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: просматриваемые 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](#_Toc7356404)

[2. Постановка задачи. 3](#_Toc7356405)

[3. Руководство пользователя. 4](#_Toc7356406)

[4. Руководство программиста. 5](#_Toc7356407)

[4.1. Описание структуры программы. 5](#_Toc7356408)

[4.2. Описание структур данных. 5](#_Toc7356409)

[4.3. Описание алгоритмов. 6](#_Toc7356410)

[5. Эксперименты. 7](#_Toc7356411)

[6. Заключение. 8](#_Toc7356412)

[7. Литература. 9](#_Toc7356413)

# Введение.

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

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

# Постановка задачи.

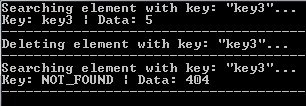
В данной лабораторной работе нужно разработать эффективную библиотеку для хранения и работы с такой структурой данных, как просматриваемая таблица.

Для этого нам нужно:

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

# Руководство пользователя.

Пример работы класса **TTable**:

****

*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:

* **ViewingTable** – содержит в себе файл **viewing\_table\_main.cpp** с реализацией примера использования класса **TTable**.
* **ViewingTableLib –** содержит в себе файлы **ViewingTable.h** и **TableElem.h**, в которых описаны и реализованы классы **TTable** и **TElem** соответственно.
* **ViewingTableTest** – содержит в себе файл **test\_viewing\_table.cpp**, в котором находится набор тестов, для проверки работоспособности классов **TElem** и **TTable**.
* **MyExceptionLib** –содержитв себе файл **MyException.h** с реализацией класса исключений **TMyException**.

# Описание структур данных.

**Класс TElem.**

Класс **TElem** 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**:

* **TElem(string \_key, T \_data)** – конструктор-инициализатор.
* **TElem(TElem &obj)** – конструктор копирования.
* **TElem<T>& operator=(TElem &obj)** – оператор присваивания.
* **bool operator==(TElem &obj)** – оператор сравнения.
* **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 TTable.**

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

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

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

* **TTable(int \_size = 10)** – конструктор-инициализатор.
* **TTable(TTable &obj)** – конструктор копирования.
* **void Put(string \_key, T \_data)** – метод, помещающий элемент в таблицу.
* **void Del(string \_key)** – метод, удаляющий элемент из таблицы.
* **TElem<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 есть свободные места, то новый элемент помещается в конец таблицы, а количество элементов таблицы увеличивается на единицу.

**Удаление элемента из таблицы.**

Сначала проверяем таблицу на пустоту. Если она не пуста, то выполняем поиск элемента по ключу. Если элемент найден, то ему присваивается значение константы **notFound**. При этом количество элементов в таблице не уменьшается.

**Поиск элемента в таблице.**

Поиск выполняется по ключу. Совершается обход таблица по количеству элементов и ключ каждого элемента сравнивается с искомым ключом. Если элемент с искомым ключом найден, то возвращается содержимое этого элемента, если же элемент не найден, 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 | 2 | 3 |
| 10000 | 0 | 278 | 281 |
| 20000 | 0 | 5594 | 5631 |

*Таблица 1. Результаты экспериментов.*

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

# Заключение.

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

* Удалось реализовать вспомогательный класс элемента таблицы **TElem**.
* Удалось реализовать класс просматриваемой таблицы **TTable**.
* Удалось протестировать методы классов **TElem** и **TTable**, а также обеспечить их работоспособность.
* Удалось реализовать класс для обработки исключений **TMyException**.

Таким образом, я смог реализовать структуру данных под названием просматриваемая таблица, элементы которой не упорядочены.

# 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.