Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ**

Задание 1 - Поиск образа в строке

тема

Преподаватель \_\_\_\_\_\_\_\_\_ Р.Ю. Царев

подпись, дата инициалы, фамилия

Студент КИ18-17/1б 031831229 \_\_\_\_\_\_\_\_\_ В.А. Прекель

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2020

Содержание

[Содержание 2](#_Toc34938806)

[1 Цель работы с постановкой задачи 3](#_Toc34938807)

[1.1 Цель работы 3](#_Toc34938808)

[1.2 Задача работы 3](#_Toc34938809)

[2 Описание реализованного алгоритма 3](#_Toc34938810)

[3 Описание программы (листинги кода) 4](#_Toc34938811)

[3.1 Lab\_01/Lab\_01\_Console/main.cpp 4](#_Toc34938812)

[3.2 Lab\_01/Lab\_01\_Lib/BoyerMoore.h 5](#_Toc34938813)

[3.3 Lab\_01/Lab\_01\_Lib/BoyerMoore.cpp 5](#_Toc34938814)

[3.4 Lab\_01/Lab\_01\_LibTests/BoyerMooreTests.cpp 8](#_Toc34938815)

[4 Результаты работы программы 9](#_Toc34938816)

# Цель работы с постановкой задачи

## Цель работы

Поиск образа в строке.

## Задача работы

Написать программу поиска образа в строке по методу Кнута, Морриса и Пратта либо Боуера и Мура (по выбору студента). Предусмотреть возможность существования в образе пробела. Ввести опцию чувствительности / нечувствительности к регистру.

Требования к выполнению лабораторной работы:

1. Строгое соответствие программы и результатов ее работы с полученным заданием.
2. Самостоятельные тестирование и отладка программы.
3. Устойчивость работы программы при любых воздействиях, задаваемых пользователем через интерфейс программы.
4. Предоставление демонстрационного примера и исходного текста программы для защиты.
5. Предоставление отчета по практическому заданию, содержащего описание реализованного алгоритма, программы, результатов работы программы (отчет необходимо загрузить на сайт курса).

# Описание реализованного алгоритма

Реализован алгоритм Боуера-Мура.

# Описание программы (листинги кода)

## Lab\_01/Lab\_01\_Console/main.cpp

#include **<cstdio>**#include **<iostream>**#include **<algorithm>**#ifdef **\_MSC\_VER**#include **<Windows.h>**#elif \_WIN32  
#include **<windows.h>**#endif  
  
#include **"BoyerMoore.h"  
  
int** main(**int** argc, **char**\*\* argv)  
{  
#ifdef **\_WIN32** SetConsoleOutputCP(CP\_UTF8);  
 SetConsoleCP(CP\_UTF8);  
#endif  
  
 std::cout << **"Введите строку (в которой проводится поиск): "**;  
 std::string y;  
 std::getline(std::cin, y);  
  
 std::cout << **"Введите образ (который ищется в строке): "**;  
 std::string x;  
 std::getline(std::cin, x);  
  
 std::cout << **"Чувствительно к регистру? [Y/n]: "**;  
 std::string t;  
 std::getline(std::cin, t);  
  
 **auto** bm = BoyerMoore(y, x, t[0] == **'N'** || t[0] == **'n'**);  
 bm.Calculate();  
 **auto** ans = bm.GetAnswer();  
  
 std::cout << **"Индексы образа в строке: "**;  
 **for** (**auto** i : \*ans)  
 {  
 std::cout << i << ((i == ans->back()) ? **""** : **", "**);  
 }  
 std::cout << std::endl;  
  
 std::cout << **"Строка с помеченными образами: "**;  
 **for** (**auto** i = 0, j = -1; i < y.size(); i++)  
 {  
 **if** (std::count(ans->begin(), ans->end(), i))  
 {  
 std::cout << **"["**;  
 j = x.size() - 1;  
 }  
 std::cout << y[i];  
 **if** (j-- == 0)  
 {  
 std::cout << **"]"**;  
 }  
 }  
 std::cout << std::endl;  
  
