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**

**«Структура данных:**

**таблица с использованием АВЛ-деревьев»**

**Выполнил:** студент группы 381706-1

Соболева Ю.А.

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

аспирант каф. МОСТ ИИТММ

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

Нижний Новгород

2019.

Содержание

[1. Введение 3](#_Toc533027637)

[2. Постановка задачи 4](#_Toc533027638)

[3. Руководство пользователя 5](#_Toc533027639)

[4. Руководство программиста 6](#_Toc533027640)

4.[1. Описание структуры программы 6](#_Toc533027641)

4.[2. Описание структур данных 6](#_Toc533027642)

4.[3. Описание алгоритмов 7](#_Toc533027643)

[5. Заключение 8](#_Toc533027644)

[6. Литература 9](#_Toc533027645)

# **Введение**

**Структура данных** - программная единица, которая определяет метод хранения и обработки различных логически связанных данных в вычислительной технике. Знание структур данных позволяет наиболее компактно и практично расположить данные в памяти компьютера. В данной работе мы рассмотрим такую структуру данных, как **таблица на АВЛ-деревьях**.

**Таблица** –  это совокупность связанных данных, хранящихся в структурированном виде. Таблица использует модель вертикальных столбцов (имеющих уникальное имя) и горизонтальных строк. Ячейка — место, где строка и столбец пересекаются. Таблица содержит определенное число столбцов, но может иметь любое количество строк. Каждая строка однозначно определяется одним или несколькими уникальными значениями, которые принимают её ячейки из определенного подмножества столбцов. Подмножество столбцов, которое уникально идентифицирует строку, называется [первичным ключом](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B2%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87). Программная реализация таблиц возможна на основе различных структур данных, например, с использованием статических или динамических одномерных массивов, линейных списков или деревьев.

**АВЛ-дерево** — сбалансированное по высоте [двоичное дерево поиска](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0): для каждой его вершины высота её двух поддеревьев различается не более чем на 1.

**Целью данной лабораторной работы** является разработка структуры данных для хранения таблиц с использованием АВЛ-дереьев, а также освоение таких инструментов разработки программного обеспечения, как система контроля версий Git и фрэймворк для разработки автоматических тестов Google Test.

# **Постановка задачи**

Выполнение работы предполагает решение следующих задач:

1. Реализация класса таблицы TTreeTab.
2. Реализация класса элемента таблицы TTElem.
3. Разработка интерфейса для данных классов.
4. Обеспечение работоспособности примера использования.
5. Реализация нескольких тестов на базе Google Test.

# **Руководство пользователя**

Пример использования таблицы ):

* В начале программы создаются пять элементов с различными значениями.
* Далее происходит внесение всех этих элементов в таблицу (кроме пустого).
* Затем программа выполняет удаление одного из элементов.
* После, реализуется метод поиска элемента по ключу.
* В конце программа находит высоту дерева, на котором построена таблица. Результаты выводится на экран.

# **Руководство программиста**

## **Описание структуры программы**

**Проект “TreeTab ” состоит из следующих файлов:**

1. mainTreeTab.cpp (в нём находится main)

**Проект “ TreeTabLib ” состоит из следующих файлов:**

1. TreeTabLib.h (описание класса “TTreeTab” реализация методов класса “TTreeTab” и перегрузка операций )
2. TreeTabLib.cpp
3. TTElenLib.h (описание класса “TTElem” реализация методов класса “TTElem” и перегрузка операций )

**Проект “ TreeTabLibTest” состоит из следующих файлов:**

1. test\_treetablib.cpp (реализация тестов для класса TTreeTab)
2. test\_main.cpp

## **Описание структур данных**

***Класс TTElem:***

*Поля:*

* string key; - ключ элемента;
* T data; - данные элемента;
* TTElem<T> \*left, \*right, \*parent; - указатели на меньший, больший элемент и на предка.

