Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий

институт

Информатики

кафедра

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

Преподаватель \_\_\_\_\_\_\_\_\_\_\_ А.С. Черниговский

подпись, дата инициалы, фамилия

Студент КИ15–17Б, 031510065 \_\_\_\_\_\_\_\_\_\_\_ Т.В. Радионов

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2017

1. **Цель, задачи**

Цель: написать программу поиска образа в строке по алгоритму Бойера – Мура – Хорспула.

Задачи: предусмотреть возможность существования в образе пробела. Ввести опцию чувствительности/нечувствительности к регистру.

1. **Описание алгоритма**

Данный алгоритм имеет похожую идею: как и в КМП образ смещается вдоль строки слева направо, однако проверка совпадения образа с текущей частью строки происходит справа налево.

Для определения смещения перед поиском происходит создание списка shiftList, который формируется с помощью функции FormationOfListOfShifts (). Для этой задачи формируется список-словарь alphabet <int, char>, в которую записывается уникальный символ с уникальным номером. Номер определяется правилом алгоритма для каждого символа. Символы в данном алфавите уникальны (при повторении символа в списке-словаре alphabet <int, char> ничего не вносится и ничего не изменяется).

Соответственно, у последнего символа списка-словаря alphabet <int, char> будет иметь значение в поле int “0”: <0, “d”>, например. Соответственно, изменяем значение данного поля по правилу: если такой символ не встречается в слове-запросе request[], то присваиваем значение равное длине слова-запроса request[], а если есть, то присваиваем значение предпоследнего вхождения символа, равного последнему и перезаписываем значение поля int последнего элемента списка-словаря alphabet <int, char>.

Итак, список смещений готов, далее алгоритм внутри функции SearchCoincidencesWithRequest() выполняет сравнение последнего символа слова-запроса request[] с символом на той же позиции в тексте. Если символы совпадают, то выполняется сравнение следующих символов (и так далее, пока алгоритм не завершит работу). Если же символ не совпадает, то: если сравниваемый символ текста существует в списке-словаре alphabet <int, char>, то «смещаем» слово-запрос request[] на значение у найденного символа из списка-словаря, а если нет, то «смещаем» слово-запрос request[] полностью на его длину и осуществляется рекурсия функции SearchCoincidencesWithRequest() и так до конца.

1. **Код алгоритма**
2. Формирование списка смещений

void FormationOfListOfShifts(string requestStr)

{

shiftList.Clear(); // Очистка списка смещений

alphabet.Clear(); // Очистка алфавита запроса

request = requestStr.ToCharArray();

lengthOfRequest = request.Count();

// Заполнение алфивита и списка смещений в обратном порядке

for (int i = lengthOfRequest - 1, j = 0; i >= 0; i--, j++)

{

// Получаем значение ключа алфавита через поиск первого вхождения символа

int resultKey;

try

{

resultKey = alphabet.First(x => x.Value == request[i]).Key;

}

catch (InvalidOperationException)

{

resultKey = -1;

}

// Если есть символ в алфавите, то добавляем значение в таблицу смещений

if (resultKey != -1)

{

shiftList.Insert(0, resultKey);

}

// Если нет символа в алфавите, то добавляем его в алфавит

else

{

alphabet.Add(j, request[i]);

shiftList.Insert(0, j);

}

}

// Замена ключа "0" в списке смещений у последнего символа запроса

int firstEntryOfIndex = -1;

// Если запрос состоит из одного символа, то присваиваем "1" и завершаем формирование списка смещений

if (lengthOfRequest == 1)

{

shiftList[0] = 1;

}

// Если запрос из более, чем один символ, то

else

{

// Определение индекса первого вхождения символа, схожего с конечным в запросе

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

{

// Если таков символ есть, то записываем его индекс

if (request[i] == request[lengthOfRequest - 1])

{

firstEntryOfIndex = i;

break;

}

}

// Если такой символ существует, то заменяем все ключи в списке смещений этого символа на вычисленное значение

if (firstEntryOfIndex != -1)

{

for (int i = lengthOfRequest - 1; i >= 0; i--)

{

// Если таков символ есть, то записываем его индекс

if (request[i] == request[lengthOfRequest - 1])

{

shiftList[i] = lengthOfRequest - 1 - firstEntryOfIndex;

}

}

}

// Иначе заменяем значение последнего элемента списка смещений на значение длины строки запроса

else

{

shiftList[shiftList.Count - 1] = shiftList.Count;

}

}

}

1. Поиск совпадений в тексте по запросу

void SearchCoincidencesWithRequest(int currentPosition, string text)

{

int shift = lengthOfRequest - 1;

int previousPosition = currentPosition;

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

{

// Если символ текста совпал с текущим символом запроса, то переходим к сравнению следующего символа текста и запроса

if (text.Substring(previousPosition, 1) == request[j].ToString())

{

previousPosition--;

shift = j - 1;

