Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет  
Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

# Лабораторная работа "Сложные поиски"

Выполнил студент гр. РИС-24-3б

Жиряков Леонид Антонович

Проверил:

Доцент кафедры ИТАС   
Ольга Андреевна Полякова

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г. Пермь, 2024

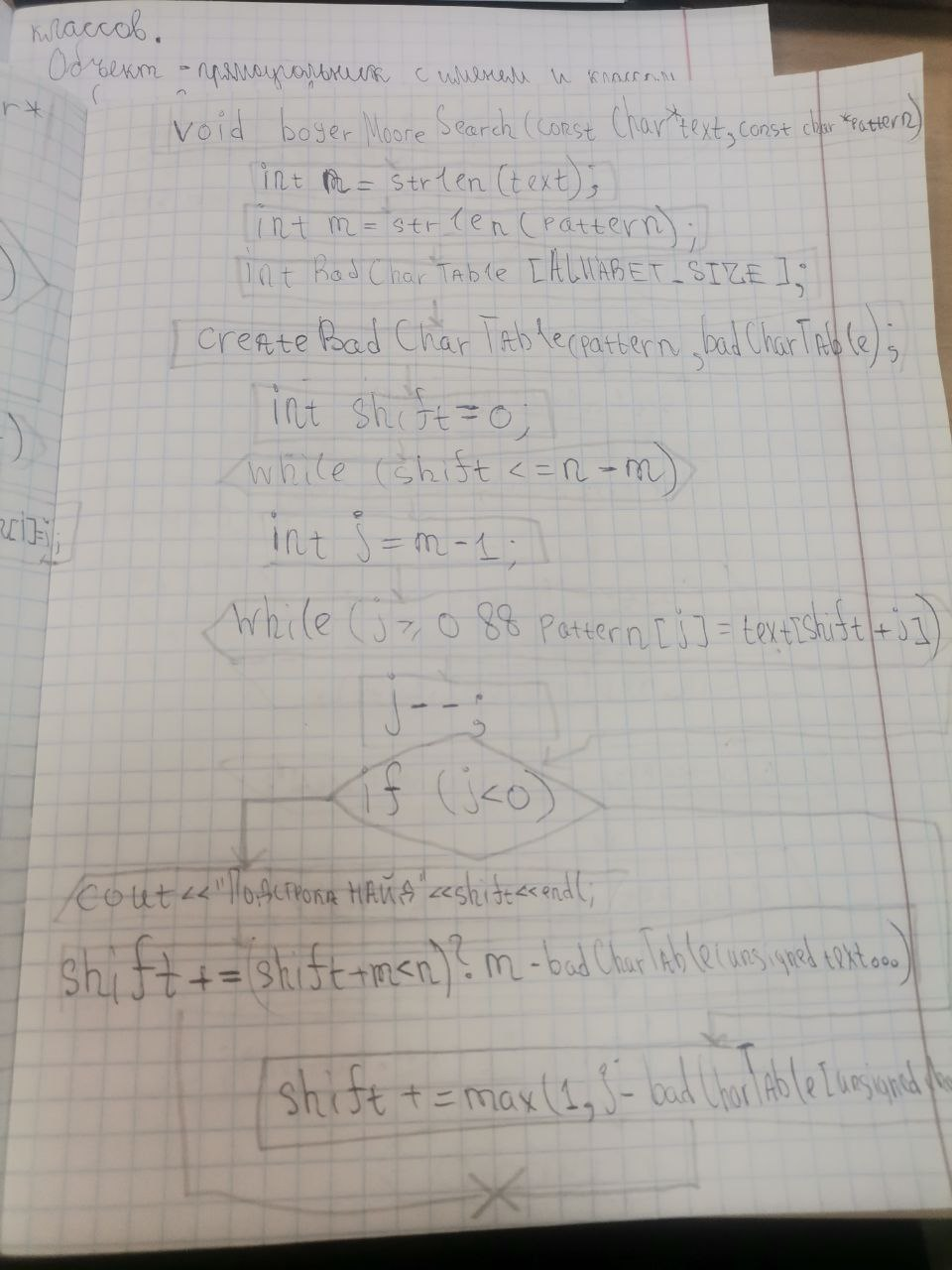
Постановка:

