

Федеральное государственное автономное образовательное учреждение высшего
образования «Национальный исследовательский университет ИТМО»
Факультет программной инженерии и компьютерной техники

Лабораторная работа №1 по дисциплине
«Низкоуровневое программирование»

Выполнил:

Миху Вадим Дмитриевич

Факультет «ПИиКТ»

Группа: P33301

Преподаватель:

Кореньков Юрий Дмитриевич

Вариант 5



Санкт-Петербург, 2023 г.

Оглавление

Задание	2
Выполнение	3
Выводы по работе	5

Задание

Вариант №5

Создать модуль, реализующий хранение в одном файле данных (выборку, размещение и гранулярное обновление) информации общим объёмом от 10GB соответствующего варианту вида.

1. Спроектировать структуры данных для представления информации в оперативной памяти
 - Для порции данных, состоящий из элементов определённого рода (см форму данных), поддерживать тривиальные значения по меньшей мере следующих типов: четырёхбайтовые целые числа и числа с плавающей точкой, текстовые строки произвольной длины, булевские значения
 - Для информации о запросе
2. Спроектировать представление данных с учетом схемы для файла данных и реализовать базовые операции для работы с ним:
 - Операции над схемой данных (создание и удаление элементов схемы)
 - Базовые операции над элементами данных в соответствии с текущим состоянием схемы (над узлами или записями заданного вида)
 - i. Вставка элемента данных
 - ii. Перечисление элементов данных
 - iii. Обновление элемента данных
 - iv. Удаление элемента данных
3. Реализовать тестовую программу для демонстрации работоспособности решения
 - Параметры для всех операций задаются посредством формирования соответствующих структур данных
 - Показать, что при выполнении операций, результат выполнения которых не отражает отношения между элементами данных, потребление оперативной памяти стремится к $O(1)$ независимо от общего объёма фактического затрагиваемых данных
 - Показать, что операция вставки выполняется за $O(1)$ независимо от размера данных, представленных в файле
 - Показать, что операция выборки без учёта отношений (но с опциональными условиями) выполняется за $O(n)$, где n – количество представленных элементов данных выбираемого вида
 - Показать, что операции обновления и удаления элемента данных выполняются не более чем за $O(n \cdot m) > t \cdot O(n+m)$, где n – количество представленных элементов данных обрабатываемого вида, m – количество фактически затронутых элементов данных
 - Показать, что размер файла данных всегда пропорционален количеству фактически размещённых элементов данных
 - Показать работоспособность решения под управлением ОС семейств Windows и *NIX

Выполнение

В ходе работы была реализована программа, исходный код которой был опубликован на GitHub

<https://github.com/filberol/lab1llp>

Был написан Cmake файл, позволяющий запускать программу на платформах Windows и *nix.

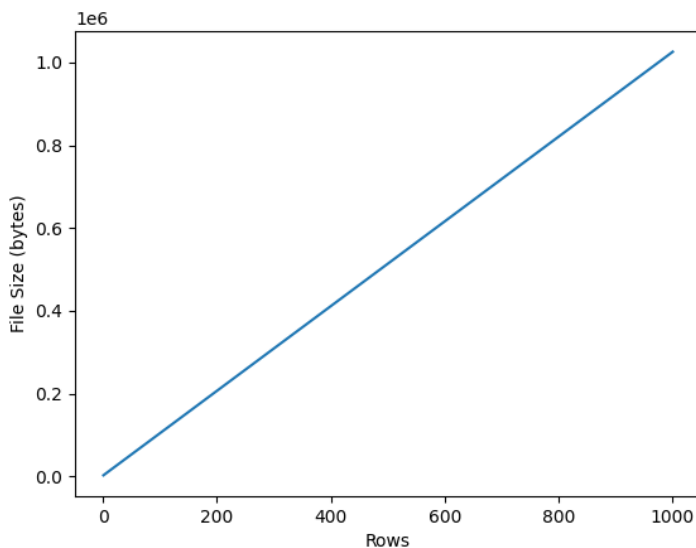
Файл сборки содержит два таргета, первый из которых запускает тесты и демонстрирует работоспособность программы, а второй запускает бенчмарки, тестирующие эффективность исполнения методов вставки и обновления строк в таблице.

