# CSCI 200: Foundational Programming Concepts & Design Lecture 15

Input / Output Streams

Reading / Writing Data Using Files

Download Lecture 15 starter code

Complete Set 2 Feedback in Canvas

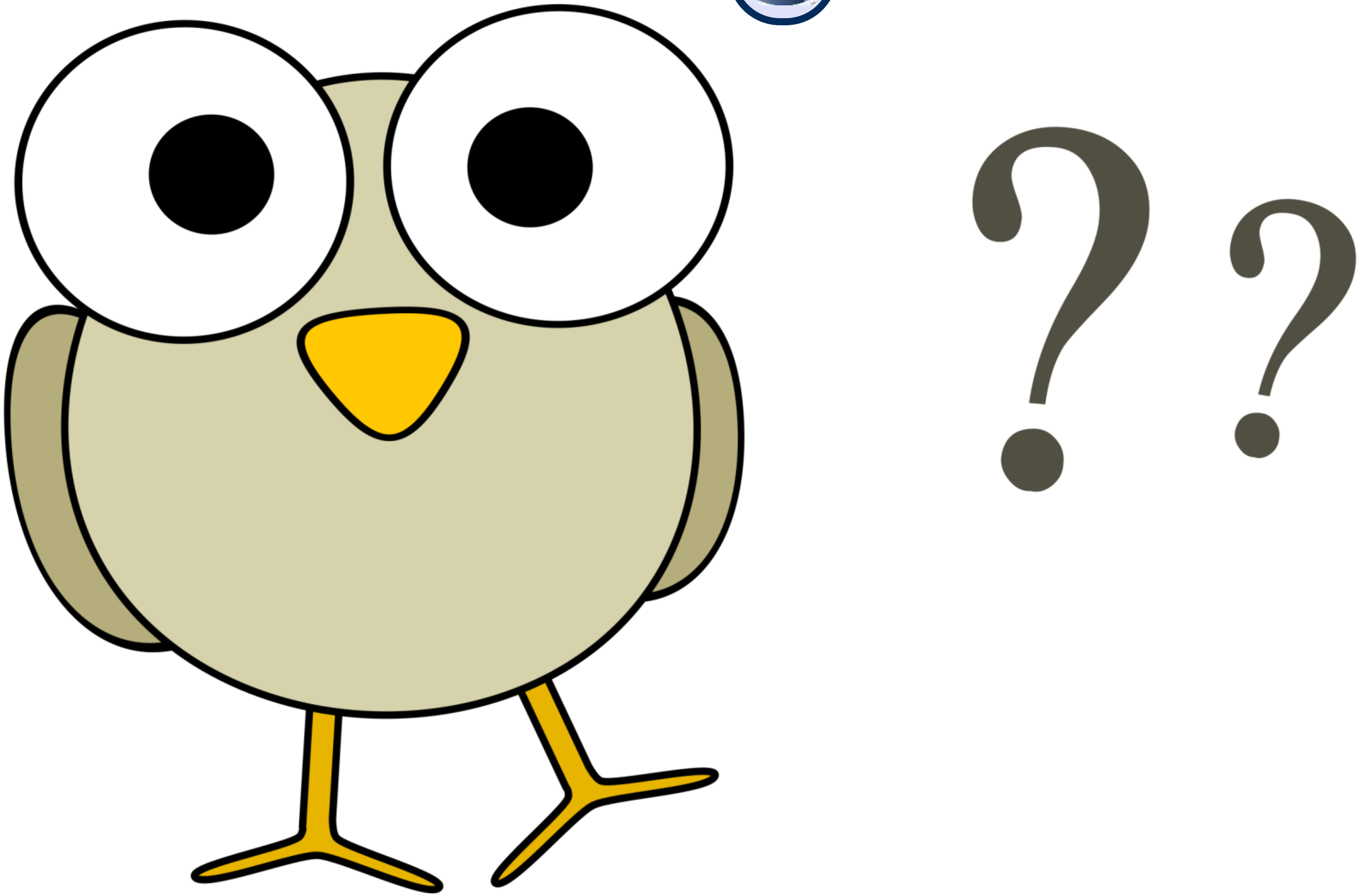
# Previously in CSCI 200

- 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.
- Explain the two ways to open a file for writing.
- Write a program that implements the corresponding pseudocode using file streams.

