**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

Институт среднего профессионального образования

**ОТЧЕТ**

**по лабораторной работе №9**

**Разработка класса по работе с текстовыми файлами**

по профессиональному модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем»

(код и наименование)

Специальность09.02.07 «Информационные системы и программирование»

(код и наименование специальности)

Выполнили студенты 3 курса 32919/1 группы

Козлов Вячеслав Алексеевич

Родионова Дарья Александровна

(Фамилия, имя, отчество)

Преподаватель: Ильин Ю. П.

Санкт-Петербург

2025

**Цель работы:** получить практические навыки создания программ обработки файлов, использования диалоговых компонентов, обобщить и расширить знания по разработке многомодульных приложений.

**Бригада 12**

**Задание 1.**

1. Внимательно прочитайте условие задачи (см. ниже). Подумайте, какие элементы интерфейса для работы с файлом необходимо разместить на форме.
2. Выполните конструирование пользовательского интерфейса, используя компоненты «Главное меню», «Панель инструментов», «Строка состояния» и «Многострочный редактор».
3. Создайте дополнительный класс (динамическую библиотеку), в который поместите методы обработки текстового файла: открытие файла, сохранение файла, вывод текста на принтер, метод поиска слов (по заданию).
4. Создайте метод обработки события выбора пункта меню «Открыть файл». При выборе этого пункта пользователю предлагается выбрать файл для открытия при помощи диалогового окна, а затем вызывается ранее созданная функция открытия файла.
5. Сохраните и откомпилируйте проект. Исправьте синтаксические ошибки.
6. Запустите проект на выполнение. Протестируйте метод выбора пункта меню «Открыть файл». Произведите отладку проекта.
7. Создайте метод выбора пункта меню «Сохранить файл» и «Сохранить как». Эти функции будут похожи. При обычном сохранении файла используем тоже имя, которое было задано в переменной при открытии файла. Если пользователь вводил текст вручную, не открывая файл, то нужно предложить ему сохранить его. При выборе пункта меню «Сохранить как» происходит вызов диалогового окна сохранения файла. Произведите отладку проекта.
8. Создайте метод выбора пункта меню «Печать». При нажатии на кнопку вызывается соответствующий метод. Произведите отладку проекта.
9. Назначьте соответствующим кнопкам на панели инструментов методы обработки событий выбора нужных пунктов меню. Дважды писать один и тот же код в программе не нужно.
10. Создайте методы обработки события выбора остальных пунктов меню: «Параметры», «Поиск слов», «Справка». Пункт меню «Параметры» должен предоставлять возможности скрыть/показать панель инструментов, настроить параметры шрифта с помощью диалогового окна «Шрифт», настроить цвет шрифта с помощью диалогового окна выбора цвета.
11. Скопируйте в отчёт тексты всех модулей. Внесите подробные комментарии.
12. Составьте тестовый сценарий (план тестирования). Занесите его в отчёт.
13. Протестируйте приложение в соответствие с тестовым сценарием. Вставьте в отчёт скриншоты проекта на этапе выполнения (результаты тестирования).
14. Продемонстрируйте работу проекта и отчёт преподавателю и получите оценку.

**Задание 2.**

Поместить в текстовый файл результаты измерений роста и веса студентов группы. Разработать метод, который определяет индекс массы тела (ИМТ) каждого студента. Индекс массы тела определяется по следующей формуле: вес (в килограммах) делится на рост в квадрате (в метрах). Используя созданный метод, определить и выдать отдельными списками студентов с недостаточным и избыточным весом. Нормальное значение ИМТ равно 18-25, ниже 18 вес недостаточен, более 25 вес избыточный.

