Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

ПНИПУ

Лабораторная работа  
«Методы Поиска. Сложные поиски»

Выполнил:   
студент группы РИС-23-1б   
Мокрушин Никита Дмитриевич

Проверила:   
доцент кафедры ИТАС   
О.А. Полякова

Пермь 2024 г.

**Метод Кнута-Морриса-Пратта**

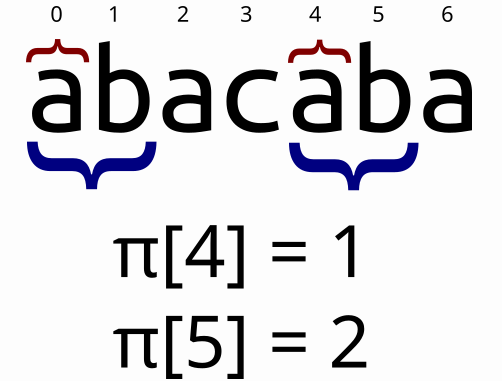
Постановка задачи:

Дана строка и подстрока, которую нужно найти. Выполнить задание в соответствии со своим вариантом, используя указанный метод поиска. Значения строки и подстроки любые (если иного не указано по заданию).

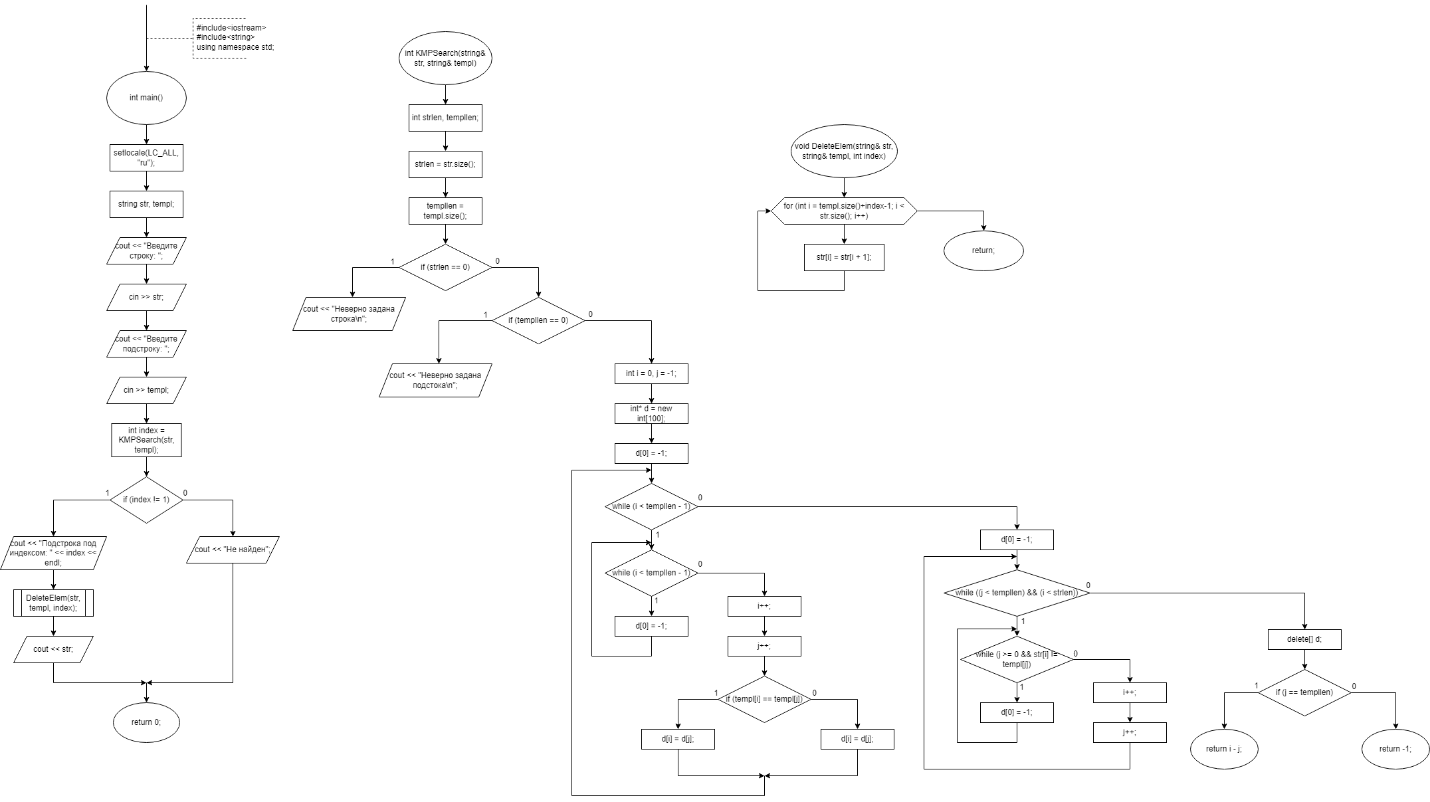
Найти подстроку strl, удалить её последний символ.

Анализ задачи:

1. Поиск Кнута — Морриса — Пратта — эффективный алгоритм, осуществляющий поиск подстроки в строке. Время работы алгоритма линейно зависит от объёма входных данных, то есть разработать асимптотически более эффективный алгоритм невозможно.
2. Префикс-функция от строки s равна массиву π, где π[i] обозначает длину максимального префикса строки s[0..i], совпадающего с её суффиксом. Тривиальные случаи (префикс равен суффиксу и равен всей строке) не учитываются.



