CIS 190: C/C++ Programming

C++ Streams

Outline

- Quick Review of File Streams
- Note about Variables in C/C++
- Streams
  - More on Input Streams
  - I/O Manipulation
  - String Streams
- Errors in C++

Reading In Files in C++

- `ifstream inStream;`
  - declare an input file variable
- `inStream.open("testFile.txt");`
  - open a file for reading
- `if (!inStream) { /* exit */ }`
  - check to make sure file was opened

- read/write specified by variable type
  - `ifstream` for reading

Writing To Files in C++

- `ofstream outStream;`
  - declare an output file variable
- `inStream.open("testFile.txt");`
  - open a file for writing
- `if (!outStream) { /* exit */ }`
  - check to make sure file was opened

- read/write specified by variable type
  - **`ofstream`** for writing

Using Streams in C++

- must have **`#include <fstream>`**

- streams do not use placeholders (**`%d`**, **`%s`**, etc.)

- **`.open()`** takes a char* (a C-style string)
  - if using a C++ string variable, you must give it:

    `cppString.c_str()`

Outline

- Quick Review of File Streams
- Note about Variables in C/C++
- Streams
  - More on Input Streams
  - I/O Manipulation
  - String Streams
- Errors in C++

Interacting with Variables in C

- in C, many of the variables we use are "primitive" variables (int, char, double, etc.)

- when we want to interact with variables using the provided libraries, we call functions and pass in those variables
  - `fopen(ifp, "input.txt", "r");`
  - `free(intArray);`
  - `strlen(string1);`

Interacting with Variables in C++

- in C++, many of the variables we use are instances of a class (like string, ifstream, etc.)

- when we want to interact with these variables, we use method calls on those variables
  - `inStream.open("input.txt");`
  - `string2.size();`

Outline

- Quick Review of File Streams
- Note about Variables in C/C++
- Streams
  - More on Input Streams
  - I/O Manipulation
  - String Streams
- Errors in C++

Using input streams

- includes both **`istream`** and **`ifstream`**

- there are many ways to use input streams, with varying levels of precision/control
  - the **`>>`** operator
  - **`read()`**
  - **`ignore()`**
  - **`getline()`**

The >> Operator

- covered last class
  - returns a boolean for (un)successful read
  - just like scanf and fscanf:
    - skips leading whitespace
    - stops at the next whitespace (without reading it in)

```
cin >> firstName >> lastName >> age;
```

read()

- ```
  istream& read (char* s,
                 streamsize n);
  ```

- takes in a character array and a size

  ```
  inStream.read(charArr, C_ARR_SIZE);
  ```

- copies a block of data without checking its contents
  ```

ignore()

- ```
  istream& ignore (streamsize n = 1,
                   int delim    = EOF);
  ```

- extracts characters and discards them until either:
  - ***n*** characters (default: 1) are discarded
  - ***delim*** (default: EOF) is reached

- takes in a character array and a size

  ```
  inStream.ignore(IGNORE_NUM_CHARS);
  ```
  ```

getline()

- ```
  istream& getline (char* s,
                    streamsize n);
  ```

- takes in a character array and a size

  ```
  inStream.getline(charArr, C_ARR_SIZE);
  ```

- stops extracting characters upon hitting **'\n'**
  - also stops if it hits EOF
  ```

Outline

- Quick Review of File Streams
- Note about Variables in C/C++
- Streams
  - More on Input Streams
  - I/O Manipulation
  - String Streams
- Errors in C++

The <iomanip> Library

- used to format output in C++
- can be used on any output stream
  - cout
  - ofstream
  - ostringstream
- must have `#include <iomanip>`

setw()

- set the width of the next output
  - ONLY works for the next output
  - NOT “sticky”

```
cout << “Hello” << setw(10)
     << “world” << “.” << endl;
  Hello     world.
```

- will not cut off the output: input given is minimum amount of space to be taken up

setfill()

- change padding character
  - ' ' (or space) is default padding character

```
cout << setfill('-') << setw(8)
     << "hey" << endl;
```

```
-----hey
```

- padding character is set until changed again
  - IS "sticky"

setprecision()

- change maximum number of digits to display
  - numbers in total, not before or after decimal

```
cout << setprecision(5)
     << 3.1415926535 << endl;
```

```
3.1416
```

- attempts to round, not always perfect
  - whole numbers almost always “behave”

setprecision() example

- temp = 12.3456789 and test = 1234567.89

