Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ПНИПУ)

ОТЧЁТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ

по теме:

МЕТОДЫ ПОИСКА

Выполнила: студентка группы РИС-22-1б

Черкасова А.А\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент кафедры ИТАС

Полякова О.А\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_/\_\_\_\_\_\_\_/\_\_\_\_\_г.

Пермь 2023

**Содержание**

[**Введение** 3](#_Toc127379672)

[**Анализ задачи** 3](#_Toc127379673)

[**Блок – схема** 4](#_Toc127379674)

[**Приложение А** 7](#_Toc127379675)

[**Приложение Б** 9](#_Toc127379676)

# **Введение**

**Постановка задачи:** Произвести поиск элемента по ключу методом:

* Прямого поиска (+)
* Интерполяционного поиска (+)
* Бинарный поиск
* Методом Бойера - Мура
* Методом Кнута - Морриса - Пратта

**Цель:** Найти элемент по ключу или его индекс.

Для достижения поставленной цели, необходимо решить следующие **задачи**:

* Провести анализ задачи
* Реализовать задачу на языке С++
* Составить блок-схему

# **Анализ задачи**

*А) Прямой поиск:*

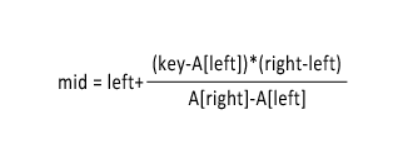
1. Установить i на начало строки S, т.е. i = 0.

2. Проверить, не вышло ли i + M за границу N строки S. Если да, то алгоритм завершен (вхождения нет).

3. Начиная с i-го символа s провести посимвольное сравнение строк S и p, т. е. S[i] и P[0], S[i+1] и P[1],…, S[i + M – 1] и P[M – 1].

4. Если хотя бы одна пара символов не совпала, то увеличить i и повторить шаг 2, иначе алгоритм завершен (вхождение найдено).

