National University of Computer & Emerging Sciences

Lecture 2
File Handling

Dr.Kifayat Alizai

Outline

- File Output Formatting
- Working with Multiple Files
- Opening a File both for Input and Output
- Error Testing
- Random Access File

File Output Formatting

- File output can be formatted the same way as screen output
  - `setw(x)`
    - Sets the *field width* where ‘x’ is number of characters to be used as field width
  - `showprecision(x)`
    - Sets the *decimal precision* to be used to format floating-point values on output operations.
- Requires `iomanip` to use manipulators

Example Code

- **set width.cpp**

- **set precision.cpp**

Working with Multiple Files

- Can have more than one files opened at a time in a program
- Files may be open for input or output
- Need to define file stream object for each file
- **Example: multiple files.cpp**

Opening a File both for Input and Output

- We can perform both the reading and writing operations on a single file.

- **Example:**

```
fstream obj("myData.txt", ios::in | ios::out);
```

Error Testing

- examine error state bits to determine stream status

Flag	Meaning
badbit	Some fatal (perhaps physical) error occurred. The stream should be considered unusable.
eofbit	End-of-input has occurred (either by encountering the physical end of a file stream or by the user terminating a console stream, such as with Ctrl-Z or Ctrl-D).
failbit	An I/O operation failed, most likely because of invalid data (e.g., letters were found when trying to read a number). The stream is still usable. The failbit flag is also set when end-of-input occurs.
goodbit	All is well; no errors. End-of-input has not yet occurred.

Member Functions / Flags

`eof()`	true if `eofbit` set, false otherwise
`fail()`	true if `failbit` or `hardfail` set, false otherwise
`bad()`	true if `badbit` set, false otherwise
`good()`	true if `goodbit` set, false otherwise
`clear()`	clear all flags (no arguments), or clear a specific flag

Example Program

```
void showState(fstream &file)
{
    cout << "File Status:\n";
    cout << "  eof bit: " << file.eof() << endl;
    cout << "  fail bit: " << file.fail() << endl;
    cout << "  bad bit: " << file.bad() << endl;
    cout << "  good bit: " << file.good() << endl;
    file.clear();  // Clear any bad bits
}
```

Random-Access Files

- Sequential access: start at beginning of file and go through data in file, in order, to end
  - to access 100th entry in file, go through 99 preceding entries first
- Random access: access data in a file in any order
  - can access 100th entry directly

File Pointers

- Each file object has associated with it two integer values called the ***get pointer*** and the ***put pointer***
  - These are not the normal pointers, used so far in our course
  - These are also called the ***current get position*** and the ***current put position***, or simply the ***current position***.
  - These values specify the **byte** number in the file where writing or reading will take place.

File Pointers

- The **seekg()** and **tellg()** functions allow to set and examine the get pointer

- The **seekp()** and **tellp()** functions perform the same actions on the put pointer.

File Pointers - Get Pointers

- **seekg()**
  - Navigates the read pointer to desired position or byte in the file
- **tellg()**
  - Returns the position of the current character in the input stream.

File Pointers - Put Pointers

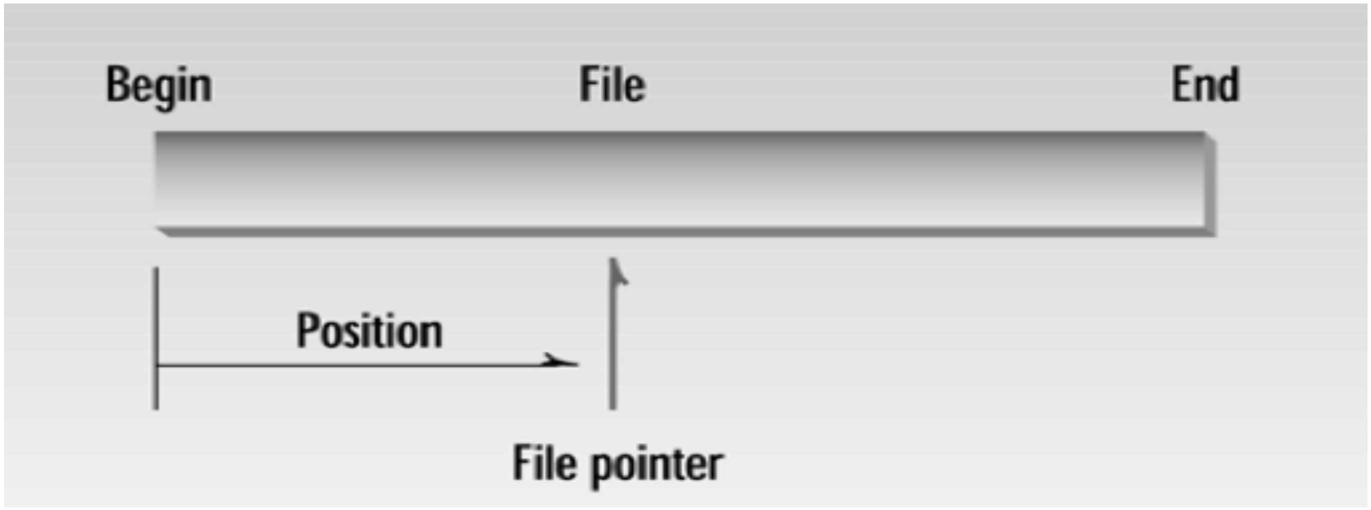
- **seekp()**
  - Navigates the write pointer to desired position or byte in the file
- **tellp()**
  - Returns the position of the current character in the output stream.

