**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**

**Савин Лев Андреевич   
Потапов Артём Константинович**

**Специальность 10.05.01 “Компьютерная безопасность”**

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

**Москва**

**2024**

1. Формулировка цели работы:

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. Описание задачи:

Создать и настроить графический пользовательский интерфейс, взаимодействующий с текстовым файлом (*\*.txt / \*.csv*), а также с модулем (библиотекой классов) текстового редактора *Microsoft Office Word*.

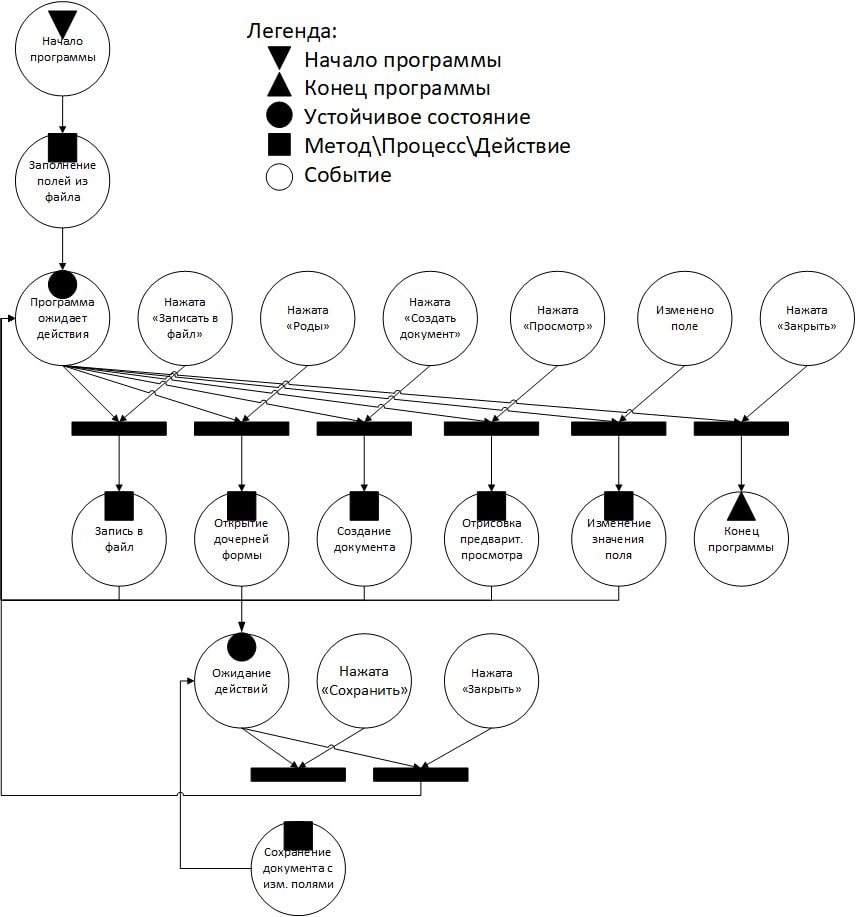
При загрузке графического пользовательского интерфейса все параметры из файла считываются в строковый массив, если файл не пуст, и изменяют состояние интерфейсных элементов управления, размещённых на графическом пользовательском интерфейсе. Если файл пуст, то все интерфейсные элементы управления графического пользовательского интерфейса остаются в исходном, нулевом состоянии.

3. Составление диаграммы классов:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

4. Составление сети Петри запрограммированного технологического процесса:



5. Составление схем алгоритмов методов:

Изображение выглядит как диаграмма, круг, снимок экрана, дизайн

Автоматически созданное описание

Настройка необходимых параметров

6. Подбор тестовых примеров:

- Нажатие кнопки "Начать работу с приложением" и активация доступа к элементам.

- Получение справочной информации.

- Закрытие приложения.

