CIS 190: C/C++ Programming

Lecture 7
C++ Streams

Outline

- Handling Streams in C++
  - Input Control
  - Output Control
  - String Streams
- Errors in C++
- Header Protection
- Homework

Using Input Streams

- input streams include

- **`istream`**
  - like reading in from the terminal
- **`ifstream`**
  - like reading in from a file
- **`istringstream`**
  - which we'll cover later today

Using Input Streams

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

Types of Whitespace

- many of the input streams delineate using whitespace
  - they'll skip leading whitespace
  - and stop at the next whitespace

- common types of whitespace:
  - space, tab, newline
  - carriage return (\r) – can cause problems
    - sometimes used in Windows and Mac files

The >> Operator

- returns a boolean for (un)successful read
- just like scanf and fscanf:
  - skips leading whitespace
  - stops at the next whitespace (without reading it in)
- appends a null terminator to strings read in

The >> Operator: Example

```
cout << “Please enter your first ”
     << “and  last name separated ”
     << “by a space: ”;
cin >> firstName >> lastName;

cout << “Please enter your age: ”
cin >> age;
```

ignore()

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

- takes in:
  - an integer (default value: 1)
  - a character delimiter (default value: EOF)

- both arguments are optional

ignore()

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

- ignore extracts characters and **discards them** until either:
  - `n` characters are extracted
  - `delim` is reached

ignore(): Example

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

```
iStream.ignore();
iStream.ignore(' ');
iStream.ignore(512);
iStream.ignore(512, ' ');
```

read()

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

- takes in:
  - a character array (a C string)
  - a size

- **`streamsize`** is a typedef of a signed integral type

read()

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

- copies a block of data of size **`n`** characters
  - stops after **`n`** characters, or at **`EOF`**
  - without checking its contents
  - **without** appending a NULL terminator
  - **without** moving through the input
    - often used in conjuction with ignore()

read(): Example

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

```
char strArr[SIZE];
inStream.read(strArr, SIZE-1);
/* do stuff with strArr */
// if you want to move on:
inStream.ignore(SIZE-1);
```

get()

- `istream& get (char &c);`

- takes in
  - a pointer to a character

- stores a single character
  - does not skip whitespace

```
cin.get(&character);
```

get()

- `int        get ();`

- returns a single character
  - the ASCII value of the character read in

```
character = cin.get();
```

Multiple Prototypes

- get() has two prototypes:

```
int      get ();
istream& get (char &c);
```

- this is called ***overloading***
- many library functions are overloaded
  - which function is called depends on the arguments

- you too can do this in C++ (we’ll cover it soon)

getline()

-
  ```
  istream& getline (char* s,
                    streamsize n);
  ```
- takes in:
  - a character array
  - a size
- extracts up to **`n`** characters
  - stops extracting characters upon hitting **`'\n'`**
  - also stops if it hits EOF

getline()

-
  ```
  istream& getline (char* s,
                    streamsize n);
  ```
- the newline is read in, and discarded
  - (not stored in the character array)
- carriage returns can cause problems, so be aware of the file's origin and format

getline(): Example

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

```
char chArray [128];
streamIn.getline(chArray, 128-1);
/* use "128-1" to leave room
   for the null terminator */
```

istream&?

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

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

-
  ```
  istream& get (char &c);
  ```

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

istream&?

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

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

-
  ```
  istream& get (char &c);
  ```

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

- all of these functions return a reference to an object of type **`istream`**

`istream&`

- **`istream`** is the class type that all other input stream types are derived from
  - like **`cin`** and input files

- the function is returning a reference to an object of type **`istream`**
  - references are *kind of* like pointers

- we'll cover this in more detail later

More Ways to Handle Input

- cplusplus.com/reference/istream/istream/
  - **`peek()`**
  - **`putback()`**
  - **`unget()`**
  - **`gcount()`**
  - **`tellg()`**

- can be very useful, but make sure you know exactly what it’s doing before you use it

Outline

- Handling Streams in C++
  - Input Control
  - Output Control
  - String Streams
- Errors in C++
- Header Protection
- Homework

Using Output Streams

- output streams include

- **`ostream`**
  - like printing out to the terminal
- **`ofstream`**
  - like writing to a file
- **`ostringstream`**
  - which we'll cover later today

