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 №12, 14**

Дисциплина: «Информатика»

Тема: Алгоритмы поиска. Работа со списками. Метод интерполяционного поиска. Методы поиска: метод Кнутта-Мориса Пратта, метод Бойера-Мура

Вариант 4

Выполнил:

Студент группы РИС-20-1б

Еске Вячеслав Сергеевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Цель работы**

Изучить алгоритмы поиска данных и реализовать их в программе.

**Постановка задачи**

1) Необходимо разработать программу, которая создаёт массивы данных, выполняет поиск по этим массивам и выдаёт результат.

2) Взаимодействие с пользователем производится через меню.

3) В зависимости от команд пользователя программа выполняет поиск данных и выводит результат

**Анализ задачи**

1. Определить какие действия необходимо выполнить для решения задачи.

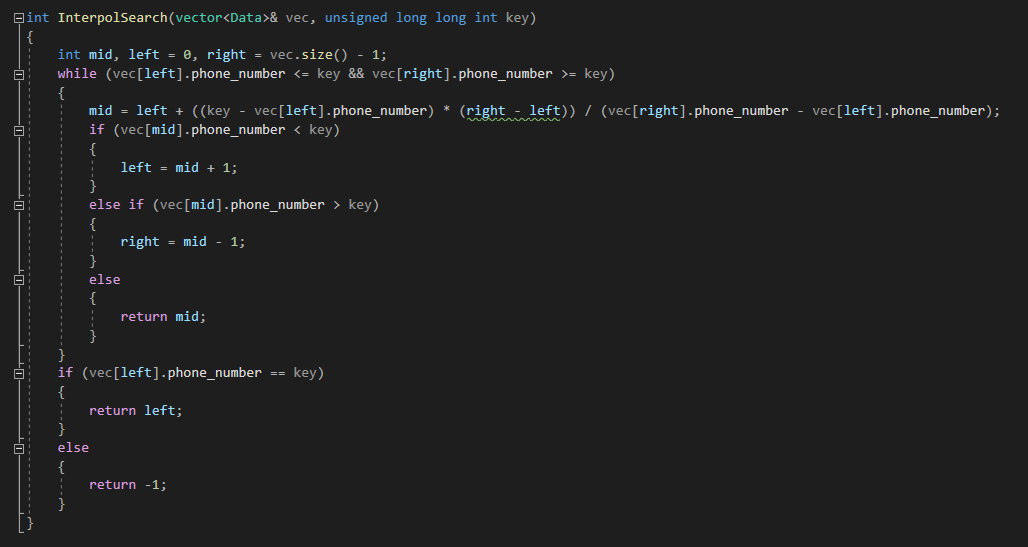
* Добавление и удаление элементов.
* Поиск данных
* Чтение из файла
* Запись в файл
* Вывод в консоль

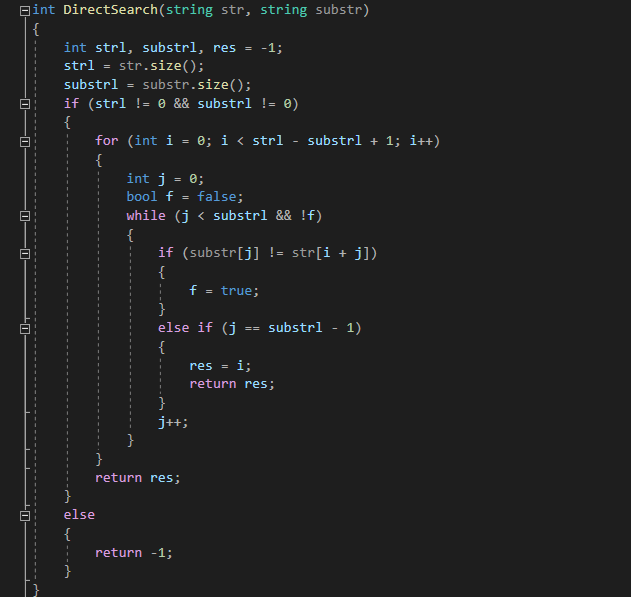
2) Информация из файла хранится в векторе

  
3) Считывание данных производится через цикл while:

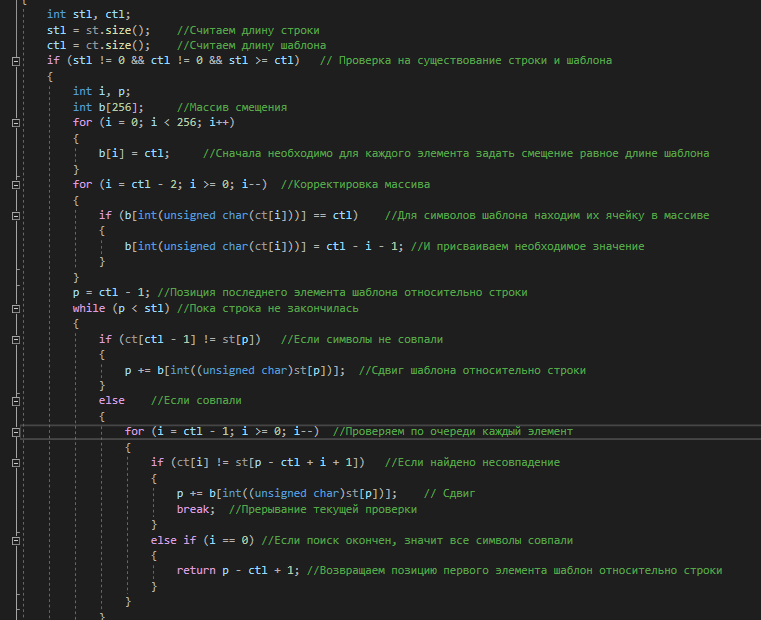


4) Интерполяционный поиск:

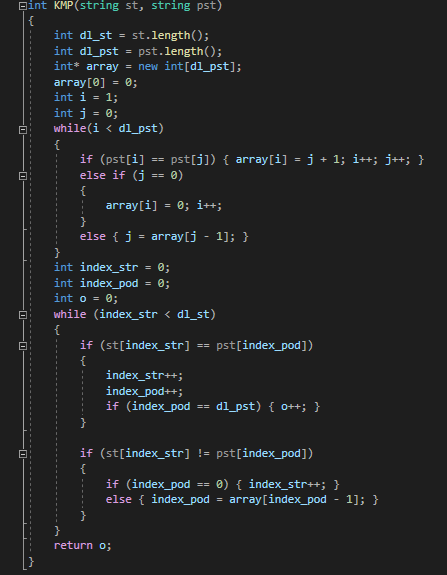
5) Прямой поиск:



6) Алгоритм Боера-Мура



7) Алгоритм Кнута-Морриса-Пратта:



**Код**

**Lab 12**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <iomanip>

#include <string>

using namespace std;

int count;

struct Person

{

char surname[30], name[30], middlename[30];

char address[100];

int pasport;

};

struct dat

{

string fio;

string date;

int pass;

string pass1;

};

void form\_file();

void read\_file();

void work\_file();

void line\_search();

Person create();

int main()

{

system("chcp 1251>null");

int oper;

do

{

cout << "1.Сформировать файл" << endl;

cout << "2.Работа с файлом" << endl;

cout << "3.Выход" << endl;

cin >> oper;

switch (oper)

{

case 1:

form\_file();

break;

case 2:

work\_file();

break;

default:

break;

}

} while (oper != 3);

return 0;

}

Person create()

{

cout << "Фамилия: ";

char surname[30];

cin >> surname;

cout << "Имя: ";

char name[30];

cin >> name;

cout << "Отчество: ";

char middlename[30];

cin >> middlename;

int pasport;

do {

cout << "Паспорт: ";

cin >> pasport;

} while (pasport < 0);

cout << "Адрес: ";

char address[100];

cin.ignore();

gets\_s(address);

Person p;

strcpy(p.address, address);

strcpy(p.middlename, middlename);

strcpy(p.name, name);

strcpy(p.surname, surname);

p.pasport = pasport;

return p;

}

void form\_file()

{

cout << "Кол-во людей = ";

int count;

cin >> count;

FILE\* file = fopen("file.dat", "wb");

if (file == NULL)

exit(1);

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

{

Person p = create();

fwrite(&p, sizeof(Person), 1, file);

if (ferror(file))

exit(2);

}

fclose(file);

}

void read\_file()

{

FILE\* file = fopen("file.dat", "rb");

Person p;

cout << setw(15) << "ФИО" << setw(23) << "Паспорт" << setw(15) << "Адрес" << endl;

while (fread(&p, sizeof(Person), 1, file))

{

char fio[30] = "";

strcat(fio, p.surname);

strcat(fio, " ");

strcat(fio, p.name);

strcat(fio, " ");

strcat(fio, p.middlename);

cout << setw(25) << fio << setw(10) << p.pasport << setw(20) << p.address << endl;

}

fclose(file);

}

void line\_search()

{

FILE\* file = fopen("file.dat", "rb");

Person p;

int key, i = 1;

bool f=false;

cout << "Введите номер паспорта, который необходимо найти: ";

cin >> key;

while (fread(&p, sizeof(Person), 1, file))

{

if (p.pasport == key) {

char fio[30] = "";

strcat(fio, p.surname);

strcat(fio, " ");

strcat(fio, p.name);

strcat(fio, " ");

strcat(fio, p.middlename);

cout << setw(25) << fio << setw(10) << p.pasport << setw(20) << p.address << endl;

f = true;

}

if (f == false) i++;

}

if (f == true) {

cout << "Находится на " << i << " месте." << endl;

}

else cout << "Такого номера нет" << endl;

fclose(file);

}

void interpol()

{

FILE\* file = fopen("file.dat", "rb");

Person p;

int key, i = 1;

bool f = false;

cout << "Введите номер паспорта, который необходимо найти: ";

cin >> key;

while (fread(&p, sizeof(Person), 1, file))

{

if (p.pasport == key) {

char fio[30] = "";

strcat(fio, p.surname);

strcat(fio, " ");

strcat(fio, p.name);

strcat(fio, " ");

strcat(fio, p.middlename);

cout << setw(25) << fio << setw(10) << p.pasport << setw(20) << p.address << endl;

f = true;

}

if (f == false) i++;

}

if (f == true) {

cout << "Находится на " << i << " месте." << endl;

}

else cout << "Такого номера нет" << endl;

fclose(file);

}

void intersearch()

{

int key, num = -1, mid = -1, left = 0, right = 4, i, temp, item;

ofstream F("F.txt");

dat N1 = { "Иванов Иван Иванович", "04.10.2002", 507263, "507263" };

dat N2 = { "Петров Петр Петрович", "18.02.2010", 263598, "263598" };

dat N3 = { "Алексеев Алексей Алексеевич", "06.12.2000", 648531, "648531" };

dat N4 = { "Александров Александр Александрович", "06.10.2002", 984563, "984563" };

dat N5 = { "Максимов Максим Максимович", "30.10.1998", 651234, "651234" };

dat a[5] = { N1, N2, N3, N4, N5 };

for (i = 0; i < 5; i++)

cout << " " << i + 1 << ") " << '\t' << a[i].fio << endl << '\t' << a[i].date << endl << '\t' << a[i].pass << endl << "\n";

cout << "Введи ключ (номер паспорта), по которому будет осуществлен поиск: "; cin >> key;

int a1[5]{ N1.pass, N2.pass, N3.pass, N4.pass, N5.pass };

for (i = 1; i < 5; i++) //Сортируем

{

temp = a1[i];

item = i - 1;

while (item >= 0 && a1[item] > temp)

{

a1[item + 1] = a1[item];

a1[item] = temp;

item--;

}

}

while (a1[left] <= key && a1[right] >= key && a1[mid] != key)

{

mid = left + ((key - a1[left]) \* (right - left)) / (a1[right] - a1[left]);

if (a1[mid] < key)

left = mid + 1;

else

if (a1[mid] > key)

right = mid - 1;

}

if (a1[mid] == key)

F << "Искомый элемент найден";

else F << "Искомый элемент не найден";

}

void pryamoy()

{

FILE\* file = fopen("file.dat", "rb");

Person p;

int key, i = 1;

bool f = false;

cout << "Введите номер паспорта, который необходимо найти: ";

cin >> key;

while (fread(&p, sizeof(Person), 1, file))

{

if (p.pasport == key) {

char fio[30] = "";

strcat(fio, p.surname);

strcat(fio, " ");

strcat(fio, p.name);

strcat(fio, " ");

strcat(fio, p.middlename);

cout << setw(25) << fio << setw(10) << p.pasport << setw(20) << p.address << endl;

f = true;

}

if (f == false) i++;

}

if (f == true) {

cout << "Находится на " << i << " месте." << endl;

}

else cout << "Такого номера нет" << endl;

fclose(file);

}

void add\_person(Person t, int pos)

{

if (pos < 1)

cout << "Некорректный номер" << endl;

else

{

FILE\* file = fopen("file.dat", "rb");

FILE\* tempfile = fopen("temp.dat", "wb");

Person p;

int index = 0;

while (fread(&p, sizeof(Person), 1, file))

{

fwrite(&p, sizeof(Person), 1, tempfile);

index++;

if (index == pos)

fwrite(&t, sizeof(Person), 1, tempfile);

}

fclose(file);

fclose(tempfile);

remove("file.dat");

rename("temp.dat", "file.dat");

if (index < pos)

cout << "Некорректный номер" << endl;

}

}

void work\_file()

{

int oper;

do

{

cout << "1.Чтение файла" << endl;

cout << "2.Линейный поиск по паспорту" << endl;

cout << "3.Интерполяционный поиск по паспорту" << endl;

cout << "4.Прямой поиск" << endl;

cout << "5.Добавление в файл" << endl;

cout << "0.Назад" << endl;

cin >> oper;

switch (oper)

{

case 1:

read\_file();

break;

case 2:

{

line\_search();

break;

}

case 3:

{

pryamoy();

break;

}

case 4:

{

interpol();

break;

}

case 5:

int pos;

do {

cout << "После какого элемента добавить? ";

cin >> pos;

} while (pos < 0);

Person p = create();

add\_person(p, pos);

break;

}

} while (oper != 0);

}

**Lab 14**

#include <iostream>

#include <fstream>

#include <string>

#include <ctime>

using namespace std;

struct Person {

string name;

string surname;

string midlename;

string adress;

string passport;

};

int BMura(string st, string ct)

{

int stl, ctl;

stl = st.size();

ctl = ct.size();

if (stl != 0 && ctl != 0 && stl >= ctl)

{

int i, p;

int b[256];

for (i = 0; i < 256; i++)

{

b[i] = ctl;

}

for (i = ctl - 2; i >= 0; i--)

{

if (b[int(unsigned char(ct[i]))] == ctl)

{

b[int(unsigned char(ct[i]))] = ctl - i - 1;

}

}

p = ctl - 1;

while (p < stl)

{

if (ct[ctl - 1] != st[p])

{

p += b[int((unsigned char)st[p])];

}

else

{

for (i = ctl - 1; i >= 0; i--)

{

if (ct[i] != st[p - ctl + i + 1])

{

p += b[int((unsigned char)st[p])];

break;

}

else if (i == 0)

{

return p - ctl + 1;

}

}

}

}

}

return -1;

}

void set\_studentsrandom(Person chel[], int start, int scale) {

int j, k;

for (int i = start; i < scale; i++) {

k = 0;

ifstream name1("C:\\Users\\super\\Desktop\\name.txt");

ifstream surname1("C:\\Users\\super\\Desktop\\surname.txt");

ifstream midlename1("C:\\Users\\super\\Desktop\\midlename.txt");

string group, temp, surname, surnameme, temp1, temp2;

int count = -1;

int rnd = rand() % 99;

while (count != rnd)

{

getline(name1, temp);

getline(surname1, temp1);

getline(midlename1, temp2);

count++;

}

chel[i].name = temp;

chel[i].surname = temp1;

chel[i].midlename = temp2;

name1.close();

surname1.close();

midlename1.close();

ifstream adress("C:\\Users\\super\\Desktop\\adress.txt");

rnd = rand() % 11;

count = -1;

while (count != rnd)

{

getline(adress, temp);

count++;

}

chel[i].adress = temp;

adress.close();

ifstream passport("C:\\Users\\super\\Desktop\\passport.txt");

rnd = rand() % 100;

count = -1;

while (count != rnd)

{

getline(passport, temp);

count++;

}

chel[i].passport = temp;

passport.close();

}

}

void show\_students(Person chel[], int start, int scale) {

for (int i = start; i < scale; i++) {

cout << "\nФИО: " << chel[i].surname << " " << chel[i].name << " " << chel[i].midlename;

cout << "\nАдрес: " << chel[i].adress;

cout << "\nПаспорт: " << chel[i].passport << "\n";

}

}

void KMP\_Search(Person\* chel, string x, string row, int size)

{

int\* arr\_pos = new int[size];

string pst=x;

int pos=0;

int dl\_st = row.length();

int dl\_pst = pst.length();

int\* array = new int[dl\_pst];

array[0] = 0;

int i = 1;

int j = 0;

for (int v = 0; v < dl\_pst; v++)

{

if (pst[i] == pst[j]) { array[i] = j + 1; i++; j++; }

else if (j == 0)

{

array[i] = 0; i++;

}

else { j = array[j - 1]; }

}

int index\_str = 0;

int index\_pod = 0;

int slovo = 0;

int o = 0;

while (index\_str < dl\_st)

{

if (row[index\_str] == pst[index\_pod])

{

index\_str++;

index\_pod++;

if (index\_pod == dl\_pst)

{

arr\_pos[o] = slovo;

o++;

}

}

if (row[index\_str] != pst[index\_pod])

{

if (index\_pod == 0) { index\_str++; }

else { index\_pod = array[index\_pod - 1]; }

}

if (row[index\_str] == ' ') slovo++;

}

cout << endl << "Совпадений: " << o << endl;

for (i = 0; i < o; i++)