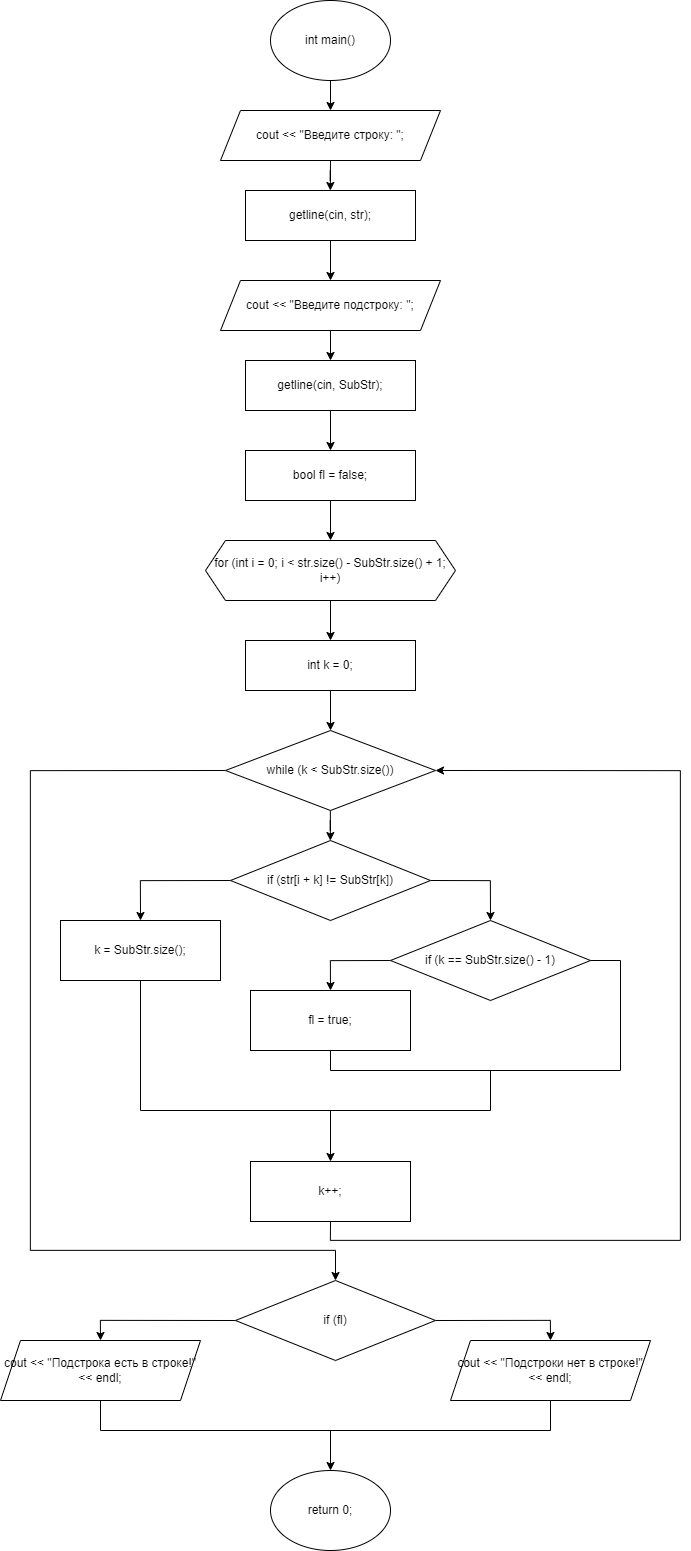
*Б) Интерполяционный поиск:*

**

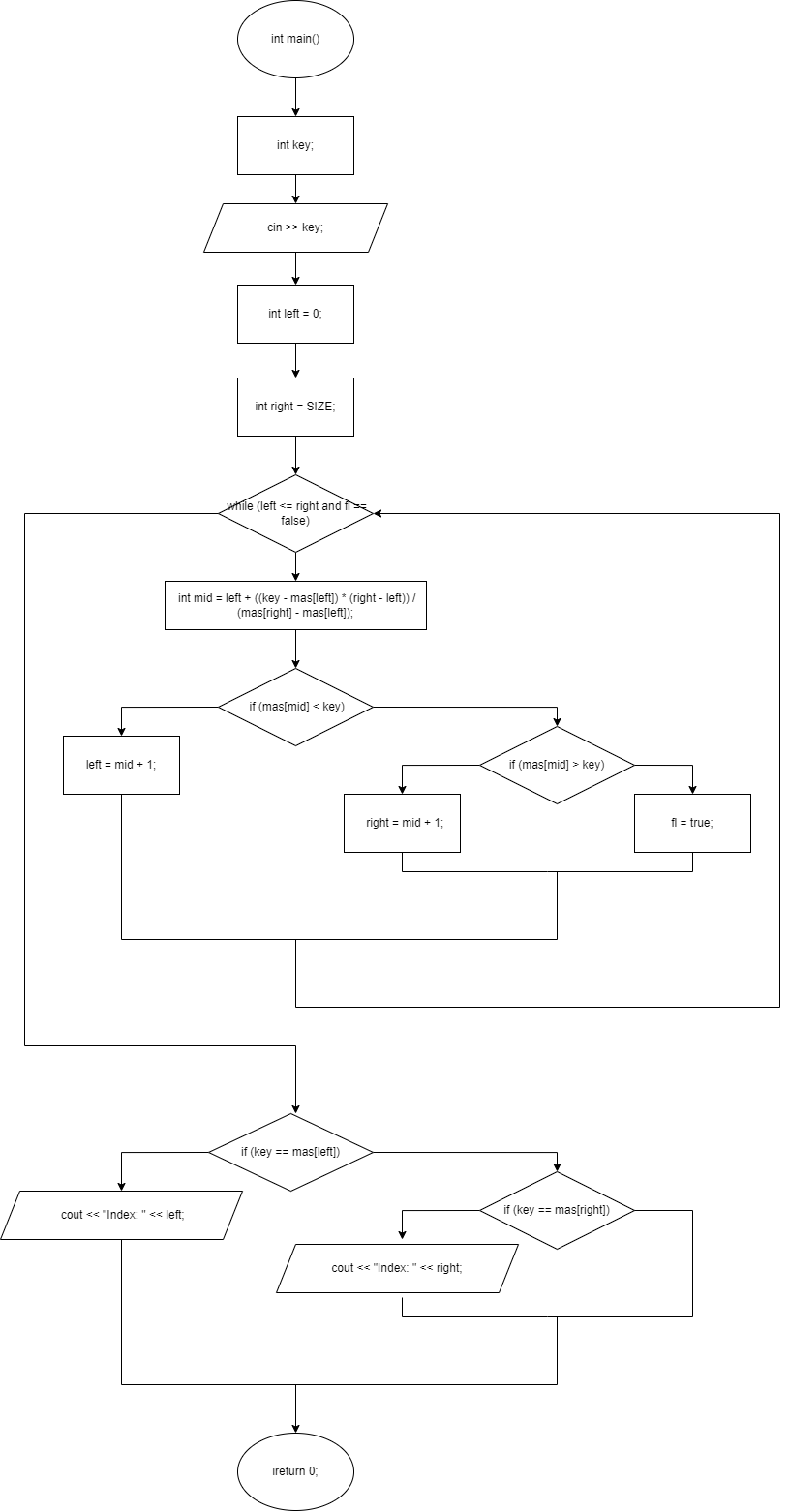
Здесь mid – номер элемента, с которым сравнивается значение ключа, key – ключ (искомый элемент), A – массив упорядоченных элементов, left и right – номера крайних элементов области поиска. Важно отметить, операция деления в формуле строго целочисленная, т. е. дробная часть, какая бы она ни была, отбрасывается.

