Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Российский университет транспорта»

(ФГАОУ ВО РУТ(МИИТ), РУТ (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа № 18

по дисциплине: «Программирование и основы алгоритмизации»

на тему: «Автоматизация Excel»

Выполнил: ст. гр. ТУУ-211

Воронов А. В.

Вариант №12

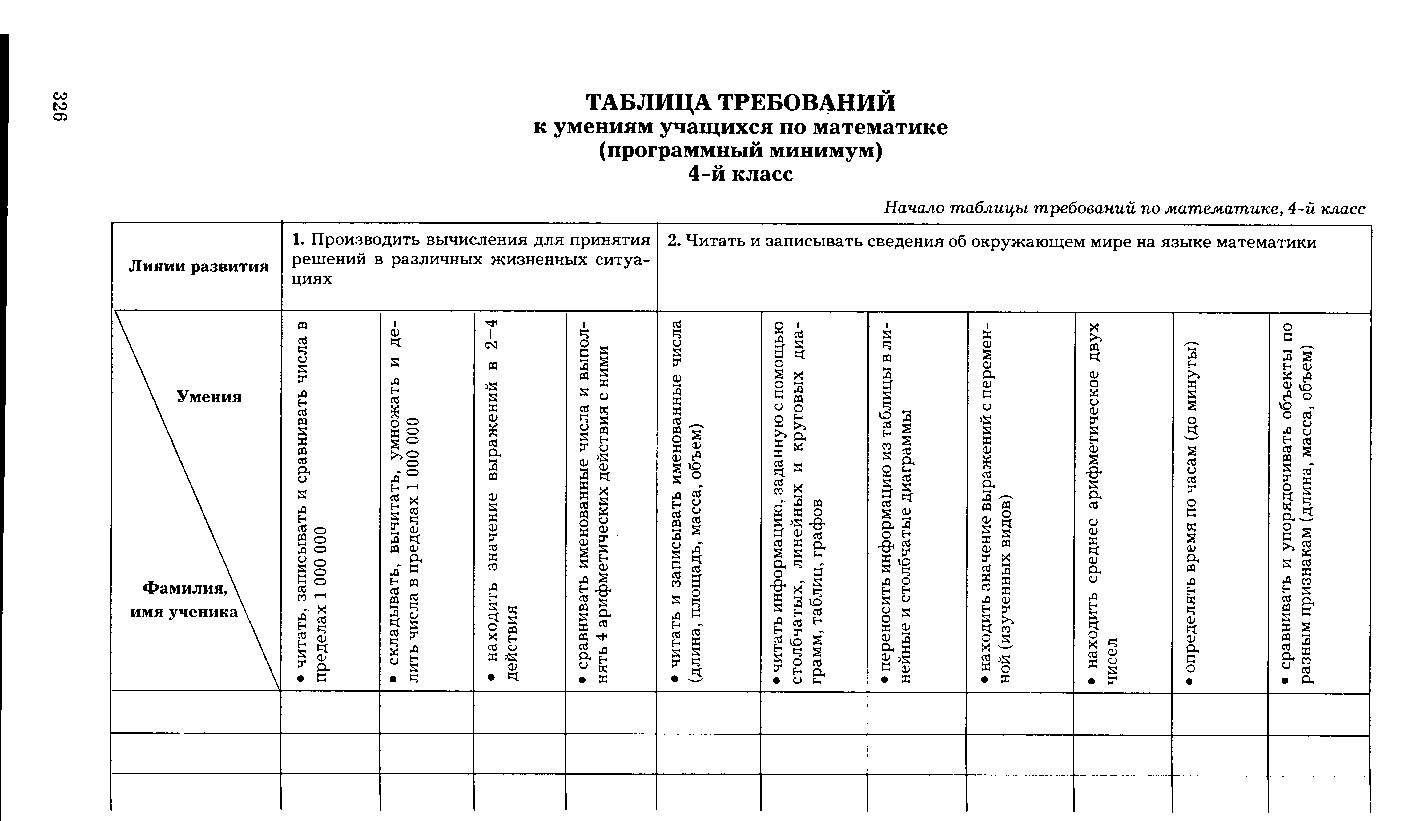
Проверил: к.т.н., доц. Сафронов А.И.

1. **Формулировка цели работы**

Освоить навыки подключения внешних модулей и их использования в разрабатываемых программах; освоить навыки программируемой настройки параметров электронных таблиц Microsoft Office Excel.

1. **Описание задачи**

В интегрированной среде разработки Microsoft Visual Studio разработать программу в режиме Windows Forms Application на языке Visual C#, представляющую собой средство взаимодействия с электронными таблицами Microsoft Office Excel.



