**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 №4**

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

Специальность09.02.07 «Информационные системы и программирование»

Выполнили студенты 3 курса 32919/1 группы

Зориков Антон Алексеевич, Лейко Матвей Александрович

Преподаватель: Ильин Ю. П.

Санкт-Петербург

2023

**Cодержание**

[**Цель работы** 3](#_Toc118149827)

[**Задание** 3](#_Toc118149828)

[**Математическая формулировка методов** 3](#_Toc118149829)

[**Текст программных модулей** 7](#_Toc118149830)

[**Протокол испытаний** 11](#_Toc118149831)

[**Результаты выполнения работы программы** 13](#_Toc118149832)

**Тема: «Разработка класса по работе с текстовыми файлами»**

**Вариант №9**

# **Цель работы:**

Получить практические навыки создания программ обработки файлов, использования диалоговых компонентов, обобщить и расширить знания по разработке многомодульных приложений.

# **Задание:**

В текстовом файле хранятся результаты измерений артериального давления (верхнее и нижнее давление в одной строке через один пробел). Разработать программу, которая загружает в текстовый редактор файл по выбору пользователя и выполняет поиск наибольшего и наименьшего значения верхнего и нижнего давления, сравнивает эти значения с эталоном (нормы артериального давления 110-130/60-80) и выводит сообщение, какое давление повышено или понижено по отношению к норме.

# **Математическая формулировка методов:**

1. Метод для получения текста из файла

public static string GetTextFromFile(string path)

{

using (FileStream fstream = File.OpenRead(path))

{

byte[] array = new byte[fstream.Length];

fstream.Read(array, 0, array.Length);

string textFromFile = Encoding.Default.GetString(array);

return textFromFile;

}

}

Данный метод принимает в качестве аргумента путь к файлу и возвращает текст из него

1. Метод для сохранения файла

public void SaveFile(string content)

{

try

{

using (StreamWriter writer = new StreamWriter(path))

{

writer.Write(content);

}

}

catch (Exception ex)

{

throw new Exception($"Ошибка сохранения файла: {ex.Message}");

}

}

Данный метод принимает на вход строку который нужно сохранить в файле, при возникновении исключения например неправильно заданный путь сохранения оно будет обработано и выброшено новое исключение

1. Метод для проверки текста на корректность

public static bool IsValid(string text)

{

string[] arr = text.Split(new char[] { ' ' });

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

{

int index = arr[i].IndexOf('/');

if (index < 0) return false;

string upper\_str = arr[i].Substring(0, index);

int upper\_int;

bool is\_upper\_valid = int.TryParse(upper\_str, out upper\_int);

string down\_str = arr[i].Substring((index + 1), arr[i].Length - index - 1);

int down\_int;

bool is\_down\_valid = int.TryParse(down\_str, out down\_int);

if ((!is\_down\_valid || !is\_upper\_valid) || (upper\_int > 320 || down\_int > 230) || (upper\_int < 10 || down\_int < 10)) return false;

}

return true;

}

Данный метод принимает на вход строку которую следует проверить на корректность и возвращает значение истинна или ложь

1. Метод для определения минимума и максимума артериального давления

public static int[] MinMax(string text)

{

string[] measurements = text.Split(new char[] { ' ' });

int minUpper = int.MaxValue, maxUpper = int.MinValue;

int minDown = int.MaxValue, maxDown = int.MinValue;

if (!IsValid(text))

{

throw new Exception("Введенная строка не корректна");

}

foreach (string measurement in measurements)

{

string[] pressureValues = measurement.Split('/');

if (pressureValues.Length == 2)

{

if (int.TryParse(pressureValues[0], out int upperPressure) &&

int.TryParse(pressureValues[1], out int downPressure))

{

minUpper = Math.Min(minUpper, upperPressure);

maxUpper = Math.Max(maxUpper, upperPressure);

minDown = Math.Min(minDown, downPressure);

maxDown = Math.Max(maxDown, downPressure);

}

}

}

return new int[] { minUpper, maxUpper, minDown, maxDown };

}

Данный метод принимает на вход строку с записями об артериальном давлении и среди них находит максимальное и минимальное, верхнее и нижнее давление, возвращает их как массив целых чисел, возможен выброс исключения

1. Метод для сравнения артериального давления с нормой

public static string CheckPressure(int[] pressureValues)