# **Блок – схема**

А) Метод прямого поиска подстроки в строке

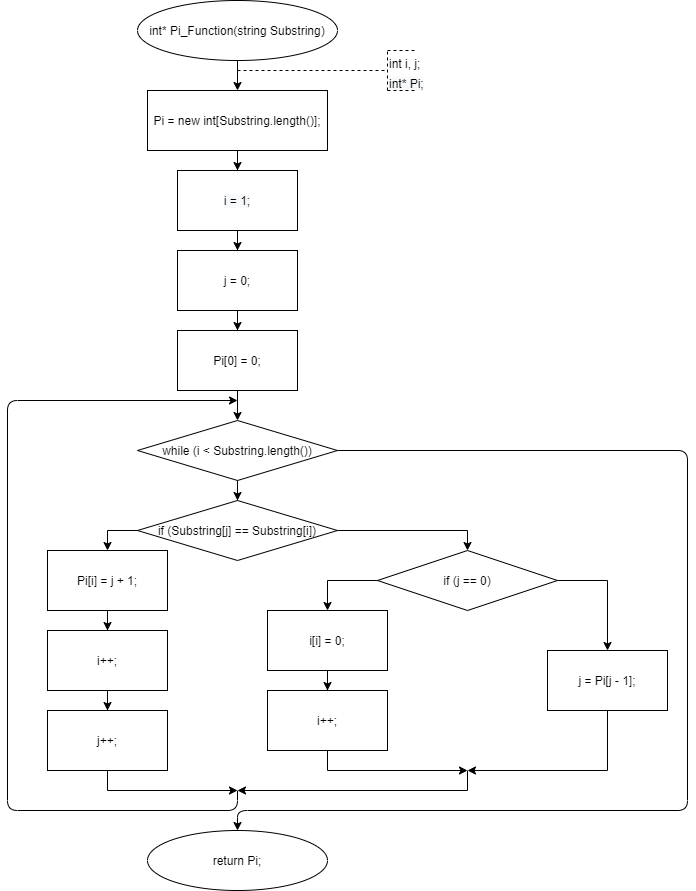
**

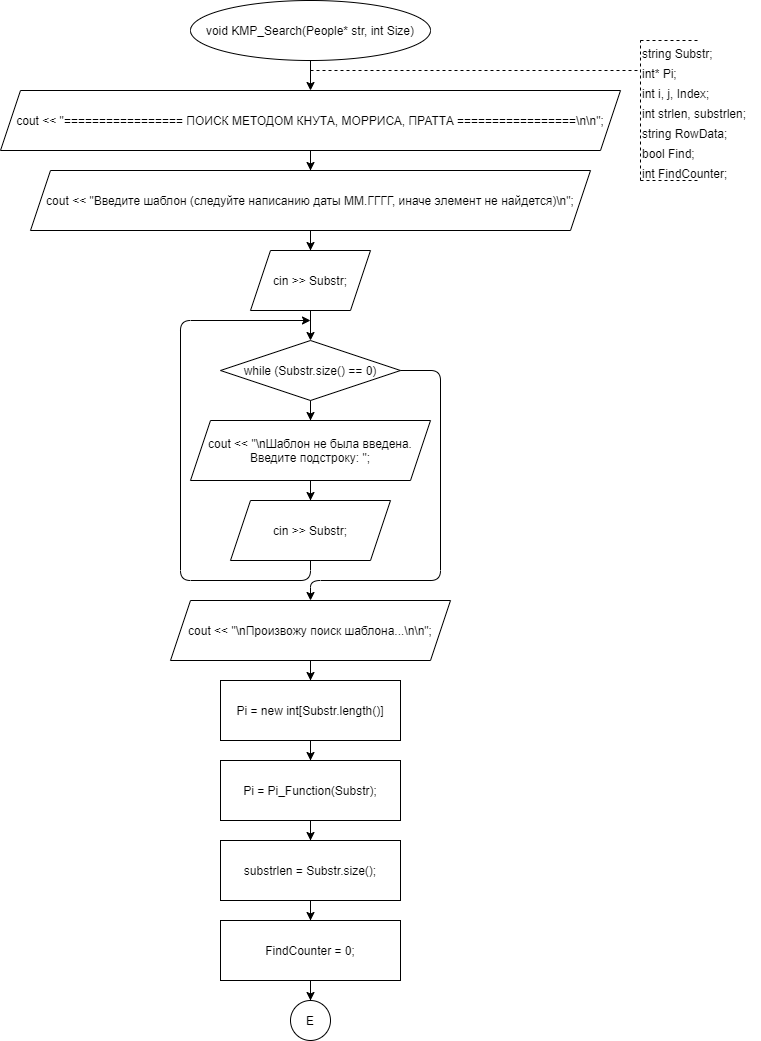
Б) Интерполяционный поиск



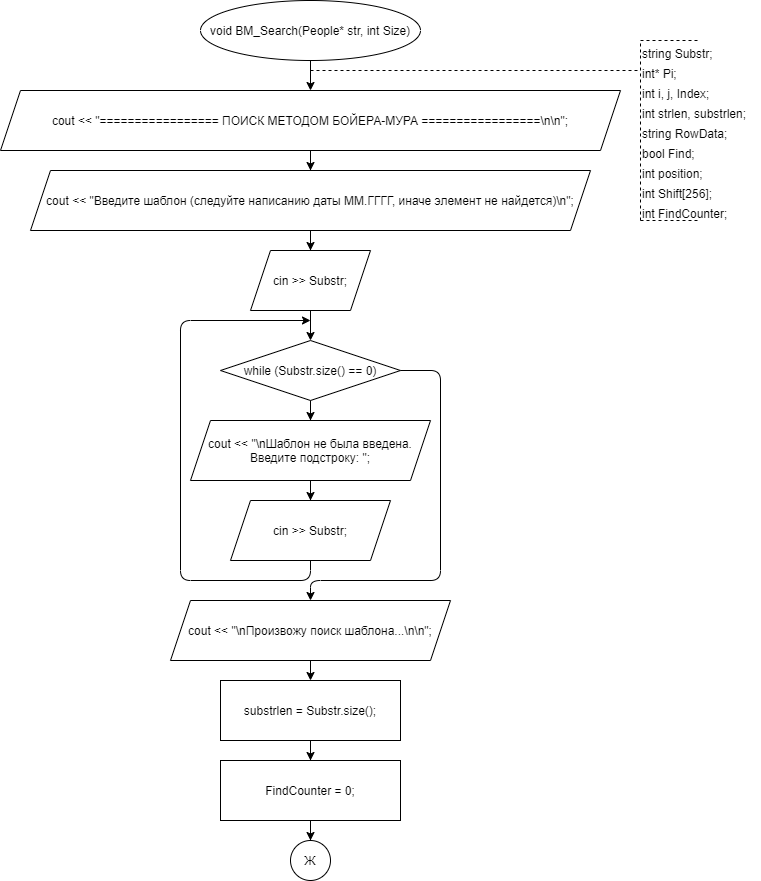
В) Бинарный поиск

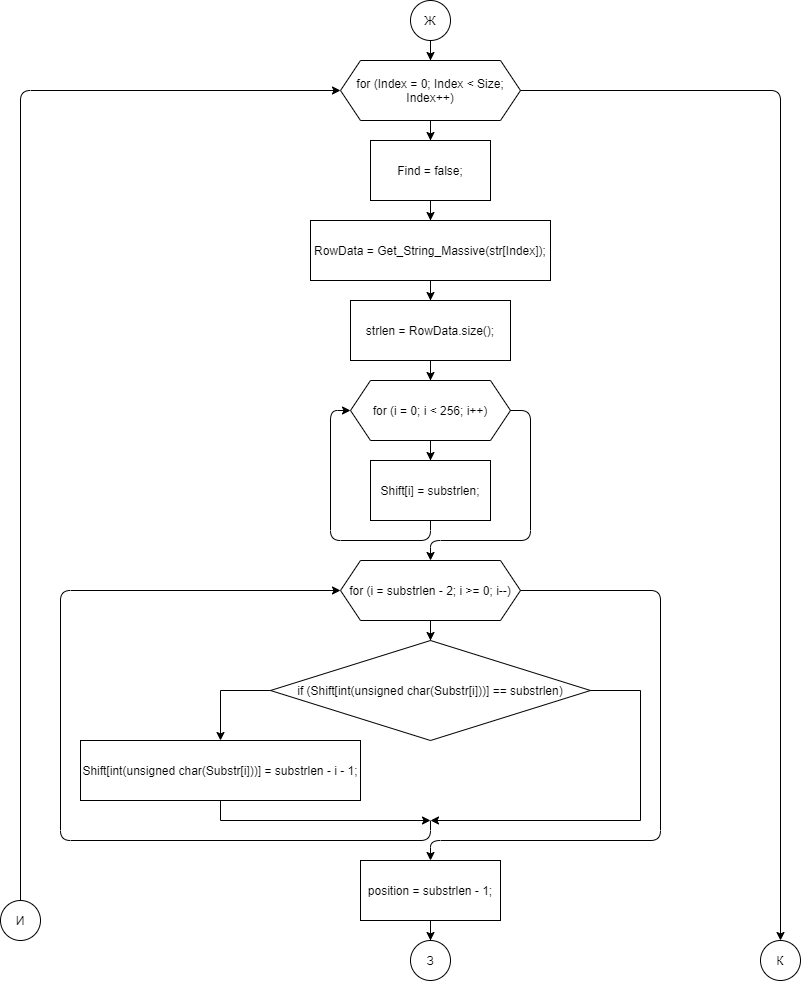
Г) Кнута – Морриса – Пратта





Г) Поиск Бойера – Мура





**Приложение А**

# **Листинг программы**

А) Метод прямого поиска подстроки в строке

// прямой поиск подстроки в строке

#include <iostream>

#include <string>

using namespace std;

int main()

{

system("chcp 1251>NULL");

string str;

cout << "Введите строку: ";

getline(cin, str);

string SubStr;

cout << "Введите подстроку: ";

getline(cin, SubStr);

bool fl = false;

for (int i = 0; i < str.size() - SubStr.size() + 1; i++)

{

int k = 0;

while (k < SubStr.size())

{

if (str[i + k] != SubStr[k])

{

k = SubStr.size();

}

else

