МИНОБРНАУКИ РОССИИ

ФГБОУ ВО Череповецкий государственный университет

Институт информационных технологий

Кафедра: МПО ЭВМ

Дисциплина: Структуры и алгоритмы обработки данных

ЛАБОРАТОРНАЯ РАБОТА № 4

«Дерево поиска. Частотный словарь.»

Выполнила:

студентка гр. 1ПИб-01-11оп Пикалова А.С.

Проверил:

Пышницкий Константин Михайлович

Череповец, 2023 г.

**Задание:**

Составить программу, реализующую частотный словарь текста.

Программа должна производить следующие действия;

1. Выделять в выбранном текстовом файле отдельные слова.

2. Помещать слова в дерево поиска.

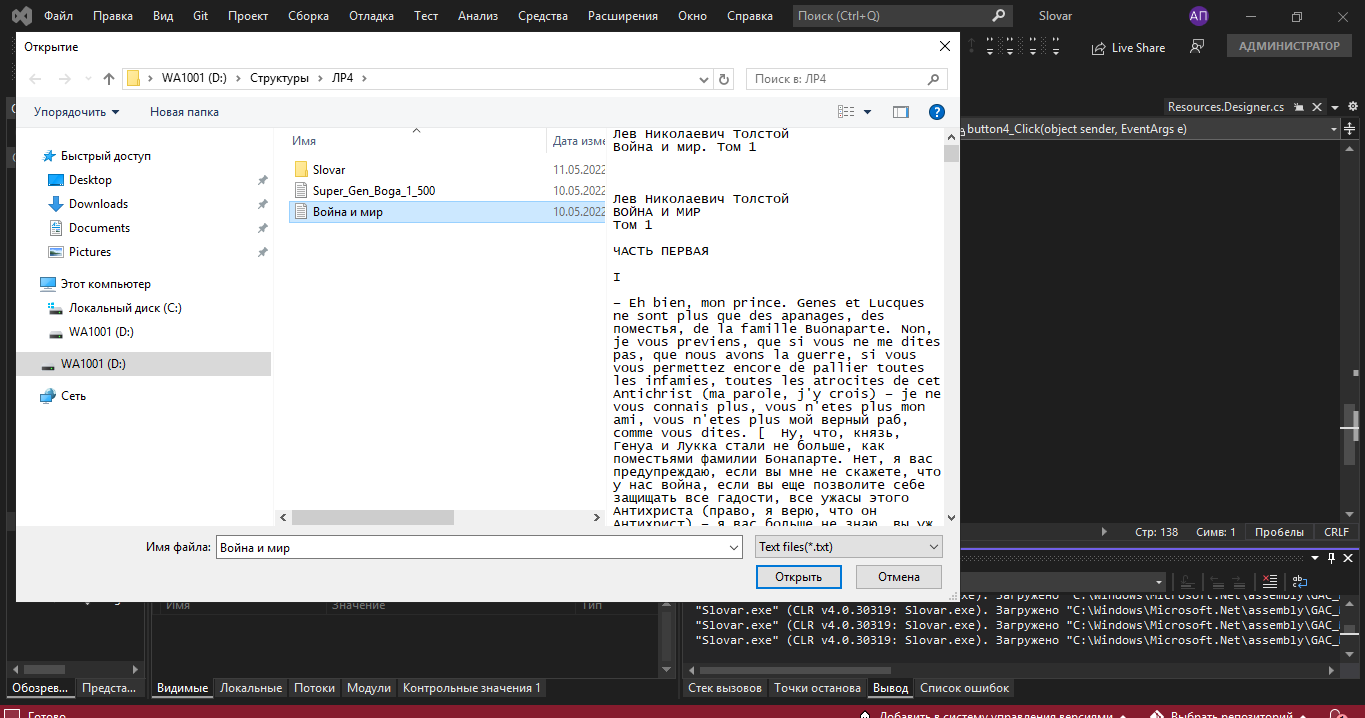
3. Выводить из дерева частотный словарь слов в алфавитном порядке.

