Отчет по практической работе №25

Наименование работы: меню и панели инструментов.

Индивидуальные задания (вариант №9):

В Индивидуальное задание №1 используя элементы меню(задание 1) и панель инструментов(задание 2) добавьте возможность сохранить файл, открыть файл и прочитать данные, сохранить файл как… , изминение шрифта и размера текста.

1. Записать в файл оценки (в баллах), полученные некото­рым студентом на экзаменах в течение всех сессий. Добавить в начало файла оценки, полученные на вступительных экзаменах.
2. Из строки, расположенной в текстовом файле, исключить все символы, входящие в нее более одного раза.

Ход работы:

1. Разработал программу в соответствии с заданием №1. Были реализованы методы, классы, массивы.

Листинг программы:

using NewNoteBlock;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab25

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

this.Text = Properties.Settings.Default.newDocName + " - " + Properties.Settings.Default.programmName;

}

public MainForm(string fileName) // Открытие программы документом

{

InitializeComponent();

if (!string.IsNullOrEmpty(fileName) && File.Exists(fileName))

{

try

{

string programmName = Properties.Settings.Default.programmName;

FileStream file = new FileStream(fileName, FileMode.Open, FileAccess.Read);

StreamReader reader = new StreamReader(file, Encoding.Default);

notebox.Text = reader.ReadToEnd();

reader.Close();

docPath = fileName;

tbChange = false;

this.Text = Path.GetFileName(fileName) + " — " + programmName;

notebox.Select(0, 0);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

bool tbChange = false;

string docPath = "";

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

{

}

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

{

tbChange = true;

TextWork.StatusAnalize(ref notebox, ref statusLinesCount, ref statusWordsCount);

}

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

{

this.Width = Properties.Settings.Default.formWidth;

this.Height = Properties.Settings.Default.formHeight;

notebox.Font = Properties.Settings.Default.textFont;

}

private void MainForm\_FormClosing(object sender, FormClosingEventArgs e)

{

Properties.Settings.Default.formWidth = this.Width;

Properties.Settings.Default.formHeight = this.Height;

Properties.Settings.Default.textTransfer = notebox.WordWrap;

Properties.Settings.Default.textFont = notebox.Font;

Properties.Settings.Default.statusStripVisible = statusStrip.Visible;

Properties.Settings.Default.Save();

if (tbChange == true)

{

DialogResult message = MessageBox.Show("Сохранить текущий документ перед выходом?", "Выход из программы", MessageBoxButtons.YesNoCancel);

if (message == DialogResult.Yes)

{

if (docPath != "")

{

FileWork.SaveFile(ref notebox, ref tbChange, ref docPath);

Application.Exit();

}

else if (docPath == "")

{

FileWork.SaveAsFile(ref notebox, ref tbChange, ref docPath);

Application.Exit();

}

}

else if (message == DialogResult.Cancel)

{

e.Cancel = true;

}

}

}

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

{

if (tbChange == true)

{

DialogResult message = MessageBox.Show("Сохранить текущий документ перед созданием нового?", "Создание документа", MessageBoxButtons.YesNoCancel);

if (message == DialogResult.Yes)

{

if (docPath != "")

{

FileWork.SaveFile(ref notebox, ref tbChange, ref docPath);

FileWork.CreateFile(ref notebox, ref tbChange, ref docPath);

}

else if (docPath == "")

{

FileWork.SaveAsFile(ref notebox, ref tbChange, ref docPath);

FileWork.CreateFile(ref notebox, ref tbChange, ref docPath);

}

}

else if (message == DialogResult.No)

{

FileWork.CreateFile(ref notebox, ref tbChange, ref docPath);

}

}

else

{

FileWork.CreateFile(ref notebox, ref tbChange, ref docPath);

}

}

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

{

if (tbChange == true)

{

DialogResult message = MessageBox.Show("Сохранить текущий документ перед открытием нового?", "Открытие документа", MessageBoxButtons.YesNoCancel);

if (message == DialogResult.Yes)

{

if (docPath != "")

{

FileWork.SaveFile(ref notebox, ref tbChange, ref docPath);

FileWork.OpenFile(ref notebox, ref tbChange, ref docPath);