{

if (k == SubStr.size() - 1)

fl = true;

}

k++;

}

}

if (fl)

cout << "Подстрока есть в строке!" << endl;

else

cout << "Подстроки нет в строке!" << endl;

}

Б) Интерполяционный поиск

// интерполяционный поиск

#include <iostream>

#include <ctime>

using namespace std;

void SORT(int\* m, int start, int end) // быстрая сортировка

{

int i = start;

int j = end;

int x = m[(i + j) / 2];

while (i <= j)

{

while (m[i] < x)

i++;

while (m[j] > x)

j--;

if (i <= j)

{

swap(m[i], m[j]);

i++;

j--;

}

}

if (j > start)

SORT(m, start, j);

if (i < end)

SORT(m, i, end);

}

int main()

{

bool fl = false;

int SIZE;

cout << "Enter N: ";

cin >> SIZE;

int\* mas = new int[SIZE];

srand(time(NULL));

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

mas[i] = rand() % 31 - 20;

SORT(mas, 0, SIZE - 1);

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

cout << mas[i] << " ";

cout << endl;

int key;

cout << "Enter key: ";

cin >> key;

int left = 0;

int right = SIZE;

while (left <= right and fl == false)

{

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

if (mas[mid] < key)

{

left = mid + 1;

}

else

{

if (mas[mid] > key)

right = mid + 1;

else

fl = true;

}

}

if (key == mas[left])

cout << "Index: " << left;

else

{

if (key == mas[right])

cout << "Index: " << right;

}

delete[] mas;

}

cout << "конечный массив: ";

PRINT(mas);

delete[] mas;

mas = nullptr;

}

}

В) Бинарный поиск

// бинарный поиск

#include <iostream>

#include <ctime>

using namespace std;

int N;

void SORT(int\* m, int start, int end) //быстрая сортировка

{

int i = start;

int j = end;

int x = m[(i + j) / 2];

while (i <= j)

{

while (m[i] < x)

i++;

while (m[j] > x)

j--;

if (i <= j)

{

swap(m[i], m[j]);

i++;

j--;