{

int minUpperPressure = pressureValues[0];

int maxUpperPressure = pressureValues[1];

int minDownPressure = pressureValues[2];

int maxDownPressure = pressureValues[3];

if(minUpperPressure == int.MaxValue || maxUpperPressure == int.MinValue || minDownPressure == int.MaxValue || maxDownPressure == int.MinValue)

{

return "Недостаточно данных для анализа";

}

string minUpperMessage = CheckPressureRange(minUpperPressure, 110, 130);

string minDownMessage = CheckPressureRange(minDownPressure, 60, 80);

string maxUpperMessage = CheckPressureRange(maxUpperPressure, 110, 130);

string maxDownMessage = CheckPressureRange(maxDownPressure, 60, 80);

return $"Минимальное верхнее давление: {minUpperPressure} ({minUpperMessage}), " +

$"Максимальное верхнее давление: {maxUpperPressure} ({maxUpperMessage}), " +

$"Минимальное нижнее давление: {minDownPressure} ({minDownMessage}), " +

$"Максимальное нижнее давление: {maxDownPressure} ({maxDownMessage})";

}

Данный метод принимает на вход массив целых чисел, содержащий записи об артериальном давлении и сравнивает их с нормой, возвращает строку

1. Метод для сравнения давления с заданной нормой

private static string CheckPressureRange(int pressure, int minRange, int maxRange)

{

if (pressure < minRange)

{

return "Ниже нормы";

}

else if (pressure > maxRange)

{

return "Выше нормы";

}

else

{

return "В норме";

}

}

Данный метод получает давление и границы нормы после чего сравнивает значения и возвращает строку

1. Метод для печати на принтер

public void PrintToPrinter()

{

try

{

PrintDocument pd = new PrintDocument();

pd.PrintPage += new PrintPageEventHandler(PrintPage);

pd.Print();

}

catch (Exception ex)

{

throw new Exception($"Ошибка печати на принтер: {ex.Message}");

}

}

Данный метод производит печать текста на принтере в случаю ошибки выбрасывает исключение

1. Метод для печати страницы на принтере

private void PrintPage(object sender, PrintPageEventArgs e)

{

try

{

using (StreamReader reader = new StreamReader(path))

{

Font printFont = new Font("Arial", 10);

float yPos = 0f;

int count = 0;

string line = null;

while (count < e.MarginBounds.Height / printFont.GetHeight(e.Graphics) &&

((line = reader.ReadLine()) != null))

{

yPos = count \* printFont.GetHeight(e.Graphics);

e.Graphics.DrawString(line, printFont, Brushes.Black, 0, yPos, new StringFormat());

count++;

}

}

}

catch (Exception ex)

{

throw new Exception($"Ошибка печати страницы: {ex.Message}");

}

}

Данный метод выполняет настройку печати и печать текста на прнтере

1. Метод для открытия файла

public static string OpenFileAndGetContents(string filePath, out string status)

{

status = "Открытие документа";

try

{

using (StreamReader sr = new StreamReader(filePath, Encoding.GetEncoding(1251)))

{

string fileContents = sr.ReadToEnd();

status = string.Empty;

return fileContents;

}

}

catch (Exception ex)

{

status = $"Ошибка чтения файла.\n{ex.ToString()}";

return null;

}

}

Данный метод получает на вход путь к файлу и переменную статуса, которую изменяет в зависимости от успешности открытия, в случае ошибку выбрасывает исключение

**Текст программных модулей:**

// библиотечный модуль \*.dll

using System;

using System.Collections.Generic;

using System.IO;

using System.Drawing;

using System.Drawing.Printing;

using System.Text;

namespace ArterialPressure

{

public class ArterialPressureClass

{

private string path;

public string Path

{

get { return path; }

set { path = value; }

}

public static string GetTextFromFile(string path)

{

using (FileStream fstream = File.OpenRead(path))

{

byte[] array = new byte[fstream.Length];

fstream.Read(array, 0, array.Length);

string textFromFile = Encoding.Default.GetString(array);

return textFromFile;

}

}

public void SaveFile(string content)

{

try

{

using (StreamWriter writer = new StreamWriter(path))

{

writer.Write(content);

}

}

catch (Exception ex)

{

throw new Exception($"Ошибка сохранения файла: {ex.Message}");