```
cout << temp << " and " << test << endl;
```

12.3457 and 1.23457e+06

set precision: 1

1e+01 and 1e+06

set precision: 2

12 and 1.2e+06

set precision: 3

12.3 and 1.23e+06

set precision: 9

12.3456789 and 1234567.89

set precision: 20

12.345678899999999345 and 1234567.8899999998976

Outline

- Quick Review of File Streams
- Note about Variables in C/C++
- Streams
  - More on Input Streams
  - I/O Manipulation
  - String Streams
- Errors in C++

String Streams

- allow us to use stream functions on strings
  - must have `#include <sstream>`
- helpful for formatting strings
- two types
  - **`ostringstream`**
  - **`istringstream`**

Uses for “I” and “O” String Streams

- common uses for **`istringstream`**
  - parsing a given string
- common uses for **`ostringstream`**
  - creating a new string with specific formatting
    - most often done using the iomanip library

The .str() Function

- function in the string stream libraries

- two different prototypes for str()

```
string str () const;
void   str (const string& s);
```

- first is “get” for string stream
- second is “set” for string stream

Using the .str() Function

- determines what you want by if arguments are passed in

- to convert from string stream to string

```
sString1.str();
```

- to clear a string stream

```
sString2.str("");
```

Outline

- Quick Review of File Streams
- Note about Variables in C/C++
- Streams
  - More on Input Streams
  - I/O Manipulation
  - String Streams
- Errors in C++

Errors in C++

- often MUCH longer than similar errors in C
  - even **more** important to start with the very first error, all the way at the top
- basic errors (typos, missing semicolons, etc.) remain largely the same
- the following slides contain some common compiler errors

Accidentally use << instead of >>