}

}

if (j > start)

SORT(m, start, j);

if (i < end)

SORT(m, i, end);

}

void PRINT(int\* m)

{

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

cout << m[i] << " ";

cout << endl;

}

int main()

{

cout << "Enter N: ";

cin >> N;

int\* mas = new int[N];

srand(time(NULL));

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

mas[i] = rand() % 101 - 50;

PRINT(mas);

int start = 0;

int end = N - 1;

SORT(mas, start, end);

PRINT(mas);

int k;

cout << "Enter key: ";

cin >> k;

bool fl = false;

int mid;

while (start <= end and fl == false)

{

mid = (start + end) / 2;

if (mas[mid] == k)

fl = true;

if (mas[mid] > k)

end = mid - 1;

if (mas[mid] < k)

start = mid + 1;

}

cout << "Index: " << mid;

}

Г) Кнута- Морриса-Пратта

// поисk Кнутта-Мориса-Пратта

#include <iostream>

#include <string>

using namespace std;

int\* func(string substr) // Функция формирвания префикс-функции (массив сдвига)

{

int size = substr.size();

int\* tab = new int[size];

int i = 1;

int j = 0;

tab[0] = 0; // Первый символ всегда имеет сдвиг 0

while (i < size)

{

if (substr[j] == substr[i])

{

tab[i] = j + 1; // Сдвиг на позицию этого символа

i++;

j++;

}

else

{

if (j == 0)

{

tab[i] = 0; // Сдвиг на первый символ

i++;

}

else

{

j = tab[j - 1]; // Сдвиг на первое вхождение символа

}

}

}

return tab;

}

int search(string str, string substr)

{

int\* tmp = func(substr);

int res = -1;

int i = 0;

int j = 0;

while (i < str.size()) // Проход по строке

{

if (substr[j] == str[i])

{

if (j == substr.size() - 1) // Если совпали последний символ шаблона и символ строки и до этого совпадения у них не было

{

res = i - j;

break;

}

else // Если совпали не последний символ шаблона и символ строки (или совпадение уже произошло)

{

j++;

i++;

}

}

else if (substr[j] != str[i])

{

if (j == 0) // Если символ первый в шаблоне

{

i++; // Переходим на следующий символ

}

else // Если смивол не первый в шаблоне

{

j = tmp[j - 1]; // Сдигаем согласно префикс-функции

}

}

}

return res;

}

int main()

{

system("chcp 1251>NULL");

string str;

string substr;

cout << "Введите строку: ";

getline(cin, str);

cout << "Введите образ: ";

getline(cin, substr);

cout << "result: " << search(str, substr) << endl;

}

Д) Поиск Бойера – Мура

// поиск Бойера - Мура

#include <iostream>

#include <string>

using namespace std;

int\* table(string substr)

{

int tab[256];

int size = substr.size(); // длина подстроки

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

{

tab[i] = size; //заполняем массив длиной образа

}

for (int i = size - 2; i >= 0; i--) // substr[size - 2] - предпоследний символ образа (индекс)

{

if (tab[int(unsigned char(substr[i]))] == size) // Для символов образа находим их ячейку в массиве - код буквы - позиция в массиве - 1

{

tab[int(unsigned char(substr[i]))] = size - i - 1; // И присваиваем необходимое значение - удалённость символа от последнего

}

}

return tab;

}

int search(string str, string substr)

{

int size = substr.size();

int\* tab = table(substr);

int res = -1;

int i = size - 1;

while(i < str.size())

{

if (substr[size - 1] != str[i])

{

i += tab[int(unsigned char(str[i]))];

// break;

}

else

{

for (int j = size - 1; j >= 0; j--)

{

if (substr[j] != str[i - size + j + 1]) // Если обнаружено несовпадение

{

i += tab[int((unsigned char)str[i])]; // Сдвиг

break; // Прерывание текущей проверки

}

if (j == 0 and substr[j] == str[i - size + j + 1]) // Если поиск окончен (Шаблон нашелся в подстроке)

{

res = i - size + 1;

break;

}

}

}

if (res == i - size + 1)

break;

}

return res;

}

int main()

{

system("chcp 1251>NULL");

string str;

string substr;

cout << "Введите строку: " ;

getline(cin, str);

cout << "Введите образ: ";

getline(cin, substr);