На изображении обозначены равные подстроки, длина которых равна значению префикс-функции в данной позиции. Префикс-функция от всей строки “abacaba” равна {0,0,1,0,1,2,3}. π[0]=π[1]=0 так как строки “a” и “ab” являются тривиальными, и поэтому не учитываются.

Блок-схема:

Код программы:

#include<iostream>

#include<string>

using namespace std;

int KMPSearch(string& str, string& templ)

{

int strlen, templlen;

strlen = str.size();

templlen = templ.size();

if (strlen == 0)

cout << "Неверно задана строка\n";

else if (templlen == 0)

cout << "Неверно задана подстока\n";

else

{

int i = 0, j = -1;

int\* d = new int[100];

d[0] = -1;

while (i < templlen - 1)

{

while (j >= 0 && templ[i] != templ[j])

j = d[j];

i++;

j++;

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

d[i] = d[j];

else

d[i] = j;

}

i = 0, j = 0;

while ((j < templlen) && (i < strlen))

{

while (j >= 0 && str[i] != templ[j])

j = d[j];

i++;

j++;

}

delete[] d;

if (j == templlen)

return i - j;

else

return -1;

}

}

void DeleteElem(string& str, string& templ, int index)

{

for (int i = templ.size()+index-1; i < str.size(); i++)

str[i] = str[i + 1];

}

int main()

{

setlocale(LC\_ALL, "ru");

string str, templ;

cout << "Введите строку: ";

cin >> str;

cout << "Введите подстроку: ";

cin >> templ;

int index = KMPSearch(str, templ);

if (index != 1)

{

cout << "Подстрока под индексом: " << index << endl;

DeleteElem(str, templ, index);

cout << str;

}

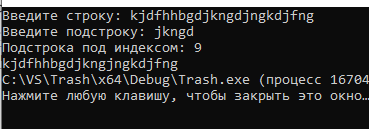
else

cout << "Не найден";

return 0;

}

**Скриншоты результатов:**



Анализ результатов:

Программа работает правильно.

**Метод Бойера-Мура**

Постановка задачи:

Дана строка и подстрока, которую нужно найти. Выполнить задание в соответствии со своим вариантом, используя указанный метод поиска. Значения строки и подстроки любые (если иного не указано по заданию).

Найти и удалить первое вхождение подстроки strl

Анализ задачи:

1. Алгоритм поиска строки Бойера — Мура — алгоритм общего назначения, предназначенный для поиска подстроки в строке.
2. Преимущество этого алгоритма в том, что ценой некоторого количества предварительных вычислений над шаблоном (но не над строкой, в которой ведётся поиск), шаблон сравнивается с исходным текстом не во всех позициях — часть проверок пропускается как заведомо не дающая результата.
3. Совмещается начало текста (строки) и шаблона, проверка начинается с последнего символа шаблона. Если символы совпадают, производится сравнение предпоследнего символа шаблона и т. д. Если все символы шаблона совпали с наложенными символами строки, значит, подстрока найдена, и выполняется поиск следующего вхождения подстроки. Если же какой-то символ шаблона не совпадает с соответствующим символом строки, шаблон сдвигается на несколько символов вправо, и проверка снова начинается с последнего символа.

Блок-схема:

Код программы:

#include<iostream>

#include<string>

using namespace std;

int main()

{

setlocale(LC\_ALL, "ru");

bool flag = false;

string c;

cout << "Введите образ" << endl;

cin >> c;

int k = c.length();

int\* arr = new int[k + 1];

int\* mas = new int[k + 1];

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

mas[i] = k - i - 1;

for (int i = k - 2; i >= 0; i--)

{

int count = 0;

for (int j = 0; j < k - 1; j++)

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

{

count += 1;

arr[i] = mas[j];

}

if (count == 1)

arr[i] = k - i - 1;

}

int templ1 = 0;

int temp;

for (int i = 0; i < k - 1; i++)

if (c[k - 1] == c[i])

{

templ1 += 1;

arr[k - 1] = mas[i];

}

if (templ1 == 0)

arr[k - 1] = k;

arr[k] = k;

string s;

cout << "Введите строку" << endl;

cin.ignore(32767, '\n');

getline(cin, s);

int templ2;

int j = 0;

for (int i = c.length() - 1; i < s.length() && flag != true; j++)

{

if (s[i] == c[c.length() - 1])

{

int count1 = 0;

for (int l = 1; l < c.length(); l++)

if (s[i - l] == c[c.length() - l])

count1 += 1;

if (count1 + c.length() == c.length())

{

flag = true;

templ2 = i;

}

}

bool flag1 = false;

for (int b = 0; b < c.length() && flag1 == false; b++)

if (s[i] == c[b])

{

i += arr[b];

flag1 = true;

}

if (flag1 == false)

i += c.length();

}

if (flag == true)

{

for (int i = 0; i < templ2 - (c.length() - 1); i++)

cout << s[i];

for (int i = templ2 + 1; i < s.length(); i++)

cout << s[i];

}

else

for (int i = 0; i < s.length(); i++)

cout << s[i];

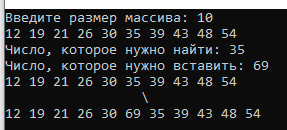
delete[] mas;

delete[] arr;

return 0;

}

Скриншоты результатов:



Анализ результатов:

Программа работает правильно. Выводится массив с добавленным элементом.