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: документное дерево

Студент:
Чернова Анна Ивановна

Группа P33301

Преподаватель:
Кореньков Юрий Дмитриевич

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2023

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

Задание:

1. Спроектировать структуры данных для представления информации в оперативной памяти

2. Спроектировать представление данных с учетом схемы для файла данных и реализовать базовые
операции для работы с ним:
3. Используя в сигнатурах только структуры данных из п.1, реализовать публичный интерфейс со
следующими операциями над файлом данных:
4. Реализовать тестовую программу для демонстрации работоспособности решения

Описание работы:

src/tests.c – тесты

src/demo.c - демо

src/element.c – функции для работы с элементом документа

src/doc.c – функции для работы с документом

src/list.c – функции для работы со списком свободных мест в файле

src/file.c – функции для работы с файлом

src/operatiin.c – основные функции для работы с документом в файле

src/condition.c – функции для работы с условиями

src/query.c - функции для работы с запросами

Структура файла и схемы документа:

```
typedef struct {
    uint64_t firstPosition;
    uint64_t endPosition;
    uint64_t count;
    int64_t root;
} head_t;

typedef struct {
    head_t header;
    FILE *F;
    sortedList_t list;
} file_t;

typedef struct meta_t {
    char name[13];
    int64_t id;
    uint64_t size;
    int64_t child;
    int64_t brother;
    uint64_t count;
} meta_t;

typedef struct {
    meta_t meta;
```

```
    element_t **elements;
} document_t;
```

Создание схемы документа и заполнение ее элементами:

```
document_t *rootDocument = createDocument("root");
if (rootDocument) {
    addElement(rootDocument, stringElement("str1", "meow"));
    addElement(rootDocument, intElement("int1", 1234));
    addElement(rootDocument, doubleElement("double1", 999.12));
    addElement(rootDocument, booleanElement("bool1", true));
    addElement(rootDocument, stringElement("str2", "meow"));
}
```

Структура запроса:

```
typedef enum queryType_t {
    SELECT,
    INSERT,
    UPDATE,
    DELETE
} queryType_t;

typedef struct query_t {
    queryType_t type;
    char *documentName;
    condition_t *condition;
    document_t *newDocument;
} query_t;
```

В файл можно добавить схему документа:

```
Добавить документ
query_t *createInsertQuery(char *parentDocumentName, document_t *newDocument,
condition_t *condition) {
    return createQuery(INSERT, parentDocumentName, condition, newDocument);
}
```

Схему документа можно прочитать из файла:

```
query_t *createDeleteQuery(char *documentName, condition_t *condition);
```

Схему документа можно удалить из файла:

```
query_t *createDeleteQuery(char *documentName, condition_t *condition);
```

Схему документа можно обновить в файле:

```
query_t *createUpdateQuery(char *documentName, document_t *newDocument,
condition_t *condition);
```