}

}

public static bool IsValid(string text)

{

string[] arr = text.Split(new char[] { ' ' });

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

{

int index = arr[i].IndexOf('/');

if (index < 0) return false;

string upper\_str = arr[i].Substring(0, index);

int upper\_int;

bool is\_upper\_valid = int.TryParse(upper\_str, out upper\_int);

string down\_str = arr[i].Substring((index + 1), arr[i].Length - index - 1);

int down\_int;

bool is\_down\_valid = int.TryParse(down\_str, out down\_int);

if ((!is\_down\_valid || !is\_upper\_valid) || (upper\_int > 320 || down\_int > 230) || (upper\_int < 10 || down\_int < 10)) return false;

}

return true;

}

public static int[] MinMax(string text)

{

string[] measurements = text.Split(new char[] { ' ' });

int minUpper = int.MaxValue, maxUpper = int.MinValue;

int minDown = int.MaxValue, maxDown = int.MinValue;

if (!IsValid(text))

{

throw new Exception("Введенная строка не корректна");

}

foreach (string measurement in measurements)

{

string[] pressureValues = measurement.Split('/');

if (pressureValues.Length == 2)

{

if (int.TryParse(pressureValues[0], out int upperPressure) &&

int.TryParse(pressureValues[1], out int downPressure))

{

minUpper = Math.Min(minUpper, upperPressure);

maxUpper = Math.Max(maxUpper, upperPressure);

minDown = Math.Min(minDown, downPressure);

maxDown = Math.Max(maxDown, downPressure);

}

}

}

return new int[] { minUpper, maxUpper, minDown, maxDown };

}

//110-130/60-80)

public static string CheckPressure(int[] pressureValues)

{

int minUpperPressure = pressureValues[0];

int maxUpperPressure = pressureValues[1];

int minDownPressure = pressureValues[2];

int maxDownPressure = pressureValues[3];

if(minUpperPressure == int.MaxValue || maxUpperPressure == int.MinValue || minDownPressure == int.MaxValue || maxDownPressure == int.MinValue)

{

return "Недостаточно данных для анализа";

}

string minUpperMessage = CheckPressureRange(minUpperPressure, 110, 130);

string minDownMessage = CheckPressureRange(minDownPressure, 60, 80);

string maxUpperMessage = CheckPressureRange(maxUpperPressure, 110, 130);

string maxDownMessage = CheckPressureRange(maxDownPressure, 60, 80);

return $"Минимальное верхнее давление: {minUpperPressure} ({minUpperMessage}), " +

$"Максимальное верхнее давление: {maxUpperPressure} ({maxUpperMessage}), " +

$"Минимальное нижнее давление: {minDownPressure} ({minDownMessage}), " +

$"Максимальное нижнее давление: {maxDownPressure} ({maxDownMessage})";

}

private static string CheckPressureRange(int pressure, int minRange, int maxRange)

{

if (pressure < minRange)

{

return "Ниже нормы";

}

else if (pressure > maxRange)

{

return "Выше нормы";

}

else

{

return "В норме";

}

}

public void PrintToPrinter()

{

try

{

PrintDocument pd = new PrintDocument();

pd.PrintPage += new PrintPageEventHandler(PrintPage);

pd.Print();

}

catch (Exception ex)

{

throw new Exception($"Ошибка печати на принтер: {ex.Message}");

}

}

private void PrintPage(object sender, PrintPageEventArgs e)

{

try

{

using (StreamReader reader = new StreamReader(path))

{

Font printFont = new Font("Arial", 10);

float yPos = 0f;

int count = 0;

string line = null;

while (count < e.MarginBounds.Height / printFont.GetHeight(e.Graphics) &&

((line = reader.ReadLine()) != null))

{

yPos = count \* printFont.GetHeight(e.Graphics);

e.Graphics.DrawString(line, printFont, Brushes.Black, 0, yPos, new StringFormat());

count++;

}

}

}

catch (Exception ex)

{

throw new Exception($"Ошибка печати страницы: {ex.Message}");

}

}

public static string OpenFileAndGetContents(string filePath, out string status)

{

status = "Открытие документа";

try

{

using (StreamReader sr = new StreamReader(filePath, Encoding.GetEncoding(1251)))

{

string fileContents = sr.ReadToEnd();

status = string.Empty;

return fileContents;