```
recover.cpp: In function 'int main()':
recover.cpp:30:10: error: no match for 'operator<<' in 'std::cin << fileName'
recover.cpp:30:10: note: candidates are:
In file included from /usr/include/c++/4.7/string:54:0,
                 from /usr/include/c++/4.7/bits/locale_classes.h:42,
                 from /usr/include/c++/4.7/bits/ios_base.h:43,
                 from /usr/include/c++/4.7/ios:43,
                 from /usr/include/c++/4.7/ostream:40,
                 from /usr/include/c++/4.7/iostream:40,
                 from recover.cpp:8:
/usr/include/c++/4.7/bits/basic_string.h:2750:5: note: template<class _CharT, class _Traits, class _Alloc> std::basic_ostream<_CharT, _Traits>& std::operator<<(std::basic_ostream<_CharT, _Traits>&, const std::basic_string<_CharT, _Traits, _Alloc>&)
/usr/include/c++/4.7/bits/basic_string.h:2750:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<_CharT, _Traits>'
In file included from /usr/include/c++/4.7/iostream:40:0,
                 from recover.cpp:8:
/usr/include/c++/4.7/ostream:469:5: note: template<class _CharT, class _Traits> std::basic_ostream<_CharT, _Traits>& std::operator<<(std::basic_ostream<_CharT, _Traits>&, _CharT)
/usr/include/c++/4.7/ostream:469:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<_CharT, _Traits>'
In file included from /usr/include/c++/4.7/iostream:40:0,
                 from recover.cpp:8:
/usr/include/c++/4.7/ostream:474:5: note: template<class _CharT, class _Traits> std::basic_ostream<_CharT, _Traits>& std::operator<<(std::basic_ostream<_CharT, _Traits>&, char)
/usr/include/c++/4.7/ostream:474:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<_CharT, _Traits>'
In file included from /usr/include/c++/4.7/iostream:40:0,
                 from recover.cpp:8:
/usr/include/c++/4.7/ostream:480:5: note: template<class _Traits> std::basic_ostream<char, _Traits>& std::operator<<(std::basic_ostream<char, _Traits>&, char)
/usr/include/c++/4.7/ostream:480:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<char, _Traits>'
In file included from /usr/include/c++/4.7/iostream:40:0,
                 from recover.cpp:8:
/usr/include/c++/4.7/ostream:486:5: note: template<class _Traits> std::basic_ostream<char, _Traits>& std::operator<<(std::basic_ostream<char, _Traits>&, signed char)
/usr/include/c++/4.7/ostream:486:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<char, _Traits>'
In file included from /usr/include/c++/4.7/iostream:40:0,
                 from recover.cpp:8:
/usr/include/c++/4.7/ostream:491:5: note: template<class _Traits> std::basic_ostream<char, _Traits>& std::operator<<(std::basic_ostream<char, _Traits>&, unsigned char)
/usr/include/c++/4.7/ostream:491:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<char, _Traits>'
In file included from /usr/include/c++/4.7/iostream:40:0,
                 from recover.cpp:8:
/usr/include/c++/4.7/ostream:511:5: note: template<class _CharT, class _Traits> std::basic_ostream<_CharT, _Traits>& std::operator<<(std::basic_ostream<_CharT, _Traits>&, const _CharT*)
/usr/include/c++/4.7/ostream:511:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<_CharT, _Traits>'
In file included from /usr/include/c++/4.7/ostream:607:0,
                 from /usr/include/c++/4.7/iostream:40,
                 from recover.cpp:8:
/usr/include/c++/4.7/bits/ostream.tcc:323:5: note: template<class _CharT, class _Traits> std::basic_ostream<_CharT, _Traits>& std::operator<<(std::basic_ostream<_CharT, _Traits>&, const char*)
/usr/include/c++/4.7/bits/ostream.tcc:323:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<_CharT, _Traits>'
In file included from /usr/include/c++/4.7/iostream:40:0,
                 from recover.cpp:8:
/usr/include/c++/4.7/ostream:528:5: note: template<class _Traits> std::basic_ostream<char, _Traits>& std::operator<<(std::basic_ostream<char, _Traits>&, const char*)
/usr/include/c++/4.7/ostream:528:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<char, _Traits>'
In file included from /usr/include/c++/4.7/iostream:40:0,
                 from recover.cpp:8:
/usr/include/c++/4.7/ostream:541:5: note: template<class _Traits> std::basic_ostream<char, _Traits>& std::operator<<(std::basic_ostream<char, _Traits>&, const signed char*)
/usr/include/c++/4.7/ostream:541:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<char, _Traits>'
In file included from /usr/include/c++/4.7/iostream:40:0,
                 from recover.cpp:8:
/usr/include/c++/4.7/ostream:546:5: note: template<class _Traits> std::basic_ostream<char, _Traits>& std::operator<<(std::basic_ostream<char, _Traits>&, const unsigned char*)
/usr/include/c++/4.7/ostream:546:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<char, _Traits>'
In file included from recover.cpp:9:0:
/usr/include/c++/4.7/iomanip:78:5: note: template<class _CharT, class _Traits> std::basic_ostream<_CharT, _Traits>& std::operator<<(std::basic_ostream<_CharT, _Traits>&, std::_Resetiosflags)
/usr/include/c++/4.7/iomanip:78:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<_CharT, _Traits>'
In file included from recover.cpp:9:0:
/usr/include/c++/4.7/iomanip:108:5: note: template<class _CharT, class _Traits> std::basic_ostream<_CharT, _Traits>& std::operator<<(std::basic_ostream<_CharT, _Traits>&, std::_Setiosflags)
/usr/include/c++/4.7/iomanip:108:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<_CharT, _Traits>'
In file included from recover.cpp:9:0:
/usr/include/c++/4.7/iomanip:142:5: note: template<class _CharT, class _Traits> std::basic_ostream<_CharT, _Traits>& std::operator<<(std::basic_ostream<_CharT, _Traits>&, std::_Setbase)
/usr/include/c++/4.7/iomanip:142:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<_CharT, _Traits>'
In file included from recover.cpp:9:0:
/usr/include/c++/4.7/iomanip:177:5: note: template<class _CharT, class _Traits> std::basic_ostream<_CharT, _Traits>& std::operator<<(std::basic_ostream<_CharT, _Traits>&, std::_Setfill<_CharT>)
/usr/include/c++/4.7/iomanip:177:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<_CharT, _Traits>'
In file included from recover.cpp:9:0:
/usr/include/c++/4.7/iomanip:207:5: note: template<class _CharT, class _Traits> std::basic_ostream<_CharT, _Traits>& std::operator<<(std::basic_ostream<_CharT, _Traits>&, std::_Setprecision)
/usr/include/c++/4.7/iomanip:207:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<_CharT, _Traits>'
In file included from recover.cpp:9:0:
/usr/include/c++/4.7/iomanip:237:5: note: template<class _CharT, class _Traits> std::basic_ostream<_CharT, _Traits>& std::operator<<(std::basic_ostream<_CharT, _Traits>&, std::_Setw)
/usr/include/c++/4.7/iomanip:237:5: note:   template argument deduction/substitution failed:
recover.cpp:30:10: note:   'std::istream {aka std::basic_istream<char>}' is not derived from 'std::basic_ostream<_CharT, _Traits>'
make: *** [recover] Error 1
```

Accidentally use << instead of >>