# On Tap For Today

- Streams
- Reading Files
- Writing Files
- Practice

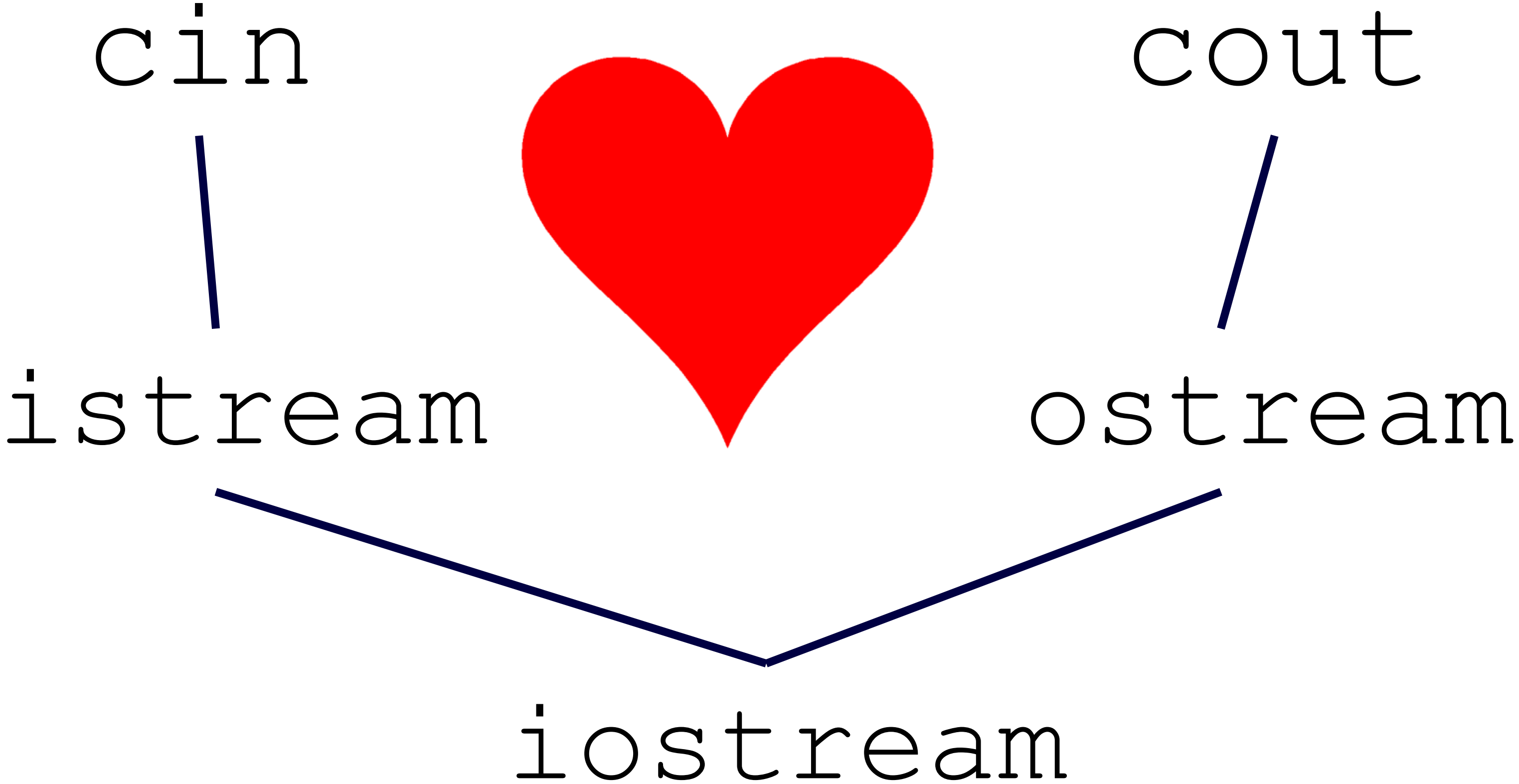
# On Tap For Today

- Streams
- Reading Files
- Writing Files
- Practice

# I/O You Know

`cin` ❤ `cout`

# I/O You Know



cin

cout

istream

ostream

iostream

# `#include <iostream>`

- `cin`
- `cout`

- `cin` and `cout` are “streams”

