Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Пермский национальный исследовательский политехнический университет
Электротехнический факультет
Кафедра информационных технологий и автоматизированных систем

Лабораторная работа

"БМХ и КМП"

Вариант: 12

Выполнил студент ИВТ-24-2б:
Шишкин Максим Григорьевич

_____
(дата, подпись)

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

_____
(дата, подпись)

Пермь 2025

Содержание

1 Постановка задачи

Написать программы, реализующие алгоритмы Кнута-Морриса-Пратта (КМП) и Бойера — Мура — Хорспула (БМХ) для поиска подстроки в строке, с выводом шагов сравнения символов и визуализацией процесса работы алгоритмов.

2 КМП

2.1 Код на C++

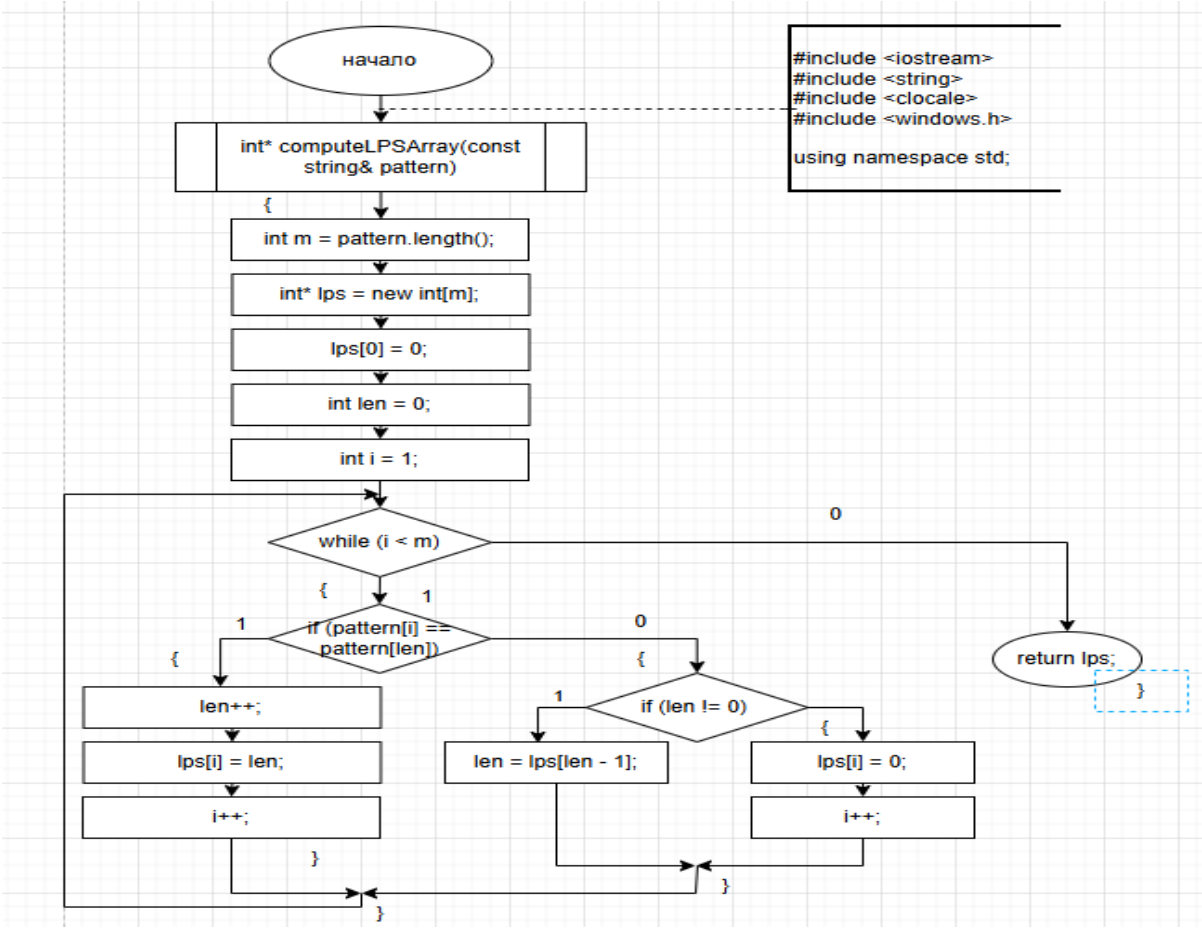
```cpp
#include <iostream>
#include <string>
#include <clocale>

using namespace std;

// Функция для вычисления таблицы префиксов (LPS)
int* computeLPSArray(const string& pattern) {
    int m = pattern.length();
    int* lps = new int[m];
    lps[0] = 0;

    int len = 0;
    int i = 1;

    while (i < m) {
        if (pattern[i] == pattern[len]) {
            len++;
            lps[i] = len;
            i++;
        }
        else {
            if (len != 0) {
                len = lps[len - 1];
            }
            else {
                lps[i] = 0;
                i++;
            }
        }
    }

    return lps;
}

// Функция поиска Кнута-Морриса-Пратта
int kmpSearch(const string& text, const string& pattern) {
    int n = text.length();
    int m = pattern.length();

    int* lps = computeLPSArray(pattern);

