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 хранения данных: упорядоченные таблицы»**

**Выполнил:** студент группы 381706-1

Шашкин Евгений Вадимович

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2019.

**Оглавление**

[1. Введение. 2](#_Toc8237860)

[2. Постановка задачи. 3](#_Toc8237861)

[3. Руководство пользователя. 4](#_Toc8237862)

[4. Руководство программиста. 5](#_Toc8237863)

[4.1. Описание структуры программы. 5](#_Toc8237864)

[4.2. Описание структур данных. 5](#_Toc8237865)

[4.3. Описание алгоритмов. 6](#_Toc8237866)

[5. Эксперименты. 7](#_Toc8237867)

[6. Заключение. 8](#_Toc8237868)

[7. Литература. 9](#_Toc8237869)

# Введение.

**Таблица** — способ структурирования данных. Представляет собой распределение данных по однотипным строкам и столбцам.

**Упорядоченная таблица** –этоодномерный массив элементов с заранее известным размером. Т.к. данные таблицы упорядочены, то поиск элементов, добавление элемента и удаление элемента из таблицы имеют сложность .

# Постановка задачи.

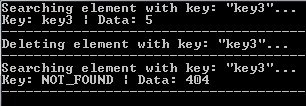
В данной лабораторной работе нужно разработать эффективную библиотеку для хранения и работы с такой структурой данных, как упорядоченная таблица.

Для этого нам нужно:

* Описать и реализовать класс элемента таблицы **TSortElem**.
* Описать и реализовать класс просматриваемой таблицы **TSortTable**.
* Протестировать класс **TSortTable** с помощью Google Test.
* Реализовать класс **TMyException** для обработки исключений, которые могут возникнуть в результате работы класса **TSortTable**.
* Продемонстрировать работу класса **TSortTable**.

# Руководство пользователя.

Пример работы класса **TSortTable**:

****

*Рис 1. Пример работы программы*

Программа работает следующим образом:

* Создается таблица.
* Таблица заполняется.
* Производится поиск элемента таблицы с ключом «key3».
* Найденный элемент выводится на экран.
* Производится удаление элемента из таблицы с ключом «key3».
* Производится повторный поиск элемента таблицы с ключом «key3».
* На экран выводится константа, которая означает, что элемент не найден.

# Руководство программиста.

# Описание структуры программы.

Программа состоит из модулей:

* **SortTable** – содержит в себе файл **sort\_table\_main.cpp** с реализацией примера использования класса **TSortTable**.
* **SortTableLib –** содержит в себе файлы **SortTable.h** и **SortTableElem.h**, в которых описаны и реализованы классы **TSortTable** и **TSortElem** соответственно.
* **SortTableTest** – содержит в себе файл **test\_sort\_table.cpp**, в котором находится набор тестов, для проверки работоспособности классов **TSortElem** и **TSortTable**.
* **MyExceptionLib** –содержитв себе файл **MyException.h** с реализацией класса исключений **TMyException**.

# Описание структур данных.

**Класс TSortElem.**

Класс **TSortElem** является шаблонным классом и содержит два поля со спецификатором **protected**:

* **string key** – строка-ключ для доступа к элементам таблицы.
* **T data** – данные, содержащиеся в элементе таблицы.

Далее идут методы класса со спецификатором доступа **public**:

* **TSortElem(string \_key, T \_data)** – конструктор-инициализатор.
* **TSortElem(TSortElem &obj)** – конструктор копирования.
* **TSortElem<T>& operator=(TSOrtElem &obj)** – оператор присваивания.
* **bool operator==(TElem &obj)** – оператор сравнения.
* **bool operator>(TSortElem<T> &obj)** – оператор «больше».
* **bool operator<(TSortElem<T> &obj)** – оператор «меньше».
* **void SetKey(string \_key)** – метод, устанавливающий ключ.
* **string GetKey()** – метод, возвращающий ключ.
* **void SetData(T \_data)** – метод, устанавливающий данные.
* **T GetData()** – метод, возвращающий данные.
* **T& GetDataAddress()** – метод, возвращающий адрес данных.

