НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Дисциплина: «Программирование»

Выполнил: Кожакин Кирилл,

студент группы БПИ173.

Преподаватели: Максименкова О. В., Подбельский В.В..

Контрольное домашнее задание 2

Вариант 5

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Оглавление

1. Условие задачи 3
2. Функции разрабатываемого приложения 4
   1. Варианты использования 4
   2. Описание интерфейса пользователя 4
3. Структура приложения 6
   1. Диаграмма классов 6
   2. Описание классов, их полей и методов 6
4. Распределение исходного кода по файлам проекта 9
5. Контрольный пример и описание результатов 9
6. Список литературы 11

Приложение (Текст программы) 11

# Условие задачи

Вариант 5

**1. Общие условия для всех вариантов**

Предложенный в варианте csv-файл содержит «сырые» данные, которые не могут напрямую использоваться с ИСС. Создать класс **Jarvis**, который будет координировать работу всех остальных классов индивидуального задания. Правила управления классами следует описать в пояснительной записке.

Класс **CSVProcessor** отвечает за разбор csv-файла. Строки, содержащие пропущенные значения (NA) требуется отображать в интерфейсе, но исключать при статистической обработке. Некорректные значения следует заменить маркером Error и также исключать при статистической обработке. При сортировке пропущенные или ошибочные значения следует группировать вначале или в конце списка.

**Придерживайтесь принципов объектно-ориентированной разработки, избегайте смешений парадигм.**

Приложение в обязательном порядке независимо от предметной области, указанной в задании, должно выполнять следующие стандартные операции со списками объектов:

1. создание списка путем ввода данных пользователем;
2. добавление в список нового объекта (параметры задаются пользователем);
3. удаление объекта из списка;
4. вывод сведений об объектах заданного пользователем типа;
5. сохранение списка объектов в указанном пользователем csv файле в режиме:
   1. сохранения в файле текущего списка объектов
   2. добавления в файл текущего списка объектов
6. построение списка объектов по данным, прочитанным из файла.

Список объектов может быть реализован в виде массива или какой-либо коллекции, например, списка. Количество создаваемых объектов заранее неизвестно. Порядок следования объектов разных типов (если такие есть) при создании списка произвольный и определяется пользователем.

Приложение обязательно должно корректно открывать и позволять модифицировать, созданные с его помощью файлы с данными.

Исходные файлы с данными могут располагаться в любой папке компьютера.

Приложение должно корректно работать с файлами, в пути к которым содержатся символы национальных алфавитов, знаки препинания, # и проч., допустимые для путей к файлам в ОС Windows символы.

Приложение должно корректно обрабатывать попытки загрузить файлы с нарушенным форматом.

Исключения, возникающие в процессе работы приложения, следует обрабатывать, обеспечивая стабильную работу приложения.

**2. Общие требования к интерфейсу программы:**

1. Приложение должно представлять собой оконное Windows приложение.

2. При добавлении данных в ИСС исходные данные пользователь вводит с помощью экранных форм, содержащих поля для текстового ввода или списки значений.

3. При отсутствии явных ограничений в варианте задания, результаты отображаются с помощью экранных форм для вывода текста или элементов типа «сетка данных» (DataGridView или аналогичный).

4. Сообщения о некорректном вводе данных, противоречивых или недопустимых значениях данных и других нештатных ситуациях отображать во всплывающих окнах типа окон сообщений.

5. В приложении в обязательно должны быть реализованы следующие элементы управления: меню управления входным\выходным файлом, панель инструментов для управления ИСС, всплывающие подсказки.

6. Остальные элементы управления используются по усмотрению разработчика.

**3. Индивидуальное задание**

В файле **chickweight.csv** содержатся данные о кормовых рационах цыплят.

Данные из него загружаются в основную таблицу ИСС.

Данные о каждом измерении (строка файла) размещаются в объектах класса Цыплёнок. Статистические характеристики для всех Цыплят вычисляет класс Курятник, находящийся с Цыплятами в отношении агрегации. Класс Курятник предоставляет методы получения групп Цыплят по типам диеты, а также для вычисления моды, медианы и средних значений всех числовых величин, объектов из этих групп. В интерфейсе предусмотреть средства для вывода соответствующих значений.

# Функции разрабатываемого приложения

## Варианты использования

Данное приложение может быть использовано в исследовательских целях. В частности, для изучения самой популярной диеты цыплят или среднего значения параметров цыплят, сидящих на определённой диете для нахождения оптимальной.

## Описание интерфейса пользователя

Окно приложения содержит:

1. Верхнее меню
   1. Загрузить файл (загружает данные из csv файла в таблицу данных);
   2. Сохранить файл (сохраняет данные в csv файл из таблицы данных);
   3. Добавить в файл (добавляет данные в csv файл из таблицы данных);
   4. Очистить таблицу (удаляет все данные из таблицы);
   5. Выход (завершает работу приложения);
2. Панель инструментов
   1. ComboBox (Позволяет выбрать диету и отображать в таблице данных цыплят с этой диетой);
   2. Кнопка «Рассчитать» (позволяет рассчитать среднее значение, моду и медиану параметров цыплят из таблицы данных);
3. Таблица данных (отображает дынные о цыплятах);
4. Таблица значений (отображает результаты вычислений).