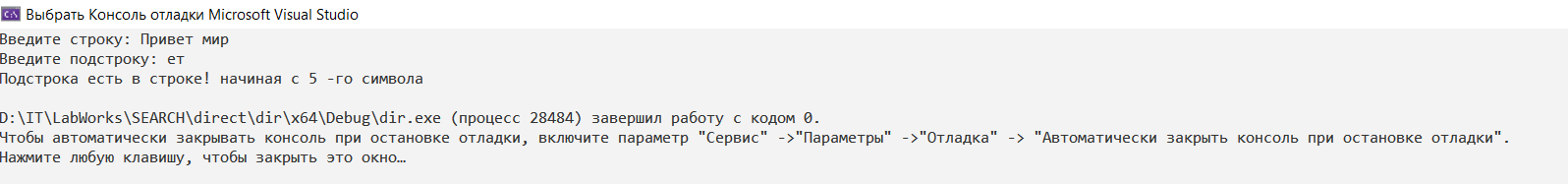
cout << "result: " << search(str, substr);

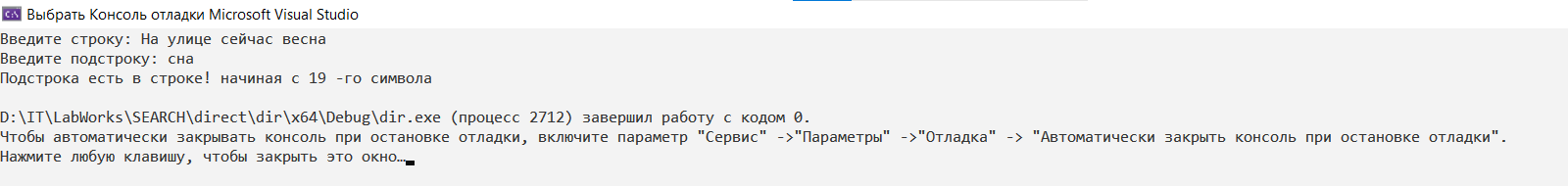
}

# **Приложение Б**

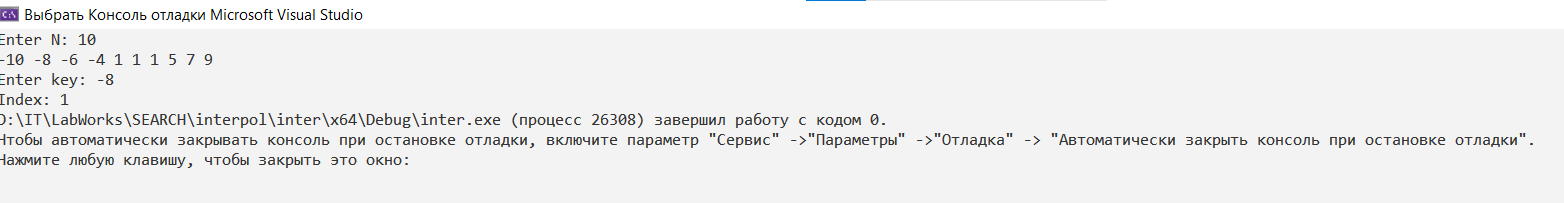
# **Результаты выполнения программы**

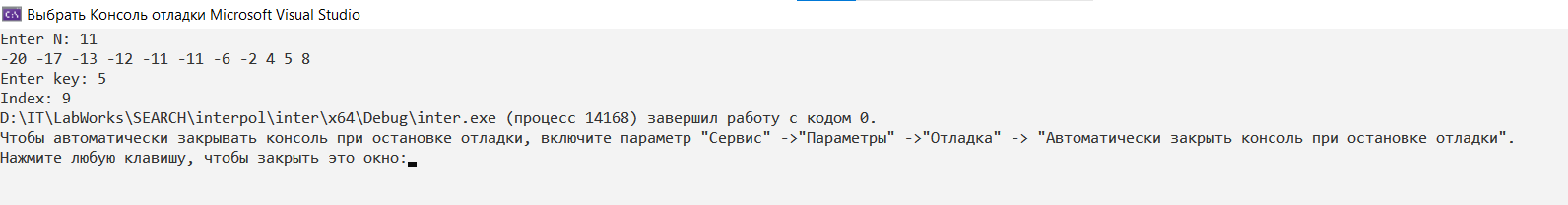
