**МИНИСТЕРСТВО ТРАНСПОРТА**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ**

**АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО**

**ОБРАЗОВАНИЯ**

**“РОССИЙСКИЙ УНИВЕРСИТЕТ**

**ТРАНСПОРТА” (МИИТ)**

**Институт транспортной техники и систем**

**управления**

**(ИТТСУ)**

**Кафедра “Управление и защита информации”**

**Тема: “Лабораторная работа №5”**

**Выполнили студенты группы ТКИ-342**

**Лазухин Максим Сергеевич**

**Макаров Григорий Борисович**

**Специальность 10.05.01 “Компьютерная безопасность”**

**Преподаватель: к.т.н. Сафронов А.И.**

**Москва**

**2024**

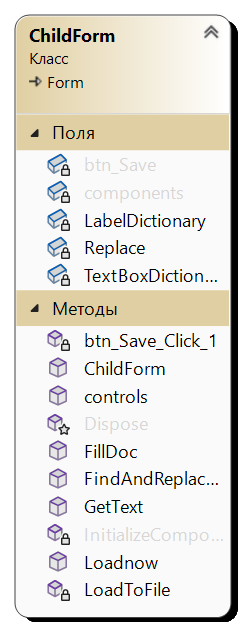
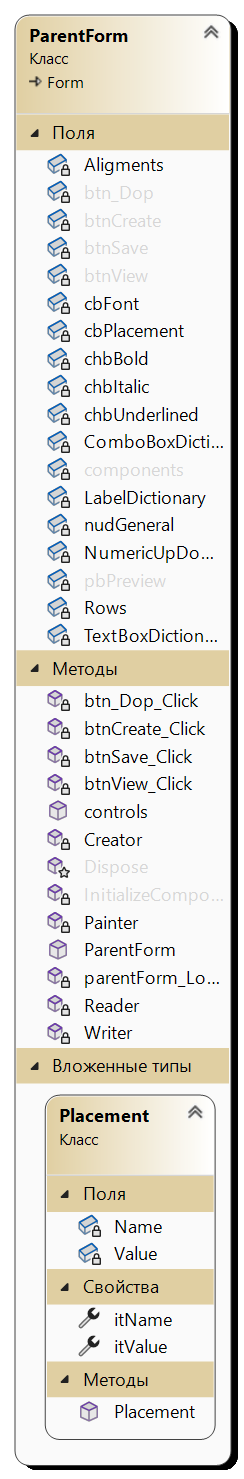
1. Цель работы:

Освоить навыки подключения внешних модулей (библиотек классов) и их использования в разрабатываемых программах; освоить навыки программируемой настройки параметров документов *Microsoft Office Word*.

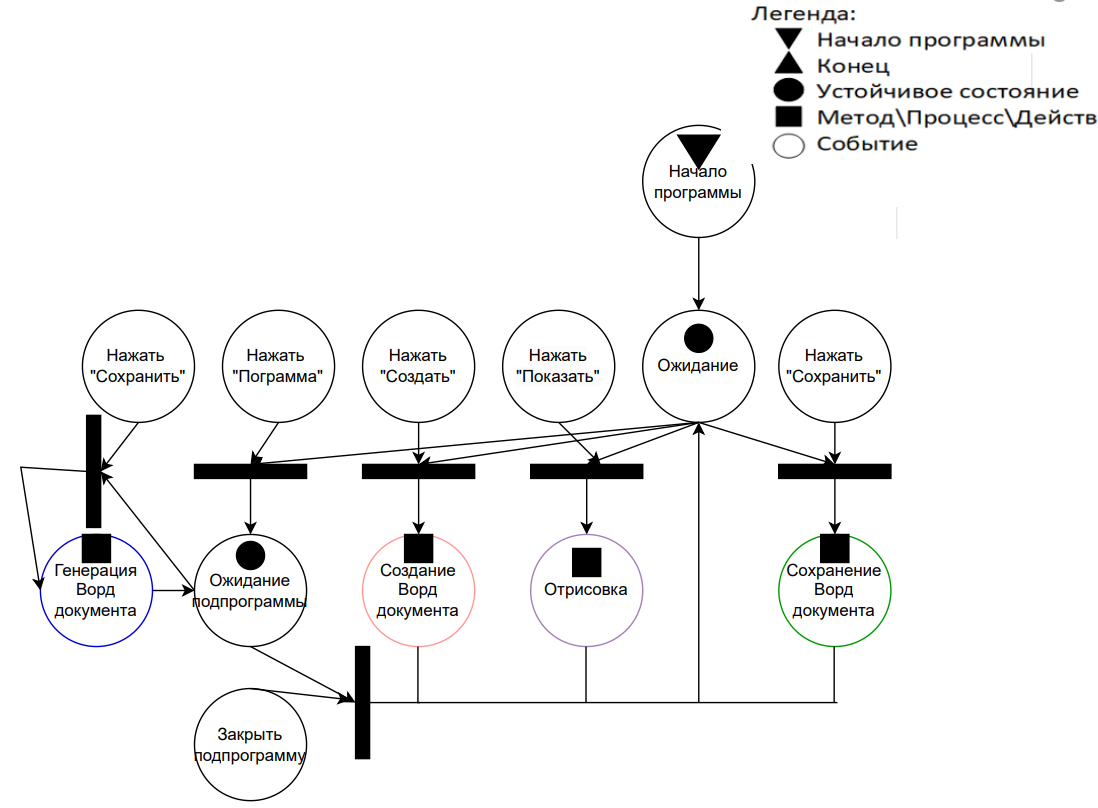
2. Задача:

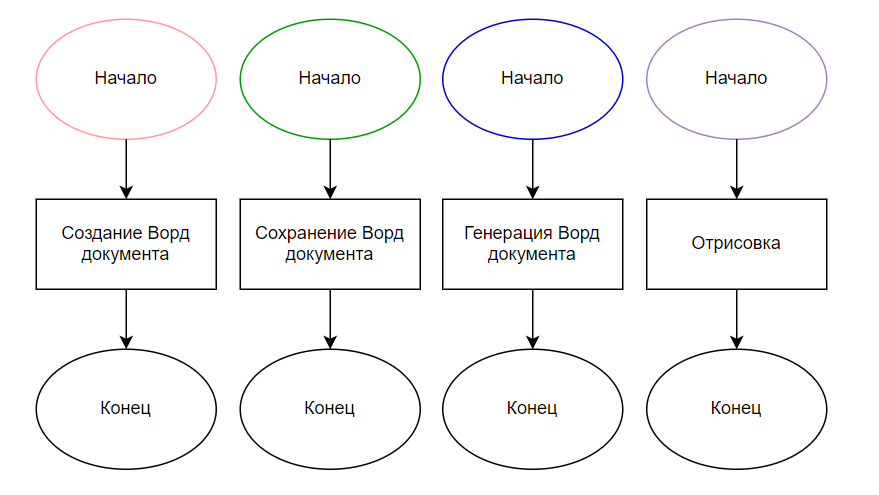
В интегрированной среде разработки *Microsoft Visual Studio* составить программу в режиме *Windows Forms Application* на языке *Visual C#*, представляющую собой средство взаимодействия с текстовым файлом, необходимым для настройки формата надписей, заносимых в документ *Microsoft Office Word.*

3. Диаграмма классов:

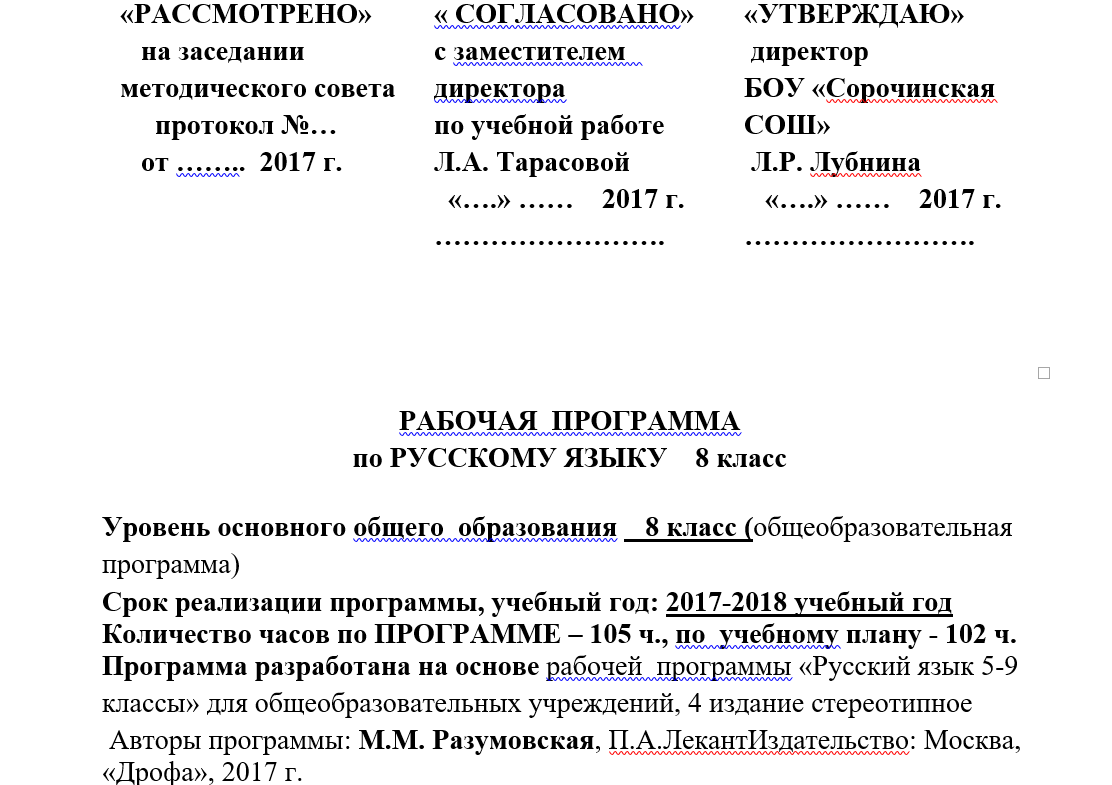


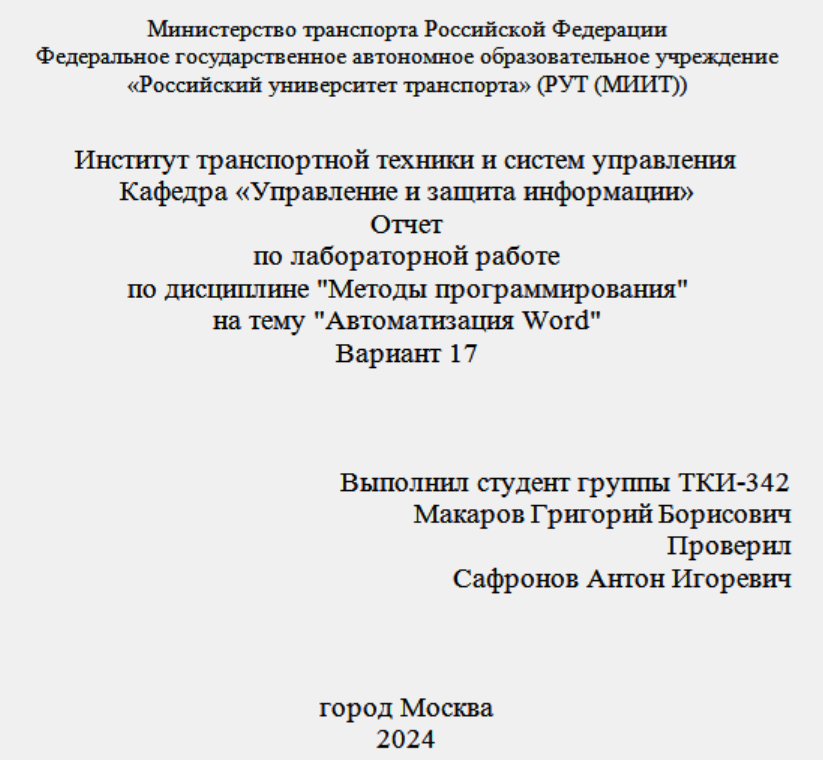
4. Сеть Петри



5. Схемы алгоритмов методов:   


6. Подбор тестовых примеров





7. Листинг кода:

**using System;**

**using System.Collections;**

**using System.Collections.Generic;**

**using System.Drawing;**

**using W = Microsoft.Office.Interop.Word;**

**using System.Windows.Forms;**

**using System.IO;**

**namespace WordLab**

**{**

**public partial class ParentForm : Form**

**{**

**Dictionary<string, Label> LabelDictionary = new Dictionary<string, Label>();**

**Dictionary<string, TextBox> TextBoxDictionary = new Dictionary<string, TextBox>();**

**Dictionary<string, NumericUpDown> NumericUpDownDictionary = new Dictionary<string, NumericUpDown>();**

**Dictionary<string, ComboBox> ComboBoxDictionary = new Dictionary<string, ComboBox>();**

**NumericUpDown nudGeneral = new NumericUpDown();**

**ComboBox cbFont = new ComboBox();**

**CheckBox chbBold = new CheckBox();**

**CheckBox chbItalic = new CheckBox();**

**CheckBox chbUnderlined = new CheckBox();**

**ComboBox cbPlacement = new ComboBox();**

**string[] Aligments = { "1", "1", "1", "1", "1", "1", "1", "2", "2", "2", "2", "1", "1" };**

**string[] Rows = { "Vedomstvo", "Regalii", "Vuz", "Institut", "Kafedra", "Document", "Rabota", "Distsiplina", "Tema", "Nomer", "Vipolneno", "Avtor", "Proveril", "Prepodovatel", "Gorod", "God" };**