 **return** 0;  
}

## Lab\_01/Lab\_01\_Lib/BoyerMoore.h

#ifndef **BOYERMOORE\_H**#define **BOYERMOORE\_H**#include **<string>**#include **<vector>  
  
class** BoyerMoore  
{  
**private**:  
 std::string y; *// строка* std::string x; *// образец* std::vector<**int**>\* borderArray;  
 std::vector<**int**>\* shiftArray;  
 std::vector<**int**>\* answer;  
  
 **void** FullSuffixMatch();  
  
 **void** PartialSuffixMatch();  
  
 **void** BM();  
  
**public**:  
 BoyerMoore(std::string y, std::string x, **bool** insensitive);  
  
 **void** Calculate();  
  
 std::vector<**int**>\* GetAnswer()  
 {  
 **return** answer;  
 }  
  
 ~BoyerMoore();  
};  
  
#endif *//BOYERMOORE\_H*

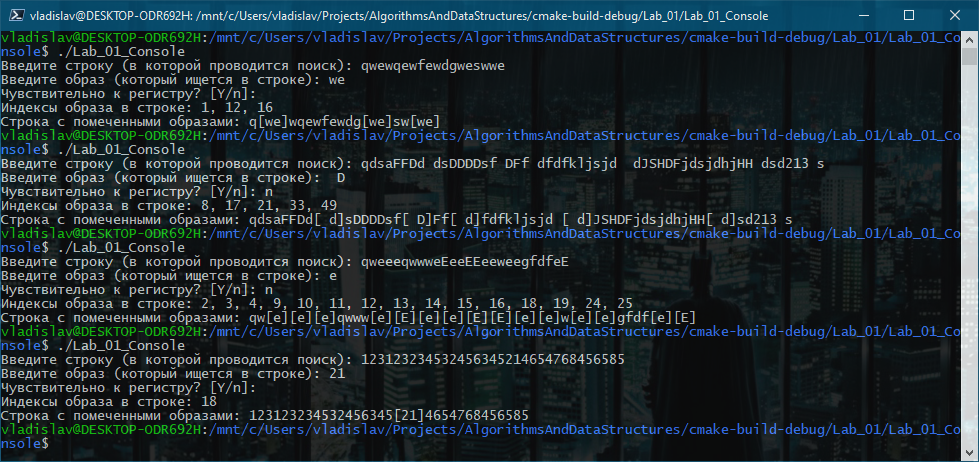
## Lab\_01/Lab\_01\_Lib/BoyerMoore.cpp

#include **<algorithm>**#include **"BoyerMoore.h"  
  
void** BoyerMoore::FullSuffixMatch()  
{  
 **int** n = x.size();  
 **int** i = n;  
 **int** j = n + 1;  
 borderArray->at(i) = j;  
  
 **while** (i > 0)  
 {  
  
 **while** (j <= n && x[i - 1] != x[j - 1])  
 {  
 **if** (shiftArray->at(j) == 0)  
 {  
 shiftArray->at(j) = j - i;  
 }  
 j = borderArray->at(j);  
 }  
 i--;  
 j--;  
 borderArray->at(i) = j;  
 }  
}  
  
**void** BoyerMoore::PartialSuffixMatch()  
{  
 **int** n = x.size();  
 **int** j;  
 j = borderArray->at(0);  
  
 **for** (**int** i = 0; i < n; i++)  
 {  
 **if** (shiftArray->at(i) == 0)  
 {  
 shiftArray->at(i) = j;  
 }  
 **if** (i == j)  
 {  
 j = borderArray->at(j);  
 }  
 }  
}  
  