{

cout << "\nФИО: " << chel[arr\_pos[i]].surname << " " << chel[arr\_pos[i]].name << " " << chel[arr\_pos[i]].midlename;

cout << "\nАдрес: " << chel[arr\_pos[i]].adress;

cout << "\nПаспорт: " << chel[arr\_pos[i]].passport << "\n";

}

}

int main()

{

system("chcp 1251>nul");

srand(time(NULL));

int size;

string passport;

int i;

do {

cout << "Введите количество людей:";

cin >> size;

} while (size <= 0);

cout << endl;

string row;

Person\* chel = new Person[size];

set\_studentsrandom(chel, 0, size);

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

{

row += chel[i].passport + " ";

}

show\_students(chel, 0, size);

cout << endl;

cout << "Поиск Кнутта-Морриса-Пратта " << endl;

cout << "\nПаспорт для поиска: ";

cin >> passport;

KMP\_Search(chel, passport, row, size);

cout << endl;

cout << "Поиск Бойра-Мюра " << endl;

cout << "\nПаспорт для поиска: ";

cin >> passport;

int pos;

int k = 0;

string row\_dop;

while (BMura(row, passport) != -1)

{

pos = BMura(row, passport);

for (i = 0; i < pos; i++)

{

if (row[i] == ' ') k++;

}

cout << "\nФИО: " << chel[k].surname << " " << chel[k].name << " " << chel[k].midlename;

cout << "\nАдрес: " << chel[k].adress;

cout << "\nПаспорт: " << chel[k].passport << "\n";

for (i = 0; i < pos; i++)

{

row[i]=0;

}

while (row[i] != ' ')

{

pos++;

row[i] = 0;

i++;