7. Листинг (код) программного обеспечения:

public partial class FrmMain : Form

{

string setupFileName = System.Windows.Forms.Application.StartupPath + "\\setup.txt";

string headerFileName = System.Windows.Forms.Application.StartupPath;

public FrmMain()

{

InitializeComponent();

}

public void GenPregnantForm()

{

FrmPregnant frmPregnant = new FrmPregnant();

frmPregnant.Owner = this;

frmPregnant.ShowDialog();

}

public string WriteDoc()

{

Application app = new Application();

Document doc = app.Documents.Add();

WriteParagraph(tbDepartment, nudDepartment, chbBDepartment, chbCDepartment, cbDepartment, doc, 1);

WriteParagraph(tbRegalias, nudRegalias, chbBRegalias, chbCRegalias, cbRegalias, doc, 2);

WriteParagraph(tbUniversityName, nudUniversityName, chbBUniversityName, chbCUniversityName, cbUniversityName, doc, 3);

WriteParagraph(tbUniversityName, nudUniversityName, chbBUniversityName, chbCUniversityName, cbUniversityName, doc, 4);

WriteParagraph(tbUniversityName, nudUniversityName, chbBUniversityName, chbCUniversityName, cbUniversityName, doc, 5);

InsertTableLine(doc, 4);

WriteParagraph(tbInstituteName, nudInstituteName, chbBInstituteName, chbCInstituteName, cbInstituteName, doc, 6);

WriteParagraph(tbDepName, nudDepName, chbBDepName, chbCDepName, cbDepName, doc, 7);

WriteParagraph(cbDocType, nudDocType, chbBDocType, chbCDocType, cbDocTypeTwo, doc, 8);

WriteParagraph(tbConcatDocWork, nudConcatDocWork, chbBConcatDocWork, chbCConcatDocWork, cbConcatDocWork, doc, 9);

WriteParagraph(cbWorkType, nudWorkType, chbBWorkType, chbCWorkType, cbWorkTypeTwo, doc, 10);

WriteParagraph(cbNumber, nudNumber, chbBNumber, chbCNumber, cbNumberTwo, doc, 11);

WriteParagraph(tbConcatWorkDiscipline, nudConcatWorkDiscipline, chbBConcatWorkDiscipline, chbCConcatWorkDiscipline, cbConcatWorkDiscipline, doc, 12);

WriteParagraph(tbDiscipline, nudDiscipline, chbBDiscipline, chbCDiscipline, cbDiscipline, doc, 13);

WriteParagraph(tbConcatDiscTheme, nudConcatDiscTheme, chbBConcatDiscTheme, chbCConcatDiscTheme, cbConcatDiscTheme, doc, 14);

WriteParagraph(tbTheme, nudTheme, chbBTheme, chbCTheme, cbTheme, doc, 15);

WriteParagraph(tbMadeBy, nudMadeBy, chbBMadeBy, chbCMadeBy, cbMadeBy, doc, 16);

WriteParagraph(tbCreatorName, nudCreatorName, chbBCreatorName, chbCCreatorName, cbCreatorName, doc, 17);

WriteParagraph(tbVariant, nudVariant, chbBVariant, chbCVariant, cbVariant, doc, 18);

WriteParagraph(tbRevisoredBy, nudRevisoredBy, chbBRevisoredBy, chbCRevisoredBy, cbRevisoredBy, doc, 19);

WriteParagraph(tbRevisorName, nudRevisorName, chbBRevisorName, chbCRevisorName, cbRevisorName, doc, 20);

WriteParagraph(tbCity, nudCity, chbBCity, chbCCity, cbCity, doc, 21);

WriteParagraph(tbYear, nudYear, chbBYear, chbCYear, cbYear, doc, 22);

string trueFilename = headerFileName + DateTime.Now.ToString("yyyyMMdd") + " Header.docx";

app.ActiveDocument.SaveAs2(headerFileName + DateTime.Now.ToString("yyyyMMdd") + " Header.docx");

doc.Close();

app.Quit();

return trueFilename;

}  
 public void WriteParagraph(Control text, NumericUpDown size, CheckBox bold, CheckBox italic, Control allign, Document document, int index)

{

\_ = document.Paragraphs.Add();

Paragraph paragraph = document.Paragraphs[index];

paragraph.Range.Text = GetText(text);

paragraph.Range.Font.Name = "Times New Roman";

paragraph.Range.Font.Size = GetSize(size);

paragraph.Range.Font.Bold = Convert.ToInt32(GetBool(bold));

paragraph.Range.Font.Italic = Convert.ToInt32(GetBool(italic));

switch (GetAllign(allign))

