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

Денисов Владислав Львович

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2019.

Содержание

[1. Введение 3](#_Toc8579667)

[2. Постановка задачи 4](#_Toc8579668)

[3. Руководство пользователя 5](#_Toc8579669)

[4. Руководство программиста 7](#_Toc8579670)

[4.1 Описание структуры программы 7](#_Toc8579671)

[4.2 Описание структур данных 7](#_Toc8579672)

[4.3 Описание алгоритмов 9](#_Toc8579673)

[5. Заключение 11](#_Toc8579674)

[6. Литература 12](#_Toc8579675)

# Введение

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

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

В упорядоченной таблице порядок расположения элементов неразрывно связан со значениями ключей. Элементы располагаются по возрастанию ключей в лексикографическом порядке. Благодаря такому подходу значительно упрощается навигация по таблице: найти элемент по ключу можно с помощью применения бинарного поиска, который имеет сравнительно неплохую сложность в плане временных затрат.

Однако из-за необходимости упорядочивания элементов усложняется организация структуры таблицы и непосредственно работа с ней.

# Постановка задачи

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

Для работы с упорядоченной таблицей будут реализованы:

* возможность создания упорядоченной 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 – TSortTable.
2. Класс элемента таблицы – TElem (используется из лабораторной работы «Структура хранения данных: просматриваемая таблица»).
3. Класс для обработки исключений – TException, которые могут возникнуть при выполнении различных операций.
4. Программа, демонстрирующая работу класса TSortTable.
5. Набор автоматических тестов с использованием Google C++ Testing Framework.

# Руководство пользователя

В качестве примера использования упорядоченной таблицы предлагается следующее.

Создаются и инициализируются 2 массива: ключи ‘Keys’ и данные 'Data'. Производится их вывод на консоль.

Строится упорядоченная таблица из этих 2 массивов при помощи сортировки вставками.

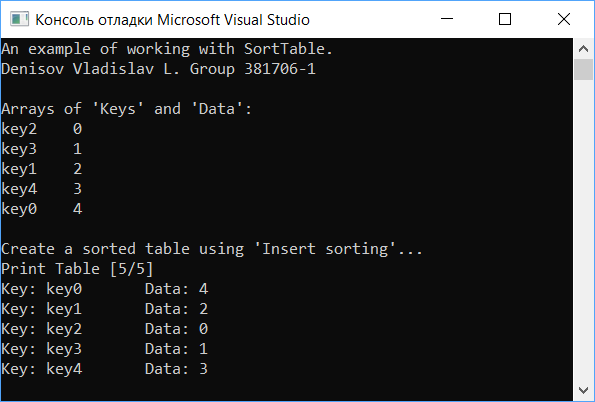


Рисунок 1 Пример работы демонстрационной программы.

Аналогично происходит создание двух таблиц с использованием алгоритмов сортировки Шелла и Хоара (Быстрая).

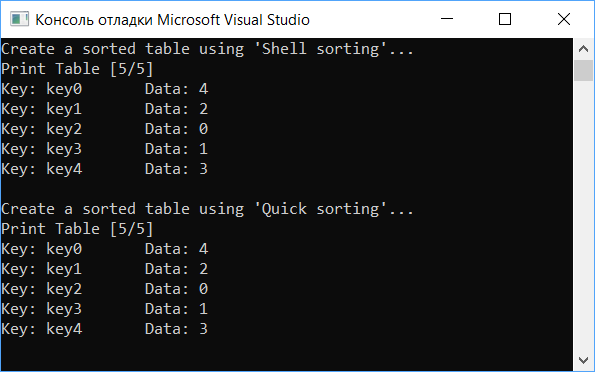


Рисунок 2 Пример работы демонстрационной программы.

После этого создается пустая таблица размером 1 элемент. В нее производится добавление первого элемента. При добавлении второго элемента происходит автоматическое увеличение размера таблицы и выполняется добавление.

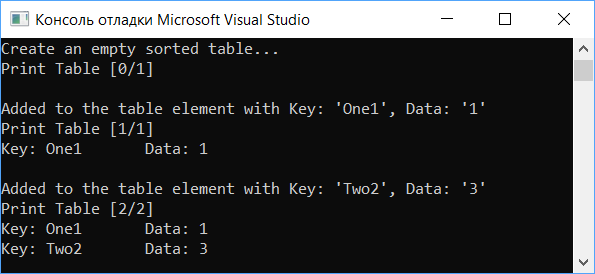


Рисунок 3 Пример работы демонстрационной программы.

Добавляется третий элемент, при этом снова увеличивается размер таблицы, т.к. свободных ячеек снова нет.

Происходит поиск элемента с указанным ключом. Выполняется его удаление. Измененная таблица выводится на консоль. Выполняется повторный поиск того же элемента – в результате возвращен системный элемент таблицы «не найден».

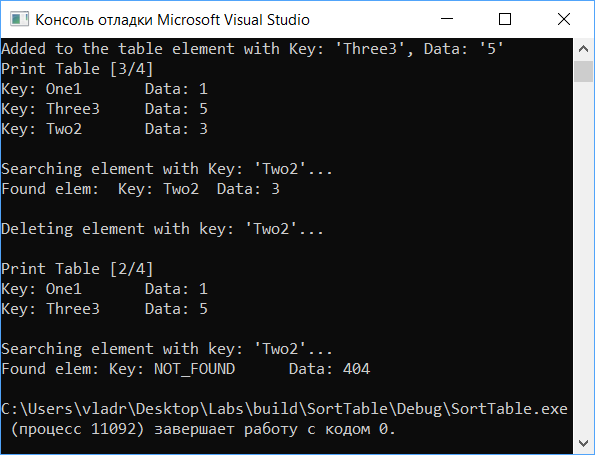


Рисунок 4 Пример работы демонстрационной программы.

