CSE 111 HW 3 – FileAnalyzer

Contents

1 Submission

- Due: 1/24/2025 11:59PM.
- Weight: 3× a normal homework assignment.
- Submit all your files to GradeScope. (`"CMakeLists.txt"` plus any source files referenced in your `"CMakeLists.txt"`).
- You have unlimited submission attempts to the autograder.
- The score of the autograder will be the score you get on the assignment. (Excluding academic dishonesty exceptions).

2 Goals

This homework serves as practice for C++ STL, Classes, and other syntax.

You will be finishing the implementation for the following options for the application `FileAnalyzer`:

- `--get-size` *Implemented for you*
- `--code-get-lines`
- `--code-check-parens`
- `--csv-get-rows`
- `--csv-get-columns`
- `--csv-verify-dimensions`
- `--png-get-width` *Implemented for you*
- `--png-get-height` *Implemented for you*
- `--text-get-letter-count`
- `--text-get-letter-count-sorted`
- `--text-get-most-common`
- `--text-get-least-common`
- `--wav-get-bitrate`
- `--wav-get-channels`

See the contents of `"tests/"` for example outputs.

3 AI Usage

When using AI LLMs, (such as ChatGPT, Claude, CoPilot, Bard, etc.), ensure careful review of output before submission.

Include citations for all AI-derived content, specifying the model used and providing a link to the conversation history as a comment. An example of an acceptable link is shown here: https://chat.openai.com/share/b80473dc-a9d9-4435-97a2-e13e41a59c38. This link was obtained from ChatGPT, by clicking the share icon in the top-right corner of the interaction.

4 Files

The following source files are given to you:

- `"CMakeLists.txt"`
- `"src/main.cpp"`
- `"src/FileAnalyzer.h"`
- `"src/FileAnalyzerFile.h"`
- `"src/FileAnalyzerFile.cpp"`
- `"src/FileAnalyzerPng.h"`
- `"src/FileAnalyzerPng.cpp"`

A `"tests/"` directory is also given to you. See the README for usage. You are welcome to add more tests.

The rest of the source files you will have to implement. It's recommended that you put your class implementations in separate files. (For example `"FileAnalyzerWav.cpp"` and `"FileAnalyzerWav.h"`). Be sure to add your created source files to `"CMakeLists.txt"`. You can use the implementation of `FileAnalyzerPng` as a reference.

5 Output

An example output for a .wav file ex: `"tests/PinkPanther30.wav/PinkPanther30.wav"` might be

```
$ ./build/FileAnalyzer tests/PinkPanther30.wav/PinkPanther30.wav --get-size
1323044
$ ./build/FileAnalyzer tests/PinkPanther30.wav/PinkPanther30.wav --wav-get-bitrate
352800
$ ./build/FileAnalyzer tests/PinkPanther30.wav/PinkPanther30.wav --wav-get-channels
1
```

6 Classes

Each `FileAnalyzer(filetype)` class should inherit from `FileAnalyzerFile`. Be sure to add a constructor that takes in `const std::string &filename`, and initializes `FileAnalyzerFile`. The constructor should print a warning if the file extension in the filename is not part of the list of supported extensions.

Note the use of `protected` in the `FileAnalyzerFile` class. Protected means that the affected member is available to children, but is private otherwise. This means that each of the sub-classes may use `contents_`.

7 Code Source Files

Supported file types: `.c`, `.cpp`, and `.h`.

7.1 Options

- `--code-get-lines`: output the number of non-empty lines in the provided file. (Newlines are denoted with `'\n'`).
- `--code-check-parens`: outputs a boolean on whether a file has balanced parentheses: `()[]{}`. There is no need to support quotes or comments.

7.2 Tips

- You can use `std::getline()` and `isspace()` to tell if a line has text.
- You can use `std::stack` to monitor whether a parenthesis has a match.

8 CSV Files

CSV (Comma-Separated Values) is a format for representing tabular data. Column values are separated with commas (`','`). Rows are separated using newlines (`'\n'`).

Supported file type: `.csv`.

8.1 Options

- `--csv-get-rows`: output the number of rows
- `--csv-get-columns`: output the number of columns in the first row
- `--csv-verify-dimensions`: output true if every row in the csv has the same number of columns as the header, false otherwise.

8.2 Tips

- Assume that no CSV file cells have any `'\n'` or `','` characters in them.
- Assume that all CSV files have a final newline.

9 PNG Files

Nothing needs to be done for PNG files. The PNG options are implemented for you.