{

case 0:

paragraph.Range.ParagraphFormat.Alignment = WdParagraphAlignment.wdAlignParagraphLeft;

break;

case 1:

paragraph.Range.ParagraphFormat.Alignment = WdParagraphAlignment.wdAlignParagraphCenter;

break;

case 2:

paragraph.Range.ParagraphFormat.Alignment = WdParagraphAlignment.wdAlignParagraphRight;

break;

case 3:

paragraph.Range.ParagraphFormat.Alignment = WdParagraphAlignment.wdAlignParagraphJustify;

break;

default:

paragraph.Range.ParagraphFormat.Alignment = WdParagraphAlignment.wdAlignParagraphCenter;

break;

}

switch (index)

{

case 7:

case 15:

case 20:

paragraph.SpaceAfter = 100;

break;

default:

paragraph.SpaceAfter = 1;

break;

}

}

public void InsertTableLine(Document document, int index)

{

\_ = document.Paragraphs.Add();

Paragraph paragraph = document.Paragraphs[index];

Table table = document.Tables.Add(paragraph.Range, 1, 1);

table.AllowAutoFit = true;

table.PreferredWidthType = WdPreferredWidthType.wdPreferredWidthAuto;

Cell cell = table.Range.Cells[1];

cell.Range.Font.Size = 1;

Borders borders = cell.Borders;

borders[WdBorderType.wdBorderBottom].LineStyle = WdLineStyle.wdLineStyleSingle;

borders[WdBorderType.wdBorderBottom].LineWidth = WdLineWidth.wdLineWidth075pt;

borders[WdBorderType.wdBorderLeft].LineStyle = WdLineStyle.wdLineStyleNone;

borders[WdBorderType.wdBorderRight].LineStyle = WdLineStyle.wdLineStyleNone;

borders[WdBorderType.wdBorderTop].LineStyle = WdLineStyle.wdLineStyleNone;

}

public string GetText(Control cntrl)

{

return cntrl.Text;

}

public int GetSize(NumericUpDown nud)

{

return Convert.ToInt32(nud.Value);

}

public bool GetBool(CheckBox checkBox)

{

return checkBox.Checked;

}  
 public int GetAllign(Control cntrl)

{

switch (cntrl.Text)

{

case "Left":

return 0;

case "Center":

return 1;

case "Right":

return 2;

case "Justified":

return 3;

default:

return 0;

}

}

public void WriteToFile(string filePath)

{

using (StreamWriter writer = new StreamWriter(filePath))

{

WriteLinesToStream(writer, tbDepartment, nudDepartment, chbBDepartment, chbCDepartment, cbDepartment);

WriteLinesToStream(writer, tbRegalias, nudRegalias, chbBRegalias, chbCRegalias, cbRegalias);

WriteLinesToStream(writer, tbUniversityName, nudUniversityName, chbBUniversityName, chbCUniversityName, cbUniversityName);

WriteLinesToStream(writer, tbInstituteName, nudInstituteName, chbBInstituteName, chbCInstituteName, cbInstituteName);

WriteLinesToStream(writer, tbDepName, nudDepName, chbBDepName, chbCDepName, cbDepName);

WriteLinesToStream(writer, tbConcatDocWork, nudConcatDocWork, chbBConcatDocWork, chbCConcatDocWork, cbConcatDocWork);

WriteLinesToStream(writer, tbConcatWorkDiscipline, nudConcatWorkDiscipline, chbBConcatWorkDiscipline, chbCConcatWorkDiscipline, cbConcatWorkDiscipline);

WriteLinesToStream(writer, tbDiscipline, nudDiscipline, chbBDiscipline, chbCDiscipline, cbDiscipline);

WriteLinesToStream(writer, tbConcatDiscTheme, nudConcatDiscTheme, chbBConcatDiscTheme, chbCConcatDiscTheme, cbConcatDiscTheme);

WriteLinesToStream(writer, tbTheme, nudTheme, chbBTheme, chbCTheme, cbTheme);

WriteLinesToStream(writer, tbMadeBy, nudMadeBy, chbBMadeBy, chbCMadeBy, cbMadeBy);

WriteLinesToStream(writer, tbCreatorName, nudCreatorName, chbBCreatorName, chbCCreatorName, cbCreatorName);

