МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**Отчет по учебной практике**

**«Организация сортированной таблицы»**

**Выполнил:** студент группы 381706-1

Нечаева Екатерина Владимировна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г

Нижний Новгород

2019.

Содержание

[1. Введение 3](#_Toc9324344)

[2. Постановка задачи 4](#_Toc9324345)

[3. Руководство пользователя 5](#_Toc9324346)

[4. Руководство программиста 6](#_Toc9324347)

[4.1 Описание структуры программы 6](#_Toc9324348)

[4.2 Описание структур данных 6](#_Toc9324349)

[4.3 Описание алгоритмов 7](#_Toc9324350)

[5. Заключение 9](#_Toc9324351)

[6. Литература 10](#_Toc9324352)

# Введение

Таблица (от лат. tabula – доска) – динамическая структура данных, базисным множеством которой является семейство линейных структур из записей (базисное отношение включения определяется операциями вставки и удаления записей).

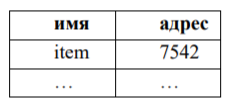
Эффективность процедуры поиска можно повысить при размещении записей в таблице в порядке возрастания (или убывания) ключей (упорядоченная, или сортированная таблица). Упорядоченность таблиц может быть организована только при возможности сравнения ключей (на множестве ключей задано отношение линейного порядка). Для поиска нужной записи в таких таблицах может быть использован быстрый метод бинарного (двоичного) поиска.

Двоичный (бинарный) поиск (также известен как метод деления пополам и дихотомия) — классический алгоритм поиска элемента в отсортированном массиве (векторе), использующий дробление массива на половины.

Сортированная таблица – это упорядоченная таблица.

Запись – кортеж, каждый элемент которого обычно именуется полем.

Имя записи (ключ) – одно из полей записи, по которому обычно осуществляется поиск записей в таблице; остальные поля образуют тело записи.

**Первое поле всех записей таблицы является ключом, поля записи без ключевого поля образуют тело записи. Например, задавая соответствие между идентификаторами переменных (именами) и адресами памяти ЭВМ, мы можем построить простейшую таблицу вида

*Рисунок 1.Пример таблицы.*

# Постановка задачи

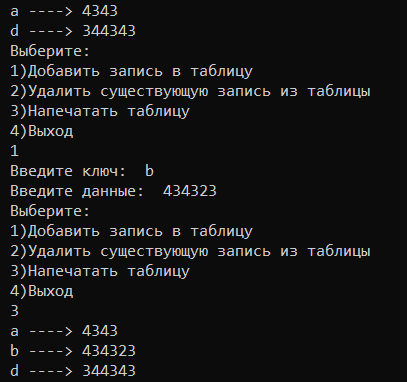
В рамках лабораторной работы ставится задача реализовать динамическую структуру данных - таблицу и базовые операции над ней:

* Поиск записи
* Вставка записи
* Удаление записи

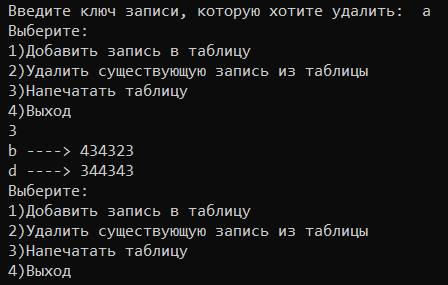
# Руководство пользователя

В начале запуска программы пользователю предлагается ввести размер таблицы. После чего перед ним появляется меню, в котором можно сделать несколько действий: добавить/удалить запись, вывести таблицу на экран и выход.

Сначала пользователь добавляет в существующую таблицу запись и вывод ее на экран. Так как таблица сортировочная, то эта запись будет стоять в порядке возрастания ключей.

*  
Рисунок 2. Пример добавления записи в сортировочную таблицу.*

После некоторых действий над таблицей, пользователь может удалить ненужную запись и вывести на консоль измененную таблицу.