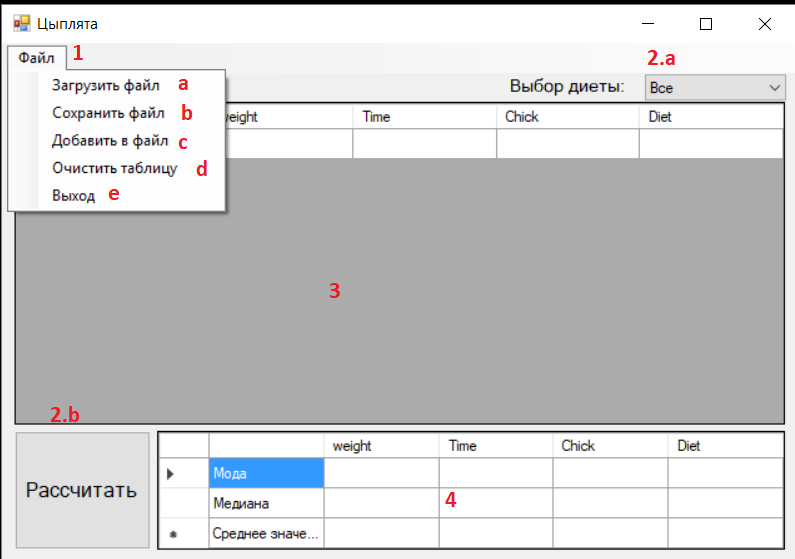


Рис 1 (Интерфейс)

# Структура приложения

## Диаграмма классов

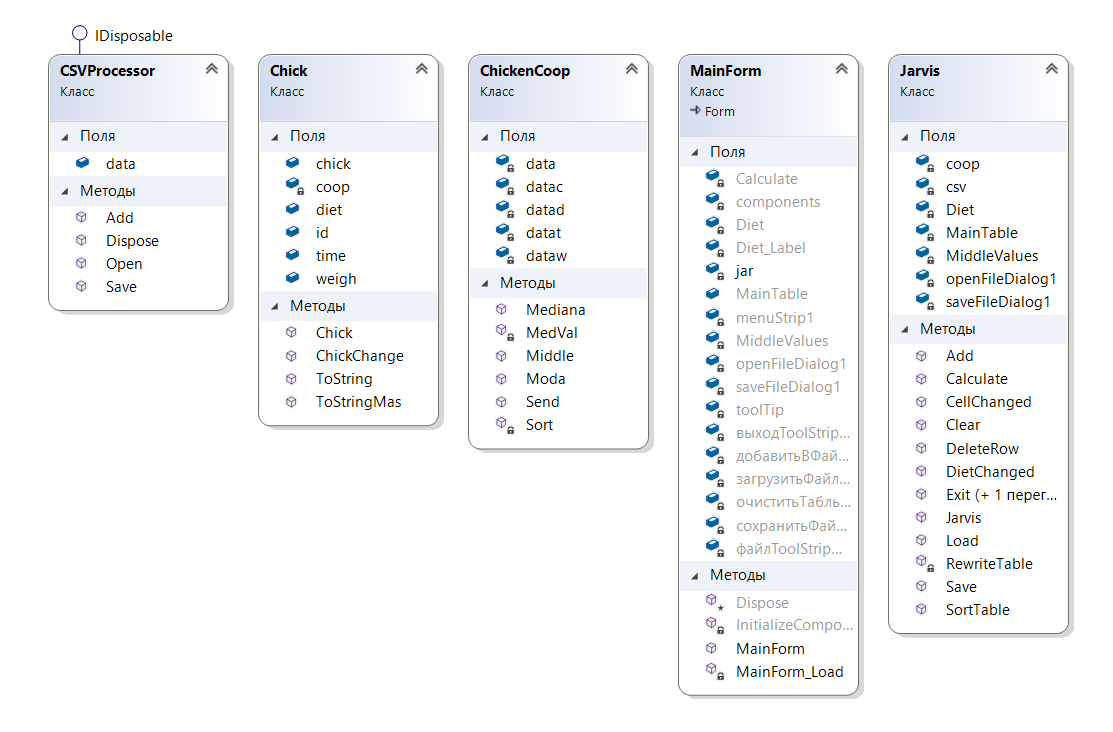


Рис 2 (Диаграмма классов)

## Описание классов, их полей и методов

1. Class CSVProcessor : IDisposable
   1. Поля:
      * *public List<Chick> data* – лист цыплят, хранящий данные после их загрузки из файла.
   2. Методы:
      * *public void Open(string filename, ChickenCoop coop, ComboBox Diet) –* метод, осуществляющий загрузку данные из csv файла в *data*;
      * *public void Save(string filename) -* метод, осуществляющий сохранение данных из *data* в файл csv;
      * *public void Add(string filename)–* метод, осуществляющий добавление данных из *data* в файл csv;
      * *public void Dispose()*- метод, осуществляющий очистку данных из *data*;
2. Class Form1
   1. Поля:
      * *Jarvis jar –* экземпляр управляющего класса;
   2. Методы:
      * *private void MainForm\_Load(object sender, EventArgs e)* - событие, происходящее при загрузке формы и задающее начальные значения объектам интерфейса;
3. Class Chick
   1. Поля:
      * *public string id –* id цыплёнка;
      * *public string weigh –* вес цыплёнка;
      * *public string time –* время диеты цыплёнка;
      * *public string chick –* вид цыплёнка;
      * *public string diet –* вид диеты цыплёнка;
      * *ChickenCoop coop –* курятник, к которому относиться цыплёнок;