WriteLinesToStream(writer, tbVariant, nudVariant, chbBVariant, chbCVariant, cbVariant);

WriteLinesToStream(writer, tbRevisoredBy, nudRevisoredBy, chbBRevisoredBy, chbCRevisoredBy, cbRevisoredBy);

WriteLinesToStream(writer, tbRevisorName, nudRevisorName, chbBRevisorName, chbCRevisorName, cbRevisorName);

WriteLinesToStream(writer, tbCity, nudCity, chbBCity, chbCCity, cbCity);

WriteLinesToStream(writer, tbYear, nudYear, chbBYear, chbCYear, cbYear);

WriteLinesToStream(writer, cbDocType, nudDocType, chbBDocType, chbCDocType, cbDocTypeTwo);

WriteLinesToStream(writer, cbWorkType, nudWorkType, chbBWorkType, chbCWorkType, cbWorkTypeTwo);

WriteLinesToStream(writer, cbNumber, nudNumber, chbBNumber, chbCNumber, cbNumberTwo);

}

}  
 private void WriteLinesToStream(StreamWriter writer, params Control[] cntrls)

{

foreach (var control in cntrls)

{

if (control is TextBox textBox)

{

writer.Write(textBox.Text + ";");

}

else if (control is NumericUpDown numericUpDown)

{

writer.Write(numericUpDown.Text + ";");

}

else if (control is CheckBox checkBox)

{

writer.Write(checkBox.Checked.ToString() + ";");

}

else if (control is ComboBox comboBox)

{

writer.Write(comboBox.Text + ";");

}

}

writer.WriteLine();

}

public void ReadFromFile()

{

List<TextBox> lines = new List<TextBox>

{

tbDepartment,

tbRegalias,

tbUniversityName,

tbInstituteName,

tbDepName,

tbConcatDocWork,

tbConcatWorkDiscipline,

tbDiscipline,

tbConcatDiscTheme,

tbTheme,

tbMadeBy,

tbCreatorName,

tbVariant,

tbRevisoredBy,

tbRevisorName,

tbCity,

tbYear

};

int i = 0;

foreach (var item in lines)

{

item.Text = GetParamFromLine(0, ReadIndexedLine(i));

++i;

}

List<NumericUpDown> nud = new List<NumericUpDown>

{

nudDepartment,

nudRegalias,

nudUniversityName,

nudInstituteName,

nudDepName,

nudConcatDocWork,

nudConcatWorkDiscipline,

nudDiscipline,

nudConcatDiscTheme,

nudTheme,

nudMadeBy,

nudCreatorName,

nudVariant,

nudRevisoredBy,

nudRevisorName,

nudCity,

nudYear,

nudDocType,

nudWorkType,

nudNumber

};

i = 0;

foreach (var item in nud)

{

item.Text = GetParamFromLine(1, ReadIndexedLine(i));

++i;

}

List<CheckBox> chbB = new List<CheckBox>

{

chbBDepartment,

chbBRegalias,

chbBUniversityName,

chbBInstituteName,

chbBDepName,

chbBConcatDocWork,

chbBConcatWorkDiscipline,

chbBDiscipline,

chbBConcatDiscTheme,

chbBTheme,

chbBMadeBy,

chbBCreatorName,

chbBVariant,

chbBRevisoredBy,

chbBRevisorName,

chbBCity,

chbBYear,

chbBDocType,

chbBWorkType,

chbBNumber

};

i = 0;

foreach (var item in chbB)

{

item.Checked = Convert.ToBoolean(GetParamFromLine(2, ReadIndexedLine(i)));

++i;

}

List<CheckBox> chbC = new List<CheckBox>

{

chbCDepartment,

chbCRegalias,

chbCUniversityName,

chbCInstituteName,

chbCDepName,

chbCConcatDocWork,

chbCConcatWorkDiscipline,

chbCDiscipline,

chbCConcatDiscTheme,

chbCTheme,

chbCMadeBy,

chbCCreatorName,

chbCVariant,

chbCRevisoredBy,

chbCRevisorName,

chbCCity,

chbCYear,

chbCDocType,

chbCWorkType,

chbCNumber

};

i = 0;

foreach (var item in chbC)

