МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. Н.И. ЛОБАЧЕВСКОГО»**

Институт информационных технологий, математики и механики

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

«Статистическая обработка результатов экзаменационной сессии (таблицы)»

**Отчет выполнил:**

студент 2 курса группы 381906-3

Михеев Иван Алексеевич

**Руководитель практики:**

Гетманская Александра Александровна

преподаватель МОСТ ИИТММ

г. Нижний Новгород 2021 г.

## **Содержание**

[**Содержание**](#_iod3k66kj3cq) **1**

[**Введение**](#_w2qsvevbr2f9) **2**

[**Теоретическая часть**](#_n0cn6pkz6g1g) **2**

[Виды таблиц](#_licbaijyi1wn) 2

[Постановка задачи](#_96e5kjcgy9sl) 4

[**Практическая часть**](#_i736bein8egs) **5**

[Описание структуры программы](#_a9rdjnfvvapv) 5

[**Заключение**](#_x9jk6x9mngi6) **8**

[**Список литературы**](#_7utj7nn8a1v9) **9**

## Введение

**Структуры данных**  —  это контейнеры, которые хранят данные в определенном формате. Этот специфический формат придает структуре данных определенные качества, которые отличают ее от других структур и делают ее пригодной (или напротив, неподходящей) для тех или иных сценариев использования.

В данной лабораторной работе мы будем рассматривать структуру данных типа таблица.

**Цель работы** – разработка структур хранения данных – таблиц четырех различных видов: несортированные и сортированные таблицы на массиве, таблицы на бинарных деревьях поиска и хэш-таблицы. Наиболее подробно мы рассмотрим хэш-таблицы и их эффективность в зависимости от выбранной хэш-функции.

## **Теоретическая часть**

### **Виды таблиц**

**Таблица** —динамическая структура данных, базисным множеством которой является семейство линейных структур из записей. Хранит в себе набор элементов одинаковой организации, каждый из которых можно представить в виде двойки <K, V>, где K — ключ, а V — тело элемента.

**Запись** – кортеж, каждый элемент которого обычно именуется полем.

**Имя записи (ключ)** – одно из полей записи, по которому обычно осуществляется поиск записей в таблице; остальные поля образуют тело записи.

1. **Несортированная (просматриваемая) таблица**

Простейшим способом отыскания нужного элемента является метод полного просмотра (сканирования), когда искомый ключ сравнивается по очереди со всеми ключами таблицы, начиная с первого, вплоть до отыскания совпадающего элемента или до исчерпания записей. Если ключи в таблице расположены в произвольном порядке (неупорядоченная таблица), этот способ является единственно возможным.

