# ФИТ НГУ, курс ООП, осенний семестр 2022

# Задача 3. Sound Processor

Написать программу, которая получает на вход один или несколько звуковых файлов формата WAV и текстовый файл с описанием того, что надо сделать со звуком. Программа выполняет преобразования звука и сохраняет результат в выходной файл формата WAV.

Программа должна запускаться следующим образом:

> sound\_processor [-h] [-c config.txt output.wav input1.wav [input2.wav …]]

WAV-файл представляет собой контейнер для хранения звукового потока. Он состоит из заголовка и данных в виде последовательности сэмплов. Сэмпл — это величина амплитуды звукового сигнала в каждый момент времени. Частота — это количество сэмплов в секунду. Последовательность сэмплов будем называть потоком (не путать с потоками ввода/вывода в C++).

Аудио-данные внутри WAV-файла могут храниться в разных форматах. Мы ограничимся поддержкой только одного варианта. **Поддерживаемый формат:**

1. Аудио-формат PCM (без кодирования).
2. Моно-звук (1 канал).
3. Разрядность 16 бит со знаком (signed 16 bit (little-endian)) — это значит, что один сэмпл представлен целым 16-битным числом со знаком.
4. Частота 44100 Гц.

Для тестирования программы можно получить WAV-файл в таком формате из любого другого аудиофайла с помощью утилиты [ffmpeg](https://ffmpeg.org/download.html):

> ffmpeg -i Radioactive.flac -f wav -bitexact -acodec pcm\_s16le -ar 44100 -ac 1 "Radioactive\_mono\_16bit.wav"

Рекомендуем использовать свои файлы, но можно также воспользоваться готовыми примерами [отсюда](https://drive.google.com/file/d/1wzvJK09WI7UzXciJLKjWT_w6g4JWruWJ/view?usp=sharing) (они уже в нужном формате).

**Конвертером** назовем компонент, который получает на вход один или несколько потоков сэмплов и выдает один выходной поток сэмплов. Конвертер имеет настраиваемые параметры.

Конвертеры, обязательные к реализации:

1. Заглушить интервал. Два параметра: начало и конец интервала времени (в секундах). В выходном потоке в этом интервале времени — тишина, остальное — как во входном потоке.
2. Миксер. Два параметра: дополнительный поток и место вставки (в секундах, по умолчанию 0). Выходной поток: каждый сэмпл является средним значением соответствующих сэмплов входных потоков. Длина выходного потока соответствует длине основного входного потока. Eсли дополнительный поток длиннее основного, то часть его обрезается (не используется), а если короче, то выходной поток за его пределами равен основному входному потоку.
3. Третий конвертер на ваш выбор (придумайте сами), но такой, чтобы результат его работы можно было услышать. У этого конвертера должен быть минимум 1 параметр.

*Можно реализовать дополнительные конвертеры и/или дополнительные параметры, расширяющие функциональность обязательных конвертеров (на усмотрение автора).*

**Конфигурация** конвертеров определяет последовательность применения конвертеров с разными параметрами. Она задается в текстовом файле такого вида:

# заглушить первые 30 секунд input1

mute 0 30

# смиксовать с input2, начиная с секунды 10

mix $2 10

# заглушить 3-ю минуту в миксованном потоке

mute 120 180

# применить ваш конвертер с нужными параметрами

your\_converter <parameters>

Каждая строка файла — это конфигурация одного конвертера. Поддерживаются комментарии (строки, начинающиеся с #).

Правила исполнения:

1. Конвертеры применяются последовательно в порядке, заданном в файле конфигурации.
2. Первый конвертер получает на вход первый входной поток (input1). Второй конвертер получает на вход выходной поток первого конвертера, и т.д.
3. Выходной поток последнего конвертера является результатом работы программы.
4. Аргументами конвертера могут быть целые числа и ссылки на дополнительные потоки. Ссылка имеет формат $n, где n — порядковый номер входного файла, начиная с 1. $1 — это ссылка на input1.wav, $2 — input2.wav, и т.д. *Можно реализовать дополнительные типы аргументов (на усмотрение автора).*
5. Если конвертер требует дополнительного потока (например, mix), то он получает его отдельную копию. Иными словами, можно несколько раз использовать один и тот же файл (включая input1), и каждый раз поток из файла читается с начала.

Программа запускается так:

> sound\_processor -c <config.txt> <output.wav> <input1.wav> [<input2.wav> …]

Она должна читать все входные файлы, файл конфигурации, выполнять конверсию и сохранять результат в output.wav. output.wav должен быть корректным WAV-файлом, а значит, содержать правильный заголовок. Но не обязательно копировать заголовки входных потоков. Достаточно сгенерировать минимально необходимый заголовок, чтобы файл можно было открыть в проигрывателе.

Если проверка работы программы на слух невозможна, можно открыть output.wav в аудио редакторе (например, [Audacity](https://www.audacityteam.org)) и проверить визуально.

### Технические требования

1. Количество используемой оперативной памяти не зависит от размера потоков.
2. Код должен быть разделен на логические части: работа с форматом WAV, конвертеры, подготовка и исполнение процесса конверсии. Работа с WAV и конвертеры полностью независимы. Последняя часть связывает всё воедино через интерфейсы.
3. Создание конвертеров нужно реализовать посредством шаблона проектирования [«фабричный метод»](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B1%D1%80%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)).
4. Разбор конфигурационного файла должен быть реализован в отдельном классе.
5. Программа должна определять ситуацию, когда кодирование входного файла не поддерживается, и выдавать корректное сообщение об ошибке.
6. Для обработки ошибок и исключительных ситуаций нужно использовать механизм исключений C++. Нужно разработать иерархию исключений, которые будут выбрасываться при исполнении. При возникновении исключения программа выводит сообщение об ошибке в std::cerr и завершается с ненулевым кодом, различным для каждого типа ошибки.
7. Справка об использовании программы должна содержать информацию о поддерживаемых конвертерах, которая генерируется динамически (поскольку набор конвертеров расширяем). Для каждого конвертера должен быть описан его смысл, параметры и синтаксис в конфигурационном файле.
8. Желательно добавить юнит-тесты. В качестве библиотеки для тестирования используйте [Google Test Framework](https://google.github.io/googletest/).

### При сдаче задания продемонстрировать

1. Вывод справки об использовании программы (-h).
2. Несколько рабочих конфигураций.
3. Несколько ошибочных ситуаций, в том числе:
   1. ошибки в аргументах командной строки,
   2. не поддерживаемая кодировка входных файлов,
   3. ошибки в файле конфигурации.