# Текст модуля библиотеки:

using System.IO;

namespace TextEditorFunctions

{

public static class TextEditor

{

public static string Open(string fileName)

{

string text = "";

try

{

using (StreamReader reader = new StreamReader(fileName, System.Text.Encoding.GetEncoding("UTF-8")))

{

text = reader.ReadToEnd();

}

}

catch

{

}

return text;

}

public static void Save(string fileName, string text) {

if (fileName == string.Empty)

{

TextEditor.SaveAs(fileName, text);

}

else if (fileName != string.Empty)

{

try

{

using (StreamWriter writer = new FileInfo(fileName).CreateText()) {

writer.Write(text);

}

}

catch

{

}

}

}

public static void SaveAs(string fileName, string text)

{

try

{

using (FileStream fstr = new FileStream(fileName, FileMode.Create)) { } ;

FileInfo fileInfo = new FileInfo(fileName);

using (StreamWriter writer = new FileInfo(fileName).CreateText())

{

writer.Write(text);

}

}

catch

{

}

}

}

}

# Текст модуля программы:

using System;

using IMTFinder;

using TextEditorFunctions;

using System.Windows.Forms;

using System.IO;

using System.Collections.Generic;

using System.Linq;

using System.Drawing.Printing;

using System.Drawing;

namespace TextEditor

{

public partial class Form1 : Form

{

public string fileName = string.Empty;

public bool isSaved;

Form2 form2 = new Form2();

public Form1()

{

InitializeComponent();

панельИнструментовToolStripMenuItem.Checked = true;

toolStripStatusLabel3.Text = "Состояние: создание файла";

}

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

{

toolStripButton1\_Click(sender, e);

}

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

{

isSaved = true;

OpenFileDialog fileDialog = new OpenFileDialog();

fileDialog.Filter = "Text|\*.txt|All|\*.\*";

fileDialog.ShowDialog();

this.fileName = fileDialog.FileName;

if (fileDialog.FileName != string.Empty)

{

string text = TextEditorFunctions.TextEditor.Open(fileDialog.FileName);

richTextBox1.Text = text;

this.Text = fileDialog.FileName;

toolStripStatusLabel3.Text = "Состояние: открытие файла";

}

}

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

{

if (fileName == string.Empty)

{

сохранитьКакToolStripMenuItem\_Click(sender, e);

}

else

{

TextEditorFunctions.TextEditor.Save(this.fileName, richTextBox1.Text);

isSaved = true;

MessageBox.Show("Файл успешно сохранен!", "", MessageBoxButtons.OK);

}

}

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

{

isSaved = true;

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Text|\*.txt|All|\*.\*";

saveFileDialog.ShowDialog();

TextEditorFunctions.TextEditor.SaveAs(saveFileDialog.FileName, richTextBox1.Text);

try

{

if (new FileInfo(saveFileDialog.FileName).Exists)

{

MessageBox.Show("Файл успешно сохранен!", "", MessageBoxButtons.OK);

}

}

catch { }

}

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

{

toolStripButton5\_Click(sender, e);

}

private void toolStripButton5\_Click(object sender, EventArgs e)

{

PrintDocument printDocument = new PrintDocument();

printDocument.PrintPage += new PrintPageEventHandler(PrintPageHandler);

PrintDialog printDialog = new PrintDialog();

printDialog.Document = printDocument;

if (printDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

printDocument.Print();

}

}

private void PrintPageHandler(object sender, PrintPageEventArgs e)

{

e.Graphics.DrawString(richTextBox1.Text, richTextBox1.Font, Brushes.Black, new PointF(50, 50));

}

private void toolStripButton2\_Click(object sender, EventArgs e)

{

открытьToolStripMenuItem\_Click(sender, e);

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

if (fileName != string.Empty && isSaved)

{

richTextBox1.Text = string.Empty;

fileName = string.Empty;

isSaved = false;

this.Text = "Новый файл";

toolStripStatusLabel3.Text = "Состояние: создание файла";

return;

}

if ((fileName != string.Empty && !isSaved) || (fileName == string.Empty && richTextBox1.Text != string.Empty))

{

DialogResult result = MessageBox.Show(

"У вас есть несохраненные изменения. Хотите сохранить изменения перед выходом?",

"Подтвердите сохранение",

MessageBoxButtons.YesNoCancel,

MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

TextEditorFunctions.TextEditor.Save(fileName, richTextBox1.Text);

richTextBox1.Text = string.Empty;

fileName = string.Empty;

isSaved = false;

this.Text = "Новый файл";

toolStripStatusLabel3.Text = "Состояние: создание файла";

