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

Денисов Владислав Львович

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2019.

Содержание

[1. Введение 3](#_Toc8377998)

[2. Постановка задачи 4](#_Toc8377999)

[3. Руководство пользователя 5](#_Toc8378000)

[4. Руководство программиста 6](#_Toc8378001)

[4.1 Описание структуры программы 6](#_Toc8378002)

[4.2 Описание структур данных 6](#_Toc8378003)

[4.3 Описание алгоритмов 8](#_Toc8378004)

[5. Заключение 9](#_Toc8378005)

[6. Литература 10](#_Toc8378006)

# Введение

Лабораторная работа направлена на практическое освоение структуры данных под названием просматриваемая таблица.

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

В просматриваемой таблице порядок расположения элементов никак не связан со значениями ключей. Поэтому поиск элемента по ключу осуществляется обычным просмотром всех элементов таблицы, начиная с первого и до искомого (если он есть в таблице; тогда поиск заканчивается успешно), или до конца таблицы (тогда поиск заканчивается неуспешно)

# Постановка задачи

В рамках лабораторной работы ставится задача реализации структуры данных – просматриваемая таблица.

Для работы с просмотровой таблицей будут реализованы:

* возможность создания просмотровой таблицы,
* метод добавление и удаления элементов в таблицу,
* метод поиска элемента в таблице,
* метод, позволяющий вывести таблицу на экран.

Программное решение будет выглядеть следующим образом:

1. Класс просмотровой таблицы – TTable.
2. Класс элемента таблицы – TElem.
3. Класс для обработки исключений – TException, которые могут возникнуть при выполнении различных операций.
4. Программа, демонстрирующая работу класса TTable.
5. Набор автоматических тестов с использованием Google C++ Testing Framework.

# Руководство пользователя

В качестве примера использования просматриваемой таблицы предлагается следующее.

Создается таблица размером для хранения 10 элементов. Производится ее вывод на экран – она будет пустой, поэтому отобразится только сообщение о выводе.

Добавляются элементы с различными ключами и данными.

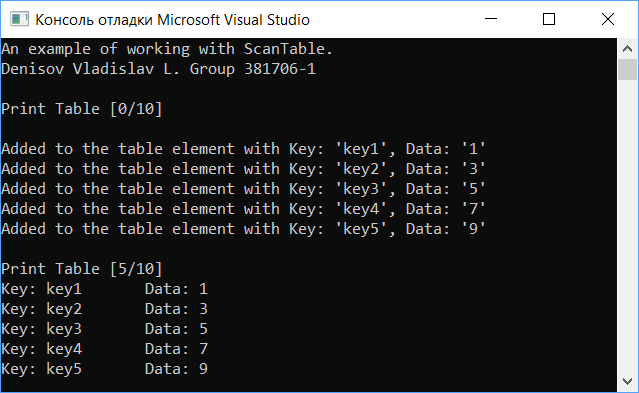


Рисунок 1 Пример работы демонстрационной программы.

Затем выполняется поиск элемента с указанным ключом и вывод найденного элемента на консоль. Удаляется элемент с указанным ключом, затем выводится таблица, в которой удалили элемент. Производится попытка найти элемент, который был удален.

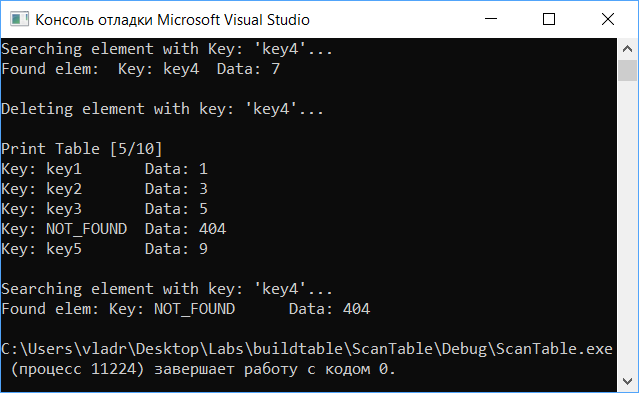


Рисунок 2 Пример работы демонстрационной программы.

# Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из следующих модулей:

* Модуль ScanTable. Содержит пример использования просмотровой таблицы. Реализация в файле *main\_ScanTable.cpp.*
* Модуль ScanTableLib – статическая библиотека. Содержит файлы *ScanTable*.h, *ScanElem.h,* в которых описаны интерфейс и реализация шаблонных классов *TTable* и *TElem*  соответственно.
* Модуль ScanTableTest. Содержит 17 тестов (6 для класса элемента таблицы и 11 для класса просмотровой таблицы), описанных в файлах *ScanTableTest.cpp и ScanElemTest.cpp,* разработанных с помощью использования Google C++ Testing Framework.
* Модуль ExceptionLib – библиотека, позволяющая создавать собственные исключения.

## Описание структур данных

#### Класс TException – класс исключений.

Класс содержит 1 **private** поле *std::string msg* – переменная, хранящая сообщение об ошибке в виде строки.

И содержит 2 **public** элемента:

*TException(std::string \_str) : msg(\_msg)* – конструктор с одним параметром.