}

else if (docPath == "")

{

FileWork.SaveAsFile(ref notebox, ref tbChange, ref docPath);

FileWork.OpenFile(ref notebox, ref tbChange, ref docPath);

}

}

else if (message == DialogResult.No)

{

FileWork.OpenFile(ref notebox, ref tbChange, ref docPath);

}

else

{

return;

}

}

else

{

FileWork.OpenFile(ref notebox, ref tbChange, ref docPath);

}

}

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

{

if (docPath != "")

{

FileWork.SaveFile(ref notebox, ref tbChange, ref docPath);

}

else

{

FileWork.SaveAsFile(ref notebox, ref tbChange, ref docPath);

}

}

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

{

FileWork.SaveAsFile(ref notebox, ref tbChange, ref docPath);

}

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

{

Application.Exit();

}

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

{

notebox.Undo();

}

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

{

if (notebox.SelectionLength > 0)

{

notebox.Cut();

}

}

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

{

if (notebox.SelectionLength > 0)

{

notebox.Copy();

}

}

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

{

notebox.Paste();

}

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

{

GoToForm gotoform = new GoToForm();

gotoform.Owner = this;

gotoform.tbLineNum.Minimum = 0;

gotoform.tbLineNum.Maximum = notebox.Lines.Count();

gotoform.ShowDialog();

}

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

{

SearchForm findText = new SearchForm();

findText.Owner = this;

findText.Show();

}

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

{

if (notebox.SelectionLength > 0)

{

notebox.SelectedText = "";

}

}

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

{

notebox.SelectAll();

}

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

{

notebox.AppendText(Environment.NewLine + Convert.ToString(System.DateTime.Now));

}

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

{

FontDialog.Font = notebox.Font;

DialogResult = FontDialog.ShowDialog();

if (DialogResult == DialogResult.OK)

{

notebox.Font = FontDialog.Font;

}

}

private void mFormatTransfer\_CheckStateChanged(object sender, EventArgs e)

{

if (mFormatTransfer.CheckState == CheckState.Checked)

{

notebox.WordWrap = true;

notebox.ScrollBars = ScrollBars.Vertical;

mEditGo.Enabled = false;

statusStrip.Visible = false;

statusLinesCount.Visible = false;

}

else

{

notebox.WordWrap = false;

notebox.ScrollBars = ScrollBars.Both;

mEditGo.Enabled = true;

statusStrip.Visible = true;

statusLinesCount.Visible = true;

}

}

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

{

TextWork.mEditEnableds(ref notebox, ref mEditCopy, ref mEditCut, ref mEditDel, ref mEditFind, ref mEditGo);

}

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

{

if(mViewStatusStrip.Checked == true)

{

statusStrip.Visible = true;

}

else

{

statusStrip.Visible = false;

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab25

{

class FileWork

{

public static void CreateFile(ref TextBox notebox, ref bool tbChange, ref string docPath) // Метод "Создать новый документ"

{

string newDocName = Properties.Settings.Default.newDocName;

string programmName = Properties.Settings.Default.programmName;

notebox.Clear();

docPath = "";

tbChange = false;

MainForm.ActiveForm.Text = newDocName + " — " + programmName;

}

public static void OpenFile(ref TextBox notebox, ref bool tbChange, ref string docPath) // Метод "Открыть документ"

{

string programmName = Properties.Settings.Default.programmName;

OpenFileDialog openDocument = new OpenFileDialog();

openDocument.Title = "Открыть текстовый документ";

openDocument.Filter = "Текстовые файлы (\*.txt) |\*.txt| Все файлы (\*.\*)|\*.\*";

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

{

FileStream file = new FileStream(openDocument.FileName, FileMode.Open, FileAccess.Read);

StreamReader reader = new StreamReader(file, Encoding.Default);

notebox.Text = reader.ReadToEnd();

reader.Close();

docPath = openDocument.FileName;

tbChange = false;

MainForm.ActiveForm.Text = openDocument.SafeFileName + " — " + programmName;

}

}

public static void SaveFile(ref TextBox notebox, ref bool tbChange, ref string docPath) // Метод "Сохранить документ"

{

FileStream file = new FileStream(docPath, FileMode.Create, FileAccess.Write);

StreamWriter writer = new StreamWriter(file, Encoding.Default);

writer.Write(notebox.Text);

writer.Close();

tbChange = false;

}

public static void SaveAsFile(ref TextBox notebox, ref bool tbChange, ref string docPath) // Метод "Сохранить документ как..."