1. Продумать и создать эргономичный графический пользовательский интерфейс с удобным предпросмотром и кнопкой «Выгрузка в Excel».

2. Нажатие на кнопку инициирует запуск на исполнение метода составления и заполнения электронной таблицы заданной по варианту структуры.

3. При запуске приложения, программа считывает файл data.csv и загружает из него данные в таблицу.

1. **Сеть Петри запрограммированного технологического процесса**

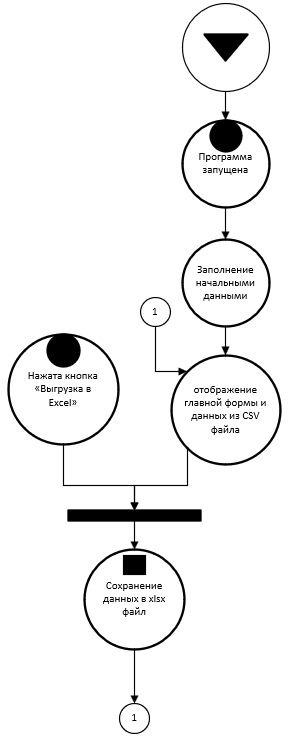


Рисунок 1 – Сеть Петри запрограммированного технологического процесса

1. **Блок-схема алгоритма**
2. **Подбор тестовых примеров**

В качестве тестовых данных созданы три ряда данных:

1) Иванов Иван Умения: 5 4 4 3 4 5 4 5 4 5 3

2) Сидоров Петр Умения: 4 4 4 5 5 5 4 4 5 5 4

3) Полева Елена Умения: 4 5 5 4 5 5 4 5 5 3 5

1. **Листинг (код) составленного программного обеспечения**

**Файл Form1.cs**

using System;

using System.Windows.Forms;

using System.IO;

using Excel = Microsoft.Office.Interop.Excel;

using System.Linq;

namespace ExcelStudent

{

public partial class Form1 : Form

{

private const string CsvFileName = "data.csv";

public Form1()

{

InitializeComponent();

LoadCsvIfExists();

}

//загрузка данных из CSV-файла

private void LoadCsvIfExists()

{

string csvFilePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "data.csv");

if (File.Exists(csvFilePath))

{

try

{

var lines = File.ReadAllLines(csvFilePath);

if (lines.Length > 1) // Проверяем, что есть хотя бы две строки (шапка и данные)

{

dataGridView1.Rows.Clear();

dataGridView1.Columns.Clear();

// Создаем столбцы: Полное имя и 11 оценок

dataGridView1.Columns.Add("FullName", "Фамилия Имя");

for (int i = 1; i <= 11; i++)

{

dataGridView1.Columns.Add($"Grade{i}", $"Оценка {i}");

}

// Загружаем данные, начиная со второй строки

foreach (var line in lines.Skip(1))

{

var values = line.Split(',');

if (values.Length == 12) // Проверяем, что строка содержит 12 элементов

{

dataGridView1.Rows.Add(values);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке CSV: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

// Создание нового Excel приложения

Excel.Application excelApp = new Excel.Application();

if (excelApp == null)

{

MessageBox.Show("Excel не установлен!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Создание новой рабочей книги

Excel.Workbook workbook = excelApp.Workbooks.Add();

Excel.Worksheet worksheet = (Excel.Worksheet)workbook.Sheets[1];

// Вставляем строку с надписями над таблицей

worksheet.Cells[1, 1] = "Линии развития";

// Объединяем ячейки для "1. Производить вычисления..."

worksheet.Range[worksheet.Cells[1, 2], worksheet.Cells[1, 5]].Merge();

worksheet.Cells[1, 2] = "1. Производить вычисления для принятия решений в различных жизненных ситуациях";

// Объединяем ячейки для "2. Читать и записывать сведения..."

worksheet.Range[worksheet.Cells[1, 6], worksheet.Cells[1, 12]].Merge();

worksheet.Cells[1, 6] = "2. Читать и записывать сведения об окружающем мире на языке математики";

// Устанавливаем выравнивание текста в объединенных ячейках

Excel.Range headerRow = worksheet.Range[worksheet.Cells[1, 1], worksheet.Cells[1, 12]];

headerRow.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

headerRow.VerticalAlignment = Excel.XlVAlign.xlVAlignCenter;

headerRow.WrapText = true;

// Устанавливаем высоту строки для строк с надписями

worksheet.Rows[1].RowHeight = 100;

