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**

оставь надежды всяк сие читающий (карин т.а.)

**Отчет по учебной практике**

**«**Структура хранения данных: просматриваемые таблицы**»**

**Выполнил:**

студент группы 381706­1

Карин Тимофей Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Проверил:**

ассистент каф. МОСТ ИИТММ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2019.

Содержание

[1. Введение 3](#_Toc1347448)

[3. Руководство пользователя 5](#_Toc1347449)

[4. Руководство программиста 6](#_Toc1347450)

[4.1. Описание структуры программы 6](#_Toc1347451)

[4.2 Описание структур данных 6](#_Toc1347452)

[4.3 Описание алгоритмов 7](#_Toc1347453)

[5. Заключение 9](#_Toc1347454)

[6. Список литературы 10](#_Toc1347455)

1. Введение

В данной лабораторной работе под таблицей понимается набор элементов (полей), состоящих из ключа и набора данных. Ключ является идентификатором элемента таблицы. Ключ уникален, то есть в одной таблице не может находиться несколько полей с одинаковым ключом.

Основным свойством просматриваемой таблицы является неупорядоченность полей. То есть порядок расположения элементов в таблице не важен. Это самый простой способ организации таблицы. Просматриваемая таблица используется в тех случаях, когда часто используется вставка элементов, но редко используется поиск.

В данной лабораторной работе будут рассмотрены таблицы, в которых ключи элементов представляют собой некоторую строку, а пользовательские данные являются шаблонным классом.

2. Постановка задачи

Целью данной лабораторной работы является написание библиотеки для работы с просматриваемыми таблицами. Для этого необходимо выполнить следующие задачи:

* Создать класс, объекты которого будут являться элементами таблицы;
* Создать класс, предназначенный для хранения таблиц и работы с ними;
* Написать пример использования библиотеки;
* Проверить на работоспособность все методы классов.

3. Руководство пользователя

Для работы пользователя была разработана небольшая программа, демонстрирующая основные возможности библиотеки. Программа расположена в файле *ScanTable.cpp* модуля *ScanTable.*

После запуска программы пользователю необходимо ввести размер создаваемой таблицы. При этом в ходе дальнейшей работы поменять размер таблицы не получится.

Элементы таблицы состоят из ключа, который представляет собой строку, и пользовательских данных, являющихся целым числом.

После создания таблицы пользователю будет предложено несколько вариантов действий:

1. добавить поле в таблицу;
2. удалить элемент из таблицы;
3. найти элемент;
4. узнать количество элементов в таблице;
5. вывести таблицу на экран;
6. выйти из программы;

Для выбора действия пользователю необходимо ввести номер соответствующего пункта.

Если в ходе работы возникнет исключительная ситуация, программа прекратит выполнение, а пользователь увидит сообщение об ошибке.

К основным исключительным ситуациям можно отнести:

* Добавление элемента с ключом, который уже присутствует в таблице;
* Добавление элемента в полностью заполненную таблицу;
* Удаление несуществующего элемента;

4. Руководство программиста

4.1. Описание структуры программы

Программа состоит из 3 основных модулей:

* *ScanTableLib –* библиотека для работы со списками. Состоит из файлов *ScanElem.h* и *ScanTable.h*, в которых соответственно находятся шаблонные классы элемента таблицы (класс *TScanElem*) и самой таблицы (класс *TScanTable*).
* *ScanTable –* пример использования библиотеки.
* *ScanTableTest –* тесты для проверки работоспособности библиотеки.