```

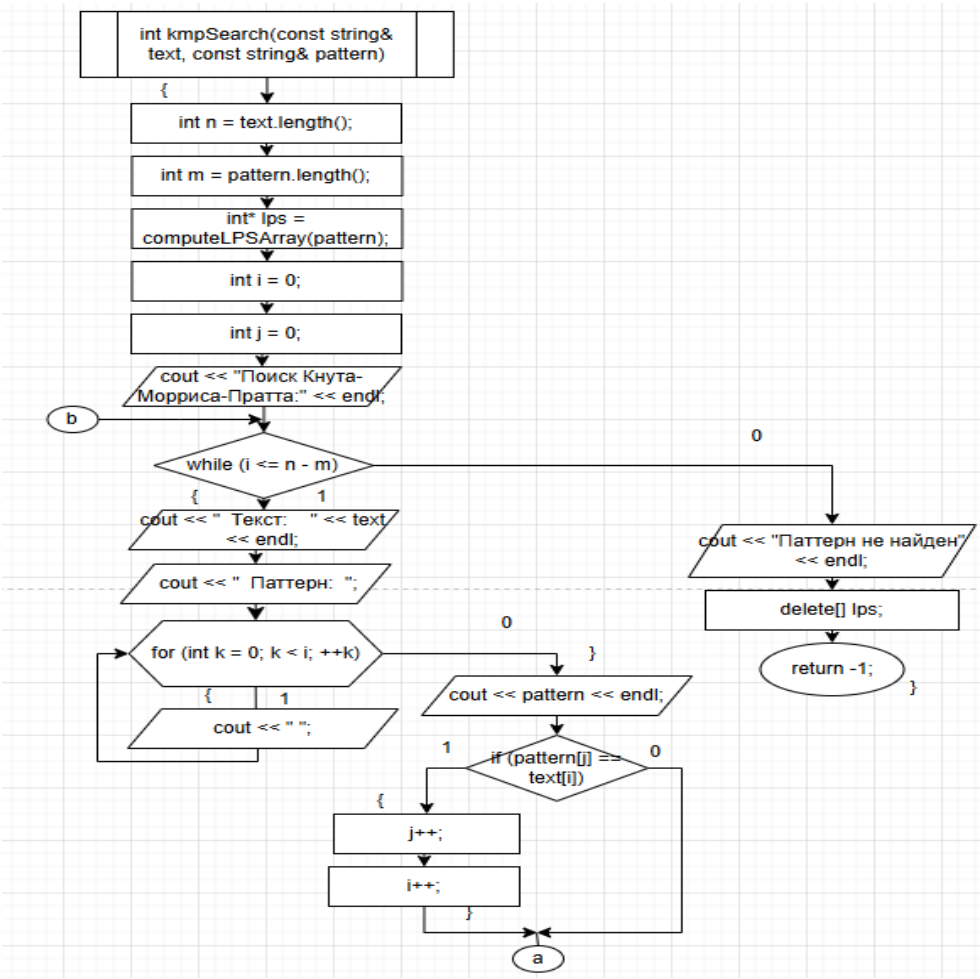
```
    int i = 0; // Индекс для text[]
    int j = 0; // Индекс для pattern[]

    cout << "Поиск Кнута-Морриса-Пратта:" << endl;

    while (i < n) {
        cout << "  Текст:    " << text << endl;
        cout << "  Паттерн:  ";
        for (int k = 0; k < i - j; ++k) {
            cout << " ";
        }
        cout << pattern << endl;

        if (pattern[j] == text[i]) {
            j++;
            i++;
        }

        if (j == m) {
            cout << "    Паттерн найден в позиции: " << i - j << endl;
            delete[] lps;
            return i - j;
        }

        if (i < n && pattern[j] != text[i]) {
            cout << "    Несовпадение в позиции текста: " << i << ", сдвиг паттерна." << endl;
            if (j != 0) {
                j = lps[j - 1];
            }
            else {
                i = i + 1;
            }
        }

        cout << endl;
    }

    cout << "Паттерн не найден" << endl;
    delete[] lps;
    return -1;
}

int main() {
    setlocale(LC_ALL, "ru");
    string text, pattern;

    cout << "Введите текст: ";
    getline(cin, text);

    cout << "Введите паттерн: ";
    getline(cin, pattern);

    int position = kmpSearch(text, pattern);

    if (position != -1) {
        cout << "Паттерн найден в позиции: " << position << endl;
    }
    else {
        cout << "Паттерн не найден в тексте." << endl;
    }

    return 0;
}
```