}

}

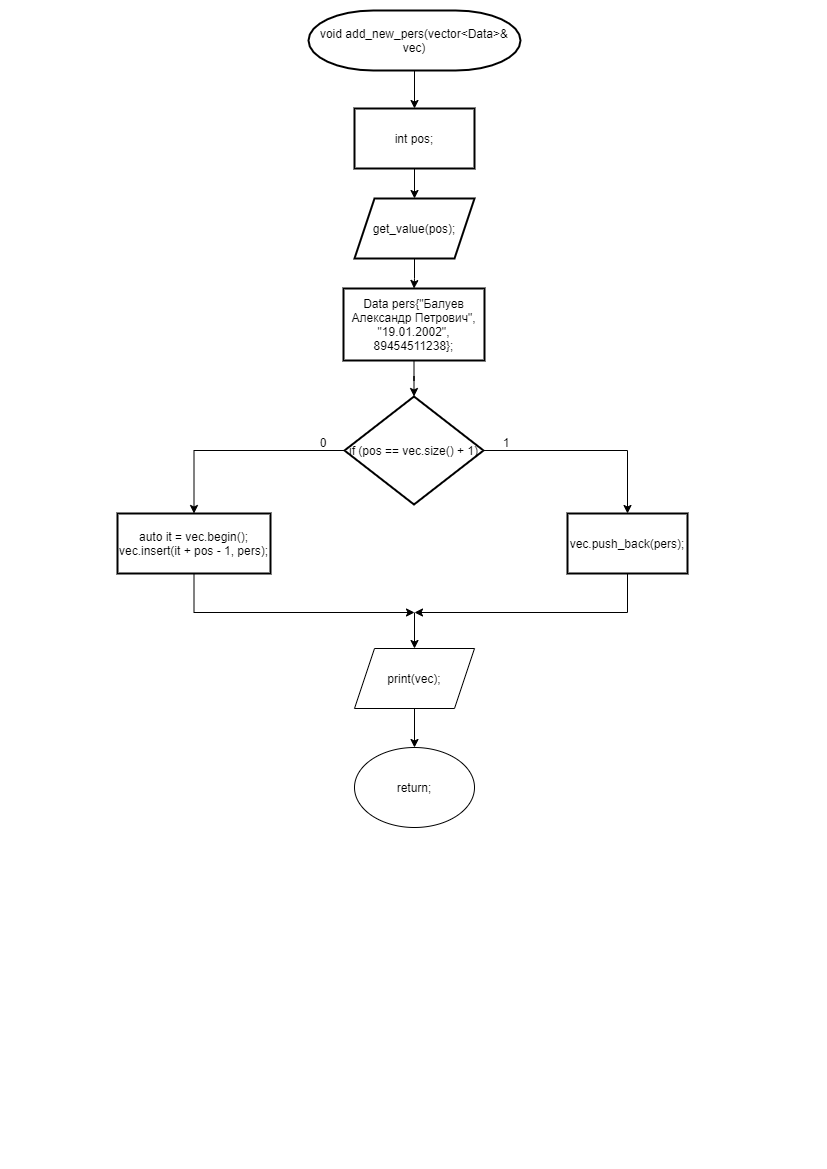
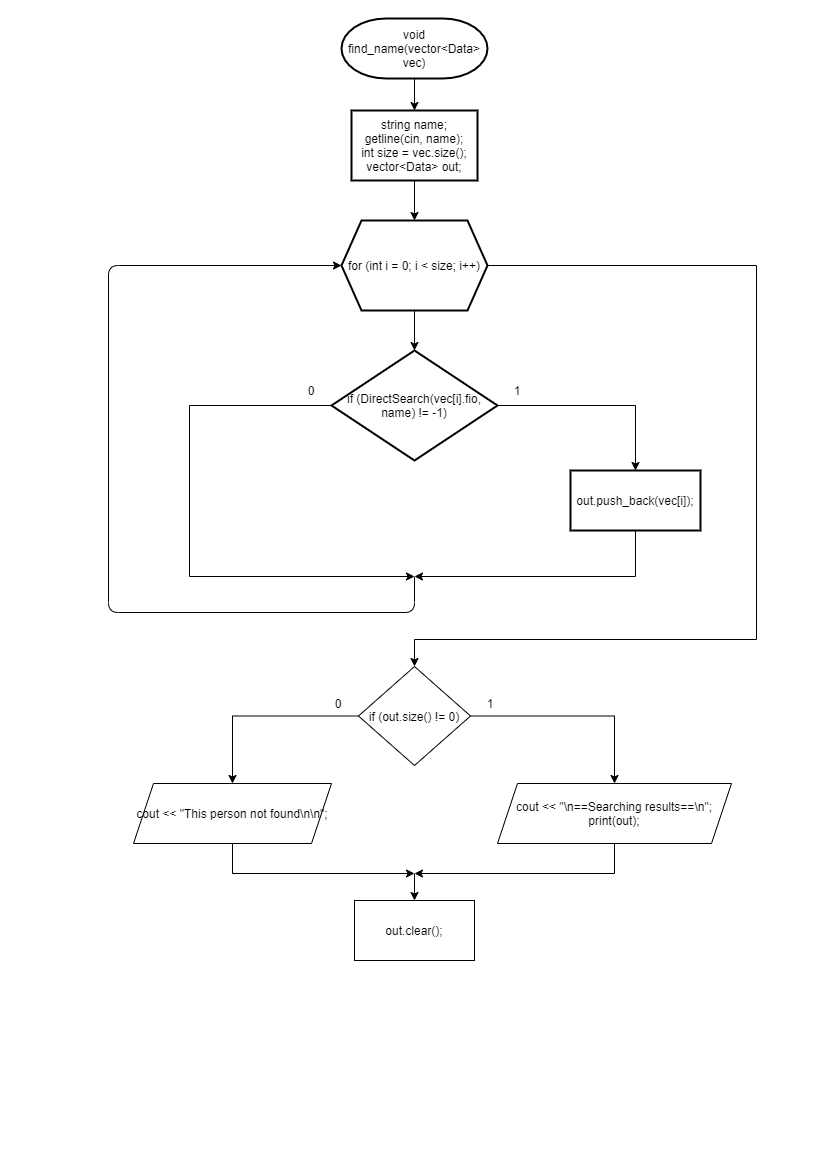
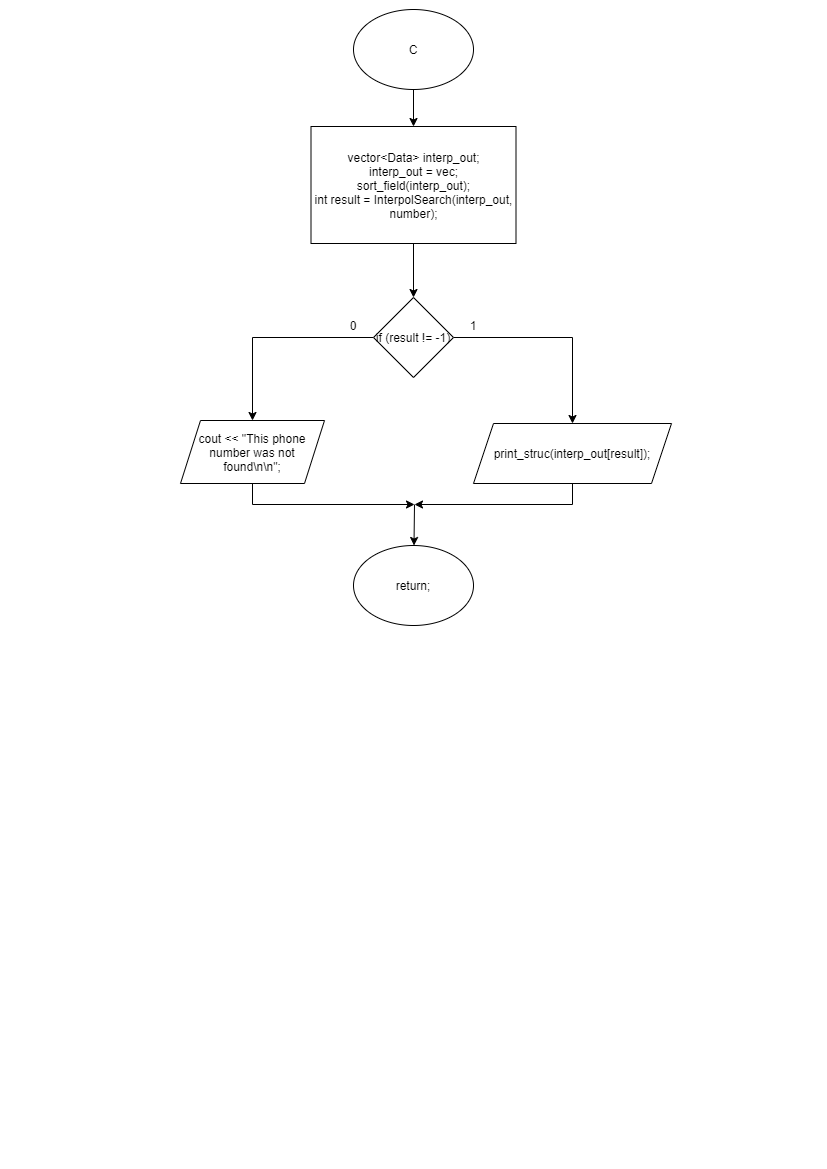
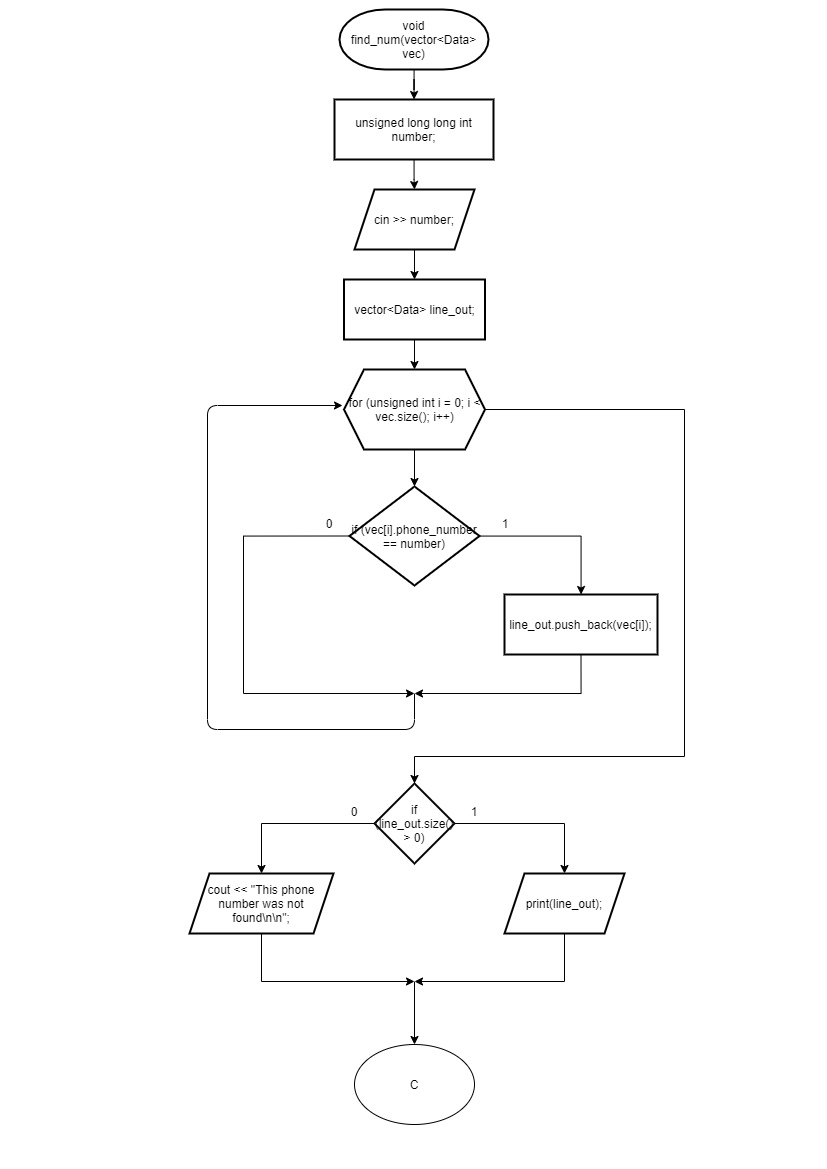
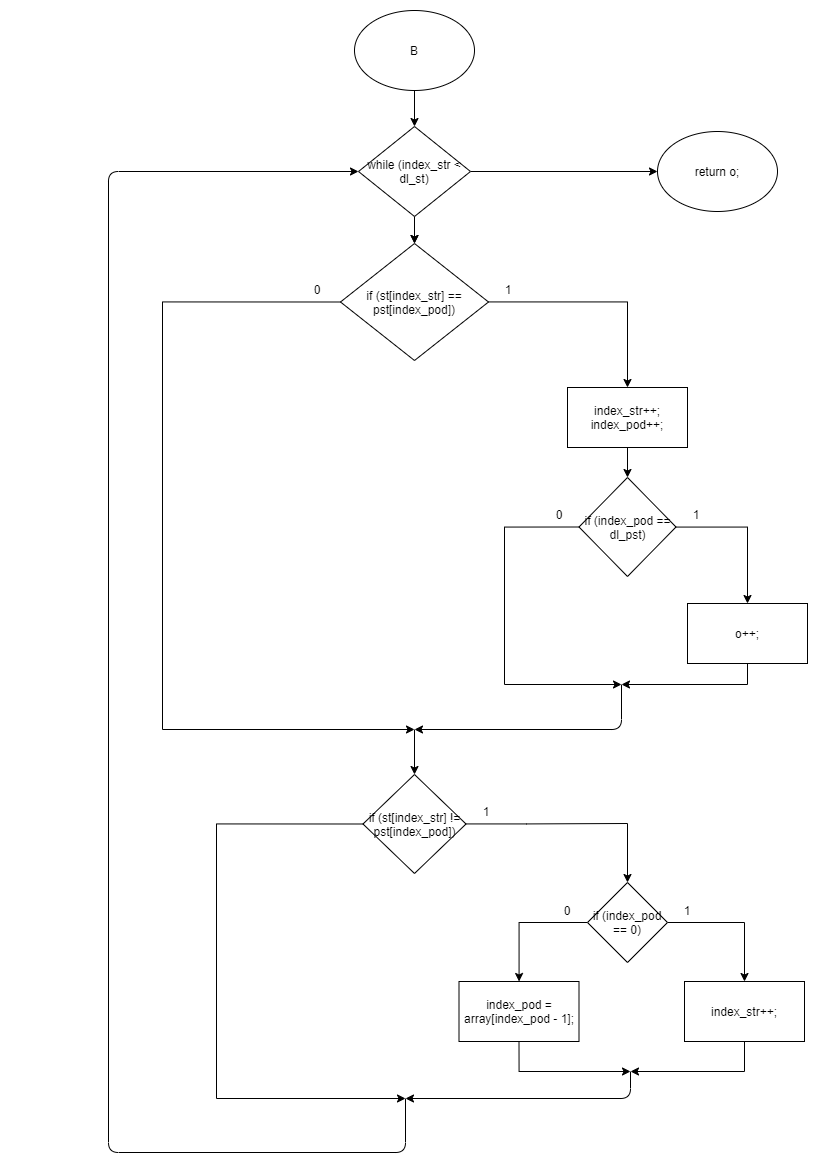
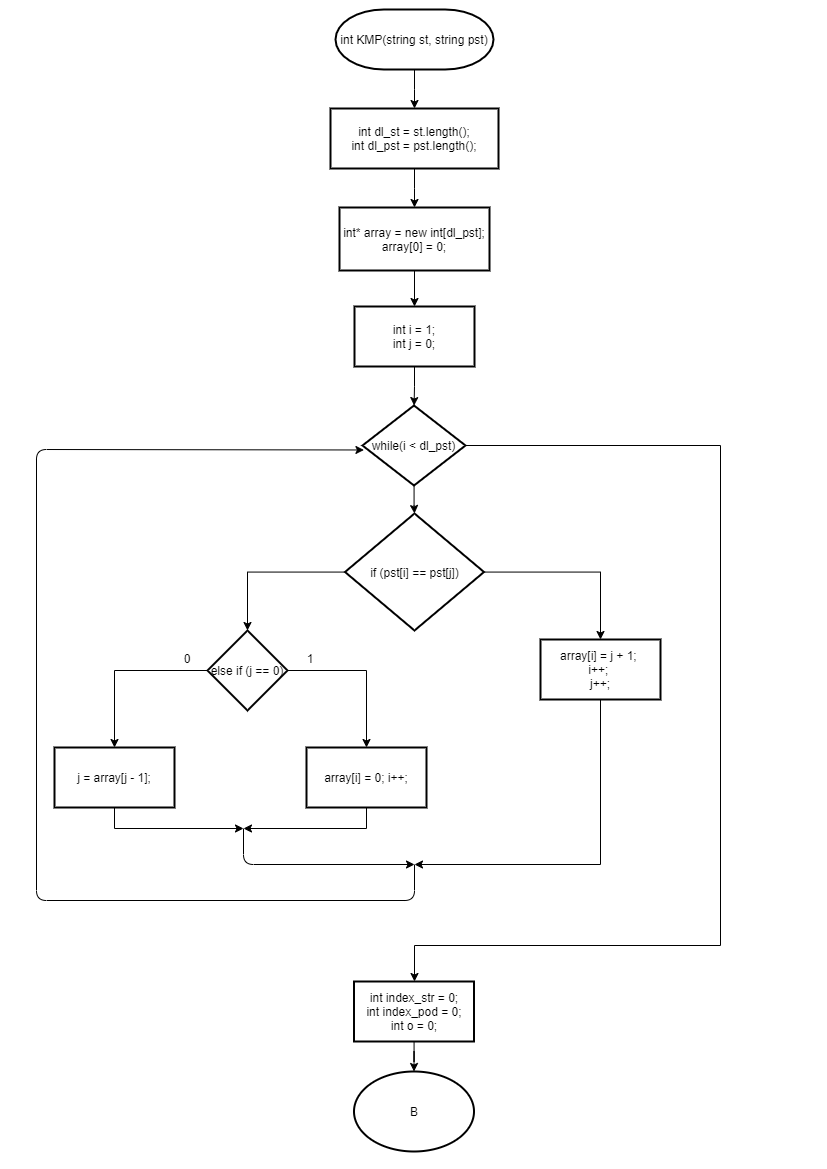
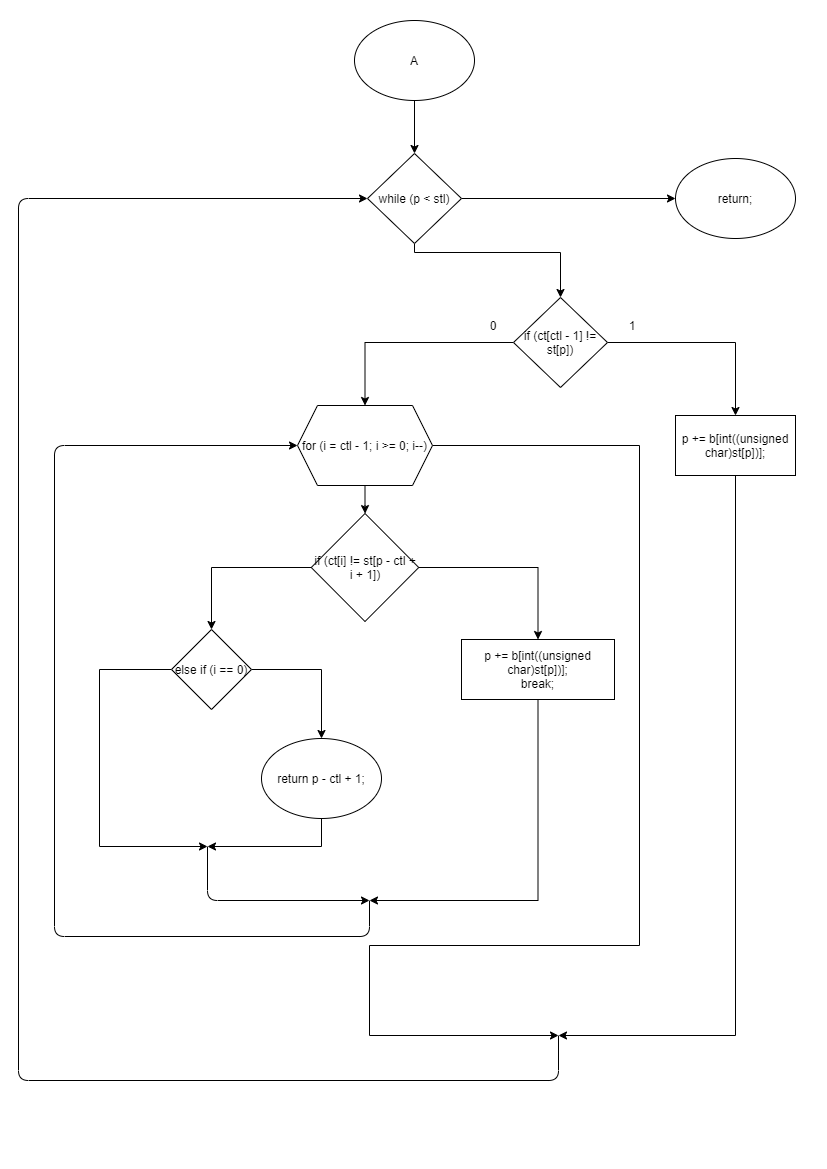