**b.** Методы:

* + - *public Chick(string id, string weigh, string time, string chick, string diet, ChickenCoop coop)* – конструктор класса;
    - *public void ChickChange(string s, int name) –* вносит изменения пользователя в лист данных.
    - *public string[] ToStringMas() -* преобразовывает данные в массив строк для записи в таблицу;
    - *public string ToString() -* преобразовывает данные в строку для записи в файл;

1. Class ChickenCoop
   1. Поля:
      * *List<Chick> data –* лист данных для цыплят в этом курятнике;
      * *List<Chick>dataw –* лист данных для цыплят в этом курятнике, отсортированный по весу;
      * *List<Chick>datat –* лист данных для цыплят в этом курятнике, отсортированный по времени;
      * *List<Chick>datac –* лист данных для цыплят в этом курятнике, отсортированный по виду;
      * *List<Chick>datad –* лист данных для цыплят в этом курятнике, отсортированный по диете;

**b.** Методы:

* + - *private void Sort()* – метод, осуществляющий сортировку *data*, и записывает результат в специальные листы;
    - *public void Send(List<Chick> data) –* получает данные о цыплятах;
    - *public string[] Moda(ComboBox Diet) –* рассчитывает моду по каждому параметру при данной диете;
    - *public string[] Mediana(ComboBox Diet)* – рассчитывает медиану по каждому параметру при данной диете;
    - *public string[] Middle(ComboBox Diet) –* рассчитывает среднее значение по каждому параметру при данной диете;
    - *private string MedVal(int i, int count, List<Chick> d) –* вспомогательный метод для вычисления медианы;

1. Class Jarvis
   1. Поля:
      * *DataGridView MainTable –* таблица данных;
      * *DataGridView MiddleValues –* таблица результатов*;*
      * *OpenFileDialog openFileDialog1 –* диалог открытия файла;
      * *SaveFileDialog saveFileDialog1 –* диалог сохранения файла*;*
      * *CSVProcessor csv –* обработчик файлов csv;
      * *ChickenCoop coop –* курятник;
      * *ComboBox Diet –* диета;

**b.** Методы:

* + - *public Jarvis(DataGridView MainTable, DataGridView MiddleValues, OpenFileDialog o, SaveFileDialog s, ComboBox Diet) –* конструктор;
    - *private void RewriteTable(int q = -1) –* перезаписывает таблицу данных в завистимости от указанной диеты;
    - *public void Load(object sender, EventArgs e) –* событие, возникающее при попытке загрузки файла;
    - *public void Save(object sender, EventArgs e) –* событие, возникающее при попытке сохранения данных в файл;
    - *public void Add(object sender, EventArgs e) –* событие, возникающее при попытке добавления данных к файлу;
    - *public void Calculate(object sender, EventArgs e) –* событие, возникающее при нажатии на кнопку «Рассчитать»;
    - *public void DeleteRow(object sender, KeyEventArgs e) –* событие, возникающее при удалении строки из таблицы;
    - *public void CellChanged(object sender, DataGridViewCellEventArgs e) –* событие, возникающее при изменении клетки таблицы данных;
    - *public void DietChanged(object sender, EventArgs e) –* событие, возникающее при изменении значения *Diet*;
    - *public void Clear(object sender, EventArgs e) –* событие, возникающее при нажатии пункта меню «Очистить таблицу»;
    - *public void Exit(object sender, FormClosingEventArgs e), public void Exit(object sender, EventArgs e) –* события*,* возникающие при выходе из приложения (1 – кнопка «Х», 2 – пункт меню «Выход»);
    - *public void SortTable(object sender, DataGridViewSortCompareEventArgs e) –* события, возникающее при попытке сортировки таблицы данных.

# Распределение исходного кода по файлам проекта

* Form1.cs. Включает класс Form1, в котором задаются начальные значения объектов интерфейса и подписка на события класса Jarvis;
* Jarvis.cs. Включает класс Jarvis, который управляет почти всеми событиями приложения.
* Chick.cs. Включает класс Chick, который содержит информацию о конкретном цыпленке в курятнике.
* ChickenCoop.cs. Включает класс ChickenCoop, который содержит информацию о наборе цыплят и производит расчёты по их параметрам.
* CSVProcessor.cs. Включает класс CSVProcessor, который производит работу с csv файлами.

# Контрольный пример и описание результатов

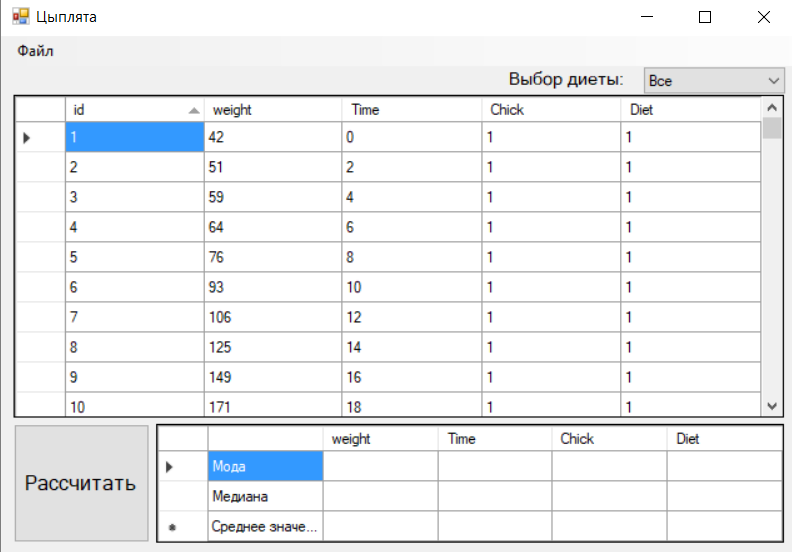


Рис 3 (Запуск программы и загрузка файла из примера)