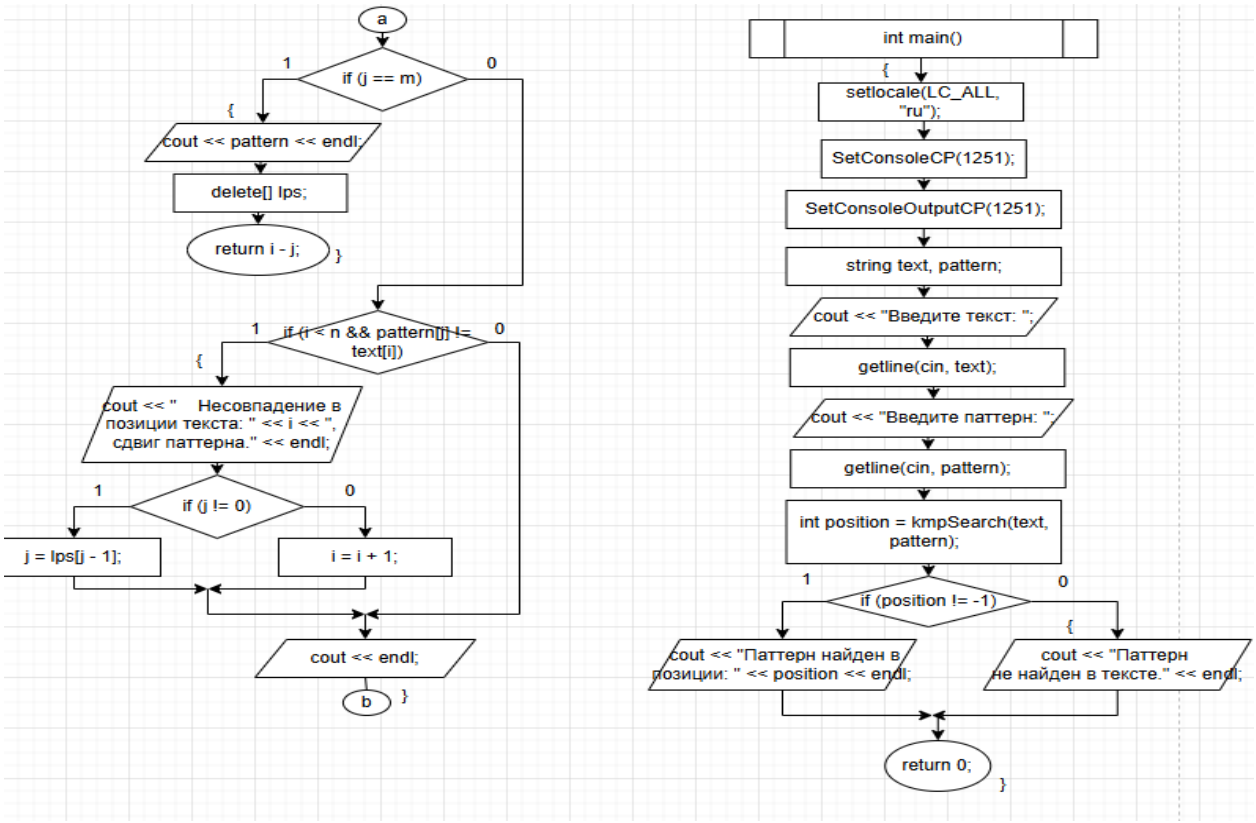
2.2 Результат работы

```
Введите текст: qwwwqw
Введите паттерн: wqw
Поиск Кнута-Морриса-Пратта:
  Текст:     qwwwqw
  Паттерн:   wqw
    Несовпадение в позиции текста: 0, сдвиг паттерна.

  Текст:     qwwwqw
  Паттерн:    wqw
    Несовпадение в позиции текста: 2, сдвиг паттерна.

  Текст:     qwwwqw
  Паттерн:     wqw
    Несовпадение в позиции текста: 3, сдвиг паттерна.

  Текст:     qwwwqw
  Паттерн:       wqw

  Текст:     qwwwqw
  Паттерн:       wqw

  Текст:     qwwwqw
  Паттерн:       wqw
    Паттерн найден в позиции: 3
Паттерн найден в позиции: 3
```

