Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

Высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий математики механики

Организация доступа по имени (таблицы)

Отчет по лабораторной работе

Выполнил:

студент ИИТММ гр. 381706-2

Антипин А.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

ассистент каф. МОСТ, ИИТММ

Лебедев И.Г\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Нижний Новгород

2019 г.

**Содержание**

[1.Введение 3](#_Toc8775501)

[2.Постановка целей и задач 4](#_Toc8775502)

[3.Руководство пользователя 5](#_Toc8775503)

[4.Руководство программиста 7](#_Toc8775504)

[4.1.Описание структуры программы 7](#_Toc8775505)

[4.2.Описание структур данных 8](#_Toc8775506)

[4.3.Описание алгоритмов 10](#_Toc8775507)

[5.Заключение 12](#_Toc8775508)

[6.Литература 13](#_Toc8775509)

# 1.Введение

Иногда для удобной организации доступа к памяти или элементам, которые лежат в этих ячейках памяти, прибегаю к организации доступа по имени элемента. Если приводить аналогию с массивом, то именем каждого элемента будет являться индекс этого элемента. Назовем такой способ доступа к данным табличным, а структуру хранения данных таблицей. Для реализации этой структуры данных нам необходим массив из элементов таблицы, которые в свою очередь состоят из хранимого элемента и его ключа. Существует множество реализаций таблиц, но в этой работе мы рассмотрим простейший их вид – просмотровые таблицы. Их особенность заключается в том, что у них нет особенностей и порядок хранения элементов определяется моментом их занесения в таблицу. Отсюда возникают закономерности, такие так поиск или удаление элемента из таблицы будет выполняться за линейное время. Тем не менее, такие таблицы достаточно легко преобразовать к сортированным, о которых речь пойдет в следующий раз.

# 2.Постановка целей и задач

Целью лабораторной работы является создание структуры хранения типа «Просмотровая 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;
* TSeeTable.

Классы TElem и TSeeTable являются шаблонными.

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

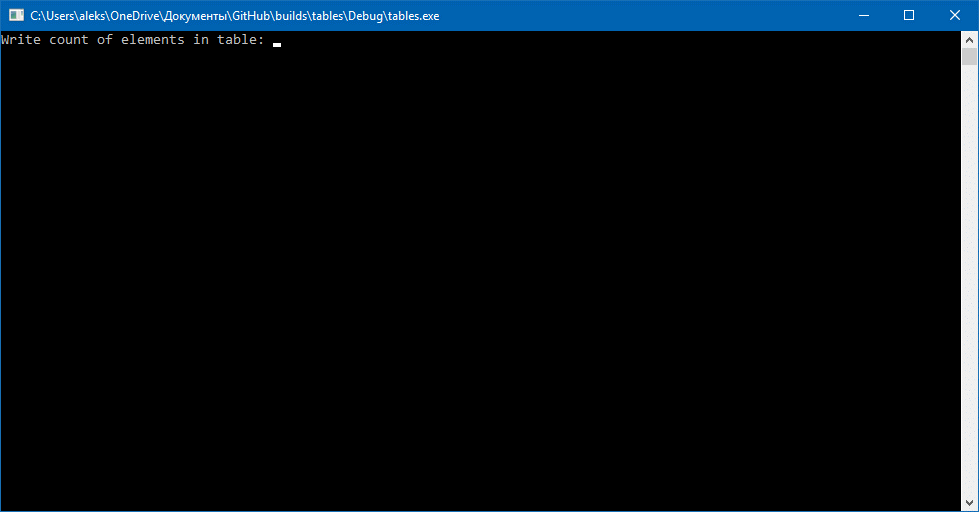
После запуска программы пользователя встречает консольное окно (рис. 1):