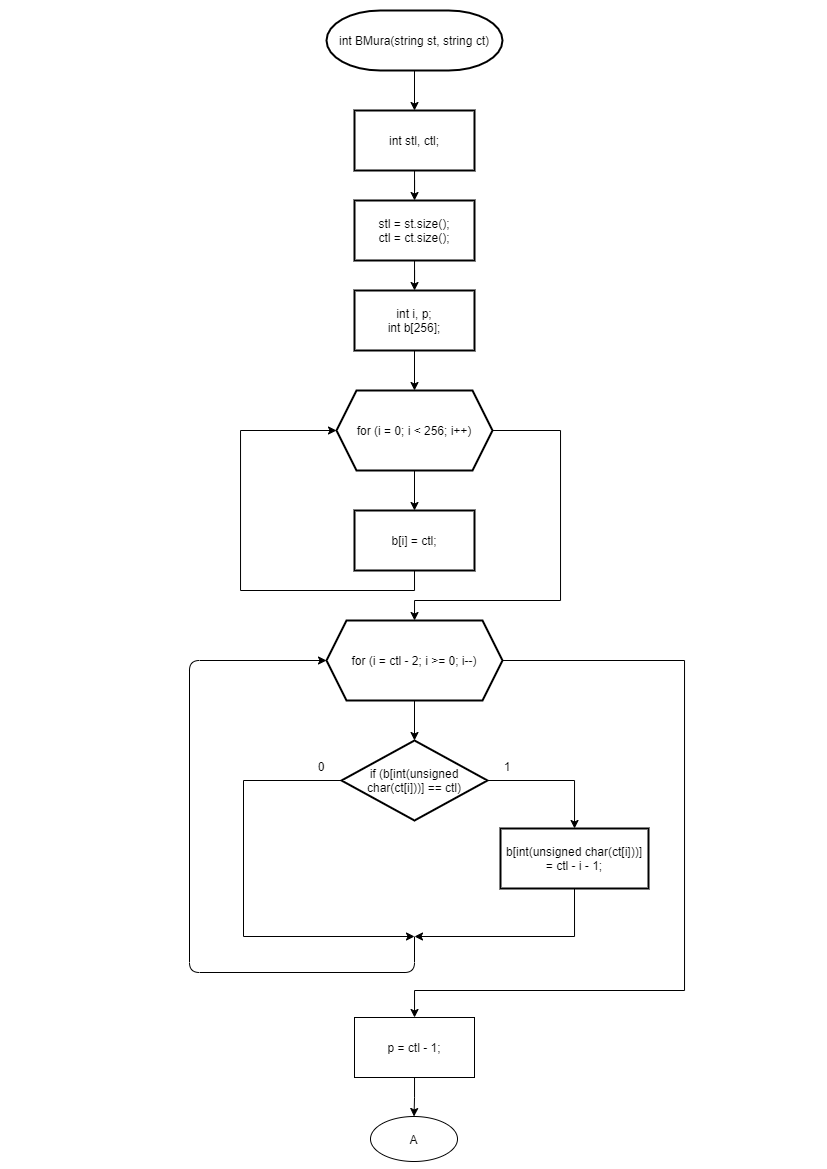
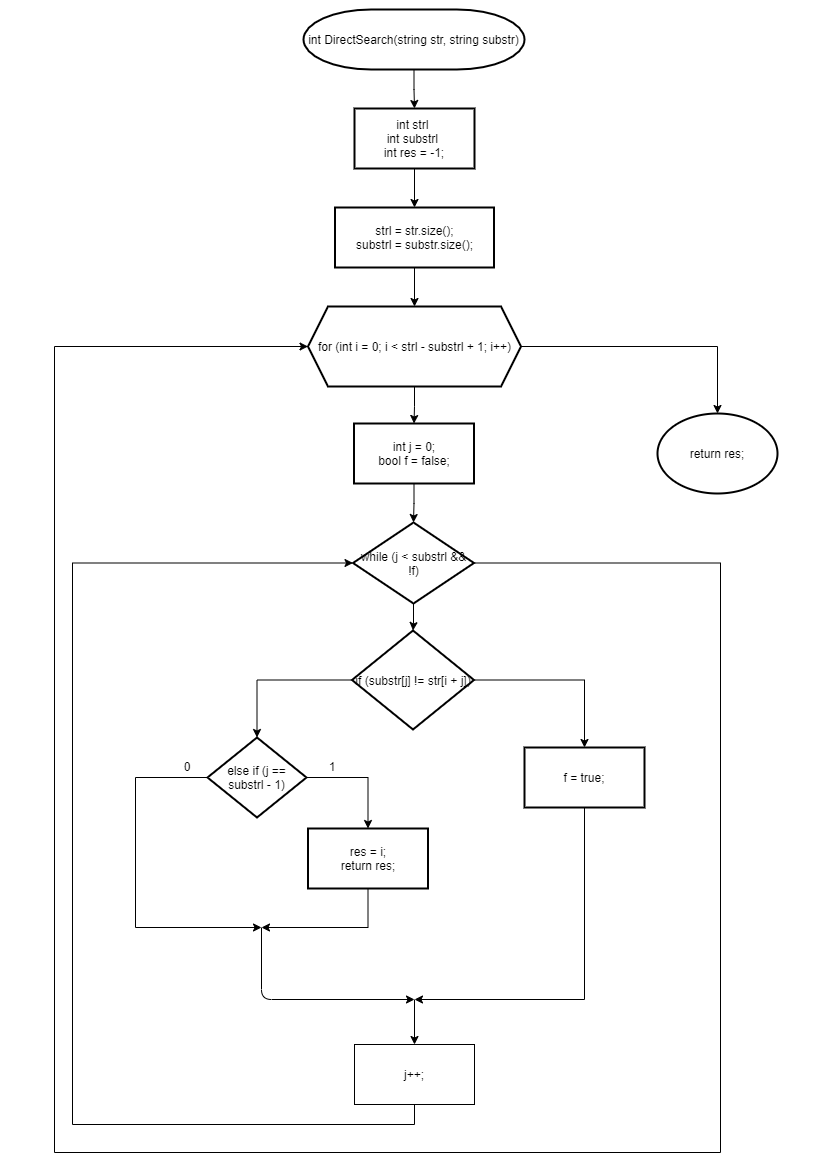
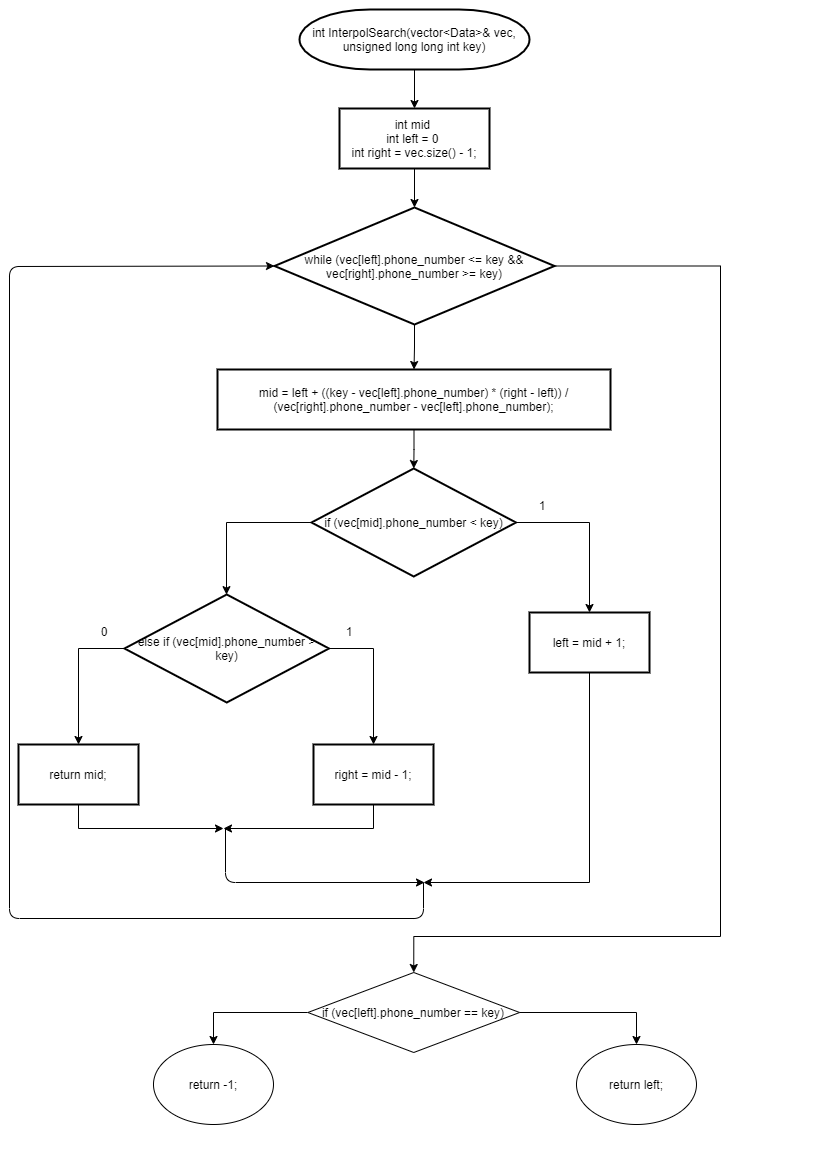
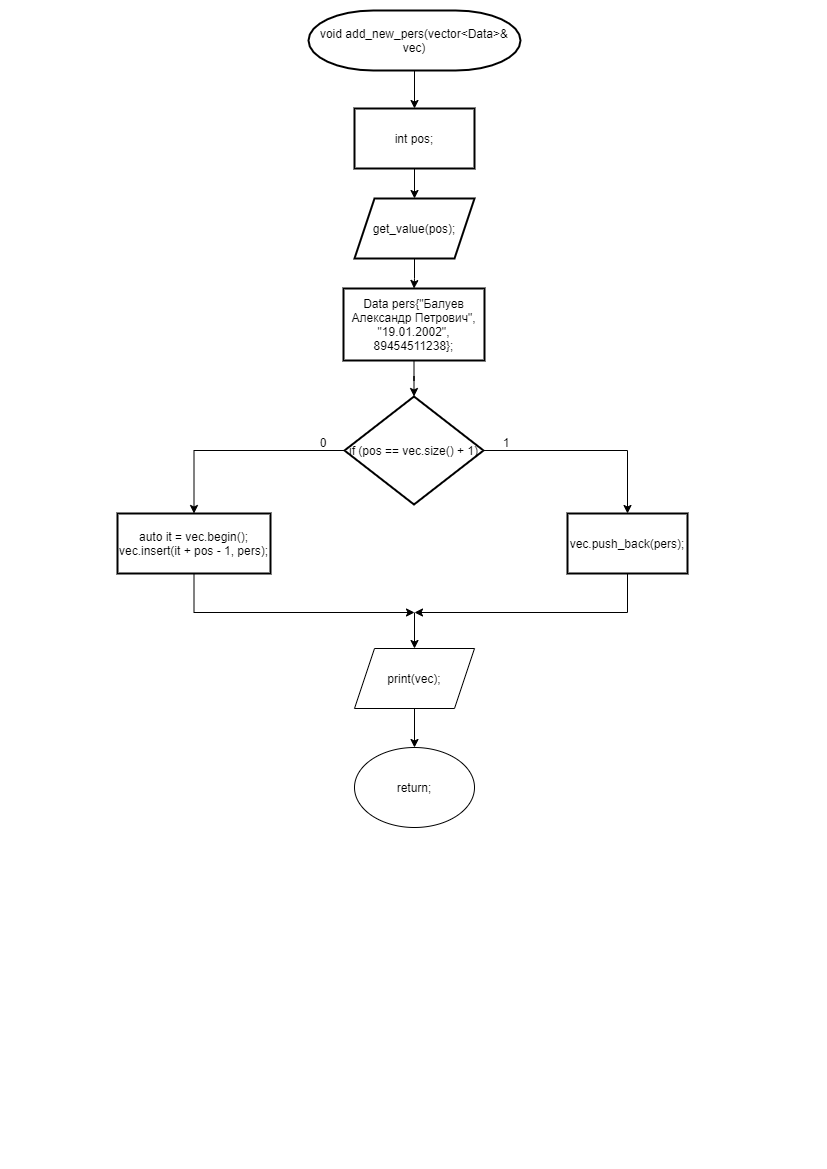
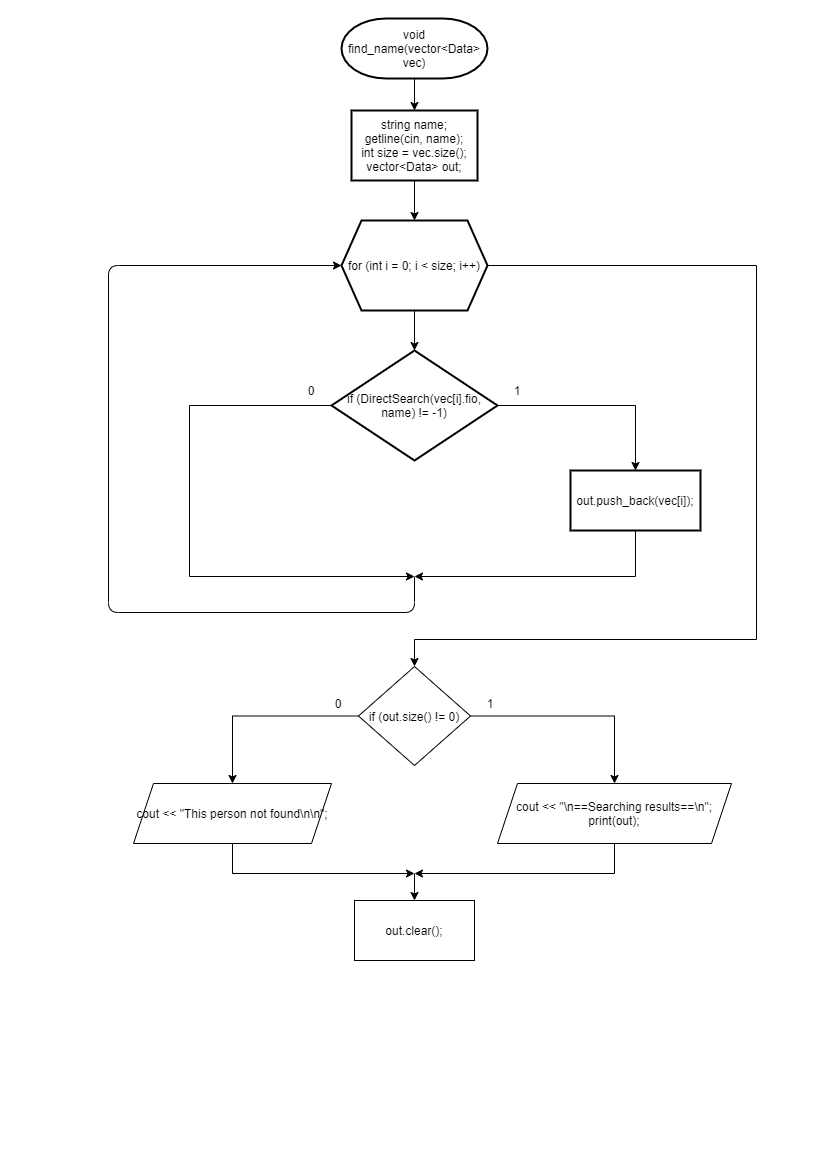
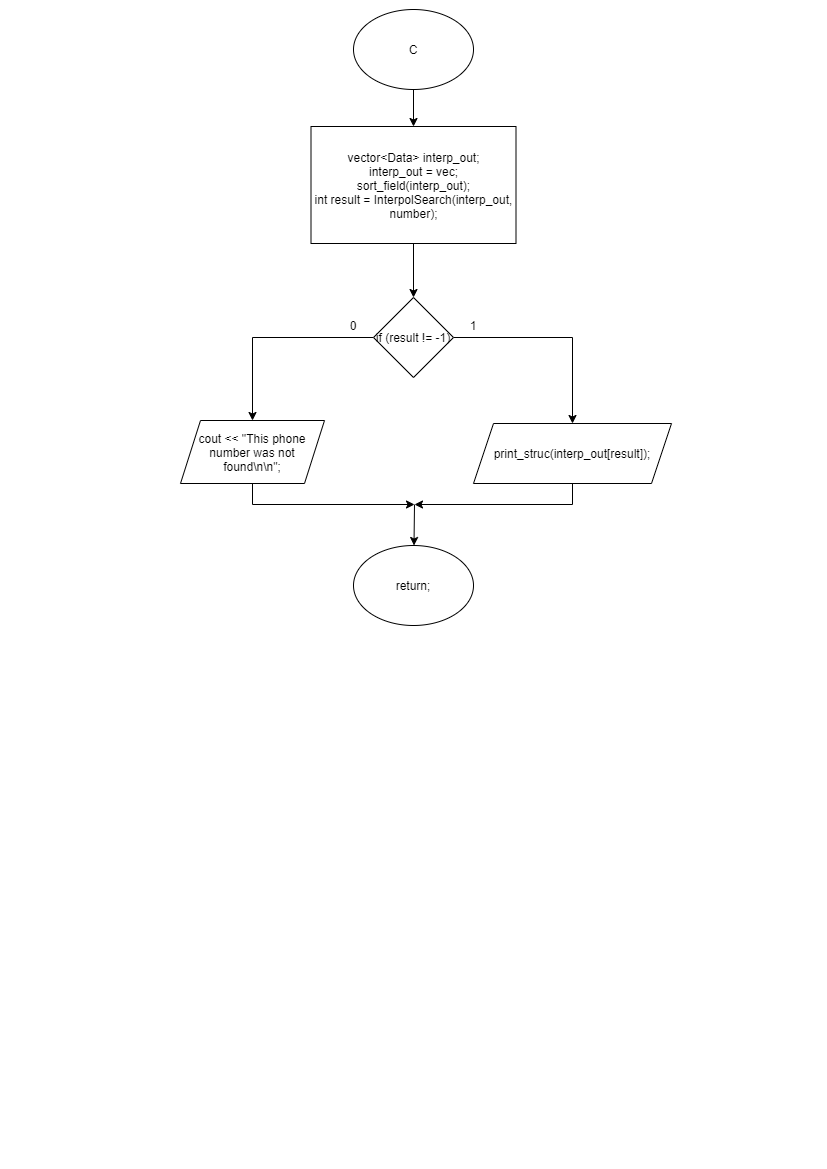
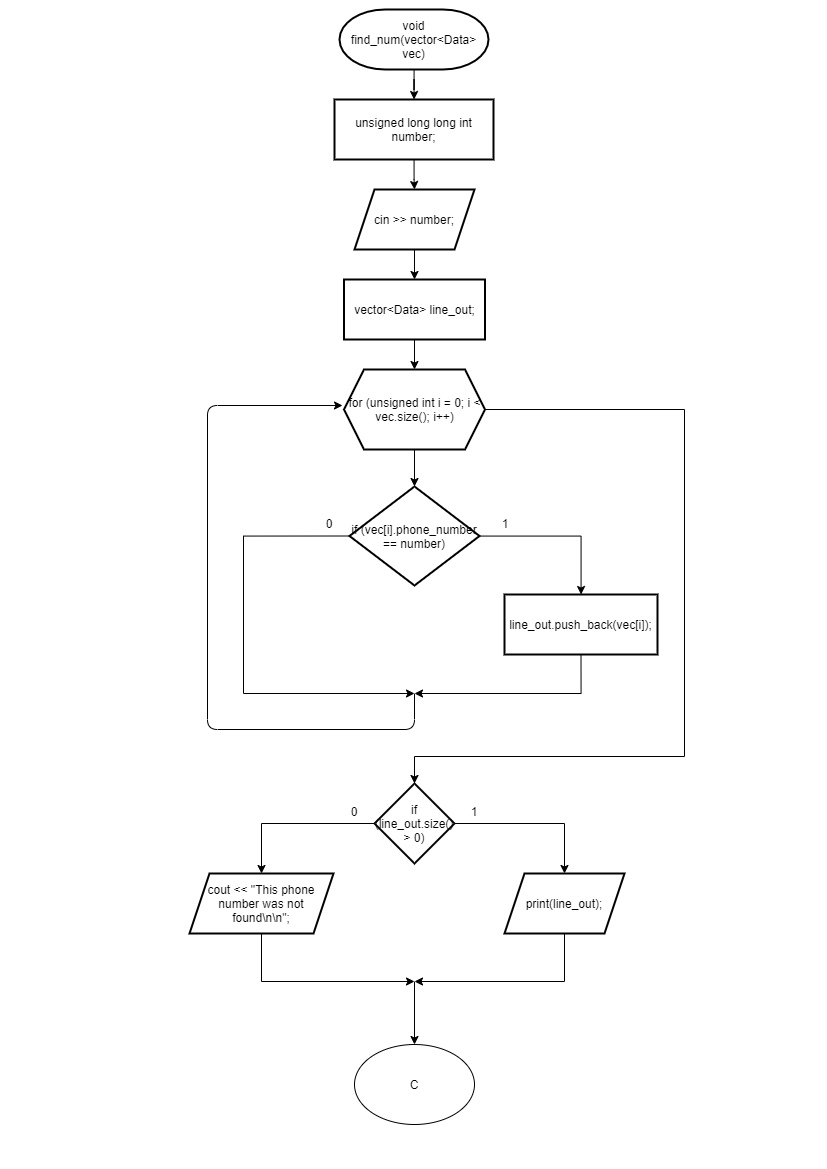