2.3 Блок-схема




int kmpSearch(const string& text, const string& pattern)
{
int n = text.length();
int m = pattern.length();
int* lps = computeLPSArray(pattern);
int i = 0;
int j = 0;
cout << "Поиск Кнута-Морриса-Пратта:" << endl;
b
while (i <= n - m)
1
0
{
cout << " Текст: " << text << endl;
cout << " Паттерн: ";
for (int k = 0; k < i; ++k)
{
1
cout << " ";
}
0
cout << pattern << endl;
if (pattern[j] == text[i])
1
0
{
j++;
i++;
}
a
cout << "Паттерн не найден" << endl;
delete[] lps;
return -1;
}


a
if (j == m)
1
0
{
cout << pattern << endl;
delete[] lps;
return i - j;
}
if (i < n && pattern[j] != text[i])
1
0
{
cout << " Несовпадение в позиции текста: " << i << ", сдвиг паттерна." << endl;
if (j != 0)
1
0
j = lps[j - 1];
i = i + 1;
cout << endl;
b
}
int main()
{
setlocale(LC_ALL, "ru");
SetConsoleCP(1251);
SetConsoleOutputCP(1251);
string text, pattern;
cout << "Введите текст: ";
getline(cin, text);
cout << "Введите паттерн: ";
getline(cin, pattern);
int position = kmpSearch(text, pattern);
if (position != -1)
1
0
{
cout << "Паттерн найден в позиции: " << position << endl;
cout << "Паттерн не найден в тексте." << endl;
return 0;
}