{

string programmName = Properties.Settings.Default.programmName;

SaveFileDialog saveAsDocument = new SaveFileDialog();

saveAsDocument.Title = "Сохранить документ как...";

saveAsDocument.FileName = "Текстовый документ";

saveAsDocument.Filter = "Текстовые файлы (\*.txt) |\*.txt| Все файлы (\*.\*)|\*.\*";

if (saveAsDocument.ShowDialog() == DialogResult.OK) //Если пользователь подтвердил сохранение

{

//Создаем файл по пути, выбранному в окне сохранения

FileStream file = new FileStream(saveAsDocument.FileName, FileMode.Create, FileAccess.Write);

StreamWriter writer = new StreamWriter(file, Encoding.Default);

writer.Write(notebox.Text); //записываем содержимое в файл

writer.Close(); //закрываем поток

tbChange = false;

docPath = saveAsDocument.FileName;

MainForm.ActiveForm.Text = Path.GetFileName(saveAsDocument.FileName) + " — " + programmName;

}

else

{

tbChange = true;

return;

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace NewNoteBlock

{

public static class TextWork

{

// Метод поиска текста в TextBox

// Для использования создаем в форме поиска глобальную переменную

// типа int = 0 для стартовой позиции поиска,

// передаем в метод ссылки на TextBox'ы с исходным и искомым текстами,

// а также необходимо указать, учитывать ли регистр букв при поиске (True - учитывать, False - не учитывать)

public static int FindTextBox(ref TextBox textBox, string findText, ref int findCutLength, bool register)

{

// Поиск с учетом регистра

if (register == true)

{

if (textBox.Text.Contains(findText))

{

// Заносим текст в переменную string, удаляем из него уже пройденный

// текст (findCutLength) в переменной nextText

string text = textBox.Text;

string nextText = text.Remove(0, findCutLength);

// Ищем в nextText

int resultPosition = nextText.IndexOf(findText);

// Если искомое выражение найдено - выделяем его, добавляем его позицию и длину

// к значению пройденного текста (findCutLenght)

if (resultPosition != -1)

{

textBox.Select(resultPosition + findCutLength, findText.Length);

textBox.ScrollToCaret();

textBox.Focus();

findCutLength += findText.Length + resultPosition;

}

// Если попытка поиска не первая, и больше совпадений в тексте нет - обнуляем

// значение пройденного текста и начинаем поиск сначала

else if (resultPosition == -1 && findCutLength != 0)

{

findCutLength = 0;

return FindTextBox(ref textBox, findText, ref findCutLength, register);

}

}

else

{

findCutLength = 0;

MessageBox.Show("По вашему запросу ничего не нашлось.", "Совпадений не найдено", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

// Поиск без учета регистра

else if (register == false)

{

if (textBox.Text.ToLower().Contains(findText.ToLower()))

{

string text = textBox.Text.ToLower();

string nextText = text.Remove(0, findCutLength);

int resultPosition = nextText.IndexOf(findText.ToLower());

if (resultPosition != -1)

{

textBox.Select(resultPosition + findCutLength, findText.Length);

textBox.ScrollToCaret();

textBox.Focus();

findCutLength += findText.Length + resultPosition;

}

else if (resultPosition == -1 && findCutLength != 0)

{

findCutLength = 0;

return FindTextBox(ref textBox, findText, ref findCutLength, register);

}

}

// Если текст изначально не содержит результатов поиска - обнуляем findCutLength, выводим сообщение

else

{

findCutLength = 0;

MessageBox.Show("По вашему запросу ничего не нашлось.", "Совпадений не найдено", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

return 0;

}

// Метод "Заменить"

public static int ReplaceTextBox(ref TextBox textBox, string findText, string replaceText, ref int findCutLength, bool register)

{

if (register == true)

{

if (textBox.Text.Contains(findText))