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

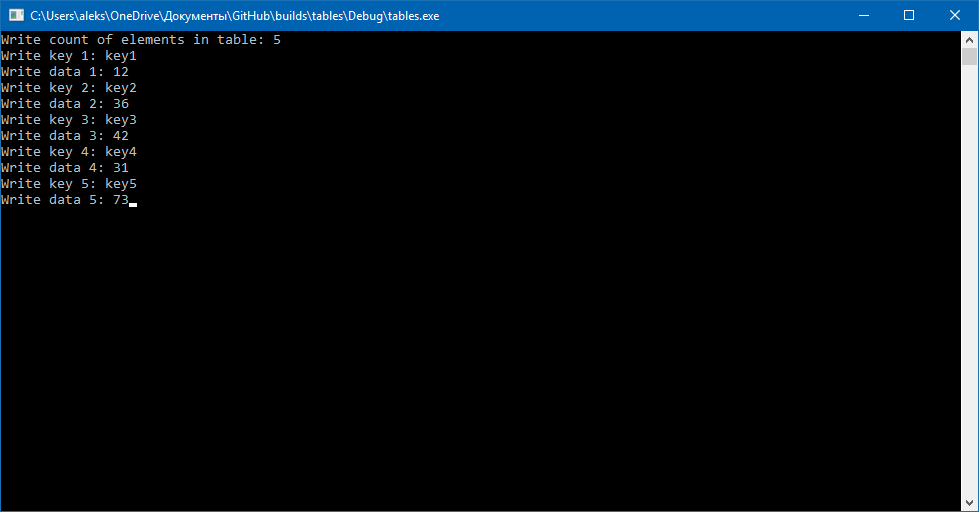
в которой сначала от пользователя требуется ввести количество элементов в таблице, а затем заполнить поля этой таблицы (1 поле – ключ, 2 поле – целочисленное число, которое будет хранить эта ячейка) (рис. 2).

рис. 2 (заполнение полей таблицы)

Далее полученная таблица будет выведена для пользователя и программа предложит удалить некоторое количество элементов, нужно ввести их количество (рис. 3).