}

// Если есть несоответствие, то сдвигаем каретку на значение из shiftList и выполняем рекурсию

else

{

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

{

// Если сравниваемый символ текста имеется в запросе, то присваиваем n позицию в shiftList

if (text.Substring(previousPosition, 1) == request[k].ToString())

{

shift = k;

break;

}

}

// Если конец текста не достигнут, то продолжаем выполнять алгоритм

currentPosition = currentPosition + shiftList[shift]; // Выполняем перевод каретки

if (currentPosition <= text.Length - 1)

{

SearchCoincidencesWithRequest(currentPosition, text); // Вызываем рекрсию

return;

}

// Иначе заканчиваем алгоритм

else

{

return;

}

}

}

// Если конец текста не достигнут, то продолжаем выполнять алгоритм

currentPosition = currentPosition + shiftList[lengthOfRequest - 1]; // Выполняем перевод каретки по значению смещения последнего символа запроса

if (currentPosition <= text.Length - 1)

{

SearchCoincidencesWithRequest(currentPosition, text); // Вызываем рекурсию

return;

}

// Иначе заканчиваем алгоритм

else

{

return;

}

}

1. **Примеры работы программ**

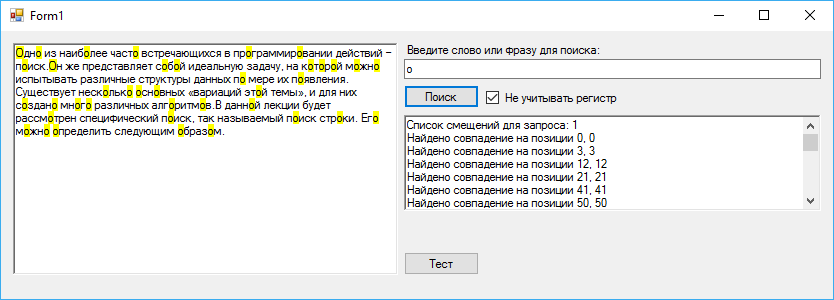


Рисунок 1 – Пример работы программы без учета регистра

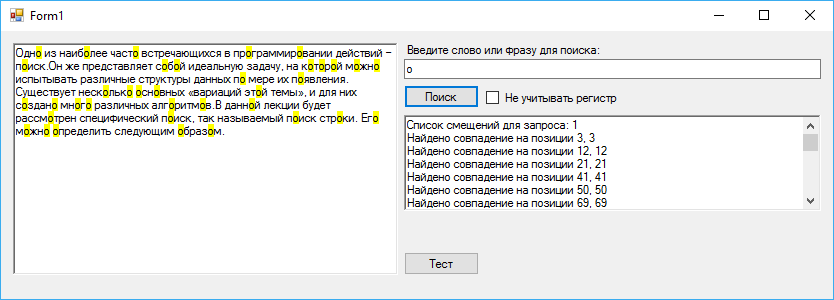
****

Рисунок 2 – Пример работы программы с учетом регистра

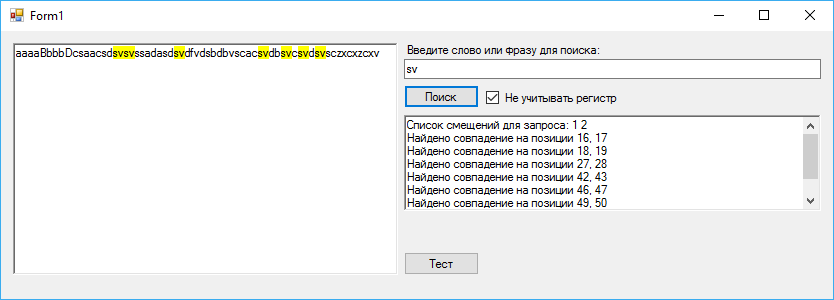
****

Рисунок 3 – Пример работы программы

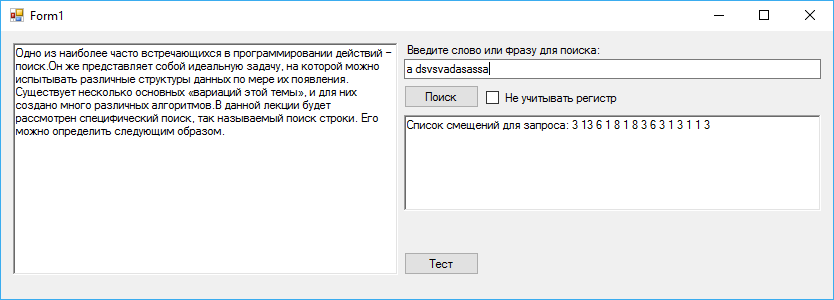


Рисунок 4 – Пример формирования списка смещений

1. **Вывод**

В ходе выполнения лабораторной работы была реализована программа с алгоритмом Бойера – Мура – Хорспула и выполнены все поставленные задачи.