{

item.Checked = Convert.ToBoolean(GetParamFromLine(3, ReadIndexedLine(i)));

++i;

}  
 List<ComboBox> cb = new List<ComboBox>

{

cbDepartment,

cbRegalias,

cbUniversityName,

cbInstituteName,

cbDepName,

cbConcatDocWork,

cbConcatWorkDiscipline,

cbDiscipline,

cbConcatDiscTheme,

cbTheme,

cbMadeBy,

cbCreatorName,

cbVariant,

cbRevisoredBy,

cbRevisorName,

cbCity,

cbYear,

cbDocTypeTwo,

cbWorkTypeTwo,

cbNumberTwo

};

i = 0;

foreach (var item in cb)

{

item.Text = GetParamFromLine(4, ReadIndexedLine(i));

++i;

}

cb.Clear();

cb.Add(cbDocType);

cb.Add(cbWorkType);

cb.Add(cbNumber);

i -= 3;

foreach (var item in cb)

{

item.Text = GetParamFromLine(0, ReadIndexedLine(i));

++i;

}

return;

}

public string ReadIndexedLine(int index)

{

if (File.Exists(setupFileName))

{

StreamReader streamReader = new StreamReader(setupFileName);

for (int i = 0; i < index; ++i)

{

streamReader.ReadLine();

}

string tempLine = streamReader.ReadLine();

streamReader.Close();

return tempLine;

}

return null;

}

public string GetParamFromLine(int paramIndex, string line)

{

string[] values = line.Split(';');

if (paramIndex >= 0 && paramIndex < values.Length)

return values[paramIndex];

return null;

}

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

{

WriteToFile(setupFileName);

}

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

{

WriteDoc();

}

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

{

pbPreview.Paint += DrawMultipleLines;

pbPreview.Invalidate();

}  
 public void DrawMultipleLines(object sender, PaintEventArgs e)

{

PictureBox pictureBox = sender as PictureBox;

Graphics g = e.Graphics;

System.Drawing.Font font = new System.Drawing.Font("Times New Roman", 12);

string[] lines = {

GetText(tbDepartment), GetText(tbRegalias), GetText(tbUniversityName), GetText(tbInstituteName), GetText(tbDepName),

GetText(cbDocType), GetText(tbConcatDocWork), GetText(cbWorkType), GetText(cbNumber), GetText(tbConcatWorkDiscipline),

GetText(tbDiscipline), GetText(tbConcatDiscTheme), GetText(tbTheme), GetText(tbMadeBy), GetText(tbCreatorName),

GetText(tbVariant), GetText(tbRevisoredBy), GetText(tbRevisorName), GetText(tbCity), GetText(tbYear)

};

StringFormat[] stringFormats =

{

new StringFormat() { Alignment = GetAllignPB(cbDepartment)},

new StringFormat() { Alignment = GetAllignPB(cbRegalias)},

new StringFormat() { Alignment = GetAllignPB(cbUniversityName)},

new StringFormat() { Alignment = GetAllignPB(cbInstituteName)},

new StringFormat() { Alignment = GetAllignPB(cbDepName)},

new StringFormat() { Alignment = GetAllignPB(cbDocTypeTwo)},

new StringFormat() { Alignment = GetAllignPB(cbConcatDocWork)},

new StringFormat() { Alignment = GetAllignPB(cbWorkTypeTwo)},

new StringFormat() { Alignment = GetAllignPB(cbNumberTwo)},

new StringFormat() { Alignment = GetAllignPB(cbConcatWorkDiscipline)},

new StringFormat() { Alignment = GetAllignPB(cbDiscipline)},

new StringFormat() { Alignment = GetAllignPB(cbConcatDiscTheme)},

new StringFormat() { Alignment = GetAllignPB(cbTheme)},

new StringFormat() { Alignment = GetAllignPB(cbMadeBy)},

new StringFormat() { Alignment = GetAllignPB(cbCreatorName)},

new StringFormat() { Alignment = GetAllignPB(cbVariant)},

new StringFormat() { Alignment = GetAllignPB(cbRevisoredBy)},

new StringFormat() { Alignment = GetAllignPB(cbRevisorName)},

new StringFormat() { Alignment = GetAllignPB(cbCity)},

new StringFormat() { Alignment = GetAllignPB(cbYear)}

};

float lineHeight = font.GetHeight(g);

float y = 10;

