МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**КУРСОВОЙ ПРОЕКТ**

по дисциплине: «Разработка приложений в визуальных средах**»**

на тему: « Программная реализация анализа напряжений

в балке при изгибе»

Исполнитель**:** Студент 2-го курса гр. 10701323

Шаплавский Н.С.

Руководитель**:**  доц.Гурский Н.Н.

Минск 2025

**Содержание**

Введение………………………………………………………………………………………………………………………….4

1.Математическая формулировка задачи………………………………………..…………………………...5

2. Описание программы…………………………………………………………….……………………..……………5

2.1. Структурная схема программы…………………...…………………..……………………………5

2.2. Описание разработанного класса………………………………...…..…………………………6

2.3. Описание динамических библиотек……………………………………………………………7

2.4. Основные возможности программы…………………………………………………………...7

2.5 Средства использования сервисов, предоставляемых Microsoft Office……..9

3. Руководство пользователя……………………………………………..……………..……………………………9

4. Методика испытаний……………………………………………..………………………………………………….17

Заключение……………………………………………..……………………………………………..……………………..20

Литература…………………………………………………………………………………………………………………….21

ПРИЛОЖЕНИЕ……………………………………………..……………………………………………………..…………22

Файл Form1.cs……………………………………………..…………………………………………………………………22

Файл mathmanager.cs……………………………………………..…………………………………………………….30

Файл officemediator.cs……………………………………………..………………………………………………..…..31

**Введение**

Современные строительные конструкции и машиностроительные системы подвергаются сложным нагрузкам, среди которых изгиб является одним из наиболее распространенных видов деформации. Расчет напряжений в балках при изгибе представляет собой фундаментальную задачу строительной механики и сопротивления материалов, имеющую важное практическое значение при проектировании несущих конструкций.

Актуальность темы обусловлена необходимостью точного определения напряженно-деформированного состояния балок - основных элементов многих инженерных конструкций. Компьютерное моделирование и программная реализация таких расчетов позволяют существенно ускорить процесс проектирования, минимизировать вероятность ошибок и обеспечить требуемый запас прочности конструкций.

Целью данной курсовой работы является разработка программного обеспечения для анализа напряжений в поперечном сечении балки при воздействии изгибающего момента.

Практическая значимость работы заключается в создании удобного инструмента для инженерных расчетов, который может быть использован в учебном процессе и при выполнении проектных работ. Разработанное приложение обладает интуитивно понятным интерфейсом и предоставляет комплексные возможности по анализу напряженного состояния балок.

1. **Математическая формулировка задачи**

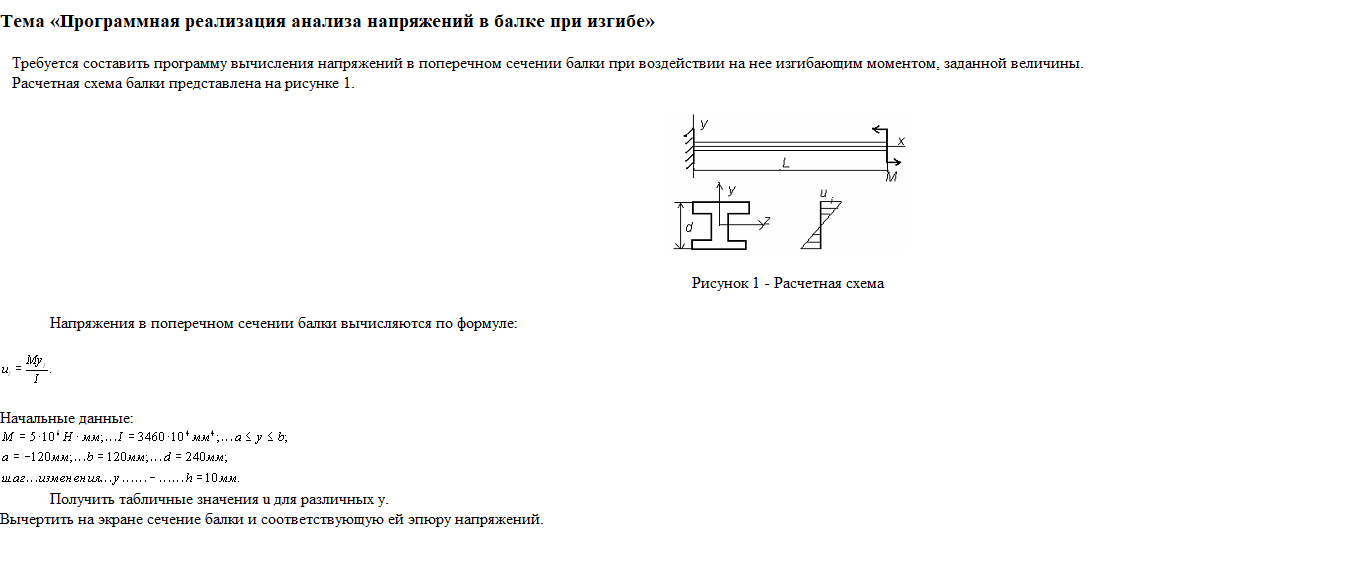
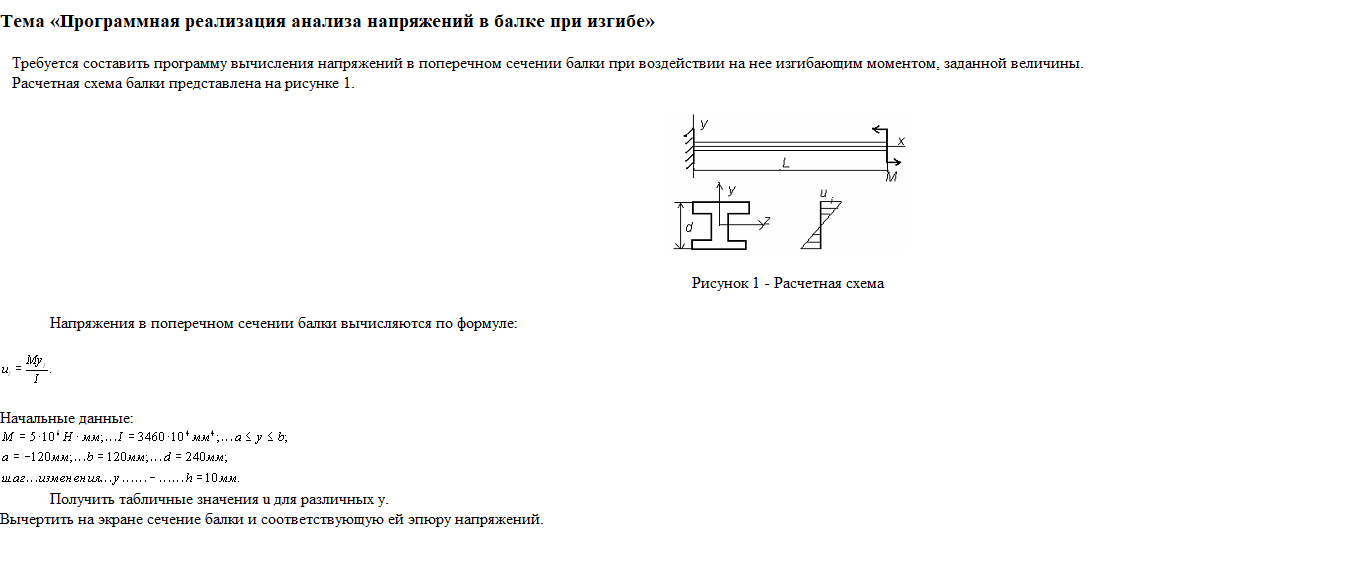
Напряжение в поперечном сечении балки при воздействии на нее изгибающим моментом (рис 1.1)