Streams come in

Streams go out

# What is a file?

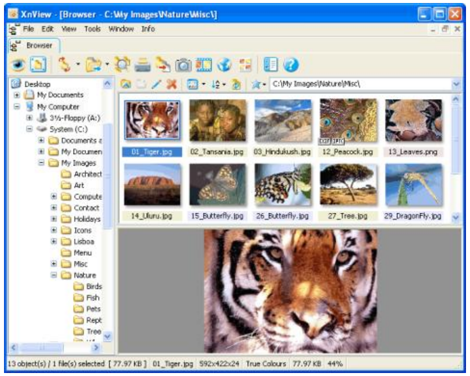
- A logical, coherent stream of bits on some persistent medium.

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
```



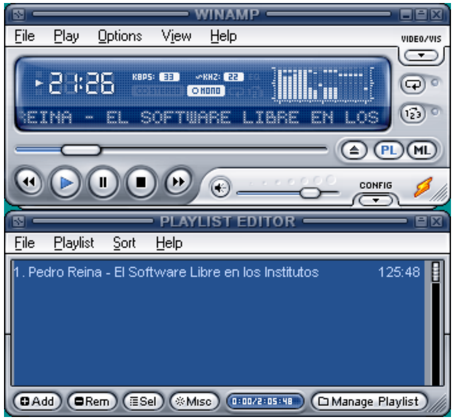
  - ...that files becomes a visible photo.

# 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
```



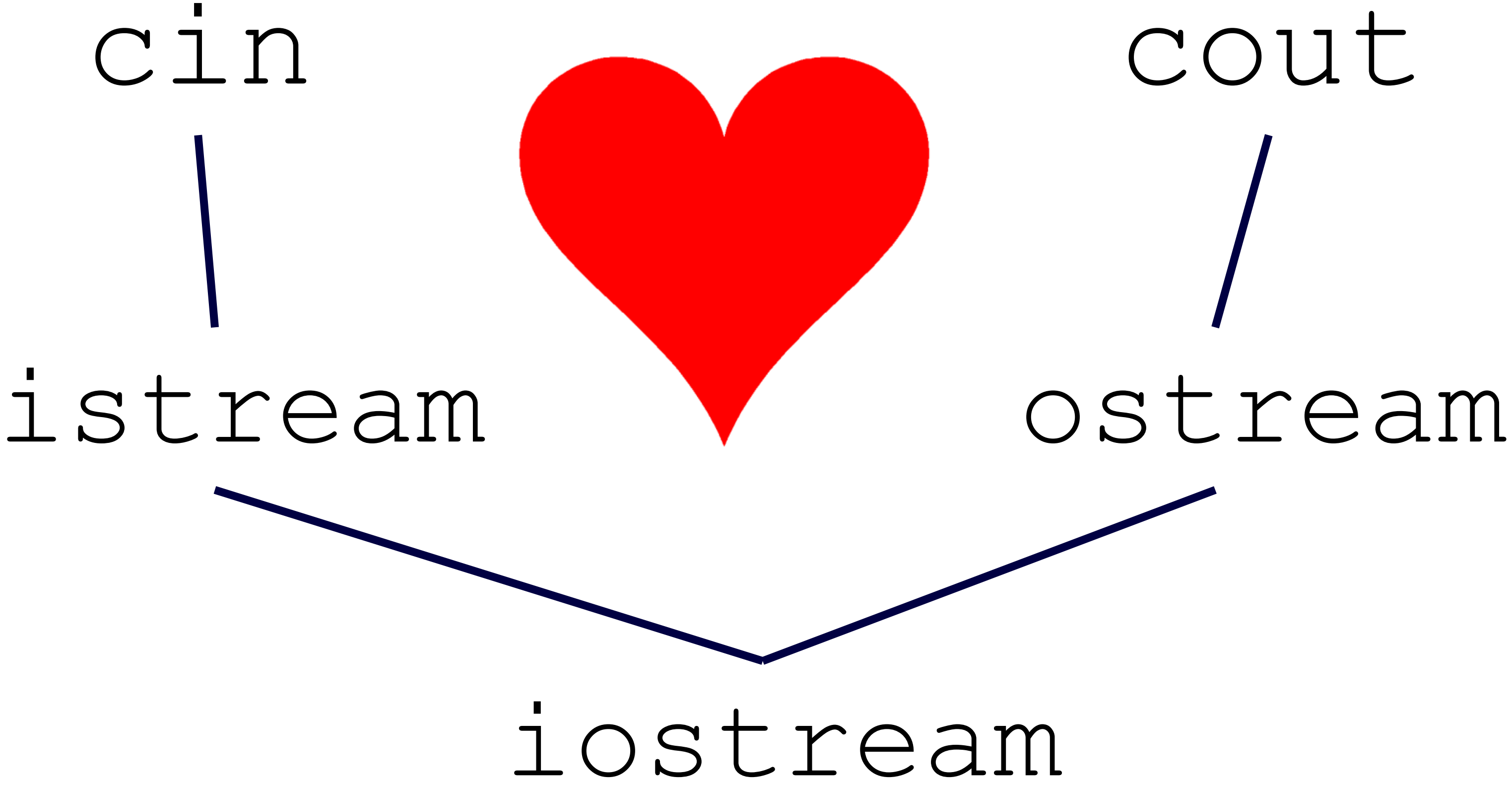
1. Pedro Reina - El Software Libre en los Institutos 125:48

