Assignment 01

CSE 241

April 3, 2024

You are to finish a C++ program which reads in an audio file[1] and finds periods of silence in it. Part of the program has been completed for you already.

[1] Specifically, a .wav file

Description

Your program will prompt the user for a filename. It will open the file and analyze it. Within your program, you will interact with the audio data as an array of `shorts`. It will print out two pieces of information:

1. The peak amplitude. That is, the value which is furthest away from 0. If -32768 is the value is furthest away from 0, you should truncate that to become 32767. Using C++'s builtin `std::abs` function will do this for you automatically.
2. A list of all starting times of periods of silence, measured in seconds

Audio

Audio is represented digitally as a 1-dimensional array of numbers. For this assignment, it will specifically be a 1-dimensional array of `shorts`, which matches how most CD-quality audio is stored.[2]

[2] For those who have some prior digital signal processing experience, note that we are considering only a single channel of audio, and the audio is linear PCM with bias 0.

As a note on terminology:

amplitude —The distance from the zero point. In our case, this is equivalent to the absolute value of the sample. For example, a sample of -2198 has an *amplitude* of 2198.

sample —Each element of an array in digital audio is referred to as a *sample*. In our case, each sample ranges from -32768 to 32767 in value, or `SHRT_MIN` to `SHRT_MAX`.

sample rate —The sample rate relates the array data to real-world time. The *sample rate* is measure in samples per second. For CD-quality sound, 44100 samples per second is standard, though your code will need to work with a variety of sample rates.

Silence

For the purposes of this assignment, we will define *silence* to be a contiguous range of samples which:

- Has a sustained amplitude below 1000. That is, there is no sample in the range that has an amplitude of 1000 or greater.
- Has a length of at least 0.5 seconds.

Output

You will output the peak amplitude as an integer amplitude. You will output each period of silence as a floating-point number representing the real-world time[3] corresponding to the *beginning* of the period of silence. You will output with 2 digits before the decimal place, and 4 digits after the decimal place. You will output a maximum of 10 periods of silence.

[3] seconds

Sample usages

```
Enter filename: s02.wav
Peak amplitude: 30988
Periods of silence:
 0.0998
 0.7498
 1.3998
 2.0498
 2.6998
 3.3498
 3.9998
 4.6498
 5.2998
 5.9498
```

```
Enter filename: s01.wav
Peak amplitude: 22768
Periods of silence:
 0.9993
 2.3743
 3.9967
```

```
Enter filename: s00.wav
Peak amplitude: 32767
Periods of silence:
 0.9999
```

```
Enter filename: inp.wav
Peak amplitude: 32767
Periods of silence:
24.0015
35.0016
```

Requirements

Restrictions

You may *not* use dynamic memory management[4] for this assign-

[4] which has not been discussed yet in class anyway

ment. Even if you know what dynamic memory management is, it is worthwhile to practice how to write code for an environment where it does not exist or is very limited. So, for this assignment, dynamic memory management is forbidden.

Global variables are also forbidden.

You may *not* call `wav_file_data` for every sample to be processed. Buffering part of the audio data into an array is required.[5]

[5] I recommend an array size of a few thousand elements.

Your total call stack memory usage must never exceed 1 million bytes.

Functions

You *must* write[6] the following function:

[6] and make use of

```
size_t find_silence(string filename, double* times, size_t max_n)
```

`times` in this case is a pointer to an array to fill up with starting times of periods of silence. `max_n` is the length of the array. If there are too many times found, the function should stop looking for more periods of silence.

The return value is the number of times filled in to the array.

Submitting

You must submit a `.zip` file. The `.zip` file should contain:

- `wav-file.h`[7]
- `wav-file.cc`[8]
- `Makefile`[9]
- `a01.cc`[10]

[7] unchanged

[8] unchanged

[9] You must compile with the address sanitizer, and `make` must compile and link an executable called `a01`

[10] Your source file

Nothing else should be in the `.zip` file.

No commenting or documentation is required with the submission. If you have received help from a friend, online, or used AI assistance, please add a comment to your submission saying so so that you cannot be processed for plagiarism.