Рисунок 1.1 – расчетная схема

Напряжение в поперечном сечении балки вычисляется по формуле:

Где

u – напряжение в точке сечения балки Н/мм²

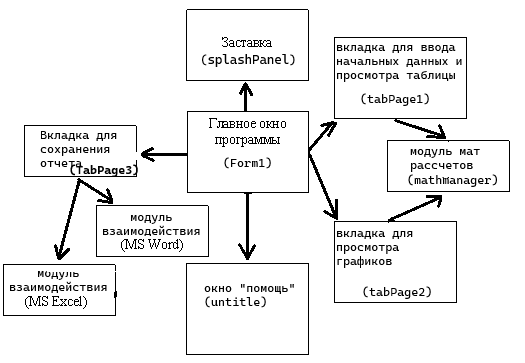
M – Изгибающий момент Н·мм

y – расстояние от центральной оси мм

I – Момент инерции поперечного сечения балки относительно нейтральной оси мм⁴

1. **Описание программы**
   1. **Структурная схема программы**

Структурно программа состоит из главного модуля, динамических библиотек, файлов помощи и обращений к другим программам. Связь модулей приведена на рисунке 2.1

рисунок 2.1- Структурная схема связей модулей программы

* 1. **Описание разработанного класса**

В результате анализа предметной области для её описания и моделирования был разработан класс **BeamStressCalculato**r, инкапсулирующий поля, методы и свойства, применимые к рассматриваемому объекту.

**private**

\_moment: double //Изгибающий момент

\_inertiaMoment: double //Момент энерции

\_minY: double //минимальная y координата(координата низа балки)

\_maxY: double //максимальная y координата(координата верха балки)

\_step: double //Шаг

\_stressTable: List<(double y, double stress)>(); //переменная для хранения значений напряжений в каждой точке y

функция CalculateStress //вычисление напряжения в конкретной точке

функция CreateStressTRable //создание таблицы напряжений

**public**

функция SetParameters //изменение параметров

функция GetStressTable //получение таблицы напряжений

* 1. **Описание динамических библиотек**

При разработке приложения была создана динамическая библиотека для взаимодействия с программами MS Office (word, excel) Officemediator

**private**

\_moment: double //Изгибающий момент

\_inertiaMoment: double //Момент энерции

\_minY: double //минимальная y координата(координата низа балки)

\_maxY: double //максимальная y координата(координата верха балки)

\_step: double //Шаг

\_stressTable: List<(double y, double stress)>(); //переменная для хранения значений напряжений в каждой точке y

**public**

Функция ExcelSave //Сохранение отчета в формате .xlsx

Функция WordSave //Сохранение отчета в формате .xlsx

* 1. **Основные возможности программы**

Программа начинается с отображения оригинальной, а затем на экране появляется главное окно программы (Модуль Form1).

Главный модуль управляет работой всех других модулей в соответствии с запросами пользователя.

Он содержит:

· Вызов вкладки «главная»;(вызывается по умолчанию)

· Вызов вкладки «График»;

· Вызов вкладки «Инструменты»;

Вкладка «главная» содержит:

· Поля для ввода исходных данных

· Кнопки для расчета и вывода информации

· Вызов окна помощи

· Поле вывода таблицы напряжений

Вкладка «График» содержит:

· Схематичный рисунок балки

· График напряжений

· график эпюры напряжений

Вкладка «Инструменты» содержит:

· Вызов *директ-диалога* с выбором пути сохранения

· Передачу информации в MS Word и MS Excel

· Вызов окна помощи

Диалог с пользователем поддерживается с помощью строки статуса, панели инструментов, кнопок, всплывающих уведомлений и других интерфейсных элементов.

Окно «помощь» содержит информацию о приложении, авторе и руководство пользователя. Во вкладке «график» имеется возможность визуально просмотреть изменение напряжение в зависимости от y, а так же просмотреть эпюру напряжений. Окно «Инструменты» служит для того, чтобы у пользователя была возможность сохранить свои вычисления в документ word и excel

Процесс логического взаимодействия пользователя c программой, назначение элементов главного окна описаны в «Руководстве пользователя».

Вопросы непосредственной программной реализации конкретных модулей приведены в приложении. По тексту программ даются достаточно полные комментарии, необходимые для описания переменных, процедур и функций, а также основных шагов реализации используемых алгоритмов.

* 1. **Средства использования сервисов, предоставленных Microsoft Office**

Кроме своей основной задачи (расчет напряжений в балки при изгибе), программа так же позволяет пользователю не терять полученный результат, а сохранять его документом MS Word и MS Exsel в нужном для него пути, а затем использовать их по своему усмотрению.

Так же, что важно, программа позволяет сохранять результаты в этих расширениях не требуя наличия пакета MS Office, что позволяет передавать полученный результат другим людям, не покупая пакет приложений

1. **Руководство пользователя**

Для активизации программы необходимо вызвать файл *Sprogram.exe*. При этом, на экране появится заставка, показанная на рисунке 3.1.