}

}

catch (Exception ex)

{

status = $"Ошибка чтения файла.\n{ex.ToString()}";

return null;

}

}

}

}

//код программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using ArterialPressure;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

string inputData = "120/80 100/90";

try

{

int[] result = ArterialPressureClass.MinMax(inputData);

string pressureCheckResult = ArterialPressureClass.CheckPressure(result);

Console.WriteLine(pressureCheckResult);

}

catch (Exception x)

{

Console.WriteLine(x.Message);

}

Console.ReadLine();

}

}

}

// код формы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using ArterialPressure;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

// имя файла

private string fn = string.Empty;

private bool docChanged = false; // true - в текст внесены изменения

string status="";

ArterialPressureClass AP = new ArterialPressureClass();

public Form1() //конструктор формы

{

InitializeComponent();

textBox1.ScrollBars = ScrollBars.Vertical; //только вертикальная прокрутка

textBox1.Text = string.Empty; //очистить текст

this.Text = "NkEdit - Новый документ"; //заголовок формы

// отобразить панель инструментов

toolStrip1.Visible = true;

ParamToolStripMenuItem.Checked = true; //установить галочку на данном пункте меню (Панель инструментов)

openFileDialog1.DefaultExt = "txt";

openFileDialog1.Filter = "текст|\*.txt";

openFileDialog1.Title = "Открыть документ";

openFileDialog1.Multiselect = false;

// настройка компонента saveDialog1

saveFileDialog1.DefaultExt = "txt";

saveFileDialog1.Filter = "текст|\*.txt";

saveFileDialog1.Title = "Сохранить документ";

}

private int SaveDocument()

{

status = "Сохранение документа";

statusStrip1.Items[2].Text = "Состояние: " + status;

int result = 0;

if (fn == string.Empty)

{

// отобразить диалог Сохранить

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

// отобразить имя файла в заголовке окна

fn = saveFileDialog1.FileName;

this.Text = fn;

}

else result = -1;

}

// сохранить файл

if (fn != string.Empty)

{

try

{

// получим информацию о файле fn

System.IO.FileInfo fi =

new System.IO.FileInfo(fn);

// поток для записи (перезаписываем файл)

System.IO.StreamWriter sw = fi.CreateText();

// записываем данные

sw.Write(textBox1.Text);

// закрываем поток

sw.Close();

result = 0;

}

catch (Exception exc)

{

MessageBox.Show(exc.ToString(), "NkEdit",

MessageBoxButtons.OK,

MessageBoxIcon.Error);

}

}

statusStrip1.Items[2].Text = "Состояние: ";

return result;

}

// выбор в меню Файл команды Создать

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

{

status = "Создание документа";

statusStrip1.Items[2].Text = "Состояние: " + status;

if (docChanged)

{

DialogResult dr;

dr = MessageBox.Show("Сохранить изменения ?", "NkEdit",

MessageBoxButtons.YesNoCancel,

MessageBoxIcon.Warning);

switch (dr)

{

case DialogResult.Yes:

if (SaveDocument() == 0)

{

textBox1.Clear();

docChanged = false;

}

break;

case DialogResult.No:

textBox1.Clear();

docChanged = false;

break;

case DialogResult.Cancel:

break;

}

}

statusStrip1.Items[2].Text = "Состояние: ";

}

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

{

status = "Открытие документа";

statusStrip1.Items[2].Text = "Состояние: " + status;

openFileDialog1.FileName = string.Empty;

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

{

fn = openFileDialog1.FileName;

AP.Path = fn;

this.Text = fn;

string fileContents = ArterialPressureClass.OpenFileAndGetContents(fn, out status);

if (fileContents != null)

{

textBox1.Text = fileContents;

textBox1.SelectionStart = textBox1.TextLength;

}

else

{

MessageBox.Show("Не удалось открыть файл.", "MEdit",

MessageBoxButtons.OK,

MessageBoxIcon.Error);

}

}

statusStrip1.Items[2].Text = "Состояние: " + status;

}

// выбор в меню Файл команды Сохранить

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

{

if (ArterialPressureClass.IsValid(textBox1.Text))

{

try {

AP.SaveFile(textBox1.Text);

}

catch(Exception ex) {

SaveFileDialog saveFileDialog = new SaveFileDialog();

