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**

Дисциплина: «Основы теории алгоритмов и структуры данных»

Тема: Методы поиска: метод Кнутта-Мориса Пратта, метод Бойера-Мура

Вариант 1

Выполнил:

Студент группы РИС-20-1б

Азмагулов Артём Вадимович

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель работы**

Работа с динамическими структурами данных.

**Постановка задачи**

Задание:

1. Создать динамический массив из записей (в соответствии с вариантом), содержащий не менее 30 элементов. Для заполнения элементов массива использовать ДСЧ.

2. Предусмотреть сохранение массива в файл и загрузку массива из файла.

3. Предусмотреть возможность добавления и удаления элементов из массива (файла).

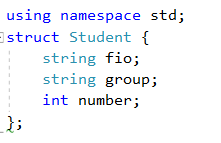
4. Реализовать поиски данных алгоритмом Кнута-Морриса-Пратта, алгоритмом Бойера-Мура и прямым поиском.

|  |  |  |
| --- | --- | --- |
| **Вариант** | **Данные** | **Ключ (string)** |
| 1 | ФИО, группа, рейтинг | ФИО |

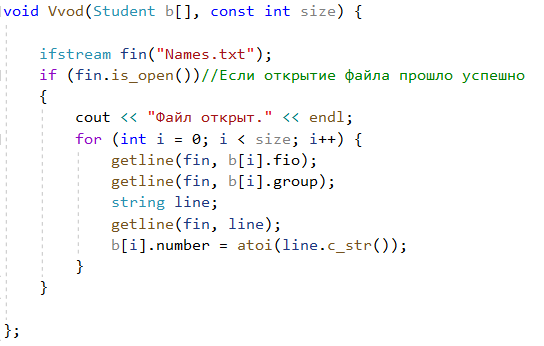
**Анализ задачи**

**1.** Для решения задачи необходимо:

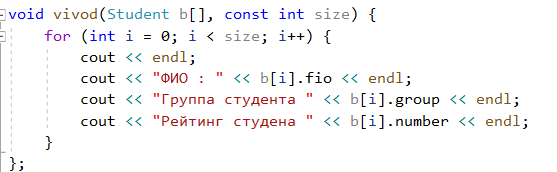
**1.1. Создать структуру данных с типами данных string, string, int:**



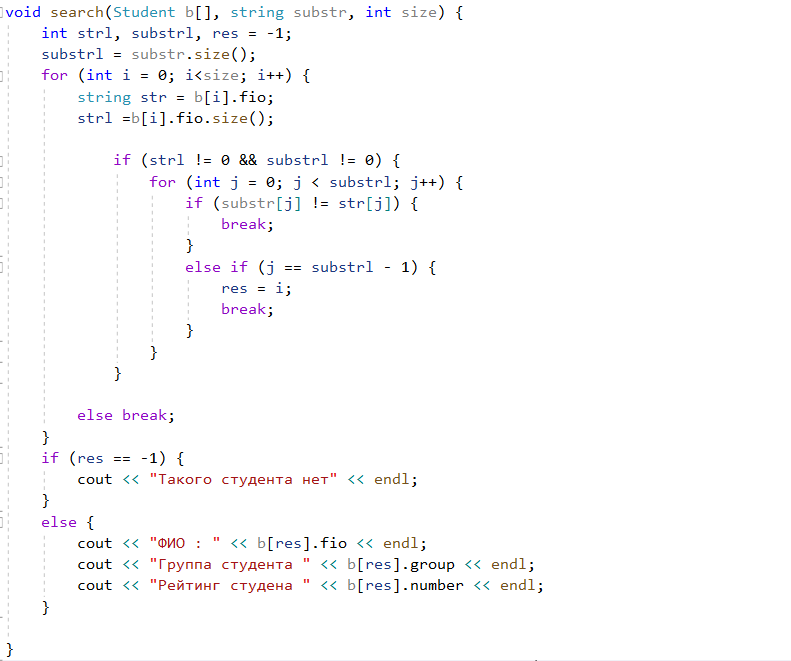
**1.2.** Разработать функцию для заполнения структуры через цикл for, из файла:



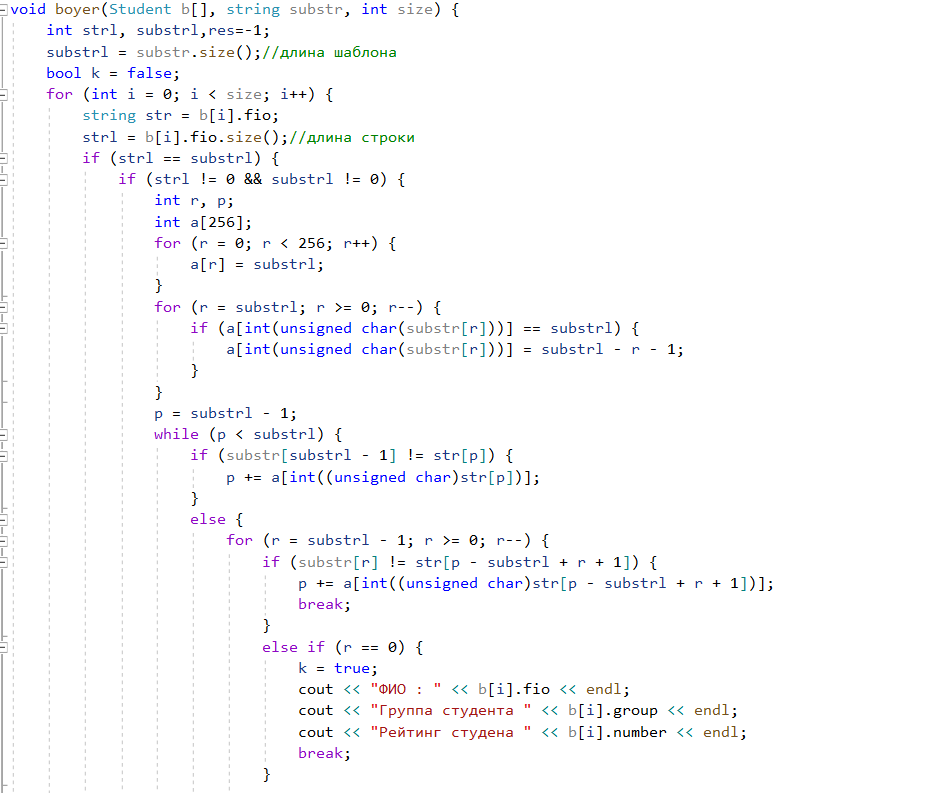
**1.3.**Разработать функцию вывода всех полученных программой значений структур:

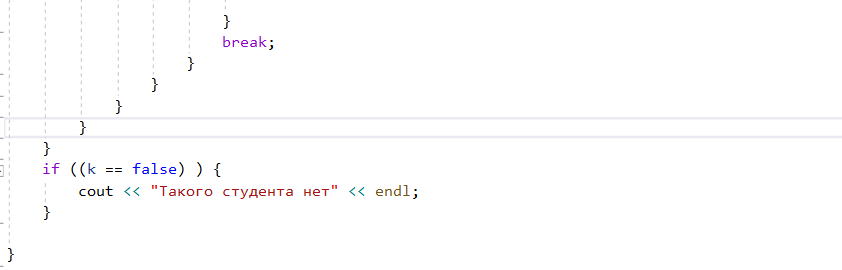


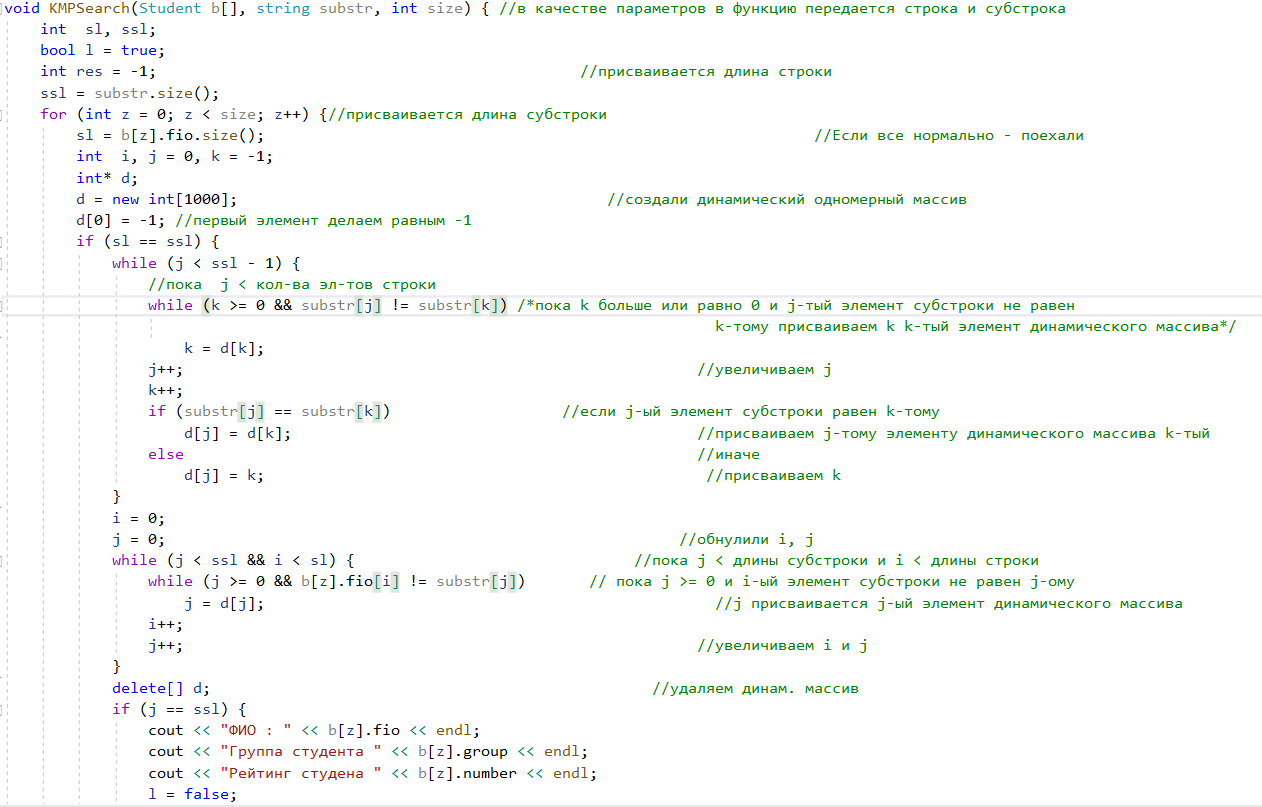
**1.4.** Реализовать прямой поиск подстроки в строке, опираясь на функцию for с дальнейшим перебором всех элементов структуры:

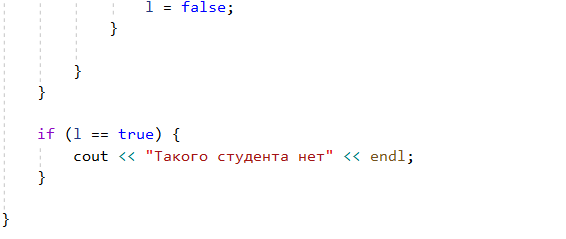


**1.5.** Реализовать алгоритм поиска Бойера-Мура, опираясь на последующий переход к следующему элементу структуры:

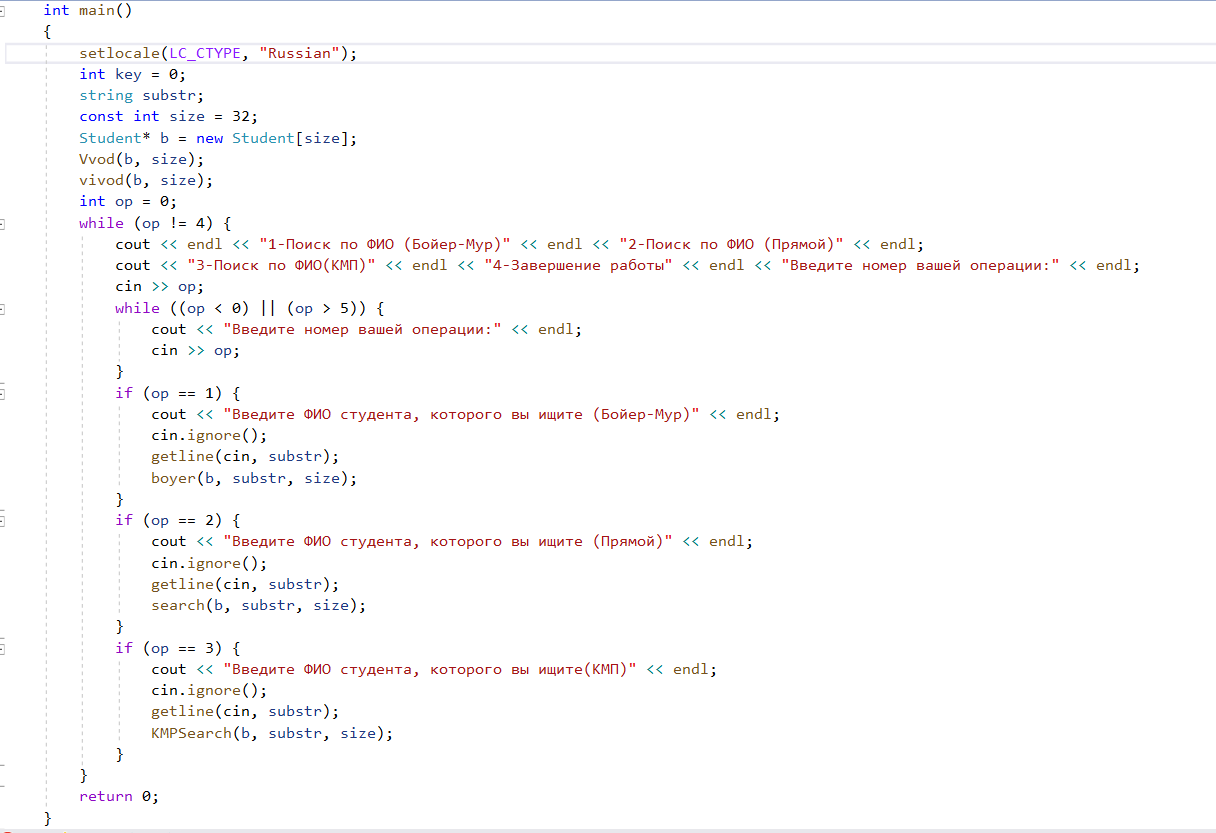




**1.6.** Реализовать поиск Кнута-Мориса-Прата на основе цикла for: 



**1.6.** Реализовать меню, с доступом ко всем выше перечисленным функциям, в том числе к функции с завершением работы:



**2.**В ходе работы были использованы следующие типы данных:

**2.1.**Для работы функции Vvod принимается обращение к элементам структуры и константному значению размера структуры:



**2.2.** Для работы функции vivod принимается обращение к элементам структуры и константному значению размера структуры(const int):



**2.3.** Для работы функции search принимается обращение к элементам структуры, константному значению размера структуры(const int) и размер символьного ключа искомого элемента(int):



**2.4.** Для работы функции KMPSearch принимается обращение к элементам структуры, константному значению размера структуры(const int) и размер символьного ключа искомого элемента(int):



**2.5.** Для работы функции boyer принимается обращение к элементам структуры, константному значению размера структуры(const int) и размер символьного ключа искомого элемента(int):



**3.**Для решения задачи данные были представлены в следующем виде:

**3.1.** Для обработки строк из файла данные были записаны в виде структуры Student, содержащий в себе строковые значения ФИО, группы и целочисленного рейтинга студента.

**4.**Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.**Ввод данных из файла "Names.txt" реализован с помощью функции getline.

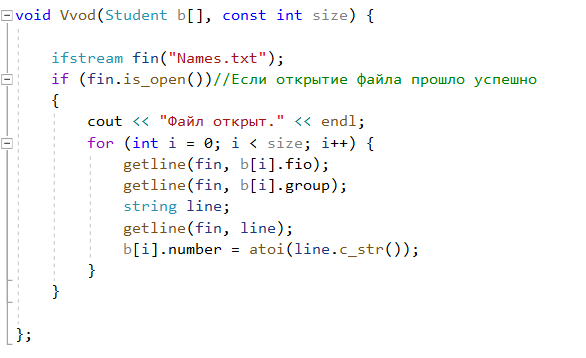


**4.2.** Вывод данных на консоль реализован с помощью оператора cout.

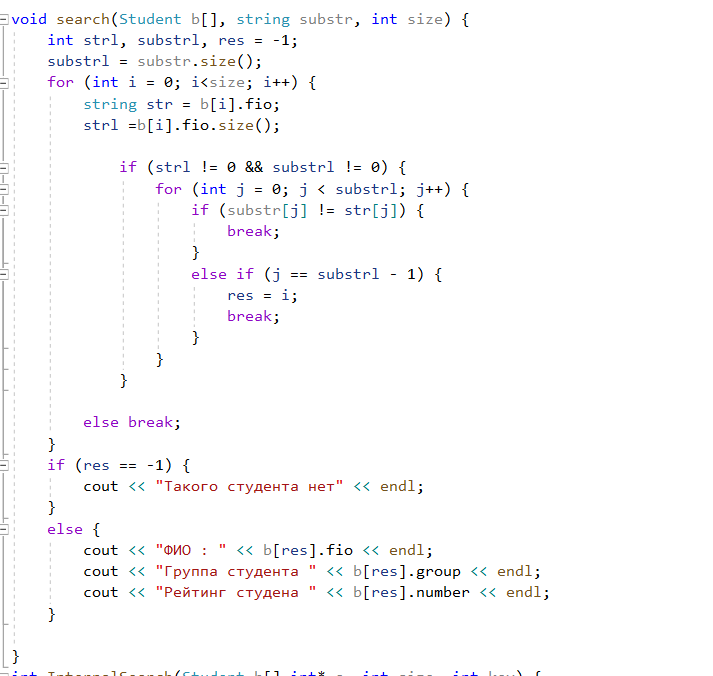


**5.**Поставленные задачи будут решены следующими действиями:

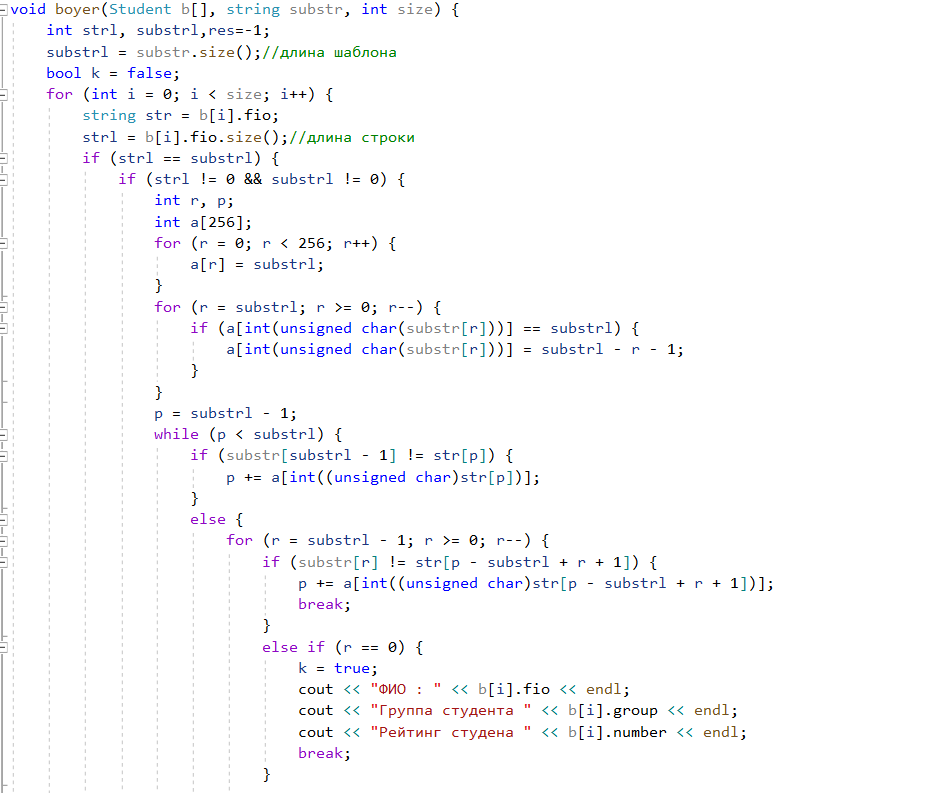
**5.1.** Для работы с информацией из файла построчно, все операции со строкой проводятся внутри цикла for, после каждого полного прохождения тела цикла производится считывание следующей строки. Повторение цикла продолжается, пока не достигнет конца файла.

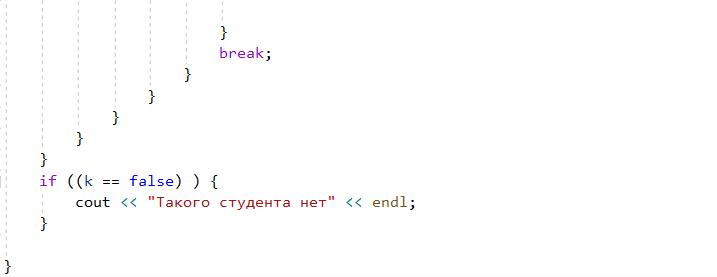


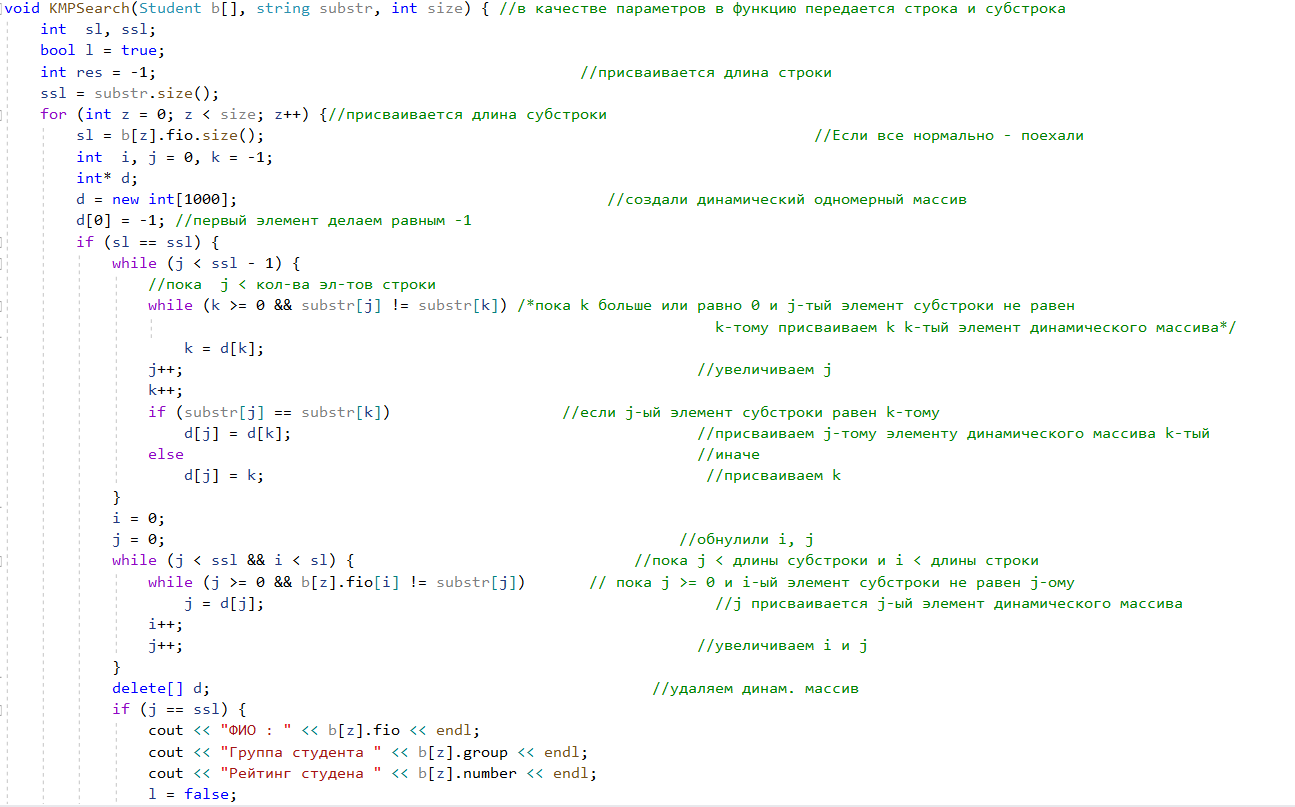
**5.2.**Поиск подстроки в строке происходит после приема ключа функцией, в дальнейшем которая проходит пошаговый перебор всех элементов структуры с последующим сравнением символом, с условием вывода местоположения элемента в случае лишь достижения конца подстроки:

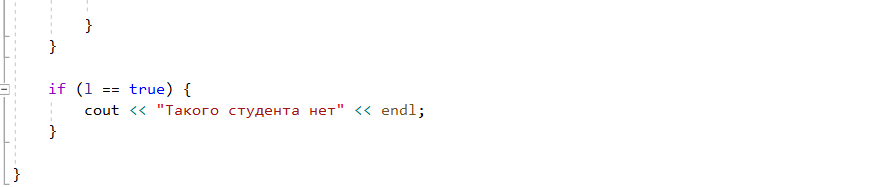


**5.3.** Поиск методом Алгоритма Бойера-Мура реализован через создание таблицы стоп символов, с последующим сравнением элементов строки с заданным ключом:





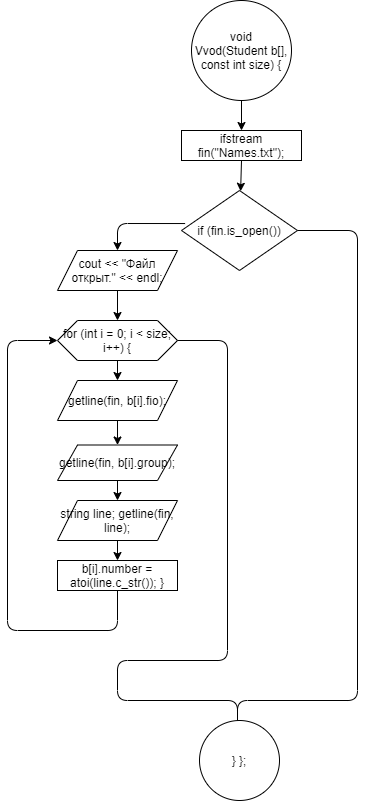
5.4. Реализация алгоритма Кнута-Морриса-Прата произведена через цикл for, с постепенным переходом ко всем элементам структуры, внутри которого происходит посимвольное сравнение строки с ключомна основе перехода к элементам строки равным первому элементу подстроки:

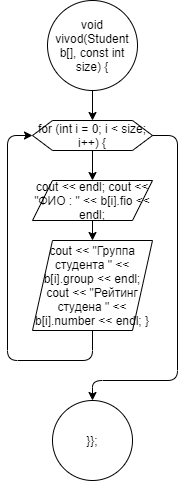


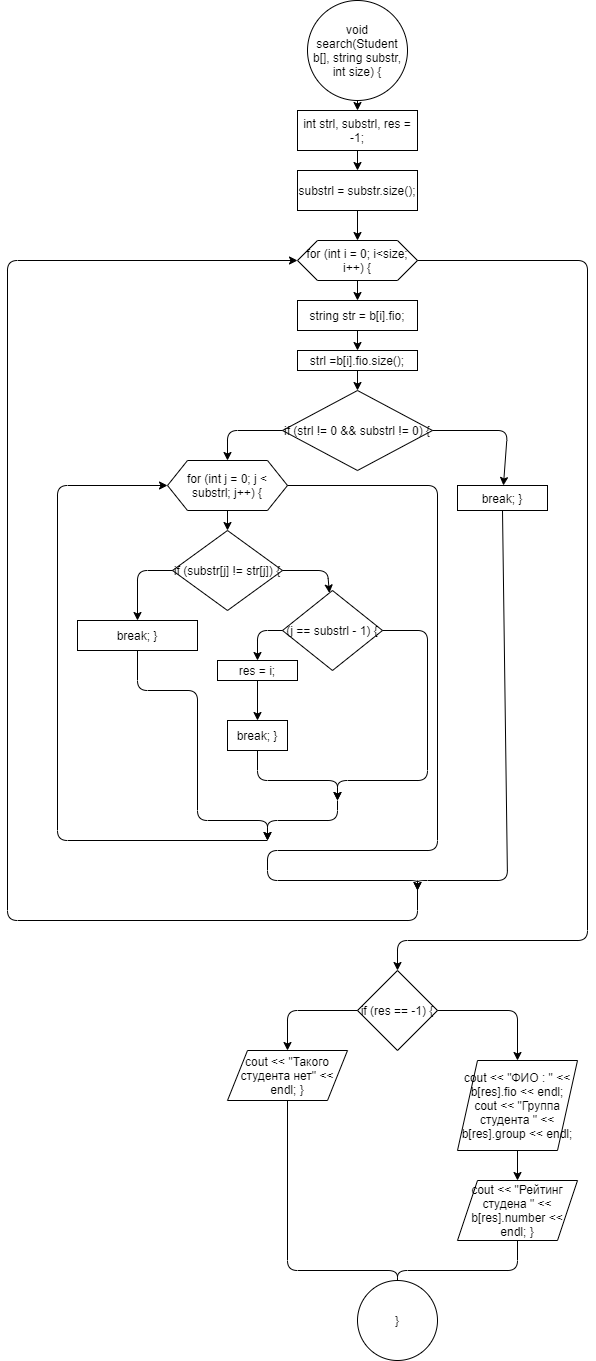
**5.4.** Организация меню основана на приеме введенного пользователем значения и её дальнейшей обработкой, завязанной на многоступенчатом условии if:

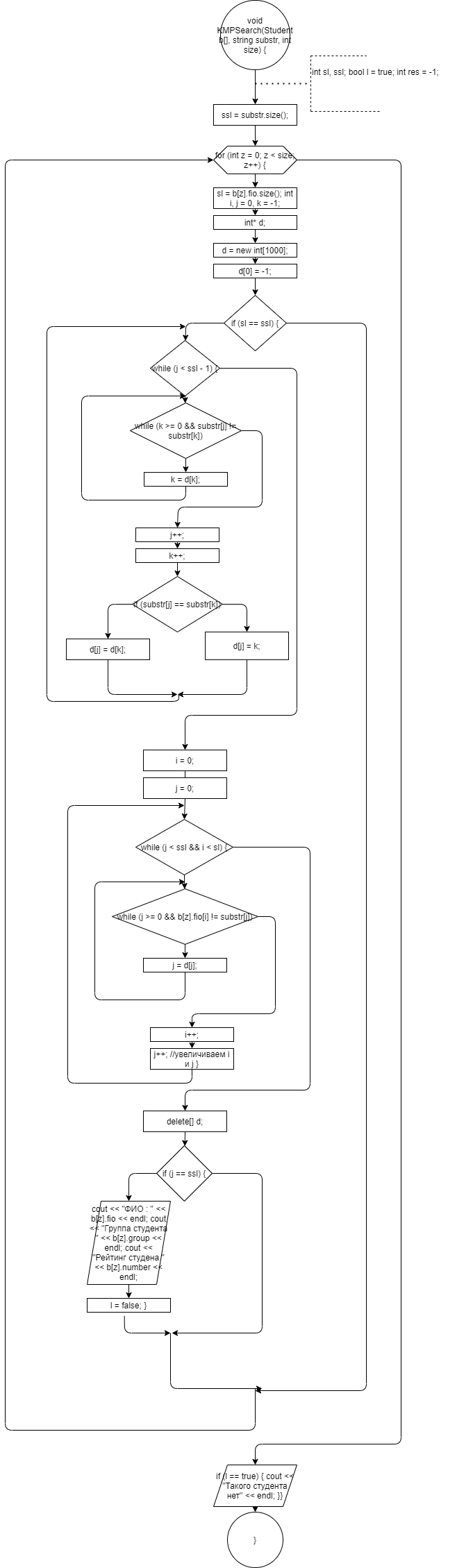


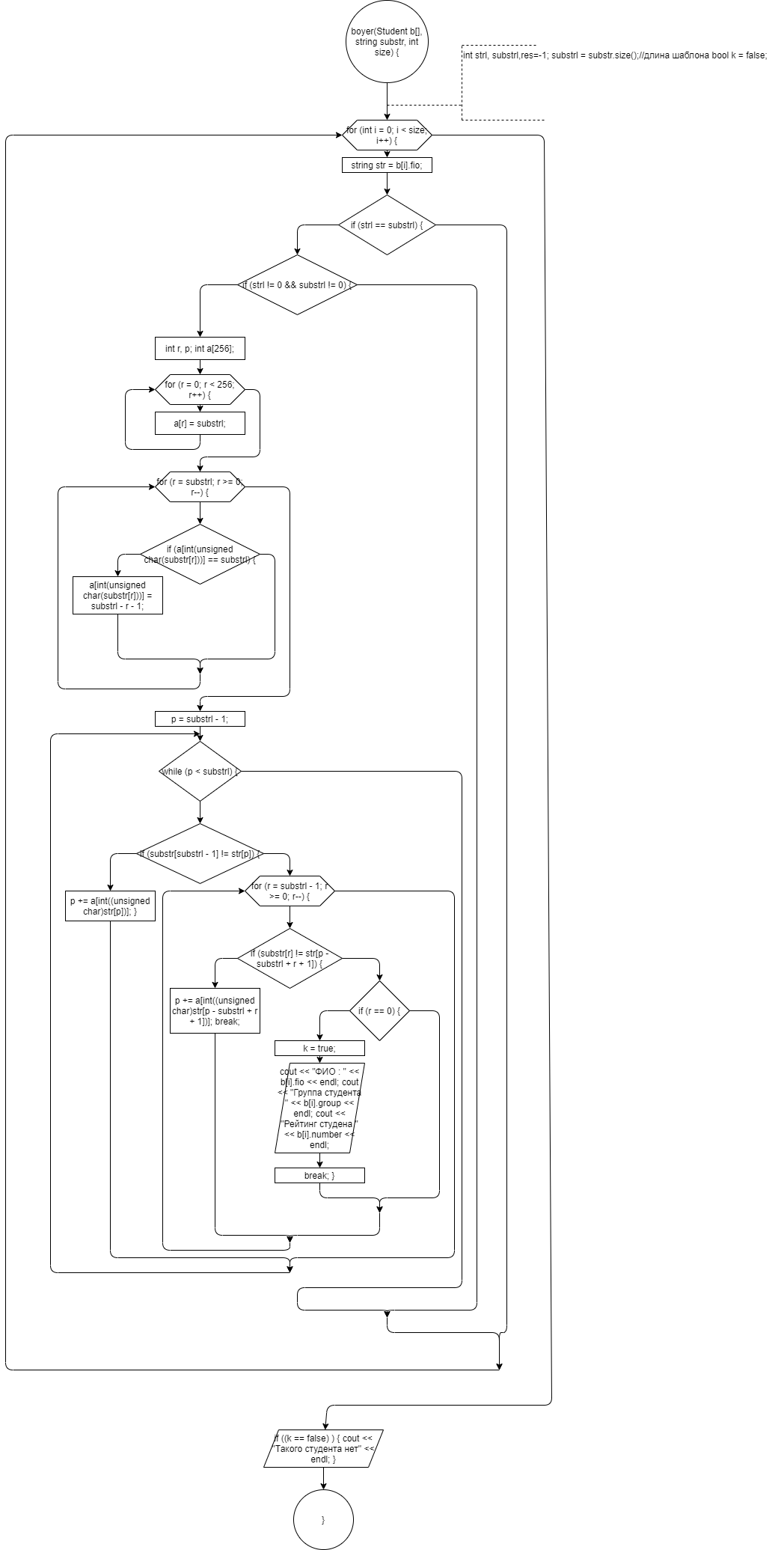
**Блок-схема программы**











**Решение**

#include <iostream>

#include <stdio.h>

#include <string>

#include <fstream>

#include <iostream>

#include <Windows.h>

using namespace std;

struct Student {

string fio;

string group;

int number;

};

void Vvod(Student b[], const int size) {

ifstream fin("Names.txt");

if (fin.is\_open())//Если открытие файла прошло успешно

{

cout << "Файл открыт." << endl;

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

getline(fin, b[i].fio);

getline(fin, b[i].group);

string line;

getline(fin, line);

b[i].number = atoi(line.c\_str());

}

}

};

void vivod(Student b[], const int size) {

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

cout << endl;

cout << "ФИО : " << b[i].fio << endl;

cout << "Группа студента " << b[i].group << endl;

cout << "Рейтинг студена " << b[i].number << endl;

}

};

void search(Student b[], string substr, int size) {

int strl, substrl, res = -1;

substrl = substr.size();

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

string str = b[i].fio;

strl = b[i].fio.size();

if (strl != 0 && substrl != 0) {

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

if (substr[j] != str[j]) {

break;

}

else if (j == substrl - 1) {

res = i;

break;

}

}

}

else break;

}

if (res == -1) {

cout << "Такого студента нет" << endl;

}

else {

cout << "ФИО : " << b[res].fio << endl;

cout << "Группа студента " << b[res].group << endl;

cout << "Рейтинг студена " << b[res].number << endl;

}

}

void boyer(Student b[], string substr, int size) {

int strl, substrl,res=-1;

substrl = substr.size();//длина шаблона

bool k = false;

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

string str = b[i].fio;

strl = b[i].fio.size();//длина строки

if (strl == substrl) {

if (strl != 0 && substrl != 0) {

int r, p;

int a[256];

for (r = 0; r < 256; r++) {

a[r] = substrl;

}

for (r = substrl; r >= 0; r--) {

if (a[int(unsigned char(substr[r]))] == substrl) {

a[int(unsigned char(substr[r]))] = substrl - r - 1;

}

}

p = substrl - 1;

while (p < substrl) {

if (substr[substrl - 1] != str[p]) {

p += a[int((unsigned char)str[p])];

}

else {

for (r = substrl - 1; r >= 0; r--) {

if (substr[r] != str[p - substrl + r + 1]) {

p += a[int((unsigned char)str[p - substrl + r + 1])];

break;

}

else if (r == 0) {

k = true;

cout << "ФИО : " << b[i].fio << endl;

cout << "Группа студента " << b[i].group << endl;

cout << "Рейтинг студена " << b[i].number << endl;

break;

}

}

break;

}

}

}

}

}

if ((k == false) ) {

cout << "Такого студента нет" << endl;

}

}

void KMPSearch(Student b[], string substr, int size) { //в качестве параметров в функцию передается строка и субстрока

int sl, ssl;

bool l = true;

int res = -1; //присваивается длина строки

ssl = substr.size();

for (int z = 0; z < size; z++) {//присваивается длина субстроки

sl = b[z].fio.size(); //Если все нормально - поехали

int i, j = 0, k = -1;

int\* d;

d = new int[1000]; //создали динамический одномерный массив

d[0] = -1; //первый элемент делаем равным -1

if (sl == ssl) {

while (j < ssl - 1) {

//пока j < кол-ва эл-тов строки

while (k >= 0 && substr[j] != substr[k]) /\*пока k больше или равно 0 и j-тый элемент субстроки не равен

k-тому присваиваем k k-тый элемент динамического массива\*/

k = d[k];

j++; //увеличиваем j

k++;

if (substr[j] == substr[k]) //если j-ый элемент субстроки равен k-тому

d[j] = d[k]; //присваиваем j-тому элементу динамического массива k-тый

else //иначе

d[j] = k; //присваиваем k

}

i = 0;

j = 0; //обнулили i, j

while (j < ssl && i < sl) { //пока j < длины субстроки и i < длины строки

while (j >= 0 && b[z].fio[i] != substr[j]) // пока j >= 0 и i-ый элемент субстроки не равен j-ому

j = d[j]; //j присваивается j-ый элемент динамического массива

i++;

j++; //увеличиваем i и j

}

delete[] d; //удаляем динам. массив

if (j == ssl) {

cout << "ФИО : " << b[z].fio << endl;

cout << "Группа студента " << b[z].group << endl;

cout << "Рейтинг студена " << b[z].number << endl;

l = false;

}

}

}

if (l == true) {

cout << "Такого студента нет" << endl;

}

}

int main()

{

setlocale(LC\_CTYPE, "Russian");

int key = 0;

string substr;

const int size = 32;

Student\* b = new Student[size];

Vvod(b, size);

vivod(b, size);

int op = 0;

while (op != 4) {

cout << endl << "1-Поиск по ФИО (Бойер-Мур)" << endl << "2-Поиск по ФИО (Прямой)" << endl;

cout << "3-Поиск по ФИО(КМП)" << endl << "4-Завершение работы" << endl << "Введите номер вашей операции:" << endl;

cin >> op;

while ((op < 0) || (op > 5)) {

cout << "Введите номер вашей операции:" << endl;

cin >> op;

}

if (op == 1) {

cout << "Введите ФИО студента, которого вы ищите (Бойер-Мур)" << endl;

cin.ignore();

getline(cin, substr);

boyer(b, substr, size);

}

if (op == 2) {

cout << "Введите ФИО студента, которого вы ищите (Прямой)" << endl;

cin.ignore();

getline(cin, substr);

search(b, substr, size);

}

if (op == 3) {

cout << "Введите ФИО студента, которого вы ищите(КМП)" << endl;

cin.ignore();

getline(cin, substr);

KMPSearch(b, substr, size);

}

}

return 0;

}

**Скриншоты результатов работы программы**