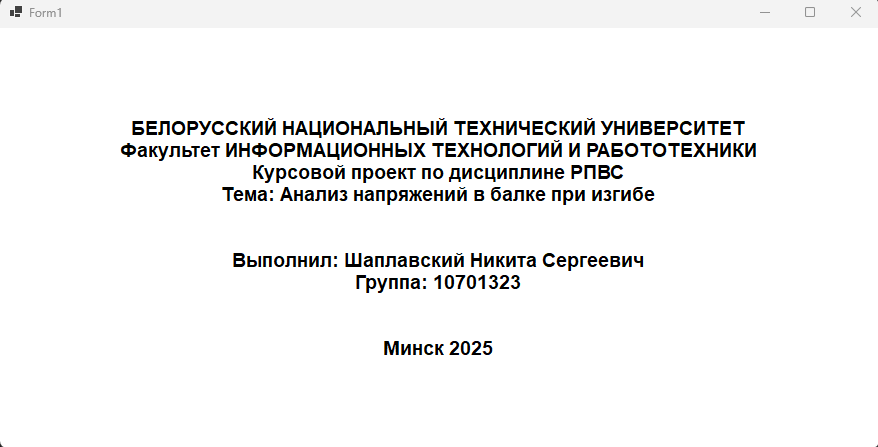


рисунок 3.1 – заставка программы

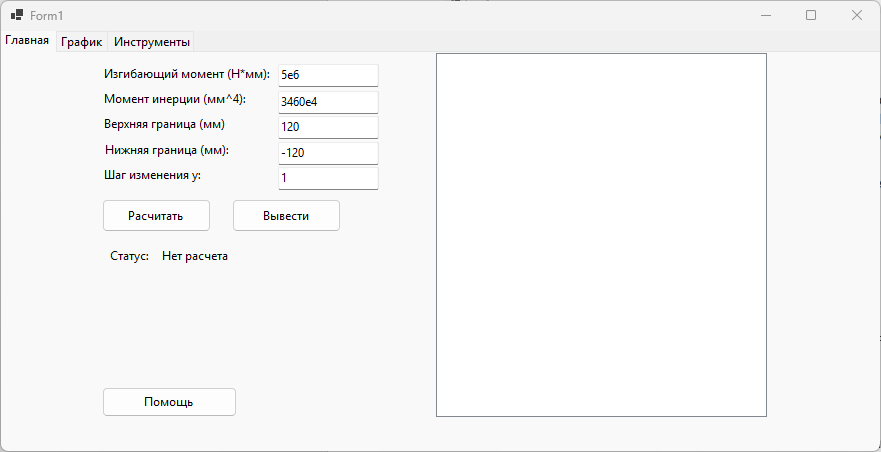
После того как заставка через несколько секунд исчезнет, появится главное окно программы (см. рисунок 3.2).

рисунок 3.2 – Главное окно программы

Как видно, данное окно состоит из следующих частей:

· вкладки;

· Области ввода данных;

· Области вывода информации;

· Управляющих кнопок;

· Кнопка вызова помощи

· поле статуса

Вкладки показаны на рисунке 3.3

рисунок 3.3 – Вкладки программы

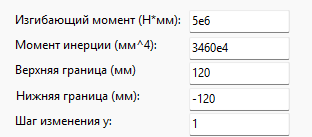
Поля ввода показаны на рисунке 3.4

рисунок 3.4 – Поля ввода

Управляющие кнопки показаны на рисунке 3.5



рисунок 3.5 - Управляющие кнопки

После ввода начальных данных нужно нажать на кнопку “рассчитать”, после чего пользователь может нажать на кнопку “Вывести” для вывода таблицы и построения графиков, либо перейти во вкладку “Инструменты” и сохранить отчет без просмотра данных.

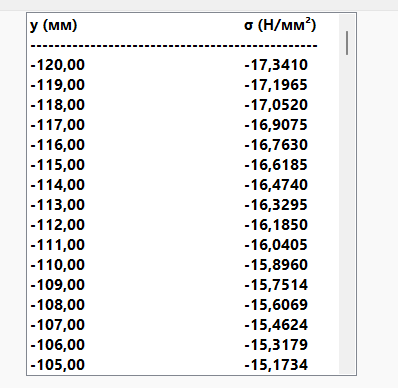
Поле вывода данных показано на рисунке 3.6

рисунок 3.6 - Поле вывода данных

для просмотра графиков пользователь может перейти во вкладку “График” (см. рисунок 3.7)

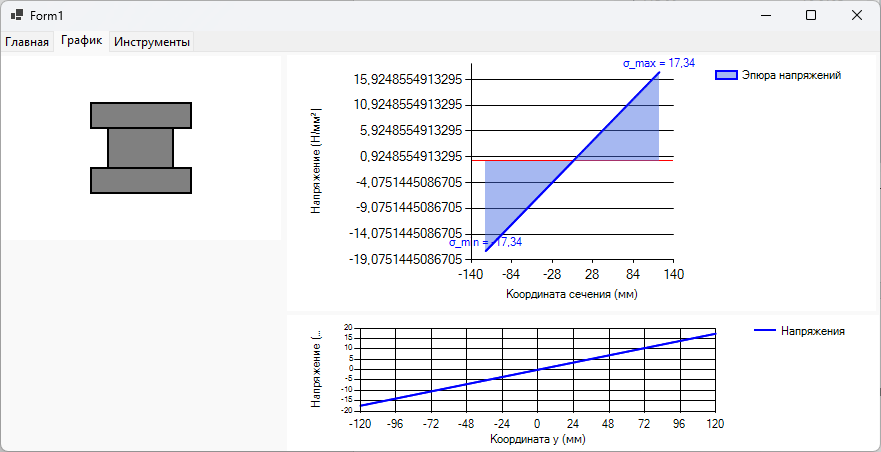


рисунок 3.7 – вкладка “График”

данное окно состоит из следующих частей:

· схематичный рисунок балки

· График эпюры напряжений

· График напряжения

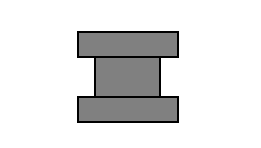
Схематичный рисунок балки показан на рисунке 3.8

