Advanced File and IO Operations

Chapter 13

Input and Output Streams

- C++ offers a set of Classes for interacting with files

Input and Output Streams

- Input Stream
  - A sequence from which data can be read
  - Used for file input

ifstream

Input and Output Streams

- Output Stream
  - A sequence to which data can be written
  - Used for file output

ofstream

Input and Output Streams

- Input-Output Stream
    - Handles file input and output

fstream

Input and Output Streams

- Functions
  - open(const char* filename)
  - open(const char* filename, ios::openmode mode
  - close()

Input and Output Streams

- Input Open modes
    - ios::binary
        - Data read or written will be in binary form
    - ios::in
        - The file will allow input operations

Input and Output Streams

- Output Open modes
  - ios::app
    - output will always take place at the end of the file
  - ios::ate
    - output will initially take place at the end of the file
  - ios::out
    - The file will allow output operations
  - ios::trunc
    - If the file contains anything, it will be discarded

Stream Manipulators

dec

endl

fixed

flush

hex

left

oct

right

scientific

setfill(char)

Stream Manipulators

setprecision(n)

setw(n)

showbase

noshowbase

showpoint

noshowpoint

showpos

noshowpos

Error Handling

- During normal file IO operations, problems can arise
    - They should always be checked when appropriate

Error Handling

- During normal file IO operations, problems can arise
    - They should always be checked when appropriate

- Streams have error bits that indicate when a specific problem happens

Error Handling

- Stream functions
  - eof() - True if the end of an input stream is encountered
  - fail() - True when an attempted operation has failed
    - either fail or hardfail (unrecoverable)
  - bad() - True an invalid operation was attempted
  - good() - True when the stream is in a good condition
  - clear() - Clears all errors

Error Handling

- How and when should we check for errors?

Stream Member Functions

- The stream class often uses the >> operator to move data from the file into the program

Stream Member Functions

- The stream class often uses the >> operator to move data from the file into the program
  - It moves a single item at a time

Stream Member Functions

- The stream class often uses the >> operator to move data from the file into the program
  - It moves a single item at a time

- What if we want to grab a complete line?

Stream Member Functions

- The stream class often uses the >> operator to move data from the file into the program
  - It moves a single item at a time

- What if we want to grab a complete line?
  - istream& getline(istream& is, string& str, char delim = ‘\n’);

Stream Member Functions

- The stream class also supports grabbing and placing a single character

Stream Member Functions

- The stream class also supports grabbing and placing a single character
  - Getting
    - int get();
    - istream& get(char& c);
  - Putting
    - ostream& put(int c);

Stream Member Functions

- Sometimes files need to be processed more than once
  - How could we accomplish this?

Stream Member Functions

- Sometimes files need to be processed more than once
  - How could we accomplish this?

- istream& seekg(streamoff off, ios_base::seekdir way);

Stream Member Functions

- Sometimes files need to be processed more than once
    - How could we accomplish this?

- istream& seekg(streamoff off, ios_base::seekdir way);
    - This function allows us to move anywhere in the file ie seekg(0, ios::beg);
    - seek ‘off’ bytes relative to ‘way’

Stream Member Functions

- tellg

Binary Files

- Up till now, we have seen files in clear text
    - Human readable

Binary Files

- Up till now, we have seen files in clear text
  - Human readable
  - ASCII characters

- Why might this be inefficient or undesirable?

Binary Files

- Up till now, we have seen files in clear text
    - Human readable
    - ASCII characters

- Why might this be inefficient or undesirable?
    - Space requirements
    - ...

Binary Files

- Welcome to binary file formats
  - Does not store data as ASCII, but Binary

Binary Files

- Welcome to binary file formats
    - Does not store data as ASCII, but Binary

- File containing the number 1337
    - ASCII
        - ‘1’ ‘3’ ‘3’ ‘7’
    - Binary
        - 00000101 00111001

Binary Files

- Opening a file in binary mode requires an extra flag

```
file.open("someFile.txt", ios::out | ios::binary);
```

Binary Files

- Writing and reading from files in binary mode are a bit different

Binary Files

- Writing and reading from files in binary mode are a bit different

```
ostream& write (const char* s, streamsize n);
istream& read (char* s, streamsize n);
```

Creating Records with Structures

- What is binary mode good for?

Creating Records with Structures

- What is binary mode good for?
  - Storing binary data

Creating Records with Structures

- A binary file allows the storing of binary data

Creating Records with Structures

- A binary file allows the storing of binary data
- What does that mean?

Creating Records with Structures

- A binary file allows the storing of binary data
- What does that mean?
  - variables
  - structs
  - ...