  - …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



cin

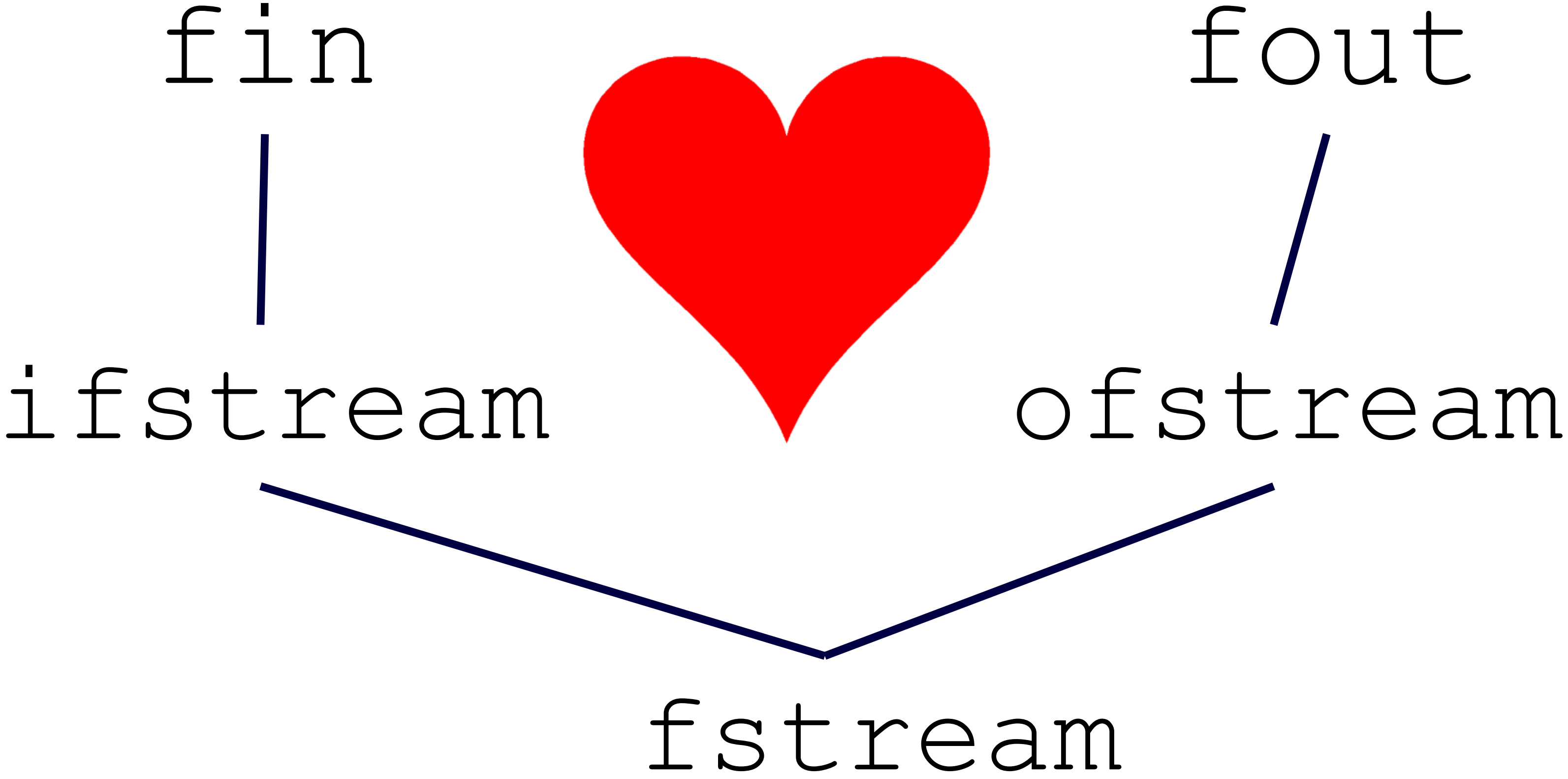
cout

istream

ostream

iostream

# I/O To Learn



fin

fout

ifstream

ofstream

fstream

# On Tap For Today

- Streams
- Reading Files
- Writing Files
- Practice

# 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

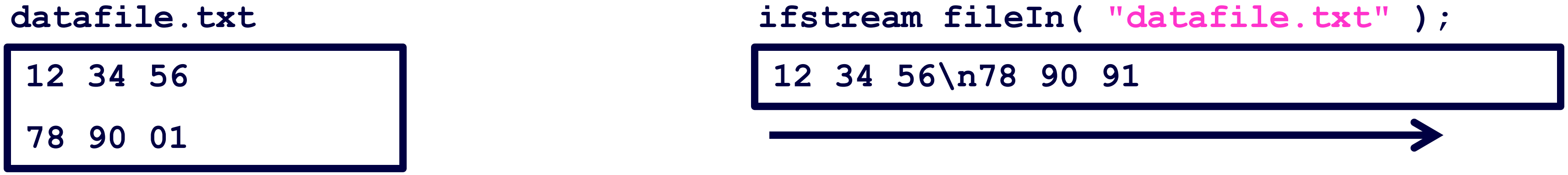
- Think of this data as “streaming” to your program