The <iomanip> Library

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

IO Manipulation

- **`iomanip`** replaces the formatting we did inside the **`printf()`** statements:

```
printf("it'll %-6s for %07.3f hours\n",
       "rain", 3.14159);
> it'll rain   for 003.142 hours
```

- **`iomanip`** isn't as compact as **`printf()`**, but it's cleaner, and the code is clearer

The <iomanip> Library Functions

- **`setw()`**
  - used to set width of field
- **`setfill()`**
  - used to set a fill character (‘0’ or ‘ ’ or ‘_’, etc.)
- **`setprecision()`**
  - used to set decimal precision
- **`left`** and **`right`**
  - used to set alignment (not actually iomanip)

“Sticky”

- most of the ***parametric manipulators*** are “sticky” – once they are set, those manipulations apply to all future parameters unless changed by another call
  - setfill(), setprecision(), and left/right
- others only apply to the directly following output, and must be re-called each parameter
  - setw()

setw()

- set the width of 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 characters to be printed

setfill()

- change padding character
  - ‘ ’ (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
```

- precision holds for all future numbers
  - **IS "sticky"**

setprecision()

- not affected by calls to setfill()
- attempts to round, but it’s not always perfect
  - ints “behave” best, then doubles; floats are worst

- an example:

  temp = 12.3456789 and **test = 1234567.89**

```
cout << temp << " and " << test << endl;
12.3457 and 1.23457e+06
```

setprecision(): Example

```
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
```

setprecision(): Example

```
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
```

Alignment

- in `printf()`, we used a negative to left align, since right align was always used by default
  - when using `ostream`, right is still default
- instead we use keywords `left` and `right`
  - note that there are no parentheses (they are not functions)
  - **IS “sticky”**

Alignment: Example

```
cout << setw(8) << “hello” << endl;
cout << setw(8) << left << “cruel”
     << endl;
cout << setw(8) << right << “world”
     << endl;
```

```
   hello
cruel
   world
```

Livecoding iomanip Examples

- we’ll be using iomanip to:
  - left and right align
  - adjust width
  - change precision
  - set fill characters

LIVECODING

Outline

- Handling Streams in C++
  - Input Control
  - Output Control
  - String Streams
- Errors in C++
- Header Protection
- Homework

String Streams

- allow us to use stream functions on strings
  - must have `#include <sstream>`

- helpful for formatting strings

- two types
  - `ostringstream`
  - `istringstream`

Using String Streams

- **`istringstream`** is an input stream, so we can use any of the functions for input manipulation
  - **`read()`**, **`>>`**, **`ignore()`**, etc.

- **`ostringstream`** is an output stream, so we can use any of the iomanip tools
  - **`setw()`**, **`setfill()`**, **`left`**, etc.

Common Uses for String Streams

- use **`istringstream`** for
  - parsing a given string
- use **`ostringstream`** for
  - creating a new string with specific formatting

The str() Function

- two different prototypes for str()

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

- another overloaded function
  – which version the program calls is determined by the arguments you pass in

Two Forms of str()

```
string str () const;
```

– converts from a string stream to a string

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

– converts from a string to a string stream

Using First Form of str()

```
string str () const;
```

- returns a string containing a copy of the current contents of the stream
  - converts from a string stream to a string

```
newStr = oldStringStream.str();
```

Using Second Form of str()

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

- wipes contents of string stream, and sets to the contents of the passed-in string
  - – converts from a string to a string stream

```
newStringStream.str(oldStr);
newStringStream.str("hello");
```

Outline

- Handling Streams in C++
  - Input Control
  - Output Control
  - String Streams
- Errors in C++
- Header Protection
- Homework

Errors in C++

- are often MUCH longer than similar errors in C
- makes it 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

???

```
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
```

???

```
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
[…]
```

???

```
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
[…]
```

Used **<<** 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
[...]
```

???

```
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‘
[…]
```

???

```
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‘
[…]
```

???