```
recover.cpp: In function 'int main()':
recover.cpp:30:10: error: no match for 'operator<<' in
'std::cin << fileName'
recover.cpp:30:10: note: candidates are:
In file included from /usr/include/c++/4.7/string:54:0,
                 from
/usr/include/c++/4.7/bits/locale_classes.h:42,
                 from /usr/include/c++/4.7/bits/ios_base.h:43,
                 from /usr/include/c++/4.7/ios:43,
                 from /usr/include/c++/4.7/ostream:40,
                 from /usr/include/c++/4.7/iostream:40,
                 from recover.cpp:8:
/usr/include/c++/4.7/bits/basic_string.h:2750:5: not
[…]
```

Accidentally use << instead of >>

recover.cpp: In function 'int main()':
recover.cpp:30:10: error: **no match for 'operator<<' in 'std::cin << fileName'**
recover.cpp:30:10: note: candidates are:
In file included from /usr/include/c++/4.7/string:54:0,
from /usr/include/c++/4.7/bits/locale_classes.h:42,
from /usr/include/c++/4.7/bits/ios_base.h:43,
from /usr/include/c++/4.7/ios:43,
from /usr/include/c++/4.7/ostream:40,
from /usr/include/c++/4.7/iostream:40,
from recover.cpp:8:
/usr/include/c++/4.7/bits/basic_string.h:2750:5: not
[…]

Forget `using namespace std;`

```
recover.cpp: In function 'int main()':
recover.cpp:22:3: error: 'string' was not declared in
this scope
recover.cpp:22:3: note: suggested alternative:
In file included from
/usr/include/c++/4.7/iosfwd:41:0,
                 from /usr/include/c++/4.7/ios:39,
                 from /usr/include/c++/4.7/ostream:40,
                 from /usr/include/c++/4.7/iostream:40,
                 from recover.cpp:8:
/usr/include/c++/4.7/bits/stringfwd.h:65:33:
note:   'std::string‘
[…]
```

Forget `using namespace std;`

recover.cpp: In function 'int main()':
recover.cpp:22:3: **error: 'string' was not declared in this scope**
recover.cpp:22:3: note: suggested alternative:
In file included from
/usr/include/c++/4.7/iosfwd:41:0,
from /usr/include/c++/4.7/ios:39,
from /usr/include/c++/4.7/ostream:40,
from /usr/include/c++/4.7/iostream:40,
from recover.cpp:8:
/usr/include/c++/4.7/bits/stringfwd.h:65:33:
note: 'std::string‘
[…]

Forget `#include <fstream>`

```
recover.cpp: In function 'int main()':
recover.cpp:23:12: error: aggregate
'std::ifstream inStream' has incomplete type
and cannot be defined
recover.cpp:24:12: error: aggregate
'std::ofstream jpegFile' has incomplete type
and cannot be defined
make: *** [recover] Error 1
```

Forget `#include <fstream>`

recover.cpp: In function 'int main()':
recover.cpp:23:12: error: aggregate 'std::ifstream inStream' has **incomplete type** and cannot be defined
recover.cpp:24:12: error: aggregate 'std::ofstream jpegFile' has incomplete type and cannot be defined
make: *** [recover] Error 1

Forget `#include <cstdlib>`

recover.cpp: In function 'int main()':
recover.cpp:37:12: error: 'exit' was not declared in this scope
recover.cpp:63:9: error: 'exit' was not declared in this scope
make: *** [recover] Error 1

Forget `#include <cstdlib>`

recover.cpp: In function 'int main()':
recover.cpp:37:12: error: **'exit' was not declared in this scope**
recover.cpp:63:9: error: 'exit' was not declared in this scope
make: *** [recover] Error 1

Homework 5

- Murder Mystery

- heavy on use of streams
  - not everything you need was covered in class
  - look at the cplusplus.com pages on streams!

- don’t worry too much about a “right” answer