**datafile.txt**

```
12 34 56

78 90 01
```

# How >> and Streams Work

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



`datafile.txt`

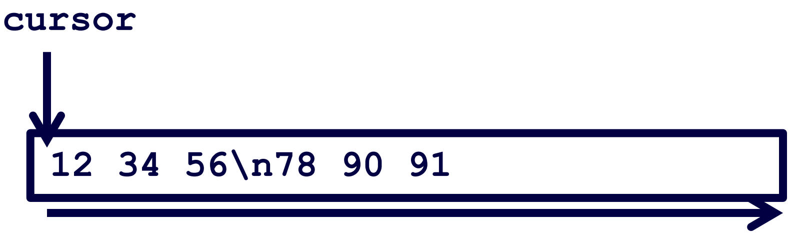
```
12 34 56

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”

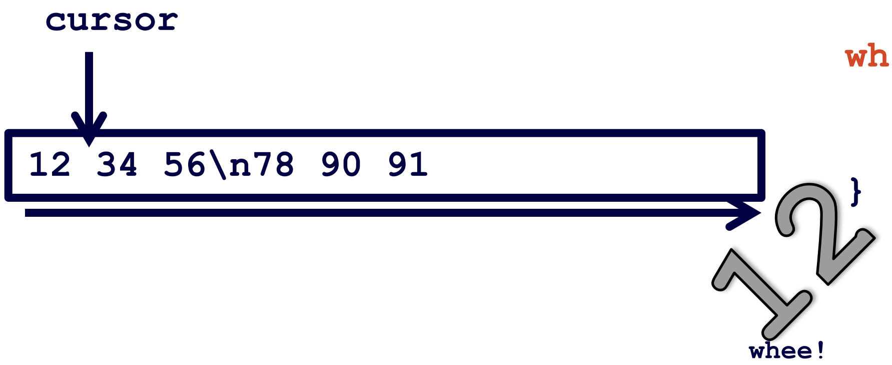


12 34 56\n78 90 91

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

# How >> and Streams Work

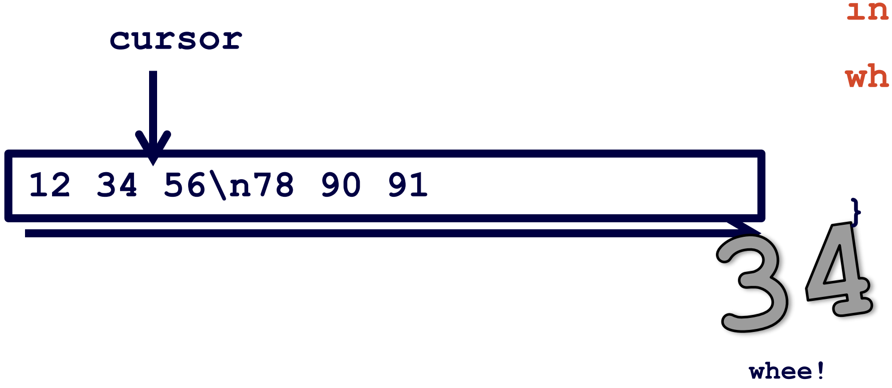
- “Computer, read the next value after the cursor up until a whitespace character”



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

# How >> and Streams Work

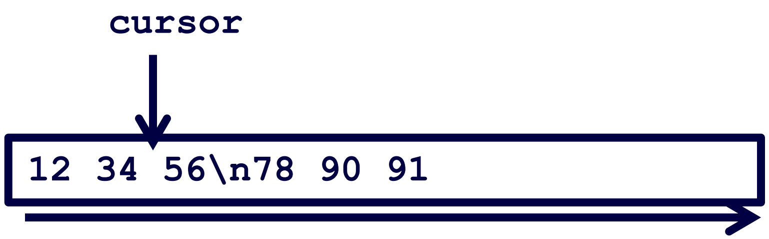
- “Computer, read the next value after the cursor up until a whitespace character”



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