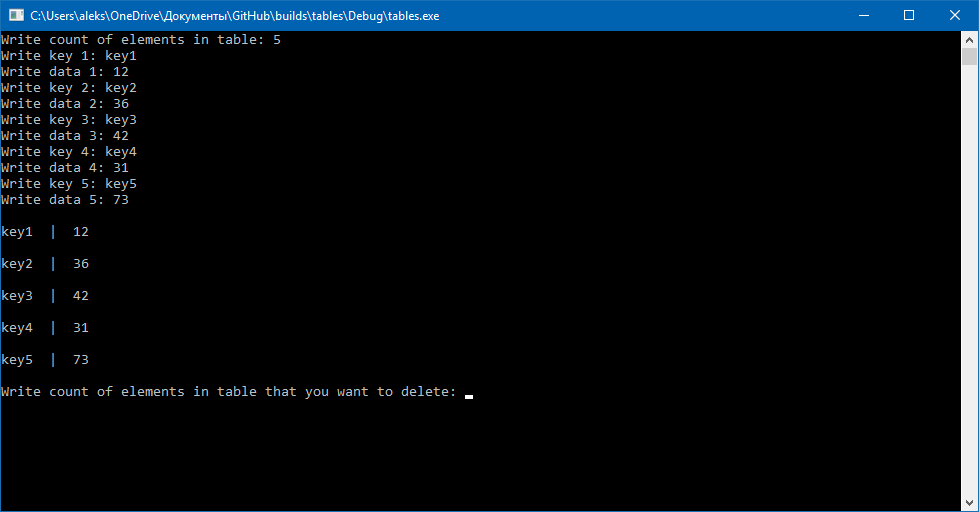
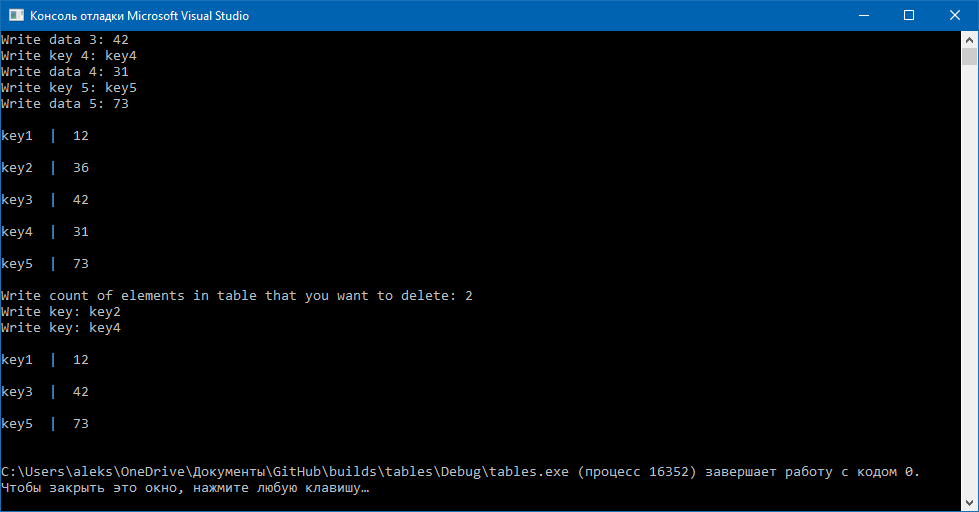
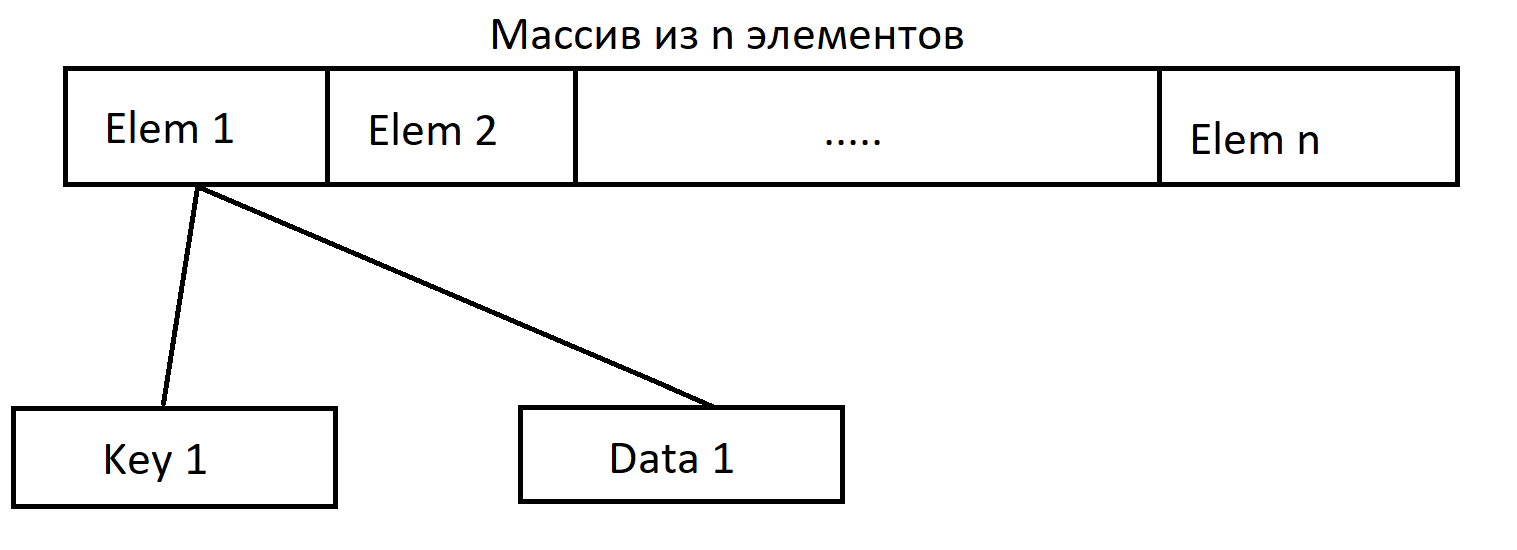
В конце программа попроси ввести ключи удаляемых элементов и выведет конечную таблицу (рис. 4).

рис. 4 (ввод ключей и удаление элементов таблицы)

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

# 4.Руководство программиста

## 4.1.Описание структуры программы



Просмотровая таблица будет реализована как 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» (TTable), который использует класс TElem.

А также проект использующий фреймворк Google Test, для проверки правильности работы этих классов и тесовый 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 TTable:**

Класс просмотровые 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 TTable, 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

**Класс String:**

char\* str – указатель на массив для хранения строки;

int count – переменная для хранения числа элементов в строке.

**Класс TElem:**

template< class T > - шаблон класса Т

T data – хранимое значение элемента;

String key – ключ элемента.