}

else if (result == DialogResult.No)

{

richTextBox1.Text = string.Empty;

fileName = string.Empty;

isSaved = false;

this.Text = "Новый файл";

toolStripStatusLabel3.Text = "Состояние: создание файла";

}

return;

}

}

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

{

if (richTextBox1.Text != string.Empty)

{

toolStripStatusLabel1.Text = $"Количество символов: {richTextBox1.Text.Length}";

toolStripStatusLabel2.Text = $"Строк: {richTextBox1.Text.Count(nchar => nchar == '\n') + 1}";

}

else

{

toolStripStatusLabel1.Text = "Количество символов: 0";

toolStripStatusLabel2.Text = "Строк: 0";

}

isSaved = false;

}

private void toolStripButton3\_Click(object sender, EventArgs e)

{

сохранитьToolStripMenuItem\_Click(sender, e);

}

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

{

toolStrip1.Visible = !панельИнструментовToolStripMenuItem.Checked;

панельИнструментовToolStripMenuItem.Checked = !панельИнструментовToolStripMenuItem.Checked;

if (панельИнструментовToolStripMenuItem.Checked)

{

richTextBox1.Height -= 60;

richTextBox1.Location = new System.Drawing.Point(6, 87);

}

else

{

richTextBox1.Height += 60;

richTextBox1.Location = new System.Drawing.Point(6, 27);

}

}

private void toolStripMenuItem1\_Click(object sender, EventArgs e)

{

FontDialog ftDialog = new FontDialog();

ftDialog.ShowDialog();

richTextBox1.Font = ftDialog.Font;

}

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

{

fileName = string.Empty;

isSaved = false;

this.Text = "Новый файл";

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

DialogResult result = MessageBox.Show(

"У вас есть несохраненные изменения. Хотите сохранить изменения перед выходом?",

"Подтвердите сохранение",

MessageBoxButtons.YesNoCancel,

MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

if (fileName == string.Empty)

{

isSaved = false;

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Text|\*.txt|All|\*.\*";

DialogResult res = saveFileDialog.ShowDialog();

if (res == DialogResult.Cancel)

{

e.Cancel = true;

MessageBox.Show("Сохранение файла было отменено.");

}

fileName = saveFileDialog.FileName;

}

TextEditorFunctions.TextEditor.Save(fileName, richTextBox1.Text);

richTextBox1.Text = string.Empty;

fileName = string.Empty;

isSaved = false;

}

else if (result == DialogResult.No)

{

richTextBox1.Text = string.Empty;

fileName = string.Empty;

isSaved = false;

}

else if(result == DialogResult.Cancel)

{

e.Cancel = true;

}

}

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

{

toolStripButton1\_Click(sender, e);

DialogResult result = MessageBox.Show(

"Функция ИМТ использует только текстовые файлы с определенным форматированием\n" +

"Пример:\n" +

"Иванов Сергей/40/1,99;\nПетров Петр/65/1,81;\nИванов Иван/5/1,7",

"Пожалуйста, используйте правильный формат!",

MessageBoxButtons.OKCancel,

MessageBoxIcon.Information);

if (result == DialogResult.OK)

{

OpenFileDialog fileDialog = new OpenFileDialog();

fileDialog.Filter = "Text|\*.txt|All|\*.\*";

fileDialog.ShowDialog();

if (fileDialog.FileName != string.Empty)

{

Person[] data = IMT.FindIMT(fileDialog.FileName);

if (data == null)

{

MessageBox.Show("Повторите попытку с другим файлом!", "Неверный формат файла!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

List<Person> smallIMT = new List<Person>();

List<Person> GoodIMT = new List<Person>();

List<Person> BigIMT = new List<Person>();

foreach (Person person in data)

{

if (person.IMT < 18)