For more info, see the PNG spec: http://www.libpng.org/pub/png/spec/1.2/png-1.2.pdf.

Supported file type: `.png`.

9.1 Options

- `--png-get-width`: output the height of the image
- `--png-get-height`: output the width of the image

9.2 Tips

- `FileAnalyzerFile::ReadBigEndianInt32()` is used to read the 4 byte big-endian values for width and height.

10 Text Files

Supported file type: `.txt`.

10.1 Options

- `--text-get-letter-count`: outputs letters with their count in alphabetical order
- `--text-get-letter-count-sorted`: outputs letters with their count in descending order by their count
- `--text-get-most-common`: outputs the most common word(s). If 2 words tie for the highest frequency, print both in alphabetical order
- `--text-get-least-common`: outputs the least common word(s). If 2 words tie for the lowest frequency, print both in alphabetical order

10.2 Tips

- A word is defined as a string of characters that are either alphabetic letters, or are the characters `'-'` or `'\''`.
- Use `isalpha()` to help you separate words.
- Use `std::unordered_map` to help you keep counters of words and characters.
- You can use an `std::vector` of `std::pair` to manage printing the list of characters.
- Use `std::sort()` to sort the `std::vector` of characters, depending on either the alphabet value, or the frequency count.
- Use `std::boolalpha` to make `std::cout` print boolean values as `true` or `false` rather than 0 or 1.

11 Wave Files

Wave files store unencrypted, lossless audio data.

Wave files have three parts: the RIFF chunk descriptor, “fmt” sub-chunk, “data” sub-chunk. The information for bitrate and channel count is in the “fmt” sub-chunk. See the following figure.

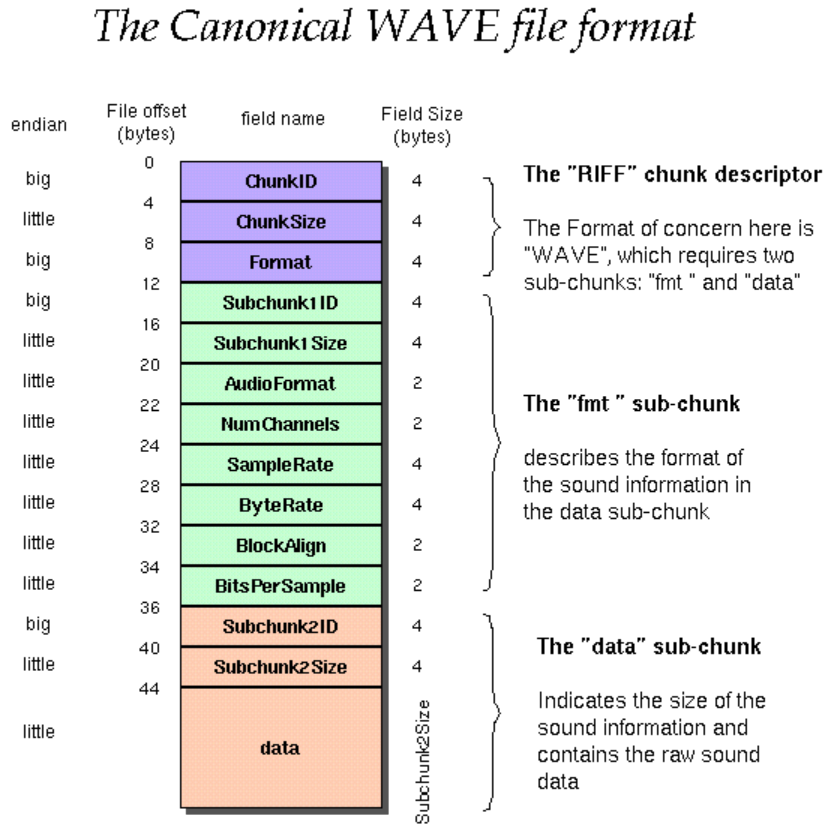

Figure from https://ccrma.stanford.edu/courses/422-winter-2014/projects/WaveFormat/

Supported file type: `.wav`.

11.1 Options

- `--wav-get-bitrate`: outputs the bitrate (8× the byterate)
- `--wav-get-channels`: outputs the number of channels

11.2 Tips

- Use `FileAnalyzerFile::ReadLittleEndianInt32()` to read the 4 byte little-endian value for "ByteRate".
- Use `FileAnalyzerFile::ReadLittleEndianInt16()` to read the 2 byte little-endian value for "NumChannels".