

Федеральное агентство по образованию Российской Федерации
Государственное образовательное учреждение
Высшего профессионального образования
Нижегородский государственный университет им. Н.И. Лобачевского
Институт информационных технологий математики механики

СОРТИРОВАННЫЕ ТАБЛИЦЫ

Отчет по лабораторной работе

Выполнил:

студент ИИТММ гр. 381903-3

Алилуев А.О. _____

Проверил:

ассистент каф. МОСТ, ИИТММ

Лебедев И.Г. _____.

Содержание

1.Введение	3
2.Постановка целей и задач	4
3.Руководство пользователя	5
4.Руководство программиста	7
4.1.Описание структуры программы	7
4.2.Описание структур данных	8
4.3.Описание алгоритмов	9
5.Эксперименты	11
6.Заключение	12
7.Литература	13

1. Введение

В предыдущей работе был разобран самый простой вид таблиц – просмотрные таблицы. Этот вид таблиц хорошо работает при небольших количествах элементов, так как операции удаления и поиска элементов происходят за линейное время. Когда элементов становится достаточно много, то ожидание выполнения операции становится существенным. Для решения подобных проблем были придуманы множества решений. Одно из таких – делать таблицу отсортированной, тогда можно применять двоичный поиск элементов, что в разы сокращает время ожидания.

2. Постановка целей и задач

Целью лабораторной работы является создание структуры хранения типа «Сортированная таблица» и методов работы с ним, таких как:

- Добавление элементов в таблицу;
- Удаление элементов из таблицы;
- Получение доступа к элементу.

Для реализации алгоритмов будет использоваться 3 класса:

- String;
- TElem;
- TSortTable.

Классы TElem и TSortTable являются шаблонными, и классы String и TElem уже были написаны для предыдущей работы.

Для проверки правильности работы этих классов будут написаны тесты с использованием фреймворка Google Test, а также тестовый образец программы, которая будет использует класс список.

3.Руководство пользователя

После запуска программы пользователя встречает консольное окно (рис. 1):

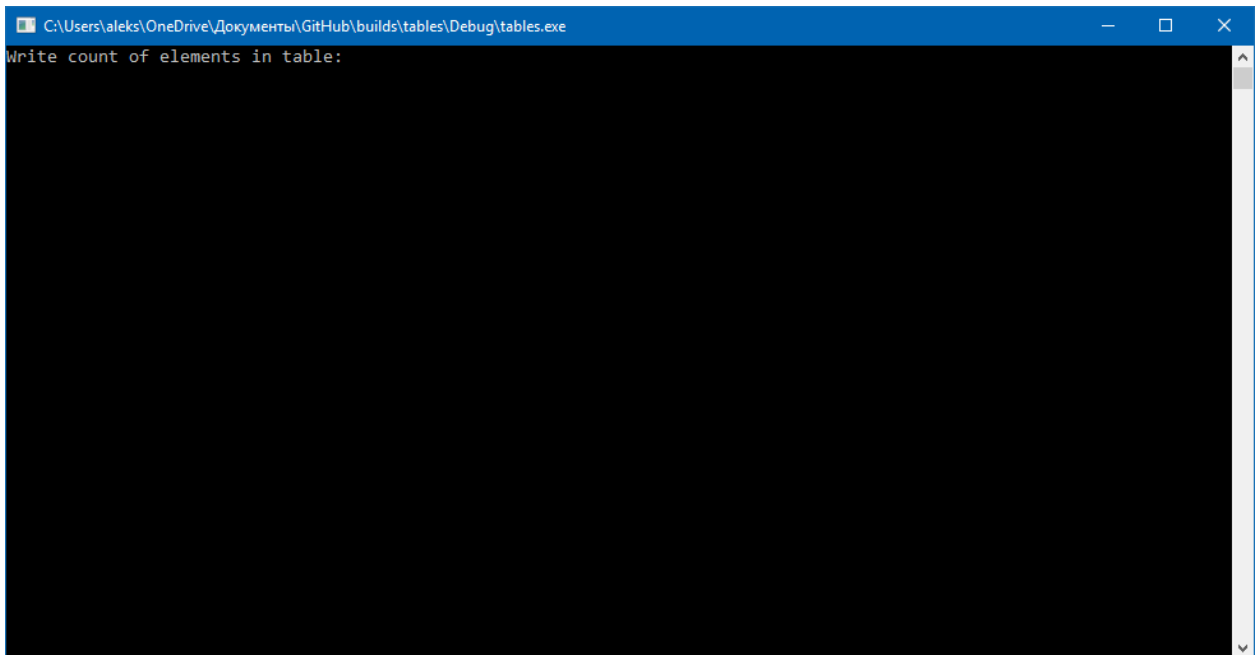


рис. 1 (вывод программы тестирования сортированных таблиц для пользователя)

в котором сначала от пользователя требуется ввести количество элементов в таблице, а затем заполнить поля этой таблицы (1 поле – ключ, 2 поле – целочисленное число, которое будет хранить эта ячейка) (рис. 2).