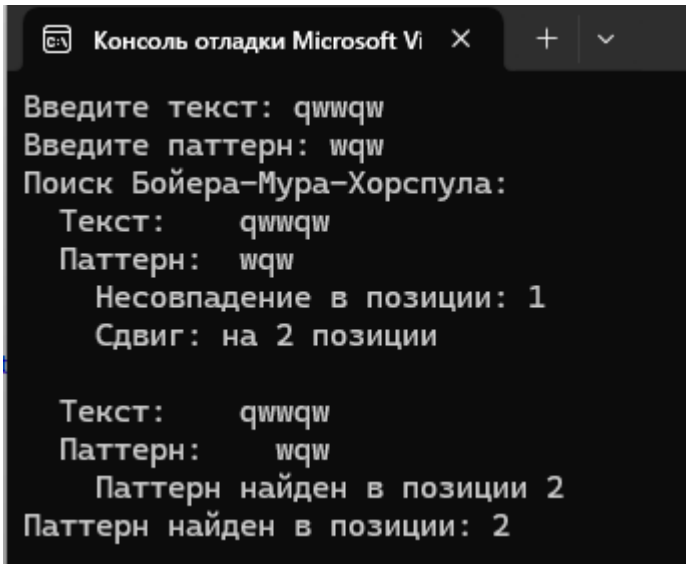
3 БМХ

3.1 Код на C++

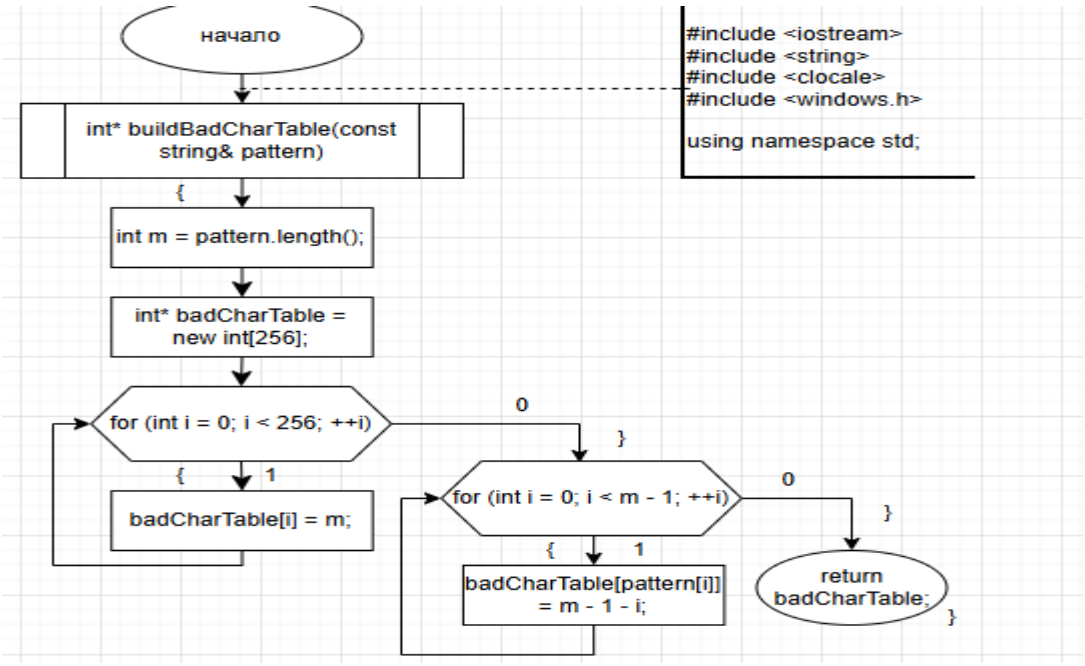
```cpp
#include <iostream>
#include <string>
#include <clocale>
#include <windows.h>

using namespace std;

// Функция для вычисления таблицы плохих символов (Bad Character Heuristic)
int* buildBadCharTable(const string& pattern) {
    int m = pattern.length();
    int* badCharTable = new int[256]; // Инициализируем таблицу размером 256 (предполагаем ASCII символы)

    for (int i = 0; i < 256; ++i) {
        badCharTable[i] = m;
    }

    for (int i = 0; i < m - 1; ++i) {
        badCharTable[pattern[i]] = m - 1 - i;
    }

    return badCharTable;
}

// Функция поиска Бойера-Мура-Хорспула
int boyerMooreHorspool(const string& text, const string& pattern) {
    int n = text.length();
    int m = pattern.length();

    if (m == 0) {
        return 0; // Пустой паттерн всегда находится в начале текста
    }

    if (m > n) {
        return -1; // Паттерн длиннее текста, поэтому не может быть найден
    }

    int* badCharTable = buildBadCharTable(pattern);

    int i = 0; // Индекс для перебора текста

    cout << "Поиск Бойера-Мура-Хорспула:" << endl;

    while (i <= n - m) {
        cout << "  Текст:    " << text << endl;
        cout << "  Паттерн:  ";
        for (int k = 0; k < i; ++k) {
            cout << " ";
        }
        cout << pattern << endl;

        int j = m - 1; // Индекс для перебора паттерна с конца

        while (j >= 0 && pattern[j] == text[i + j]) {
            --j;
        }

        if (j < 0) {
            cout << "    Паттерн найден в позиции " << i << endl;
            delete[] badCharTable; // Освобождаем память
            return i; // Паттерн найден
        }
        else {
            cout << "    Несовпадение в позиции: " << i + j << endl;
            cout << "    Сдвиг: на " << badCharTable[text[i + m - 1]] << " позиции" << endl;
            i += badCharTable[text[i + m - 1]]; // Сдвиг на основе таблицы плохих символов
        }
        cout << endl;
    }
```