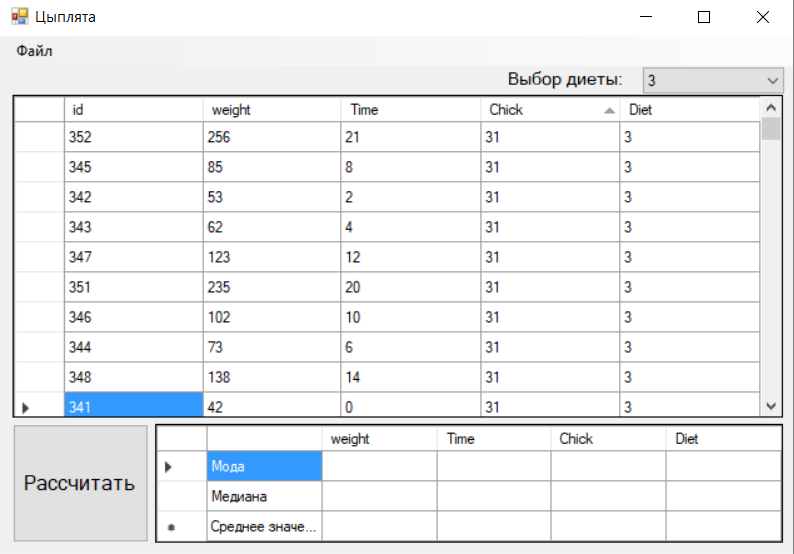


Рис 4 (Выбрана диета 3 и отсортировано по параметру Chick)

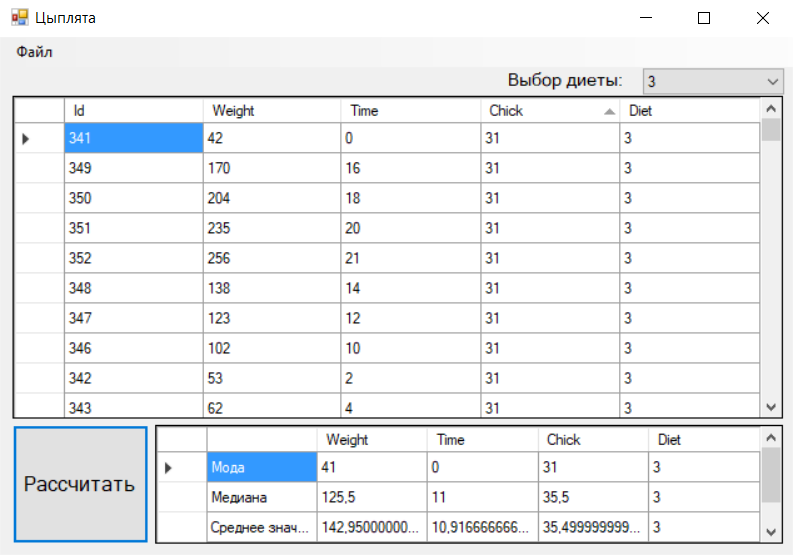


Рис 5 (Нажатие кнопки рассчитать)

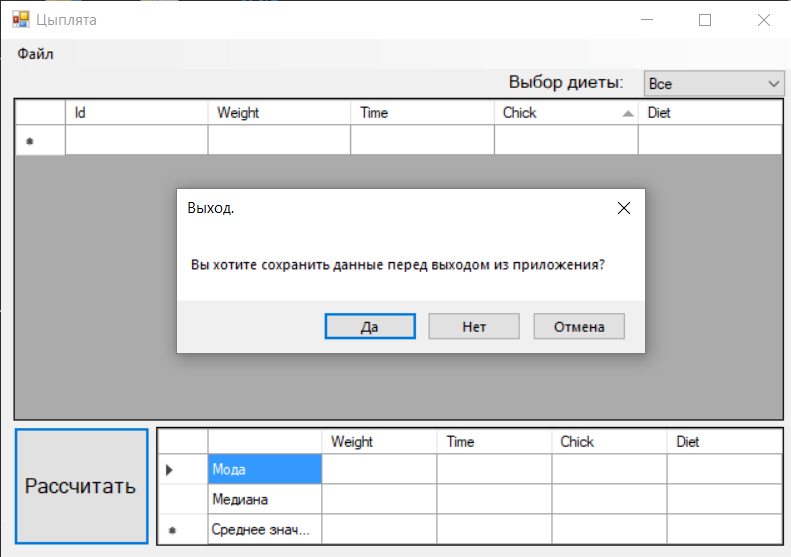


Рис 6 (Нажатие кнопки «Х» или пункта меню «Выход»)

# Список литературы

* [https://docs.microsoft.com](https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/)

# Приложение (Текст программы)

Файл Form1.cs

/\*

Кожакин Кирилл Геннадьевич, группа БПИ173.

Дисциплина "Программирование".

Контрольное домашнее задание. Модуль 3. 2017-2018.

Контрольное домашнее задание 2.

Вариант 5.

Предложенный в варианте csv-файл содержит «сырые» данные, которые не могут напрямую использоваться с ИСС.

Создать класс Jarvis, который будет координировать работу всех остальных классов индивидуального задания.

Правила управления классами следует описать в пояснительной записке.

