МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине

“Низкоуровневое программирование”

Вариант № 3

Граф узлов с атрибутами

Студент:

Степанов Михаил
Андреевич

Группа P33312

Преподаватель:

Кореньков Юрий Дмитриевич

Санкт-Петербург, 2023

Задание:

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

- Спроектировать структуры данных для представления информации в оперативной памяти
    - Для порции данных, состоящий из элементов определённого рода (см форму данных), поддержать 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
- 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)
        - Вставка элемента данных
        - Перечисление элементов данных
        - Обновление элемента данных
        - Удаление элемента данных

- Используя в сигнатурах только структуры данных из п.1, реализовать публичный интерфейс со следующими операциями над файлом данных:
    - Добавление, удаление и получение информации о элементах схемы данных, размещаемых в файле данных, на уровне, соответствующем виду узлов или записей
    - Добавление нового элемента данных определённого вида
    - Выборка набора элементов данных с учётом заданных условий и отношений со смежными элементами данных (по свойствам/полями/атрибутам и логическим связям соответственно)
    - Обновление элементов данных, соответствующих заданным условиям
    - Удаление элементов данных, соответствующих заданным условиям
- 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*m) → t 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
- Результаты тестирования по п.4 представить в составе отчёта, при этом:
    - В части 3 привести описание структур данных, разработанных в соответствии с п.1
    - В части 4 описать решение, реализованное в соответствии с пп.2-3
    - В часть 5 включить графики на основе тестов, демонстрирующие амортизированные показатели ресурсоёмкости по п. 4

Описание:

- *node* - именованный узел семантической сети

```
struct node {
    int32_t id{};
    std::string node_name;
    std::string node_class;
    std::unordered_map<std::string, property> props;
    std::unordered_map<std::string, relationship> relations;
}
```

- *property* - именованная характеристика узла

```
struct property {
    enum type {INT, BOOL, FLOAT, STRING};
    type tag;
    int32_t int_val;
    bool bool_val{};
    float float_val{};
    std::string string_val;
}
```

- *relationship* - именованное отношение между узлами

```
struct relationship {
    std::string name;
```

```
    std::string related_with;
}
```

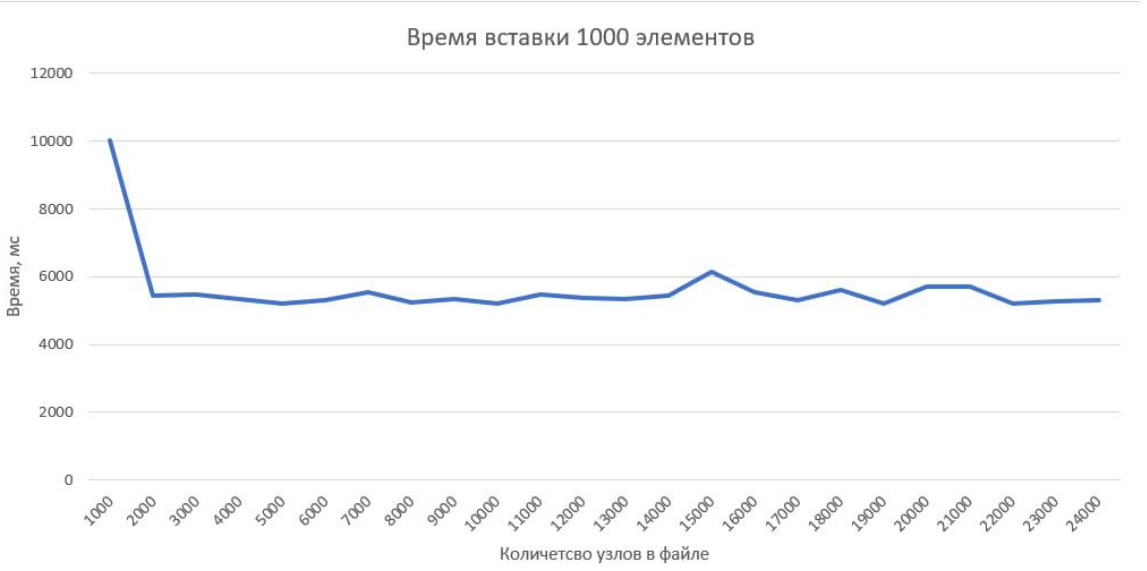
- *meta* - метаинформация о хранящихся данных

```
struct meta{
    int32_t node_count = 0;
    std::vector<int32_t> free;
    std::unordered_map<std::string, int32_t> node_names;
    std::unordered_map<std::string, std::unordered_set<int32_t>>
node_classes;
}
```

Аспекты реализации:

- Первые 256 Мб файла отведены под метаинформацию о данных, записывающихся в файл
- Каждый узел занимает 4 Кб памяти
- Доступ к конкретному узлу осуществляется за O(1), благодаря использованию unordered_map при записи данных о node в метаинформацию
- При удалении элемента удаляется только метаинформация об этом элементе, при последующем обращении к записи память под удаленный элемент будет занята новым элементом
- Сериализация и десериализация осуществляется с помощью библиотеки *cereal*

Результаты:




Время выборки 1000 эелментов
140
120
100
80
60
40
20
0
Время, мс
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000
19000
20000
21000
22000
23000
24000
Количество узлов в файле


Время на удаление 1000 элементов
120
100
80
60
40
20
0
Время, мс
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000
19000
20000
21000
22000
23000
24000
Количество узлов в файле

Вывод:

В результате выполнения лабораторной работы был разработан модуль, реализующий хранение в одном файле данных в виде графа узлов с атрибутами, объем которых может достигать 10GB. Модуль поддерживает операции select, insert, update, delete. Модуль может работать под управлением ОС семейств Windows и *NIX.