рисунок 3.8 - Схематичный рисунок балки

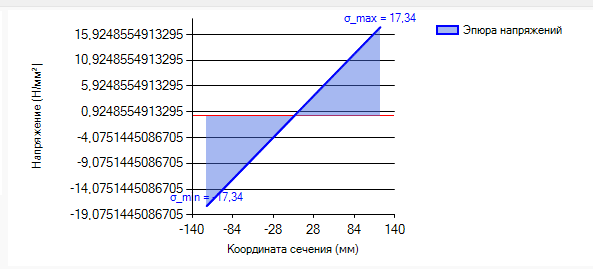
График эпюры показан на рисунке 3.9

рисунок 3.9 - График эпюры

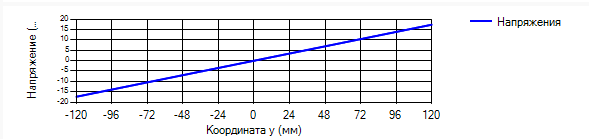
График напряжения показан на рисунке 3.10

рисунок 3.10 - График напряжения

Для сохранения отчета пользователь должен перейти на вкладку “Инструменты”, которая показана на рисунке 3.11

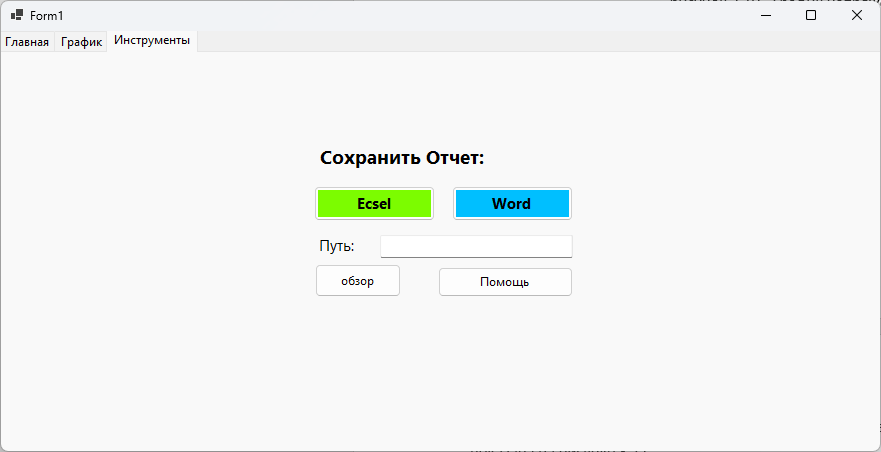


рисунок 3.11 – вкладка “Инструменты”

данное окно состоит из следующих частей:

· вкладки;

· Область ввода данных;

· Управляющих кнопок;

· Кнопка вызова помощи

Область ввода данных показана на рисунке 3.12, представляет собой поле для ввода пути куда сохраниться отчет, так же это можно сделать с помощью кнопки “обзор”(Показана на рисунке 3.13), после чего вылезет диалоговое окно выбора директории (рисунок 3.14), и выбранная директория запишется в поле ввод(рисунок 3.15)

рисунок 3.12 – поле ввода пути для сохранения отчета

рисунок 3.13 – кнопка “обзор”(выбора директории)

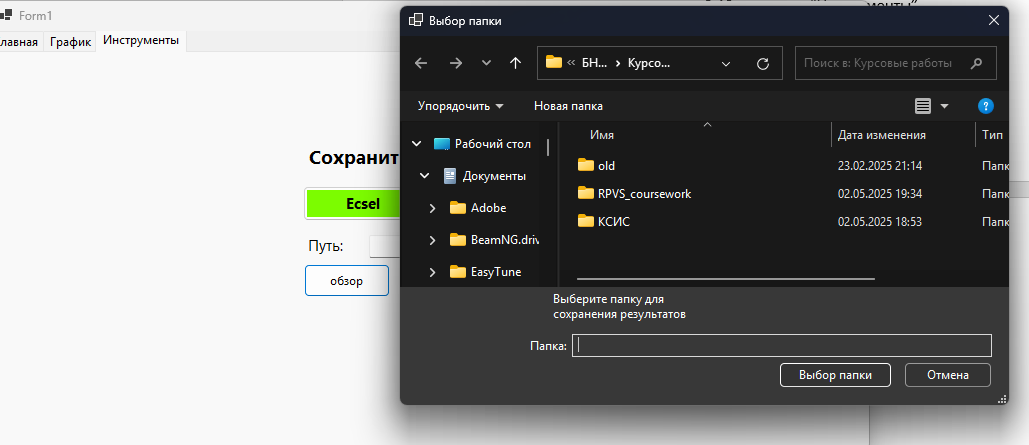


рисунок 3.14 – Диалоговое окно выбора директории

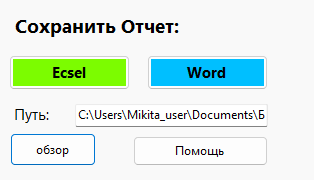


рисунок 3.15– поле ввода с записанным путем

После этих действий пользователь может нажать на кнопку “Exsel” для сохранения отчета в формате документа exsel, либо на кнопку “Word” для сохранения в формате документа Word. Кнопки действий представлены на рисунке 3.16



рисунок 3.16 – кнопки действий

Так же на этой вкладке как и на главной есть кнопка “Помощь”(показана на рисунке 3.17)

рисунок 3.17 – кнопки “Помощь”

После нажатия данный кнопки откроется HTML помощник в котором будет полная документация программы (о программе, руководство пользователя, об авторе) показан на рисунке 3.18

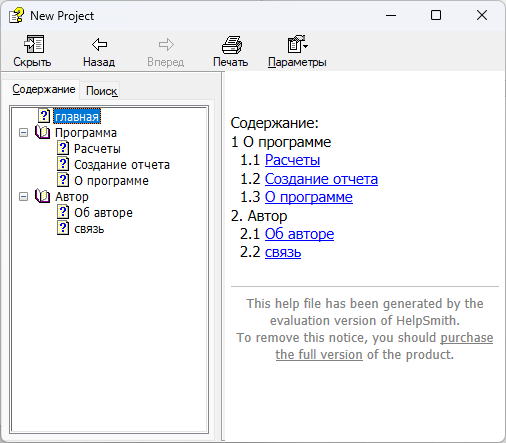


рисунок 3.18 – HTML помощник

Для выхода из помощника следует нажать на крестик в правом верхнем углу помошника

Для выхода из программы так же следует нажать на крестик в правом верхнем углу

1. **Методика испытаний**

Целью проведения испытаний является проверка работоспособности (надежности) программы при различных условиях ее функционирования.

Программа должна обеспечивать корректность ввода исходных данных (путем осуществления соответствующих проверок и информирования пользователя о возникших неточностях в работе), а также получение непротиворечивого результата.

Для демонстрации работоспособности программы необходимо провести ряд испытаний с различными начальными условиями.