Класс CSVProcessor отвечает за разбор csv-файла. Строки, содержащие пропущенные значения (NA) требуется отображать в интерфейсе,

но исключать при статистической обработке. Некорректные значения следует заменить маркером Error и также исключать при статистической обработке.

При сортировке пропущенные или ошибочные значения следует группировать вначале или в конце списка.

Список объектов может быть реализован в виде массива или какой-либо коллекции, например, списка. Количество создаваемых объектов заранее неизвестно.

Порядок следования объектов разных типов (если такие есть) при создании списка произвольный и определяется пользователем.

Приложение обязательно должно корректно открывать и позволять модифицировать, созданные с его помощью файлы с данными.

Исходные файлы с данными могут располагаться в любой папке компьютера.

Приложение должно корректно работать с файлами, в пути к которым содержатся символы национальных алфавитов, знаки препинания, # и проч.,

допустимые для путей к файлам в ОС Windows символы.

Приложение должно корректно обрабатывать попытки загрузить файлы с нарушенным форматом.

Исключения, возникающие в процессе работы приложения, следует обрабатывать, обеспечивая стабильную работу приложения.

Данные о каждом измерении (строка файла) размещаются в объектах класса Цыплёнок.

Статистические характеристики для всех Цыплят вычисляет класс Курятник, находящийся с Цыплятами в отношении агрегации.

Класс Курятник предоставляет методы получения групп Цыплят по типам диеты, а также для вычисления моды,

медианы и средних значений всех числовых величин, объектов из этих групп. В интерфейсе предусмотреть средства для вывода соответствующих значений.

\*/

using System;

using System.Windows.Forms;

namespace kdz

{

public partial class MainForm : Form

{

Jarvis jar;

public MainForm()

{

InitializeComponent();

jar = new Jarvis(MainTable, MiddleValues, openFileDialog1, saveFileDialog1, Diet);

загрузитьФайлToolStripMenuItem.Click += jar.Load;

сохранитьФайлToolStripMenuItem.Click += jar.Save;

добавитьВФайлToolStripMenuItem.Click += jar.Add;

MainTable.CellValueChanged += jar.CellChanged;

Diet.SelectionChangeCommitted += jar.DietChanged;

Calculate.Click += jar.Calculate;

очиститьТабльцуToolStripMenuItem.Click += jar.Clear;

MainTable.KeyDown += jar.DeleteRow;

выходToolStripMenuItem.Click += jar.Exit;

FormClosing += jar.Exit;

MainTable.SortCompare += jar.SortTable;

toolTip.SetToolTip(Calculate, "Расчитать моды, медианы и средние значения по данным в таблице.");

toolTip.SetToolTip(Diet, "Диеты.");

}

/// <summary>

/// Загрузка форма и присваивание начальных значений таблицам и полю выбора диет.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

MainTable.ColumnCount = 5;

MainTable.Columns[0].Name = "Id";

MainTable.Columns[1].Name = "Weight";

MainTable.Columns[2].Name = "Time";

MainTable.Columns[3].Name = "Chick";

MainTable.Columns[4].Name = "Diet";

MiddleValues.ColumnCount = 5;

MiddleValues.Columns[1].Name = "Weight";

MiddleValues.Columns[2].Name = "Time";

MiddleValues.Columns[3].Name = "Chick";

MiddleValues.Columns[4].Name = "Diet";

MiddleValues.RowCount = 3;

MiddleValues.Rows[0].Cells[0].Value = "Мода";

MiddleValues.Rows[1].Cells[0].Value = "Медиана";

MiddleValues.Rows[2].Cells[0].Value = "Среднее значение";

Diet.SelectedItem = Diet.Items[0];

}

}

}

Файл CSVProcessor.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

using System.Windows.Forms;

namespace kdz

{ /// <summary>

/// Класс, обрабатывающий csv файлы.

/// </summary>

class CSVProcessor : IDisposable

{

/// <summary>

/// Лист цыплят.

/// </summary>

public List<Chick> data = new List<Chick> { };

/// <summary>

/// Метод загрузки csv файла.

/// </summary>

/// <param name="filename">Имя файла</param>

/// <param name="coop">Курятник</param>

/// <param name="Diet">Диета</param>

public void Open(string filename, ChickenCoop coop, ComboBox Diet)

{

try

{

StreamReader fs = new StreamReader(filename);

string s = "";

string[] v;

int i;

s = fs.ReadLine();

try

{

while (s != null)

{

s = fs.ReadLine();

if (s != null)

{

s = s.Replace("\"", "");

v = s.Split(',');

data.Add(new Chick(v[0] == "" ? "NA" : v[0], v[1] == "" ? "NA" : v[1],

v[2] == "" ? "NA" : v[2], v[3] == "" ? "NA" : v[3], v[4] == "" ? "NA" : v[4], coop));

if (int.TryParse(v[4], out i) && !Diet.Items.Contains(v[4])) Diet.Items.Add(v[4]);

}

}

}

catch(Exception)

{

MessageBox.Show("Файл имеет неверный формат");

data.Clear();

}

fs.Close();

}

catch(Exception e)

{

MessageBox.Show(e.Message);

}

}

/// <summary>

/// Метод сохранения данных в csv файл.

/// </summary>

/// <param name="filename">Имя файла</param>

public void Save(string filename)

{

try

{

StreamWriter sw = new StreamWriter(filename, false, Encoding.Unicode);

sw.Write("Id,Weigh,Time,Chick,Diet");

sw.WriteLine();

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

{

sw.Write(data[i].ToString());

sw.Write("\r\n");

}

sw.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Метод добавления данных к csv файлн.