**void** BoyerMoore::BM()  
{  
 **int** m = x.length();  
 **int** n = y.length();  
  
 **if** (m == 0)  
 {  
 answer->push\_back(-1);  
 **return**;  
 }  
 **if** (m == 1)  
 {  
 **for** (**auto** i = 0; i < n; i++)  
 {  
 **if** (y[i] == x[0])  
 {  
 answer->push\_back(i);  
 }  
 }  
 **return**;  
 }  
  
 borderArray->assign(m + 1, 0);  
 shiftArray->assign(m + 1, 0);  
  
 FullSuffixMatch();  
 PartialSuffixMatch();  
  
 **int** shift = 0;  
  
 **while** (shift <= (n - m))  
 {  
 **int** j = m - 1;  
 **while** (j >= 0 && x[j] == y[shift + j])  
 {  
 j--;  
 }  
  
 **if** (j < 0)  
 {  
 answer->push\_back(shift);  
 shift += shiftArray->at(0);  
 }  
 **else** {  
 shift += shiftArray->at(j + 1);  
 }  
 }  
  
 **if** (answer->empty())  
 {  
 answer->push\_back(-1);  
 }  
}  
  
BoyerMoore::BoyerMoore(std::string y, std::string x, **bool** insensitive)  
{  
 **this**->y = std::move(y);  
 **this**->x = std::move(x);  
  
 **if** (insensitive)  
 {  
 std::transform(**this**->y.begin(), **this**->y.end(), **this**->y.begin(),  
 [](**auto** c)  
 { **return** std::tolower(c); });  
 std::transform(**this**->x.begin(), **this**->x.end(), **this**->x.begin(),  
 [](**auto** c)  
 { **return** std::tolower(c); });  
 }  
  
 borderArray = **new** std::vector<**int**>();  
 shiftArray = **new** std::vector<**int**>();  
 answer = **new** std::vector<**int**>();  
}  
  
**void** BoyerMoore::Calculate()  
{  
 BM();  
}  
  
BoyerMoore::~BoyerMoore()  
{  
 **delete** borderArray;  
 **delete** shiftArray;  
 **delete** answer;  
}

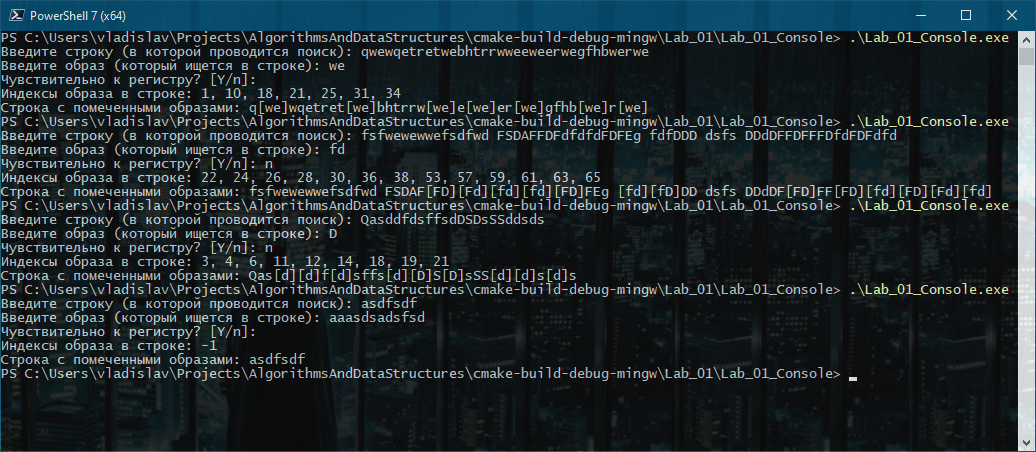
## Lab\_01/Lab\_01\_LibTests/BoyerMooreTests.cpp

#include **<vector>**#include **<gtest/gtest.h>**#include **"BoyerMoore.h"  
  
using namespace** testing;  
  
**TEST**(BoyerMooreTests, BoyerMooreTest1)  
{  
 **auto** bm = **new** BoyerMoore(**"rgsgfdsq125rerdsgs"**, **"r"**, **false**);  
 bm->Calculate();  
 **auto** answer = bm->GetAnswer();  
  
