Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

# Низкоуровневое программирование
# Лабораторная работа №1

Выполнил студент: **Соловьев П.А.**
Группа: **P33302**
Преподаватель: **Кореньков Ю.Д.**

Санкт-Петербург
2023

# Цели

Целью данной работы является разработка модуля, реализующего хранение в файле данных информации, поддерживающим операции вставки, удаления, выборки и обновления элементов данных. По варианту необходимо реализовать документо-ориентированную базу данных.

# Задачи

## Создание Git репозитория и настройка системы сборки

Первым делом был создан локальный git репозиторий и написана конфигурация для CMake. IDE успешно подхватила конфигурацию и собрала минимальный пример кода. Далее был создан удаленный репозиторий на Github, чтобы не бояться что ssd-шник вдруг умрет.

## Проектирование и реализация.

Следовал подходу снизу-вверх. Начал с реализации функций взаимодействия с ФС, далее имплементировал уже основной функционал.

## Тестирование

Test-Driven Development не получился, но старался по мере написания кода добавлять тесты к новому функционалу. Далее были бенчмарки для замера производительности. Написал Github Workflow для автоматической сборки и тестирования проекта на обеих платформах.

## Отчет

Собственно, написание этого отчета.

# Выполнение

## Описание работы

Программа состоит из трех основных модулей - модуль с реализаций, тестами и бенчмарками. Внешне модуль, реализующий документное дерево, предоставляет интерфейс для выполнения основных операций с узлами документного дерева. Для этого используются отдельные структуры. Для тестирование используется фреймворк  GoogleTest .

## Аспекты реализации

Все данные хранятся в одном файле, который разбит на страницы фиксированной длины. В заголовке файла хранится мета-информация о содержимом документа. Страница в свою очередь содержит метадату (информацию о элементе) и сами данные (элементы). Для управления страницами в памяти используется структура `PageManager`, которая хранит текующую свободную страницу в памяти и общую информацию о страницах. Слой взаимодействия с файлом скрыт за абстракцией `FileManager`. Страницы отображаются в память 1 к 1, что позволяет записывать и читать страницу из файла за 1 операцию с диском. В памяти страницы хранятся в виде связного списка, элементы которого аллоцируются и освобождаются по мере прохода через файл.

На более высоком уровне реализованы итераторы `PageIterator` и `ItemIterator`, использующие «под капотом» `PageManager` и `Page` для получения/добавления элемента. На этом уровне уже отсутвует прямое взаимодействие с файлом, а аллокация памяти под страницы/элементы производится в отдельных функциях со счетчиками элементов во избежание утечек.

## Графики

Batch = 20 элементов

Вставка элементов проводилась поочередно с удалением. Вставляем 20 случайных элементов, 10 удаляем в цикле. Замеры производились каждые 20 операций.

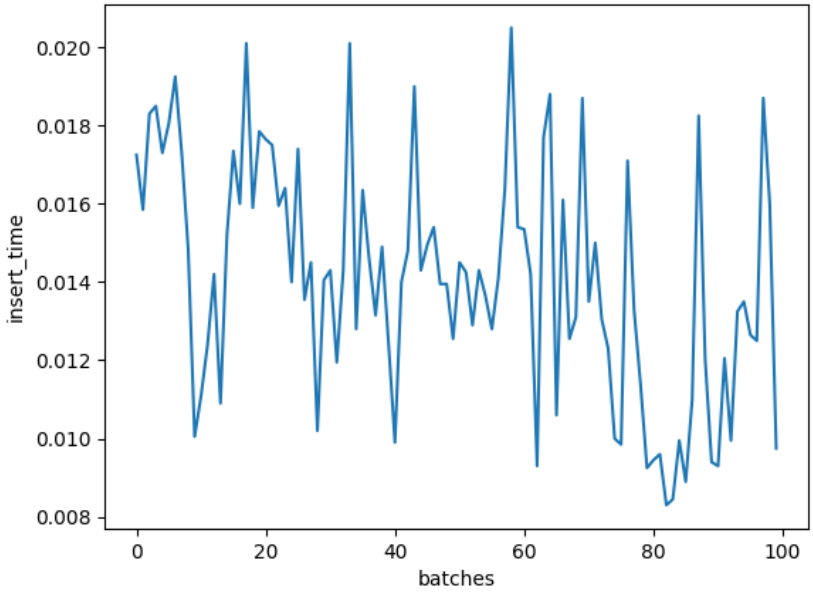


Рисунок 1: Вставка элементов (ms)

