File Input/Output

Kevin Schmidt, Susan Lindsey, Charlie Dey

File Streams

Create a file stream:

- abstraction representing a device to perform I/O operations
- represented as source or destination of characters of indefinite length
- Link file stream to the name of the file

- Input Stream:

  supplies data to the program
- Output Stream:

  receives data from the program

Stream Classes

File streams can be defined using the classes:

- `ofstream`: used for writing to files
- `ifstream`: used for reading from files
- `fstream`: both read from and write to files
    - all contained in header file fstream.h

- These classes derived from **iostream** (istream + ostream)
    - `cin`: object class of istream
    - `cout`: object class of ostream

Opening & Closing Files

Function `open()` can be used to open files

- Basic usage:

```
file-stream-class stream-object;
stream-object.open(“filename”);
```

Function `close()` can be used to close files

- Basic usage:

```
stream-object.close();
```

An Example (ex01.cc)

```
#include <iostream>
#include <fstream>
using std::ofstream;

int main(){
  ofstream myfile;
  myfile.open("output.txt");
// no mode specified, defaults to ios::in or ios::out
  myfile << "Writing this to a file.\n";
  myfile.close();
  return 0;
}
```

Modes of Files

Mode is an optional parameter that can be passed to the function `open()`

Some of the flags that can be used for mode:

`ios::in`	Open for input operations
`ios::out`	Open for output operations
`ios::app`	All output operations are performed at the end of the file

Mode Example (ex02.cc)

```
#include <iostream>
#include <fstream>
using std::ofstream; // class to read and write to files
using std::ios;

int main(){
  ofstream myfile;
  myfile.open("output.txt",ios::app); // opening in mode
ios::app
  myfile << "Also writing this to a file.\n";
  myfile.close();
  return 0;
}
```

Checking State Flags

`bad()` returns true if a reading or writing operation fails.

`fail()` returns true in the same cases as bad(), but also in the case that a format error happens

`eof()` returns true if a file open for reading has reached the end

`good()` is the most generic state flag: it returns true if none of the stream error state flags is set

State Flags Example (ex03.cc)

```
#include <iostream>
#include <fstream>
#include <string>
using std::ifstream;
using std::string;
using std::cout;
using std::endl;
int main() {
  string line;
  ifstream myfile("output.txt");
  if (myfile.is_open()){
    while (myfile.good()){
      getline (myfile,line);
      cout << line << endl;}
    myfile.close();}
  else cout << "Unable to open file";
  return 0;
}
```

Working with Multiple Files

If you want to process a set of files sequentially, you can use the same stream object multiple times

```
ofstream myfile;
myfile.open("data1.txt");
...
myfile.close();
...
myfile.open("data2.txt");
...
myfile.close();
```