20 - Files and Streams

December 6, 2022

COMP2404

Darryl Hill

Files and Streams

Contents

Streams Review

Stream:

- A sequence of bytes
- Data going from **source** to **sink**
    - Perhaps buffered
    - Source to program
    - Program to source
- Data source and sink examples:
    - Keyboard, console
    - Files
    - Printers
    - Network adapters

Streams

`iostream` library has generic I/O template specializations.

- `istream`
  - important object - `cin`
- `ostream`
  - important objects - `cout, cerr, clog`

Streams

Characteristics of Streams:

- Maintain error bits
  - `good, bad, fail` bits
- Provide member functions to test the error bits.

Streams

- Overloaded ! operator:
  - Returns `true` if one of the error bits is true.
    - Lets us loop by testing the stream.
    - Exit if ! returns true.
- Cast to `void*` operator
  - If we test the stream invoked implicitly
  - Converts stream to a pointer
    - Null if one of the error bits is true
    - Non-null otherwise

That is to say, we can test the stream itself: `if (cin)` for example.

coding example <p1>

Coding Example p1 Notes

```
string s1, s2; int num;
```

```
while (cin>>s1>>s2>>num)
```

This will loop forever if we continue to enter things correctly.

```
while !cin.eof()
```

- end of file operator, which is ctl-d

`cin.get()` reads a single character.

```
char str[MAX_BUF];
```

`cin.getLine(str, MAX_BUF)` to end on a newline or
`cin.getLine(str, MAX_BUF,'*')` to end when * is entered

Streams

Two types of input for input streams:

- Formatted data:
  - We use the `>>` operator
  - We know (or expect) what type of data we are receiving
- Unformatted data:
  - We use `get()` or `getline()` to retrieve the characters.
  - This will be in `char` format.

End-of-file marker:

- Value is OS independent.
- Tested with `eof()`

Files

A stream kept in persistent storage

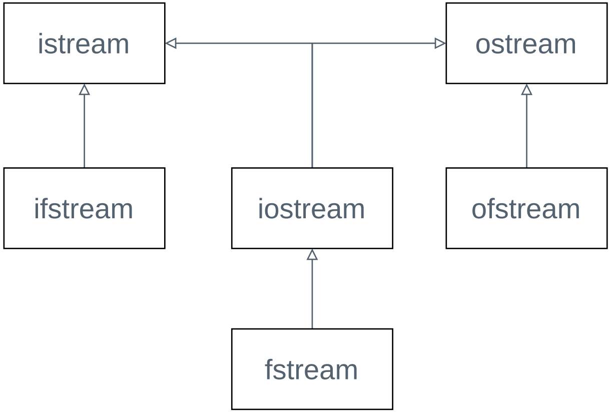
- also called ***non-volatile storage***

Files are

- an array of bytes terminated by an eof marker
- C++ represents files as objects

Files

- `iostream` library has file I/O template specializations



Files

- `ifstream` derived from `istream`
  - Represents input files.
- `ofstream` derived from `ostream`
  - Represents output files.
- Both maintain a file buffer object
  - File buffer destructor closes the file.
- Can use ! and cast to `void*` operators
  - Since they are streams.

Files as Objects

Useful member functions

- constructor
  - can optionally open the file
  - second argument is the mode
  - input files may reposition the pointer
- file management - `open()`, `close()`

coding example <p2> - stream insertion operator we write for objects works for files as well

Files as Objects

Useful `ofstream` member functions

- `<<`
- `put()`
- `flush`

Error State Flags

Stream object contain flags (bits)

- `good` bit - no errors
  - member function - `good()`
- `fail` bit - formatting error
  - member function - `fail()`
- `bad` bit - unrecoverable error
  - member function - `bad()`

`iostream` objects also have

- `eof` bit - end of file
  - member function - `eof()`

`clear()` - resets flags, good to 1, all others to 0

The end