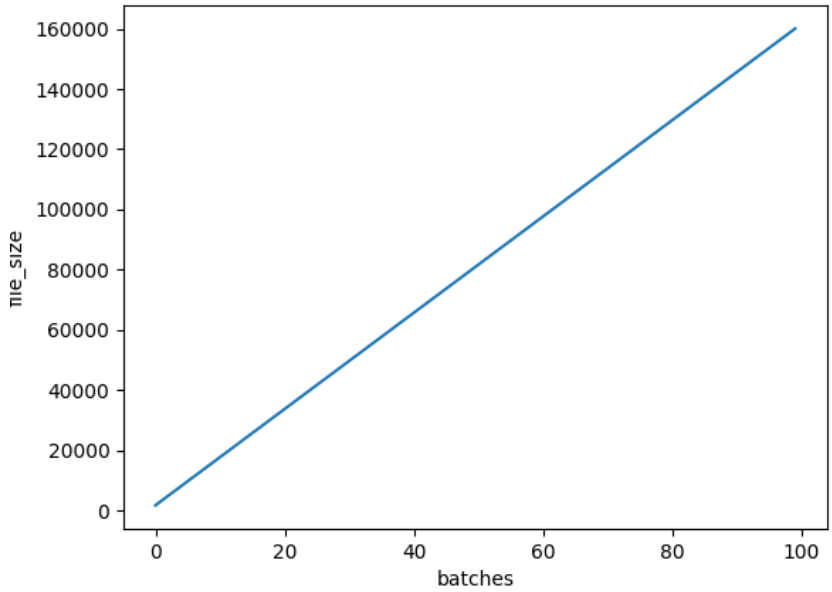


Рисунок 2: Размер файла (bytes)

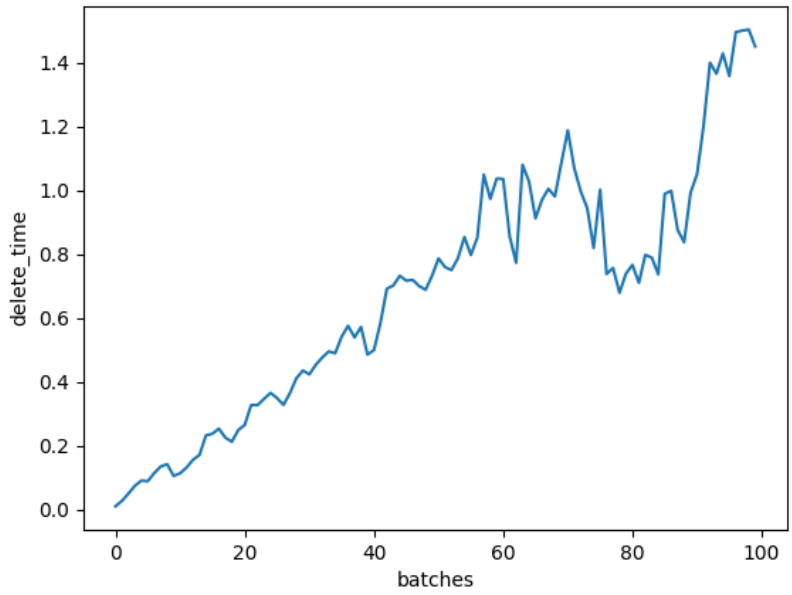


Рисунок 3: Удаления элементов (ms)