Выполнить сложные методы поиска:

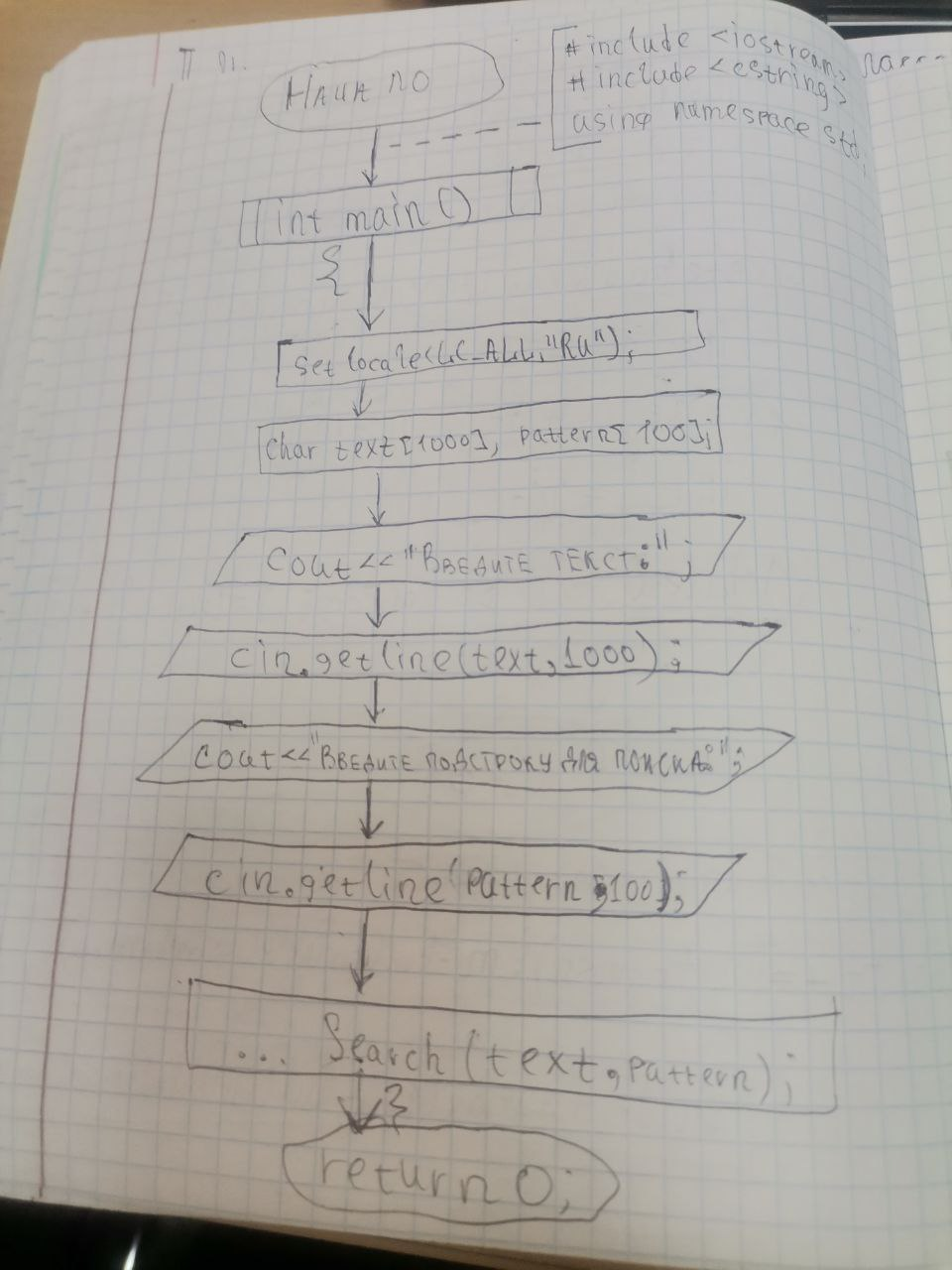
Бойер-Мур, Кнута-Пратта-Морра, Нахождение подстроки в строке.