/// </summary>

/// <param name="filename">Имя файла</param>

public void Add(string filename)

{

try

{

StreamWriter sw = new StreamWriter(new FileStream(filename, FileMode.Append, FileAccess.Write), Encoding.Unicode);

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

{

sw.Write(data[i].ToString());

sw.Write("\r\n");

}

sw.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Метод очистки данных.

/// </summary>

public void Dispose()

{

data.Clear();

}

}

}

Файл Chick.cs

namespace kdz

{

class Chick

{

public string id, weigh, time, chick, diet;

ChickenCoop coop;

/// <summary>

/// Конструктор цыплёнка.

/// </summary>

/// <param name="id">id</param>

/// <param name="weigh">вес</param>

/// <param name="time">время</param>

/// <param name="chick">вид</param>

/// <param name="diet">диета</param>

/// <param name="coop">курятник</param>

public Chick(string id, string weigh, string time, string chick, string diet, ChickenCoop coop)

{

int i;

if (!int.TryParse(id, out i) & id != "NA") this.id = "Error"; else this.id = id;

if (!int.TryParse(weigh, out i) & weigh != "NA") this.weigh = "Error"; else this.weigh = weigh;

if (!int.TryParse(time, out i) & time != "NA") this.time = "Error"; else this.time = time;

if (!int.TryParse(chick, out i) & chick != "NA") this.chick = "Error"; else this.chick = chick;

if (!int.TryParse(diet, out i) & diet != "NA") this.diet = "Error"; else this.diet = diet;

this.coop = coop;

}

/// <summary>

/// Измене данных экземпляры.

/// </summary>

/// <param name="s">значение параметра</param>

/// <param name="name">имя параметра</param>

public void ChickChange(string s, int name)

{

int i;

switch (name)

{

case 0:

if (!int.TryParse(s, out i) & s != "NA") id = "Error"; else id = s;

break;

case 1:

if (!int.TryParse(s, out i) & s != "NA") weigh = "Error"; else weigh = s;

break;

case 2:

if (!int.TryParse(s, out i) & s != "NA") time = "Error"; else time = s;

break;

case 3:

if (!int.TryParse(s, out i) & s != "NA") chick = "Error"; else chick = s;

break;

case 4:

if (!int.TryParse(s, out i) & s != "NA") diet = "Error"; else diet = s;

break;

}

}

/// <summary>

/// Преобразовывает данные для записи в таблицу.

/// </summary>

/// <returns>Массив параметров</returns>

public string[] ToStringMas()

{

string[] s = new string[5] { id == "NA" ? "" : id, weigh == "NA" ? "" : weigh,

time == "NA" ? "" : time, chick == "NA" ? "" : chick, diet == "NA" ? "" : diet };

return s;

}

/// <summary>

/// Преобразовывает данные для записи в файл.

/// </summary>

/// <returns>Строку параметров.</returns>

public override string ToString()

{

return id + "," + weigh + "," + time + "," + chick + "," + diet;

}

}

}

Файл ChickenCoop.cs

using System.Collections.Generic;

using System.Windows.Forms;

namespace kdz

{

class ChickenCoop

{

List<Chick> data, dataw, datat, datac, datad;

/// <summary>

/// Сортировка данных по всем параметрам кроме id.

/// </summary>

private void Sort()

{

dataw = data.GetRange(0, data.Count);

datat = data.GetRange(0, data.Count);

datac = data.GetRange(0, data.Count);

datad = data.GetRange(0, data.Count);

dataw.Sort(delegate (Chick x, Chick y)

{

if (x.weigh == "NA" || x.weigh == "Error") return 1;

else if (y.weigh == "NA" || y.weigh == "Error") return -1;

else if (int.Parse(y.weigh) < int.Parse(x.weigh)) return 1;

else if (int.Parse(y.weigh) > int.Parse(x.weigh)) return -1;

return 0;

});

datat.Sort(delegate (Chick x, Chick y)

{

if (x.time == "NA" || x.time == "Error") return 1;

else if (y.time == "NA" || y.time == "Error") return -1;

else if (int.Parse(y.time) < int.Parse(x.time)) return 1;

else if (int.Parse(y.time) > int.Parse(x.time)) return -1;

return 0;

});

datac.Sort(delegate (Chick x, Chick y)

{

if (x.chick == "NA" || x.chick == "Error") return 1;

else if (y.chick == "NA" || y.chick == "Error") return -1;

else if (int.Parse(y.chick) < int.Parse(x.chick)) return 1;

else if (int.Parse(y.chick) > int.Parse(x.chick)) return -1;

return 0;

});

datad.Sort(delegate (Chick x, Chick y)

{

if (x.diet == "NA" || x.diet == "Error") return 1;

else if (y.diet == "NA" || y.diet == "Error") return -1;

else if (int.Parse(y.diet) < int.Parse(x.diet)) return 1;

else if (int.Parse(y.diet) > int.Parse(x.diet)) return -1;

return 0;

});

}

/// <summary>

/// Функция получения данных.

/// </summary>

/// <param name="data"></param>

public void Send(List<Chick> data)

{

this.data = data;

Sort();

}

/// <summary>

/// Вычисление моды.