Тесты выполнялись в среде:

ОС: Windows 11 Professional

CPU: Intel core i7 10700 4.4 МГц

RAM: 32Gb

Тест №1

Попытка ввести неверный формат данных в поля ввода во вкладке “Главная”

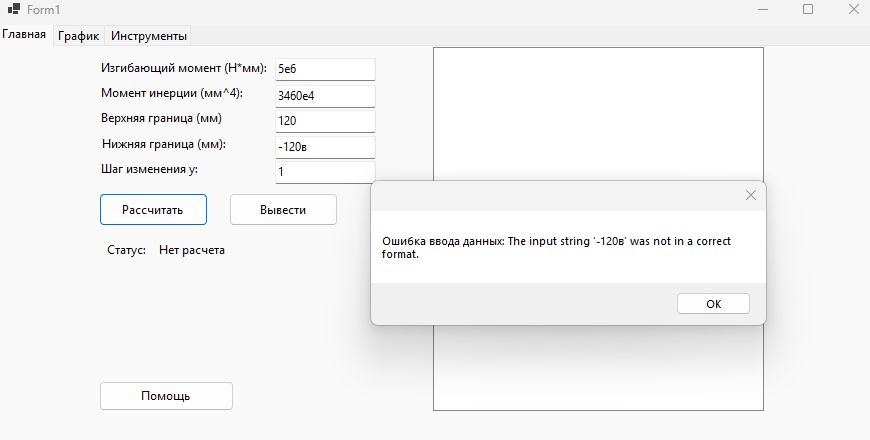
Результат, тест проведен успешно, после нажатия кнопки, рассчитать вылазит уведомление о неверном вводе(рисунок 4.1)

рисунок 4.1 – Уведомление об ошибке

Тест №2

Ввод неверных данных во вкладке “Главная”, например поставить отрицательный шаг и т.п.

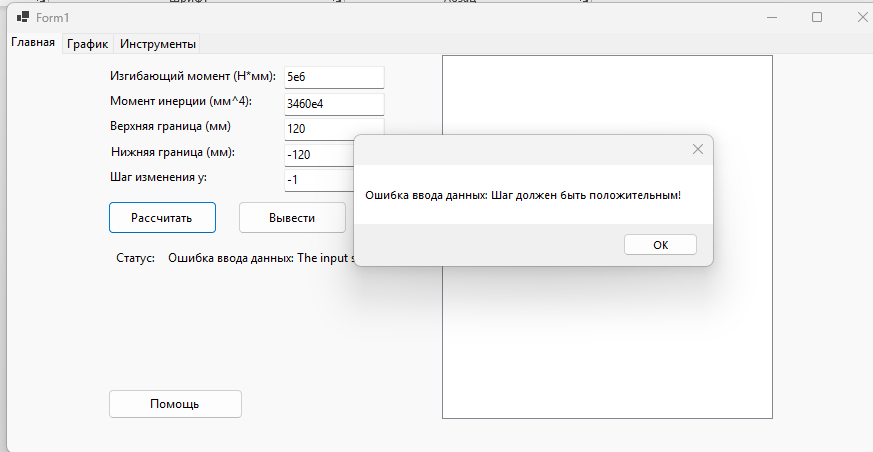
Результат: Тест проведен успешно, после нажатия кнопки Рассчитать, вылазит уведомление о неверном вводе(рисунок 4.2)

рисунок 4.2 – Уведомление об ошибке

Таким образом, проведенное тестирование программы не выявило сбойных ситуаций и некорректностей в ее работе. Следует считать, что в целом программа протестирована, отвечает поставленным требованиям и вполне работоспособна.

**Заключение**

В ходе выполнения курсового проекта была разработана программа для анализа напряжений в балке при изгибе, соответствующая поставленным требованиям. Приложение позволяет автоматизировать расчеты, минимизировать вероятность ошибок и визуализировать результаты, что делает его полезным инструментом для инженерных расчетов.

1. Реализован расчет напряжений с учетом заданных параметров балки (изгибающего момента, момента инерции, границ сечения).

2. Создан интуитивно понятный интерфейс

3. Обеспечена интеграция с MS Office:

- Экспорт отчетов в форматы Word и Excel без необходимости установки пакета Office (благодаря использованию библиотек DocX и ClosedXML).

4. Проведено тестирование, подтвердившее корректность работы программы при различных входных данных, включая обработку ошибок ввода.

Программа успешно решает поставленную задачу и может быть использована в учебном процессе, а также как вспомогательный инструмент в инженерных расчетах. Разработанные методы и подходы демонстрируют эффективность применения современных технологий (C#, Windows Forms, Open XML) для автоматизации инженерных задач.

В ходе выполнения курсовой работы:

· были закреплены знания по курсу «Конструирование программ и языки программирования»;

· приобретен опыт при разработке объектно-ориентированных программ;

· изучены принципы создания динамических библиотек;

- Изучены методы интеграции с приложениями через библиотеки DocX и ClosedXML для экспорта в Word/Excel без COM-технологий.

**Литература**

1. **Microsoft Docs: Windows Forms**– официальная документация по разработке приложений на C# и .NET.

<https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms>

2. **Xceed Words.NET Documentation** – руководство по работе с библиотекой для создания Word-документов.

<https://xceed.com/xceed-words-for-net/>

3. **ClosedXML: GitHub & Documentation** – создание Excel-файлов без использования Interop.

<https://github.com/ClosedXML/ClosedXML>

4. **Chart Controls in .NET** – официальный гайд по построению графиков в Windows Forms.

<https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.datavisualization.charting>

5. **Stack Overflow: C# и WinForms** – решения распространённых проблем при разработке.

<https://stackoverflow.com/questions/tagged/c%23+winforms>

6. **TutorialsTeacher: C# Programming** – основы языка и работы с Windows Forms.

https://www.tutorialsteacher.com/csharp

7. **Open XML SDK Documentation** – работа с DOCX/XLSX на низком уровне (альтернатива Interop).