Пример создания, заполнения документного дерева и выполнения запроса для него:

```
file_t *file = createFile("file");

    document_t *rootDocument = createDocument("root");
    if (rootDocument) {
        addElement(rootDocument, stringElement("str1", "meow"));
        addElement(rootDocument, intElement("int1", 1234));
        addElement(rootDocument, doubleElement("double1", 999.12));
        addElement(rootDocument, booleanElement("bool1", true));
        addElement(rootDocument, stringElement("str2", "meow"));
    }

    document_t *childDocument = createDocument("child");
    if (childDocument) {
```

```
        addElement(childDocument, intElement("int1", 383));
        addElement(childDocument, intElement("int2", 444));
    }

    document_t *grandChildDocument = createDocument("grandChild");
    if (grandChildDocument) {
        addElement(grandChildDocument, intElement("int1", 3));
        addElement(grandChildDocument, intElement("int2", 100));
        addElement(grandChildDocument, booleanElement("bool1", false));
    }

    query_t *insertRootDocument = createInsertQuery(NULL, rootDocument,
NULL);
    condition_t *condEquals = condEqual(stringElement("str1", "meow"));
    query_t *insertChildDocument = createInsertQuery("root", childDocument,
condEquals);
    query_t *insertGrChildDocument = createInsertQuery("child",
grandChildDocument, NULL);

    executeQuery(file, insertRootDocument, NULL);
    executeQuery(file, insertChildDocument, NULL);
    executeQuery(file, insertGrChildDocument, NULL);

    destroyDocument(rootDocument);
    destroyDocument(childDocument);
    destroyDocument(grandChildDocument);
    destroyQuery(insertRootDocument);
    destroyCondition(condEquals);
    destroyQuery(insertChildDocument);
    destroyQuery(insertGrChildDocument);

    printf("%s\n", "INSERT DOCUMENTS");
    printf("%s\n", "DOCUMENT TREE STATUS:");
    printDocumentTree(file, "root");

    queryResult_t *selectChildResult = createQueryResult();
    query_t *selectQueryChild = createSelectQuery("child", NULL);
    executeQuery(file, selectQueryChild, selectChildResult);

    printf("%s\n", "SELECT DOCUMENT \"child\" AND CHILD DOCUMENTS");
    printf("%s\n", "SELECT RESULT:");
    printQueryResult(selectChildResult);

    destroyQuery(selectQueryChild);
    destroyQueryResult(selectChildResult);

    condition_t *conditionOr = condAnd(condLess(intElement("int1", 1000)),
condGreater(intElement("int1", 300)));
    queryResult_t *selectResult = createQueryResult();
    query_t *selectQueryWithCondition = createSelectQuery("root",
conditionOr);
    executeQuery(file, selectQueryWithCondition, selectResult);

    printf("%s\n", "SELECT DOCUMENT \"root\" AND CHILD DOCUMENTS BY
CONDITION");
    printf("%s\n", "SELECT RESULT:");
    printQueryResult(selectResult);

    destroyCondition(conditionOr);
    destroyQuery(selectQueryWithCondition);
    destroyQueryResult(selectResult);

    condition_t *updateCondition = condEqual(booleanElement("bool1", true));
    document_t *updateRootDocument = createDocument("updateDoc");
```

```
    addElement(updateRootDocument, stringElement("str2", "updateMeow"));
    query_t *updateRootQuery = createUpdateQuery("root", updateRootDocument,
updateCondition);
    executeQuery(file, updateRootQuery, createQueryResult());

    printf("%s\n", "UPDATE DOCUMENT \"root\" AND CHILD DOCUMENTS BY
CONDITION");
    printf("%s\n", "DOCUMENT TREE STATUS:");
    printDocumentTree(file, "root");

    destroyCondition(updateCondition);
    destroyDocument(updateRootDocument);
    destroyQuery(updateRootQuery);

    query_t *deleteGrChildQuery = createDeleteQuery("grandChild", NULL);
    executeQuery(file, deleteGrChildQuery, NULL);
    printf("%s\n", "DELETE DOCUMENT \"grandChild\" DOCUMENT");
    printf("%s\n", "DOCUMENT TREE STATUS:");
    printDocumentTree(file, "root");

    destroyQuery(deleteGrChildQuery);

    query_t *deleteRootQuery = createDeleteQuery("root", NULL);
    executeQuery(file, deleteRootQuery, NULL);

    printf("%s\n", "DELETE DOCUMENT \"root\" AND DOCUMENT CHILDREN
DOCUMENTS");
    printf("%s\n", "DOCUMENT TREE STATUS:");
    printDocumentTree(file, "root");

    destroyQuery(deleteRootQuery);

    closeFile(file);
```

Результат выполнения:

```
INSERT DOCUMENTS
DOCUMENT TREE STATUS:
root{
    child{
        grandChild{
            }
        }
    }
grandChild{
    int1 = 3
    int2 = 100
    bool1 = false
}
child{
    int1 = 383
    int2 = 444
}
root{
    str1 = "meow"
    int1 = 1234
    double1 = 999.120000
    bool1 = true
    str2 = "meow"
```

```
}
SELECT DOCUMENT "child" AND CHILD DOCUMENTS
SELECT RESULT:
child{
	int1 = 383
	int2 = 444
}
grandChild{
	int1 = 3
	int2 = 100
	bool1 = false
}
SELECT DOCUMENT "root" AND CHILD DOCUMENTS BY CONDITION
SELECT RESULT:
child{
	int1 = 383
	int2 = 444
}
UPDATE DOCUMENT "root" AND CHILD DOCUMENTS BY CONDITION
DOCUMENT TREE STATUS:
root{
	child{
		grandChild{
			}
		}
	}
grandChild{
	int1 = 3
	int2 = 100
	bool1 = false
}
child{
	int1 = 383
EMPTY
```