4.2 Описание структур данных

Рассмотрим класс *TScanElem*. Класс шаблонный, *T –* шаблонный тип данных, присваиваемый пользовательским данным. Рассмотрим поля и методы:

Со спецификатором доступа protected:

* *TString key* – ключ;
* *T data* – данные;

Со спецификатором доступа public:

* *TScanElem() –* конструктор по умолчанию.
* *TScanElem(TString k, T d) -* конструктор инициализатор (key=k, data=d);
* *TScanElem(TScanElem<T>& A) –* конструктор копирования;
* *TScanElem<T>& operator=(TScanElem<T>& A) –* оператор присваивания;
* *TString GetKey() –* получить ключ элемента;
* *T GetData() –* получить данные;
* *void SetKey(TString k) –* установить ключ в значение *k*;
* *void SetData(T d) –* установить данные в значение *d;*
* *bool operator==(TScanElem<T>& A) –* оператор сравнения. Возвращает true, если ключи совпадают, и false в остальных случаях.
* *bool operator>(TScanElem<T>& A) –* оператор сравнения. Сравнивает ключи в лексикографическом порядке;
* *bool operator< (TScanElem<T>& A) -* оператор сравнения. Сравнивает ключи в лексикографическом порядке;
* *friend ostream& operator<<(ostream& ostr, const TScanElem<T1> SE) –* оператор вывода;

Рассмотрим шаблонный класс TScanTable. Рассмотрим поля и методы:

Со спецификатором доступа protected:

* *TScanElem<T>\* mas –* массив элементов таблицы;
* *unsigned size –* размер таблицы;
* *unsigned count –* количество элементов в таблице;

Со спецификатором доступа public:

* *TScanElem<T> not\_find –* элемент, означающий несуществующее поле;
* *TScanTable(int n = 0) –* конструктор. *n* – размер таблицы;
* *TScanTable(TScanTable<T>& A) –* конструктор копирования;
* *void Put(TString k, T d) –* добавление поля с ключом *k* и данными *d*;
* *void Del(TString k) –* удаление элемента с ключом *k*;
* *int GetCount() –* получить количество записей в таблице;
* *TScanElem<T>& Search(TString k) –* поиск элемента с ключом *k*;
* *T& operator[](TString k) –* оператор индексации;
* *~TScanTable() –* деструктор;
* *friend ostream& operator<<(ostream& ostr, const TScanTable<T1> &A) –* оператор вывода.

4.3 Описание алгоритмов

Добавление элемента в таблицу:

Проверяем, есть ли в таблице свободное место. Если нет, возбуждается исключение. Если место есть, то элемент добавляется в конец таблицы, а count увеличивается на 1.

Сложность алгоритма: О(1), т.к. позиция добавляемого элемента заранее известна.

Удаление элемента из таблицы:

Если таблица пустая, то возбуждается исключение. Если в таблице есть элементы, то методом поиска ищется удаляемый элемент. Если искомый элемент найден, на его место встаёт последний элемент таблицы, а count уменьшается на 1. Если элемент не с данным ключом не найден, то возбуждается исключение.

Сложность алгоритма: О(N), т.к. используется линейный поиск, при котором просматривается каждый элемент. N-количество записей в таблице.

Поиск элемента:

Поиск элемента проверяет все ключи элементов таблицы на равенство со строкой, пришедшей в качестве параметра. Если элемент с таким ключом удалось найти, то возвращается ссылка на этот элемент. Если нет, то возвращается ссылка на элемент not\_find.

Сложность алгоритма: О(N), т.к. используется поочерёдная проверка всех элементов на равенство. N-количество записей в таблице.

5. Заключение

В ходе проделанной работы удалось достичь следующих результатов:

* Разработана библиотека для простой работы с просматриваемыми таблицами;
* Написан пример использования программы;
* Обработаны основные исключительные ситуации;
* Проверена работоспособность библиотеки с помощью тестов фреймворка Google Test.

6. Список литературы

1. Ахо Альфред В, Хопкрофт Джон Э и Ульман Джеффри Д Структуры данных и алгоритмы [Книга]. - [б.м.] : Вильямс, 2003.
2. Лафоре Роберт Структуры данных и алгоритмы в Java [Книга]. - СПб : Питер, 2013. - 2 : стр. 704.
3. Павловская Т. А. C/C++ Программирование на языке высокого уровня [Книга]. - СПб : Питер, 2003.
4. Страуструп Бьерн Язык программирования C++ Бином, 2004.