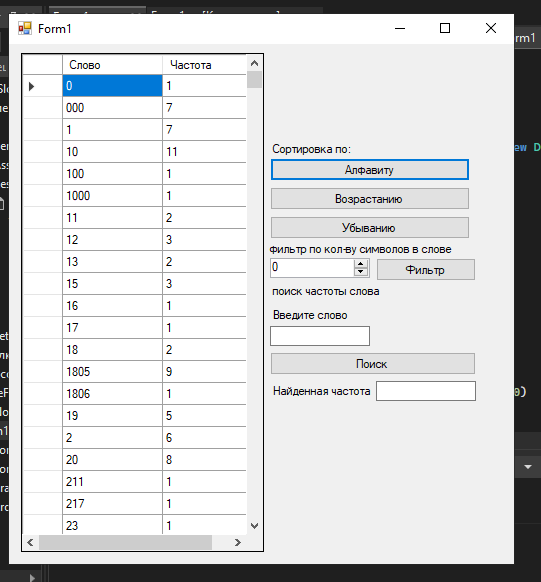
4. Выводить частотный словарь слов, отсортированный в порядке частоты их употребления.

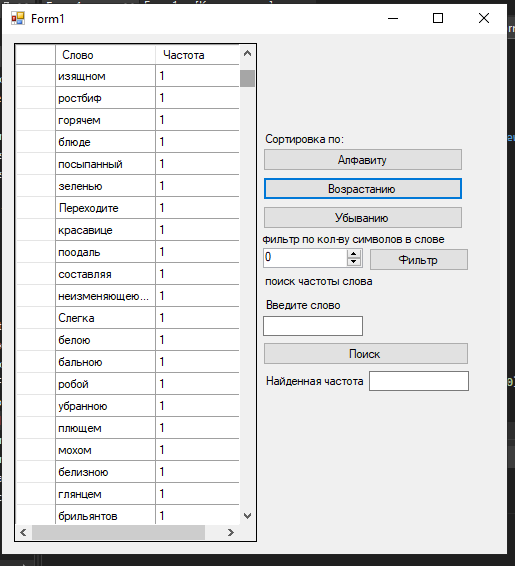
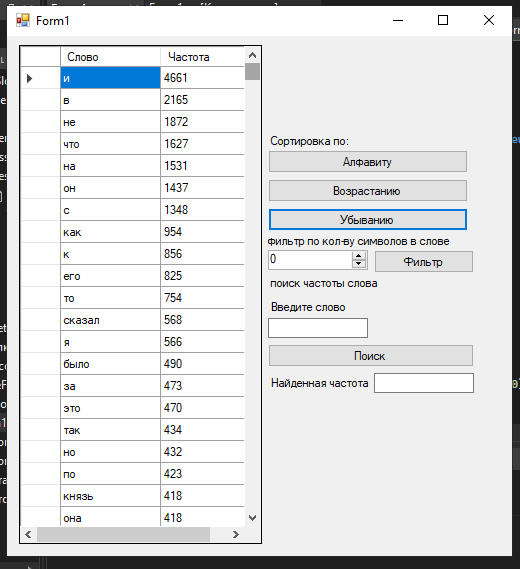
5. Производить поиск заданного слова.

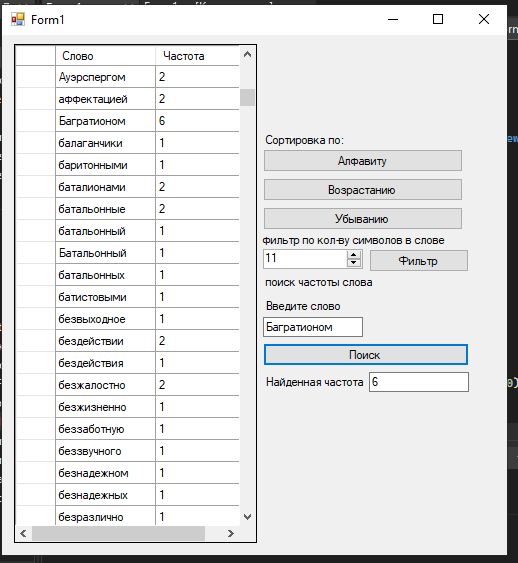
6. Производить фильтрацию, оставляющую в словаре слова определенной длины.