**Класс TSortTable.**

Класс **TSortTable** является шаблонным классом и содержит четыре поля со спецификатором **protected**:

* **TSortElem<T> notFound** – специальная константа, которая означает, что элемент не найден.
* **TSortElem<T> \*mas** – массив элементов таблицы.
* **int size** – размер таблицы.
* **int count** – количество элементов в таблице.

Далее идут методы класса со спецификатором доступа **public**:

* **TSortTable(string\* \_keys, T\* \_data, int \_size)** – конструктор-инициализатор.
* **TSortTable(int \_size)** – конструктор по умолчанию.
* **TSortTable(TSortTable &obj)** – конструктор копирования.
* **void Put(string \_key, T \_data)** – метод, помещающий элемент в таблицу.
* **void Del(string \_key)** – метод, удаляющий элемент из таблицы.
* **TSortElem<T>& Search(string \_key)** – метод поиска элемента в таблице.
* **T& operator[](string \_key)** – оператор доступа к элементам таблицы по ключу.
* **void SetSize(int \_size)** – метод, устанавливающий размер таблицы.
* **int GetSize()** – метод, возвращающий размер таблицы.
* **void SetCount(int \_count)** – метод, устанавливающий количество элементов в таблице.
* **int GetCount()** – метод, возвращающий количество элементов в таблице.

**Класс TMyException.**

Класс **TMyException** содержит одно поле со спецификатором доступа **private**:

* **string str** – строка, хранящая сообщение об ошибке.

Далее идут методы класса со спецификатором доступа **public**:

* **TMyException(std::string \_str)** – конструктор инициализатор.
* **void what()** – метод вывода ошибки на экран.

# Описание алгоритмов.

**Добавление элемента в таблицу.**

Сначала проверяем таблицу на полноту. Если в ней нет свободного места, то с помощью метода **Resize(int \_size)** увеличиваем размер таблицы вдвое. Затем, с помощью бинарного поиска ищем место, куда можно поместить новый элемент так, чтобы таблица оставалась упорядоченной. После того, как мы нашли нужное место, сдвигаем элементы таблицы, стоящие после найденного места, на одно место вправо. И наконец помещаем новый элемент в найденное место.

**Удаление элемента из таблицы.**

Сначала с помощью бинарного поиска ищем нужный элемент. После того как мы нашли элемент, мы сдвигаем элементы таблицы, стоящие после найденного элемента, на одно место влево.

**Поиск элемента в таблице.**

Поиск выполняется по ключу. Совершается обход таблицы с помощью бинарного поиска. Если элемент с искомым ключом найден, то он возвращается. Если нет, то возвращается константа **notFound**.

# Эксперименты.

Параметры ПК:

* Операционная система: Windows 7 Ultimate.
* Процессор: AMD Athlon™ II X4 635 Processor 2.90 GHz.
* Версия Visual Studio: 2017.

|  |  |  |  |
| --- | --- | --- | --- |
| Количество элементов в таблице | Время работы метода **Put(string \_key, T data)** (в млс) | Время работы метода **Del(string \_key)** (в млс) | Время работы метода **Search(string \_key)** (в млс) |
| 1000 | 0 | 0 | 0 |
| 10000 | 2 | 2 | 2 |
| 20000 | 4 | 33 | 3 |

*Таблица 1. Результаты экспериментов.*

Таким образом, можно увидеть, что сложность работы методов **Put(string \_key, T data)**, **Search(string \_key)** и **Del(string \_key)** – .

# Заключение.

В данной лабораторной работе мне удалось реализовать библиотеку для хранения и работы с упорядоченными таблицами, а именно:

* Удалось реализовать вспомогательный класс элемента таблицы **TSortElem**.
* Удалось реализовать класс упорядоченной таблицы **TSortTable**.
* Удалось протестировать методы классов **TSortElem** и **TSortTable**, а также обеспечить их работоспособность.
* Удалось реализовать класс для обработки исключений **TMyException**.

Таким образом, я смог реализовать структуру данных под названием упорядоченная таблица, элементы которой упорядочены по ключам.

# Литература.

1. Ссылка из Википедии про таблицы:

<https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0_(%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D1%8F)>

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», 2015.