На этом работа демонстрационной программы завершается.

# Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из следующих модулей:

* Модуль SortTable. Содержит пример использования упорядоченной таблицы. Реализация в файле *main\_SortTable.cpp.*
* Модуль SortTableLib – статическая библиотека. Содержит файл *SortTable*.h*,* в котором описан интерфейс и реализация шаблонного класса *TSortTable*.
* Модуль SortTableTest. Содержит 17 тестов, описанных в файле *SortTableTest.cpp* иразработанных 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.

bool operator>(TElem &elem) – оператор «больше».

bool operator<(TElem &elem) – оператор «меньше».

**Дружественная функция:**

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

#### Класс TSortTable – класс упорядоченной таблицы.

Рассмотрим класс *TSortTable* подробно.

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

Причем, класс TSortTable является дружественным для класса TTable (просматриваемой таблицы), чтобы иметь доступ к его полям для удобства работы.

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

TElem<T> notFound – отсутствующий в таблице элемент.

TElem<T> \*mas – указатель на область памяти, выделенную под элементы таблицы.

int size, count – максимальный размер таблицы и число занятых элементов.

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

TSortTable(int \_size = 10) – конструктор инициализатор.

TSortTable(TSortTable &sorttable) – конструктор копирования.

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

int GetCount() – получение числа добавленных элементов.

void Resize(int \_size) – увеличение размера таблицы.

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.

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

**Дружественная функция:**

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

## Описание алгоритмов

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

**Добавление элемента в таблицу**

Выполняется проверка на число заполненных элементов. Если свободных ячеек нет, то производится увеличение таблицы в 2 раза.

Если в таблице нет ни одного элемента, то новый добавляется в первую ячейку.

В противном случае производится бинарный поиск позиции, в которую будет производится добавление элемента. Поиск выполняется с учетом того, что должно сохраняться условие упорядоченности ключей элементов в лексикографическом порядке.

Когда позиция найдена, все элементы, расположенные правее этой позиции, сдвигаются вправо по массиву на одну ячейку. Выполняется добавление нового элемента в найденную позицию и увеличение счетчика числа существующих элементов на единицу.

Выполнение операции производится за O(N), т.к. при каждом добавлении выполняется поиск подходящей позиции и сдвиг элементов с большими значениями ключей.

**Удаление элемента из таблицы**

Выполняем проверку таблицы на пустоту. Если элементов в таблице нет, то выполнение функции завершается.

Если таблица не пуста, то выполняется бинарный поиск элемента по ключу. В случае успешного обнаружения элемента элементы стоявшие правее найденной позиции, т.е. имеющие большее значение ключа, сдвигаются на одну ячейку влево по массиву. Количество элементов в таблице уменьшается на единицу.

Выполнение операции производится за O(N), т.к. при каждом добавлении выполняется поиск подходящей позиции и сдвиг элементов с большими значениями ключей.

**Поиск элемента в таблице**

Поиск выполняется по ключу, который передается в качестве аргумента при вызове функции поиска. Применяется бинарный поиск.

Берётся средний элемент из массива: если он совпадает с тем, запрос на поиск которого выполнялся, то задача решена. В противном случае, если заданное значение больше этого элемента, то известно, что оно расположено в правой части массива. А если меньше — то в левой. Эти подмассивы разбиваются до тех пор, пока не будет обнаружен искомый элемент.

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

Поиск в данном случае выполняется со сложностью O(LogN).

**Создание таблицы путем упорядочивания входных данных с помощью алгоритма быстрой сортировки.**

Входные данные представляют собой 2 массива: ключей и информации, которую требуется поместить в таблицу.

Строится временная таблица и заполняется полученными данными по принципу просмотровой. Затем она передается в функцию, которая выполняет быструю сортировку.

Алгоритм быстрой сортировки представляет собой следующее.

Выбирается опорный элемент, в нашем случае – центральный. Таблица разделяется на две части следующим образом: все элементы из левой части, которые больше или равны опорному, перемещаются в правую, аналогично, все элементы из правой, которые меньше или равны опорному перемещаются в левую часть.

В результате этого в левой части массива останутся элементы, которые меньше или равны центральному, а в правой — больше либо равны.

Затем это действие рекурсивно повторяется для левой и правой частей таблицы.

Содержимое временной таблицы копируется в структуру для хранения упорядоченной таблицы.

Создание таблицы с помощью сортировки вставками и Шелла производится по аналогичному сценарию.

# Заключение

В результате лабораторной работы была разработана библиотека, реализующая шаблонный класс упорядоченной таблицы. Она позволяет создать объект класса упорядоченной 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 упорядоченной таблицей. Описание примера работы с упорядоченной 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. Википедия: свободная электронная энциклопедия: на русском языке [Электронный ресурс] // URL: <https://ru.wikipedia.org/wiki/Двоичный_поиск>
3. Хабр: Введение в анализ сложности алгоритмов [Электронный ресурс] // URL: https://habr.com/ru/post/195996/