**Класс TTable:**

template< class T > - шаблон класса Т

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: |
| String::String() | Конструктор по умолчанию для класса String. |
| String::String(const char\* \_str) | Конструктор с параметром для класса String, который принимает указатель на массив символов. |
| String::String(const String& \_str) | Конструктор копирования класса String. |
| String::~String() | Деструктор класса String. |
| int String::GetCount() const | Метод, который возвращает количество символов в строке. |
| String& String::operator=(const String& \_str) | Перегрузка оператора присваивания. |
| bool String::operator==(const String& \_str) const | Перегрузка оператора равно для строки. |
| bool String::operator!=(const String& \_str) const | Перегрузка оператора не равно для строки. |
| bool String::operator<(const String& \_str) const | Перегрузка оператора меньше для строки. |
| bool String::operator>(const String& \_str) const | Перегрузка оператора больше для строки. |
| char& String::operator[](const int pos) const | Возвращение элемента массива символов по его номеру. |
| TElem<T>::TElem() | Конструктор по умолчанию для элемента таблицы. |
| TElem<T>::TElem(const T& \_data, const String& \_key) | Конструктор с параметрами, который принимает хранимый элемент и его ключ. |
| TElem<T>::~TElem() | Деструктор для элемента таблицы. |
| TElem<T>& TElem<T>::operator=(const TElem<T>& elem) | Перегрузка оператора присваивания для элемента. |
| bool TElem<T>::operator==(const TElem<T>& elem) | Перегрузка оператора равно для элемента таблица. |
| bool TElem<T>::operator!=(const TElem<T>& elem) | Перегрузка оператора не равно для элемента таблицы. |
| T& TElem<T>::GetData() | Возвращает значение, которое хранится в элементе таблицы. |
| void TElem<T>::SetData(const T& \_date) | Метод, который позволяет изменить значение, которое хранится в элементе таблицы. |
| String& TElem<T>::GetKey() | Возвращает ключ элемента таблицы. |
| void TElem<T>::SetKey(const String& \_key) | Позволяет изменить ключ элемента таблицы. |
| TElem<T> TTable<T>::st; | Инициализация статического поля класса |
| TTable<T>::TTable(const int \_size) | Конструктор таблицы, который принимает максимальное количество элементов этой таблицы. |
| TTable<T>::TTable(const TTable<T>& table) | Конструктор копирования для просмотровой таблицы. |
| TTable<T>::~TTable() | Деструктор таблицы. |
| int TTable<T>::GetCount() const | Возвращает текущее количество элементов в таблице. |
| bool TTable<T>::Add(const TElem<T>& elem) | Метод, который позволяет добавить элемент в таблицу, принимая ссылку на уже существующий элемент. |
| String& TTable<T>::Add(const T& \_data) | Метод, который позволяет добавить элемент в таблицу, принимая значение для этого элемента, а ключ будет сгенерирован автоматически. |
| bool TTable<T>::Del(TElem<T>& elem) | Метод, который позволяет удалить элемент из таблицы по такому же элементу. |
| bool TTable<T>::Del(const String& \_key) | Метод, который позволяет удалить элемент из таблицы по ключу. |
| TElem<T>& TTable<T>::Search(const String& \_key) const | Метод, осуществляющий поиск элемента в таблице по его ключу. |
| T& TTable<T>::operator[](const String& \_key) const | Метод, который возвращает значение элемента по ключу. |
| void TTable<T>::Expansion(const int newsize) | Метод, который позволяет увеличить максимальный размер таблицы. |

## 4.3.Описание алгоритмов

**Подробное описание некоторых методов**

Добавление элемента по значению:

* Проверка не закончилось ли место в таблице, если да, то расширяем ее в 2 раза;
* Кладем значение элемента в нужный элемент массива и создаем строку со стандартным именем;
* Если это первый элемент массива, то присваиваем ключу этого элемента значение стандартного имени, увеличиваем количество элементов в таблице и возвращаем ключ.
* Если нет, строка со стандартным ключом переходит в строку из одного символа – первой буквы предыдущего элемента.

Удаление элемента таблицы по ключу:

* Проверка на пустоту таблицы, если пуста, то возвращаем false;
* Проходим в цикле по всем элементам таблицы, и если находим нудный, то сдвигаем все оставшиеся элементы таблицы на 1 вперед, уменьшаем количество элементов на 1 и возвращаем true;
* Если элемент не нашли, то возвращаем false.

# 5.Заключение

В заключении можно сказать, что все поставленные цели и задачи были выполнены, а именно: созданы классы «String», «TElem» и «TTable» с реализованными методами добавления, удаления и доступа к элементам таблицы, а также написаны к ним тесты, и они успешно пройдены.

# 6.Литература

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