{

smallIMT.Add(person);

}

else if (person.IMT > 25)

{

BigIMT.Add(person);

}

else

{

GoodIMT.Add(person);

}

}

richTextBox1.AppendText("Недостаточный ИМТ (< 18):\n");

foreach (Person person in smallIMT)

{

richTextBox1.AppendText(person.ToString() + "\n");

}

richTextBox1.AppendText("\nИМТ в норме (18 - 25):\n");

foreach (Person person in GoodIMT)

{

richTextBox1.AppendText(person.ToString() + "\n");

}

richTextBox1.AppendText("\nИзбыточный ИМТ (> 25):\n");

foreach (Person person in BigIMT)

{

richTextBox1.AppendText(person.ToString() + "\n");

}

}

}

else

{

MessageBox.Show("Отмена");

}

}

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

{

if (!form2.Created) {

form2 = new Form2();

form2.Show();

}

else

{

form2.Activate();

}

}

}

}

# Текст модуля для 2ого задания:

using System;

using System.Collections.Generic;

using System.IO;

namespace IMTFinder

{

public static class IMT

{

public static Person[] FindIMT(string fileName)

{

Person[] data = IMT.Parse(IMT.Import(fileName));

if (data == null)

{

return null;

}

for (int i = 0; i < data.Length - 1; i++)

{

for (int j = 0; j < data.Length - i - 1; j++)

{

if (data[j].IMT > data[i].IMT)

{

Person temp = data[j];

data[j] = data[j + 1];

data[j + 1] = temp;

}

}

}

return data;

}

public static string[] Import(string fileName)

{

string[] result = new string[0];

try

{

using (StreamReader reader = new StreamReader(fileName, System.Text.Encoding.UTF8))

{

string text = reader.ReadToEnd().TrimEnd(';').Replace("\n", "").Replace("\r", "");

result = text.Split(';', (char)StringSplitOptions.RemoveEmptyEntries);

}

}

catch { }

return result;

}

public static Person[] Parse(string[] data)

{

Person[] result = new Person[data.Length];

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

if (data[i] != "" && data[i] != string.Empty) {

string[] temp = data[i].Split('/');

Person pers = new Person();

try

{

pers.FullName = temp[0];

pers.Weight = Math.Round(double.Parse(temp[1]), 2);

pers.Height = Math.Round(double.Parse(temp[2]), 2);

pers.IMT = Math.Round(pers.Weight / Math.Pow(pers.Height, 2), 2);

result[i] = pers;

}

catch(IndexOutOfRangeException)

{

return null;

}

}

}

return result;

}

}

public class Person

{

public string FullName { get; set; }

public double Height { get; set; }

public double Weight { get; set; }

public double IMT { get; set; }

public override string ToString()

{

return $"{this.FullName}, вес: {this.Weight}, рост: {this.Height}, ИМТ: {this.IMT}";