# How get() and Streams Work

- “Computer, read ONE character after the cursor”



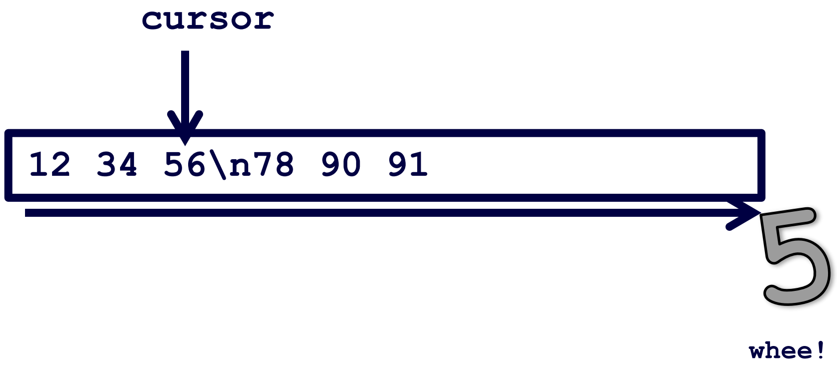
cursor

12 34 56\n78 90 91

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

# How get() and Streams Work

- “Computer, read ONE character after the cursor”



cursor

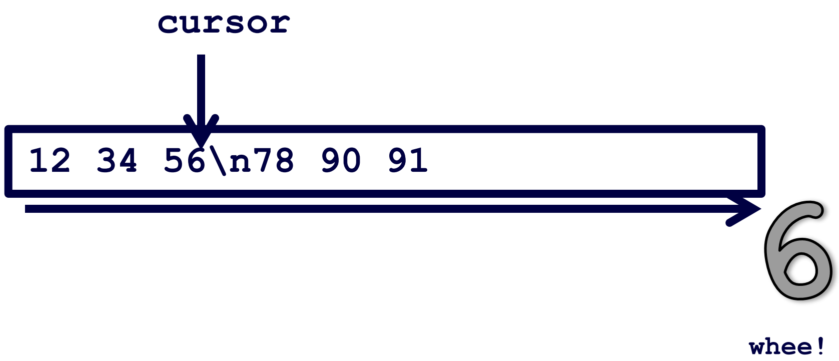
12 34 56\n78 90 91

whee!

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

# How get() and Streams Work

- “Computer, read ONE character after the cursor”



cursor

12 34 56\n78 90 91

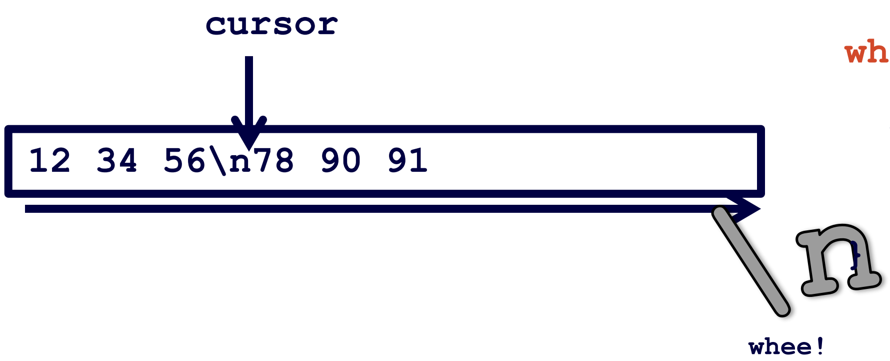
6

whee!

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

# How get() and Streams Work

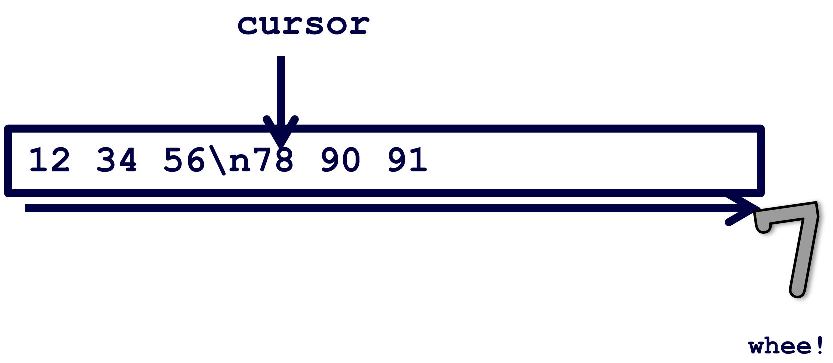
- “Computer, read ONE character after the cursor”