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

saveFileDialog.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

DialogResult result = saveFileDialog.ShowDialog();

if (result == DialogResult.OK)

{

string filePath = saveFileDialog.FileName;

AP.Path = filePath;

if (ArterialPressureClass.IsValid(textBox1.Text))

AP.SaveFile(textBox1.Text);

else

MessageBox.Show("Сохраняемые данные не корректны, в файле должны находиться только записи об артериальном виде через пробел (120/90 110/80)");

}

}

}

else

MessageBox.Show("Сохраняемые данные не корректны, в файле должны находиться только записи об артериальном виде через пробел (120/90 110/80)");

}

// выбор в меню Файл команды Выход

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

{

this.Close();

}

// выбор в меню Параметры команды Панель инструментов

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

{

// отобразить/скрыть панель инструментов

toolStrip1.Visible = ! toolStrip1.Visible;

ParamToolStripMenuItem.Checked = ! ParamToolStripMenuItem.Checked;

}

// выбор в меню Параметры команды Шрифт

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

{

status = "Выбор шрифта";

statusStrip1.Items[2].Text = "Состояние: " + status;

fontDialog1.Font = textBox1.Font;

if (fontDialog1.ShowDialog() == DialogResult.OK)

{

textBox1.Font = fontDialog1.Font;

}

statusStrip1.Items[2].Text = "Состояние: ";

}

//обработка события TextChanged

private void textBox1\_TextChanged(object sender, EventArgs e)

{

docChanged = true;

int len = textBox1.Text.Length; // кол-во символов в поле редактирования

statusStrip1.Items[0].Text = "Число знаков: " + len.ToString("D");

int lines = textBox1.Lines.Length;

statusStrip1.Items[1].Text = "Число строк: " + lines.ToString("D");

status = "Редактирование документа";

statusStrip1.Items[2].Text = "Состояние: " + status;

}

// пользователь сделал щелчок на кнопке "Закрыть окно"

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

{

if (docChanged )

{

DialogResult dr;

dr = MessageBox.Show("Сохранить изменения?", "NkEdit",

MessageBoxButtons.YesNoCancel,

MessageBoxIcon.Warning);

switch ( dr )

{

case DialogResult.Yes :

status = "Сохранение документа";

if ( SaveDocument() != 0)

// пользователь отменил операцию сохранения файла

e.Cancel = true; // отменить закрытие окна программы

break;

case DialogResult.No: ;

break;

case DialogResult.Cancel:

// отменить закрытие окна программы

e.Cancel = true;

break;

}

}

}

// выбор в меню Справка команды О программе

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

{

status = "Справка о программе";

statusStrip1.Items[2].Text = "Состояние: " + status;

Form2 about = new Form2();

about.ShowDialog();

statusStrip1.Items[2].Text = "Состояние: ";

}

// выбор в меню Файл команды Печать

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

{

status = "Печать документа";

statusStrip1.Items[2].Text = "Состояние: " + status;

try

{

AP.PrintToPrinter();

} catch {

MessageBox.Show("Пожалуйста сначала откройте файл");

}

statusStrip1.Items[2].Text = "Состояние: ";

}

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

{

try

{

MessageBox.Show(ArterialPressureClass.CheckPressure(ArterialPressureClass.MinMax(textBox1.Text)));

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

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

saveFileDialog.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

DialogResult result = saveFileDialog.ShowDialog();

if (result == DialogResult.OK)

{

string filePath = saveFileDialog.FileName;

AP.Path = filePath;

if (ArterialPressureClass.IsValid(textBox1.Text))

AP.SaveFile(textBox1.Text);

else

MessageBox.Show("Сохраняемые данные не корректны, в файле должны находиться только записи об артериальном виде через пробел (120/90 110/80)");