}

}

}

# Тестовый сценарий:

|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Результаты** |
| Способность программы создать новый файл | Нажать кнопку "Создать файл":  1. остались несохраненные изменения; | Появляется сообщение для подтверждения сохранения изменений | Соответствует ожиданиям. См. рис. 1 |
| 2. все изменения сохранены. | Открывается новый пустой файл для редактирования | Соответствует ожиданиям. См. рис. 2 |
| Способность программы сохранить файл | Ввести текст в редактор, нажать "Сохранить" и выбрать путь:  1. изменения вносились в новый файл | Предлагается дать файлу имя и выбрать место хранения, появляется сообщение об успешном сохранении | Соответствует ожиданиям. См. рис. 3, 4 |
| 2. изменения вносились в существующий файл | Файл сохраняется по указанному пути, появляется сообщение об успешном сохранении | Соответсвует ожиданиям. См рис.5 |
| Способность программы открыть существующий файл | Нажать "Открыть файл" и выбрать файл с текстом | Текст из файла загружается в редактор | Соответствует ожиданиям. См. рис.6-7 |
| Способность программы корректно учитывать количество символов | Ввести текст в редакторе | В нижней строке состояния отображается актуальное количество символов | Соответствует ожиданиям. См. рис. 8 |
| Способность программы учитывать количество строк | Ввести несколько строк текста | В нижней строке состояния отображается актуальное количество строк | Соответствует ожиданиям. См. рис. 9 |
| Способность программы печатать документ | Нажать кнопку "Печать" | Открывается диалог печати | Соответствует ожиданиям. См. рис. 10-12 |
| Способность программы уведомлять о несохраненных изменениях | Ввести текст, не сохраняя, и попытаться закрыть программу | Открывается диалоговое окно с предупреждением о несохраненных изменениях | Соответствует ожиданиям. См. рис. 13 |
| Способность программы выполнять функцию расчета ИМТ | Нажать кнопку «ИМТ», нажать «ок» после прочтения справки, выбрать файл с:  1. корректными данными | Появляется справка по функции, рассчитывается ИМТ для каждого студента. Результаты делятся на 3 группы (недостаточный ИМТ, ИМТ в норме, Избыточный ИМТ) | Соответствует ожиданиям. См. рис. 14-16 |
| 2. данными, неудовлетворяющими инструкции в справке | Появляется сообщение об ошибке | Соответствует ожиданиям. См. рис. 17-18 |

# Результаты тестирования:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 1. Сообщение для подтверждения сохранения изменений

Изображение выглядит как снимок экрана, текст, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 2. Открыт пустой файл

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 3. Сохранение нового файла

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 4. Сообщение об успешном сохранении файла

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 5. Сообщение об успешном сохранении изменений

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 6. Текст файла в блокноте

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 7. В редакторе отобразился текст файла

Изображение выглядит как снимок экрана, текст, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 8. Строка состояния корректно отображает кол-во символов в документе

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 9. Строка состояния корректно отображает кол-во строк в файле

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 10. Открытый диалог печати

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 11. Печать в файл

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 12. Успешное создание pdf-файла для печати

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 13. Предупреждение о несохраненных изменениях

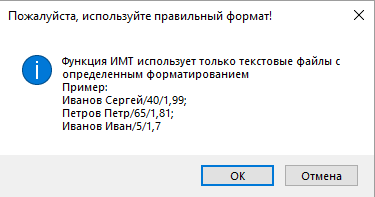


Рисунок 14. Справка по функции

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 15. Сводный файл с информацией о студентах

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 16. Рассчитанный и распределенный ИМТ

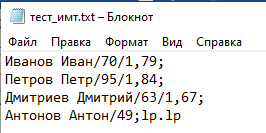


Рисунок 17. Файл с некорректной информацией о студентах

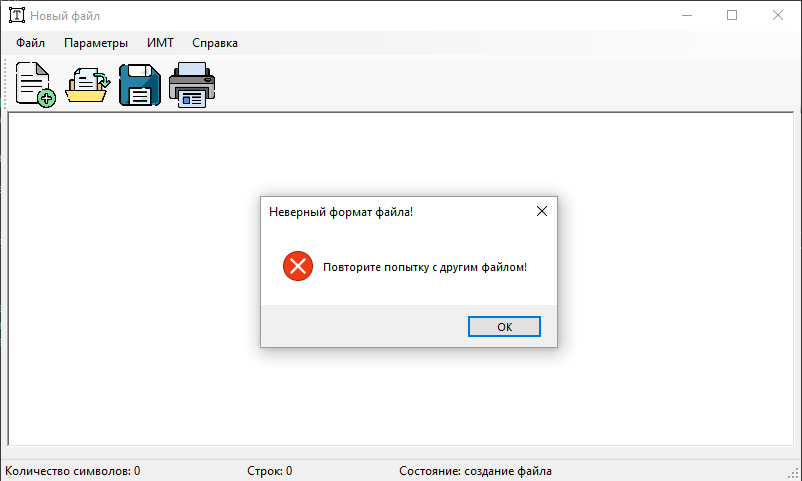


Рисунок 18. Сообщение об ошибке