Министерство образования и науки Российской Федерации

Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №17**

Дисциплина: «Основы теории алгоритмов и структуры данных»

Тема: Методы быстрого поиска: Хеширование

Вариант 18

Выполнил:

Студент группы РИС-20-1б

Тараканов Д. М.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

Пермь

2021 год

# **Цель работы**

Получить практические навыки работы с методом быстрого поиска.

# **Постановка задачи**

1. Создать динамический массив из записей (в соответствии с вариантом), содержащий не менее 100 элементов.
2. Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для поиска использовать хэш-таблицу.
3. Подсчитать количество коллизий.

(18 вариант)

1. Данные: ФИО, дата\_рождения, №телефона.
2. Ключ: №телефона.
3. Хэш-функция: M\*(k\*Amod10).
4. Метод расширения: метод открытой адресации.

# **Анализ задачи**

1. Определить какие действия необходимо выполнить для решения задачи:

* Разработать структуру People которая хранит в себе данные (ФИО, дата\_рождения, № телефона).

struct People{

string fio;

string birth\_data;

string number;

};

* Разработать хэш функцию для заполнения хэш таблицы.

int HashFunction(string k, int n) {

unsigned long long int key = stoll(k);

int M = 12/n;

int A = 3;

return M \* (A \* key % 10);

}

* Реализовать метод расширения хэш таблицы – метод открытой адресации.

if ((key[index] != "-") && (index < size-1)) {

count++;

while ((key[index] != "-") && (index < size-1)) {

index++;

}

}

* Реализовать считывание данных из файла и добавление их в таблицу.

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

getline(input, el.fio);

getline(input, el.birth\_data);

getline(input, el.number);

index = HashFunction(el.number, N);

while ((index >= size) || (key[size - 1] != "-")) {

People\* dataR = new People[size\*2];

string\* keyR = new string[size\*2];

for (int j = 0; j < size; j++) {

dataR[j] = data[j];

keyR[j] = key[j];

}

for (int j = size; j < size \* 2; j++) {

keyR[j] = "-";

}

delete[]data;

delete[]key;

data = dataR;

key = keyR;

size = size \* 2;

}

if ((key[index] != "-") && (index < size-1)) {

count++;

while ((key[index] != "-") && (index < size-1)) {

index++;

}

}

data[index] = el;

key[index] = el.number;

}

1. С какими типами данных действие надо сделать, в каком виде эти данные будут представлены:

* Для хранения ФИО, даты рождения, № телефона в структуре реализованы переменные fio, birth\_data, number.

string fio;

string birth\_data;

string number;

* Для хранения данных реализован динамический массив data типа People.

People\* data = new People[N];

* Для хранения ключей реализован динамический массив key типа string.

string\* key = new string[N];