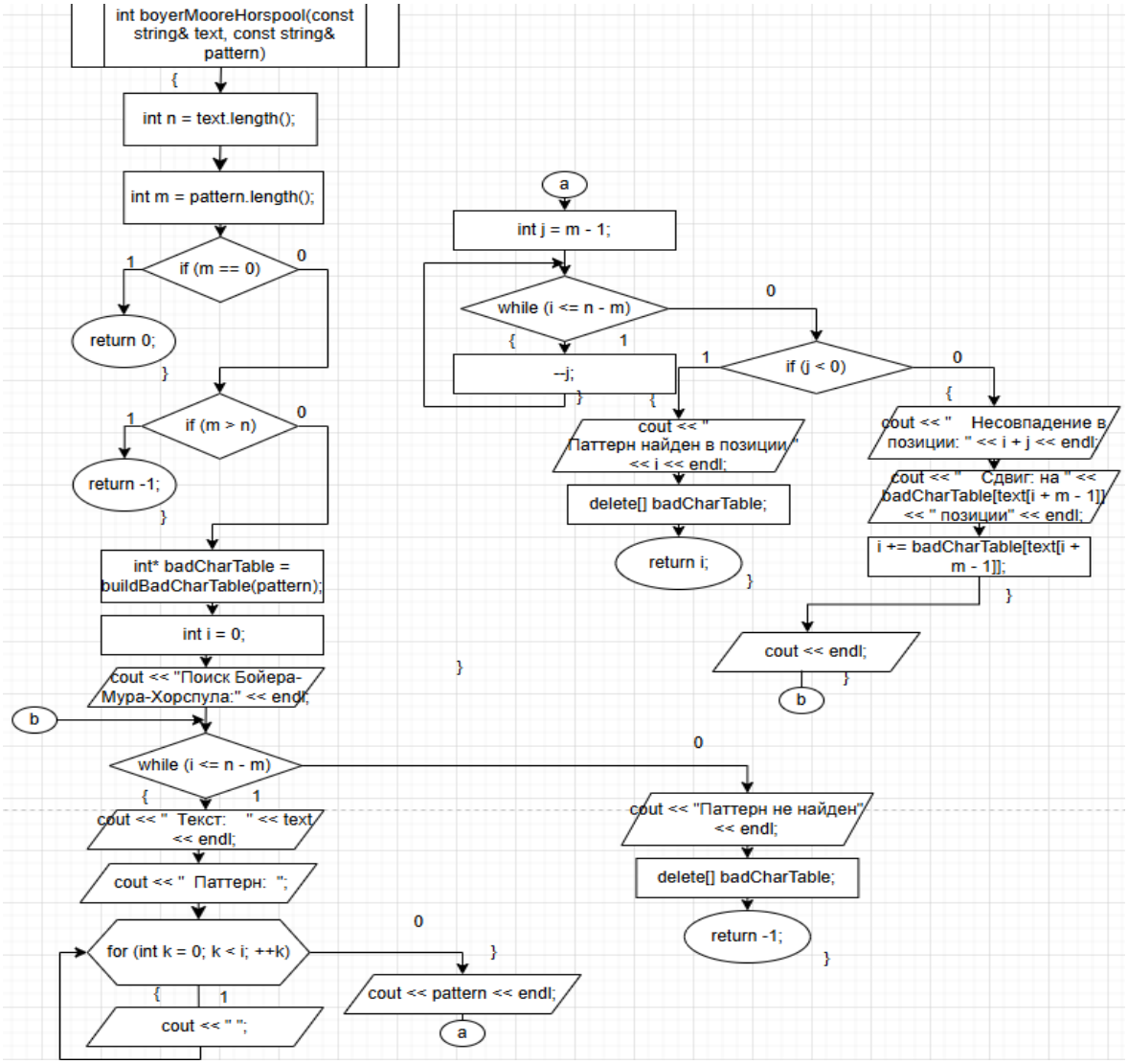
```
69
70          cout << "Паттерн не найден" << endl;
71          delete[] badCharTable; // Освобождаем память
72          return -1; // Паттерн не найден
73      }
74
75      int main() {
76          setlocale(LC_ALL, "ru");
77          SetConsoleCP(1251);
78          SetConsoleOutputCP(1251);
79          string text, pattern;
80
81          cout << "Введите текст: ";
82          getline(cin, text);
83
84          cout << "Введите паттерн: ";
85          getl
                std::ostream std::cout
                Поиск в Интернете
86
87          int                eHorspool(text, pattern);
88
89          if (position != -1) {
90              cout << "Паттерн найден в позиции: " << position << endl;
91          }
92          else {
93              cout << "Паттерн не найден в тексте." << endl;
94          }
95
96          return 0;
97      }
```