**Тестирование:**







Текст программы:

Код в файле Form1.cs

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;

namespace Slovar

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

openFileDialog1.Filter = "Text files(\*.txt)|\*.txt|All files(\*.\*)|\*.\*";

}

public Dictionary<string, int> dictionary = new Dictionary<string, int>();

void FormTree(Node root, string word)

{

if (root.word == null)

{

dictionary.Add(word, 1);

root.word = word;

root.freq = 1;

root.left = new Node();

root.right = new Node();

}

else

{

if (word.CompareTo(root.word) < 0)

FormTree(root.left, word);

else if (word.CompareTo(root.word) > 0)

FormTree(root.right, word);

else

{

root.freq++;

dictionary[word]++;

}

}

}

void PrintTree(Node root)

{

if (root != null)

{

if (root.left.word != null)

{

PrintTree(root.left);

}

dataGridView1.Rows.Add(root.word, root.freq);

if (root.right.word != null)

{

PrintTree(root.right);

}

}

}

void Search(Node root, String word)

{

textBox2.Clear();

if (root.left.word != null && word.CompareTo(root.word) < 0)

Search(root.left, word);

else if (root.right.word != null && word.CompareTo(root.word) > 0)

Search(root.right, word);

else if (word.CompareTo(root.word) == 0)

textBox2.Text += root.freq;

else

textBox2.Text = "Слово не найдено";

}

void Filtr(Node node, int x)

{

if (node.left.word != null)

{

Filtr(node.left, x);

}

if (node.word.Length == x)

{

dataGridView1.Rows.Add(node.word, node.freq);

}

if (node.right.word != null)

{

Filtr(node.right, x);

}

}

String[] str;

Node node = new Node();

private void Form1\_Load(object sender, EventArgs e)

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

// получаем выбранный файл

string filename = openFileDialog1.FileName;

str = System.IO.File.ReadAllText(filename, Encoding.Default).Split(' ', '\n', ',', '.', '"', '\r', '(', ')', '!', '?', '[', ']', '-', '/', '»', '«', ';', ':', '„', '…').ToArray();

for (int i = 0; i < str.Length; i++)

{

if (str[i].CompareTo(" ") > 0)

FormTree(node, str[i]);

}

}

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

{

dataGridView1.AllowUserToAddRows = true;

dataGridView1.Rows.Clear();

PrintTree(node);

dataGridView1.AllowUserToAddRows = false;

}

private void button2\_Click(object sender, EventArgs e)

{

dataGridView1.AllowUserToAddRows = true;

dataGridView1.Rows.Clear();

foreach (var x in dictionary.OrderBy(key => key.Value))//перебирает элементы массивов, векторов или любых других наборов данных. Он присваивает значение текущего элемента переменной итератора, объявленной внутри цикла.

{

dataGridView1.Rows.Add(x.Key, x.Value);

}

dataGridView1.AllowUserToAddRows = false;

}

private void button3\_Click(object sender, EventArgs e)

{

dataGridView1.AllowUserToAddRows = true;

dataGridView1.Rows.Clear();

foreach (var x in dictionary.OrderByDescending(key => key.Value))

{

dataGridView1.Rows.Add(x.Key, x.Value);

}

dataGridView1.AllowUserToAddRows = false;

}

private void button4\_Click(object sender, EventArgs e)

{

dataGridView1.AllowUserToAddRows = true;

dataGridView1.Rows.Clear();

Filtr(node, Convert.ToInt32(numericUpDown1.Value));

dataGridView1.AllowUserToAddRows = false;

}

private void button5\_Click(object sender, EventArgs e)

{

Search(node, textBox1.Text);

}

}

}

Код в файле CodeFile1.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Slovar

{

internal class Node

{

public string word = null;

public int freq;

public Node left;

public Node right;

}

}

Алгоритм:

1. Инициализируется словарь Dictionary<string, int> dictionary = new Dictionary<string, int>();//словарь, который хранит слово и ключ , переменная String[] str; и данную Node node = new Node();
2. Вызывается функция public Form1(), в которой открывается файл (openFileDialog1);
3. Вызывается функция Form1Load (object sender, EventArgs e) ;
   1. Если (openFileDialog1.ShowDialog() == DialogResult.Cancel), то
      1. Возращает значение
   2. получаем выбранный файл в переменную filename типа string;
   3. в строке str обрабатываем строку filename, считываем данные и делим на лексемы с помощью функций Split и ToArray();.
   4. Пока выполняется (int i = 0; i < str.Length; i++):
      1. Если (str[i].CompareTo(" ") > 0), то:
         1. Вызывется функция FormTree(node, str[i]); и передает туда узел и лексему.

CompareTo - Сравнивает данный экземпляр с указанным объектом System.Object и показывает,

// расположен ли экземпляр перед, после или в той же позиции в порядке сортировки,

// что и заданный объект System.Object.

1. функция FormTree(Node root, string word) Строит бинарное дерево в функции :

1. Функция принимает значения - указатель на узел дерева и word строку для добавления в дерево.

2. Если node = NULL

2.1. Создать новый узел для строки root.word = word; root.freq = 1; root.left = new Node(); root.right = new Node();

3. Иначе

3.1. Если word.CompareTo(root.word) < 0

3.1.1. Вызов функции formTree для левого потомка root.left и строки word

3.2. Иначе если word.CompareTo(root.word) > 0

3.2.1. Вызов функции formTree для правого потомка root.right и строки word

3.3. Иначе

3.3.1Увеличить значение root.freq++; dictionary[word]++;

Бинарное дерево поиска строится по принципу – 1 элемент корень дерева, остальные элементы сравниваются с корнем (или своим узлом) по, что для каждого элемента n все элементы в левом поддереве n будут меньше, чем n, а все элементы в правом поддереве — будут больше, чем n. Т.е. данные в бинарном дереве хранятся уже в отсортированном виде. При каждой операции вставки нового узла, отсортированный порядок дерева сохраняется. Когда дерево, сбалансированное его сложность, составляет O(log2n), а в худшем случае O(n).

1. При нажатии на кнопку «Алфавиту» в таблицу выводится из дерева частотный словарь слов в алфавитном порядке. Обход дерева осуществляется симметрично. Почему?

В симметричном обходе, потому что сначала посещаются левое поддерево, затем посещается корень и потом правое поддерево. Узел посещается путем применения функции обращения к элементу, записанному в узле.

При обходе бинарного дерева поиска в симметричном порядке получается отсортированный по возрастанию значения ключей. Эти значения выводятся в таблицу dataGridView1.

1. При нажатии на кнопку Возрастание словарь выводится в таблицу в порядке возрастания частоты появления слова, а при нажатии на кнопку «Убывание», наоборот. Откуда взялась последовательность слов по возрастанию (убыванию) частот?

С помощью цикла foreach перебираются элементы. Выполняется итерация по словарю(смотрится элемент и его ключ) .С помощью функции OrderBy (Упорядочивает последовательность по ключу в порядке возрастания), а с помощью функции OrderByDescending (Упорядочивает последовательность по ключу в порядке убывания) и внутри цикла foreach записываются в таблицу слова по возрастанию/убыванию.

1. При нажатии на кнопку Фильтр считывается значение из numericUpDown1, которое вводит пользователь. После этого происходит вызов функции Filtr(Node node, int x): если кол-во символов в слове (node.word.Length) удовлетворяет значению, выставленному в numericUpDown1 Value, тогда выводится слово и его частота употребления в таблицу. Иначе перебирает дерево и ищет это значение. Если слово не найдено выводится сообщение – слово не найдено!
2. При нажатии на кнопку Поиск считывается строка из textbox1. После чего вызывается функция поиска Search.
   1. Если строка совпадает с словом в дереве ((word.CompareTo(root.word) иона не пустая, то:
      1. в textbox2 выводится частота употребления и в таблица отображает найденное слово и частоту. В ином случае сообщается о том, что слово не было найдено.