<https://docs.microsoft.com/ru-ru/office/open-xml/open-xml-sdk>

**Приложение**

**Файл Form1.cs**

using System;

using System.Diagnostics;

using System.Reflection;

using System.Windows.Forms.DataVisualization.Charting;

namespace Sprogram

{

public partial class Form1 : Form

{

private Panel splashPanel;

private BeamStressCalculator beamStressCalculator = new BeamStressCalculator();

public Form1()

{

InitializeComponent();

ShowSplashInsideForm();

DrawIBeamInPictureBox();

}

private async void ShowSplashInsideForm()

{

// Создаем панель-заставку

splashPanel = new Panel()

{

Dock = DockStyle.Fill,

BackColor = Color.White

};

var label = new Label()

{

Text = "БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ\nФакультет ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И РАБОТОТЕХНИКИ\nКурсовой проект по дисциплине РПВС\nТема: Анализ напряжений в балке при изгибе\n\n\nВыполнил: Шаплавский Никита Сергеевич\nГруппа: 10701323\n\n\nМинск 2025",

Dock = DockStyle.Fill,

Font = new Font("Arial", 14, FontStyle.Bold),

TextAlign = ContentAlignment.MiddleCenter

};

splashPanel.Controls.Add(label);

this.Controls.Add(splashPanel);

splashPanel.BringToFront();

// Ждём 5 секунд, не блокируя интерфейс

await Task.Delay(5000);

// Удаляем панель-заставку

splashPanel.Visible = false;

splashPanel.Dispose();

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

double inter = Convert.ToDouble(textBox1.Text);

double moment = Convert.ToDouble(textBox2.Text);

double minY = Convert.ToDouble(textBox3.Text);

double maxY = Convert.ToDouble(textBox4.Text);

double step = Convert.ToDouble(textBox5.Text);

beamStressCalculator.SetParameters(inter, moment, maxY, minY, step);

label7.Text = "Параметры рассчитаны";

}

catch (Exception ex)

{

MessageBox.Show("Ошибка ввода данных: " + ex.Message);

label7.Text = "Ошибка ввода данных: " + ex.Message;

return;

}

}

private void button2\_Click(object sender, EventArgs e)

{

try

{

// Проверка, были ли заданы параметры

if (beamStressCalculator.Moment == 0 || beamStressCalculator.InertiaMoment == 0)

{

MessageBox.Show("Сначала задайте параметры балки!", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

var stressTable = beamStressCalculator.GetStressTable();

chart1.Series.Clear();

chart1.ChartAreas.Clear();

// Очищаем ListBox перед заполнением

listBox1.Items.Clear();

// Добавляем заголовок таблицы

listBox1.Items.Add("y (мм)\t\t\tσ (Н/мм²)");

listBox1.Items.Add("------------------------------------------------");

// Заполняем ListBox данными

foreach (var point in stressTable)

{

listBox1.Items.Add($"{point.y:F2}\t\t\t{point.stress:F4}");

}

// Создаем и настраиваем область графика

ChartArea chartArea = new ChartArea("MainArea")

{

AxisX = { Title = "Координата y (мм)" },

AxisY = { Title = "Напряжение (Н/мм²)" }

};

chart1.ChartAreas.Add(chartArea);

// Создаем серию данных

Series series = new Series("Напряжения")

{

ChartType = SeriesChartType.Line,

Color = Color.Blue,

BorderWidth = 2

};

// Заполняем данными

foreach (var point in stressTable)

{

series.Points.AddXY(point.y, point.stress);

}

chart1.Series.Add(series);

// Настройка масштаба

chartArea.AxisX.Minimum = beamStressCalculator.MinY;

chartArea.AxisX.Maximum = beamStressCalculator.MaxY;

chartArea.AxisX.Interval = (beamStressCalculator.MaxY - beamStressCalculator.MinY) / 10;

// Добавляем линию нейтральной оси

AddNeutralAxisLine(chartArea);

label7.Text = "График успешно построен";

DrawBeamWithStressDiagram();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

label7.Text = "Ошибка при построении графика";

}

}

private void AddNeutralAxisLine(ChartArea chartArea)

{

StripLine neutralLine = new StripLine

{

Interval = 0,

IntervalOffset = 0,

StripWidth = 0.1,

BackColor = Color.Red,

BorderDashStyle = ChartDashStyle.Dash,

BorderWidth = 1,

ToolTip = "Нейтральная ось (y=0)"

};

chartArea.AxisY.StripLines.Add(neutralLine);

}

private void DrawIBeamInPictureBox()

{

Bitmap bmp = new Bitmap(pictureBox1.Width, pictureBox1.Height);

using (Graphics g = Graphics.FromImage(bmp))

{

g.Clear(Color.White);

Pen pen = new Pen(Color.Black, 2);

Brush brush = Brushes.Gray;

// Размеры сечения

int beamWidth = 100; // ширина верхней/нижней полки

int flangeHeight = 25; // высота полок (верх/низ)

int webWidth = 65; // толщина вертикальной стойки (стержня)

int webHeight = 40; // высота стойки

// Начальная точка (левый верхний угол верхней полки)

int x = (pictureBox1.Width - beamWidth) / 2;

int y = (pictureBox1.Height - (2 \* flangeHeight + webHeight)) / 2;

// Верхняя полка

Rectangle topFlange = new Rectangle(x, y, beamWidth, flangeHeight);

// Вертикальная стойка

int webX = x + (beamWidth - webWidth) / 2;

int webY = y + flangeHeight;

Rectangle web = new Rectangle(webX, webY, webWidth, webHeight);

// Нижняя полка

int bottomY = y + flangeHeight + webHeight;

Rectangle bottomFlange = new Rectangle(x, bottomY, beamWidth, flangeHeight);

// Рисуем и закрашиваем

g.FillRectangle(brush, topFlange);

g.FillRectangle(brush, web);

g.FillRectangle(brush, bottomFlange);

g.DrawRectangle(pen, topFlange.X, topFlange.Y, topFlange.Width, topFlange.Height);

g.DrawRectangle(pen, web.X, web.Y, web.Width, web.Height);

g.DrawRectangle(pen, bottomFlange.X, bottomFlange.Y, bottomFlange.Width, bottomFlange.Height);

}

pictureBox1.Image = bmp;

}

private void DrawBeamWithStressDiagram()

{

try

{

chart2.Series.Clear();

chart2.ChartAreas.Clear();

// Создаем область для графика

ChartArea chartArea = new ChartArea("BeamView")

{

AxisX = {

Title = "Координата сечения (мм)",

Minimum = beamStressCalculator.MinY - 20,

Maximum = beamStressCalculator.MaxY + 20,

MajorGrid = { Enabled = false }

},

AxisY = {

Title = "Напряжение (Н/мм²)",

IntervalAutoMode = IntervalAutoMode.VariableCount

}

};

chart2.ChartAreas.Add(chartArea);