}

}

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

{

}

}

}

# **Протокол испытаний:**

*Таблица 1.* Протокол испытаний

|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Фактические результаты** |
| Корректные данные | string inputData = "120/80 100/90";  try  {  int[] result = ArterialPressureClass.MinMax(inputData);  string pressureCheckResult = ArterialPressureClass.CheckPressure(result);  Console.WriteLine(pressureCheckResult);  }  catch(Exception x)  {  Console.WriteLine(x.Message);  }      Console.ReadLine(); | Сообщение c выводом классификации давления. | Минимальное верхнее давление: 100 (Ниже нормы), Максимальное верхнее давление: 120 (В норме), Минимальное нижнее давление: 80 (В норме), Максимальное нижнее давление: 90 (Выше нормы) |
| Проверка неверного ввода давления. | string inputData = "упуп/80";  try  {  int[] result = ArterialPressureClass.MinMax(inputData);  string pressureCheckResult = ArterialPressureClass.CheckPressure(result);  Console.WriteLine(pressureCheckResult);  }  catch(Exception x)  {  Console.WriteLine(x.Message);  }      Console.ReadLine(); | Сообщение об ошибке “Введенная строка не корректна” | Введенная строка не корректна |
| Проверка ввода слишком высокого давления. | static void Main(string[] args)  {  string inputData = "1102913/80";  try  {  int[] result = ArterialPressureClass.MinMax(inputData);  string pressureCheckResult = ArterialPressureClass.CheckPressure(result);  Console.WriteLine(pressureCheckResult);  }  catch(Exception x)  {  Console.WriteLine(x.Message);  }      Console.ReadLine(); {  Console.WriteLine(x.Message);  }      Console.ReadLine(); | Сообщение об ошибке “Введенная строка не корректна” |  |
| Проверка ввода слишком низкого давления. | static void Main(string[] args)  {  string inputData = "120/1";  try  {  int[] result = ArterialPressureClass.MinMax(inputData);  string pressureCheckResult = ArterialPressureClass.CheckPressure(result);  Console.WriteLine(pressureCheckResult);  }  catch(Exception x)  {  Console.WriteLine(x.Message);  }      Console.ReadLine(); | Сообщение об ошибке “Введенная строка не корректна” |  |

**Результаты выполнения работы программы:**

1. Ввод корректного значений для давления. (рис.1).



рис.1 Вывод класификаций давлений из исходных данных

1. Ввод некорректных значений в давления. (рис.2)



рис.2Сообщение об ошибке “Введенная строка не корректна”

1. Ввод слишком большого давления. (рис.3)



рис.3Сообщение об ошибке “Введенная строка не корректна”

1. Ввод слишком низкого давления. (рис.4)



рис.4Сообщение об ошибке “Введенная строка не корректна”.

**Форма:**

1. Запуск формы

При запуске формы на экране появилась возможность взаимодействия с элементами интерфейса