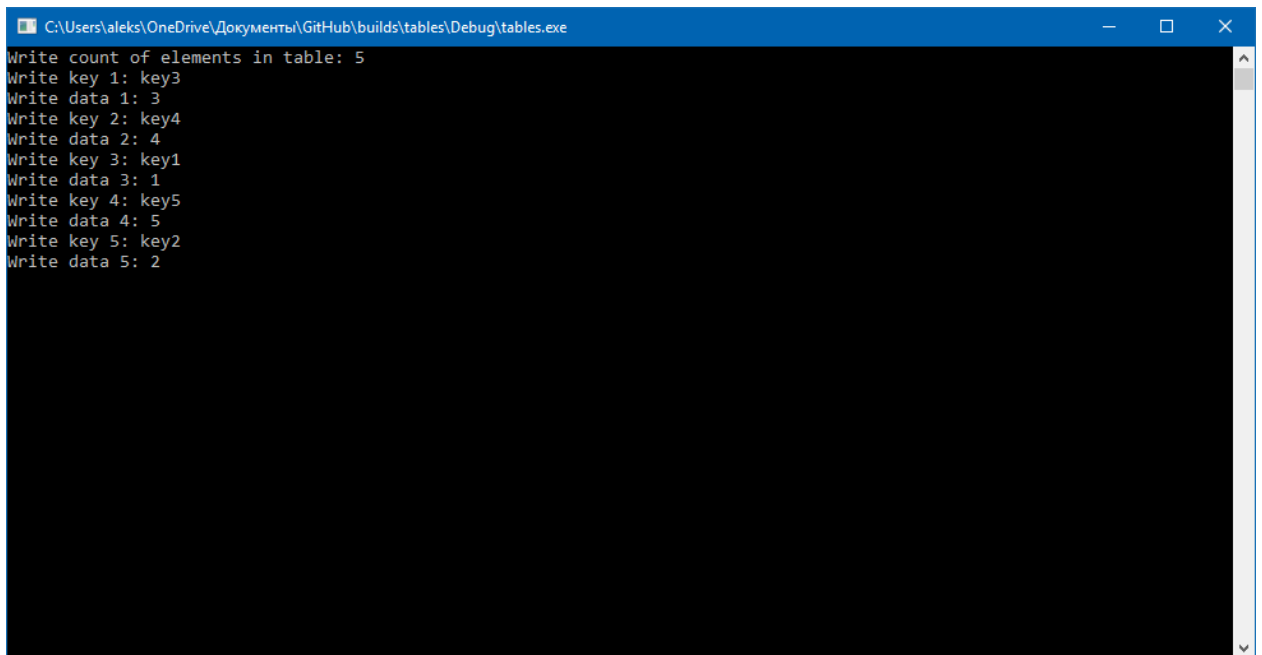
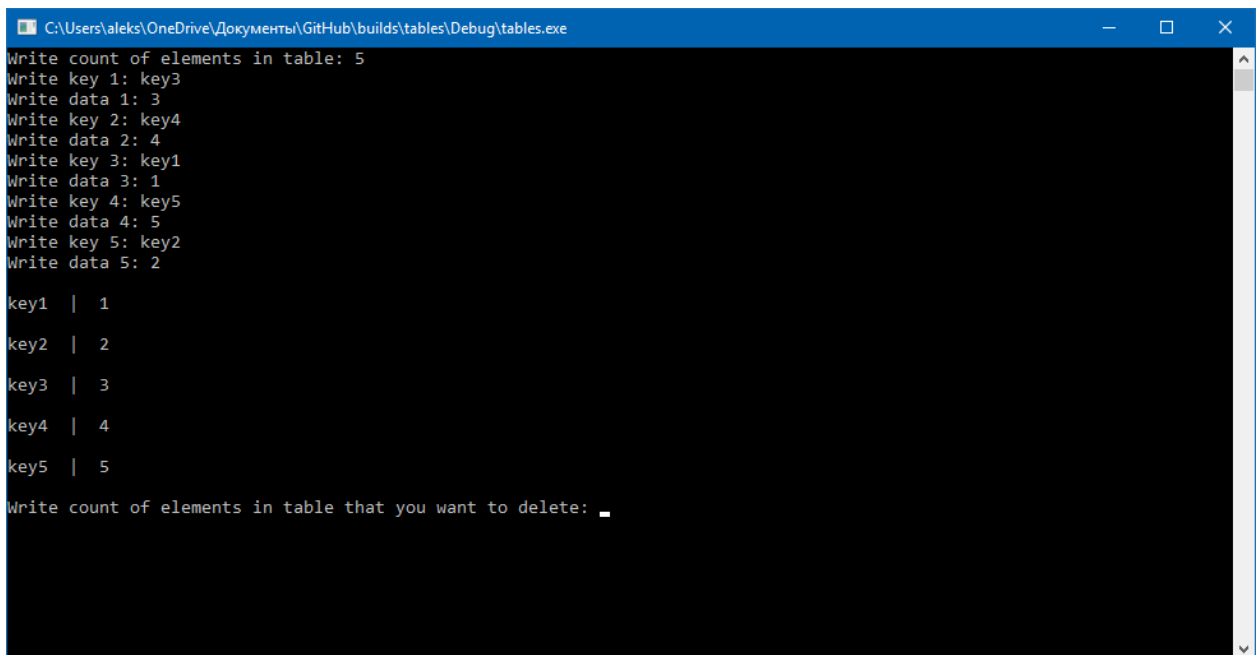


