CS102 – Review for exam over Chapter 8

Vocabulary to know

File Stream Primary storage (memory) Volatile Secondary storage Bit Byte	Input (Reading a file) Output (Writing to a file) File Types: binary and text files Opening a file Closing a file

Doing I/O with text files

INPUT	OUTPUT
Know how to declare and open an input stream ifstream data_file; data_file.open ("\\cs102\\programs\\lab8.dat"); Know how to read a line from a stream while (getline (data_file, input_line)) cout << input_line << endl; Know how to read using >> data_file >> a_line; Know how to read a character from a stream data_file.get (ch)	Know how to declare and open an output stream ofstream data_file; data_file.open ("\\cs102\\programs\\lab8.dat"); Know how to write to a stream out_file << data_to_write << endl; Know how to use I/O manipulators Setw, setfill, left, right, fixed, setprecision Know how to write a character to a stream data_file.put (ch)

String streams

Know how to open and use a string stream for input and the effects of doing this
istringstream strm;
strm.str("January 24, 1973");
string month;
int day;
string comma;
int year;
strm >> month >> day >> comma
>> year;
	Know how to open and use a string stream for output and the effects of doing this
	ostringstream strm;
	string month = "January";
	int day = 24;
	int year = 1973;
	strm << month << " " << day << ", "
	<< year;

Sequential and random access files

SEQUENTIAL ACCESS FILES	RANDOM ACCESS FILES (WITH UNIFORM LENGTH RECORDS)
Records can have varying lengths All records have to be accessed in order The file cannot be updated in place They are easy to program, but slow to process	Records have a fixed length--the same for all records You can access the records in any order

Random Access Files

Know how to declare and open a random access input stream

fstream random_file;

random_file.open (filename, status);

status should be one of

ios::in, ios::out, or ios::binary or several, separated by | (or)

1. With uniform length records

Reading, writing data

Usually, the data is read into/written from a struct

To read data into a struct

data_file.read ((char *) &a_record, sizeof (a_record));

Here, a_record is the struct variable

Positioning the file read pointer

Usually, you should position the pointer to the correct record

If you want to read record n, first use

data_file.seekg ((n-1)*sizeof(a_record));

Then read the record as above

2. Where the file has a header record which describes the structure of the file

Actually, this is very similar to the example above

You read the byte(s) that tell where a record is and how long it is

You position using seekg

You read the data using read

The command prompt

Be able to access and use arguments from the command prompt

int main (int argc, char* argv [ ])

Know what they are used for: to specify options and file names

The command prompt can also be used to redirect input and output

Know how to do it (< for input, > for output, and | to connect programs)

Know that cin and cout work with input/output redirection

Primary storage vs. secondary storage

Memory is another word for primary storage
Primary storage is volatile: data stored there is lost when power is lost
Everything happens in primary storage, because it is fast
When a program is running, it's in primary storage
Secondary storage
Some examples are: hard disk, DVD, flash drive
Secondary storage is permanent

Leftovers

It's good to use functions for file I/O

If you read a file name as a string, you need to convert it to a Cstring

```
in_file.open (file_name.c_str ());
```

Checking for errors

- After opening a file, you should always check for errors
  - An input file may not exist
  - The operating system may not let you create an output file