// Добавляем рамки вокруг верхней строки

Excel.Range topRowRange = worksheet.Range[worksheet.Cells[1, 1], worksheet.Cells[1, 12]];

topRowRange.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

topRowRange.Borders.Color = System.Drawing.Color.Black.ToArgb();

// Разделяем ячейку "Фамилия" по диагонали

Excel.Range surnameCell = worksheet.Cells[2, 1];

// Добавляем границы для ячейки

surnameCell.Borders[Excel.XlBordersIndex.xlEdgeBottom].LineStyle = Excel.XlLineStyle.xlContinuous;

surnameCell.Borders[Excel.XlBordersIndex.xlEdgeRight].LineStyle = Excel.XlLineStyle.xlContinuous;

surnameCell.Borders.Color = System.Drawing.Color.Black.ToArgb();

// Разделяем ячейку "Фамилия" по диагонали

surnameCell.Borders[Excel.XlBordersIndex.xlDiagonalDown].LineStyle = Excel.XlLineStyle.xlContinuous; // Диагональная линия

// Настроим текст для верхней части (слева)

surnameCell.Value = "Fam"; // Устанавливаем текст

surnameCell.HorizontalAlignment = Excel.XlHAlign.xlHAlignLeft; // Выравнивание по центру

surnameCell.VerticalAlignment = Excel.XlVAlign.xlVAlignTop; // Верхнее выравнивание

surnameCell.Font.Size = 8; // Размер шрифта

// Настроим текст для нижней части (справа)

Excel.Range bottomPart = worksheet.Cells[2, 1]; // Это та же ячейка, что и surnameCell

bottomPart.Value = "Умения"; // Устанавливаем текст

bottomPart.HorizontalAlignment = Excel.XlHAlign.xlHAlignRight; // Выравнивание по центру

bottomPart.VerticalAlignment = Excel.XlVAlign.xlVAlignBottom; // Нижнее выравнивание

bottomPart.Font.Size = 8; // Размер шрифта

// Установим цвет для диагональной линии

surnameCell.Borders[Excel.XlBordersIndex.xlDiagonalDown].Color = System.Drawing.Color.Black.ToArgb();

// Изменяем ширину столбца для улучшения видимости текста

worksheet.Columns[1].ColumnWidth = 25; // Увеличиваем ширину столбца для "Фамилия"

// Настроим ширину для других столбцов в верхней строке

worksheet.Columns[2].ColumnWidth = 30; // Для первого столбца заголовка

worksheet.Columns[3].ColumnWidth = 30; // Для второго столбца заголовка

worksheet.Columns[4].ColumnWidth = 30; // Для третьего столбца заголовка

worksheet.Columns[5].ColumnWidth = 30; // Для четвертого столбца заголовка

worksheet.Columns[6].ColumnWidth = 30; // Для пятого столбца заголовка

worksheet.Columns[7].ColumnWidth = 30; // Для шестого столбца заголовка

worksheet.Columns[8].ColumnWidth = 30; // Для седьмого столбца заголовка

worksheet.Columns[9].ColumnWidth = 30; // Для восьмого столбца заголовка

worksheet.Columns[10].ColumnWidth = 30; // Для девятого столбца заголовка

worksheet.Columns[11].ColumnWidth = 30; // Для десятого столбца заголовка

worksheet.Columns[12].ColumnWidth = 30; // Для одиннадцатого столбца заголовка

// Настроим шрифт для верхней части

surnameCell.Font.Size = 8;

surnameCell.VerticalAlignment = Excel.XlVAlign.xlVAlignTop; // Выравнивание для верхней части

// Устанавливаем стиль для ячеек таблицы

worksheet.Cells.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

worksheet.Cells.VerticalAlignment = Excel.XlVAlign.xlVAlignCenter;

// Получение данных из DataGridView и запись в Excel

for (int i = 0; i < dataGridView1.ColumnCount; i++)

{

// Записываем заголовки столбцов в строку 2 (наша таблица начинается со строки 2)

worksheet.Cells[2, i + 1] = dataGridView1.Columns[i].HeaderText;

// Поворот текста в заголовке на 90 градусов

Excel.Range headerCell = worksheet.Cells[2, i + 1];

headerCell.Orientation = 90;

// Включаем перенос текста в заголовке

headerCell.WrapText = true;

// Устанавливаем границы для заголовков

headerCell.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

headerCell.Borders.Color = System.Drawing.Color.Black.ToArgb();

}

// Устанавливаем ограничение по высоте строки заголовков

worksheet.Rows[2].RowHeight = 150;

// Заполнение данных в Excel

for (int row = 0; row < dataGridView1.RowCount; row++)

{

for (int col = 0; col < dataGridView1.ColumnCount; col++)

{

worksheet.Cells[row + 3, col + 1] = dataGridView1.Rows[row].Cells[col].Value?.ToString();

// Включаем перенос текста для данных

Excel.Range dataCell = worksheet.Cells[row + 3, col + 1];

dataCell.WrapText = true;

// Устанавливаем границы для данных

dataCell.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

dataCell.Borders.Color = System.Drawing.Color.Black.ToArgb();

}

}

// Автоматическая настройка высоты строк для корректного отображения текста

worksheet.Rows.AutoFit();

worksheet.Columns.AutoFit();

// Отображение Excel

excelApp.Visible = true;

// Освобождение ресурсов

workbook = null;

worksheet = null;

}

}

}

1. **Графический пользовательский интерфейс программного обеспечения и его описание**

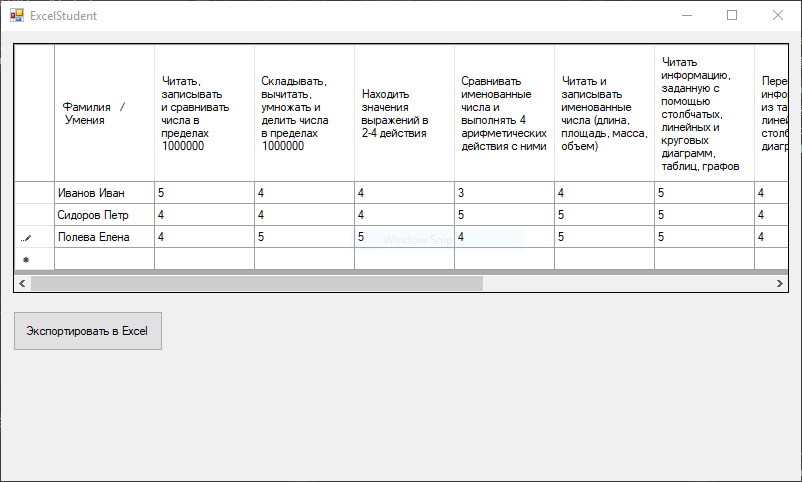


Рисунок 3 – Интерфейс главной формы

На форме присутствует таблица, подписанная «Предпросмотр». В этой таблице отображается загруженная информация из csv файла. По кнопке «Выгрузка в Excel» происходит экспорт данных.

1. **Расчёт тестовых примеров с использованием составленного программного обеспечения**

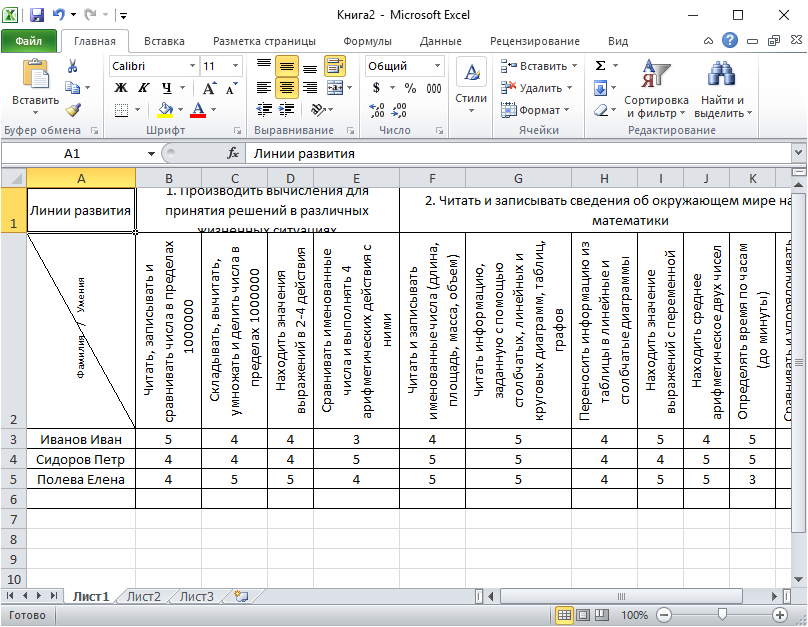


Рисунок 5 – Созданный автоматически файл формата xlsx

1. **Формулировка вывода о проделанной работе**

Освоил навыки подключения внешних модулей и их использования в разрабатываемых программах в интегрированной среде разработки Microsoft Visual Studio.

Освоил навыки программируемой настройки параметров электронных таблиц Microsoft Office Excel.