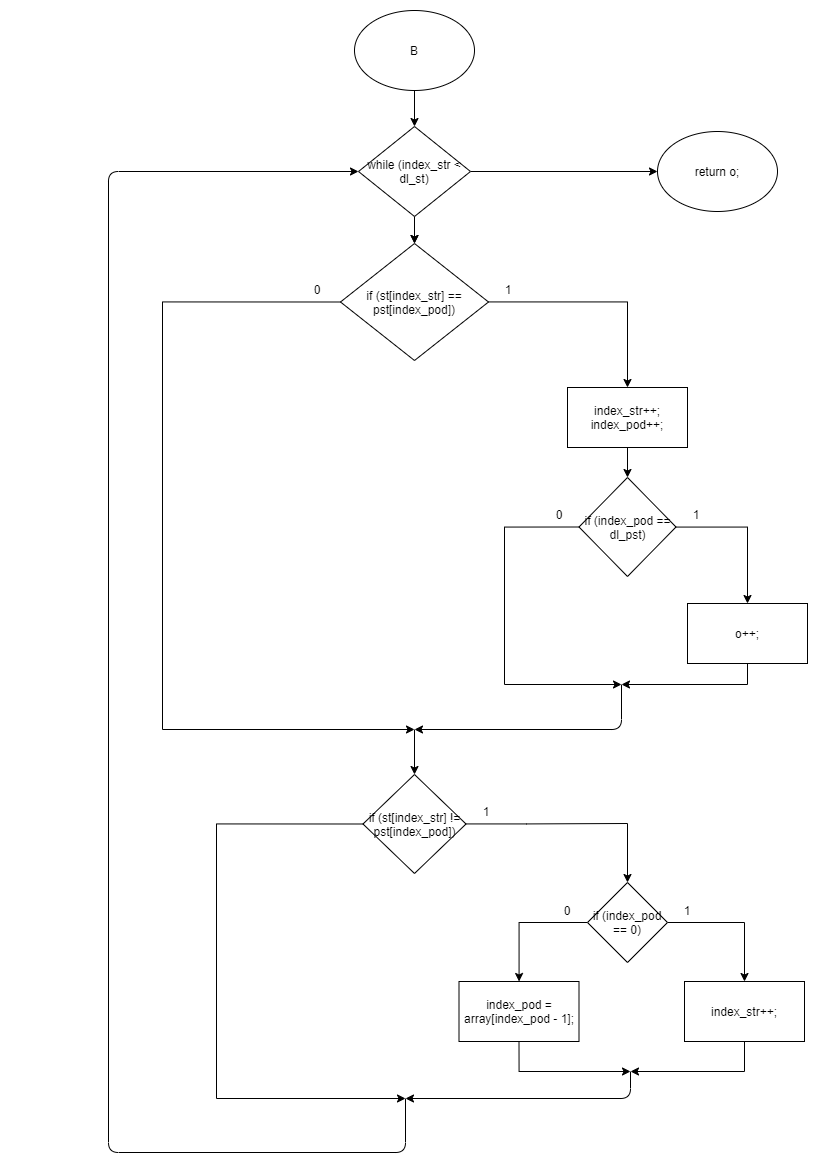
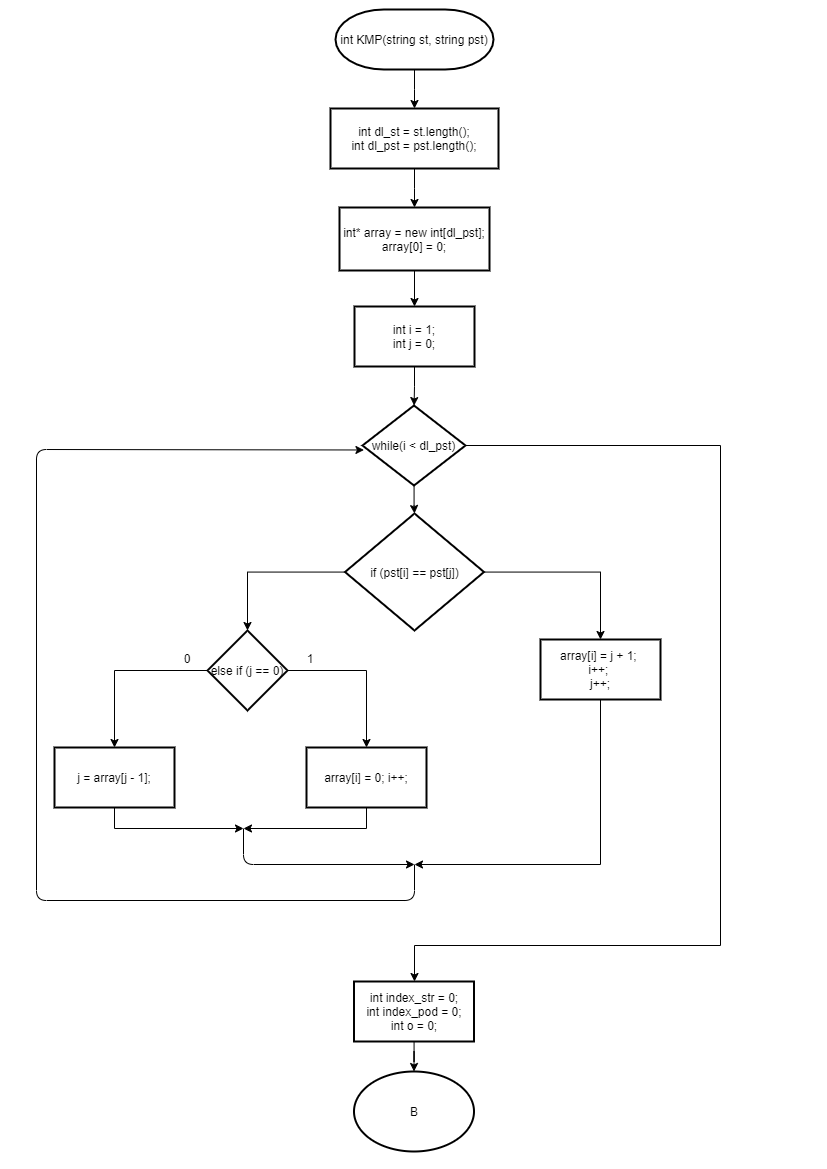
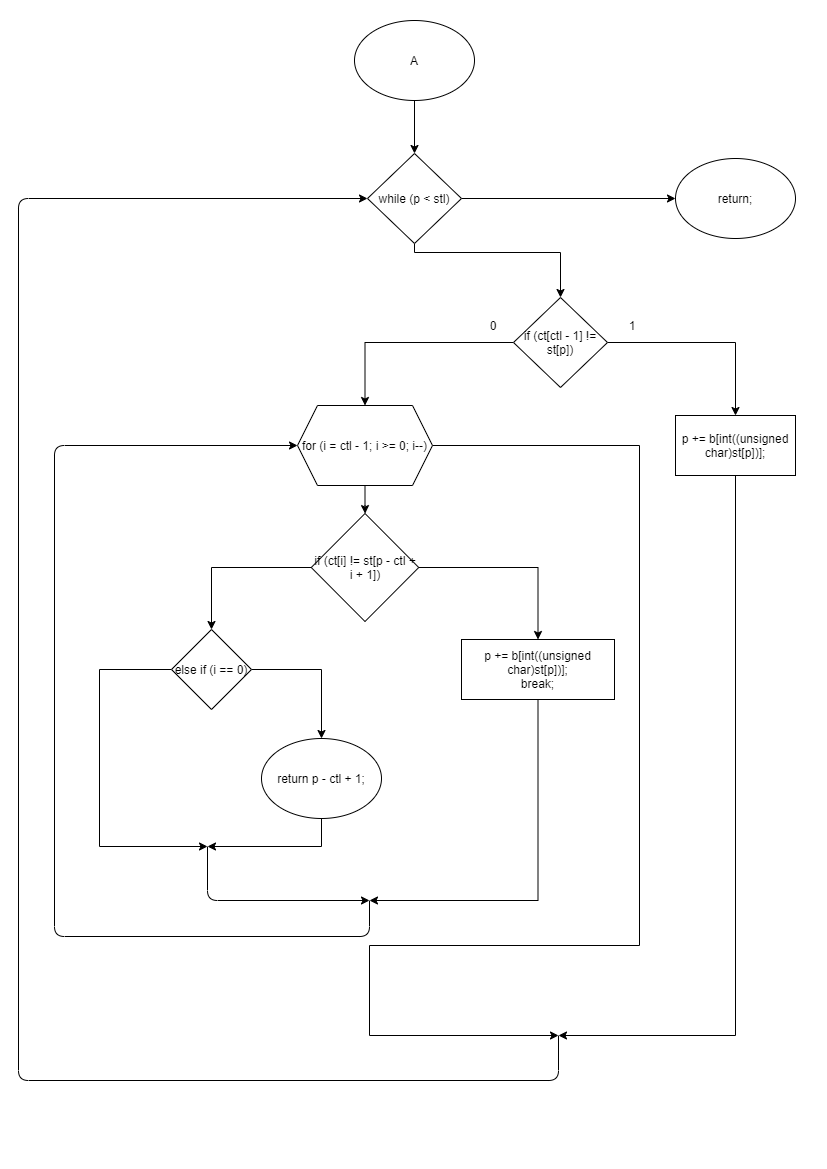
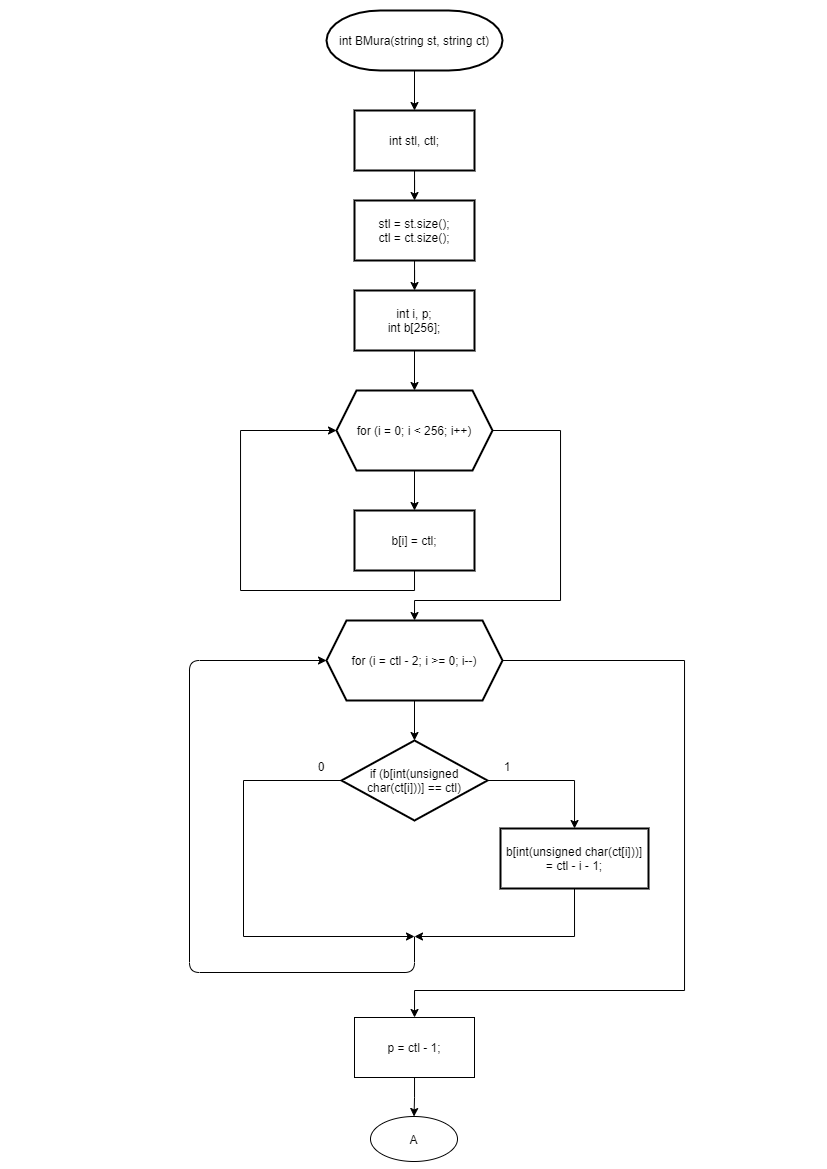
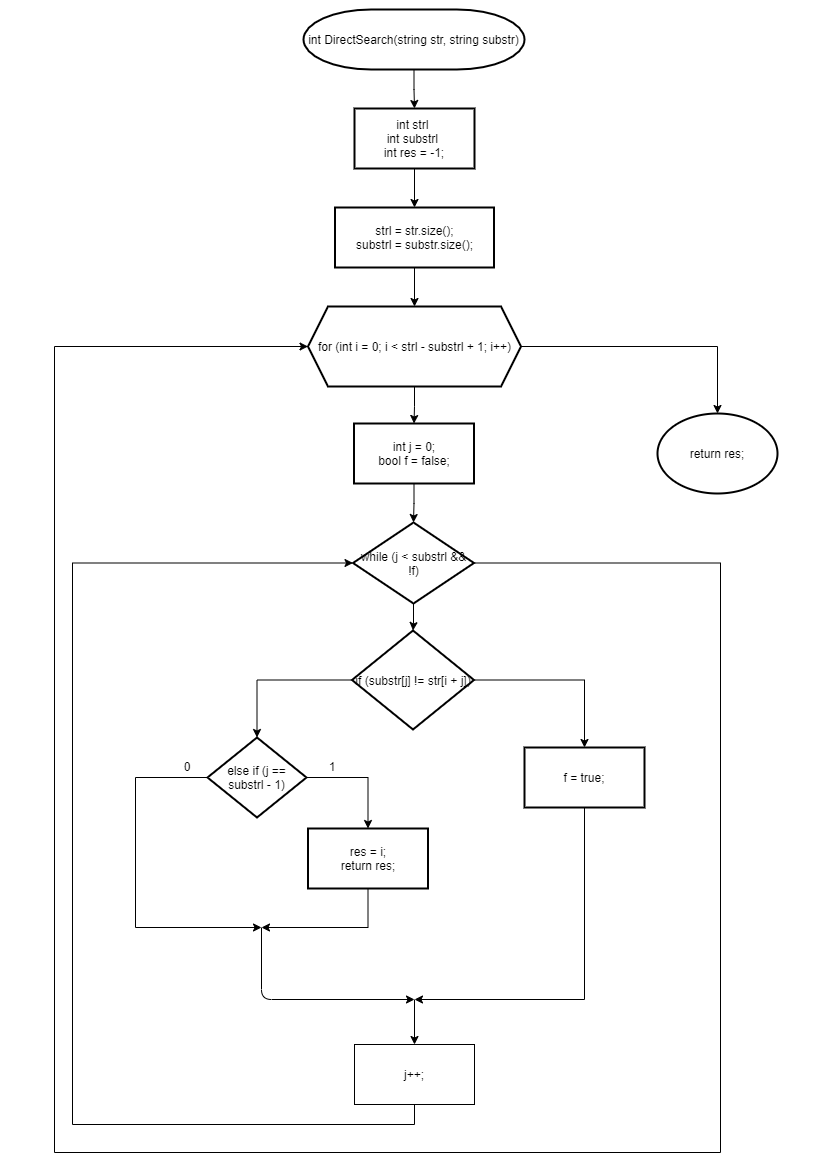
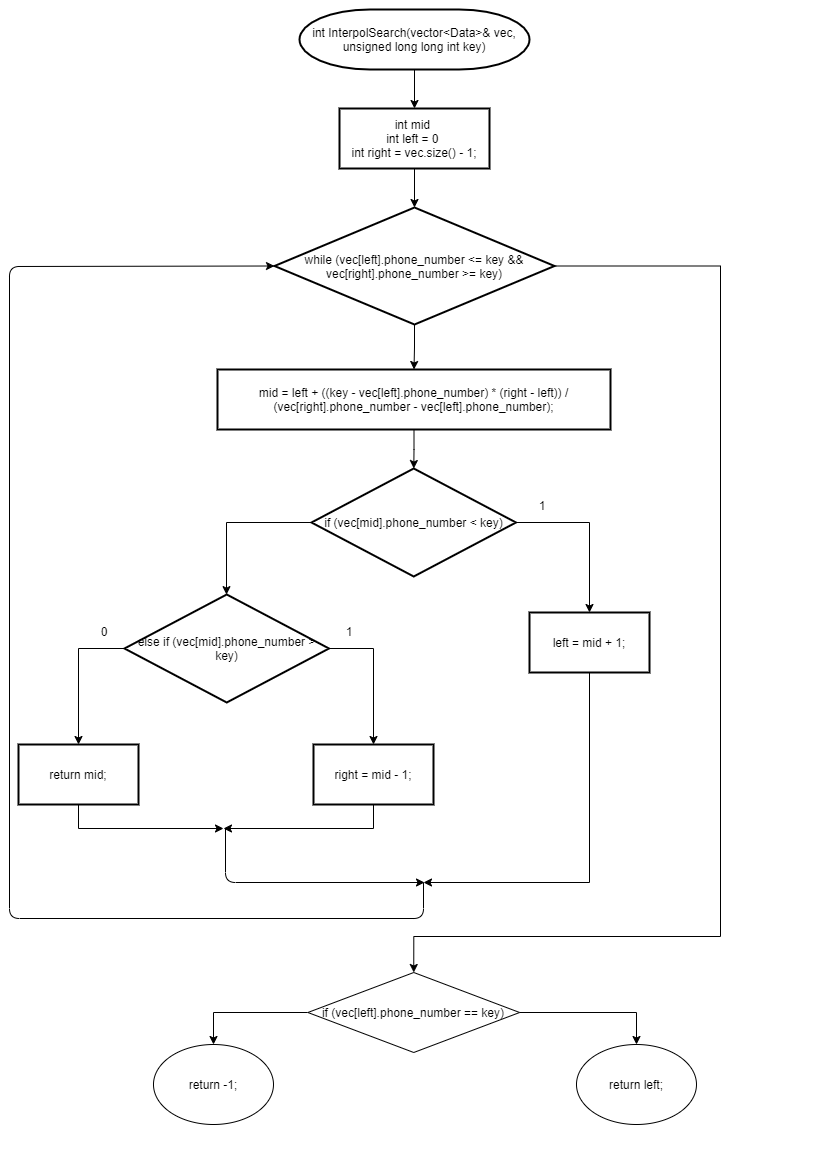
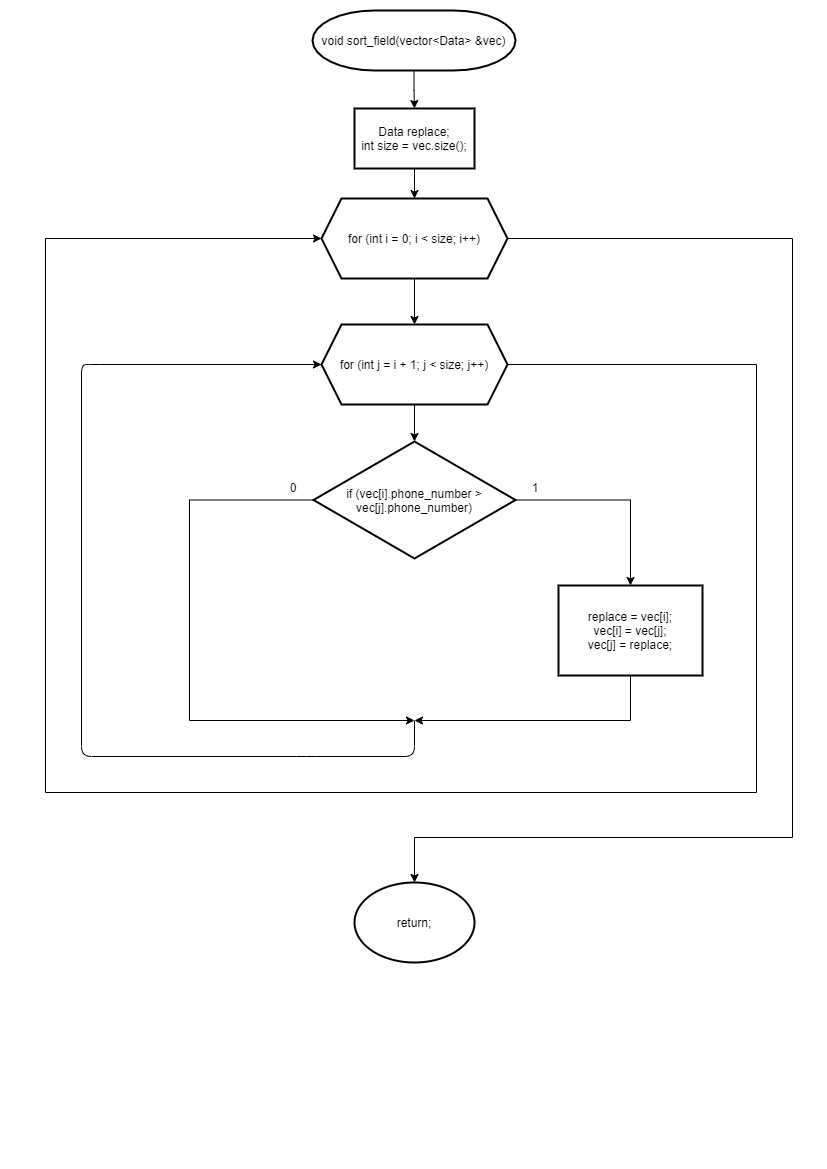