А) Прямой поиск



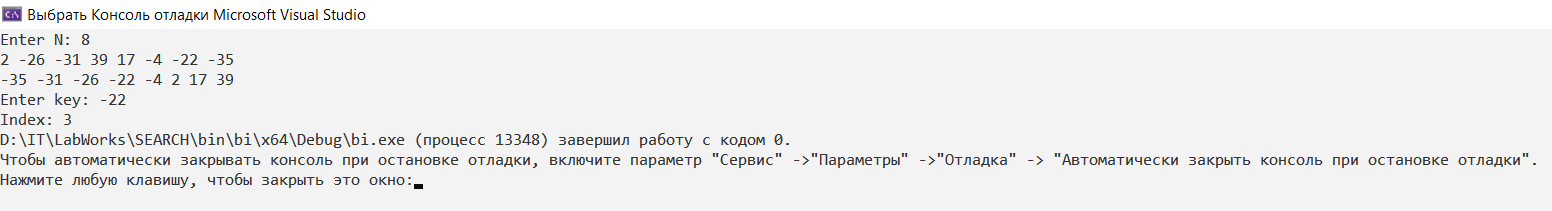


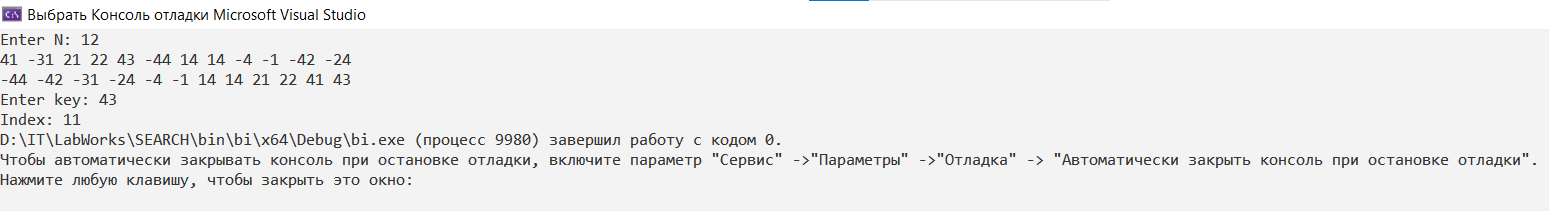
Б) Интерполяционный поиск



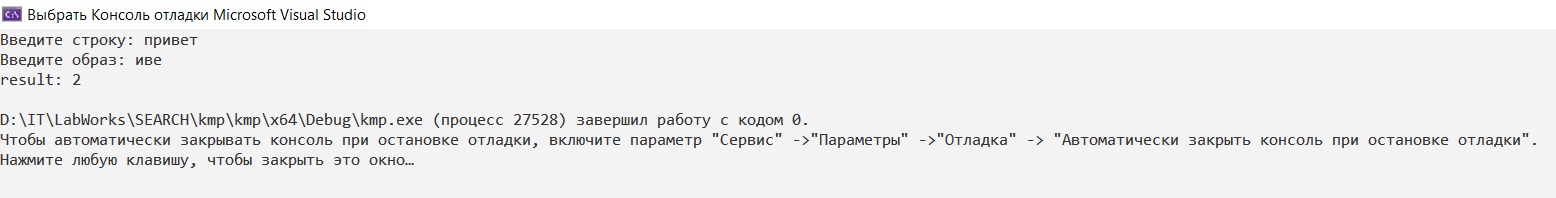
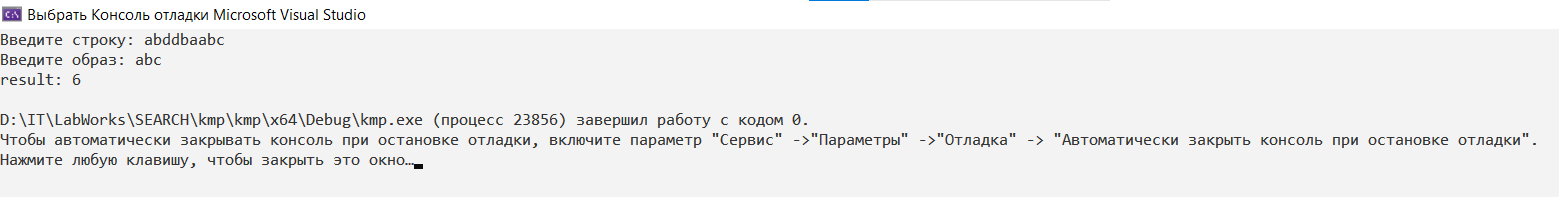


В) Бинарный поиск





Г) Кнута- Морриса-Пратта



Д) Бойера – Мура