  
*Рисунок 3. Пример удаления записи.*

# Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из следующих модулей:

* Модуль sorttable. Содержит пример использования таблицы (mainsorttable.cpp)
* Модуль libsorttable – статическая библиотека. Содержит заголовочные файлы sortelem.h и sorttable.h, в первом описана одна запись таблицы(элемент), во втором сама сортировочная таблица.
* Модуль sorttableTest. Содержит тесты, проверяющие правильность работы методов (elemTest.cpp, sorttableTest.cpp)
* Модуль libthrows – библиотека исключений. Содержит заголовочный файл throws.h, в котором он реализован.

## Описание структур данных

#### Класс TException – класс исключений.

Содержит одно защищенное поле *string str,* переменная строкового типа, которая хранит сообщение об исключении.

Содержит публичные методы

* TException(std::string \_str); //конструктор-инициализатор
* void Print(); //вывод данного исключения на консоль

#### Класс TElem.

Защищенные поля:

* string key; //ключ
* T data; //данные

Содержит публичные методы:

* TElem(); //конструктор по умолчанию
* TElem(string \_key, T \_data); //конструктор инициализатор
* TElem(TElem<T> &A); //конструктор копирования
* void SetKey(string \_key); //установить ключ
* void SetData(T \_data); //установить данные
* string GetKey(); //получить ключ
* T& GetData(); //получить данные
* TElem<T>& operator=(TElem<T> &A); //присваивание
* bool operator==(TElem<T> &A); //сравнение на равенство
* bool operator!=(TElem<T> &A); //сравнение на не равенство

#### Класс TSortTable.

Защищенные поля:

* TElem<T> not\_find;
* TElem<T>\* mas; //массив записей
* int size; //размер таблицы
* int count; //количество занятых записей в таблице

Содержит публичные методы:

* TSortTable(string\* \_key, T\* \_data, int \_size = 3); //конструктор иницализатор
* TSortTable(int \_size = 10); //конструктор инициализатор
* TSortTable(TSortTable<T> &A); //конструктор копирования
* int Put(string \_key, T \_data); //добавление записи
* void Del(string \_key); //удаление записи
* TElem<T>& Search(string k); //поиск записи по ключу
* void Resize(int \_size); //изменение размера таблицы
* T& operator[](string k); //индексация
* int GetCount(); //получить количество занятых записей в таблице
* int GetSize(); //получить размер таблицы
* void Print(); //вывод таблицы

## Описание алгоритмов

**Добавление записи в таблицу.**

На вход подается переменная строкового типа, которая обозначает ключ таблицы, и сами данные.

Если вставка записи происходит в уже заполненную таблицу, вызывается метод перепаковки таблицы, т.е увеличение размера таблицы вдвое. Иначе, с помощью бинарного поиска ищем место в таблице, куда добавить запись с данным ключом. Увеличиваем количество занятых записей на единицу.

**Поиск записи в таблице.**

На вход подается переменная строкового типа-ключ, по которому мы будем искать нужную запись. С помощью бинарного поиска ищем эту запись по ключу.

Если она нашлась, возвращаем эту запись. Иначе, возвращаем специальный элемент, который означает пустую запись.

**Удаление записи из таблицы.**

На вход подается переменная строкового типа-ключ, по которому мы будем искать нужную запись. С помощью бинарного поиска ищем нужную запись, если она нашлась, смещаем массив записей, начиная с найденной, вверх на единицу. Тем самым удаляем нужную запись. Уменьшаем количество занятых записей на единицу.

# Заключение

В ходе выполнения данной лабораторной была реализована динамическая структура данных - таблица. Если сравнивать сложности операций над сортированной и просмотровой таблицами, будет видно, что операции над сортированными таблицами работают быстрее, чем над просмотровыми. Были реализованы методы для работы с таблицей, а именно: вставка записи, удаление записи и поиск записи в таблице.

Были разработаны тесты, проверяющие корректную работу методов данных классов, с помощью Google C++ Testing Framework.

# Литература

1. Учебно-методическое пособие [Электронный ресурс]

<http://www.unn.ru/books/met_files/Pract_ADS.pdf>

1. Организация доступа по имени [Электронный ресурс]

<https://www.studsell.com/view/147836/>

1. Двоичный поиск [Электронный ресурс]

<https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA>