# 2 Разработка программного комплекса по поиску подстроки в строке с использованием алгоритма Бойера-Мура

### 2.1 Листинг программы с коммента1риями

Листинг 1 – Конструктор

namespace WindowsFormsApp4

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();// инициализация компонентов формы

}

static int NO\_OF\_CHARS = 256;

//Функция полезности для получения максимум двух целых

static int max(int a, int b) { return (a > b) ? a : b; }

Листинг 2 – Создание функции для предварительной обработки

//Функция предварительной обработки для Бойера Мура

//плохая эвристика

static void badCharHeuristic(char[] str, int size, int[] badchar)

{

int i;

// Инициализировать все вхождения как -1

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

badchar[i] = -1;

// Заполнение фактического значения последнего вхождения

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

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

}

Листинг 3 – создание функции поиска слова(подстроки) в тексте по заданным параметрам, используя эвристический характер алгоритма

/\* Функция поиска по шаблону, использующая плохой

эвристический характер алгоритма Бойера Мура \*/

static void search(char[] txt, char[] pat)

{

int m = pat.Length;

int n = txt.Length;

if (m == 0)

{

MessageBox.Show("Нет подстроки, которую нужно искать в тексте!");

}

if (n == 0)

{

MessageBox.Show("Нет текста,в котором нужно искать слово!");

}

int[] badchar = new int[NO\_OF\_CHARS];

/\* Заполнение массива плохих символов, вызвав

функцию предварительной обработки badCharHeuristic ()

для данного шаблона \*/

badCharHeuristic(pat, m, badchar);

int s = 0; // s - сдвиг шаблона с

while (s <= (n - m))

{

int j = m - 1;

/\* Продолжает уменьшать индекс J шаблона, в то время как символы шаблона и текста совпадают при этом сдвиге \*/

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

j--;

/\* Если паттерн присутствует при текущем сдвиге, то индекс j станет -1 после описанного выше цикла \*/

if (j < 0)

{

MessageBox.Show("Совпадение подстроки в строке(искомое слово) нашлось в строке текстового файла при сдвиге на " + s);

/\* Сдвиг шаблона так, чтобы следующий символ в тексте совпадал с последним вхождением его в шаблон.

Условие s + m < n необходимо для случая, когда шаблон возникает в конце текста \*/

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

}

else

/\* Сдвиньте шаблон так, чтобы плохой символ в тексте совпадал с последним его появлением в шаблоне.

Функция max используется, чтобы убедиться, что мы получаем положительный сдвиг.

Мы можем получить отрицательный сдвиг, если последнее появление плохого символа в шаблоне находится справа от текущего символа.\*/

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

}

}  
Были добавлены диалоговые окна – MessageBox.

Листинг 4 – Создание кнопки поиска слова(подстроки) в тексте

private void button1\_Click(object sender, EventArgs e)

{

string stroka = richTextBox1.Text;

string slovo = textBox1.Text;

char[] txt = stroka.ToCharArray();

char[] pat = slovo.ToCharArray();

search(txt, pat);

string find = textBox1.Text;

if (richTextBox1.Text.Contains(find))

{

int i = 0;

while (i <= richTextBox1.Text.Length - find.Length)

{

i = richTextBox1.Text.IndexOf(find, i);

if (i < 0) break;

richTextBox1.SelectionStart = i;

richTextBox1.SelectionLength = find.Length;

richTextBox1.SelectionColor = Color.Blue;

i += find.Length;

}

}

else

{

// Окно сообщения будет отображаться, если введенное слово не найдено

MessageBox.Show("Не найдено ни одного соответствия результатов");

}

}

Листинг 5 – Создание кнопки «Открыть файл» в ToolStripMenu

private void открытьToolStripMenuItem1\_Click(object sender, EventArgs e)

{

using (OpenFileDialog dialog = new OpenFileDialog())

{

dialog.Filter = "Текстовые файлы|\*.txt";

if (dialog.ShowDialog() == DialogResult.OK)

{

richTextBox1.Text = File.ReadAllText(dialog.FileName);

}

}

}

С помощью OpenFileDialog можно выбрать любой текстовый файл на компьютере.

Листинг 6 – Создание кнопки «Выход» в ToolStripMenu

private void выходToolStripMenuItem1\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show("Вы хотите выйти?", "Кнопка выхода", MessageBoxButtons.YesNo,MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

Close();

}

}

С помощью данной кнопки полностью закрывается форма.

Листинг 7 – Создание кнопки «Справка» в ToolStripMenu

private void справкаToolStripMenuItem1\_Click(object sender, EventArgs e)

{

MessageBox.Show("Для того чтобы начать работу с программой, вам нужно ознакомиться с интерфейсом программы и отрыть исходный текстовый файл," +

"написанный на английском языке.", "Справка", MessageBoxButtons.OK);

}

Нажимая на данную кнопку появляется информация по работе с программой.

### 2.2 Интерфейс программы

На главном экране программы находятся поля для ввода слова, которое нужно найти, поле для отображения исходного текста, кнопка поиска текста, меню для выбора текстового файла, а также для выхода из программы или получения информации о работе программы.

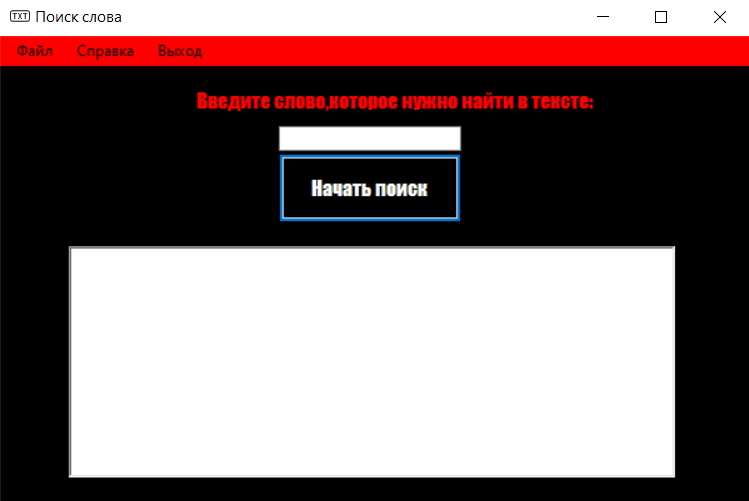


Рисунок - Главный экран

Когда программа запустилась нужно нажать кнопку «Справка» в левом верхнем углу для того чтобы ознакомиться с программы или кнопку «Файл» в том же углу.

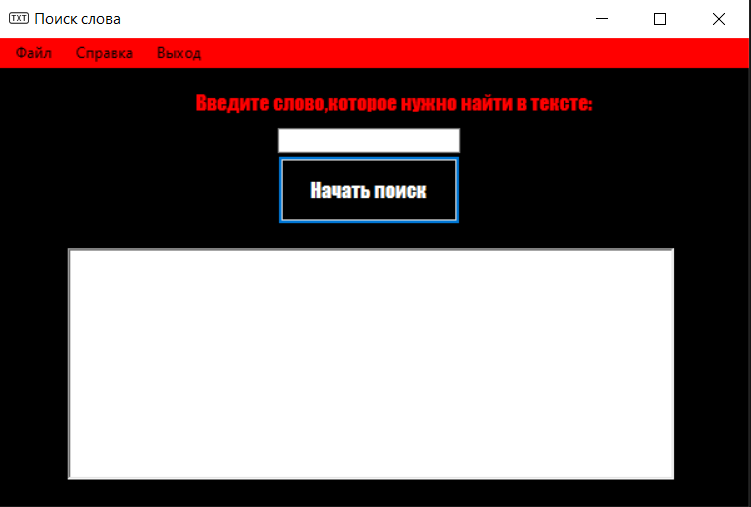


Рисунок - Начало работы

После этого выбираем текстовый файл на компьютере, нажимаем кнопку начать поиск, и если слово есть в тексте, оно выделяется синим цветом.

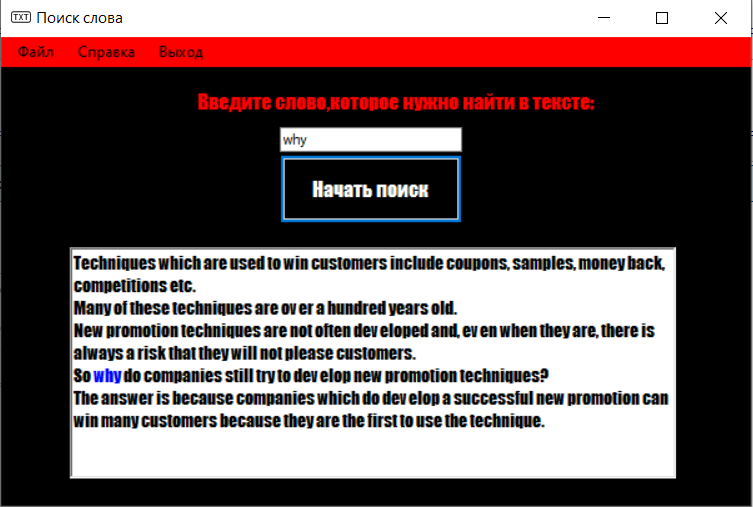


Рисунок - Результат работы программы

### 2.3 Тестирование программы

Тест №1.

Случай, когда слово(подстрока) повторяется в тексте:

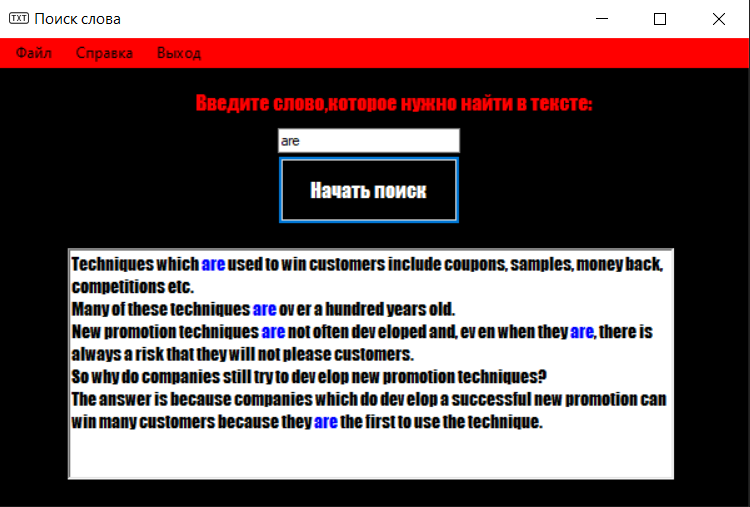


Рисунок - Случай, когда в тексте найдено несколько слов

Тест №2.

Случай, когда в тексте(строке) найдено только одно слово:

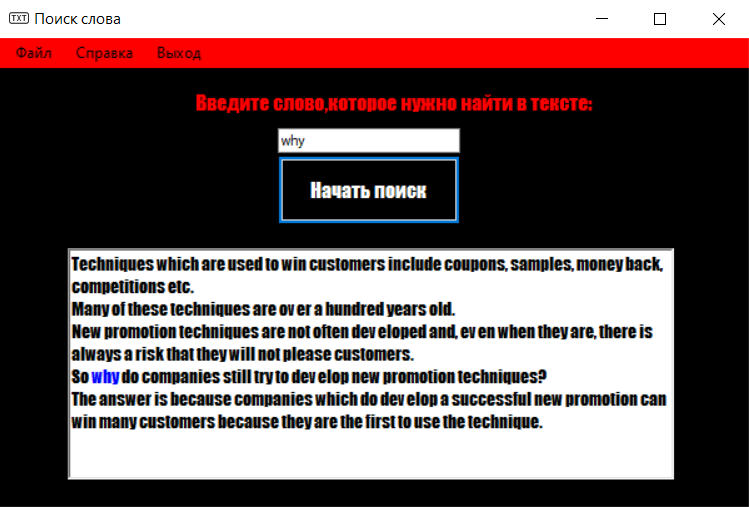


Рисунок - Случай, когда найдено одно слово

Тест №3.

Случай, когда в тексте(строке) не найдено слова(подстроки).

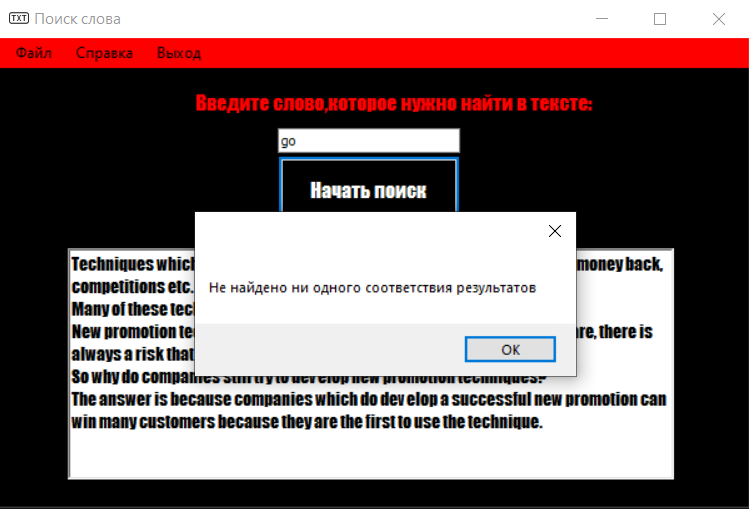


Рисунок - случай, когда не найдено ни одного слова

# Приложение А

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace WindowsFormsApp4

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();// инициализация компонентов формы

