Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №17**

Дисциплина: «Основы теории алгоритмов и структуры данных»

Тема: Методы быстрого поиска. Хеширование

Вариант 3

Выполнил:

Студент группы ИВТ-20-2б

Солдатов Алексей Павлович

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель задачи**

Изучить построение функции хеширования и алгоритмов хеширования данных

Научиться разрабатывать алгоритмы открытого и закрытого хеширования при

решении задач на языке C++.

Для достижения поставленной цели необходимо решить следующие задачи:

* Проанализировать теоретические сведения о хеш таблицах
* Проанализировать методы разрешения коллизий
* Продумать ввод баз данных из файла

**Постановка задачи**

1. Создать динамический массив из записей (в соответствии с вариантом),

содержащий не менее 100 элементов. Для заполнения элементов массива

использовать ДСЧ.

2. Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для поиска использовать хеш-таблицу.

3. Подсчитать количество коллизий при размере хеш-таблицы 40, 75 и 90

элементов.

**Анализ задачи**

1. Определить какие операции должны быть выполнены по заданию:

* Создание хеш функции

int Hash\_rem(string s, int n)

{

int l;

l = s.size();

return (l % n);

}

* Создание базы данных

people\* data\_base = new people[len];

string\* key = new string[len];

* Подсчет и разрешение коллизий методом открытой адресации

if (menu == 2)

Shell\_Sort(city, size);

if (menu == 1)

Hoare(city, left, size - 1);

* Реализация алгоритма прямого поиска

1. Для решения задачи используются переменные:

Несколько целочисленных переменных для управления циклами, контроля размеров массивов и контроля длины строк

int len = 0;

int adress;

people x;

string user\_search;

int error = 0;

Указатели

people\* data\_base = new people[len];

string\* key = new string[len];

Структура для ввода и хранения названий городов, численности их населения

struct people

{

string fio;

string cardnum;

string summ;

};

1. Ввод данных осуществляется посредством функции

getline(F1, x.fio);

Вывод данных осуществляется посредством функции

cout << i << ": " << key[i] << endl;

1. Для решения задачи будут использованы циклы. Например, цикл для ввода информации в структуры

for (int i = 0; i < size; i++)

{

cout << "Город: ";

cin >> city[i].name;

cout << "Численость населения: ";

cin >> city[i].population;

}

1. Для защиты от случайных ошибок используются циклы на корректный ввод информации

while (size < 1)

{

cout << "Введите кол-во городов: ";

cin >> size;

cout << endl;

}

while (menu != 1 && menu != 2)

{

cout << "Выбери метод сортировки: \n 1. Хоара \n 2. Шелла \n Ввод: ";

cin >> menu;

}

**Код**

#include <iostream>

#include <list>

#include <fstream>

#include <windows.h>

#include <math.h>

#include <string>

using namespace std;

struct people

{

string fio;

string cardnum;

string summ;

};

int Hash\_rem(string s, int n)

{

int l;

l = s.size();

return (l % n);

}

int main()

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

int len = 0;

int adress;

people x;

string user\_search;

int error = 0;

while (len < 2 || len>100)

{

cout << "Введите размер таблицы: ";

cin >> len;

}

const int N = len;

cin.get();

people\* data\_base = new people[len];

string\* key = new string[len];

for (int i = 0; i < len; i++) key[i] = "Пусто";

ifstream F1("F1.txt");

ifstream F2("F2.txt");

ifstream F3("F3.txt");

for (int i = 0; i < N; i++)

{

getline(F1, x.fio);

getline(F2, x.cardnum);

getline(F3, x.summ);

adress = Hash\_rem(x.fio, N);

while (adress >= len || key[len - 1] != "Пусто")

{

people\* data\_base\_new = new people[len \* 2];

string\* key\_new = new string[len \* 2];

for (int i = 0; i < len; i++)

{

data\_base\_new[i] = data\_base[i];

key\_new[i] = key[i];

}

for (int i = len; i < len \* 2; i++) key\_new[i] = "Пусто";

delete[] data\_base;

delete[] key;

data\_base = data\_base\_new;

key = key\_new;

len \*= 2;

}

if (key[adress] != "Пусто")

{

error++;

while (key[adress] != "Пусто") adress++;

cout << endl;

}

data\_base[adress] = x;

key[adress] = x.fio;

}

cout << "Таблица создана. Коллизий: " << error << endl;

for (int i = 0; i < len; i++)

{

cout << i << ": " << key[i] << endl;

}

cout << "Введите запрос\n";

getline(cin, user\_search);

adress = Hash\_rem(user\_search, N);

while (adress < len)

{

if (key[adress] == user\_search)

{

cout << "Объект найден:\n";

cout << "ФИО- " << data\_base[adress].fio << endl;

cout << "Номер счета- " << data\_base[adress].cardnum << endl;

cout << "Сумма- " << data\_base[adress].summ << endl;

break;

}

if (key[adress] == "Пусто")

{

cout << "Неизвестный объект [Пустая ячейка]\n";

break;

}

adress++;

}

if (adress == len) cout << "Неизвестный объект [Пустая ячейка]\n";

}

**Блок-схема**



**Работа кода**