**public ParentForm()**

**{**

**InitializeComponent();**

**}**

**private void parentForm\_Load(object sender, EventArgs e)**

**{**

**controls();**

**Reader();**

**}**

**public void controls()**

**{**

**Label Labelgenerator(int x, int y, string label)**

**{**

**Label example = new Label();**

**example.Location = new Point(x, y);**

**example.Text = label;**

**example.AutoSize = true;**

**example.Font = new Font("Times New Roman", 10);**

**example.ForeColor = Color.Black;**

**return example;**

**}**

**void Labelfiller(int counter)**

**{**

**int x = 50;**

**int y = 15;**

**for (int i = 0; i < counter; i++)**

**{**

**string baseName = "Label";**

**string label = $"{Rows[i]}";**

**string dynamicVariableName = $"{baseName}{Rows[i]}";**

**LabelDictionary[dynamicVariableName] = Labelgenerator(x, y, label);**

**Controls.Add(LabelDictionary[dynamicVariableName]);**

**y = LabelDictionary[dynamicVariableName].Bottom + 10;**

**}**

**}**

**Labelfiller(16);**

**TextBox TextBoxgenerator(int x, int y)**

**{**

**TextBox example = new TextBox();**

**example.Location = new Point(x, y);**

**example.Size = new Size(100, 20);**

**example.Font = new Font("Times New Roman", 10);**

**example.ForeColor = Color.Black;**

**return example;**

**}**

**void TextBoxfiller(int counter)**

**{**

**int x = LabelDictionary["LabelVedomstvo"].Right + 30;**

**int y = 15;**

**for (int i = 0; i < counter; i++)**

**{**

**string baseName = "TextBox";**

**string dynamicVariableName = $"{baseName}{Rows[i]}";**

**TextBoxDictionary[dynamicVariableName] = TextBoxgenerator(x, y);**

**Controls.Add(TextBoxDictionary[dynamicVariableName]);**

**y = LabelDictionary[$"Label{Rows[i]}"].Bottom + 10;**

**}**

**}**

**TextBoxfiller(16);**

**NumericUpDown NumericUpDowngenerator(int x, int y)**

**{**

**NumericUpDown example = new NumericUpDown();**

**example.Location = new Point(x, y);**

**example.Value = 14;**

**example.Size = new Size(40, 20);**

**return example;**

**}**

**void NumericUpDownfiller(int counter)**

**{**

**int x = TextBoxDictionary["TextBoxVedomstvo"].Right + 30;**

**int y = LabelDictionary[$"Label{Rows[2]}"].Bottom + 10;**

**for (int i = 3; i < counter; i++)**

**{**

**string baseName = "NumericUpDown";**

**string dynamicVariableName = $"{baseName}{Rows[i]}";**

**NumericUpDownDictionary[dynamicVariableName] = NumericUpDowngenerator(x, y);**

**Controls.Add(NumericUpDownDictionary[dynamicVariableName]);**

**y = LabelDictionary[$"Label{Rows[i]}"].Bottom + 10;**

**}**

**}**

**NumericUpDownfiller(16);**

**ComboBox ComboBoxgenerator(int x, int y)**

**{**

**ComboBox example = new ComboBox();**

**example.Location = new Point(x, y);**

**example.Size = new Size(120, 20);**

**example.Items.Add("Times New Roman");**

**example.Items.Add("Calibri");**

**example.SelectedIndex = 0;**

**return example;**

**}**

**void ComboBoxfiller(int counter)**

**{**

**int x = NumericUpDownDictionary["NumericUpDownGorod"].Right + 30;**

**int y = LabelDictionary[$"Label{Rows[2]}"].Bottom + 10;**

**for (int i = 3; i < counter; i++)**

**{**

**string baseName = "ComboBox";**

**string dynamicVariableName = $"{baseName}{Rows[i]}";**

**ComboBoxDictionary[dynamicVariableName] = ComboBoxgenerator(x, y);**

**Controls.Add(ComboBoxDictionary[dynamicVariableName]);**

**y = LabelDictionary[$"Label{Rows[i]}"].Bottom + 10;**

**}**

**}**

**ComboBoxfiller(16);**

**nudGeneral = NumericUpDowngenerator(TextBoxDictionary["TextBoxVedomstvo"].Right + 30, LabelDictionary[$"Label{Rows[0]}"].Bottom + 10);**

**Controls.Add(nudGeneral);**

**cbFont = ComboBoxgenerator(nudGeneral.Right + 30, LabelDictionary[$"Label{Rows[0]}"].Bottom + 10);**

**Controls.Add(cbFont);**

**CheckBox CheckBoxgenerator(int x, int y, string text)**

**{**

**CheckBox example = new CheckBox();**

**example.Location = new Point(x, y);**

**example.AutoSize = true;**

**example.Text = text;**

**return example;**

**}**

**chbBold = CheckBoxgenerator(cbFont.Right + 30, LabelDictionary[$"Label{Rows[0]}"].Bottom + 10, "Bold");**

**Controls.Add(chbBold);**

**chbItalic = CheckBoxgenerator(cbFont.Right + 30, LabelDictionary[$"Label{Rows[1]}"].Bottom + 10, "Italic");**

**Controls.Add(chbItalic);**

**chbUnderlined = CheckBoxgenerator(cbFont.Right + 30, 15, "Underlined");**

**Controls.Add(chbUnderlined);**

**cbPlacement = ComboBoxgenerator(chbBold.Right + 30, LabelDictionary[$"Label{Rows[0]}"].Bottom + 10);**

**cbPlacement.Items.Clear();**

**ArrayList Placements = new ArrayList();**

**Placements.Add(new Placement("Left", "0"));**

**Placements.Add(new Placement("Right", "2"));**

**Placements.Add(new Placement("Center", "1"));**

**cbPlacement.DataSource = Placements;**

**cbPlacement.DisplayMember = "itName";**

**cbPlacement.ValueMember = "itValue";**

**Controls.Add(cbPlacement);**

**}**

**void Writer()**

**{**

**using (StreamWriter sw = new StreamWriter("Setter.txt"))**

**{**

**for (int i = 0; i < 3; i++)**

**{**

**sw.WriteLine(TextBoxDictionary[$"TextBox{Rows[i]}"].Text + ";" + nudGeneral.Value + ";" + cbFont.Text + ";" + chbBold.Checked + ";" + chbItalic.Checked + ";" + chbUnderlined.Checked + ";" + Convert.ToInt32(cbPlacement.SelectedValue) + ";");**

**}**

**for (int i = 3; i < 16; i++)**

**{**

**sw.WriteLine(TextBoxDictionary[$"TextBox{Rows[i]}"].Text + ";" + NumericUpDownDictionary[$"NumericUpDown{Rows[i]}"].Value + ";" + ComboBoxDictionary[$"ComboBox{Rows[i]}"].Text + ";");**