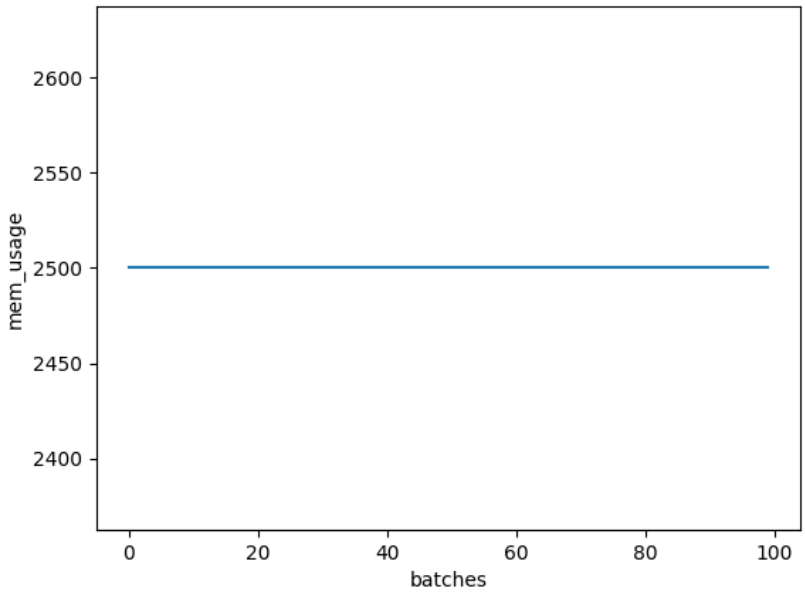


Рисунок 4: Использование памяти (bytes)

Чередование вставки и удаления. 100 повторений:

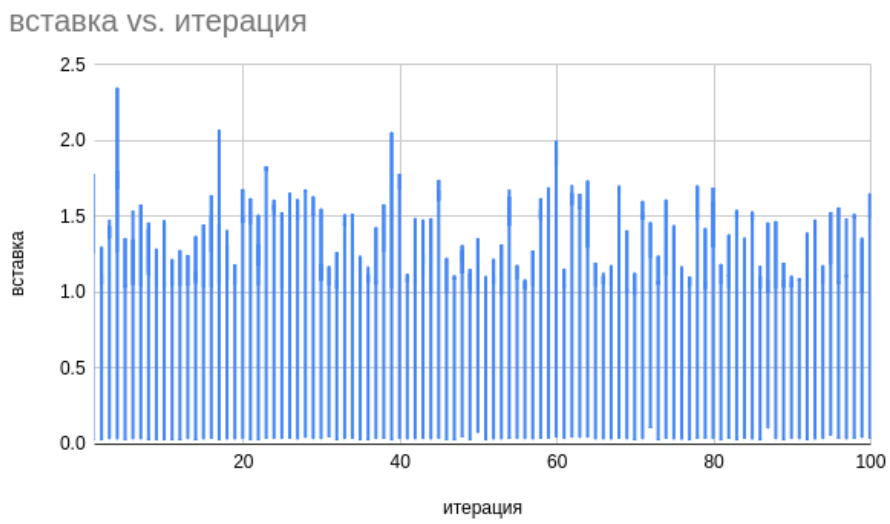


Рисунок 5: Вставка 500, замер каждые 100 элементов

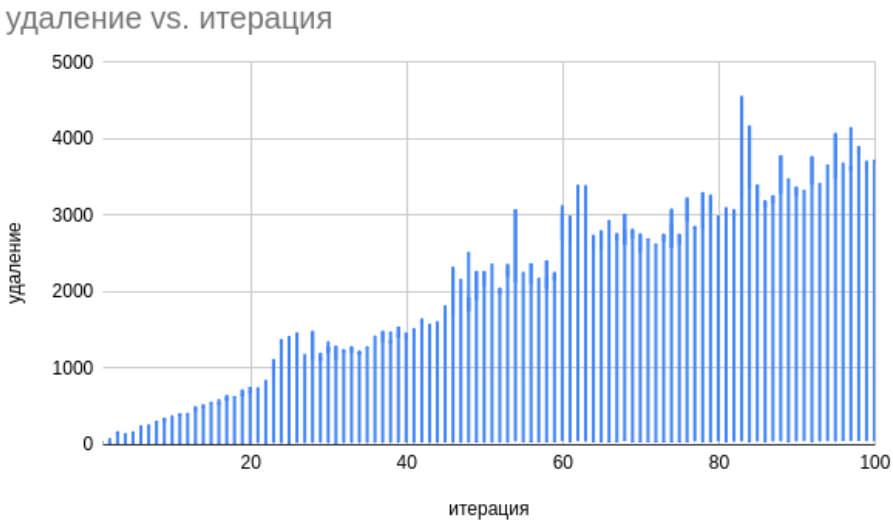


Рисунок 6: Удаление 400, замер каждые 100 элементов

## Выводы

В результате выполнения работы был разработан модуль, реализующий документо-ориентированную базу данных. По результатам тестов можно судить, что временная ложность выполнения системой запросов соответствует требуемой.

В ходе выполнения задания я научился проектировать и реализовывать крупное приложение на языке C, справляться с `SEGFAULT`-ами, логировать работу приложения и измерять его производительность.