*void Print()* – метод отображения ошибки на консоль.

#### Класс TElem – класс элемента таблицы.

Рассмотрим класс *TElem* подробно.

template <class T> class TElem {…} – класс элемента таблицы является шаблонным, что позволяет использовать его для хранения данных любого типа.

**Элементы класса, объявленные со спецификатором protected:**

string key – значение ключа.

T data – хранимая информация.

**Элементы класса, объявленные со спецификатором public:**

TElem(string \_key = "", T \_data = 0) – конструктор инициализатор.

TElem(TElem &copy) – конструктор копирования.

void SetKey(string \_key) – метод изменения значения ключа.

string GetKey() – метод получения значения ключа.

void SetData(T \_data) – метод изменения значения хранимой информации.

T GetData() – метод получения значения хранимой информации.

T& GetDataAddress() – метод возвращающий адрес на хранимую информацию.

**Перегруженные операторы:**

TElem<T>& operator=(TElem &elem) – оператор присваивания.

bool operator==(TElem &elem) – оператор сравнения на равенство элементов таблицы.

bool operator!=(TElem<T> &elem) – оператор проверки на неравенство элементов таблицы.

**Дружественная функция:**

friend ostream& operator<<(ostream& out, TElem<T>& elem) – вывод элемента таблицы на консоль.

#### Класс TTable – класс таблицы.

Рассмотрим класс *TTable* подробно.

template <class T> class TTable {…} – класс просмотровой таблицы является шаблонным, что позволяет использовать его для хранения данных любого типа.

**Элементы класса, объявленные со спецификатором protected:**

TElem<T> notFound

TElem<T> \*mas;

int size, count;размер стека.

**Элементы класса, объявленные со спецификатором public:**

T TTable(int \_size = 10) – конструктор инициализатор.

TTable(TTable &table) – конструктор копирования.

int GetSize() – получение максимального числа элементов в таблице.

int GetCount() – получение числа добавленных элементов.

void Put(string \_key, T \_data) – метод добавления элемента.

void Del(string \_key) – метод удаления элемента из таблицы по ключу.

TElem<T>& Search(string \_key) – метод поиска элемента в таблице.

**Перегруженный оператор индексации:**

T& operator[](string \_key) – оператор 0-based индексации.

**Дружественная функция:**

friend ostream& operator<<(ostream& out, TTable<T>& table) – вывод просматриваемой таблицы на консоль.

## Описание алгоритмов

Рассмотрим некоторые алгоритмы, работа которых не очевидна на первый взгляд.

**Добавление элемента в таблицу**

Выполняем проверку таблицы на полноту. Если свободных мест нет, то возбуждается исключение.

Если в ней есть свободные места, то новый элемент помещается в конец таблицы, а количество элементов таблицы увеличивается на единицу.

Выполнение операции производится за O(1), т.к. при каждом добавлении изначально известна позиция последнего добавленного элемента, искать подходящую позицию не требуется.

**Удаление элемента из таблицы**

Выполняем проверку таблицы на пустоту. Если элементов в таблице нет, то возбуждается исключение.

Если таблица не пуста, то выполняется поиск элемента по ключу. В случае успешного обнаружения элемента ему присваивается системное значение notFound**,** что означает отсутствие элемента в таблице. При этом количество элементов в таблице не уменьшается, т.к. элемент физически остается в таблице, но не содержит полезной информации. Перепаковка не предусмотрена, поэтому удаленные элементы нельзя затереть.

Удаление выполняется со сложностью O(N), т.к. выполняется поиск удаляемого элемента таблицы, при котором элементы просматриваются последовательно.

**Поиск элемента в таблице**

Поиск выполняется по ключу, который передается в качестве аргумента при вызове функции поиска. Совершается просмотр таблицы, начиная с первого до последнего занятого. Ключ каждого просматриваемого элемента сравнивается с искомым ключом.

Если элемент с искомым ключом найден, то возвращается хранимая в этом элементе информация.

Если же элемент не найден, то возвращается системное значение notFound, что означает отсутствие элемента в таблице.

Поиск выполняется со сложностью O(N), т.к. выполняется последовательный просмотр элементов таблицы.

# Заключение

В результате лабораторной работы была разработана библиотека, реализующая шаблонный класс просматриваемой таблицы. Она позволяет создать объект класса просматриваемой таблицы и выполнить с ним простейшие операции, задача реализации которых была поставлена в начале данной лабораторной работы.

Были разработаны и доведены до успешного выполнения тесты, разработанные для данного программного проекта с использованием Google C++ Testing Framework.

Программное решение было продемонстрировано с помощью простейшего набора операций над просматриваемой таблицей. Описание примера работы с просматриваемой таблицей было представлено в разделе «Руководство пользователя».

# Литература

1. Википедия: свободная электронная энциклопедия: на русском языке [Электронный ресурс] // URL: https://ru.wikipedia.org/wiki/Таблица\_(значения)
2. Студенческие материалы Xstud – Просмотровые таблицы [Электронный ресурс] // URL: https://xstud.ru/97614/matematicheskaya\_logika\_i\_teoriya\_algoritmov/prosmatrivaemye\_tablitsy