Specifying the Position

The rules on below slides are same both for seekg() and seekp()

Specifying the Position

- The seekg() function works in two ways.
  - single argument represents the position from the start of the file
- Example: seekg(0)



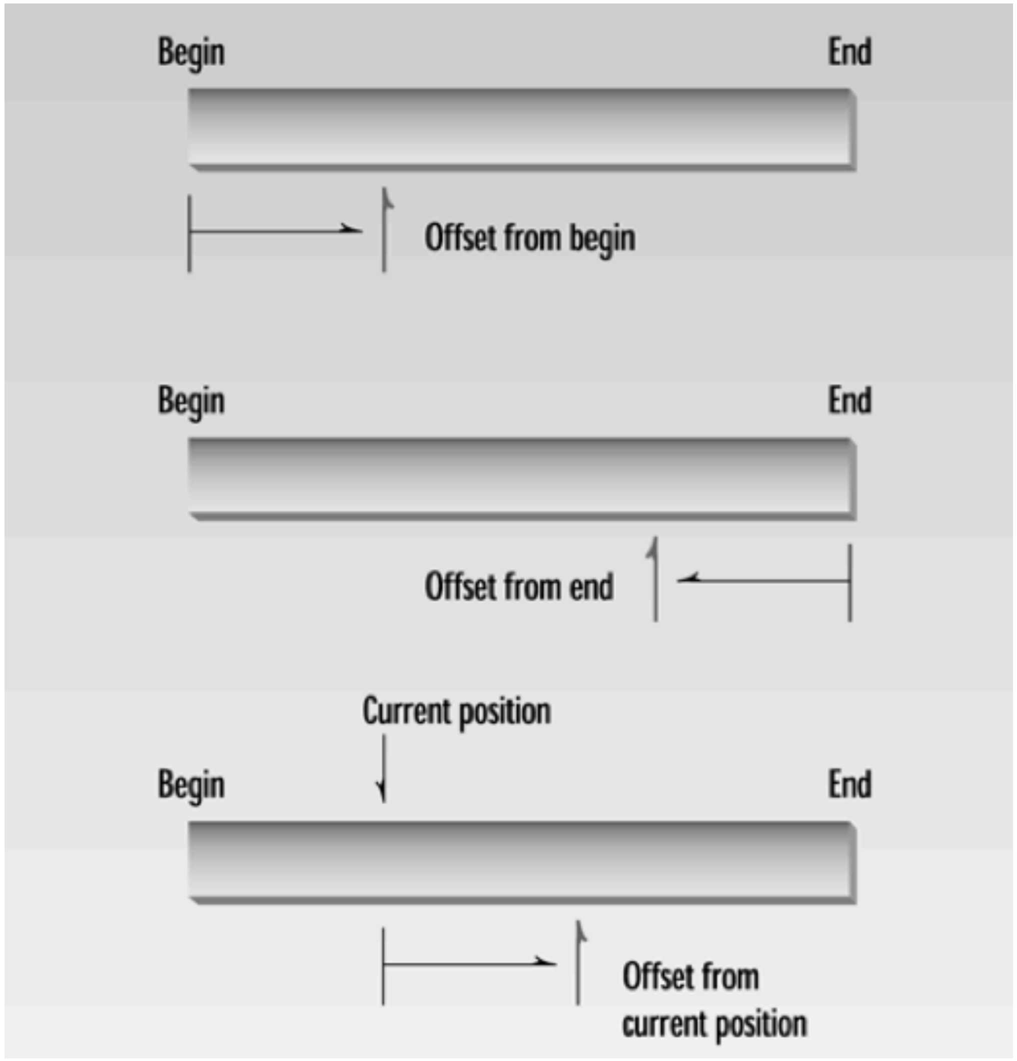
Specifying the Position

- The seekg() function works in two ways.
  - single argument represents the position from the start of the file
  - **Example: seekg(0)**
- seekg() with two arguments
  - the first argument sets an offset from a particular location in the file
  - the second specifies the location from which the offset is measured.
  - **Example: seekp(10L, ios::beg)**

Specifying the Position

- There are three possibilities for the second argument:
  - **beg** is the beginning of the file,
  - **cur** is the current pointer position,
  - **end** is the end of the file.
- Example: seekp(-10, ios::end)
  - sets the put pointer to 10 bytes from the end of the file.

Specifying the Position



Summary of seekp() and seekg()

- `seekg,seekp` arguments:
  offset: number of bytes, as a `long`
  mode flag: starting point to compute offset

- Examples:

```
inData.seekg(25L, ios::beg);
// set read position at 26th byte
// from beginning of file

outData.seekp(-10L, ios::cur);
// set write position 10 bytes
// before current position
```

Example code

- **Purpose**: To demonstrate working of seekg and tellg
- **Code:** randomAccess - seekg tellg.cpp

Example code

- **Purpose**: To demonstrate working of seekp and tellp
- **Code:** randomAccess - seekp tellp.cpp

EOF in Random Access

- If `eof` is true, it must be **cleared** before performing `seekg` or `seekp`
- Example:

```
fileObj.clear();
fileObj.seekg(0L, ios::beg);
```

- clear() function clears the stream i.e. makes eof as false
  - When eof is false, the pointer can be navigated

Example code

- **Purpose**: To demonstrate working of eof marker
- **Code:** randomAccess - eof marker.cpp

Execution Time Comparison in Data Access - Sequential vs Random

- Time to reach EOF

  Code:

  speedTest - random access file - eof.cpp

  speedTest - seq access file - eof.cpp

- Time to search for specific words

  Code:

  speedTest - random access file - searchWords.cpp

  speedTest - seq access file - searchWords.cpp