/// </summary>

/// <param name="Diet">диета</param>

/// <returns>Массив мод по параметрам</returns>

public string[] Moda(ComboBox Diet)

{

string[] moda = new string[5] {"Мода", "NA", "NA", "NA", "NA" };

int[] col, col\_max = new int[4] { 0, 0, 0, 0 };

if (Diet.Text == "Все")

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

{

col = new int[4] { 0, 0, 0, 0 };

for (int j = i; j < data.Count; j++)

{

if (dataw[i].weigh != "Error" && dataw[i].weigh != "NA" && dataw[i].weigh == dataw[j].weigh) col[0]++;

if (datat[i].time != "Error" && datat[i].time != "NA" && datat[i].time == datat[j].time) col[1]++;

if (datac[i].chick != "Error" && datac[i].chick != "NA" && datac[i].chick == datac[j].chick) col[2]++;

if (datad[i].diet != "Error" && datad[i].diet != "NA" && datad[i].diet == datad[j].diet) col[3]++;

}

if (col[0] > col\_max[0])

{

col\_max[0] = col[0];

moda[1] = dataw[i].weigh;

}

if (col[1] > col\_max[1])

{

col\_max[1] = col[1];

moda[2] = datat[i].time;

}

if (col[2] > col\_max[2])

{

col\_max[2] = col[2];

moda[3] = datac[i].chick;

}

if (col[3] > col\_max[3])

{

col\_max[3] = col[3];

moda[4] = datad[i].diet;

}

}

else

{

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

{

col = new int[3] { 1, 1, 1 };

for (int j = i; j < data.Count; j++)

{

if (dataw[i].weigh != "Error" && dataw[i].weigh != "NA" && dataw[i].weigh == dataw[j].weigh && dataw[i].diet == Diet.Text && dataw[j].diet == Diet.Text) col[0]++;

if (datat[i].time != "Error" && datat[i].time != "NA" && datat[i].time == datat[j].time && datat[i].diet == Diet.Text && datat[j].diet == Diet.Text) col[1]++;

if (datac[i].chick != "Error" && datac[i].chick != "NA" && datac[i].chick == datac[j].chick && datac[i].diet == Diet.Text && datac[j].diet == Diet.Text) col[2]++;

}

if (col[0] > col\_max[0])

{

col\_max[0] = col[0];

moda[1] = dataw[i].weigh;

}

if (col[1] > col\_max[1])

{

col\_max[1] = col[1];

moda[2] = datat[i].time;

}

if (col[2] > col\_max[2])

{

col\_max[2] = col[2];

moda[3] = datac[i].chick;

}

}

moda[4] = Diet.Text;

}

return moda;

}

/// <summary>

/// Вычисление медианы.

/// </summary>

/// <param name="Diet">диета</param>

/// <returns>Массив медиан по параметрам</returns>

public string[] Mediana(ComboBox Diet)

{

string[] mediana = new string[5] { "Медиана", "0", "0", "0", "0" };

List<Chick> d = new List<Chick> { };

if (Diet.Text == "Все")

{

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

if (dataw[i].weigh != "Error" && dataw[i].weigh != "NA") d.Add(dataw[i]);

mediana[1] = d.Count > 0 ? MedVal(1, d.Count, d) : "NA";

d = new List<Chick> { };

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

if (datat[i].time != "Error" && datat[i].time != "NA") d.Add(datat[i]);

mediana[2] = d.Count > 0 ? MedVal(2, d.Count, d) : "NA";

d = new List<Chick> { };

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

if (datac[i].chick != "Error" && datac[i].chick != "NA") d.Add(datac[i]);

mediana[3] = d.Count > 0 ? MedVal(3, d.Count, d) : "NA";

d = new List<Chick> { };

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

if (datad[i].diet != "Error" && datad[i].diet != "NA") d.Add(datad[i]);

mediana[4] = d.Count > 0 ? MedVal(4, d.Count, d) : "NA";

}

else

{

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

if (dataw[i].weigh != "Error" && dataw[i].weigh != "NA" && dataw[i].diet == Diet.Text) d.Add(dataw[i]);

mediana[1] = d.Count > 0 ? MedVal(1, d.Count, d) : "NA";

d = new List<Chick> { };

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

if (datat[i].time != "Error" && datat[i].time != "NA" && datat[i].diet == Diet.Text) d.Add(datat[i]);

mediana[2] = d.Count > 0 ? MedVal(2, d.Count, d) : "NA";

d = new List<Chick> { };

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

if (datac[i].chick != "Error" && datac[i].chick != "NA" && datac[i].diet == Diet.Text) d.Add(datac[i]);

mediana[3] = d.Count > 0 ? MedVal(3, d.Count, d) : "NA";

mediana[4] = Diet.Text;

}

return mediana;

}

/// <summary>

/// Вычисление Среднего значения.

