COMSC-165 Lecture Topic 10 Text File I/O in C++

Reference

Deitel, chapter 8
cplusplus.com tutorial
Virginia Tech notes

File Libraries

`#include <fstream>`
`using std::ifstream;` // for text file input
`using std::ofstream;` // for text file output

`using std::ios;` under #include for iostream

File Formats and Modes

Text: variable-length records,
  EOLs, human-readable
Binary: fixed-size records, memory image
  machine-readable
modes: input (read a file), output (create a file),
  and append (add to an existing file)

Text File I/O Overview

file I/O objects (C++)
  `fstream` for binary file I/O
  `ifstream` for text file input
  `ofstream` for text file output
declare `ifstream fin;` // an object
`fin.open("e:/myfile.txt");`
declare `ofstream fout;` // an object
`fout.open("e:/myfile.txt");`
using TEXT file objects
  use `fin >>` and `fout <<`
possible file object errors
  file missing
  cannot create file
  `/ vs \\`

File Input Details

`ifstream fin; fin.open("e:/ifile.txt");`
  `cin;` -- always keyboard input
  interchangable with fin!
possible errors

Text File Output Details

`<fstream>`
  `ofstream fout; fout.open("e:/ofile.txt");`
  `cout;` -- always screen output
  interchangable!
possible errors
  file cannot be created
  disk full during writing
other issues
  new file or "overwrite"? `ios::noreplace`
  new file or "append"? `ios::app`

Binary File I/O

record I/O (i.e., structures)
  `fstream fin; fin.open("data.dat", ios::binary|ios::in);`
  `.read((char*)&noon, sizeof(tod));`
  `fstream fout; fout.open("data.dat", ios::binary|ios::out);`
  `.write((char*)&noon, sizeof(tod));`
Note: use an `fstream` object for binary I/O...
  ...NOT an ifstream or ofstream object
to get file size
  `fin.seekg(0, ios::end);`
  `long size = fin.tellg();`
  to rewind: `seekg(0, ios::beg)`
file headers
random access with `seekg()`

binary I/O is *easier* that text file I/O!
"read" and "write" have two simple parameters:
  memory location of what to read into or from
  how many bytes to read or write
  position pointer automatically managed

File I/O Error Handling

possible errors, e.g.
  file missing -- `if (!fin)`
  cannot create file -- `if (!fout)`
check for failure or success of command
  `if (!fin)` *or* `if (fin)`
  `if (!fout)` *or* `if (fout)`
  `while (fin)`
to reset after failure to open:
  `fin.clear();`

file not found (`!fin`)
read past end of file (`fin.eof()` or `!fin`)
sentinel value to mark end of file
string input w/**`fin.getline()`**