рис. 2 (заполнение полей таблицы)

Далее полученная таблица будет выведена для пользователя (обратите внимание, что поля таблицы будут отсортированы по возрастанию по ключам) и программа предложит удалить некоторое количество элементов, нужно ввести их количество (рис. 3).



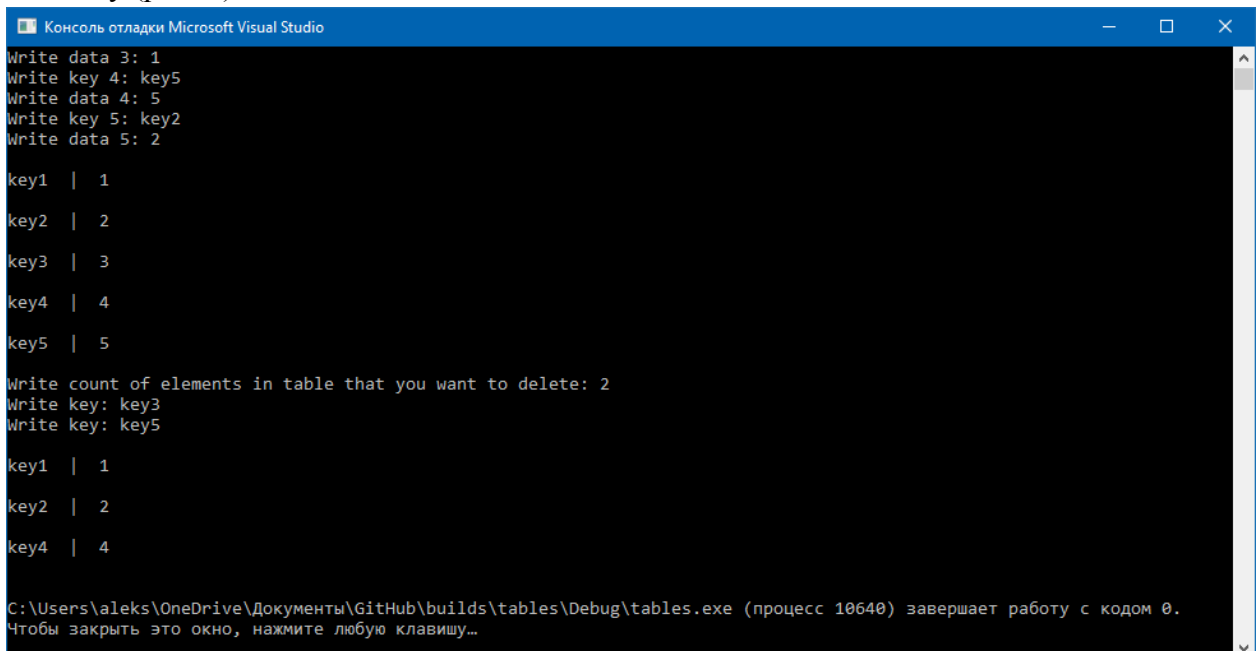
```
C:\Users\aleks\OneDrive\Документы\GitHub\builds\tables\Debug\tables.exe
Write count of elements in table: 5
Write key 1: key3
Write data 1: 3
Write key 2: key4
Write data 2: 4
Write key 3: key1
Write data 3: 1
Write key 4: key5
Write data 4: 5
Write key 5: key2
Write data 5: 2

key1 | 1
key2 | 2
key3 | 3
key4 | 4
key5 | 5

Write count of elements in table that you want to delete: _
```