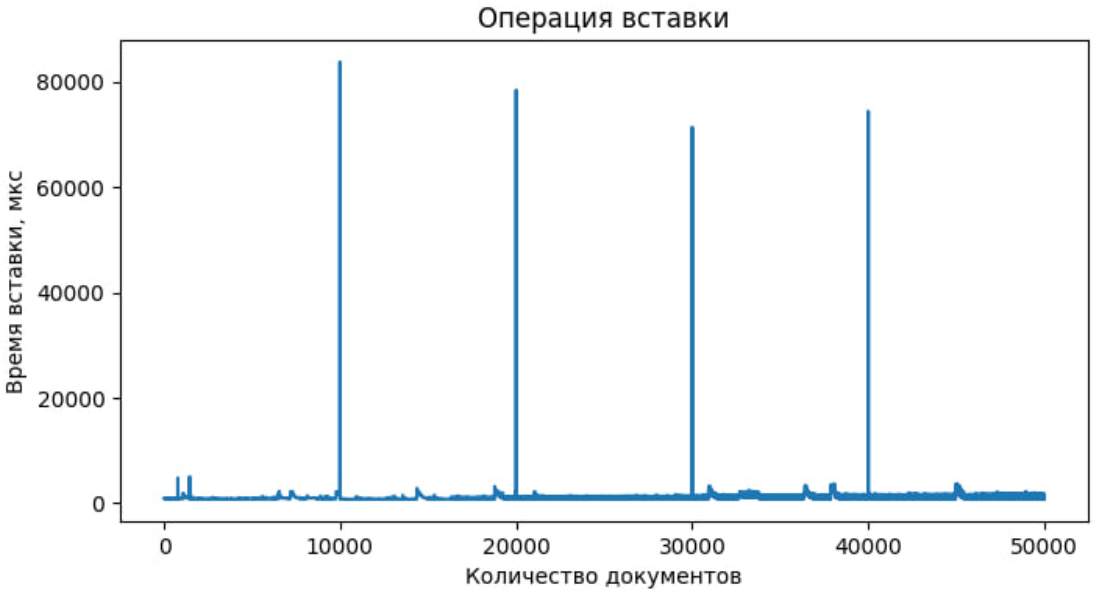
Аспекты реализации:

Файл содержит заголовок, массив индексов документов и документы.

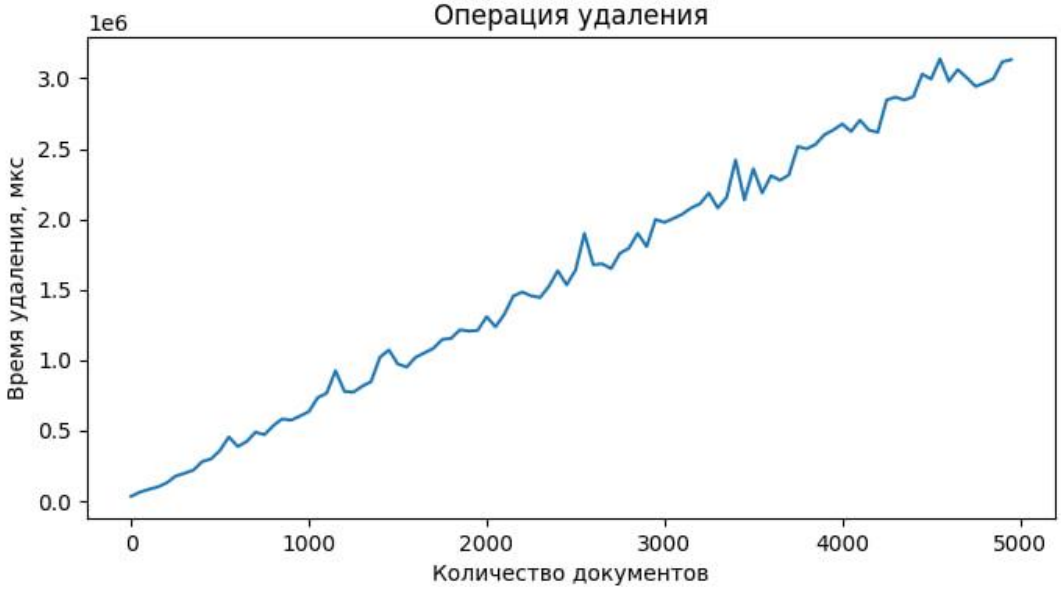
При удалении документа его индекс обновляется, становится доступным для записи нового документа и добавляется в отсортированный список свободных мест в файлу (еще не использовавшихся и старых индексов). Когда свободных индексов нет, нужно выделять место под новые индекса

Результаты тестов:

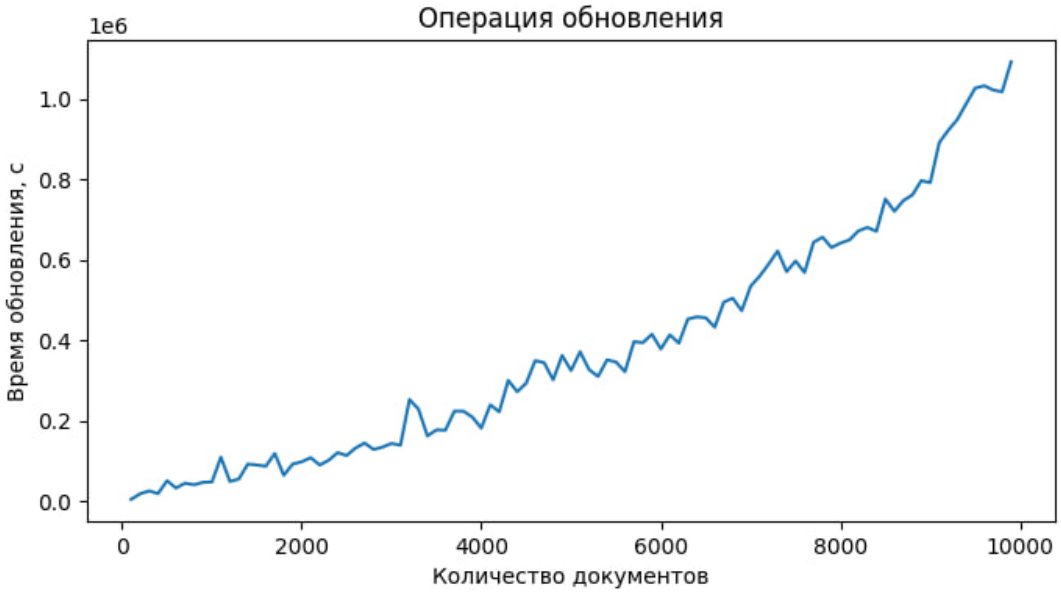
Операция вставки – на графике видно, что осуществляется за O(1)



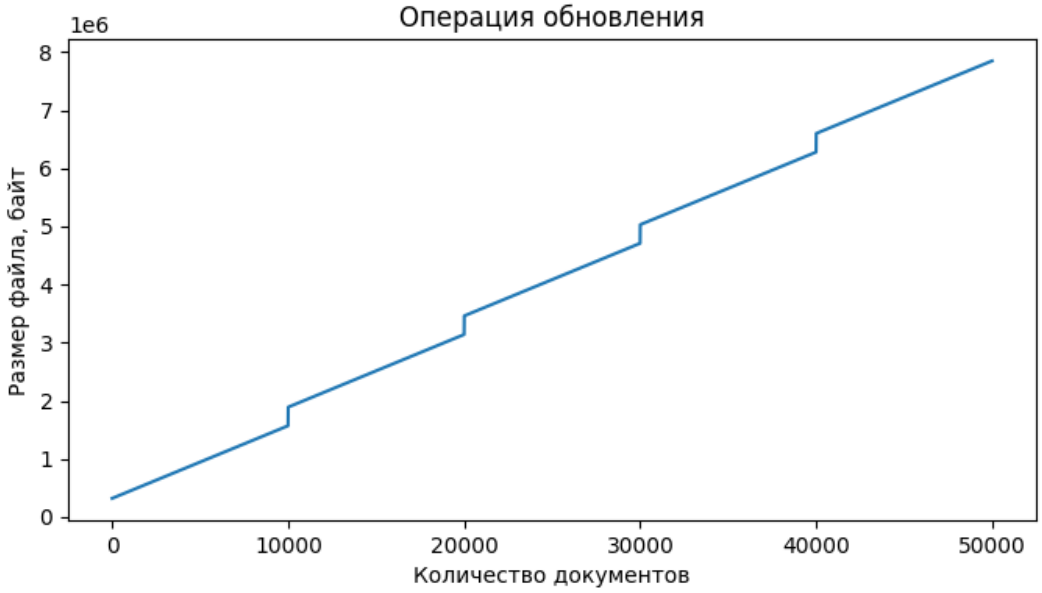
Операция удаления – по графику видно, что сложность алгоритма находится между O(n+m) и (n*m)



