CSCI 200: Foundational Programming Concepts & Design Lecture 15

Output Streams

Writing Data Using Files

Output Formatting

Complete Set2 Feedback

Access code: **imagine**

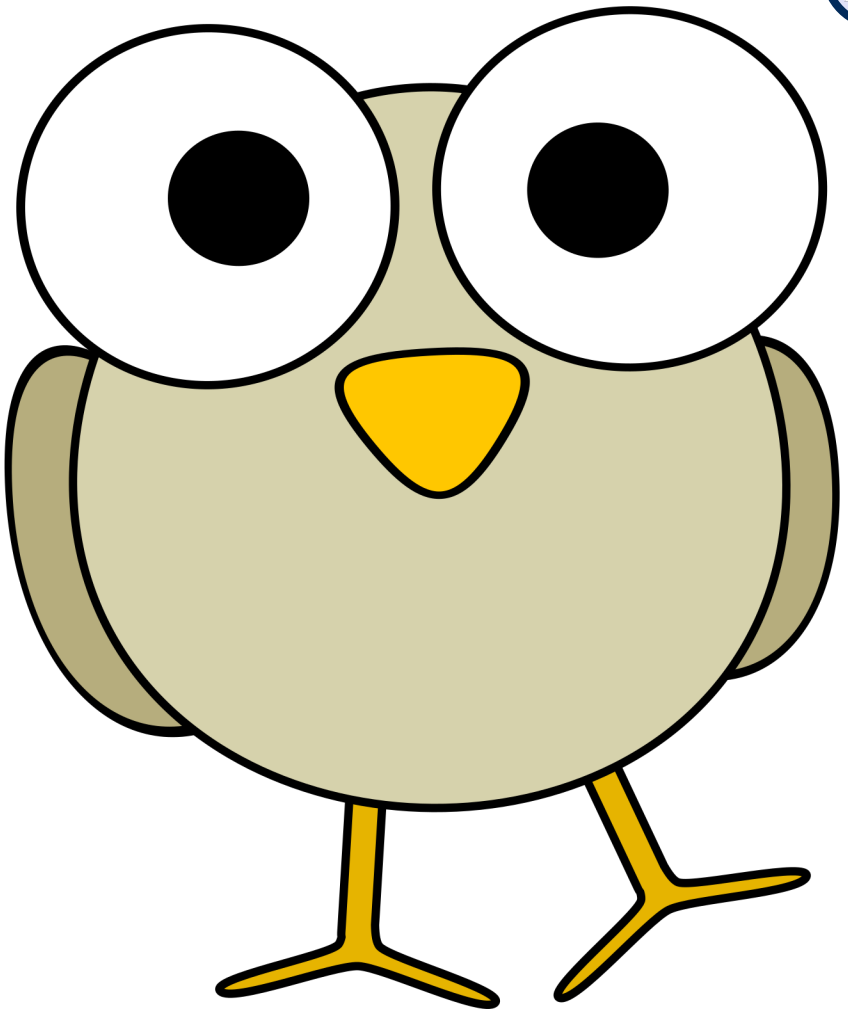
Previously in CSCI 200

- Reading to a file: 6 steps to read/write
    1. Include header
    2. Declare file stream
    3. Open file
    4. Check for error opening
    5. Read data
    6. Close file
- Functions associated with file streams:
  - open()
  - fail() / is_open()
  - close()

Previously in CSCI 200

- Input validation
  - Check value entered is of proper type
  - Check value entered is of proper range
- Do not assume a smart user
  - Cleanly handle invalid/bad input

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.
- Create a program with formatted output.

On Tap For Today

- Writing Files
  - Output Formatting
- Practice

On Tap For Today

- Writing Files
  - Output Formatting
- Practice

Writing Files

- "Computer, create an output filestream called myOutput that'll let me write (stream) content to filename.ext"

```
ofstream myOutput( "filename.ext" );
```

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

To WRITE to a File

- Include the fstream library
- Declare/open your output filestream
- Check for an error
- Write to the file
- Close the file

To WRITE to a File

- Include the fstream library

```
#include <fstream>
```

- Declare/open your output filestream

- Check for an error

- Write to the file

- Close the file

To WRITE to a File

- Include the fstream library

```
#include <fstream>
```

- Declare/open your output filestream

```
ofstream myOut ( "myData.txt" );
```

- Check for an error

- Write to the file

- Close the file

To WRITE to a File

- Include the fstream library

```
#include <fstream>
```

- Declare/open your output filestream

```
ofstream myOut;
```

- Check for an error

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

- Write to the file

- Close the file

To WRITE to a File

- Include the fstream library

```
#include <fstream>
```

- Declare/open your output filestream

```
ofstream myOut ( "myData.txt" );
```

- Check for an error

- Write to the file

- Close the file

To WRITE to a File

- Include the fstream library

```
#include <fstream>
```

- Declare/open your output filestream

```
ofstream myOut ( "myData.txt" );
```

- Check for an error

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

- Write to the file

- Close the file

To WRITE to a File

- Include the fstream library

```
#include <fstream>
```

- Declare/open your output filestream

```
ofstream myOut ( "myData.txt" );
```

- Check for an error

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

- Write to the file

- Close the file

To WRITE to a File

- Include the fstream library

```
#include <fstream>
```

- Declare/open your output filestream

```
ofstream myOut ( "myData.txt" );
```

- Check for an error

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

- Write to the file

- Close the file

To WRITE to a File

- Include the fstream library

```
#include <fstream>
```

- Declare/open your output filestream

```
ofstream myOut ( "myData.txt" );
```

- Check for an error

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

- Write to the file

- Close the file

To WRITE to a File

- Include the fstream library

```
#include <fstream>
```

- Declare/open your output filestream

```
ofstream myOut ( "myData.txt" );
```

- Check for an error

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

- Write to the file

```
myOut << "LOTS of data" << endl;
```

- Close the file

To WRITE to a File

- Include the fstream library

```
#include <fstream>
```

- Declare/open your output filestream

```
ofstream myOut ( "myData.txt" );
```

- Check for an error

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

- Write to the file

```
myOut << "LOTS of data" << endl;
```

- Close the file

```
myOut.close();
```

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

- Writing Files
  - Output Formatting
- Practice

Simple Formatting

- Special “escape characters” for output
  - '\n' → new line
  - '\t' → tab
  - '\\' → backslash aka whack
  - '\"' → quotation mark (whack double quote)
  - '\'' → apostrophe (whack single quote)
  - ' ' → space

'\n' or endl ?

- Both insert newline

- **endl** also flushes the stream

```
cout << "print now" << flush;
// ... later on ...
cout << " add to prior output" << endl;
```

Output Manipulators

- Include the iomanip library

```
#include <iomanip>
using namespace std;
```

- Manipulators modify output format
- Can modify output to any destination (standard out or file out)
  - Each destination has its own formatting
  - Need to specify formatting per destination

iomanip

- Examples

```
// floating point display
cout << fixed;                  // use decimal notation
cout << scientific;             // use scientific notation
cout << setprecision( 3 ); // 3 positions of precision

// alignment
cout << setw( 10 );             // set width of 10 to display
cout << left;                   // left align in column
cout << right;                  // right align in column
cout << setfill('-');           // fill allocated space with
```

On Tap For Today

- Writing Files
  - Output Formatting

- Practice

To Do For Next Time

- Start Set3 with L3A