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

{

RectangleF rect = new RectangleF(10, y, pictureBox.Width - 20, lineHeight);

g.DrawString(lines[i], font, Brushes.Black, rect, stringFormats[i]);

y += lineHeight;

if (i == 4 || i == 12 || i == 17)

{

y += 20;

}

}

}

public StringAlignment GetAllignPB(Control cntrl)

{

switch (cntrl.Text)

{

case "Left":

return StringAlignment.Near;

case "Center":

return StringAlignment.Center;

case "Right":

return StringAlignment.Far;

case "Justified":

return StringAlignment.Near;

default:

return StringAlignment.Near;

}

}

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

{

GenPregnantForm();

}

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

{

pbPreview.Paint -= DrawMultipleLines;

ReadFromFile();

}  
-------------------------------------------------------------------------------------------------------------------------------  
namespace ProgMethods\_Work5\_

{

public partial class FrmPregnant : Form

{

public FrmPregnant()

{

InitializeComponent();

}

public void LoadToFile()

{

Application app = new Application();

Document doc = app.Documents.Open(@"C:\Users\user\Documents\Pregnant.docx");

FillDocument(doc);

doc.SaveAs2(@"C:\Users\user\Documents\Pregnant\_filled.docx");

doc.Close();

app.Quit();

}

public void FillDocument(Document doc)

{

FindAndReplaceText(doc, "<TO>", GetText(tbTo,false));

FindAndReplaceText(doc, "<FROM>", GetText(tbFrom,false));

FindAndReplaceText(doc, "<DAY>", GetText(dtpDays, true));

FindAndReplaceText(doc, "<MONTH>", GetText(dtpDays, false));

FindAndReplaceText(doc, "<NUM\_DAYS>", GetText(nudNumDays, false));

FindAndReplaceText(doc, "<SERIAL\_NUM>", GetText(tbSerialNum, false));

FindAndReplaceText(doc, "<SERIAL\_DAY>", GetText(dtpSerialDay, true));

FindAndReplaceText(doc, "<SERIAL\_MONTH>", GetText(dtpSerialDay, false));

FindAndReplaceText(doc, "<NOW\_DAY>", GetText(dtpNowadays, true));

FindAndReplaceText(doc, "<NOW\_MONTH>", GetText(dtpNowadays, false));

FindAndReplaceText(doc, "<SIGN>", GetText(tbSign, false));

FindAndReplaceText(doc, "<SUP\_JOB\_TITLE>", GetText(tbSupJop, false));

FindAndReplaceText(doc, "<SUP\_SIGN>", GetText(tbSupSign, false));

FindAndReplaceText(doc, "<SUP\_FIO>", GetText(tbSupFIO, false));

FindAndReplaceText(doc, "<HR\_SIGN>", GetText(tbHRSign, false));

FindAndReplaceText(doc, "<HR\_FIO>", GetText(tbHRFIO, false));

}

public void FindAndReplaceText(Document doc, string findText, string replaceText)

{

var range = doc.Content;

range.Find.ClearFormatting();

range.Find.Text = findText;

bool found = range.Find.Execute();

while (found)

{

range.Text = range.Text.Replace(findText, replaceText);

range.Collapse(WdCollapseDirection.wdCollapseEnd);

found = range.Find.Execute();

}

}

public string GetText(Control cntrl, bool day)

{

if (cntrl is TextBox)

{

TextBox textBox = (TextBox)cntrl;

return textBox.Text;

}

else if (cntrl is NumericUpDown)

{

NumericUpDown nud = (NumericUpDown)cntrl;

return nud.Value.ToString();

}

else if (cntrl is DateTimePicker) { }

{

DateTimePicker dtp = (DateTimePicker)cntrl;

if (day == true)

{

return dtp.Value.Day.ToString();

}

else

{

return dtp.Value.Month.ToString();

}

}

}

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

{

LoadToFile();

}

}

}

8. Графический пользовательский интерфейс:  
Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание  
Заголовок главной формы должен содержать надпись вида: «Задание №5 выполнил: [Фамилия И.О. автора]; Номер варианта: [Номер]; Дата выполнения: [дд/мм/гггг]».  
  
