Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ПНИПУ)

Электротехнический факультет

Дисциплина: основы алгоритмизации и программирования, 2 семестр

ОТЧЁТ

«Лабораторная работа. Хэши.»

Выполнил

Студент группы РИС-22-2Б

Хохряков А.С.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Пермь 2023

# Содержание отчета

1. Постановка задачи.
2. Анализ задачи.
3. Блок-схема
4. Текст программы

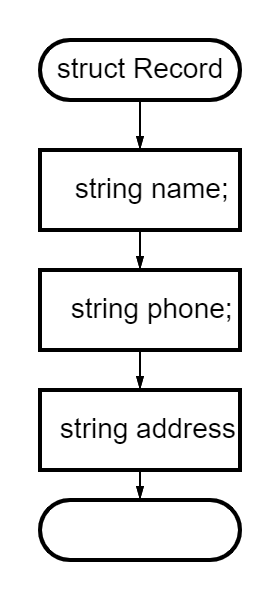
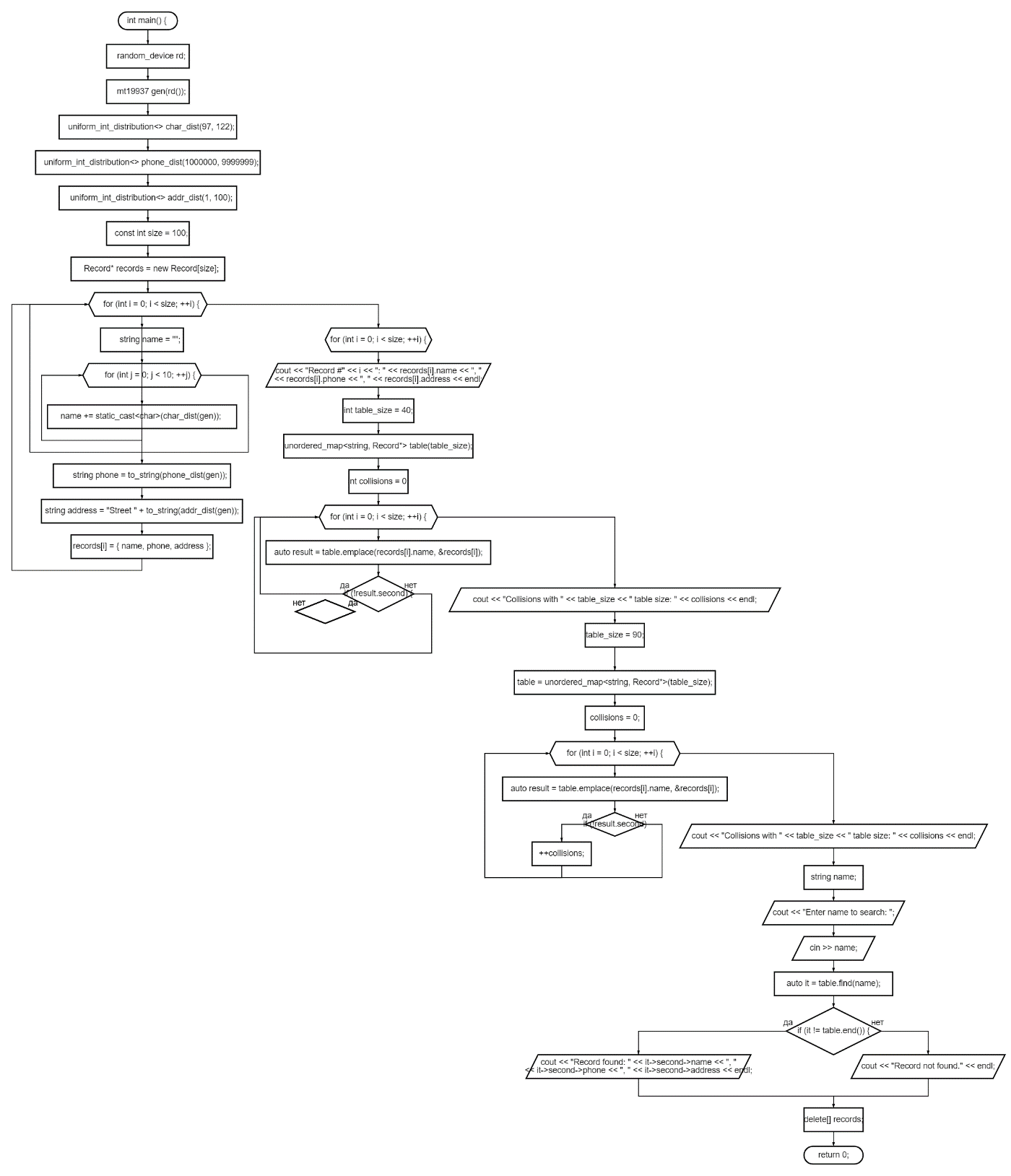
Постановка задачи