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‘
[…]

Forgot `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‘
[…]

???

```
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
```

???

```
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
```

???

```
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
```

Forgot `#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

???

```
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
```

???

```
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

Outline

- Handling Streams in C++
  - Input Control
  - Output Control
  - String Streams
- Errors in C++
- Header Protection
- Homework

Headers in C++

- handled the same way as in C

- including user ".h" files:

```
#include "userFile.h"
```

- including C++ libraries

```
#include <iostream>
```

An Example

```
typedef struct bar{
  int a;
} BAR;
```

<u>bar.h</u>

```
#include "bar.h"

typedef struct foo{
  BAR x;
  char y;
} FOO;
```

<u>foo.h</u>

```
#include "bar.h"
#include "foo.h"

int main()
{
  BAR i;
  FOO j;

  /* ... */

  return 0;
}
```

<u>main.c</u>

An Example

```
typedef struct bar{
  int a;
} BAR;

        bar.h
```

```
#include "bar.h"

typedef struct foo{
  BAR x;
  char y;
} FOO;

        foo.h
```

```
#include "bar.h"
#include "foo.h"

int main()
{
  BAR i;
  FOO j;

  /* ... */

  return 0;
}

        main.c
```

when we try to compile this…

An Example

```c
typedef struct bar{
  int a;
} BAR;
```

bar.h

```c
#include "bar.h"
#include "foo.h"

int main()
{
  BAR i;
```

when we try to compile this…

```
In file included from foo.h:1:0,
                 from main.c:2:
bar.h:1:16: error: redefinition of 'struct bar'
In file included from main.c:1:0:
bar.h:1:16: note: originally defined here
In file included from foo.h:1:0,
                 from main.c:2:
bar.h:3:3: error: conflicting types for 'BAR'
In file included from main.c:1:0:
bar.h:3:3: note: previous declaration of 'BAR' was here
```

An Example

```c
typedef struct bar{
  int a;
} BAR;
```

bar.h

```c
#include "bar.h"
#include "foo.h"

int main()
{
  BAR i;
```

when we try to compile this...

```
In file included from foo.h:1:0,
                 from main.c:2:
bar.h:1:16: error: redefinition of 'struct bar'
In file included from main.c:1:0:
bar.h:1:16: note: originally defined here
In file included from foo.h:1:0,
                 from main.c:2:
bar.h:3:3: error: conflicting types for 'BAR'
In file included from main.c:1:0:
bar.h:3:3: note: previous declaration of 'BAR' was here
```

What the Compiler is “Seeing”

```
typedef struct bar{
  int a;
} BAR;

        bar.h
```

```
#include “bar.h”




typedef struct foo{
  BAR x;
  char y;
} FOO;

        foo.h
```

```
#include “bar.h”



#include “foo.h”








int main() {
  BAR i;
  FOO j;
  /* ... */
  return 0;
}
       main.c
```

What the Compiler is "Seeing"

```
typedef struct bar{
  int a;
} BAR;

        bar.h
```

```
typedef struct bar{
  int a;
} BAR;

typedef struct foo{
  BAR x;
  char y;
} FOO;

        foo.h
```

$\}$ #include "bar.h"

```
#include "bar.h"



#include "foo.h"









int main() {
  BAR i;
  FOO j;
  /* ... */
  return 0;
}
       main.c
```

What the Compiler is “Seeing”

```
typedef struct bar{
  int a;
} BAR;

        bar.h
```

```
typedef struct bar{
  int a;
} BAR;

typedef struct foo{
  BAR x;
  char y;
} FOO;

        foo.h
```

`#include “bar.h”`

```
typedef struct bar{
  int a;
} BAR;

#include “foo.h”







int main() {
  BAR i;
  FOO j;
  /* ... */
  return 0;
}
        main.c
```

`#include “bar.h”`

What the Compiler is "Seeing"

```
typedef struct bar{
  int a;
} BAR;

        bar.h
```

```
typedef struct bar{
  int a;
} BAR;

typedef struct foo{
  BAR x;
  char y;
} FOO;

        foo.h
```

#include "bar.h"

```
typedef struct bar{
  int a;
} BAR;

typedef struct bar{
  int a;
} BAR;

typedef struct foo{
  BAR x;
  char y;
} FOO;

int main() {
  BAR i;
  FOO j;
  /* ... */
  return 0;
}
            main.c
```

#include "bar.h"

#include "foo.h"

What the Compiler is "Seeing"

```
typedef struct bar{
  int a;
} BAR;

       bar.h
```

```
typedef struct bar{
  int a;
} BAR;

typedef struct foo{
  BAR x;
  char y;
} FOO;

       foo.h
```

#include "bar.h"