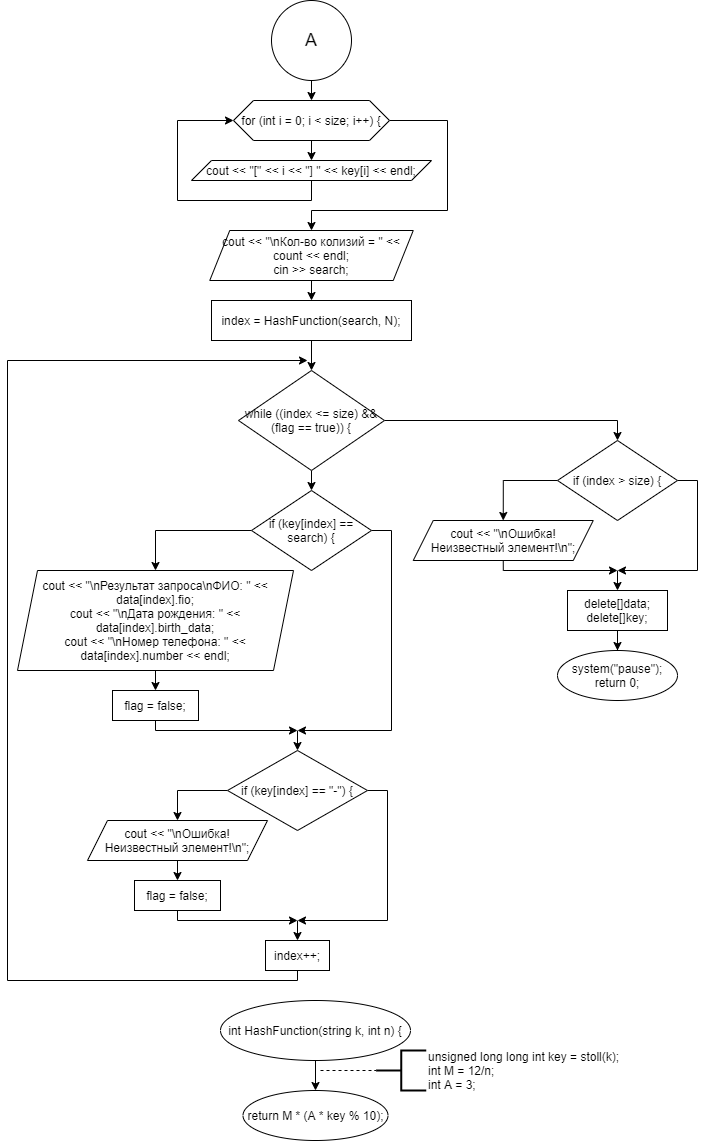
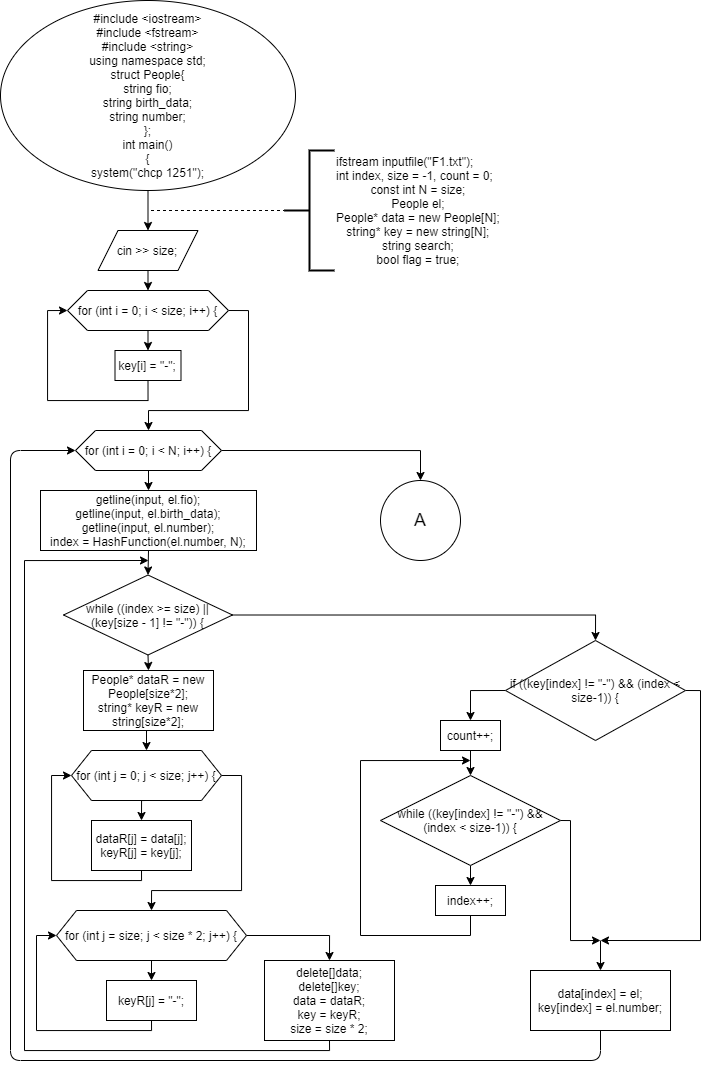
* Для чтения данных из файла реализована переменная inputfile типа ifstream

ifstream inputfile("F1.txt");

* Для хранения ключа поиска реализована переменная search типа string.

string search;

**Блок схема**



# **Код программы на языке C++**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct People{

string fio;

string birth\_data;

string number;

};

//H(k) = [M(kAmod1)]

int HashFunction(string k, int n) {

unsigned long long int key = stoll(k);

int M = 12/n;

int A = 3;

return M \* (A \* key % 10);

}

int main() {

system("chcp 1251");

ifstream input("F1.txt");

if (input.is\_open()) {

int index, size = -1, count = 0;

while ((size < 2) || (size > 10)) {

cout << "\nВведите размер таблицы:\n";

cin >> size;

}

const int N = size;

People el;

People\* data = new People[N];

string\* key = new string[N];

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

key[i] = "-";

}

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

getline(input, el.fio);

getline(input, el.birth\_data);

getline(input, el.number);

index = HashFunction(el.number, N);

while ((index >= size) || (key[size - 1] != "-")) {

People\* dataR = new People[size\*2];

string\* keyR = new string[size\*2];

for (int j = 0; j < size; j++) {

dataR[j] = data[j];

keyR[j] = key[j];

}

for (int j = size; j < size \* 2; j++) {

keyR[j] = "-";

}

delete[]data;

delete[]key;

data = dataR;

key = keyR;

size = size \* 2;

}

if ((key[index] != "-") && (index < size-1)) {

count++;

while ((key[index] != "-") && (index < size-1)) {

index++;

}

}

data[index] = el;

key[index] = el.number;

}

cout << "\nХэш таблица:\n";

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

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

}

cout << "\nКол-во колизий = " << count << endl;

string otvet = "Да";

while (otvet != "Нет") {

string search;

bool flag = true;

cout << "\nВведите запрос:\n";

cin >> search;

index = HashFunction(search, N);

while ((index < size) && (flag == true)) {

if (key[index] == search) {

cout << "\nРезультат запроса\nФИО: " << data[index].fio;

cout << "\nДата рождения: " << data[index].birth\_data;

cout << "\nНомер телефона: " << data[index].number << endl;

flag = false;

}

if (key[index] == "-") {

cout << "\nОшибка! Неизвестный элемент!\n";

flag = false;

}

index++;

}

if (index == size) {

cout << "\nОшибка! Неизвестный элемент!\n";

}

cout << "\nПродолжить? (Да/Нет)\n";

cin >> otvet;

}

delete[]data;

delete[]key;

}

else {

cout << "\nОшибка! Не удалось открыть файл F1.txt\n";

}

system("pause");

return 0;

}

# **Скриншоты тестов**