  
Имя автора-разработчика графического пользовательского интерфейса вводится через свойства (Properties) экранной формы и не должно быть подвержено изменению/редактированию в коде. Это же имя должно быть использовано для вставки в строку Ф.И.О. автора-составителя на титульном листе.  
  
  
Имя и должность лица, проверяющего работу, вводится либо через текстовое поле (Text Box), либо через комбинированный список (Combo Box), если условием индивидуального задания по варианту не предусмотрено иное

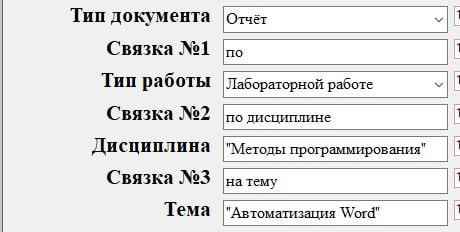
Изображение выглядит как текст, снимок экрана, Шрифт, число

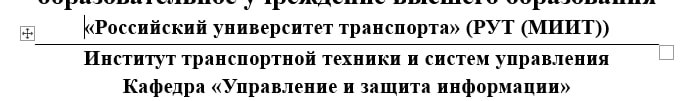
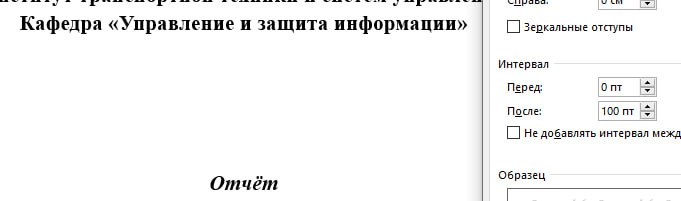
Автоматически созданное описание  
Предусмотрен на графическом пользовательском интерфейсе комбинированный список (Combo Box), в котором выбирается вид отчётного документа

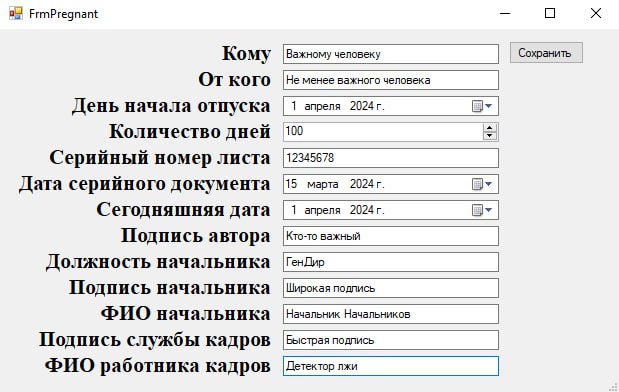
Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание  
Предусмотрен на графическом пользовательском интерфейсе комбинированный список (Combo Box), в котором выбирается вид работы  
Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание  
Предусмотрен комбинированный список (Combo Box), позволяющий выбрать номер работы (диапазон от 1 до 10).

  
 Предусмотрено текстовое поле для ввода (Text Box) наименования работы. Предусмотрено текстовое поле для ввода (Text Box) темы работы.

Предусмотрено текстовое поле для ввода (Text Box) наименования дисциплины, в рамках которой выполняется работа.  
  
Между названием вуза и наименованием кафедры должен иметь место один из интерфейсных элементов управления нижняя граница таблицы, состоящей из одной строки и одной колонки, растянутой по ширине листа (номер варианта делится на 3 и даёт в остатке единицу  
  
Настроен титульный лист без выполнения межстрочных пробелов (пустых абзацев). Для решения задачи необходимо настраивать только абзацный отступ перед и после созданной надписи



Индивидуальное задание:

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

9. Расчёт тестовых примеров:

Изображение выглядит как текст, Шрифт, типография, Графика

Автоматически созданное описание

Создание одного поля с максимальным кеглем

Предпросмотр документа   
Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

10. Вывод:

В ходе работы были успешно освоены навыки подключения внешних модулей (библиотек классов) и их использования в разрабатываемых программах; освоены навыки программируемой настройки параметров документов *Microsoft Office Word*.