}

static int NO\_OF\_CHARS = 256;

//Функция полезности для получения максимум двух целых

static int max(int a, int b) { return (a > b) ? a : b; }

//Функция предварительной обработки для Бойера Мура

//плохая эвристика

static void badCharHeuristic(char[] str, int size, int[] badchar)

{

int i;

// Инициализировать все вхождения как -1

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

badchar[i] = -1;

// Заполнение фактического значения последнего вхождения

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

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

}

/\* Функция поиска по шаблону, использующая плохой

эвристический характер алгоритма Бойера Мура \*/

static void search(char[] txt, char[] pat)

{

int m = pat.Length;

int n = txt.Length;

if (m == 0)

{

MessageBox.Show("Нет подстроки, которую нужно искать в тексте!");

}

if (n == 0)

{

MessageBox.Show("Нет текста,в котором нужно искать слово!");

}

int[] badchar = new int[NO\_OF\_CHARS];

/\* Заполнение массива плохих символов, вызвав

функцию предварительной обработки badCharHeuristic ()

для данного шаблона \*/

badCharHeuristic(pat, m, badchar);

int s = 0; // s - сдвиг шаблона с

while (s <= (n - m))

{

int j = m - 1;

/\* Продолжает уменьшать индекс J шаблона, в то время как символы шаблона и текста совпадают при этом сдвиге \*/

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

j--;

/\* Если паттерн присутствует при текущем сдвиге, то индекс j станет -1 после описанного выше цикла \*/

if (j < 0)

{

MessageBox.Show("Совпадение подстроки в строке(искомое слово) нашлось в строке текстового файла при сдвиге на " + s);

/\* Сдвиг шаблона так, чтобы следующий символ в тексте совпадал с последним вхождением его в шаблон.

Условие s + m < n необходимо для случая, когда шаблон возникает в конце текста \*/

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

}

else

/\* Сдвиньте шаблон так, чтобы плохой символ в тексте совпадал с последним его появлением в шаблоне.

Функция max используется, чтобы убедиться, что мы получаем положительный сдвиг.

Мы можем получить отрицательный сдвиг, если последнее появление плохого символа в шаблоне находится справа от текущего символа.\*/

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

}

}

private void button1\_Click(object sender, EventArgs e)

{

string stroka = richTextBox1.Text;

string slovo = textBox1.Text;

char[] txt = stroka.ToCharArray();

char[] pat = slovo.ToCharArray();

search(txt, pat);

string find = textBox1.Text;

if (richTextBox1.Text.Contains(find))

{

int i = 0;

while (i <= richTextBox1.Text.Length - find.Length)

{

i = richTextBox1.Text.IndexOf(find, i);

if (i < 0) break;

richTextBox1.SelectionStart = i;

richTextBox1.SelectionLength = find.Length;

richTextBox1.SelectionColor = Color.Blue;

i += find.Length;

}

}

else

{

// Окно сообщения будет отображаться, если введенное слово не найдено

MessageBox.Show("Не найдено ни одного соответствия результатов");

}

}

private void открытьToolStripMenuItem1\_Click(object sender, EventArgs e)

{

using (OpenFileDialog dialog = new OpenFileDialog())

{

dialog.Filter = "Текстовые файлы|\*.txt";

if (dialog.ShowDialog() == DialogResult.OK)

{

richTextBox1.Text = File.ReadAllText(dialog.FileName);

}

}

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void richTextBox1\_TextChanged(object sender, EventArgs e)

{

}

private void выходToolStripMenuItem1\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show("Вы хотите выйти?", "Кнопка выхода", MessageBoxButtons.YesNo,MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

Close();

}

}

private void справкаToolStripMenuItem1\_Click(object sender, EventArgs e)

{

MessageBox.Show("Для того чтобы начать работу с программой, вам нужно ознакомиться с интерфейсом программы и отрыть исходный текстовый файл," +

"написанный на английском языке.", "Справка", MessageBoxButtons.OK);

}

}

}