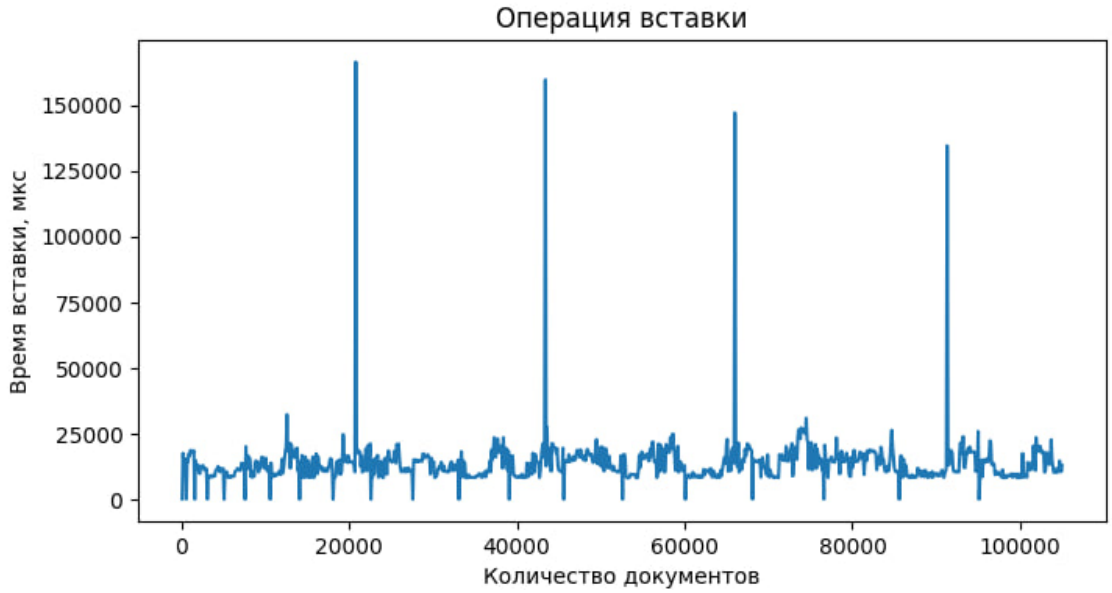
Операция обновления – выполняется за O(n), где n – количество строк таблицы.



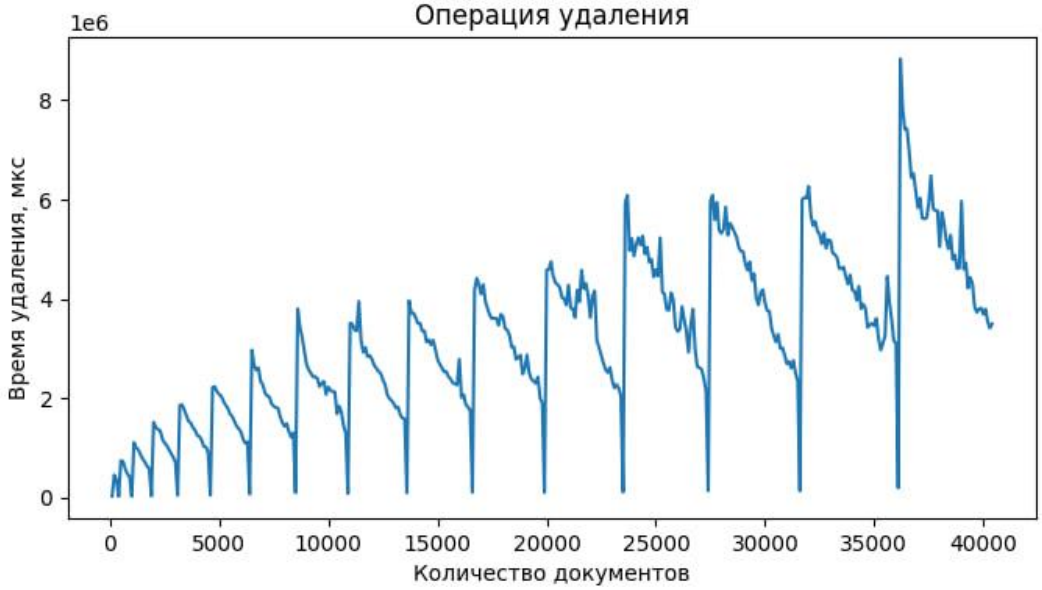
Зависимость размера файла от записанных документов – линейная:



Время каждый 100 операций вставки при последовательном добавлении 500*n строки и удалении 300* строк:



Время каждый 100 операций удаления при последовательном добавлении 500*n строки и удалении 300* строк:



Вывод:

В процессе выполнения лабораторной работы был разработан модель на языке Си, который занимается хранением в одном файле данных.