```
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 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
- Writing Files
- Practice

# Use it like `cout`

- “Computer, push this string to the screen”

```
cout << "this goes to the screen";
```

- “Computer, push this string to my file”

```
ofstream myOutput( "filename.ext" );
myOutput << "this goes to a file";
```

Insertion operator

# Terminology

- >> : extraction operator
  - Extracts data from a stream
  - Use with ifstream, cin
- << : insertion operator
  - Inserts data into a stream
  - Use with ofstream, cout
- Points direction stream is going

# Precedence Table

| Category | Precedence | Operator | Associativity |
|---|---|---|---|
| Parenthesis | 1 | ( ) | Innermost First |
| Postfix Unary Operators | 2 | a++ a-- f() a. | |
| Prefix Unary Operators | 3 | ++a --a +a -a !a ~a (type)a &a *p new delete | Right to Left |
| Binary Operators | 4 | a*b a/b a%b | Left to Right |
| | 5 | a+b a-b | |
| Shift Operators | 6 | a<<b a>>b | |
| Relational Operators | 7 | a<b a>b a<=b a>=b | |
| | 8 | a==b a!=b | |
| Bitwise Operators | 9 | a&b | |
| | 10 | a^b | |
| | 11 | a\|b | |
| Logical Operators | 12 | a&&b | |
| | 13 | a\|\|b | |
| Assignment Operators | 14 | a=b a+=b a-=b a*=b a/=b a%=b a&=b a^=b a\|=b | Right to Left |

# File Output Boilerplate

```
#include <fstream>
#include <iostream>
using namespace std;
int main() {
  ofstream fileOut( "FILENAME" );
  if( fileOut.fail() ) {
    cerr << "Error opening file to write" << endl;
    return -1;
  }
  fileOut << "write out all your data";
  fileOut.close();
  return 0;
}
```

# File Output Boilerplate

```
#include <fstream>
#include <iostream>
using namespace std;
int main() {
  ofstream fileOut( "FILENAME", ios::app );
  if( fileOut.fail() ) {
    cerr << "Error opening file to write" << endl;
    return -1;
  }
  fileOut << "write out all your data";
  fileOut.close();
  return 0;
}
```

# On Tap For Today

- Streams
- Reading Files
- Writing Files
- Practice

# To Do For Next Time

- Start L3A to decipher the moosage