// Получаем данные напряжений

var stressData = beamStressCalculator.GetStressTable();

// 1. Серия для эпюры напряжений

Series stressSeries = new Series("Эпюра напряжений")

{

ChartType = SeriesChartType.Area,

Color = Color.FromArgb(120, 65, 105, 225), // Полупрозрачный синий

BorderColor = Color.Blue,

BorderWidth = 2

};

// 2. Серия для контура балки

/\*Series beamSeries = new Series("Балка")

{

ChartType = SeriesChartType.Line,

Color = Color.Black,

BorderWidth = 3

};\*/

// Заполняем данные

foreach (var point in stressData)

{

stressSeries.Points.AddXY(point.y, point.stress);

}

// Рисуем контур балки (вертикальные линии)

double beamWidth = 5; // Условная толщина для визуализации

double minY = beamStressCalculator.MinY;

double maxY = beamStressCalculator.MaxY;

/\*

// Левая граница балки

beamSeries.Points.AddXY(minY, 0);

beamSeries.Points.AddXY(minY, stressData.Min(p => p.stress));

// Правая граница балки

beamSeries.Points.AddXY(maxY, stressData.Max(p => p.stress));

beamSeries.Points.AddXY(maxY, 0);

chart2.Series.Add(beamSeries);

\*/

// Добавляем серии

chart2.Series.Add(stressSeries);

// Настраиваем внешний вид

chartArea.AxisY.Minimum = stressData.Min(p => p.stress) \* 1.1;

chartArea.AxisY.Maximum = stressData.Max(p => p.stress) \* 1.1;

// Добавляем нейтральную ось

StripLine neutralAxis = new StripLine()

{

Interval = 0,

IntervalOffset = 0,

StripWidth = 0.2,

BackColor = Color.Red,

BorderDashStyle = ChartDashStyle.Dash

};

chartArea.AxisY.StripLines.Add(neutralAxis);

// Подписи экстремальных значений

var maxPoint = stressSeries.Points.FindMaxByValue();

maxPoint.Label = $"σ\_max = {maxPoint.YValues[0]:F2}";

maxPoint.LabelForeColor = Color.Blue;

var minPoint = stressSeries.Points.FindMinByValue();

minPoint.Label = $"σ\_min = {minPoint.YValues[0]:F2}";

minPoint.LabelForeColor = Color.Blue;

label7.Text = "Балка с эпюрой напряжений построена";

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при построении: {ex.Message}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button3\_Click(object sender, EventArgs e)

{

try

{

if (string.IsNullOrWhiteSpace(textBox6.Text))

{

MessageBox.Show("Сначала укажите папку для сохранения!", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

// Создаем полный путь к файлу

string fileName = "beam\_stress\_analysis.xlsx";

string fullPath = Path.Combine(textBox6.Text, fileName);

// Проверяем и создаем директорию, если нужно

Directory.CreateDirectory(Path.GetDirectoryName(fullPath));

var mediator = new OfficeMediator(

beamStressCalculator.GetStressTable(),

beamStressCalculator.Moment,

beamStressCalculator.InertiaMoment,

beamStressCalculator.MinY,

beamStressCalculator.MaxY);

mediator.ExcelSave(fullPath);

MessageBox.Show($"Файл успешно сохранен:\n{fullPath}", "Успех",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка экспорта в Excel:\n{ex.Message}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button4\_Click(object sender, EventArgs e)

{

try

{

if (string.IsNullOrWhiteSpace(textBox6.Text))

{

MessageBox.Show("Сначала укажите папку для сохранения!", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

// Создаем полный путь к файлу

string fileName = "beam\_analysis\_report.docx";

string fullPath = Path.Combine(textBox6.Text, fileName);

// Проверяем и создаем директорию, если нужно

Directory.CreateDirectory(Path.GetDirectoryName(fullPath));

// Получаем изображение графика

var chartImage = GetChartImage(chart2);

var mediator = new OfficeMediator(

beamStressCalculator.GetStressTable(),

beamStressCalculator.Moment,

beamStressCalculator.InertiaMoment,

beamStressCalculator.MinY,

beamStressCalculator.MaxY);

mediator.WordSave(fullPath, chartImage);

MessageBox.Show($"Файл успешно сохранен:\n{fullPath}", "Успех",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка экспорта в Word:\n{ex.Message}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private Image GetChartImage(Chart chart)

{

using (var ms = new MemoryStream())

{

chart.SaveImage(ms, ChartImageFormat.Png);

return Image.FromStream(ms);

}

}

private void button5\_Click(object sender, EventArgs e)

{

// Настраиваем диалог выбора папки

folderBrowserDialog1.Description = "Выберите папку для сохранения результатов";

folderBrowserDialog1.ShowNewFolderButton = true; // Разрешаем создавать новые папки

folderBrowserDialog1.RootFolder = Environment.SpecialFolder.MyDocuments; // Начальная папка

// Показываем диалог и проверяем результат

if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)

{

// Получаем выбранный путь и записываем в TextBox

textBox6.Text = folderBrowserDialog1.SelectedPath;

// Опционально: добавляем слеш в конце, если его нет

if (!textBox6.Text.EndsWith("\\"))

{

textBox6.Text += "\\";

}

}

}

private void button6\_Click(object sender, EventArgs e)

{

try

{

string helpFilePath = "Untitled.chm";

// Проверяем существование файла

if (!File.Exists(helpFilePath))

{

// Если файл не найден в текущей директории, попробуем найти в директории приложения

helpFilePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Untitled.chm");

if (!File.Exists(helpFilePath))

{

MessageBox.Show("Файл справки 'Untitled.chm' не найден!", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

// Открываем CHM-файл с помощью стандартного средства Windows

Process.Start(new ProcessStartInfo(helpFilePath) { UseShellExecute = true });

}

catch (Exception ex)

{

MessageBox.Show($"Не удалось открыть файл справки: {ex.Message}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

**Файл mathmanager.cs**

using System;

using System.Collections.Generic;

namespace Sprogram

{

public class BeamStressCalculator

{

private double \_moment;

private double \_inertiaMoment;

private double \_minY;

private double \_maxY;

private double \_step;

List<(double, double)> \_stressTable = new List<(double y, double stress)>();

public double Moment => \_moment;

public double InertiaMoment => \_inertiaMoment;

public double MinY => \_minY;

public double MaxY => \_maxY;

public double Height => \_maxY - \_minY;

public void SetParameters(double moment, double inertiaMoment, double minY, double maxY, double step)