Блок-схема:

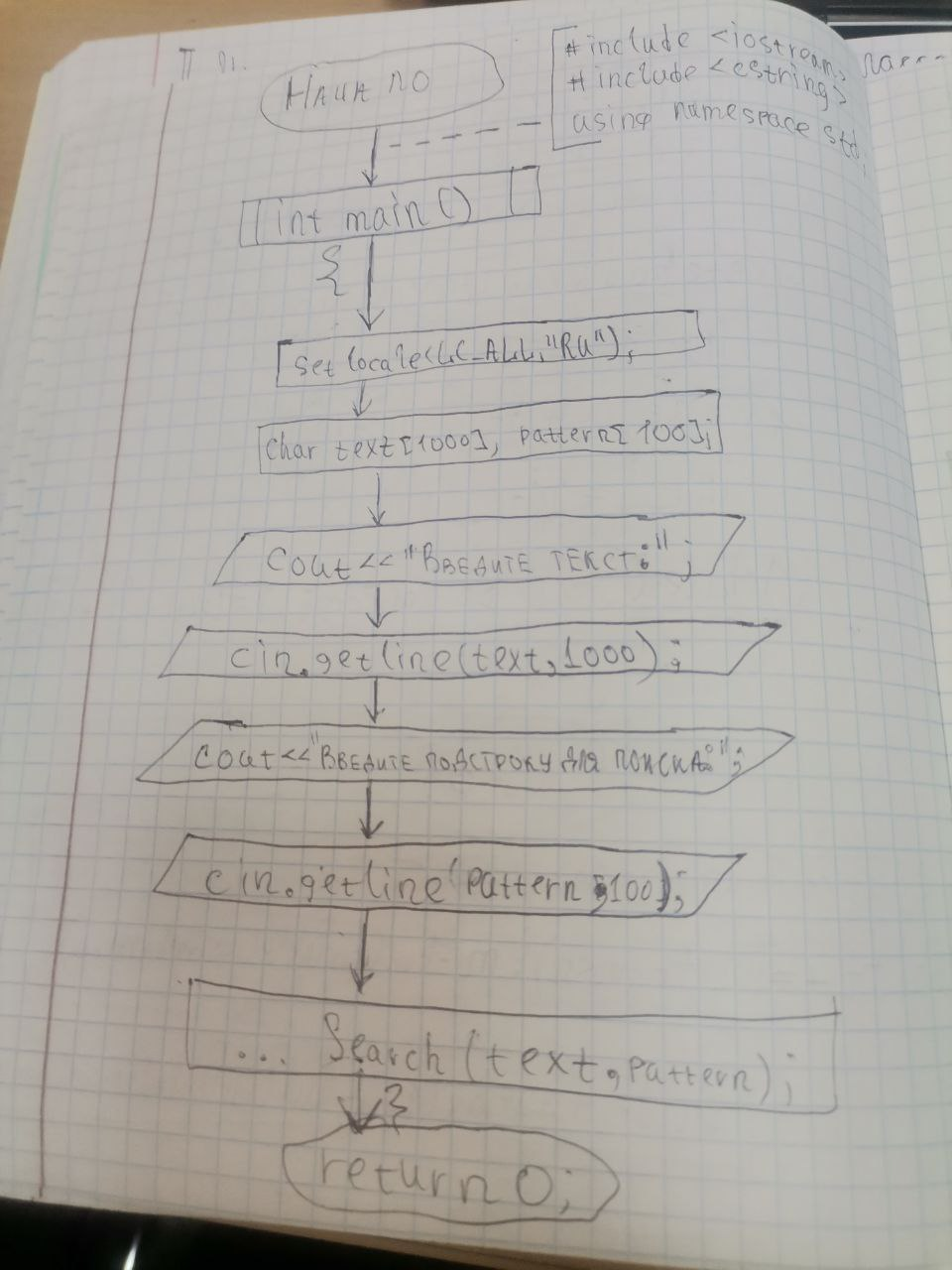
Бойер-Мур:

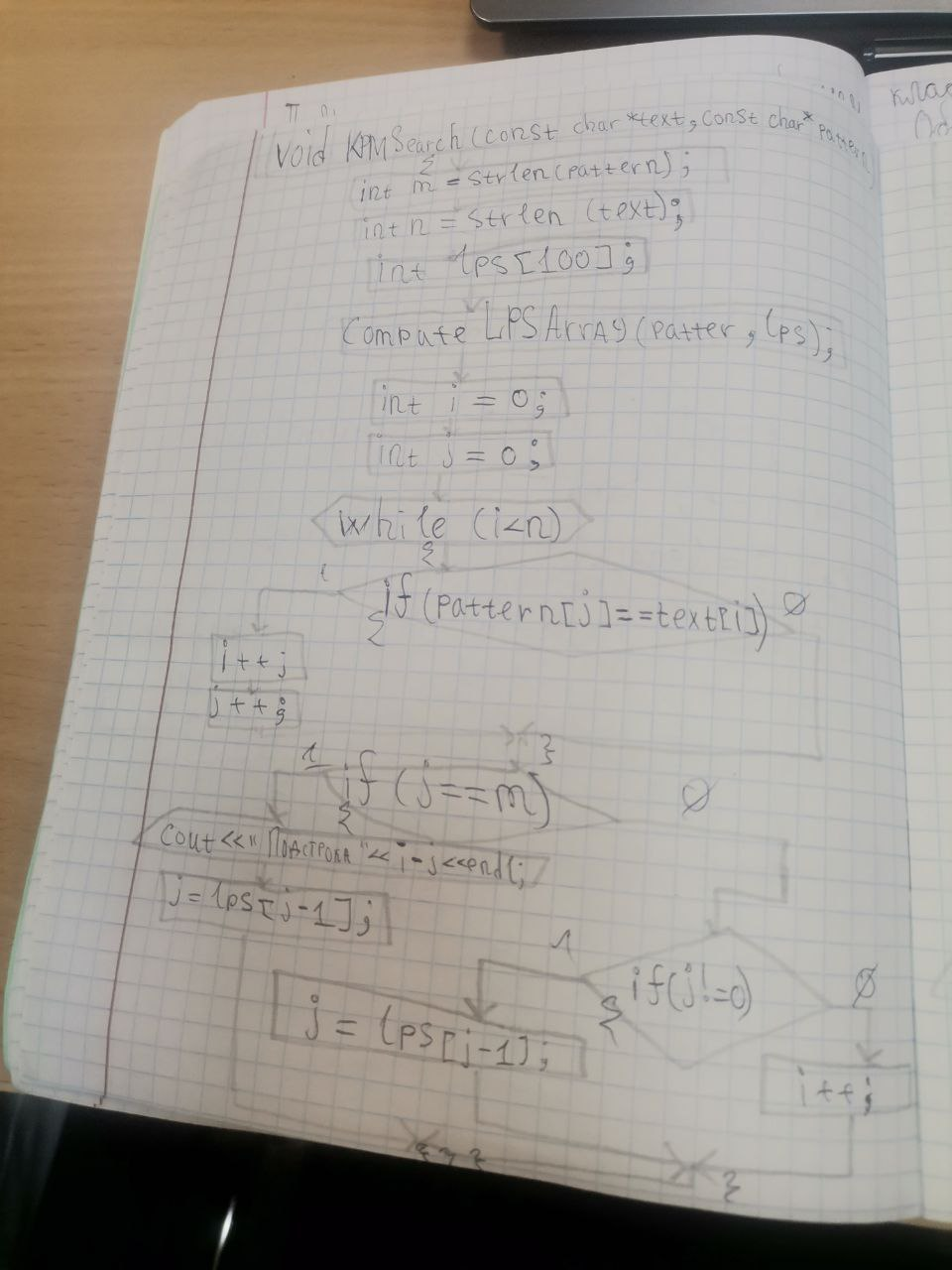