Creating Records with Structures

- A binary file allows the storing of binary data
- What does that mean?
  - variables
  - structs
  - ...
- This is often called serialization

Creating Records with Structures

- Storing structs in files is pretty straight forward

```
struct Coordinates{
        int x;
        int y;
};

Coordinates coords;
os.write(reinterpret_cast<char*>(&coords), sizeof(coords));
```

Creating Records with Structures

- Storing structs in files is pretty straight forward

```
struct Coordinates{
        int x;
        int y;
};

Coordinates coords;
os.write(reinterpret_cast<char*>(&coords), sizeof(coords));
```

How would we write a program that stored and read many structs to and from a file?

Random-Access Files

- Sometimes sequential access of a file just won’t do

Random-Access Files

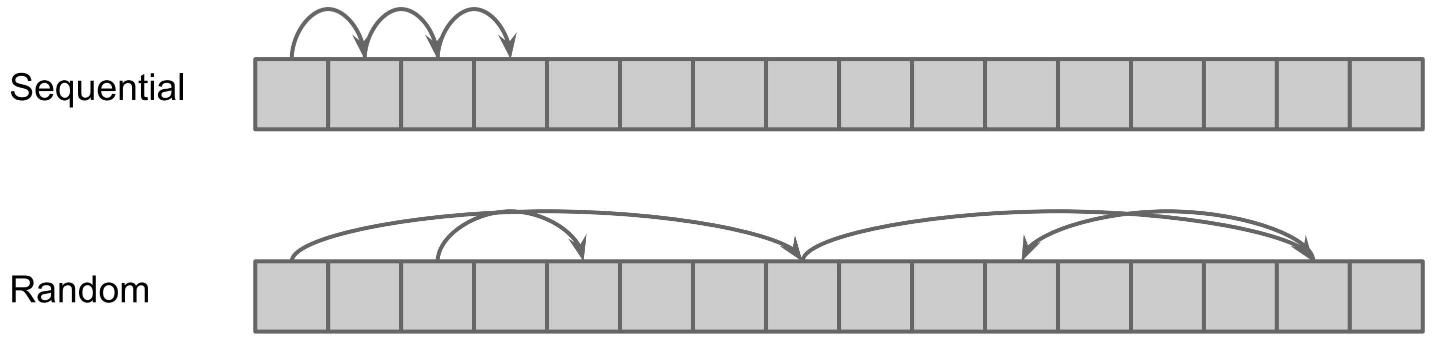
- Sometimes sequential access of a file just won’t do
    - Random-Access

Random-Access Files

- What are the differences between sequential and random access?

Random-Access Files

- What are the differences between sequential and random access?



Random-Access Files

- Random-Access is achieved through the seekp function

Random-Access Files

- Random-Access is achieved through the seekp function

- You probably remember seekg, but

Random-Access Files

- Random-Access is achieved through the seekp function

- You probably remember seekg, but
  - seekp is used for output files (think of ‘p’ for ‘put’)
  - seekg is used for input files (think of ‘g’ for ‘get’)

Random-Access Files

```
ostream& seekp(streampos pos);
ostream& seekp(streamoff off, ios_base::seekdir way);
```

off

    offset value relative to the way pointer

way

    ios_base::beg - Beginning of the stream
    ios_base::cur - Current position in the stream
    ios_base::end - End of the stream

Random-Access Files

```
file.seekp(43, ios::beg);
```

Sets the write position to the 44th byte (byte 43) from the beginning

```
file.seekp(-12, ios::end);
```

Sets the write position to the 13th byte (byte 12) from the end of the file

...

Random-Access Files

- Don’t forget about your current position in the file
    - tellp()

Random-Access Files

- Don’t forget about your current position in the file
    - tellp()
        - Used for output files (‘put’)
    - tellg()
        - Used for input files (‘get’)

Opening an Input and Output file

- Sometimes, it is useful to parse a file, modify something, then write it back to the file

Opening an Input and Output file

- Sometimes, it is useful to parse a file, modify something, then write it back to the file
    - This has the potential to be combersom if we have to open the file as input, load the item, close the file, open the file as output, move to the appropriate position, and write

Opening an Input and Output file

- This can be accomplished simpler
  - Open the file as input AND output

Opening an Input and Output file

- This can be accomplished simpler
  - Open the file as input AND output

```
fstream file(“db.dat”, ios::in | ios::out);
```

or

```
file.open(“db.dat”, ios::in | ios::out);
```

Opening an Input and Output file

- This can be accomplished simpler
  - Open the file as input AND output

```
fstream file(“db.dat”, ios::in | ios::out);
or
file.open(“db.dat”, ios::in | ios::out);
or for binary
file.open(“db.dat”, ios::in | ios::out | ios::binary);
```

Opening an Input and Output file

- This can be accomplished simpler
  - Open the file as input AND output

```
fstream file(“db.dat”, ios::in | ios::out);
```

or

```
file.open(“db.dat”, ios::in | ios::out);
```

or for binary

```
file.open(“db.dat”, ios::in | ios::out | ios::binary);
```

How can we write a progam that navigates to a specific coordinate in a file and modifies it?