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: просмотровые таблицы»**

**Выполнил:** студент группы 381706-1

Кольтюшкина Янина Вадимовна

\_\_\_\_\_\_\_\_\_\_\_ 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.

**Целью** данной лабораторной работы является практическая реализация структуры данных – просмотровая таблица

Ознакомимся со следующими базовыми понятиями.

**Таблица** - двухмерный объект, состоящий из строк и столбцов, который используется для хранения данных.

**Просмотровая 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.

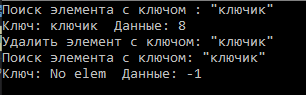
Одной из главных задач данной лабораторной работы является разработка и реализация структуры хранения «просмотровая таблица»

Программное решение будет выглядеть следующим образом:

1. Описание и реализация класса элемента таблицы **TElem**.
2. Описание и реализация класса просмотровой таблицы **TTable**.
3. Тестирование класса **TTable** с помощью Google Test.
4. Реализация класса **ExceptionLib** для обработки исключений, которые могут возникнуть в результате работы класса **TTable**.
5. Демонстрация работы класса **TTable**.

# Руководство пользователя.

Пример работы класса **TTable**:



*Рис 1. Пример работы программы*

При запуске программы создается таблица из 6 элементов. Далее она заполняется по следующему принципу: ключом элемента служит русскоязычная строка, где к каждому последующему ключу прибавляется одна буква, чтобы получить из «к» слово «ключик», данными элементов служат элементы ряда Фибоначчи. Затем производится поиск элемента с ключом «ключик» и вывод его данных на экран. После этого элемент с этим же ключом удаляется и вновь производится его поиск, результат которого означает, что элемент не найден.

# Руководство программиста.

# Описание структуры программы.

* Модуль **ViewingTable** – содержит пример использования класса **TTable**. Реализация в файле **viewing\_table\_main.**.
* Модуль **ViewingTableLib –** содержит описание и реализацию классов **TTable** и **TElem** в файлах **ViewingTable.h** и **TableElem.h**.
* Модуль **ViewingTableTest** – содержит в себе файл **test\_viewing\_table.cpp**, в котором находится набор тестов, для проверки работоспособности классов **TElem** и **TTable**.
* Модуль **ExceptionLib** –содержитв себе файл **ExceptionLib.h** с реализацией класса исключений **MyException**.

# 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 &elem)** – конструктор копирования.
* **void SetKey(string \_key)** – метод, устанавливающий ключ.
* **void SetData(T \_data)** – метод, устанавливающий данные.
* **T GetData()** – метод, возвращающий данные.
* **T& GetDataAddress()** – метод, возвращающий адрес данных.
* **TElem<T>& operator=(TElem &elem)** – оператор присваивания.
* **bool operator==(TElem &elem)** – оператор сравнения.
* **string GetKey()** – метод, возвращающий ключ.

**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 leng** – размер таблицы.
* **int count** – количество элементов в таблице.

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

* **TTable(int \_size = 10)** – конструктор-инициализатор.
* **TTable(TTable &elem)** – конструктор копирования.
* **void SetLeng(int \_leng)** – метод, устанавливающий размер таблицы.
* **void SetCount(int \_count)** – метод, устанавливающий количество элементов в таблице.
* **int GetLeng()** – метод, возвращающий размер таблицы.
* **int GetCount()** – метод, возвращающий количество элементов в таблице.
* **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.

**Класс ExceptionLib.**

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

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

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

* **Exception(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.

Эксперименты проводились на ПК с следующими параметрами:

1. Операционная система: Windows 10
2. Процессор: Intel(R) Core™ i7-4710MQ CPU @ 2.50 GHz
3. 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 | 287 | 293 |
| 20000 | 0 | 5613 | 5657 |

*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 этой лабораторной работе я смогла справиться со всеми поставленными задачами: реализовала вспомогательный класс элемента таблицы **TElem**, сам класс просмотровой таблицы **TTable**, были реализованы тесты и обработкаисключений. В конечном итоге мне удалось реализовать структуру данных «просмотровая таблицы».

# 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.