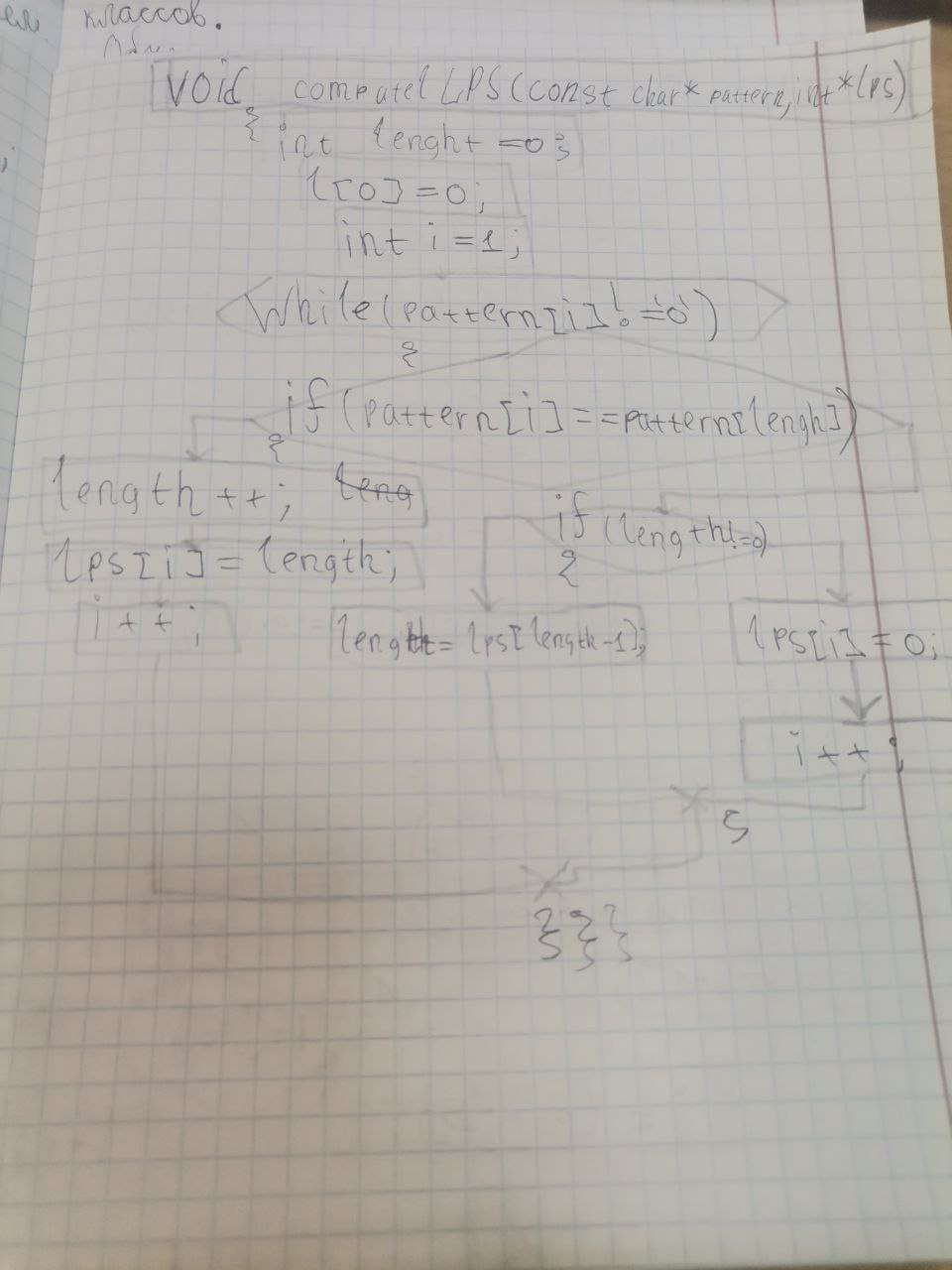