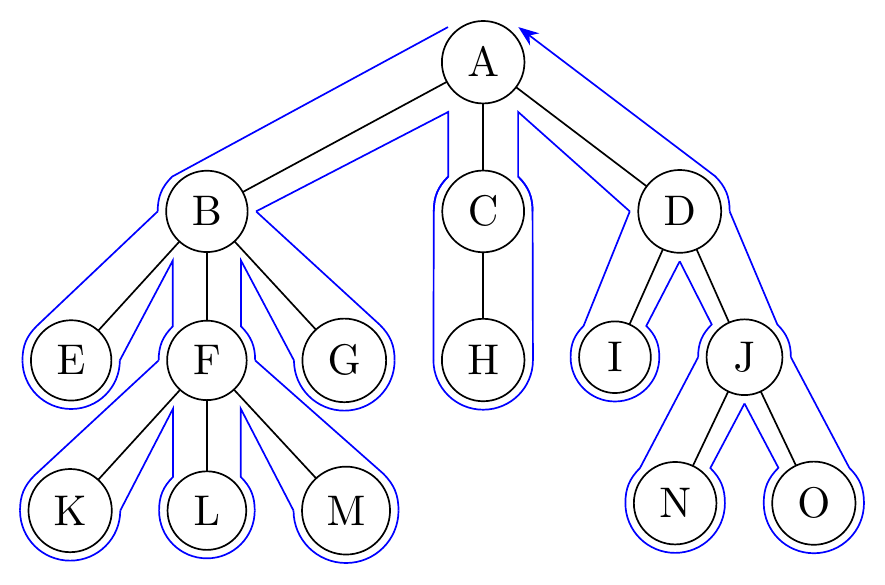
1. **Упорядоченные таблицы**

Эффективность процедуры поиска можно повысить при размещении записей в таблице в порядке возрастания (или убывания) ключей (упорядоченная, или отсортированная таблица). Упорядоченность таблиц может быть организована только при возможности сравнения ключей (на множестве ключей задано отношение линейного порядка). Вместе с тем, в упорядоченных таблицах усложняется реализация операций вставки и удаления записей, при выполнении которых для сохранения упорядоченности становится необходимой перепаковка записей таблицы.

1. **Таблицы на основе деревьев поиска**

**Бинарное дерево** — это иерархическая структура данных, в которой каждый узел имеет значение (оно же является в данном случае и ключом) и ссылки на левого и правого потомка. Узел, находящийся на самом верхнем уровне (не являющийся чьим либо потомком) называется корнем. Узлы, не имеющие потомков (оба потомка которых равны NULL) называются листьями.

Таблицы на основе бинарного дерева обладают теми же свойствами, а их использование обеспечивает сложность в среднем для всех операций обработки (поиска, вставки и удаления) таблиц.



1. **Таблицы с вычисляемыми адресами (Хэш-таблицы)**

**Хеш-таблицей** называется структура данных, предназначенная для реализации ассоциативного массива, такого в котором адресация реализуется посредством хеш-функции.

**Хеш-функция** – это функция, преобразующая ключ key в некоторый индекс i равный h(key), где h(key) – хеш-код (хеш-сумма, хеш) key. Весь процесс получения индексов хеш-таблицы называется хеширование.

Использование таблиц с вычисляемыми адресами обеспечивает сложность в среднем для всех операций обработки (поиска, вставки и удаления) таблиц. Но при использовании таблиц с вычисляемыми адресами может возникнуть проблема коллизии.

**Коллизия** **хеш**-**функции** — это такая пара блоков x и y, результат **хеш**-**функции** **hash**() от которых дает в результате одинаковый блок z.

Существуют методы позволяющие избегать появления коллизий:

1. Метод открытого перемешивания (адресации)
2. Метод цепочек

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип таблицы** | **insert** | **remove** | **find** |
| Несортированная таблица |  |  |  |
| Упорядоченная таблица |  |  |  |
| Таблица на основе деревьев поиска |  |  |  |
| Хэш-таблицы |  |  |  |

**\*** Для упорядоченной не учтено время необходимое для перепаковок для операций вставки и удаления

### **Постановка задачи**

В условиях данной лабораторной работы необходимо реализовать таблицы 4 типов: Scan, Sort, Tree, Hash. Для каждой из таблиц должны быть реализованы следующие методы: добавление новой строки, удаление строки и поиск строки, добавление/вставка (insert), удаление (delete) и поиск (find) записи.

Используемая интегрированная среда разработки (IDE) Visual Studio 2019, язык – C++.

## 

## **Практическая часть**

### Описание структуры программы

Программа состоит из следующих заголовочных файлов:

* ttable.h
* arraytab.h
* scantab.h
* sorttab.h
* arrhash.h
* hashtab.h
* treetab.h
* treenode.h

и содержит следующие исполнительные файлы:

* arraytab.cpp
* scantab.cpp
* sorttab.cpp
* arrhash.cpp
* treetab.cpp
* main.cpp

1. **trecord.h** - реализует часть функционала таблиц

class TTabRecord

Методы:

SetKey() - установить ключ для элемента в таблице

GetKey() - получить ключ элемента в таблице

SetValuePtr() - установить значение элемента в таблице

GetValuePtr() - получить значение элемента в таблице

TTabRecord& operator=() - присвоить ключ элемента

virtual int operator==() - сравнить ключ элемента

virtual int operator< () - сравнить значение элемента

virtual int operator> () - сравнить значение элемента

2. **ttable.h** - абстрактный базовый класс, содержит спецификации методов таблицы

class TTable

Библиотеки и заголовочные файлы:

#include "trecord.h"

#include <iostream>

Методы:

GetDataCount() - получить текущее количество элементов в таблице

GetEfficiency() - показатель эффективности выполнения операции

IsEmpty() - проверка на пустоту таблицы

IsFull() - проверка на заполненность таблицы

FindRecord() - поиск элементов в таблице

InsRecord() - вставка элементов в таблицу

DelRecord() - удаление элементов из таблицы

Reset() - установить на первую запись

IsTabEnded() - завершена ли таблица

GoNext() - переход к следующей записи

GetKey() - получить ключ элемента

GetValuePtr() - получить значение элемента

Print() - вывод таблицы

#### 

3. **arraytab.h** - заголовочный файл с классом, обеспечивающим реализацию просматриваемых таблиц

class TArrayTable

Библиотеки и заголовочные файлы:

#include "ttable.h"

Методы:

int GetTabSize() - размер таблицы

virtual int SetCurrentPos() - установить позицию на запись в таблице

int GetCurrentPos() - получить указатель на текущую позицию

4. **sorttable.h** - модуль с классом, обеспечивающим реализацию упорядоченных таблиц

class TSortTable

Библиотеки и заголовочные файлы:

#include "arraytab.h"

Методы:

void MergeData() - передача во вспомогательную таблицу

void QuickSort() - быстрая сортировка

void QuickSplit() - вспомогательная функция для быстрой сортировки

void Merge() - слияние двух таблиц

5. **treetab.h** - таблицы со структурой хранения в виде деревьев поиска

class TTreeTable

Библиотеки и заголовочные файлы:

#include "treenode.h"

#include "ttable.h"

#include <stack>

Методы:

PrintTreeTab() - выводит таблицу

6. **treenode.h** - класс для узлов деревьев поиска

class TTreeNode

Библиотеки и заголовочные файлы:

#include <iostream>

Методы:

PTTreeNode GetLeft() - указатель на левое поддерево

PTTreeNode GetRight() - указатель на правое поддерево

7. **hashtab.h** и **arrhash.h** - классы для хэш-таблиц

class THashTable и class TArrayHash

Библиотеки и заголовочные файлы:

#include "ttable.h"

Методы:

HashFunc() - непосредственно хэш-функция

8. **main.cpp** - главный исполнительный файл, реализует интерфейс

## **Заключение**

По итогам данной лабораторной работы были написаны 4 вида таблиц, а также методы вставки, удаления и поиска записи для каждой из них. Также была реализована иерархия классов таблиц, изучены особенности выполнения отдельных операций над таблицами и рассмотрена проблема статистической обработки результатов экзаменационной успеваемости студентов. В ходе выполнения работы ознакомился с такими понятиями как хэширование и корреляция.

## 

## Список литературы

<https://github.com/ITrickStar/Table_lab> - реализация лабораторной работы

1. Лабораторный практикум - Барышева И.В., Мееров И.Б., Сысоев А.В., Шестакова Н.В. Под редакцией Гергеля В.П.
2. Искусство программирование. Сортировка и поиск - Дональд Кнут.
3. <https://nuancesprog.ru/p/12140/>
4. <https://habr.com/ru/post/267855/>
5. <https://kvodo.ru/hash-table.html>
6. <http://genius.pstu.ru/file.php/1/pupils_works_2017/MuhinaAlisa.pdf>
7. [https://neerc.ifmo.ru/wiki/index.php?title=Разрешение\_коллизий](https://neerc.ifmo.ru/wiki/index.php?title=%D0%A0%D0%B0%D0%B7%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D0%BB%D0%BB%D0%B8%D0%B7%D0%B8%D0%B9)