{

if (textBox.SelectedText == "" || textBox.SelectedText != findText)

{

string text = textBox.Text;

string nextText = text.Remove(0, findCutLength);

int resultPosition = nextText.IndexOf(findText);

if (resultPosition != -1)

{

textBox.Select(resultPosition + findCutLength, findText.Length);

textBox.ScrollToCaret();

textBox.Focus();

findCutLength += findText.Length + resultPosition;

}

else if (resultPosition == -1 && findCutLength != 0)

{

findCutLength = 0;

return ReplaceTextBox(ref textBox, findText, replaceText, ref findCutLength, register);

}

}

else if (textBox.SelectedText == findText)

{

textBox.SelectedText = replaceText;

}

}

else

{

findCutLength = 0;

MessageBox.Show("По вашему запросу ничего не нашлось.", "Совпадений не найдено", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

else if (register == false)

{

if (textBox.Text.ToLower().Contains(findText.ToLower()))

{

if (textBox.SelectedText == "" || textBox.SelectedText.ToLower() != findText.ToLower())

{

string text = textBox.Text.ToLower();

string nextText = text.Remove(0, findCutLength);

int resultPosition = nextText.IndexOf(findText.ToLower());

if (resultPosition != -1)

{

textBox.Select(resultPosition + findCutLength, findText.Length);

textBox.ScrollToCaret();

textBox.Focus();

findCutLength += findText.Length + resultPosition;

}

else if (resultPosition == -1 && findCutLength != 0)

{

findCutLength = 0;

return ReplaceTextBox(ref textBox, findText, replaceText, ref findCutLength, register);

}

}

else if (textBox.SelectedText.ToLower() == findText.ToLower())

{

textBox.SelectedText = replaceText;

}

}

else

{

findCutLength = 0;

MessageBox.Show("По вашему запросу ничего не нашлось.", "Совпадений не найдено", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

return 0;

}

// Метод "Заменить всё"

public static int ReplaceAllTextBox(ref TextBox textBox, string findText, string replaceText, bool register)

{

if (register == true)

{

string text = textBox.Text;

string words = findText;

if (textBox.Text.Contains(words))

{

int startPosition = text.IndexOf(words);

textBox.Select(startPosition, words.Length);

textBox.SelectedText = replaceText;

return ReplaceAllTextBox(ref textBox, findText, replaceText, register);

}

else

{

MessageBox.Show("Замены произведены успешно.", "Заменить всё", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

else if (register == false)

{

string text = textBox.Text.ToLower();

string words = findText.ToLower();

if (text.Contains(words))

{

int startPosition = text.IndexOf(words);

textBox.Select(startPosition, findText.Length);

textBox.SelectedText = replaceText;

return ReplaceAllTextBox(ref textBox, findText, replaceText, register);

}

else

{

MessageBox.Show("Замены произведены успешно.", "Заменить всё", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

return 0;

}

public static void mEditEnableds(ref TextBox notebox, ref ToolStripMenuItem mEditCopy, ref ToolStripMenuItem mEditCut, ref ToolStripMenuItem mEditDel, ref ToolStripMenuItem mEditFind, ref ToolStripMenuItem mEditGo)

{

if (notebox.Text.Length < 1)

{

mEditCopy.Enabled = false;

mEditCut.Enabled = false;

mEditDel.Enabled = false;

mEditFind.Enabled = false;

mEditGo.Enabled = false;

}

else

{

mEditCopy.Enabled = true;

mEditCut.Enabled = true;

mEditDel.Enabled = true;

mEditFind.Enabled = true;

mEditGo.Enabled = true;

}

}

public static void StatusAnalize(ref TextBox notebox, ref ToolStripStatusLabel statusLinesCount, ref ToolStripStatusLabel statusWordsCount)

{

string text = notebox.Text;

// Количество строк в тексте

statusLinesCount.Text = notebox.Lines.Count().ToString();

// Количество слов в тексте

statusWordsCount.Text = text.Split(new Char[] { ' ', '\t', '\n', '\r', '!', '@', '#', '$', '%', '^', '&', '\*', '(', ')', '-',