**}**

**}**

**}**

**void Creator()**

**{**

**W.Application app = new W.Application();**

**W.Document doc = app.Documents.Add(Visible: true);**

**doc = app.Documents.Add();**

**StreamReader sr = new StreamReader("Setter.txt");**

**string line;**

**string[] pieces;**

**for (int i = 0; i < 3; i++)**

**{**

**line = sr.ReadLine();**

**pieces = line.Split(';');**

**app.Selection.Font.Name = pieces[2];**

**app.Selection.Font.Size = int.Parse(pieces[1]);**

**app.Selection.Font.Bold = Convert.ToInt32(Convert.ToBoolean(pieces[3]));**

**app.Selection.Font.Italic = Convert.ToInt32(Convert.ToBoolean(pieces[4]));**

**if (pieces[4] == "True")**

**{**

**app.Selection.Font.Underline = W.WdUnderline.wdUnderlineSingle;**

**}**

**app.Selection.ParagraphFormat.Alignment = (W.WdParagraphAlignment)int.Parse(pieces[6]);**

**app.Selection.TypeText(pieces[0]);**

**app.Selection.TypeParagraph();**

**}**

**app.Selection.TypeText("\n");**

**app.Selection.TypeText("\n");**

**for (int i = 5; i < 18; i++)**

**{**

**if (i == 12 || i == 16)**

**{**

**app.Selection.TypeText("\n");**

**app.Selection.TypeText("\n");**

**app.Selection.TypeText("\n");**

**app.Selection.TypeText("\n");**

**app.Selection.TypeText("\n");**

**}**

**app.Selection.ParagraphFormat.Alignment = (W.WdParagraphAlignment)Convert.ToInt32(Aligments[i - 5]);**

**app.Selection.Font.Underline = W.WdUnderline.wdUnderlineNone;**

**app.Selection.Font.Bold = 0;**

**app.Selection.Font.Italic = 0;**

**line = sr.ReadLine();**

**pieces = line.Split(';');**

**app.Selection.Font.Name = pieces[2];**

**app.Selection.Font.Size = int.Parse(pieces[1]);**

**app.Selection.TypeText(pieces[0]);**

**app.Selection.TypeParagraph();**

**}**

**sr.Close();**

**doc.Save();**

**app.Quit();**

**}**

**void Reader()**

**{**

**using (StreamReader sr = new StreamReader("Setter.txt"))**

**{**

**for (int i = 0; i < 3; i++)**

**{**

**string temp = sr.ReadLine();**

**int split = temp.IndexOf(";");**

**TextBoxDictionary[$"TextBox{Rows[i]}"].Text = temp.Substring(0, split);**

**temp = temp.Substring(split + 1, temp.Length - split - 1);**

**split = temp.IndexOf(";");**

**nudGeneral.Value = Convert.ToInt32(temp.Substring(0, split));**

**temp = temp.Substring(split + 1, temp.Length - split - 1);**

**split = temp.IndexOf(";");**

**cbFont.Text = temp.Substring(0, split);**

**temp = temp.Substring(split + 1, temp.Length - split - 1);**

**split = temp.IndexOf(";");**

**chbBold.Checked = Convert.ToBoolean(temp.Substring(0, split));**

**temp = temp.Substring(split + 1, temp.Length - split - 1);**

**split = temp.IndexOf(";");**

**chbItalic.Checked = Convert.ToBoolean(temp.Substring(0, split));**

**temp = temp.Substring(split + 1, temp.Length - split - 1);**

**split = temp.IndexOf(";");**

**chbUnderlined.Checked = Convert.ToBoolean(temp.Substring(0, split));**

**temp = temp.Substring(split + 1, temp.Length - split - 1);**

**split = temp.IndexOf(";");**

**cbPlacement.SelectedItem = temp.Substring(0, split);**

**}**

**for (int i = 3; i < 16; i++)**

**{**

**string temp = sr.ReadLine();**

**int split = temp.IndexOf(";");**

**TextBoxDictionary[$"TextBox{Rows[i]}"].Text = temp.Substring(0, split);**

**temp = temp.Substring(split + 1, temp.Length - split - 1);**

**split = temp.IndexOf(";");**

**NumericUpDownDictionary[$"NumericUpDown{Rows[i]}"].Value = Convert.ToInt32(temp.Substring(0, split));**

**temp = temp.Substring(split + 1, temp.Length - split - 1);**

**split = temp.IndexOf(";");**

**ComboBoxDictionary[$"ComboBox{Rows[i]}"].Text = temp.Substring(0, split);**

**}**

**sr.Close();**

**}**

**}**

**void Painter(object sender, PaintEventArgs