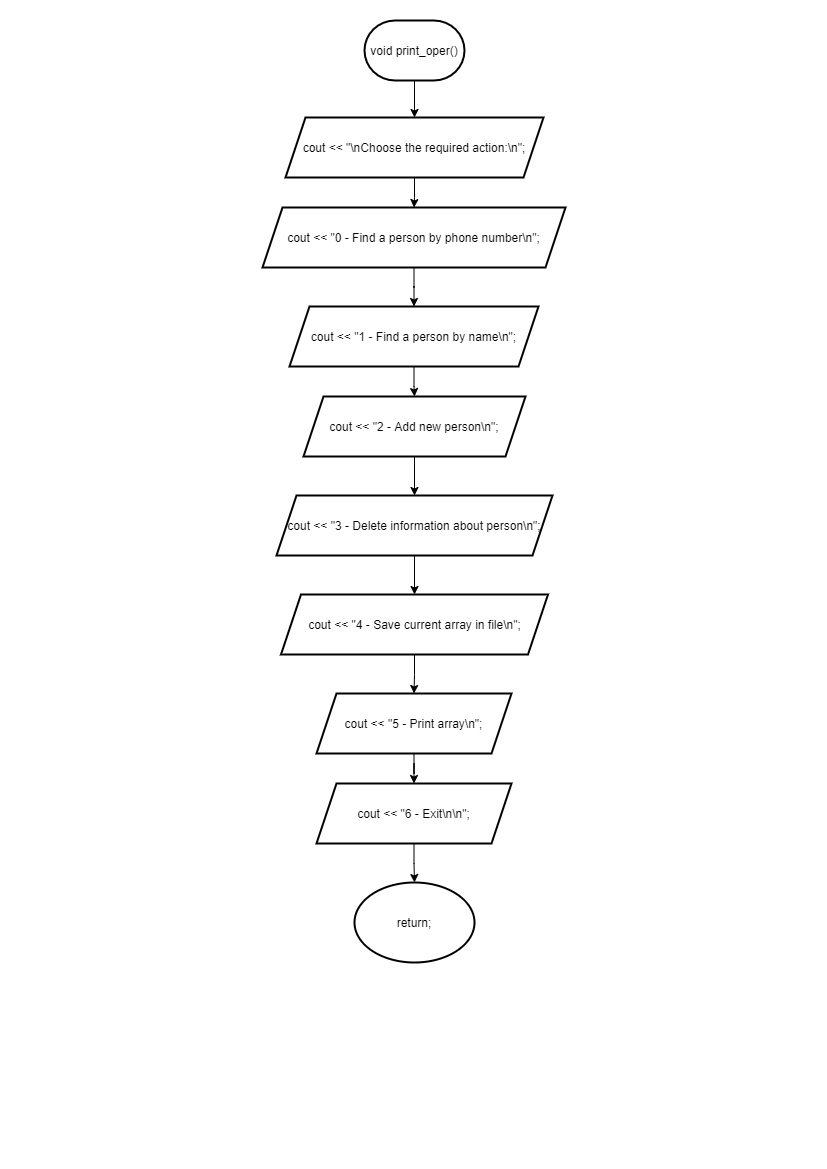
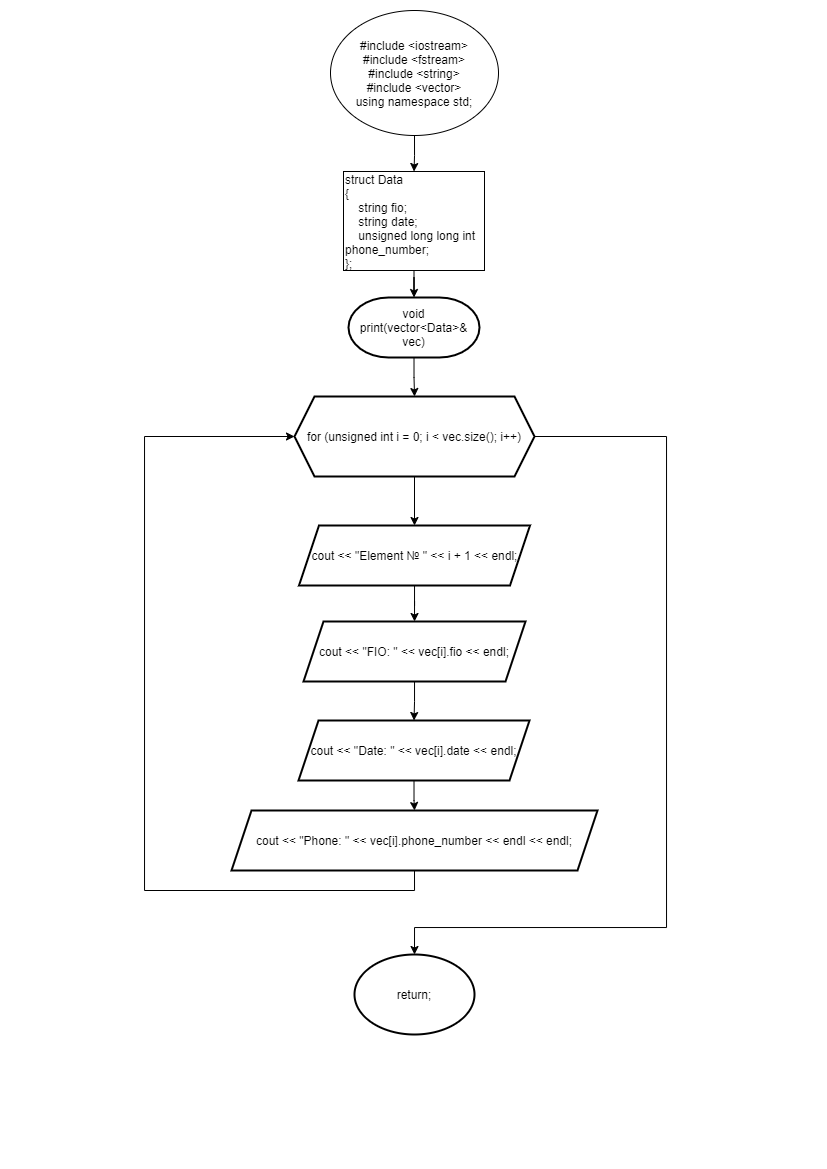
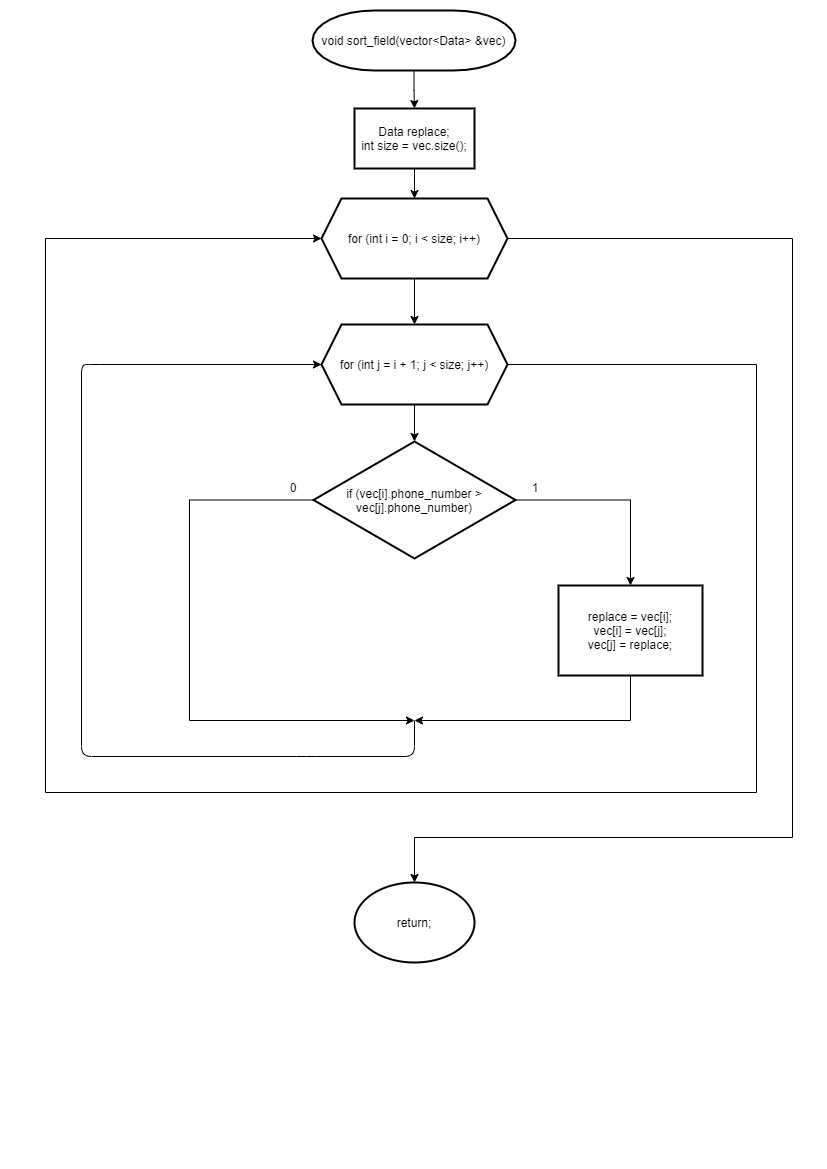
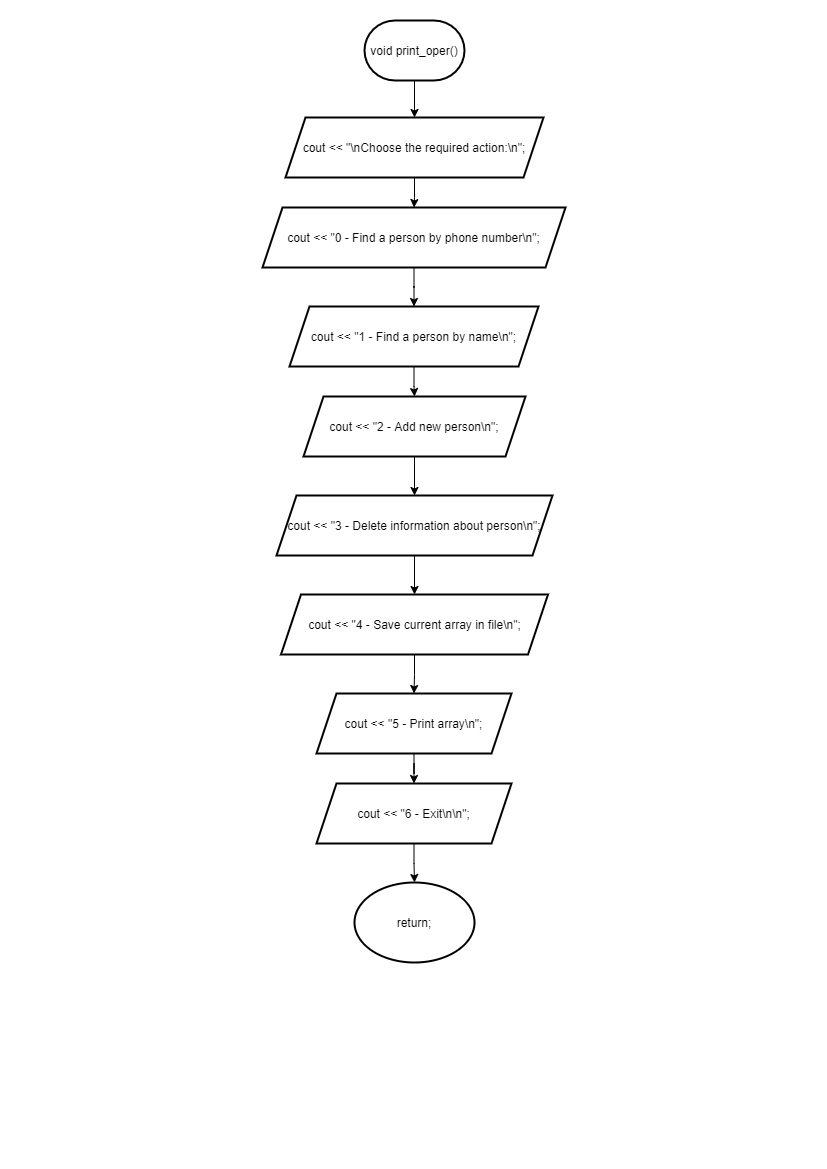
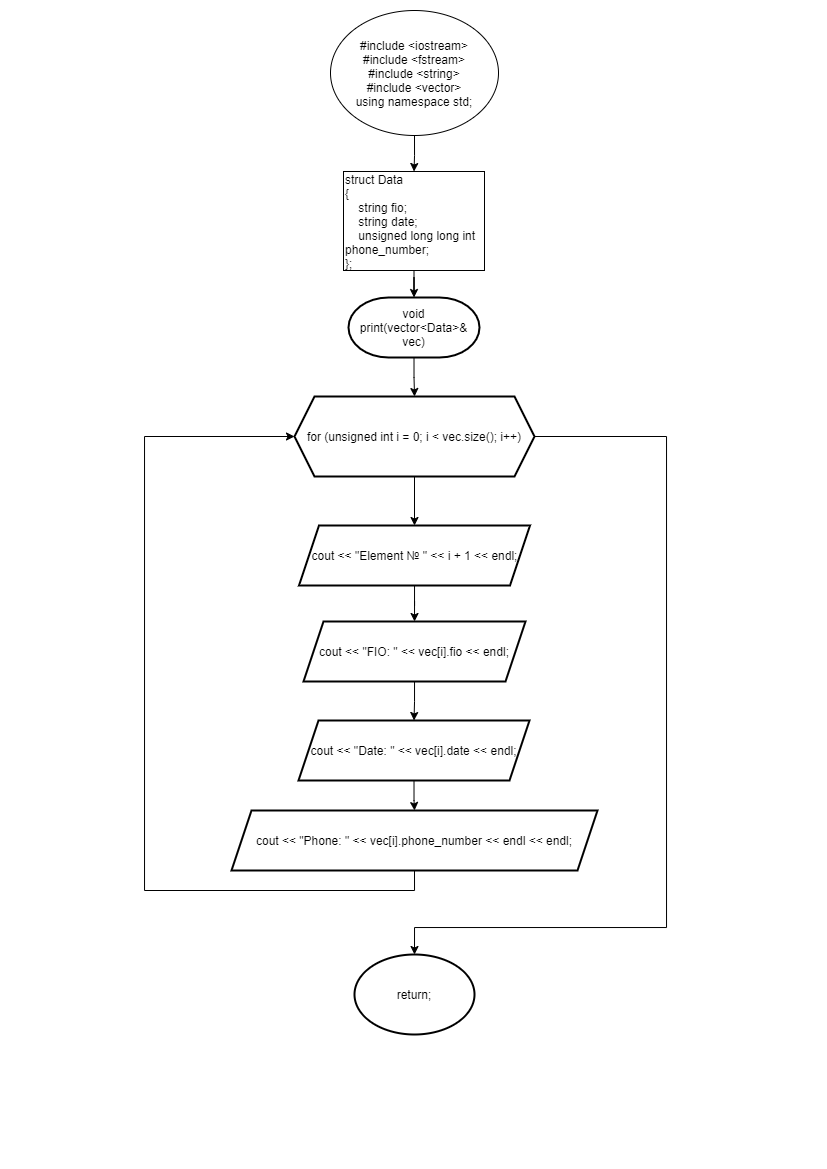
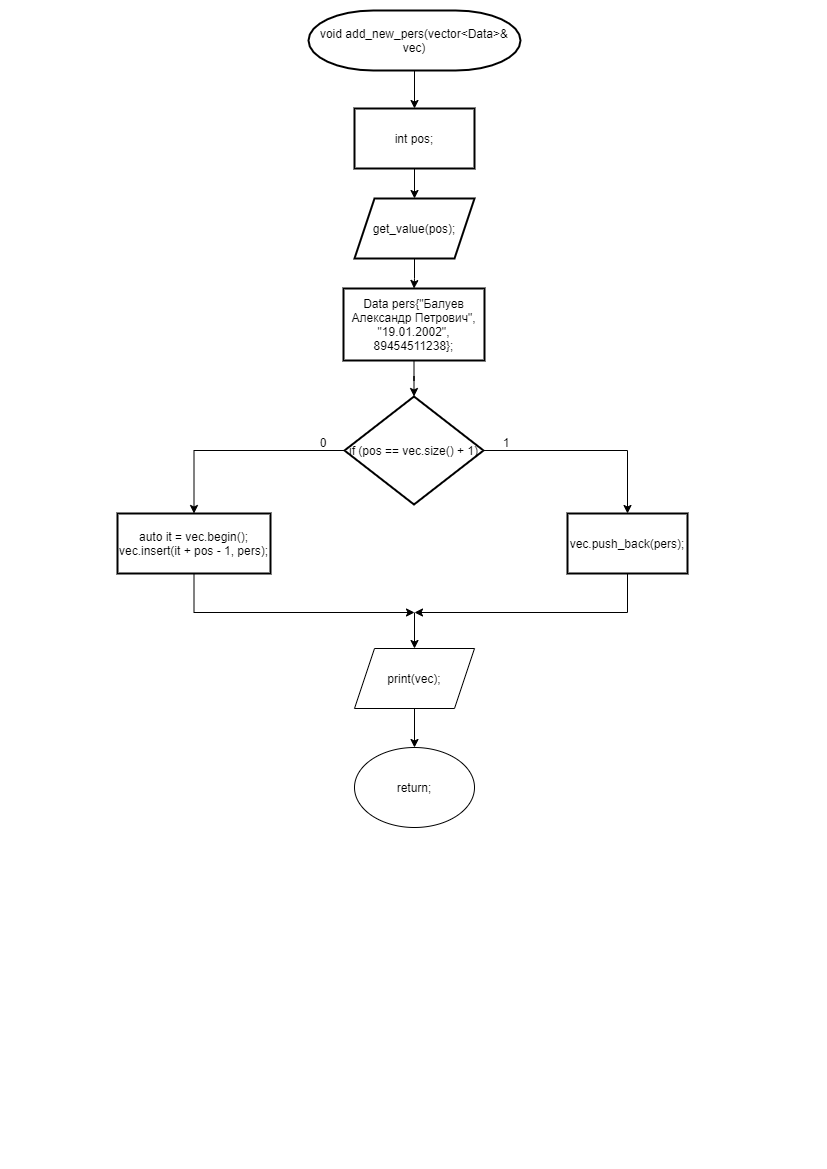
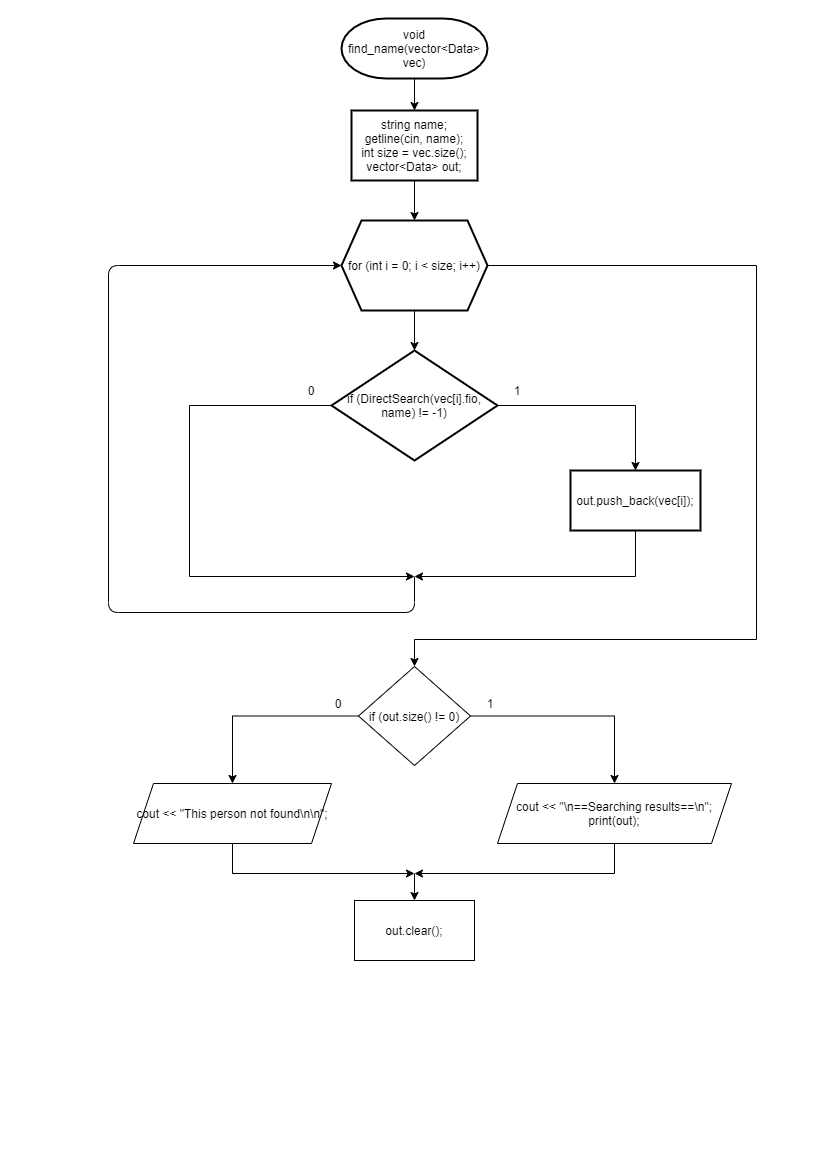
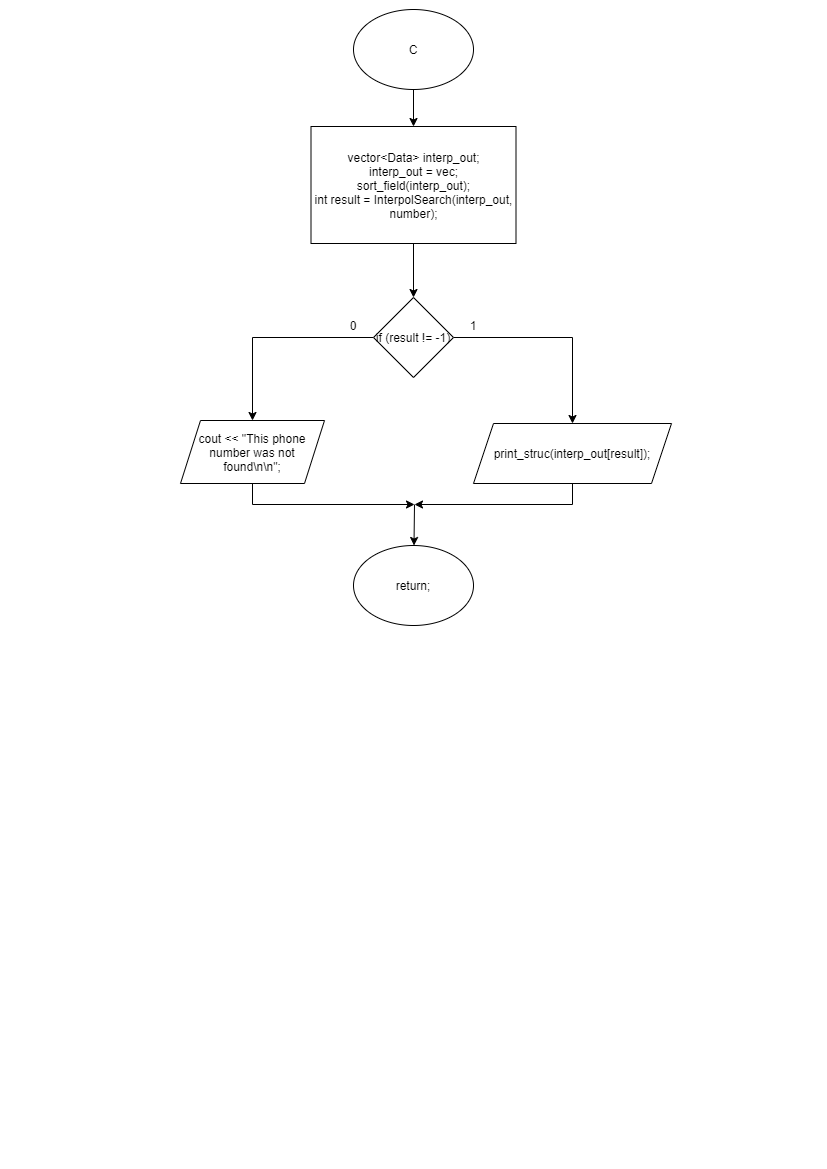
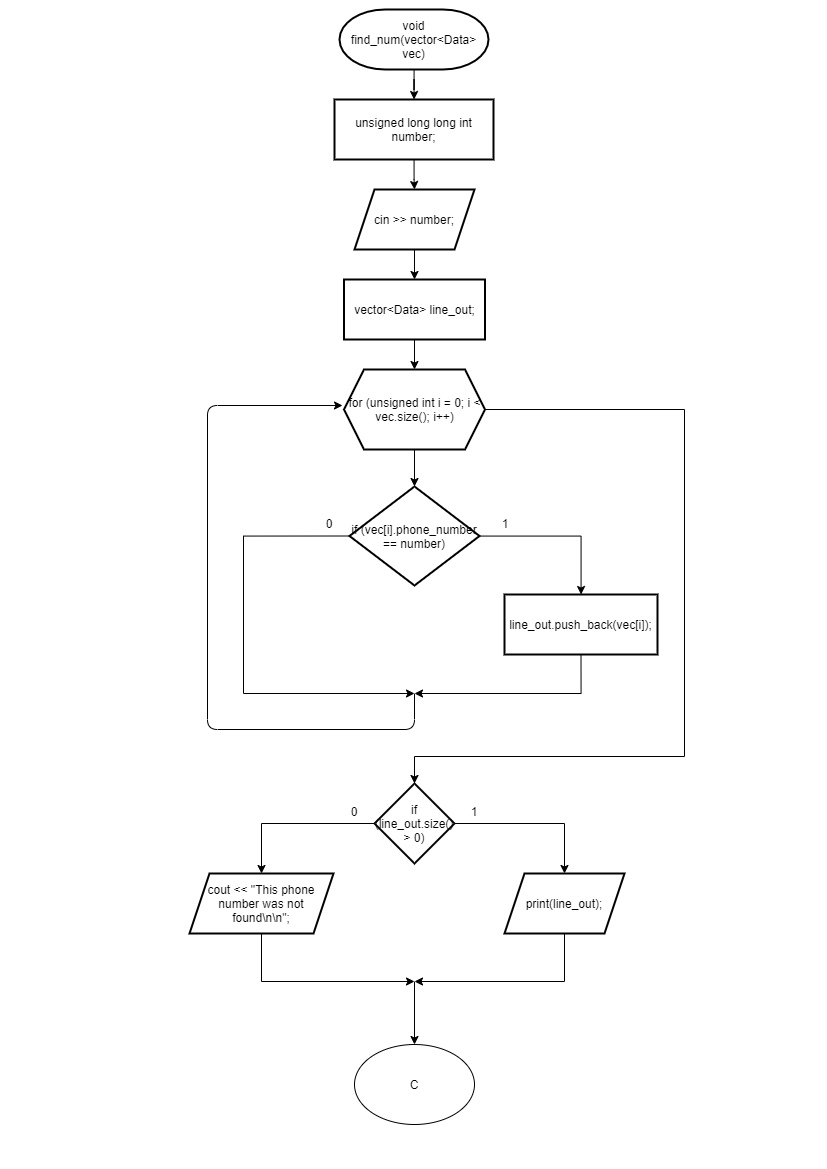
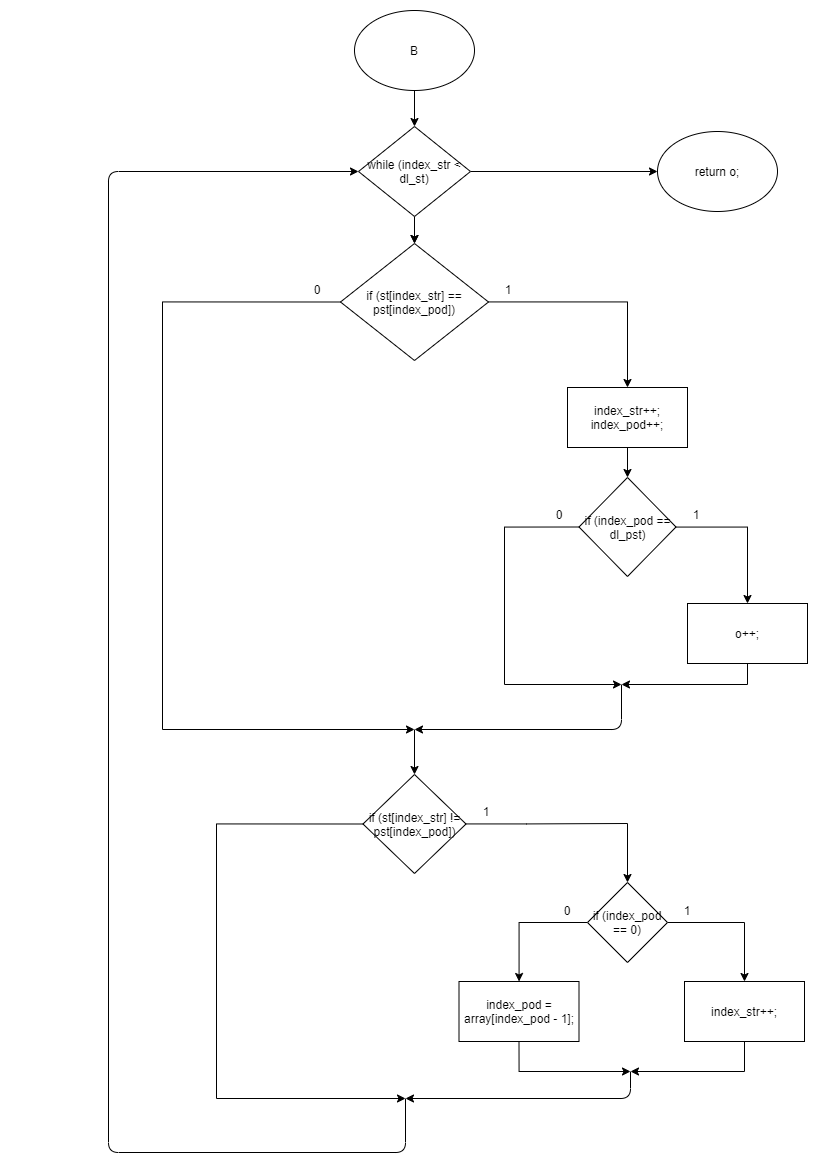
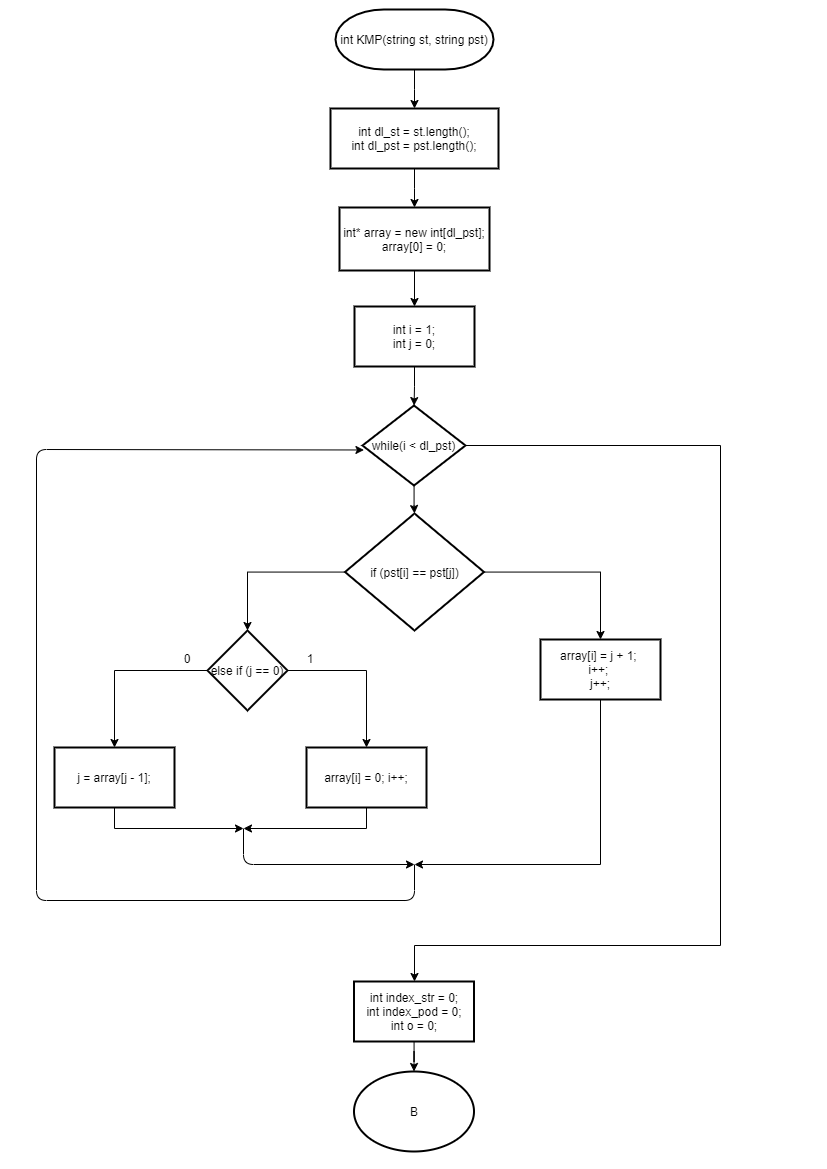
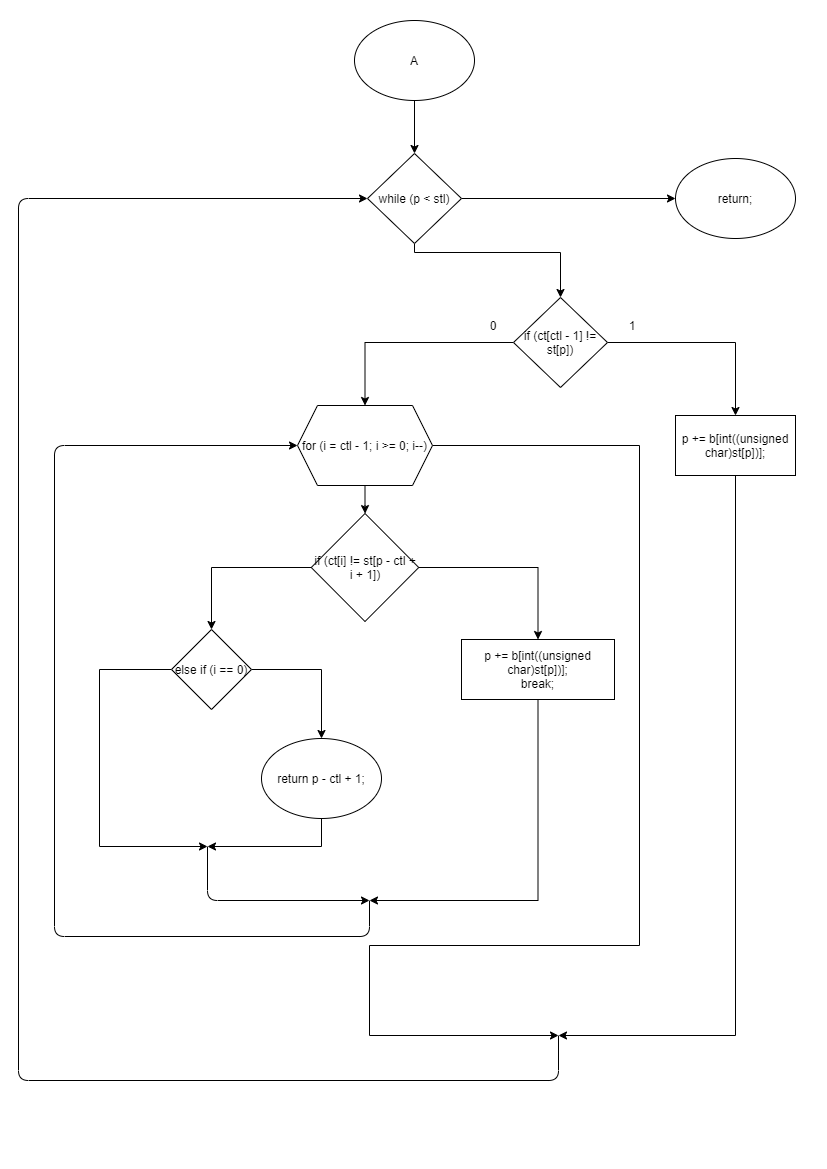
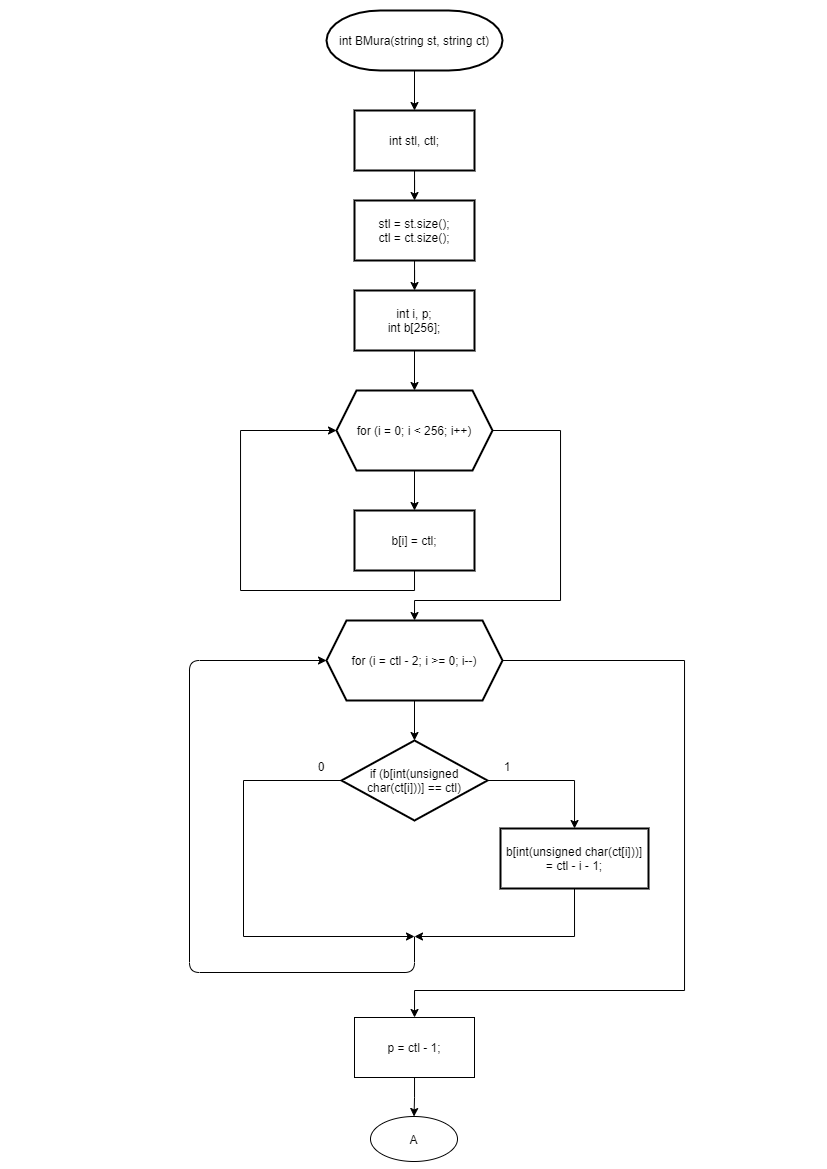
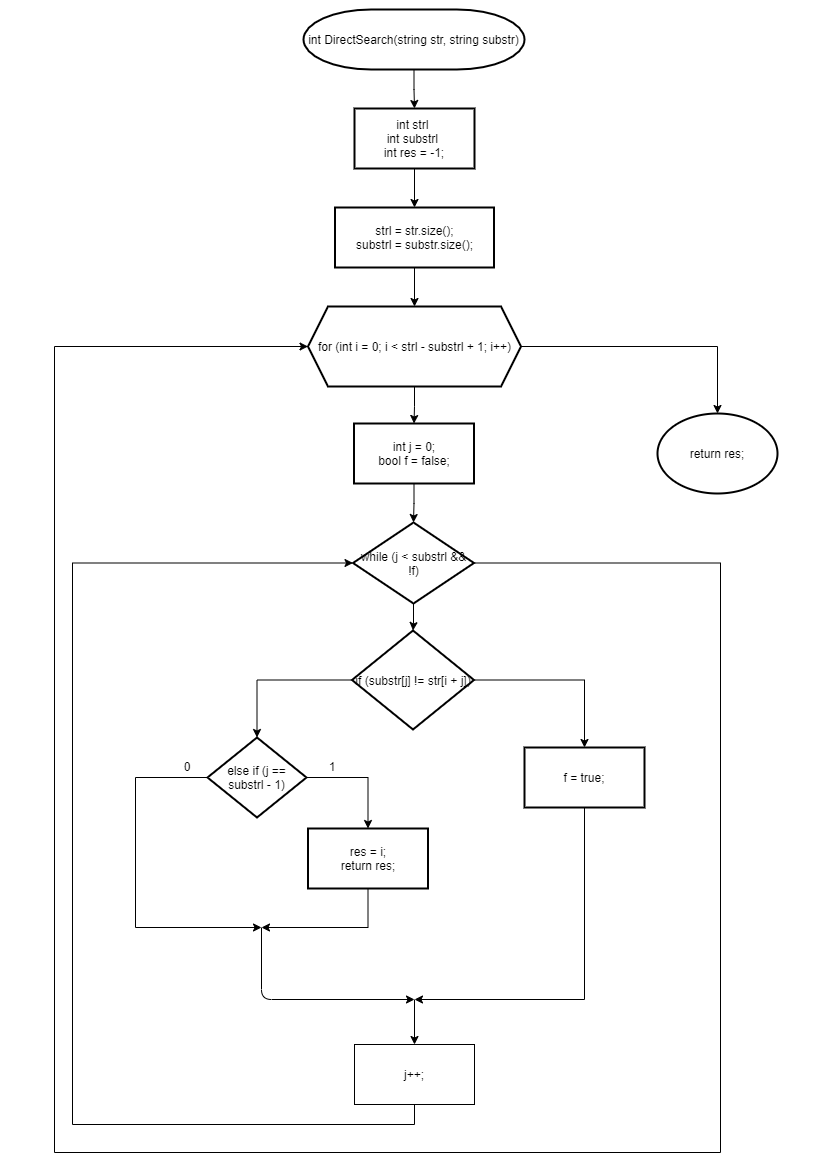