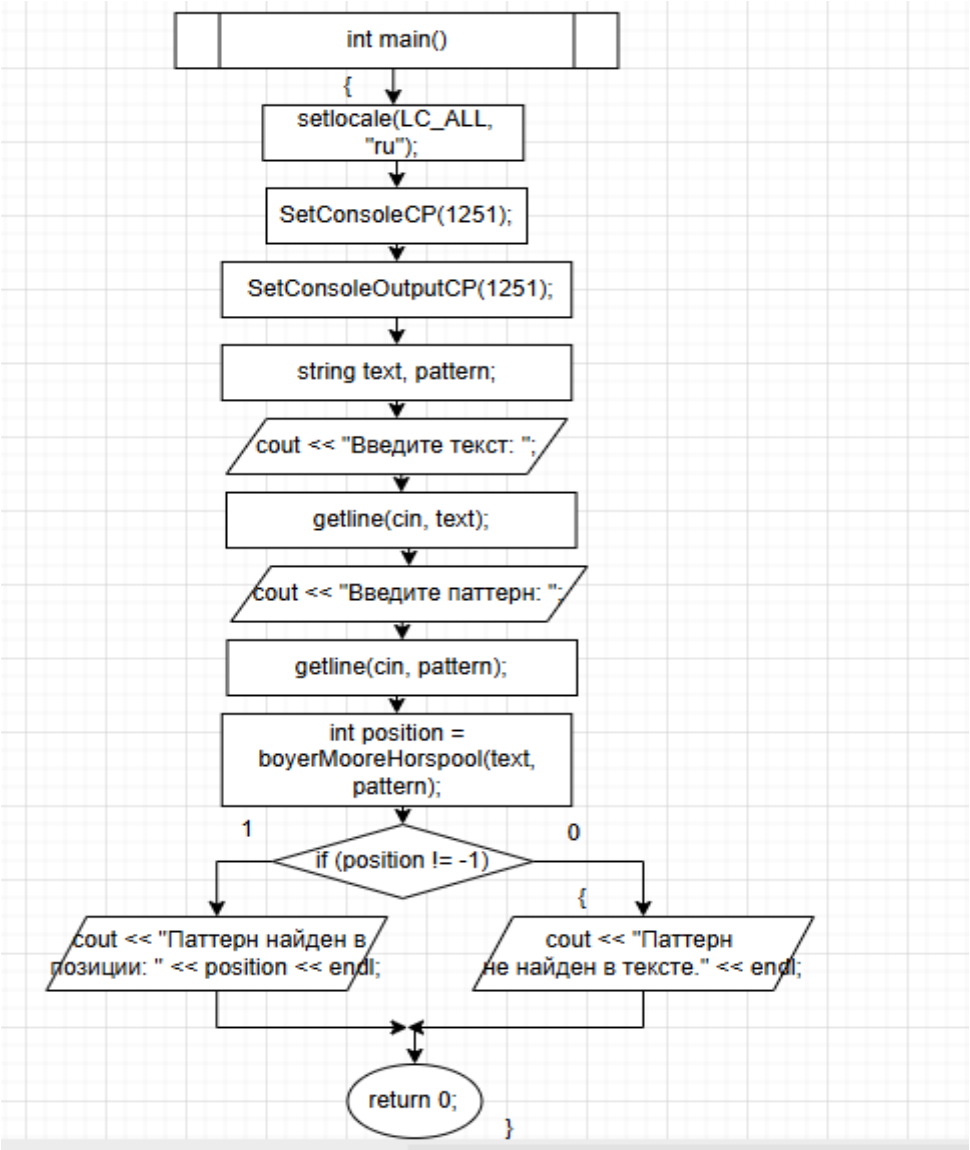
3.2 Результат работы



3.3 Блок-схема


начало
#include <iostream>
#include <string>
#include <clocale>
#include <windows.h>
using namespace std;
int* buildBadCharTable(const string& pattern)
{
int m = pattern.length();
int* badCharTable = new int[256];
for (int i = 0; i < 256; ++i)
badCharTable[i] = m;
for (int i = 0; i < m - 1; ++i)
badCharTable[pattern[i]] = m - 1 - i;
return badCharTable;
}


int boyerMooreHorspool(const string& text, const string& pattern)
int n = text.length();
int m = pattern.length();
if (m == 0)
return 0;
if (m > n)
return -1;
int* badCharTable = buildBadCharTable(pattern);
int i = 0;
cout << "Поиск Бойера-Мура-Хорспула:" << endl;
while (i <= n - m)
cout << " Текст: " << text << endl;
cout << " Паттерн: ";
for (int k = 0; k < i; ++k)
cout << " ";
cout << pattern << endl;
int j = m - 1;
while (i <= n - m)
--j;
if (j < 0)
cout << " Паттерн найден в позиции " << i << endl;
delete[] badCharTable;
return i;
cout << " Несовпадение в позиции: " << i + j << endl;
cout << " Сдвиг: на " << badCharTable[text[i + m - 1]] << " позиции" << endl;
i += badCharTable[text[i + m - 1]];
cout << endl;
cout << "Паттерн не найден" << endl;
delete[] badCharTable;
return -1;



4 Ссылка на github

ссылка на github - https://github.com/MAKSPOWERO/mas1