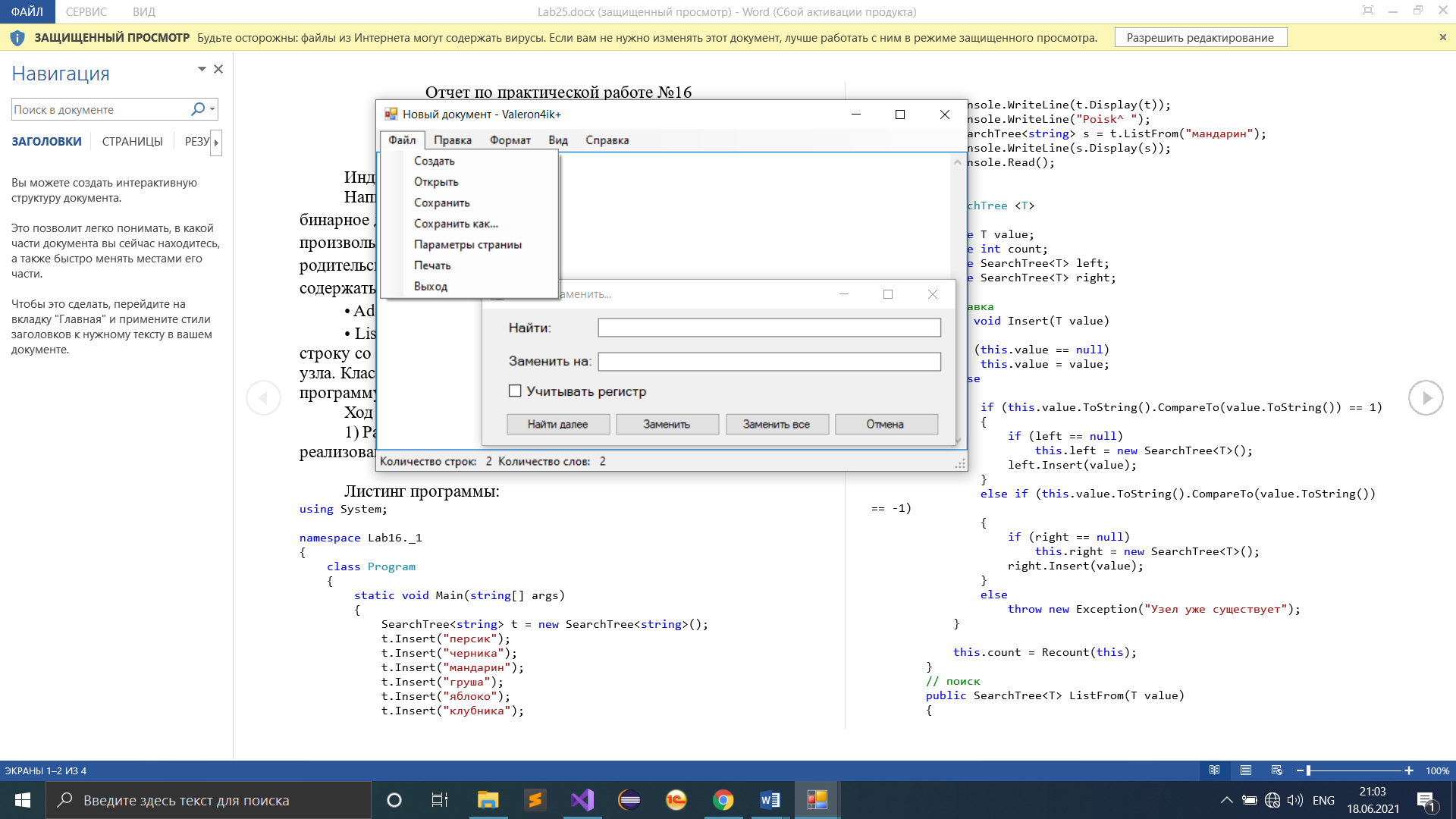
'\_', '+', '=', '[', '{', ']', '}', '/', '\\', '|', '"', ':', ';', '.', ',', '>', '<' }, StringSplitOptions.RemoveEmptyEntries).Length.ToString();

}

}

}

Результат выполнения программы Рис 1.1.



**Рисунок 1.1 – Пример работы программы работы с файлами**

Отметка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_