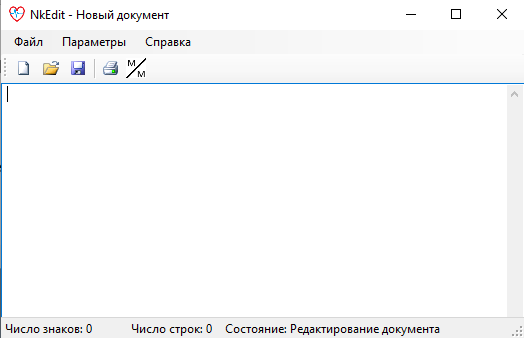


рис.5 запуск формы

1. Возможность ввода данных

Мы можем вводить данные

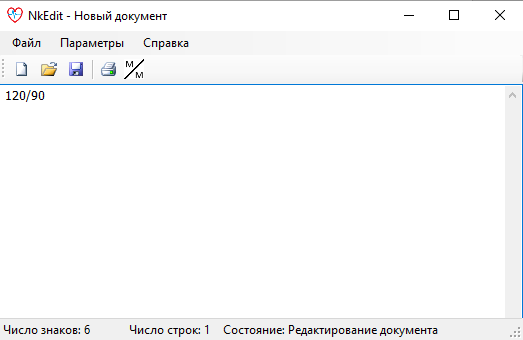


рис.6 данные в форме

1. Возможность создания нового документа

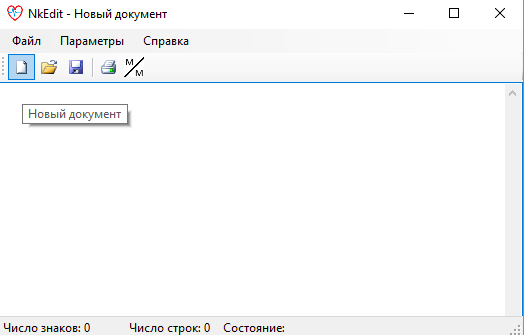


рис.7 новый документ

1. Возможность открытия документа

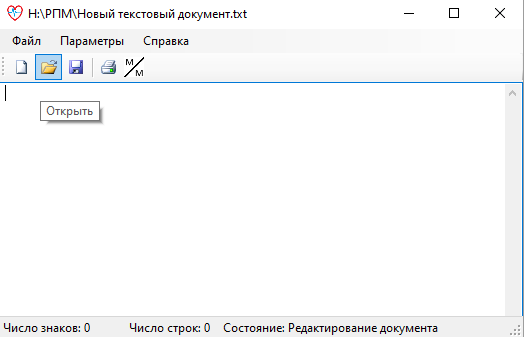


рис.8 открытие документа

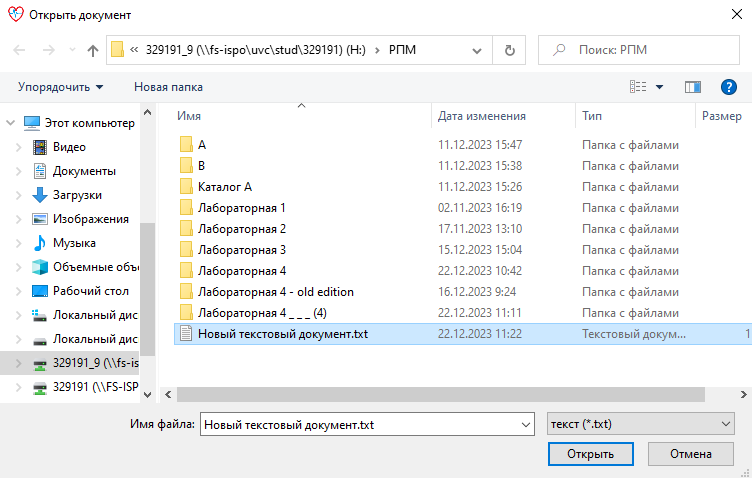


рис.9 открытие документа в проводнике

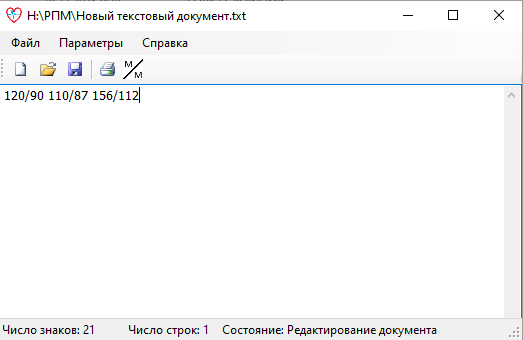


рис.9 открытый файл в форме

1. Возможность сохранения документа

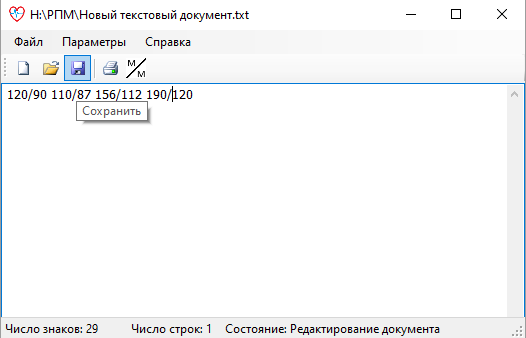


рис.9 сохранение файла

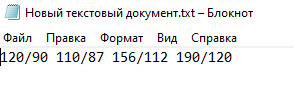


рис.10 измененный файл после сохранения