Создать динамический массив из записей: ФИО, номер телефона, адрес, содержащий не менее 100 элементов. Для заполнения элементов массива использовать ДСЧ. Выполнить поиск элемента в массиве по ключу ФИО. Для поиска использовать хэш-таблицу. Подсчитать количество коллизий при размере хэш-таблицы 40, 75 и 90 элементов.

Анализ задачи

1. Создание динамического массива из записей типа Record и заполнение его случайными данными (случайные строки для имени, случайное число для телефона и случайное число для адреса).
2. Вывод содержимого массива на экран.
3. Создание хэш-таблицы с использованием STL-класса unordered\_map. В данном случае хэш-функцией выступает хэш от имени (Record.name), а значением в таблице является указатель на соответствующую запись типа Record.
4. Заполнение хэш-таблицы данными из массива записей с подсчетом коллизий (Collisions).
5. Поиск элемента в хэш-таблице по имени.
6. Освобождение памяти, занятой массивом.

Блок-Схема



Текст программы

#include <iostream>

#include <string>

#include <random>

#include <unordered\_map>

using namespace std;

struct Record {

string name;

string phone;

string address;

};

int main() {

random\_device rd;

mt19937 gen(rd());

uniform\_int\_distribution<> char\_dist(97, 122);

uniform\_int\_distribution<> phone\_dist(1000000, 9999999);

uniform\_int\_distribution<> addr\_dist(1, 100);

const int size = 100;

Record\* records = new Record[size];

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

string name = "";

for (int j = 0; j < 10; ++j) {

name += static\_cast<char>(char\_dist(gen));

}

string phone = to\_string(phone\_dist(gen));

string address = "Street " + to\_string(addr\_dist(gen));

records[i] = { name, phone, address };

}

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

cout << "Record #" << i << ": " << records[i].name << ", "

<< records[i].phone << ", " << records[i].address << endl;

}

int table\_size = 40;

unordered\_map<string, Record\*> table(table\_size);

int collisions = 0;

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

auto result = table.emplace(records[i].name, &records[i]);

if (!result.second) {

++collisions;

}

}

cout << "Collisions with " << table\_size << " table size: " << collisions << endl;

table\_size = 75;

table = unordered\_map<string, Record\*>(table\_size);

collisions = 0;

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

auto result = table.emplace(records[i].name, &records[i]);

if (!result.second) {

++collisions;

}

}

cout << "Collisions with " << table\_size << " table size: " << collisions << endl;

table\_size = 90;

table = unordered\_map<string, Record\*>(table\_size);

collisions = 0;

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

auto result = table.emplace(records[i].name, &records[i]);

if (!result.second) {

++collisions;

}

}

cout << "Collisions with " << table\_size << " table size: " << collisions << endl;

string name;

cout << "Enter name to search: ";

cin >> name;

auto it = table.find(name);

if (it != table.end()) {

cout << "Record found: " << it->second->name << ", "

<< it->second->phone << ", " << it->second->address << endl;

}

else {

cout << "Record not found." << endl;

}

delete[] records;

return 0;

}