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

Силенко Дмитрий Игоревич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2019.

**Оглавление**

[1. Введение. 2](#_Toc8320681)

[2. Постановка задачи. 3](#_Toc8320682)

[3. Руководство пользователя. 4](#_Toc8320683)

[4. Руководство программиста. 5](#_Toc8320684)

[4.1. Описание структуры программы. 5](#_Toc8320685)

[4.2. Описание структур данных. 5](#_Toc8320686)

[4.3. Описание алгоритмов. 6](#_Toc8320687)

[5. Эксперименты. 8](#_Toc8320688)

[6. Заключение. 9](#_Toc8320689)

[7. Литература. 10](#_Toc8320690)

# Введение.

**Основная цель данной работы** – разработка динамической структуры данных – просмотровой таблицы.

Но для начала необходимо разобраться, что такое таблица и упорядоченная таблица и как они работают.

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

**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 Выполнение работы предполагает решение следующих задач:

1. Разработка и реализация классов элемент упорядоченной таблицы и упорядоченная таблица – TSortElem, TSortTable.
2. Пример программы, демонстрирующая работу класса TSortTable.
3. Написание набора автоматических тестов с использованием Google C++ Testing Framework и проверка работоспособности методов классов.

# Руководство пользователя.

При запуске программы создается таблица, состоящая из 6 элементов. Номер ключа соответствует номеру элемента в таблице. Выполняется поиск и вывод на кран элементов по ключу k1 и k2. После чего элемент с ключом k2 удаляется из таблицы и производится повторный поиск его, а также элемента, которого в таблице в принципе нет. И в том и в другом случае программа оповещает о том, что подобных элементов нет.

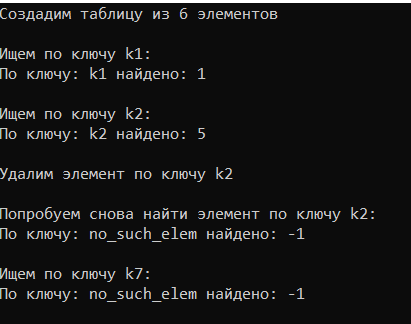


Рисунок 1 Поиск по разным ключам

# Руководство программиста.

# Описание структуры программы.

Программа состоит из следующих модулей:

* Модуль SortTableLib. Статическая библиотека. Включает в себя заголовочные файлы SortTableElem.h и SortTable.h, в которых описаны методы с реализаций шаблонных классов *TSortElem и TSortTable.*
* Модуль SortTableTest. Набор тестов для класса TSortTable. Включает в себя файл test\_sort\_table*.cpp.* Разработаны они с помощью использования Google C++ Testing Framework.
* Модуль SortTable. Пример использования списка. Включает в себя файл с реализацией sort\_table*\_main.cpp*

# Описание структур данных.

*Класс TSortElem:*

Поля:

String key; - строка-ключ для доступа к элементам таблицы

T data; - сами данные

Конструкторы:

TSortElem(string \_key =””, T \_data =0);

TSortElem(TSortElem &selem);

Методы:

void SetKey(string \_key); - задать ключ

void SetData(T \_data); - задать данные

string GetKey(); - получить ключ

T GetData(); - получить данные

T& GetDataAddress(); - получить адрес данных

TSortElem<T>& operator=(TSortElem<T> &selem);

bool operator==(TSortElem<T> &selem);

bool operator>(TSortElem<T> &selem);

bool operator<(TSortElem<T> &selem);

*Класс TSortTable:*

Поля:

TSortElem<T> notFound; - элемент не найден

TSortElem<T> \*mas; - массив элементов таблицы

int length; - длина таблицы

int count; - количество элементов в таблице

Конструкторы:

TSortTable(int \_length = 10);

TSortTable(string\* \_keys, T\* \_data, int \_length);

TSortTable(TSortTable<T> &stable);

Методы:

void SetSize(int \_size); - задать размер таблицы

void SetCount(int \_count); - задать количество элементов в таблице

int GetLength(); - получить размер таблицы

int GetCount(); - получить количество элементов в таблице

int Put(string \_key, T \_data); - положить элемент в таблицу

void Del(string \_key); - удалить элемент из таблицы

TSortElem<T>& Search(string \_key); - найти элемент в таблице

void Resize(int \_size); - перепаковка

T& operator[](string \_key); - доступ к элементу по ключу

# 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.

Эксперименты проводились на ПК с следующими параметрами:

1. Операционная система: Windows 10 Домашняя
2. Процессор: Intel(R) Core™ i5-8250U CPU @ 1.60 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 | 0 | 0 |
| 10000 | 3 | 3 | 3 |
| 20000 | 3 | 30 | 4 |

*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.

Эта лабораторная работа дала возможность более детально разобраться с устройством работы упорядоченной таблицы. В ходе выполнения, был реализован вспомогательный шаблонный класс элемента упорядоченной таблицы TSortElem и основной шаблонный класс упорядоченной таблицы TSortTable, описанный в специально разработанной библиотеке SortTableLib. Все это позволило нам реализовать базовую работу с упорядоченной таблицей: поиск элемента по ключу, добавление и удаление элемента, перепаковка.

Помимо этого, для проверки работоспособности всех методов, были написаны автоматические тесты, реализованные с использованием Google C++ Testing Framework. Пример использования списка для пользователя также написан и успешно работает.

# 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.