*Конструкторы и деструктор:*

* TTElem (string \_key = “”, T \_data=0); - конструктор инициализации;
* TTElem(TTElem<T> &A); - конструктор копирования;

*Перегруженные операторы:*

* TTElem& operator = (TElem &A); - оператор присваивания;
* bool operator == (TElem &A); - оператор равно;
* bool operator != (TElem &A);- оператор неравно;

***Класс TTreeTab:***

*Поля:*

* TTElem<T>\* root; - указатель на корень;
* int count; - количество элементов;

*Конструкторы и деструктор:*

* TTreeTab (); - по умолчанию;
* TTreeTab (int \_size, string\* \_key, int\* \_data); - приведения типа;
* TTreeTab (TTreeTab& A); - копирования;

*Методы:*

* int Height(TTElem<T>\* k); - количество потоков у элемента.
* void ChangeRoot(); - балансировка дерева;
* void Put (string \_key, T \_data); - положить элемент по ключу и данным;
* void Put (TTElem<T> &A); - положить элемент;
* void Del (string \_key); - удалить по ключу;
* TTElem<T>\* Search(string \_key);
* bool IsEmpty(); - проверка таблицы на пустоту;

## **Описание алгоритмов**

Метод **Height:**

Если у элемента нет потомков возвращаем -1. Иначе если существует левый потомок, прибавляем к левой высоте (найденной рекурсивно) единицу. Аналогично поступаем с правым потомком. Метод возвращает большую из левой и правой высот.

Метод **ChangeRoot:**

Используются 4 типа вращений:

* Большое левое вращение



Данное вращение используется тогда, когда (высота b-поддерева — высота L) = 2 и высота c-поддерева > высота R.

* Малое правое вращение



Данное вращение используется тогда, когда (высота b-поддерева — высота R) = 2 и высота С <= высота L.

* Большое правое вращение



Данное вращение используется тогда, когда (высота b-поддерева — высота R) = 2 и высота c-поддерева > высота L.

* Малое левое вращение.

Метод **Put**:

Процесс включения вершины состоит из трех частей

1. Прохода по пути поиска, пока не убедимся, что ключа в дереве нет.
2. Включения новой вершины в дерево и определения результирующих показателей балансировки.
3. «Отступления» назад по пути поиска и проверки в каждой вершине показателя сбалансированности. Если необходимо — балансировка.

Метод **Del:**

Для простоты опишем рекурсивный алгоритм удаления. Если вершина — лист, то удалим её и вызовем балансировку всех её предков в порядке от родителя к корню. Иначе найдём самую близкую по значению вершину в поддереве наибольшей высоты (правом или левом) и переместим её на место удаляемой вершины, при этом вызвав процедуру её удаления.

# **Заключение**

В ходе выполнения лабораторной работы я смогла реализовать такую структуру данных, как стэк. Вместе с ней, согласно заданному интерфейсу, я реализовала класс TTreeTab и TTElem. Написание нескольких своих тестов, помогло мне разобраться с системой автоматических тестов Google Test.

В результате проделанной работы у меня получилось

1. Реализовать класс таблицы TTreeTab.

2. Реализовать класс элемента таблицы TTElem.

2. Разработать интерфейс для данных классов.

3. Обеспечить работоспособность примера использования.

4. Реализовать некоторые тесты на базе Google Test.

Таким образом, данная лабораторная работа отвечает всем поставленным задачам. Это означает, что цель работы была достигнута.

# **Литература**

1. Лабораторный практикум. Составители: Барышева И.В., Мееров И.Б., Сысоев А.В., Шестакова Н.В. Под редакцией Гергеля В.П. Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2017. – 105с.

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

1. Национальный открытый университет «Интуит». Курс «Основы программирования». Лекция 11: Структуры данных: общее понятие, реализация. Простейшие структуры данных: очередь, стек. Использование стека и обратная польская запись.

URL: <https://www.intuit.ru/studies/courses/2193/67/lecture/1980?page=3>