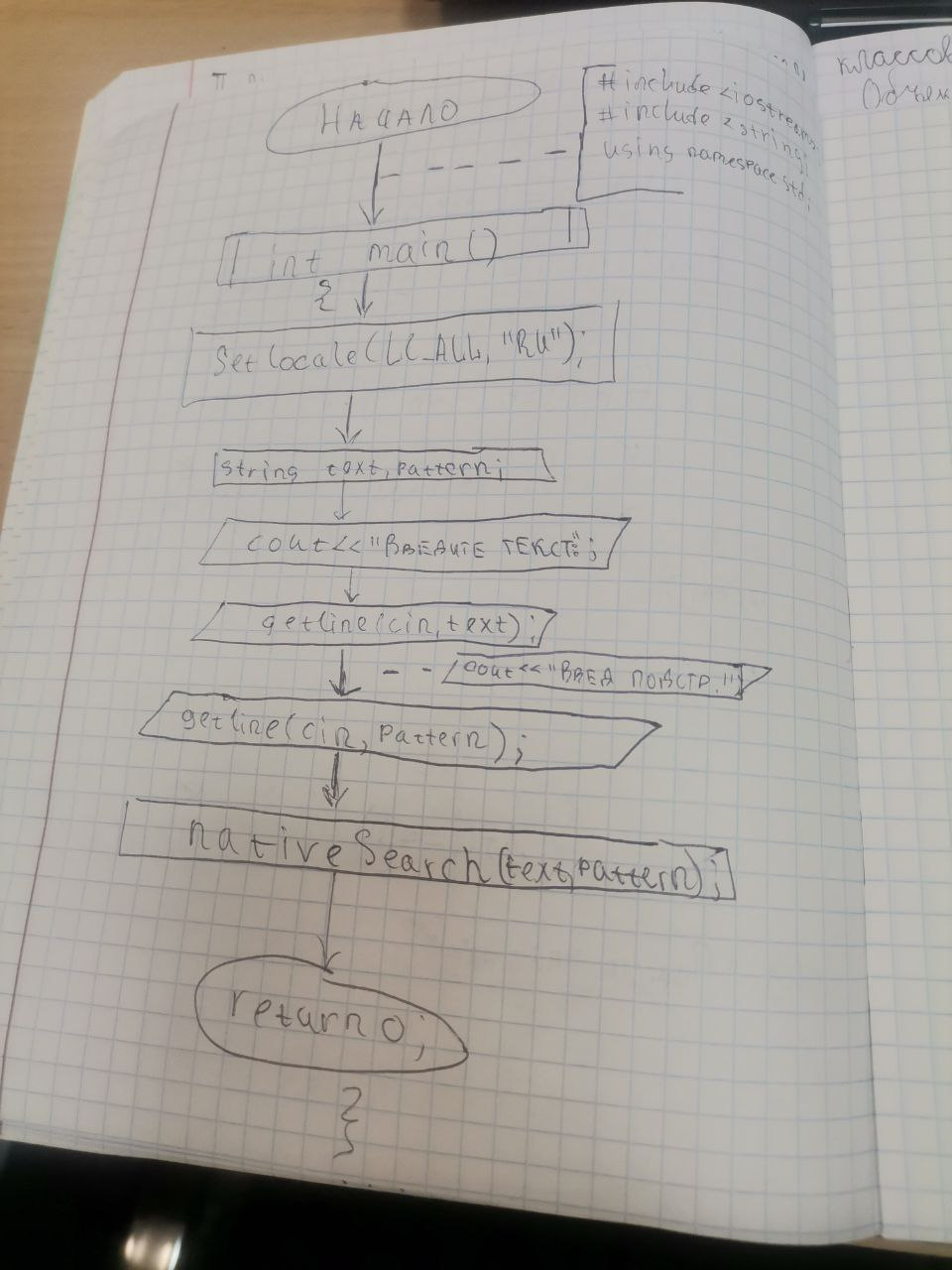
Кнут-Пратт-Морр:

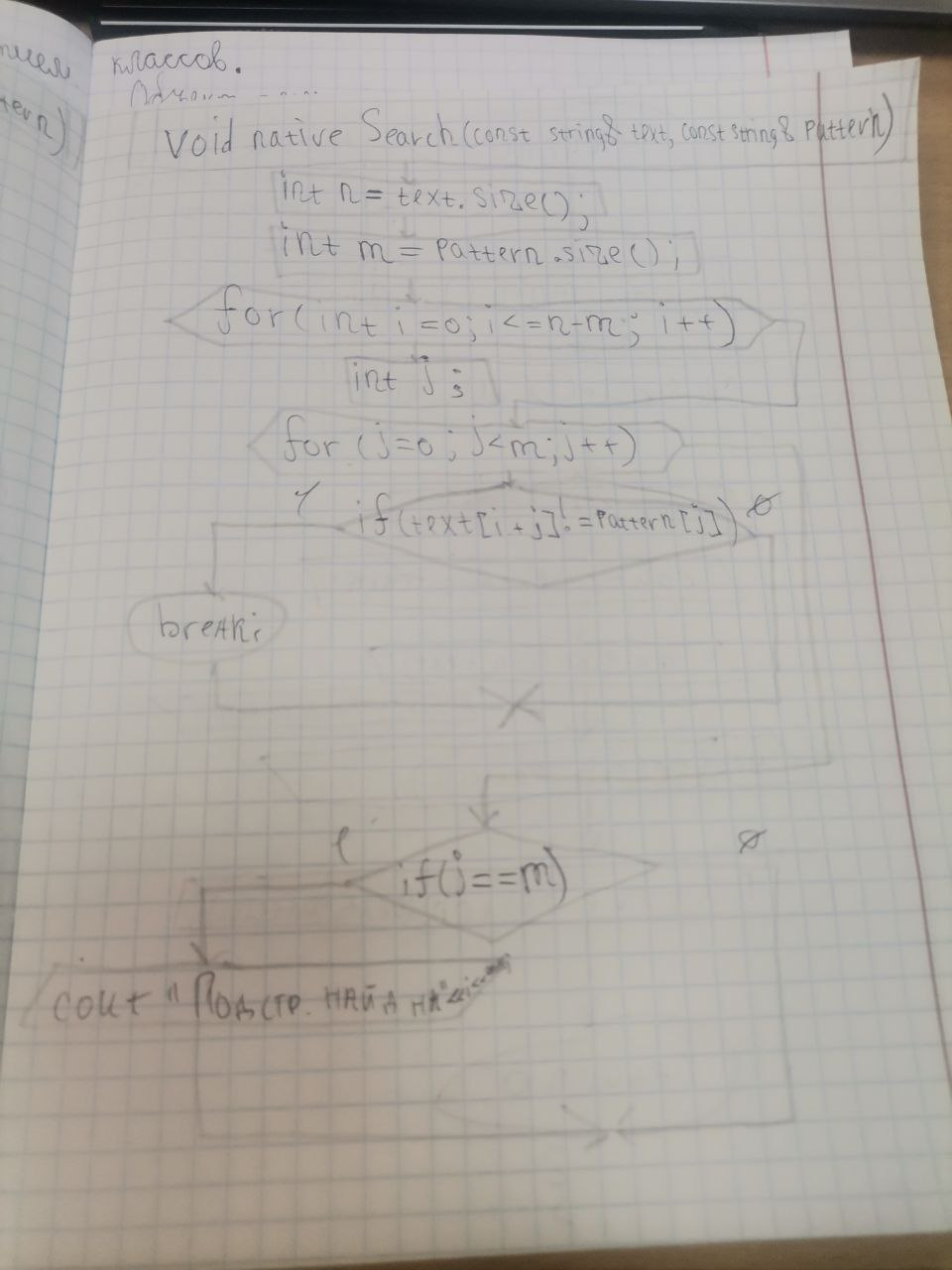






Нахождение подстроки в строке:





Код:

Бойер-Мур:

#include <iostream>

#include <cstring>

using namespace std;

// Максимальный размер алфавита (ASCII)

const int ALPHABET\_SIZE = 256;

// Функция для создания таблицы смещений

void createBadCharTable(const char\* pattern, int badCharTable[])

{

// Инициализируем все значения -1

for (int i = 0; i < ALPHABET\_SIZE; i++)

{

badCharTable[i] = -1;

}

// Заполняем таблицу последними вхождениями символов подстроки

for (int i = 0; pattern[i] != '\0'; i++)

{

badCharTable[(unsigned char)pattern[i]] = i;

}

}

// Функция поиска подстроки в строке с использованием алгоритма Бойера-Мура

void boyerMooreSearch(const char\* text, const char\* pattern)

{

int m = strlen(pattern);

int n = strlen(text);

int badCharTable[ALPHABET\_SIZE];

createBadCharTable(pattern, badCharTable);

int shift = 0; // Начальный сдвиг

while (shift <= n - m)

