include the fstream library

```
#include <fstream>

using namespace std;
```

- Declare/open the input filestream

```
ifstream fileIn( "myData.txt" );
```

- Check for an error

```
if( fileIn.fail() ) { ... }
```

- Read data

- Close the file

Reading From A File

- Include the fstream library

```
#include <fstream>

using namespace std;
```

- Declare/open the input filestream

```
ifstream fileIn( "myData.txt" );
```

- Check for an error

```
if( !fileIn ) { ... }
```

- Read data

- Close the file

Reading From A File

- Include the fstream library

```
#include <fstream>

using namespace std;
```

- Declare/open the input filestream

```
ifstream fileIn( "myData.txt" );
```

- Check for an error

```
if( !fileIn.is_open() ) { ... }
```

- Read data

- Close the file

Reading From A File

- Include the fstream library

```
#include <fstream>

using namespace std;
```

- Declare/open the input filestream

```
ifstream fileIn( "myData.txt" );
```

- Check for an error

```
if( fileIn.fail() ) { ... }
```

- Read data

- Close the file

Reading From A File

- Include the fstream library

```
#include <fstream>

using namespace std;
```

- Declare/open the input filestream

```
ifstream fileIn( "myData.txt" );
```

- Check for an error

```
if( fileIn.fail() ) { ... }
```

- Read data

```
fileIn >> x;
```

- Close the file

Reading From A File

- Include the fstream library

```
#include <fstream>

using namespace std;
```

- Declare/open the input filestream

```
ifstream fileIn( "myData.txt" );
```

- Check for an error

```
if( fileIn.fail() ) { ... }
```

- Read data

```
fileIn >> x;
```

- Close the file

```
fileIn.close();
```

Reading Files Boilerplate

```
#include <fstream>
#include <iostream>
using namespace std;
int main() {
  ifstream myDataIn( "FILENAME" );
  if( myDataIn.fail() ) {
    cerr << "Could not open \"FILENAME\"" << endl;
    return -1;
  }
  char x; // or int x, double x, etc.
  while( !myDataIn.eof() ) {
    myDataIn >> x;
    // do marvelous things and print results
  }
  myDataIn.close();
  return 0;
```

On Tap For Today

- Streams
- Reading Files
  - Reading Paradigms
  - Stream Errors
- Practice

Reading Files Boilerplate

```
#include <fstream>
#include <iostream>
using namespace std;
int main() {
  ifstream myDataIn( "FILENAME" );
  if( myDataIn.fail() ) {
    cerr << "Could not open \"FILENAME\"" << endl;
    return -1;
  }
  char x; // or int x, double x, etc.
  while( !myDataIn.eof() ) {
    myDataIn >> x;
    // do marvelous things and print results
  }
  myDataIn.close();
  return 0;
```

Reading Files Boilerplate

```
#include <fstream>
#include <iostream>
using namespace std;
int main() {
  ifstream myDataIn( "FILENAME" );
  if( myDataIn.fail() ) {
    cerr << "Could not open \"FILENAME\"" << endl;
    return -1;
  }
  char x; // or int x, double x, etc.
  while( !myDataIn.eof() ) {
    myDataIn >> x;
    // do marvelous things and print results
  }
  myDataIn.close();
  return 0;
```

Read

Evaluate

Print

Loop

Reading Paradigms

- Dependent on how data in our file is formatted

Reading Paradigms

- Dependent on how data in our file is formatted
  - First Line of file = number of lines (records)
  - Counter-controlled loop

datafile.txt

```
4
1 3
2 4
5 6
7 8
```

```
ifstream fileIn( "datafile.txt" );
int numLines;
fileIn >> numLines;
for( int i = 0; i < numLines; i++ ) {
  int x, y;
  fileIn >> x >> y;
  // do magic
}
```

Reading Paradigms

- Dependent on how data in our file is formatted
  - Last Line of file = special value to indicate data end
  - Sentinal-controlled loop

datafile.txt

```
4
1
2
5
-9999
```

```
const int SENTINAL_VALUE = -9999;
ifstream fileIn( "datafile.txt" );
int x;
while( true ) {
  fileIn >> x;
  if( x == SENTINAL_VALUE ) break;
  // do magic
}
```

Reading Paradigms

- Dependent on how data in our file is formatted
  - No knowledge within file
  - End-of-data loop

datafile.txt

```
4
1
2
5
3
```

```
ifstream fileIn( "datafile.txt" );
while( !fileIn.eof() ) {
  int x;
  fileIn >> x;
  // do magic
}
```

On Tap For Today

- Streams
- Reading Files
  - Reading Paradigms
  - Stream Errors
- Practice

Input Error

A.
```
int i, j;
cin >> i >> j;
```

B.
```
double x, y;
cin >> x >> y;
```

C.
```
char c1, c2;
cin >> c1 >> c2;
```

Data entered

```
1.2.3
```

- What is the value of each variable?

Input Errors Occur When..

- Input != variable type
- Stream variable placed in error state
- All future inputs ignored

- To Do
  1. Test whether reading variable is in error state
  2. If yes, print to cerr and exit

Example

```
int main() {
  int x, y;
  cin >> x >> y;
  if( cin.fail() ) {
    cerr << "error encountered from read" << endl;
    return -1;
  }
  // do other stuff
}
```

Input Errors Occur When..

- Input != variable type
- Stream variable placed in error state
- All future inputs ignored

- To Do
  1. Test whether reading variable is in error state
  2. If yes, print to cerr and ~~exit~~ goto end of line and try again

Example

```
int main() {
  int x, y;
  while( true ) { // loop until we get good data
    cin >> x >> y;

    if( !cin.fail() ) break;  // we succeeded, break out of loop

    cerr << "error encountered from read" << endl;
    cin.clear(); // clear error

    char badChar; // clear out rest of input line
    do { badChar = cin.get(); } while( badChar != '\n' );
    cout << "Enter two integers: ";
  }
  // do other stuff
}
```

Accepting Only Valid Values

- May receive data of proper data type, but may be wrong value

```
int main() {
  while( true ) { // loop until we get good data
    char userValue;
    cout << "Enter q to quit: ";
    cin >> userValue;
    if( userValue == 'q' ) break;

    cout << "Invalid value." << endl;
  }
}
```

On Tap For Today

- Streams
- Reading Files
  - Reading Paradigms
  - Stream Errors
- Practice

To Do For Next Time

- Set2 due tomorrow
- Be continuing with zyBooks