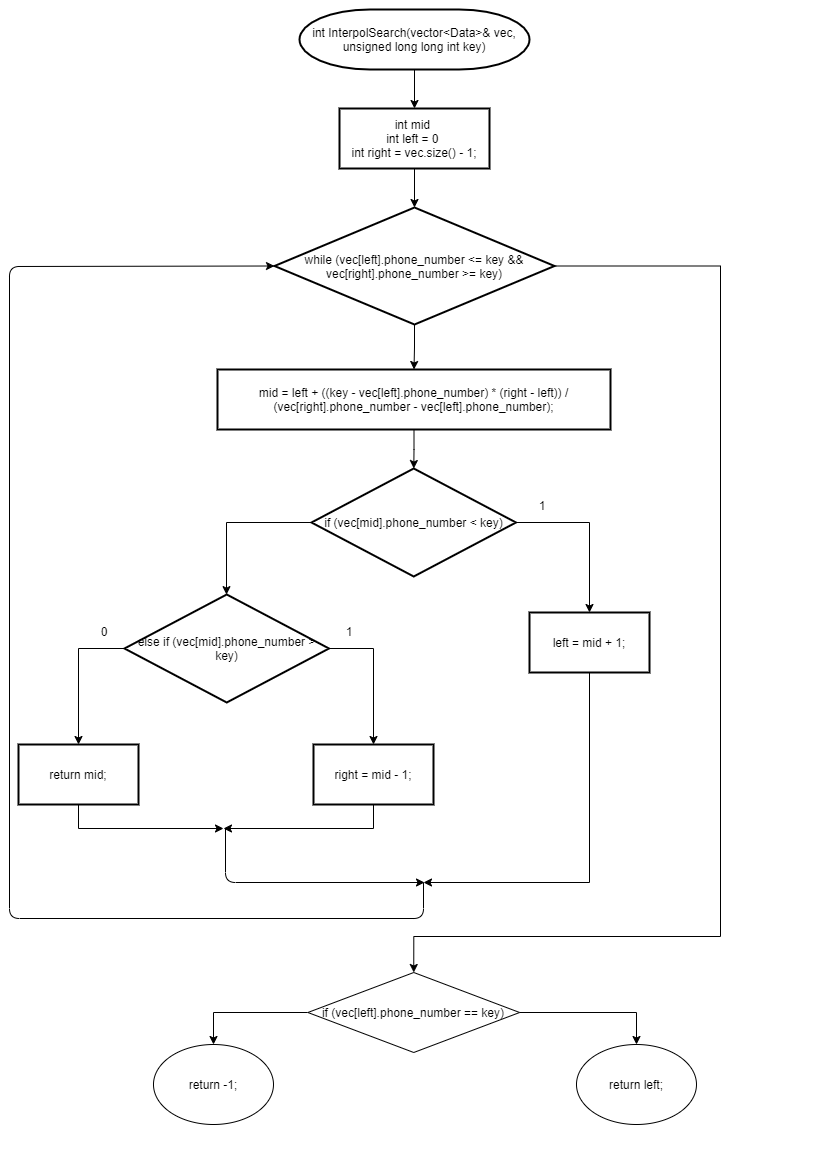
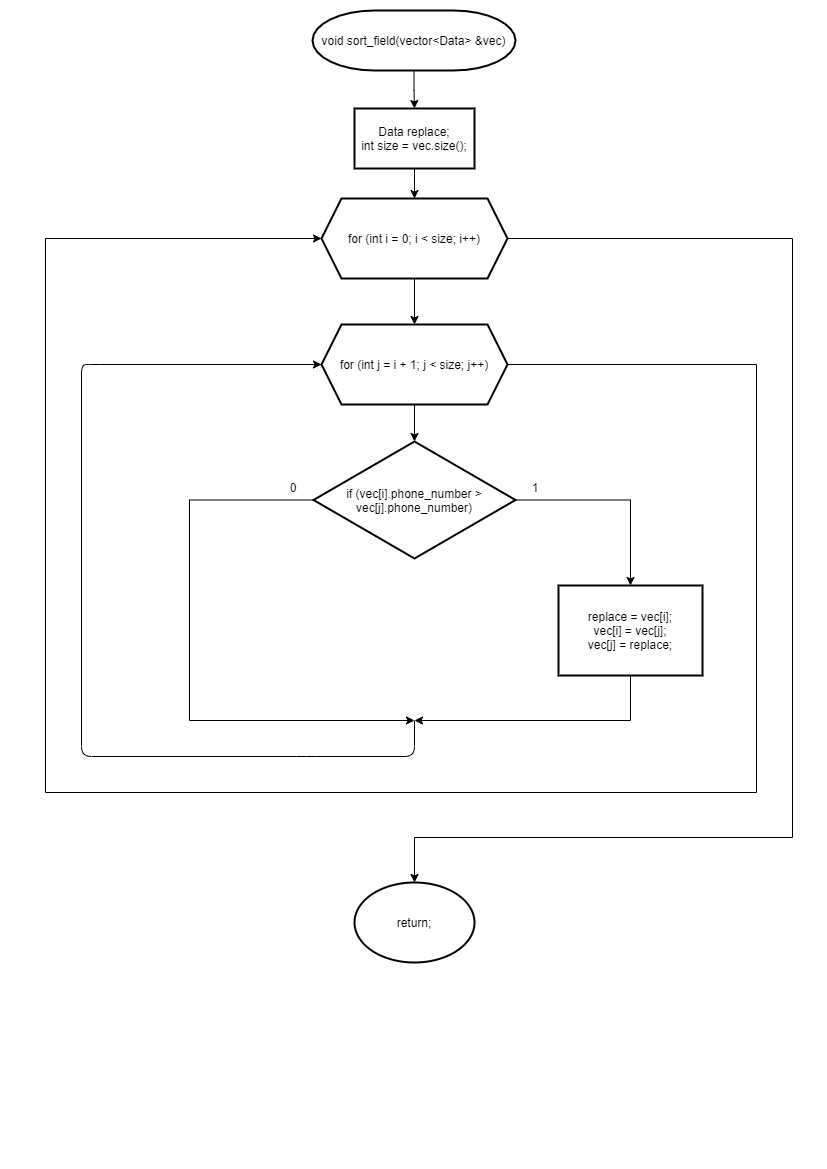
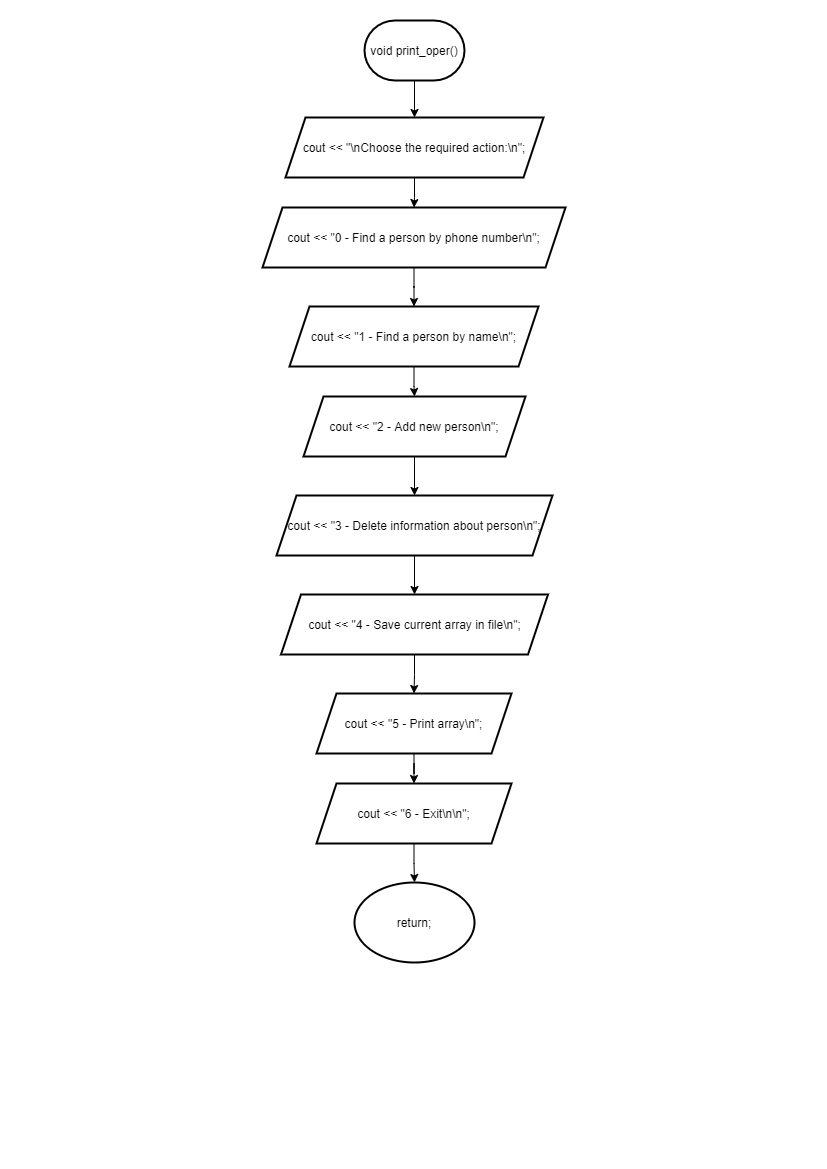
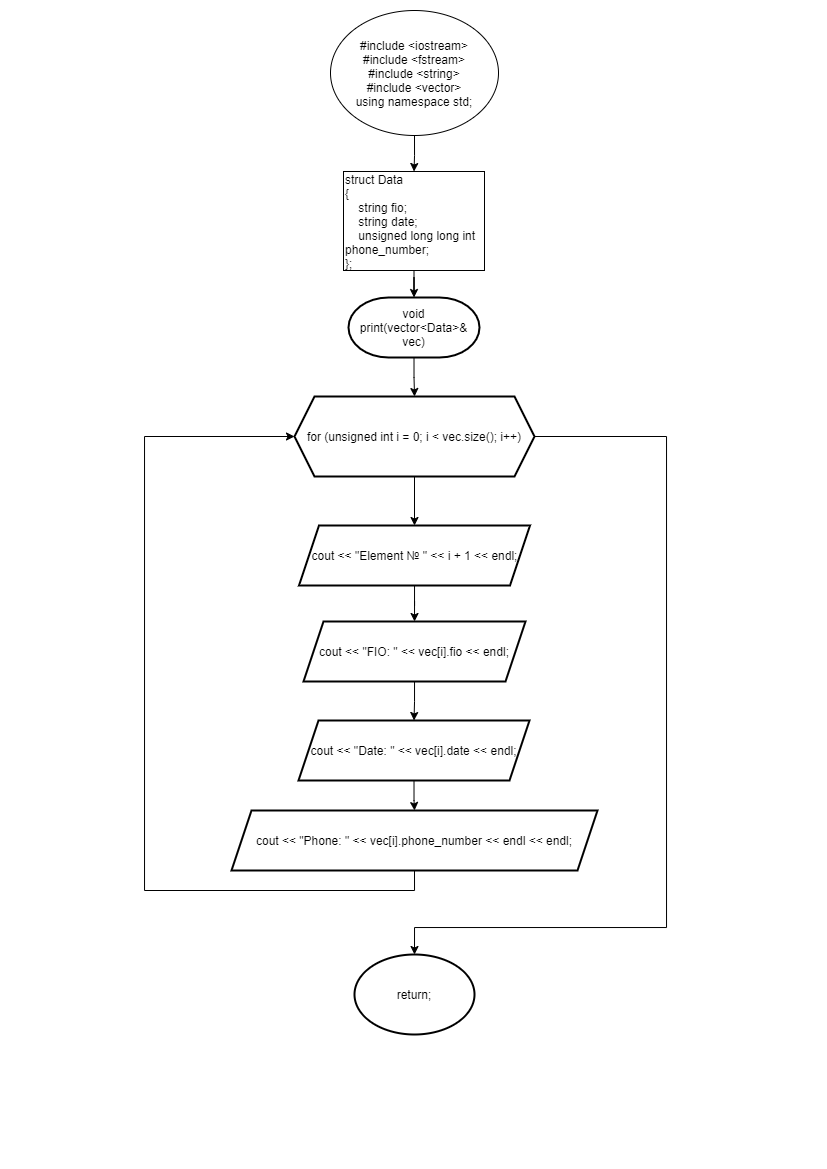
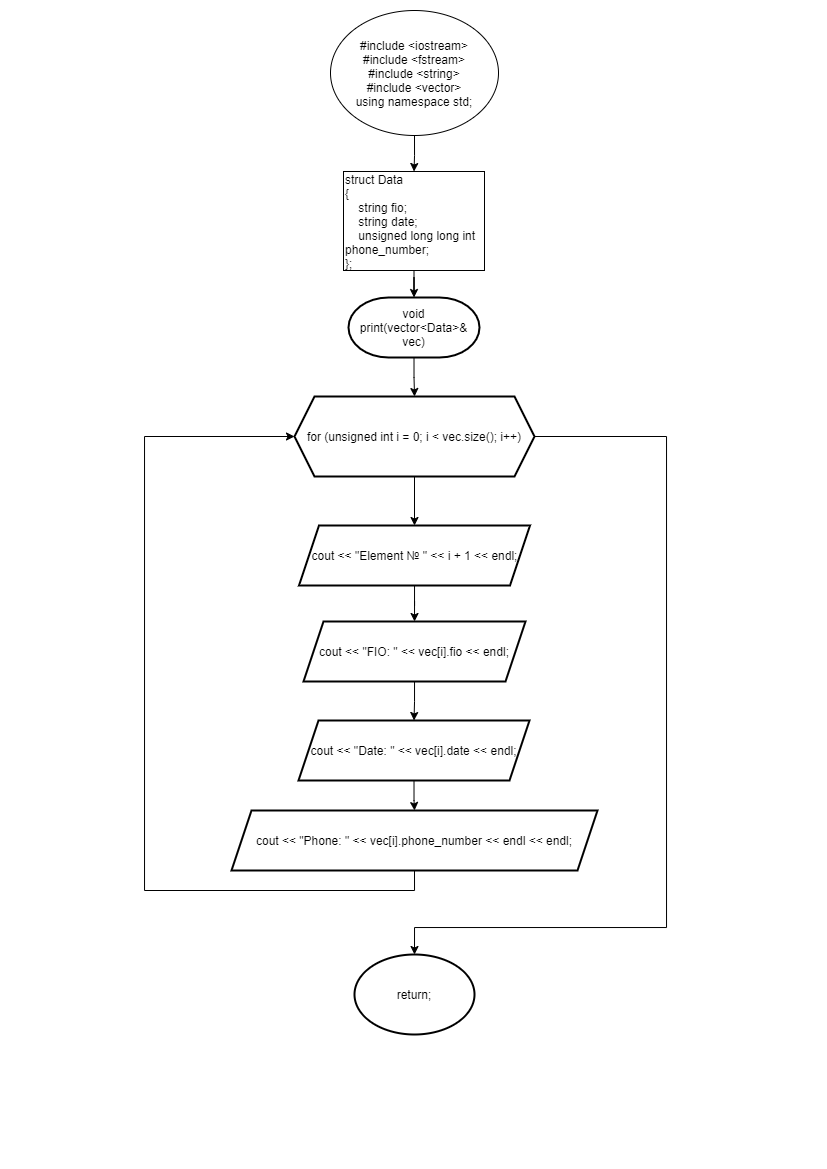
delete[] chel;

return 0;

system("pause");

};

**Блок-схема**



**Скриншоты тестов**