1. Возможность печати документа

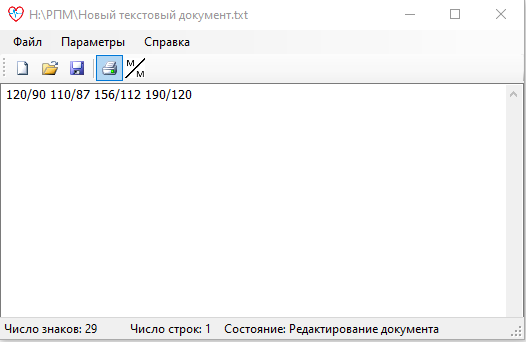


рис.11 кнопка печати документа

1. Возможность вывода максимального и минимального верхнего и нижнего давления

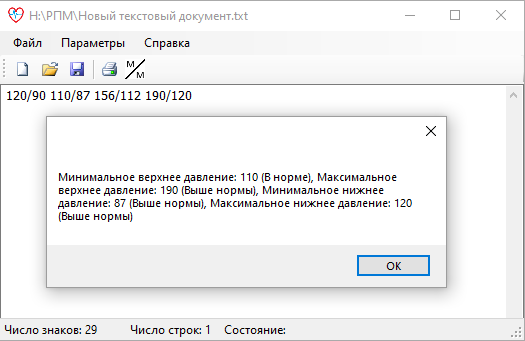


рис.11 максимальное и минимальное верхнее и нижнее давление

1. Окно справки

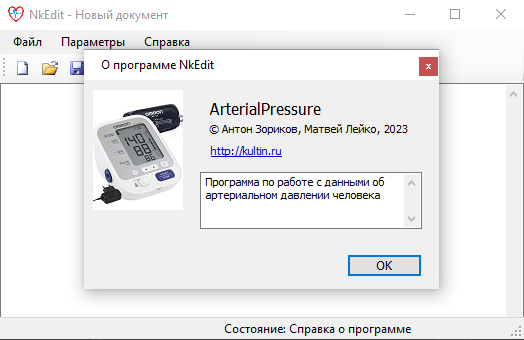


рис.12 окно справки

1. Возможность изменения шрифта

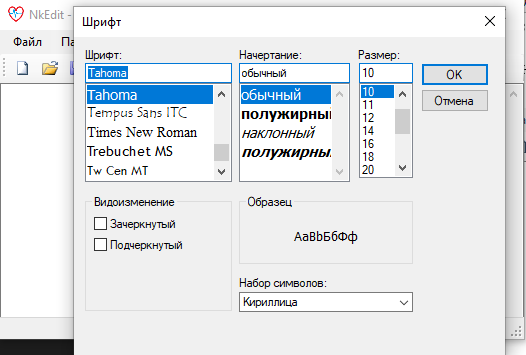


рис.13 окно выбора шрифтов

1. Возможность сохранения файла как

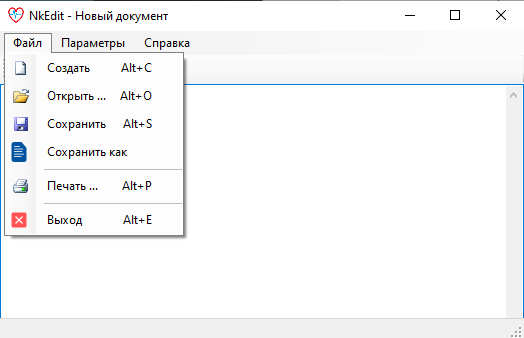


рис.14 меню файлов

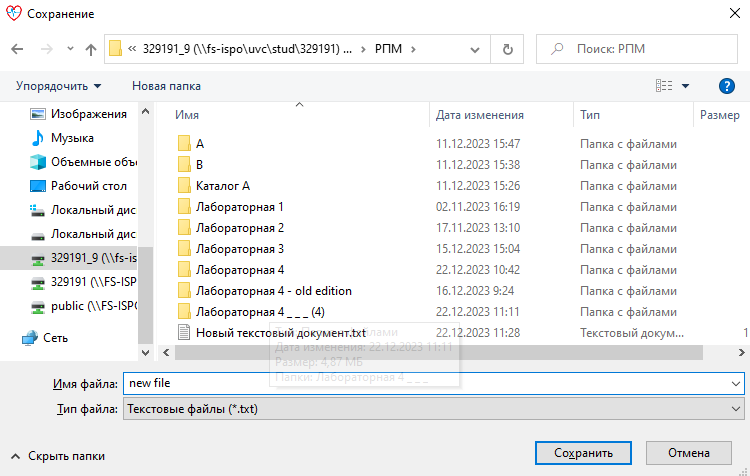


рис.15 выбор пути сохранения файла через проводник

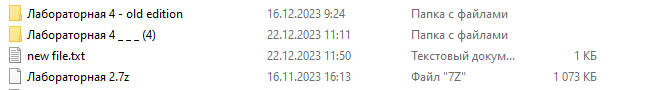


рис.16 файл успешно сохранен