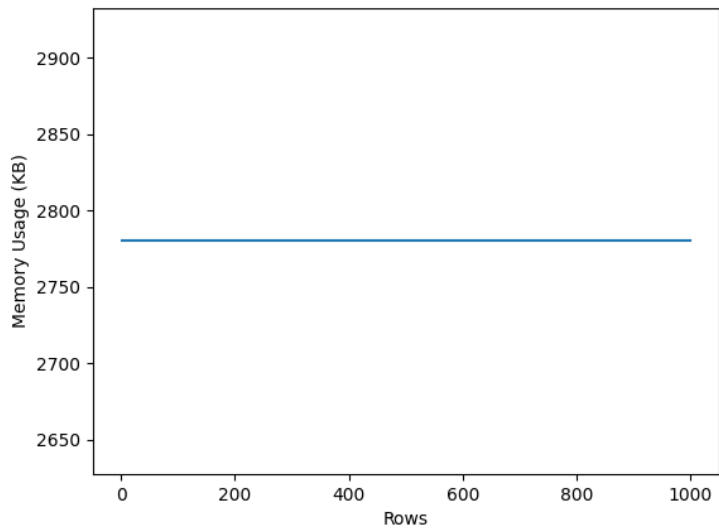
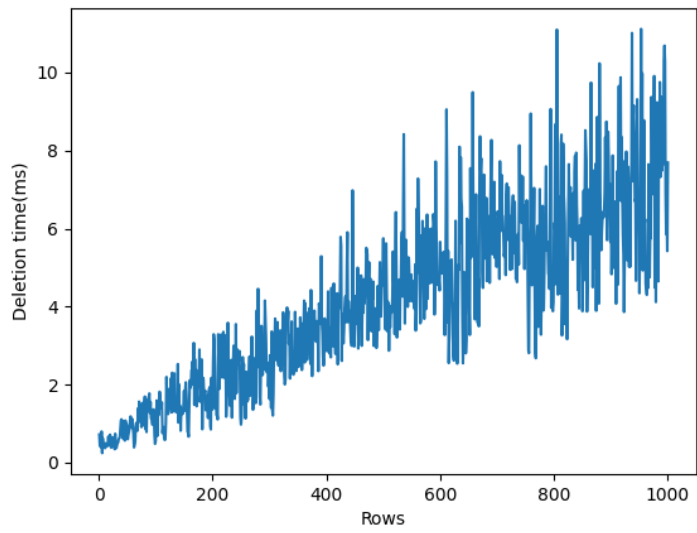
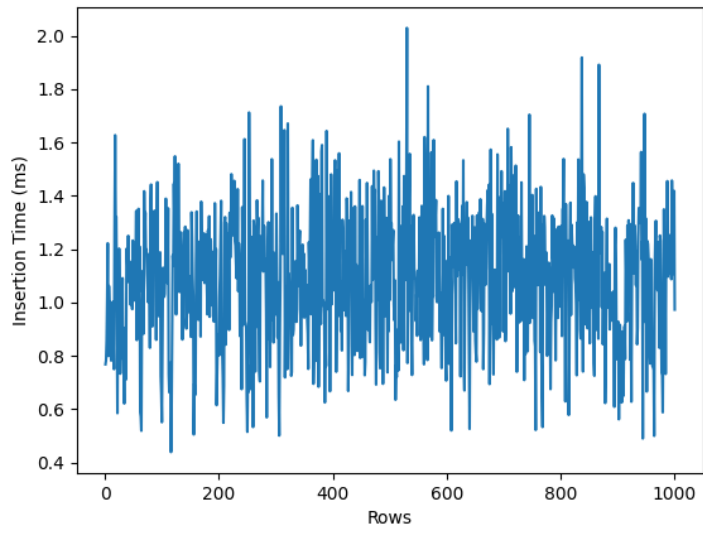
Бенчмарки запускаются только на платформе Linux, потому что используют встроенные инструменты.

Реализация операций:

1. Аллокация. Файл разбит на секторы, в которых содержатся некоторые данные. Секторы реализованы в виде связного списка, они могут динамически занимать и высвобождаться.
2. Создание таблиц. Таблица содержит в себе метаданные, а также набор данных о схеме таблицы.
3. Последовательные данные. Последовательные элементы также реализованы в виде связного списка, что обеспечивает вставку элемента за константу. Поиск же осуществляется последовательно.
4. Выборка. Все выборки и модификации реализованы через итерацию и сравнение.
5. Удаление. Удаление происходит с помощью переназначения ссылок и освобождения секторов.

Приведем графики, отражающие скорость операций и расхода памяти в зависимости от заполненности файла (графики были построены с помощью скрипта, который лежит в репозитории вместе с результатами тестирования):





Проведенные тесты:

```
Test 1 - Read and write header
Header read from the beginning of the file:
Table Count: 10
Sector Table Count: 5
First sector at end free: 1
Test 2 - Does allocating change header
Allocated sectors 1 and 4
Test 3 - Trying to read something from sector
Allocated from sector 1
Test table hash 0
Test 4 - Are table indices synchronized?
Default global malloced Tables found
Hash: -1960042739      Sector: 3
Hash: 2008523766      Sector: 13
Modified Tables found
Hash: -1960042739      Sector: 33
Test 5 - Writing data to sector
Allocated sector 3
Writing TEst STring - result 0
Reading TEst STring - result 0
Test 6 - Are tables saved correctly
Tables found
Hash: -1960042739      Sector: 3
Table scheme sector 3
Name: table1
Columns count: 3
Column id      type Int
Column value   type Bool
Column string  type VarChar
Test 7 - Filling table
test_string
1      2
```

Выводы по работе

В ходе работы была реализована программа, позволяющая хранить большое количество данных, организованных в виде реляционных таблиц. Закреплены на практике навыки организации данных с указанными в задании ограничениями и работа с ними, с использованием построенных данных.