e)**

**{**

**Graphics p = e.Graphics;**

**float y = 10;**

**StringFormat Placement = new StringFormat();**

**Placement.Alignment = (StringAlignment)Convert.ToInt32(cbPlacement.SelectedValue);**

**for (int i = 0; i < 3; i++)**

**{**

**string text = TextBoxDictionary[$"TextBox{Rows[i]}"].Text;**

**int size = Convert.ToInt32(nudGeneral.Value);**

**System.Drawing.Font font = new System.Drawing.Font(cbFont.Text, size);**

**RectangleF rect = new RectangleF(10, y, pbPreview.Width - 10, font.GetHeight(p));**

**p.DrawString(text, font, Brushes.Black, rect, Placement);**

**y += font.GetHeight(p);**

**}**

**for (int i = 3; i < 5; i++)**

**{**

**int size = 10;**

**System.Drawing.Font font = new System.Drawing.Font(cbFont.Text, size);**

**RectangleF rect = new RectangleF(10, y, pbPreview.Width - 10, font.GetHeight(p));**

**p.DrawString("\n", font, Brushes.Black, rect, Placement);**

**y += font.GetHeight(p);**

**}**

**for (int i = 5; i < 18; i++)**

**{**

**Placement.Alignment = (StringAlignment)Convert.ToInt32(Aligments[i - 5]);**

**string text = TextBoxDictionary[$"TextBox{Rows[i-2]}"].Text;**

**int size = Convert.ToInt32(NumericUpDownDictionary[$"NumericUpDown{Rows[i-2]}"].Value);**

**System.Drawing.Font font = new System.Drawing.Font($"{ComboBoxDictionary[$"ComboBox{Rows[i-2]}"].Text}", size);**

**RectangleF rect = new RectangleF(10, y, pbPreview.Width - 10, font.GetHeight(p));**

**p.DrawString(text, font, Brushes.Black, rect, Placement);**

**y += font.GetHeight(p);**

**if (i == 11 || i == 15)**

**{**

**for ( int i1 = 0; i1 < 3; i1++)**

**{**

**p.DrawString("\n", font, Brushes.Black, rect, Placement);**

**y += font.GetHeight(p);**

**}**

**}**

**}**

**}**

**private void btnCreate\_Click(object sender, EventArgs e)**

**{**

**Creator();**

**}**

**private void