рис. 3 (вывод таблицы на экран и запрос на ввод количества удаляемых элементов)

В конце программа попроси ввести ключи удаляемых элементов и выведет конечную таблицу (рис. 4).



```
Консоль отладки Microsoft Visual Studio
Write data 3: 1
Write key 4: key5
Write data 4: 5
Write key 5: key2
Write data 5: 2

key1 | 1
key2 | 2
key3 | 3
key4 | 4
key5 | 5

Write count of elements in table that you want to delete: 2
Write key: key3
Write key: key5

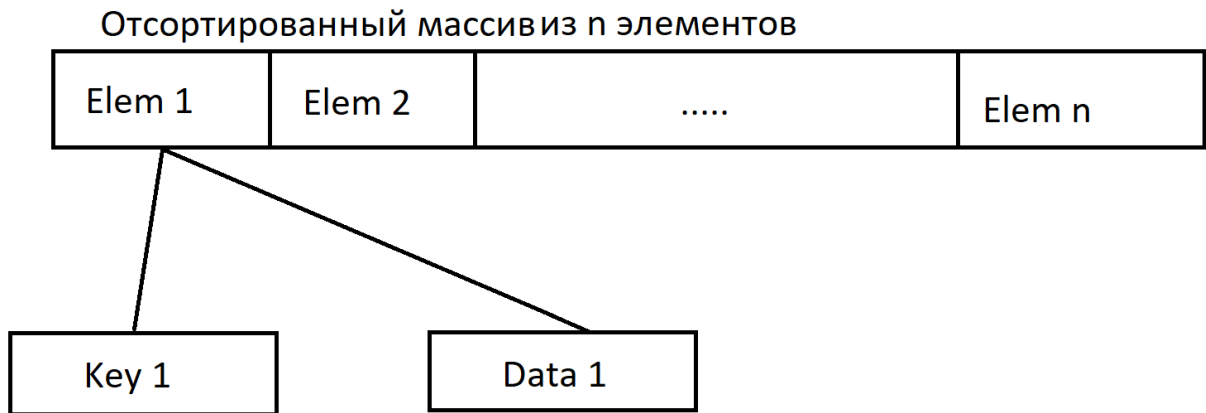
key1 | 1
key2 | 2
key4 | 4

C:\Users\aleks\OneDrive\Документы\GitHub\builds\tables\Debug\tables.exe (процесс 10640) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу...
```

рис. 4 (ввод ключей и удаление элементов таблицы)

4.Руководство программиста

4.1.Описание структуры программы



Сортированная таблица будет реализована как сортированный массив элементов таблицы, каждый элемент включает в себя ключ и значение, которое хранит этот элемент:

То есть для реализации алгоритмов будет использовано 3 класса:

- Класс «Строка» (String).
- Класс «Элемент Таблицы» (TElem), который будет использовать класс String.
- Класс «Таблица» (TSortTable), который использует класс TElem.

А также проект использующий фреймворк Google Test, для проверки правильности работы этих классов и тестовый проект, который будет показываться пользователю.

Класс String:

Класс строка реализует функции работы с массивом символов, такие как: сравнение, присвоение, доступ к элементам массива.

Класс TElem:

Класс элемент таблицы содержит реализацию работы с элементами. В нем реализованы такие методы, как: сравнение элементов, доступ к ключам и данным.

Класс TSortTable:

Класс сортированные таблицы содержит реализацию работы с таблицами. В нем реализованы такие методы, как: положить элемент в таблицу, удалить элемент из таблицы, получить значение по ключу и др.

Класс gtest:

Класс gtest реализует тестирование классов String, TElem и TSortTable, по средствам фреймворка Google Test. Тесты пишутся для каждого метода классов, каждого ветвления этих методов и для всех возможных исключений этих методов.

Проект table:

В данном проекте реализован примет использования таблиц, показанный пользователю.

4.2.Описание структур данных

Реализация классов String и TElem не изменилась с предыдущей работы, поэтому их описание пропустим.

Класс TSortTable:

template< class T > - шаблон класса T

static TElem<T> st – статический пустой элемент таблицы, которым первоначально заполняем таблицу;

TElem<T>* node – указатель на массив элементов таблицы;

int size – текущий размер таблицы;

int count – текущее количество занятых элементов таблицы.