{

int j = m - 1;

// Пока символы совпадают, сдвигаем j

while (j >= 0 && pattern[j] == text[shift + j])

{

j--;

}

// Если j < 0, значит, найдено совпадение

if (j < 0)

{

cout << "Подстрока найдена в позиции: " << shift << endl;

// Сдвигаем на размер подстроки или по таблице смещений

shift += (shift + m < n) ? m - badCharTable[(unsigned char)text[shift + m]] : 1;

}

else

{

// Сдвигаем по таблице смещений

shift += max(1, j - badCharTable[(unsigned char)text[shift + j]]);

}

}

}

int main()

{

setlocale(LC\_ALL, "RU");

char text[1000], pattern[100];

cout << "Введите текст: ";

cin.getline(text, 1000);

cout << "Введите подстроку для поиска: ";

cin.getline(pattern, 100);

boyerMooreSearch(text, pattern);

return 0; // Завершение программы

}

Кнут-Пратт-Морр:

#include <iostream>

#include <cstring>

using namespace std;

// Функция для создания массива префиксов

void computeLPSArray(const char\* pattern, int\* lps)

{

int length = 0; // Длина предыдущего наибольшего префикса

lps[0] = 0; // lps[0] всегда 0

int i = 1;

// Построение массива lps

while (pattern[i] != '\0')

{

if (pattern[i] == pattern[length])

{

length++;

lps[i] = length;

i++;

}

else

{

if (length != 0)

{

length = lps[length - 1]; // Попробуем с предыдущего префикса

}

else

{

lps[i] = 0;

i++;

}

}

}

}

// Функция поиска подстроки методом КМП

void KMPSearch(const char\* text, const char\* pattern)

{

int m = strlen(pattern);

int n = strlen(text);

int lps[100]; // Массив для хранения префиксов

computeLPSArray(pattern, lps); // Создаем массив префиксов

int i = 0; // Индекс для текста

int j = 0; // Индекс для подстроки

while (i < n) {

if (pattern[j] == text[i])

{

i++;

j++;

}

if (j == m)

{

cout << "Подстрока найдена в позиции: " << i - j << endl;

j = lps[j - 1]; // Продолжаем искать

}

else if (i < n && pattern[j] != text[i])

{

// Не совпадает

if (j != 0)

{

j = lps[j - 1]; // Сдвигаем j по массиву lps

}

else

{

i++;

}

}

}

}

int main()

{

setlocale(LC\_ALL, "RU");

char text[1000], pattern[100];

cout << "Введите текст: ";

cin.getline(text, 1000);

cout << "Введите подстроку для поиска: ";

cin.getline(pattern, 100);

KMPSearch(text, pattern); // Вызов функции поиска

return 0; // Завершение программы

}

Нахождение подстроки в строке:

#include <iostream>

#include <string>

using namespace std;

// Функция для поиска подстроки в строке

void naiveSearch(const string& text, const string& pattern)

{

int n = text.size();

int m = pattern.size();

for (int i = 0; i <= n - m; i++)

{

int j;

for (j = 0; j < m; j++)

{

if (text[i + j] != pattern[j])

{

break; // Если символы не совпадают, выходим из внутреннего цикла

}

}

// Если все символы совпали

if (j == m)

{

cout << "Подстрока найдена на позиции: " << i << endl;

}

}

}

int main()

{

setlocale(LC\_ALL, "RU");

string text, pattern;

cout << "Введите текст: ";

getline(cin, text);

cout << "Введите подстроку для поиска: ";

getline(cin, pattern);

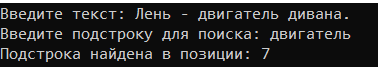
naiveSearch(text, pattern);

return 0;

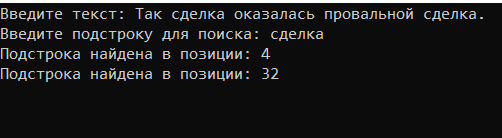
}

Вывод:

Бойер-Мур:



Кнут-Пратт-Морр:



Нахождение подстроки в строке:

