CSE 143

Stream I/O Classes and Files

[A11-A15, A38-A50]

Standard Streams (review)

- Standard streams defined in **`iostream.h`** :
  - **`cin`**: standard *input* stream (usually keyboard)
  - **`cout`**: standard *output* stream (usually screen)
  - **`cerr`**: standard *error* stream (also usually directed to the screen)
- Operations
  - "get from": cin >> k >> j;
  - "put to": cout << "the answer is " << z << endl;

Formatted Output

- Various "manipulators" in <iomanip.h> can be included in the output stream to control the appearance of the next thing written.

```
cout<<fixed<<12.34<<' '<<scientific<<12.34;
```

produces

```
12.34 1.234e1
```

- Available operations include left/ right justify, set width, set number of digits of precision, …. See textbook for list.

Stream I/O for New Types

- Stream operations >>, << are defined in C++ for basic types
- New definitions of >>, << can be given for user-defined types (classes)

```
// write Complex number as <real,imag>
ostream& operator<<(ostream &s,Complex z){
   return s << '<' << z.re << ','
            << z.im << '>';
}
```

- This allows cout<<x<<y; How?

How cout<<x<<y works

- >> and << are binary left-associative operations.
- The meaning of

```
cout << x << y << z;
```

is

```
((cout << x) << y) << z;
```

- So after writing desired output, operator<< must return a reference to the stream so it can be used for further output.
- [CSE143] Not crucial to understand this in detail

Stream States

- All streams have a "state".
- All streams are objects (instances of stream classes)
- Several member functions are available to check or set state.

```
cin.eof();   // true if cin eof reached
cin.clear(); // set state to good
```

- The stream itself can be used in an expression to check its state

```
if (!cin)
  cerr << "error on cin" << endl;
```

Input Errors

- Stream input "fails" if the next thing in the input has the wrong format or if there is no more data (end of file).
- If an input operation fails, the variable involved is not changed.

```
if (cin >> k)
  cout << "new value for k read ok";
else
  cout << "input failed; "
       << "k not changed";
```


Input Errors (cont)

- Once a stream input operation has failed, any further operations will also fail until the stream state is reset.

```
// suppose next input is "xyz"
cin >> k;  // fails; k unchanged
cin >> j;  // cin state not good, so
           // nothing happens
cin.clear();  // cin can be used for
              // input again
```


Example: Copy Integers

- This program copies integers from cin to cout until an input operation fails. Each integer is written on a separate output line.

```
#include <iostream.h>
int main() {
  int j;
  while (cin >> j)
     cout << j << '\n';

  return 0;
}
```


Unformatted Stream I/O

- >> and << provide formatted I/O. Member functions provide unformatted (character-level) I/O.
- Examples:

```
char ch; char s[100];
cin.get(ch);  // read 1 character into ch
cin.getline(s); // read next line into s
cout.put(ch); // write 1 character ch
```

- Variations available to limit how many characters are read, specify end-of-line characters, etc.

File Concepts (review)

- Input (read) vs output (write)
- File name (on disk) vs file variable (in program)
- open
- close
- end of file

Stream Classes

- cin and cout are defined in <iostream.h>.
- Library <fstream.h> contains similar classes for file I/O
- Input stream classes:
  - istream: console input (cin)
  - ifstream: file input
- Output stream classes
  - ostream: console output (cout, cerr)
  - ofstream: file output

File I/O

- To open a file, give the disk file name as an argument when the file variable is created.

```
#include <fstream.h>
int main() {
  int k;
  ifstream here("source.txt");
  ofstream there("dest.txt");
  // copy one integer from here to there
  here >> k;
  there << k;
}
```

- Files are automatically closed when the program terminates.

File I/O Example

- Example: Copy contents of user-specified input file to user-specified output file.

```
#include <fstream.h>
int main() {
  char filename[300];
  cout << "input file? ";
  cin >> filename;
  ifstream source(filename);
  cout << "output file? ";
  cin >> filename;
  ofstream dest(filename);
  ...
```


File I/O Example (cont)

```
  ...
  char ch;
  while (source.get(ch))
     dest.put(ch);

  return 0;
}
```

- This version ignores various possible errors. What are they?

Stream Class Relationships

- Every ifstream (file) is also a istream.
  - An ifstream is an "enhanced" istream that has extra capabilities to work with disk files
  - An ifstream object can be used wherever an istream object is needed (function parameter, for example)
- But the reverse is not true. An istream is not also an ifstream.
  - So if an ifstream is explicitly called for, cin can't be used
- A similar relationship holds between ofstreams and ostreams.