Описание методов:

Метод:	Описание:
TElem<T> TSortTable<T>::st;	Инициализация статического поля класса
TSortTable<T>::TSortTable(const int _size)	Конструктор таблицы, который принимает максимальное количество элементов этой таблицы.
TSortTable<T>::TSortTable(const TSortTable<T>& table)	Конструктор копирования для сортированной таблицы.
TSortTable<T>::TSortTable(const TTable<T>& table, const int number_sort)	Конструктор, который принимает просмотрную таблицу и номер сортировки, а затем на их основе формирует сортированную таблицу.
TSortTable<T>::~~TSortTable()	Деструктор таблицы.
int TSortTable<T>::GetCount() const	Возвращает текущее количество элементов в таблице.

<code>bool TSortTable<T>::Add(TElem<T> & elem)</code>	Метод, который позволяет добавить элемент в таблицу, принимая ссылку на уже существующий элемент.
<code>String& TSortTable<T>::Add(const T& data)</code>	Метод, который позволяет добавить элемент в таблицу, принимая значение для этого элемента, а ключ будет сгенерирован автоматически.
<code>bool TSortTable<T>::Del(TElem<T>& elem)</code>	Метод, который позволяет удалить элемент из таблицы по копии элемента.
<code>bool TSortTable<T>::Del(const String& key)</code>	Метод, который позволяет удалить элемент из таблицы по ключу.
<code>TElem<T>& TSortTable<T>::Search(const String& key) const</code>	Метод, осуществляющий бинарный поиск элемента в таблице по его ключу.
<code>T& TSortTable<T>::operator[](const t String& key) const</code>	Перегрузка оператора индексации, который возвращает значение элемента по ключу.
<code>void TSortTable<T>::Expansion(const t int newsize)</code>	Метод, который позволяет увеличить максимальный размер таблицы.
<code>void TSortTable<T>::InsertSort(TTab le<T>& seetable)</code>	Метод сортировки вставками.
<code>void TSortTable<T>::MergeSort(TTa ble<T>& seetable, const int n, const int start)</code>	Метод сортировки слияниями.
<code>void TSortTable<T>::QuickSort(TTab le<T>& seetable, const int low, const int high)</code>	Метод быстрой сортировки.

4.3.Описание алгоритмов

Подробное описание некоторых методов

Добавление элемента по его копии:

- Проверка не проверка на то, является ли новый элемент первым в таблице, если да, то присваиваем значению элемента из массива значение параметра, увеличиваем количество элементов в таблице на 1 и возвращаем успех;
- Проверка на окончание места в таблице, если оно кончилось, то увеличиваем его в два раза;

- В цикле проходим по всем элементам, лежащим в таблице, если ключ пришедшего элемента больше текущего, то цикл переходит на следующую итерацию. Если меньше, то пришедший элемент встает на место большего, а все последующие элементы сдвигаются. Если пришедший элемент оказался самым большим, то он встает на последнее место в массиве;
- Количество элементов в массиве увеличивается и возвращается успех.

Удаление элемента таблицы по ключу:

- Заводятся переменные для бинарного поиска (левая, правая граница и текущий элемент);
- Пока расстояние между правой и левой границей больше 1, то:
 - Если ключ нужного элемента меньше текущего элемента, то смещаем область поиска на левую половину от текущей;
 - Если больше, то на правую половину;
 - Если текущий элемент совпал с нужным, то в цикле смещаем все последующие элементы массива влево на единицу, уменьшаем общее количество элементов на 1 и возвращаем успех;
- Если мы не нашли нужный элемент, то возвращаем неудачу.
-

5. Эксперименты

Сортировка	Количество сравнений
Insert sort	72
Merge sort	53
Quick sort	27

В экспериментах проверим сколько нужно операций сравнения для разных сортировок, таких как: сортировка вставками ($O(n^2)$), сортировка слияниями ($O(n \cdot \log(n))$) и быстрая сортировка ($O(n \cdot \log(n))$). Для проверки сортировок была сформирована таблица из 18 элементов, которые записаны в шахматном порядке для каждой сортировки.

Из результатов тестов можно узнать, что быстрее всего работает быстрая сортировка, затем сортировка слияниями и медленнее всех работает сортировка вставками.

6. Заключение

В заключении можно сказать, что у сортированных таблиц добавление элемента происходит дольше, чем у просмотрных, так как приходится проходить по всему массиву и сравнивать элементы, чтобы понять куда нужно ставить элемент, но при этом методы удаления у поиска элемента происходят значительно быстрее за счет возможности применения бинарного поиска. В работе реализован класс «TSortTable» с написанными методами добавления, удаления и доступа к элементам таблицы, а также написаны к ним тесты, и они успешно пройдены.

7.Литература

- Учебные материалы к учебному курсу «Методы программирования» - Гергель В.П.