{

if (inertiaMoment <= 0)

throw new ArgumentException("МОМЕНТ ИНЕРЦИИ ДОЛЖЕН БЫТЬ БОЛЬШЕ 0!!!");

if (minY >= maxY)

throw new ArgumentException("Нижняя граница должна быть меньше верхней!");

if (step <= 0)

throw new ArgumentException("Шаг должен быть положительным!");

\_moment = moment;

\_inertiaMoment = inertiaMoment;

\_minY = minY;

\_maxY = maxY;

\_step = step;

CreateStressTRable();

}

private void CreateStressTRable()

{

var stressTable = new List<(double y, double stress)>();

// Добавляем начальную точку

double firstY = \_minY;

stressTable.Add((firstY, CalculateStress(firstY)));

\_stressTable.Add((firstY, CalculateStress(firstY)));

// Добавляем промежуточные точки

for (double y = \_minY + \_step; y < \_maxY; y += \_step)

{

stressTable.Add((y, CalculateStress(y)));

\_stressTable.Add((y, CalculateStress(y)));

}

// Добавляем конечную точку (если не совпадает с последней)

if (\_maxY != stressTable.Last().y)

{

stressTable.Add((\_maxY, CalculateStress(\_maxY)));

\_stressTable.Add((\_maxY, CalculateStress(\_maxY)));

}

if (stressTable.Count == 0)

throw new InvalidOperationException("Не удалось создать таблицу напряжений. Проверьте параметры.");

}

public List<(double y, double stress)> GetStressTable()

{

return \_stressTable;

}

public List<(double y, double stress)> GetTable()

{

return \_stressTable;

}

private double CalculateStress(double y)

{

if (y < \_minY || y > \_maxY)

throw new ArgumentOutOfRangeException(nameof(y), $"y должен быть в диапазоне [{\_minY}, {\_maxY}]");

return (\_moment \* y) / \_inertiaMoment;

}

}

**Файл officemediator.cs**

using System;

using System.Collections.Generic;

using System.DirectoryServices;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using Xceed.Document.NET; // Для DocX

using Xceed.Words.NET; // Для Xceed.Words.NET

using System.Drawing;

using System.Drawing.Imaging;

namespace Sprogram

{

internal class OfficeMediator

{

private readonly List<(double y, double stress)> \_stressTable;

private readonly double \_moment;

private readonly double \_inertiaMoment;

private readonly double \_minY;

private readonly double \_maxY;

public OfficeMediator(List<(double y, double stress)> stressTable,

double moment,

double inertiaMoment,

double minY,

double maxY)

{

\_stressTable = stressTable;

\_moment = moment;

\_inertiaMoment = inertiaMoment;

\_minY = minY;

\_maxY = maxY;

}

public void ExcelSave(string filePath)

{

try

{

using (var workbook = new ClosedXML.Excel.XLWorkbook())

{

var worksheet = workbook.Worksheets.Add("Напряжения");

// Заголовки

worksheet.Cell(1, 1).Value = "y (мм)";

worksheet.Cell(1, 2).Value = "σ (Н/мм²)";

// Данные

for (int i = 0; i < \_stressTable.Count; i++)

{

worksheet.Cell(i + 2, 1).Value = \_stressTable[i].y;

worksheet.Cell(i + 2, 2).Value = \_stressTable[i].stress;

}

// Автоширина

worksheet.Columns().AdjustToContents();

workbook.SaveAs(filePath);

}

}

catch (Exception ex)

{

throw new Exception($"Ошибка экспорта в Excel: {ex.Message}");

}

}

public void WordSave(string filePath, System.Drawing.Image chartImage = null)

{

try

{

// Создаем новый документ

using (var document = DocX.Create(filePath))

{

// 1. Добавляем заголовок

var title = document.InsertParagraph("Отчет о напряжениях в балке");

title.FontSize(16).Bold().SpacingAfter(40).Alignment = Alignment.center;

// 2. Добавляем параметры балки

var parameters = document.InsertParagraph();

parameters.AppendLine($"Изгибающий момент: {\_moment} Н·мм")

.AppendLine($"Момент инерции: {\_inertiaMoment} мм⁴")

.AppendLine($"Границы сечения: {\_minY}...{\_maxY} мм")

.SpacingAfter(20);

// 3. Добавляем таблицу данных

var table = document.AddTable(\_stressTable.Count + 1, 2);

// Заголовки таблицы

table.Rows[0].Cells[0].Paragraphs.First().Append("y (мм)").Bold();

table.Rows[0].Cells[1].Paragraphs.First().Append("σ (Н/мм²)").Bold();

// Заполняем таблицу данными

for (int i = 0; i < \_stressTable.Count; i++)

{

table.Rows[i + 1].Cells[0].Paragraphs.First().Append(\_stressTable[i].y.ToString("F2"));

table.Rows[i + 1].Cells[1].Paragraphs.First().Append(\_stressTable[i].stress.ToString("F4"));

}

document.InsertTable(table);

document.InsertParagraph().SpacingAfter(20);

// 4. Добавляем график, если он передан

if (chartImage != null)

{

// Сохраняем изображение во временный файл

string tempImagePath = Path.Combine(Path.GetTempPath(), "chart\_temp.png");

// Убедимся, что сохраняем в правильном формате

using (var stream = new FileStream(tempImagePath, FileMode.Create))

{

chartImage.Save(stream, ImageFormat.Png);

}

try

{

// Добавляем изображение в документ

var image = document.AddImage(tempImagePath);

var picture = image.CreatePicture();

// Центрируем изображение

var imageParagraph = document.InsertParagraph();

imageParagraph.AppendPicture(picture).Alignment = Alignment.center;

imageParagraph.SpacingAfter(20);

}

finally

{

// Удаляем временный файл

if (File.Exists(tempImagePath))

{

File.Delete(tempImagePath);

}

}

}

// Сохраняем документ

document.Save();

}

}

catch (Exception ex)

{

throw new Exception($"Ошибка экспорта в Word: {ex.Message}");

}

}

private void ReleaseExcelObjects(params object[] objects)

{

foreach (var obj in objects)

{

try { if (obj != null) Marshal.ReleaseComObject(obj); }

catch { /\* ignored \*/ }

}

GC.Collect();

}

private void ReleaseWordObjects(params object[] objects)

{

foreach (var obj in objects)

{

try { if (obj != null) Marshal.ReleaseComObject(obj); }

catch { /\* ignored \*/ }

}

GC.Collect();

}

}

}