```
typedef struct bar{
  int a;
} BAR;

typedef struct bar{
  int a;
} BAR;

typedef struct foo{
  BAR x;
  char y;
} FOO;

int main() {
  BAR i;
  FOO j;
  /* ... */
  return 0;
}
          main.c
```

#include "bar.h"

#include "foo.h"

Header Protection

- for our program to work, we need to have the definition of the **`BAR`** struct in both:
  - **`foo.h`**
  - **`main.c`**

- the easiest way to solve this problem is through the use of **header guards**

Header Guards

- in each “.h” file, use the following:

Header Guards

- in each ".h" file, use the following:

```
#ifndef BAR_H    if not (previously) defined
```

Header Guards

- in each ".h" file, use the following:

```
#ifndef BAR_H    if not (previously) defined
#define BAR_H    then define
```

Header Guards

- in each ".h" file, use the following:

```
#ifndef BAR_H     if not (previously) defined
#define BAR_H     then define


[CONTENTS OF .H FILE GO HERE]
```

Header Guards

- in each “.h” file, use the following:

```
#ifndef BAR_H       if not (previously) defined
#define BAR_H       then define


 [CONTENTS OF .H FILE GO HERE]


#endif /* BAR_H */  stop the “if” at this
                    point (end of the file)
```

A Fixed Example

```c
typedef struct bar{
  int a;
} BAR;
```

bar.h

```c
#include “bar.h”

typedef struct foo{
  BAR x;
  char y;
} FOO;
```

foo.h

```c
#include “bar.h”
#include “foo.h”

int main()
{
  BAR i;
  FOO j;

  /* ... */

  return 0;
}
```

main.c

A Fixed Example

```c
#ifndef BAR_H
#define BAR_H

typedef struct bar{
  int a;
} BAR;

#endif /*BAR_H*/
```

bar.h

```c
#ifndef FOO_H
#define FOO_H

#include “bar.h”

typedef struct foo{
  BAR x;
  char y;
} FOO;

#endif /*FOO_H*/
```

foo.h

```c
#include “bar.h”
#include “foo.h”

int main()
{
  BAR i;
  FOO j;

  /* ... */

  return 0;
}
```

main.c

What the Compiler “Sees” – Fixed

```
typedef struct bar{
  int a;
} BAR;
```

bar.h

```
typedef struct bar{
  int a;
} BAR;

typedef struct foo{
  BAR x;
  char y;
} FOO;
```

#include “bar.h”

foo.h

```
typedef struct bar{
  int a;
} BAR;

typedef struct bar{
  int a;
} BAR;

typedef struct foo{
  BAR x;
  char y;
} FOO;

int main() {
  BAR i;
  FOO j;
  /* ... */
  return 0;
}
```

#include “bar.h”

#include “foo.h”

main.c

What the Compiler "Sees" – Fixed

```c
typedef struct bar{
  int a;
} BAR;
```

bar.h

```c
typedef struct bar{
  int a;
} BAR;

typedef struct foo{
  BAR x;
  char y;
} FOO;
```

foo.h

#include "bar.h"

```c
typedef struct bar{
  int a;
} BAR;

typedef struct bar{
  int a;
} BAR;

typedef struct foo{
  BAR x;
  char y;
} FOO;

int main() {
  BAR i;
  FOO j;
  /* ... */
  return 0;
}
```

main.c

#include "bar.h"

#include "foo.h"

What the Compiler “Sees” – Fixed

```c
typedef struct bar{
  int a;
} BAR;

        bar.h
```

```c
typedef struct bar{
  int a;
} BAR;

typedef struct foo{
  BAR x;
  char y;
} FOO;

        foo.h
```

#include “bar.h”

```c
typedef struct bar{
  int a;
} BAR;




typedef struct foo{
  BAR x;
  char y;
} FOO;

int main() {
  BAR i;
  FOO j;
  /* ... */
  return 0;
}
          main.c
```

#include “bar.h”

#include “foo.h”

Using Header Guards

- can prevent a lot of errors
- **still need to be mindful!!!**
- don’t just include every possible header and let header guards handle it for you

Outline

- Handling Streams in C++
  - Input Control
  - Output Control
  - String Streams
- Errors in C++
- Header Protection
- Homework

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!
- should be much easier (and shorter) than Homework 4B