Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ПНИПУ)

Электротехнический факультет

Дисциплина: основы алгоритмизации и программирования, 2 семестр

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Тема: «Типы поисков в С++»

Выполнил

Студент РИС-24-1б

Конькова С. С.

Проверил

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2025

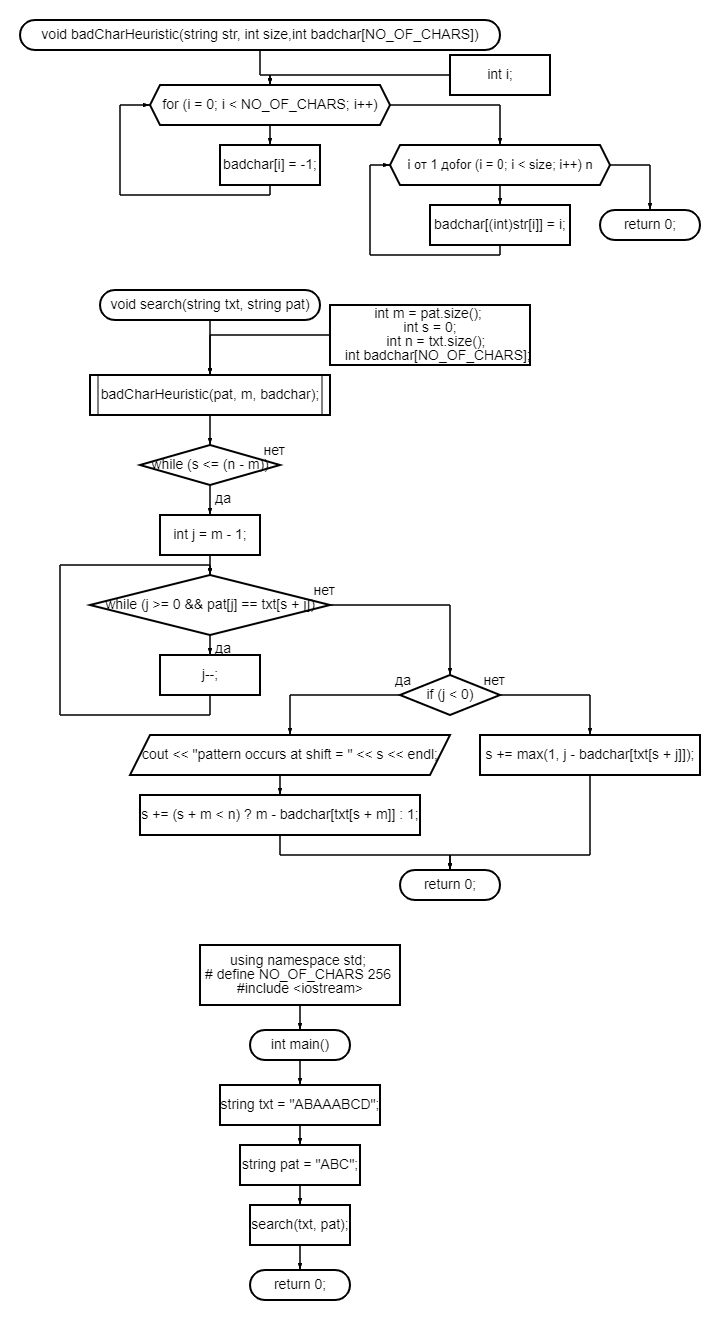
**Анализ:**

Кнутта-Мориса-Пратта:

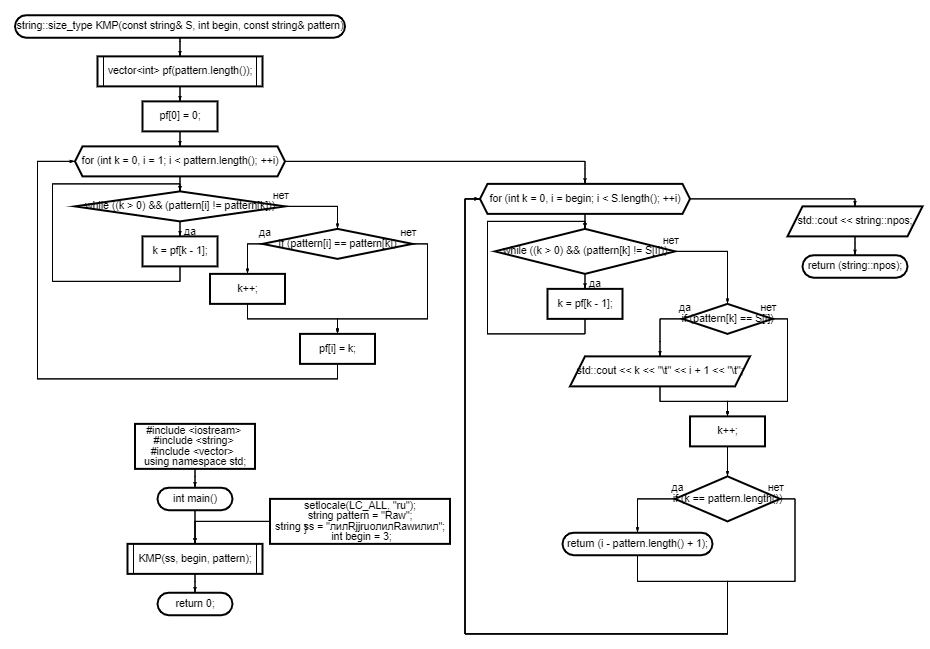
1. Строится массив, который содержит длины наибольших собственных префиксов, одновременно являющихся суффиксами для каждого префикса подстроки. Это позволяет избежать лишних сравнений при несовпадениях.
2. Сравнивается подстрока с текстом посимвольно.
3. При несовпадении используется массив lps для определения, на сколько символов можно сдвинуть подстроку без потери возможных совпадений.

Бойера-Мура:

1. Создаются две эвристики для сдвига
2. Если символ текста не совпадает с символом подстроки, подстрока сдвигается так, чтобы последнее вхождение этого символа в подстроке совпало с текущей позицией в тексте.
3. Если найдено частичное совпадение суффикса подстроки, сдвиг выполняется так, чтобы следующий возможный суффикс совпал.
4. Сравнение символов подстроки с текстом выполняется справа налево.
5. При несовпадении выбирается максимальный сдвиг из двух правил.
6. Поиск Бойера-Мура



1. Поиск Кнутта-Мориса-Пратта



**Коды программ**

1. Поиск Буэра-Мура

using namespace std;

# define NO\_OF\_CHARS 256

#include <iostream>

void badCharHeuristic(string str, int size,

int badchar[NO\_OF\_CHARS])

{

int i;

for (i = 0; i < NO\_OF\_CHARS; i++)

badchar[i] = -1;

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

badchar[(int)str[i]] = i;

}

void search(string txt, string pat)

{

int m = pat.size();

int n = txt.size();

int badchar[NO\_OF\_CHARS];

badCharHeuristic(pat, m, badchar);

int s = 0;

while (s <= (n - m))

{

int j = m - 1;

while (j >= 0 && pat[j] == txt[s + j])

j--;

if (j < 0)

{

cout << "pattern occurs at shift = " << s << endl;

s += (s + m < n) ? m - badchar[txt[s + m]] : 1;

}

else

s += max(1, j - badchar[txt[s + j]]);

}

}

int main()

{

string txt = "ABAAABCD";

string pat = "ABC";

search(txt, pat);

return 0;

}

1. Поиск Кнутта-Мориса-Пратта

#include <iostream>

#include <string>

#include <vector>

using namespace std;

string::size\_type KMP(const string& S, int begin, const string& pattern)

{

vector<int> pf(pattern.length());

pf[0] = 0;

for (int k = 0, i = 1; i < pattern.length(); ++i)

{

while ((k > 0) && (pattern[i] != pattern[k]))

k = pf[k - 1];

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

k++;

pf[i] = k;

}

for (int k = 0, i = begin; i < S.length(); ++i)

{

while ((k > 0) && (pattern[k] != S[i]))

k = pf[k - 1];

if (pattern[k] == S[i])

std::cout << k << "\t" << i + 1 << "\t";

k++;

if (k == pattern.length())

return (i - pattern.length() + 1);

}

return (string::npos);

std::cout << string::npos;

}

int main()

{

setlocale(LC\_ALL, "ru");

string pattern = "Rаw";

string ss = "лилRjjruoлилRawилил";

int begin = 3;

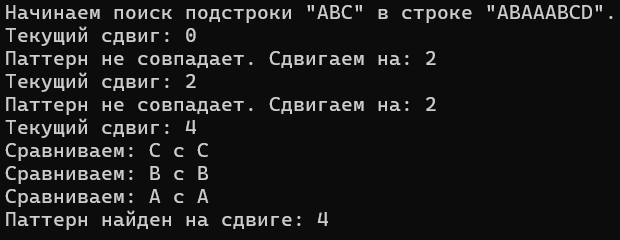
KMP(ss, begin, pattern);

return 0;

}

**Результаты работы**

1. Поиск Бойера-Мура



1. Поиск Кнутта-Мориса-Пратта