btnSave\_Click(object sender, EventArgs e)**

**{**

**Writer();**

**}**

**private void btnView\_Click(object sender, EventArgs e)**

**{**

**pbPreview.Paint += Painter;**

**pbPreview.Invalidate();**

**}**

**public class Placement**

**{**

**private string Name;**

**private string Value;**

**public Placement(string strName, string strValue)**

**{**

**this.Name = strName;**

**this.Value = strValue;**

**}**

**public string itName**

**{**

**get**

**{**

**return Name;**

**}**

**}**

**public string itValue**

**{**

**get**

**{**

**return Value;**

**}**

**}**

**}**

**private void btn\_Dop\_Click(object sender, EventArgs e)**

**{**

**ChildForm Child = new ChildForm();**

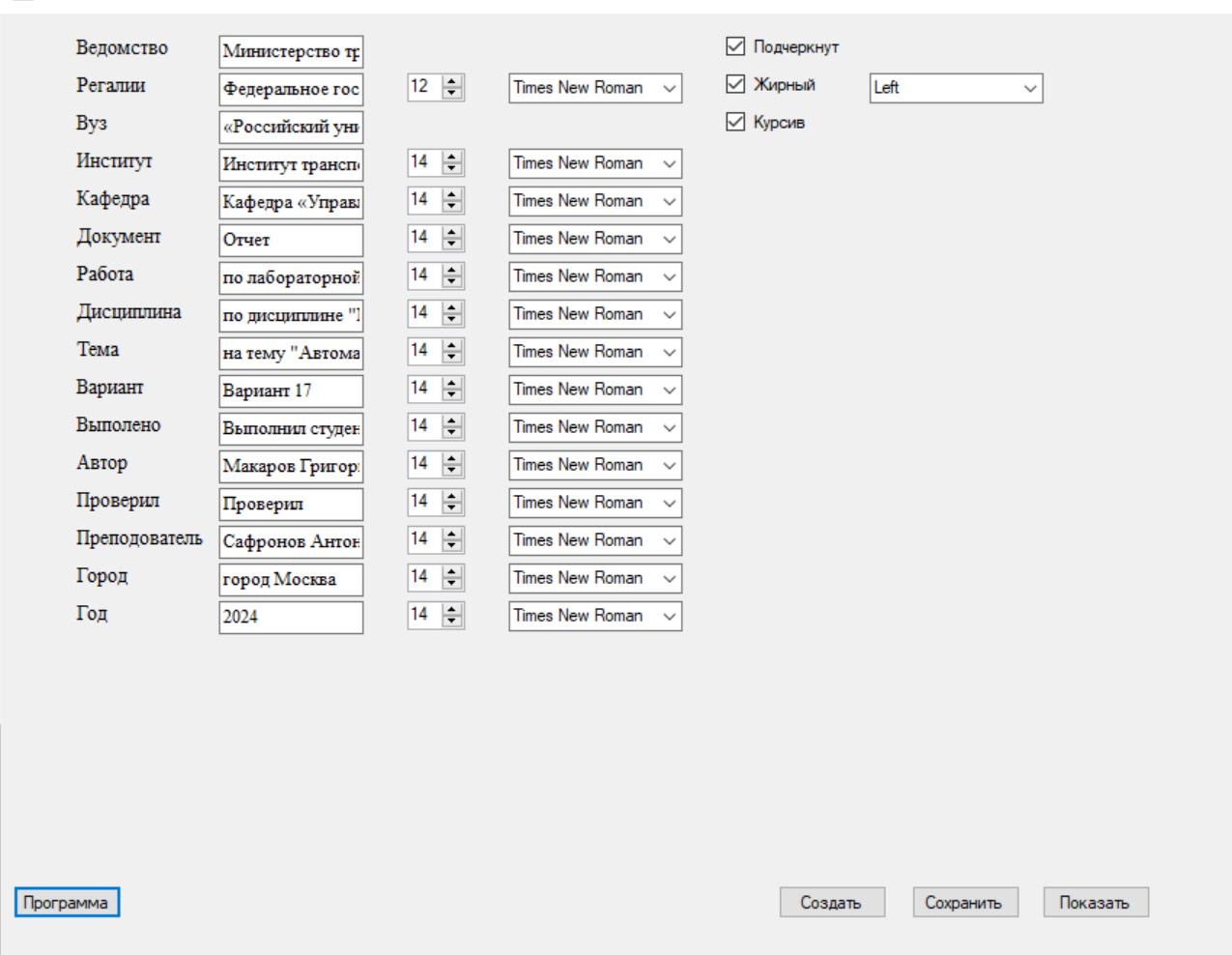
**Child.Show();**

**}**

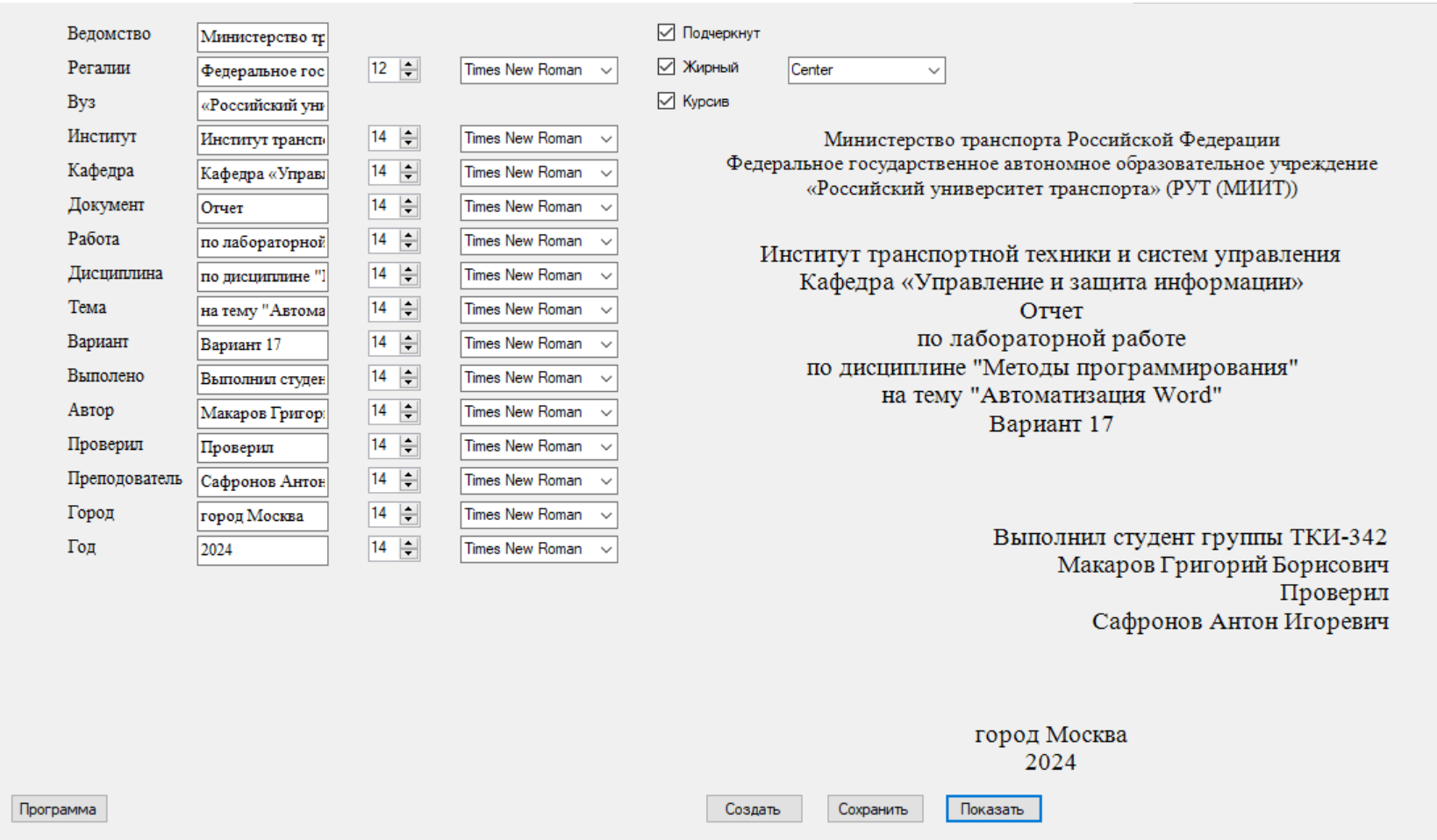
**}**

**}**

1. **Графический пользовательский интерфейс**

****

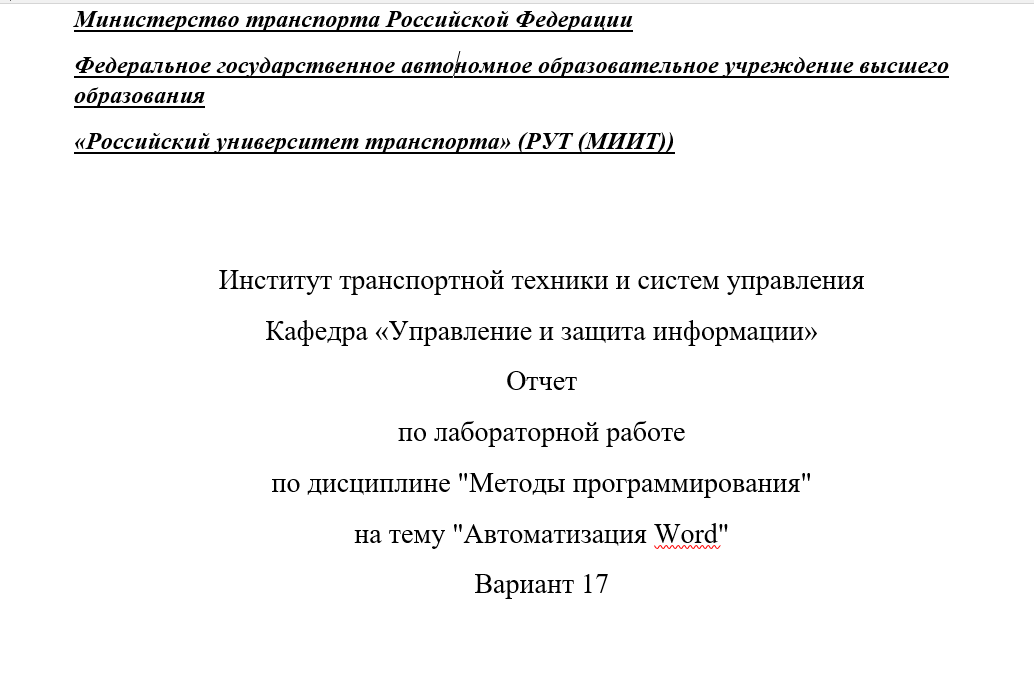
ГПИ основной страницы

****

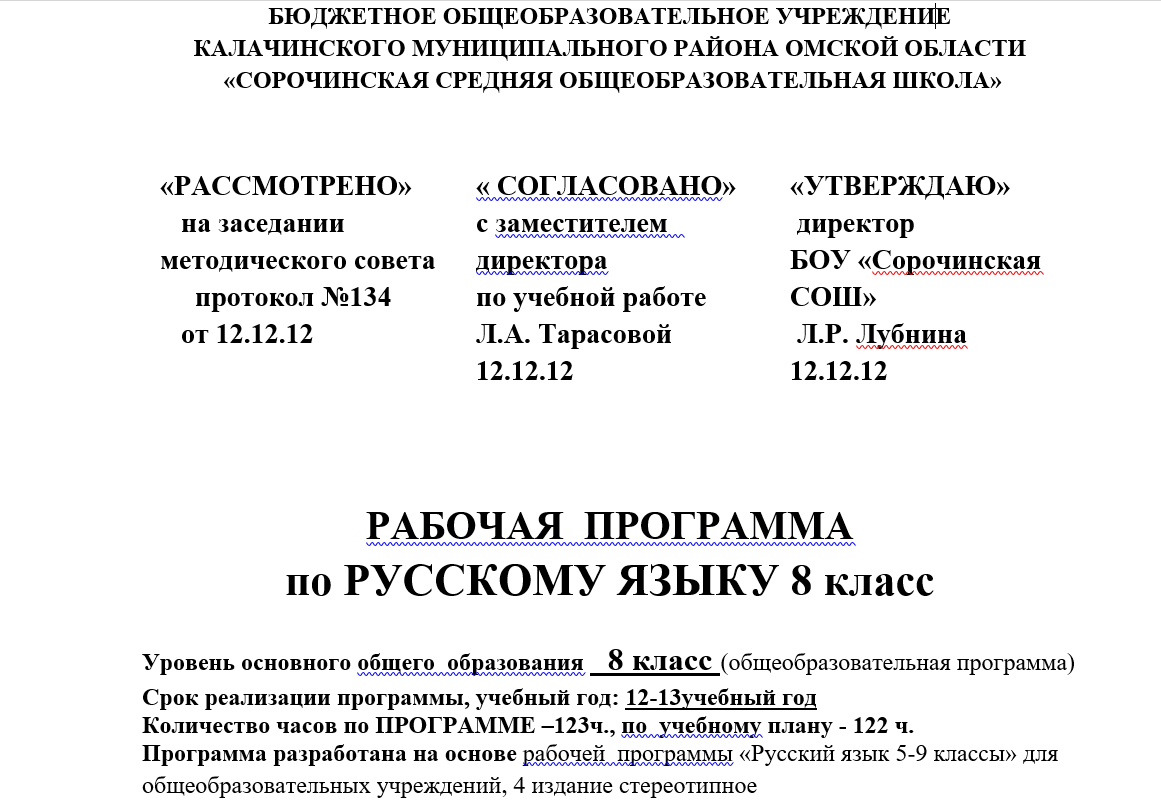
ГПИ основной страницы после нажатия кнопки «Показать»

****

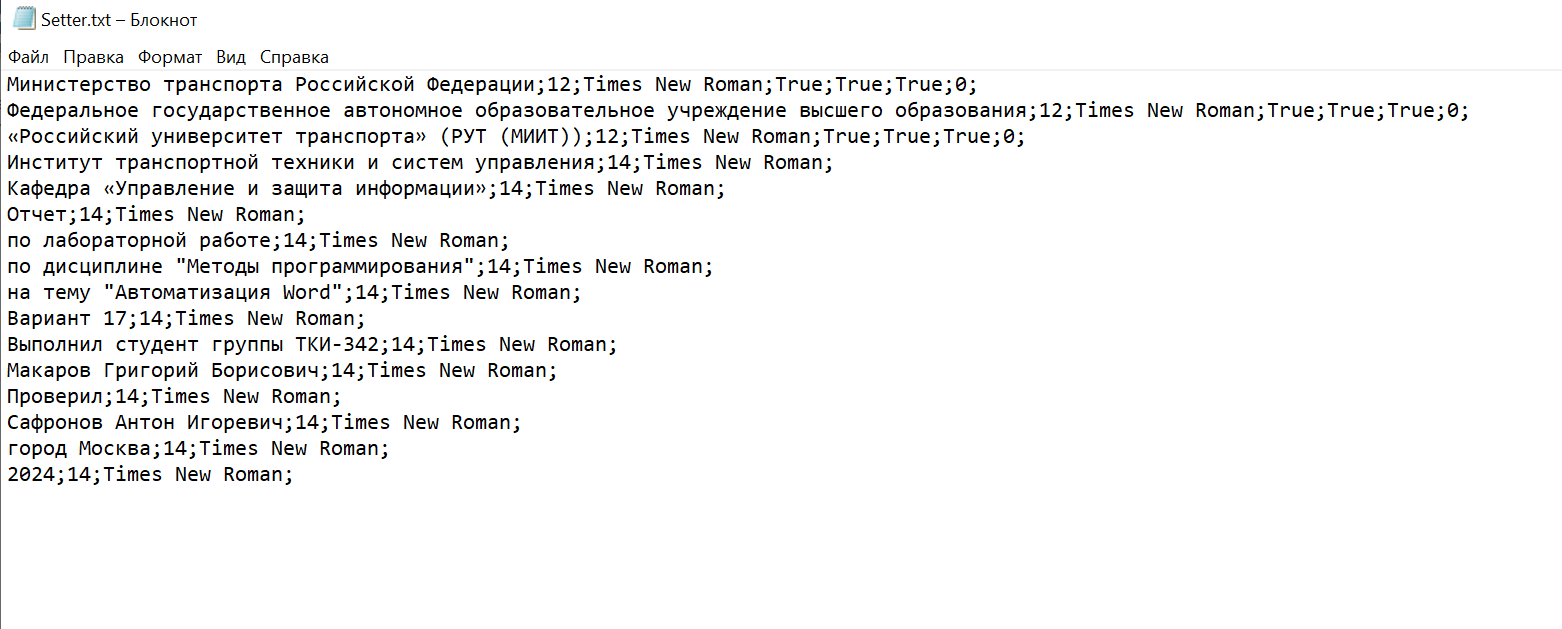
ГПИ подпрограммы



Часть документа сгенерированного программой



Часть документа сгенерированного подпрограммой



Содержимое файла Setter.txt.

1. **Вывод**

В ходе выполнения задания были освоении навыки подключения внешних модулей и библиотек классов, а также навыки программной настройки параметров электронных таблиц Microsoft Office Word. Был разработан графический интерфейс с функционалом создания, заполнения и предпросмотра Word файлов по заданному шаблону. В процессе разработки была решена задача по имитации динамической генерации переменных на языке c#, не поддерживающем такую функцию. Имитация производится с помощью структуры данных «Словарь» путем генерации ключей и вызова псевдосгенерированных переменных только через эти ключи.