 **ASSERT\_EQ**(answer->size(), 3);  
 **EXPECT\_EQ**(answer->at(0), 0);  
 **EXPECT\_EQ**(answer->at(1), 11);  
 **EXPECT\_EQ**(answer->at(2), 13);  
  
 **delete** bm;  
}  
  
**TEST**(BoyerMooreTests, BoyerMooreTest2)  
{  
 **auto** bm = **new** BoyerMoore(**"123123"**, **"23"**, **false**);  
 bm->Calculate();  
 **auto** answer = bm->GetAnswer();  
  
 **ASSERT\_EQ**(answer->size(), 2);  
 **EXPECT\_EQ**(answer->at(0), 1);  
 **EXPECT\_EQ**(answer->at(1), 4);  
  
 **delete** bm;  
}  
  
**TEST**(BoyerMooreTests, BoyerMooreTest3)  
{  
 **auto** bm = **new** BoyerMoore(**"ABAAABCDBBABCDDEBCABC"**, **"ABC"**, **false**);  
 bm->Calculate();  
 **auto** answer = bm->GetAnswer();  
  
 **ASSERT\_EQ**(answer->size(), 3);  
 **EXPECT\_EQ**(answer->at(0), 4);  
 **EXPECT\_EQ**(answer->at(1), 10);  
 **EXPECT\_EQ**(answer->at(2), 18);  
  
 **delete** bm;  
}  
  
**TEST**(BoyerMooreTests, BoyerMooreTest4)  
{  
 **auto** bm = **new** BoyerMoore(**"GCATCGCAGAGAGTATACAGTACG"**, **"GCAGAGAG"**, **false**);  
 bm->Calculate();  
 **auto** answer = bm->GetAnswer();  
  
 **ASSERT\_EQ**(answer->size(), 1);  
 **EXPECT\_EQ**(answer->at(0), 5);  
  
 **delete** bm;  
}  
  
**TEST**(BoyerMooreTests, BoyerMooreTest5)  
{  
 **auto** bm = BoyerMoore(**"GCATCGCAGAGAGTATACAGTACG"**, **"GcagAGAG"**, **true**);  
 bm.Calculate();  
 **auto** answer = bm.GetAnswer();  
  
 **ASSERT\_EQ**(answer->size(), 1);  
 **EXPECT\_EQ**(answer->at(0), 5);  
}

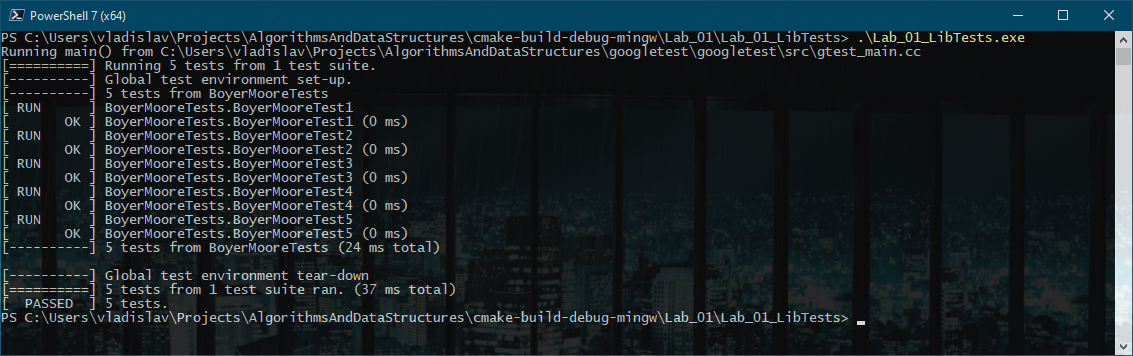
# Результаты работы программы



1. Запуск 1 (WSL Ubuntu 18.04, gcc)



1. Запуск 2 (Windows 10, mingw)



1. Запуск тестов