Алгоритмічні мови та програмування

Лабораторна робота № 8

«Маніпуляції зі складними структурами. Нормалізація WAVE-файлу»

Теоретичні відомості:

WAVE-формат призначений для зберігання звуку у цифровій формі у вигляді виборок (samples). Формат поширений на багатьох платформах. Він побудований на принципі контейнерів (вкладених одна в одну структур). Зовнішній контейнер побудований у фоматі RIFF. Файл має таку структуру:

- *Заголовок контейнеру RIFF*
- Тіло контейнеру RIFF
    - Тип даних контейнеру RIFF («WAVE»)
    - ✓ *Заголовок контейнеру формату*
    - ✓ Тіло контейнеру формату
    - ▪ *Заголовок контейнеру даних*
    - ▪ Тіло контейнеру даних

Контейнери мають такі структури:

Зсув	Розмір	Назва за стандартом	Опис	Значення
0x00	4	Chunk ID	Назва контейнера	RIFF
0x04	4	Chunk Data Size	Розмір тіла контейнера	(розмір файла)-8
0x08	4	RIFF Type	Тип даних контейнера	WAVE
0x0C	4	Format Chank ID	Назва контейнера	fmt
0x10	4	Format Chunk Size	Розмір тіла контейнера	0x10, 0x12, …
0x14	2	Format tag	Тип формату	1 – без компресії
0x16	2	Number of channels	**Кількість каналів звуку**	1 або 2
0x18	4	Sample rate	Швидкість потоку (виборок за секунду)	
0x1C	4	Average bytes per second	Швидкість потоку (байт за секунду)	
0x20	2	Block Align	Вирівнювання виборки	1,2,…
0x22	2	Bits per sample	**Кількість біт на виборку**	8 або 16
0x24	2	Size of extension	Розмір додаткового блоку	
0x26	…	Extension	Додатковий блок	
…	…			
	4	Data chunk ID	Назва контейнера	data
	4	Data chunk size	Розмір тіла контейнера	Сума розмірів усіх виборок
	…	Samples…	Виборки	

Слід зауважити, що структура контейнера формату нерегулярна - якщо значення «`Format Chunk Size`» дорівнює 0x10, то поля «`Size of extension`» та «`Extension`» відсутні.

Між контейнерами формату та даних у різних реалізаціях можуть розташовуватись ще кілька контейнерів. Для надійного позиціонування початку контейнера даних, його слід визначати за строкою «data».

Виборки у контейнері даних записуються залежно від кількості каналів та кількості біт на виборку.

Для одноканального (моно) звуку, виборки йдуть послідовно, для двоканального (стерео) – по черзі, спочатку виборка першого (лівого) каналу, потім – другого (правого).

Для 8-бітного звуку виборки займають по 1 байту та обробляються, як тип `unsignrd char`, причому відсутності звуку відповідає значення 128 (0x80). Для 16-бітного звуку виборки займають по 2 байти, та обробляються як `signed short` і мають значення в діапазоні –32768 (-0x8000) до 32767 (0x7fff), причому відсутності сигналу відповідає значення 0.

Нормалізацією цифрового звуку називають помноження значень виборок на такий коефіцієнт, щоб найбільше за амплітудою значення досягло максимальної для даного типу величини. Таким чином відбувається підвищення гучності звуку без внесення спотворень.

Завдання:

Написати програму, яка виконує такі дії:

1. Отримує з командної строки, або стандартного вводу два імені файлів – вхідного та вихідного.
2. Об’являє структуру riff_header згідно наведеної вище таблиці.
3. Об’являє покажчики на масиви значень виборок для чотирьох випадків: 8-біт моно, 8-біт стерео, 16-біт моно, 16-біт стерео.
4. Вичитує з файлу стркутуру формату.
5. Виходячи зі значень кількості каналів та кількості біт на виборку, виділяє пам’ять під відповідні масиви виборок, обчислює кількість виборок та вичитує виборки у масиви.
6. Знаходить найбільше та найменьше значення амплітуди виборки.
7. Обчислює коефіцієнти нормалізації для додатніх та від’ємних амплітуд.
8. Виводить у стандартний потік докладні відомості про файл та обчислені коефіцієнти нормалізації.
9. Нормалізує масиви виборок.

10. Записує у вихідний файл структуру формату та масиви виборок.

Записаний файл прослухати за допомогою програвача, та пересвідчитись, що він зібраний правильно

Примітки:

Для правильного зберігання даних формату, застосовуйте такі визначення:

```
typedef unsigned char   byte1;
typedef unsigned short  byte2;
typedef unsigned int    byte4;
typedef struct _riff_header {
        char    riff_title[4];
        byte4   riff_chank_size;
        char    wave_title[4];
        char    fmt_title[4];
        byte4   fmt_size;
        byte2   format_tag;
        byte2   n_channels;
        byte4   n_samples_per_sec;
        byte4   n_avg_bytes_per_sec;
        byte2   n_block_align;
        byte2   n_bits_per_sample;
} riff_header;
```

Для читання та запису неформатованих даних у файл, використовуйте функції:

```
#include <stdio.h>
size_t
fread(void * restrict ptr, size_t size, size_t nmemb,
FILE * restrict stream);
size_t
fwrite(const void * restrict ptr, size_t size, size_t
nmemb,
FILE * restrict stream);
```

тут `ptr` – покажчик на дані для читання/запису, `size` – розмір елементу даних, `nmemb` – кількість елементів даних, `stream` – відкритий файл.