/// </summary>

/// <param name="Diet">диета</param>

/// <returns>Массив средних значений по параметрам</returns>

public string[] Middle(ComboBox Diet)

{

string[] middle = new string[5] { "Среднее значение", "0", "0", "0", "0" };

int[] col = new int[4] { 0, 0, 0, 0 };

if (Diet.Text == "Все")

{

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

{

if (data[i].weigh != "Error" && data[i].weigh != "NA")

middle[1] = ((double.Parse(middle[1]) \* col[0] + double.Parse(data[i].weigh)) / ++col[0]).ToString();

if (data[i].time != "Error" && data[i].time != "NA")

middle[2] = ((double.Parse(middle[2]) \* col[1] + double.Parse(data[i].time)) / ++col[1]).ToString();

if (data[i].chick != "Error" && data[i].chick != "NA")

middle[3] = ((double.Parse(middle[3]) \* col[2] + double.Parse(data[i].chick)) / ++col[2]).ToString();

if (data[i].diet != "Error" && data[i].diet != "NA")

middle[4] = ((double.Parse(middle[4]) \* col[3] + double.Parse(data[i].diet)) / ++col[3]).ToString();

}

middle[1] = col[0] > 0 ? middle[1]:"NA";

middle[2] = col[1] > 0 ? middle[2]:"NA";

middle[3] = col[2] > 0 ? middle[3]:"NA";

middle[4] = col[3] > 0 ? middle[4]:"NA";

}

else

{

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

if (data[i].diet == Diet.Text)

{

if (data[i].weigh != "Error" && data[i].weigh != "NA")

middle[1] = ((double.Parse(middle[1]) \* col[0] + double.Parse(data[i].weigh)) / ++col[0]).ToString();

if (data[i].time != "Error" && data[i].time != "NA")

middle[2] = ((double.Parse(middle[2]) \* col[1] + double.Parse(data[i].time)) / ++col[1]).ToString();

if (data[i].chick != "Error" && data[i].chick != "NA")

middle[3] = ((double.Parse(middle[3]) \* col[2] + double.Parse(data[i].chick)) / ++col[2]).ToString();

}

middle[1] = col[0] > 0 ? middle[1] : "NA";

middle[2] = col[1] > 0 ? middle[2] : "NA";

middle[3] = col[2] > 0 ? middle[3] : "NA";

middle[4] = Diet.Text;

}

return middle;

}

/// <summary>

/// Вспомогательный метод для вычисления медианы.

/// </summary>

/// <param name="i">имя параметра</param>

/// <param name="count">количество элементов</param>

/// <param name="d">лист элементов</param>

/// <returns>медиана</returns>

private string MedVal(int i, int count, List<Chick> d)

{

switch (i)

{

case 1:

return ((int.Parse(d[count / 2].weigh) + int.Parse(d[(count - 1) / 2].weigh)) / 2.0).ToString();

case 2:

return ((int.Parse(d[count / 2].time) + int.Parse(d[(count - 1) / 2].time)) / 2.0).ToString();

case 3:

return ((int.Parse(d[count / 2].chick) + int.Parse(d[(count - 1) / 2].chick)) / 2.0).ToString();

case 4:

return ((int.Parse(d[count / 2].diet) + int.Parse(d[(count - 1) / 2].diet)) / 2.0).ToString();

default:

return "";

}

}

}

}

Файл Jarvis.cs

using System;

using System.Windows.Forms;

namespace kdz

{

/// <summary>

/// Класс управления.

/// </summary>

class Jarvis

{

DataGridView MainTable, MiddleValues;

OpenFileDialog openFileDialog1;

SaveFileDialog saveFileDialog1;

CSVProcessor csv;

ChickenCoop coop = new ChickenCoop();

ComboBox Diet;

/// <summary>

/// Конструктор.

/// </summary>

/// <param name="MainTable">таблица данных</param>

/// <param name="MiddleValues">таблица результатов</param>

/// <param name="o">диалог загрузки файла</param>

/// <param name="s">диалог сохранения файла</param>

/// <param name="Diet">диета</param>

public Jarvis(DataGridView MainTable, DataGridView MiddleValues, OpenFileDialog o, SaveFileDialog s, ComboBox Diet)

{

this.MainTable = MainTable;

this.MiddleValues = MiddleValues;

openFileDialog1 = o;

saveFileDialog1 = s;

this.Diet = Diet;

csv = new CSVProcessor();

}

/// <summary>

/// Перезапись таблицы данных.

/// </summary>

/// <param name="q">диета</param>

private void RewriteTable(int q = -1)

{

MainTable.Rows.Clear();

for (int i = 0; i < csv.data.Count; i++)

{

if (q < 0) MainTable.Rows.Add(csv.data[i].ToStringMas());

else if (q.ToString() == csv.data[i].diet) MainTable.Rows.Add(csv.data[i].ToStringMas());

}

}

/// <summary>

/// Событие загрузки файла.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void Load(object sender, EventArgs e)

{

openFileDialog1.Filter = "CSV Files (\*.csv)|\*.csv";

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

return;

// получаем выбранный файл

string filename = openFileDialog1.FileName;

csv = new CSVProcessor();

coop = new ChickenCoop();

csv.Open(filename, coop, Diet);

RewriteTable();

}

/// <summary>

/// Событие сохранения файла.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void Save(object sender, EventArgs e)

{

saveFileDialog1.Filter = "CSV Files (\*.csv)|\*.csv";

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

return;

string filename = saveFileDialog1.FileName;

csv.Save(filename);

}

/// <summary>

/// Событие добавления данных к файлу.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void Add(object sender, EventArgs e)

{

saveFileDialog1.Filter = "CSV Files (\*.csv)|\*.csv";

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

return;

string filename = saveFileDialog1.FileName;

csv.Add(filename);

}

/// <summary>

/// Событие удаления строки.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void DeleteRow(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Delete || e.KeyCode == Keys.Back)

for (int i = 0; i < MainTable.SelectedRows.Count; i++)

csv.data.RemoveAt(MainTable.SelectedRows[i].Index);

}

/// <summary>

/// Событие изменения клетки таблицы.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void CellChanged(object sender, DataGridViewCellEventArgs e)

{

int y = e.RowIndex;

int x = e.ColumnIndex;

int i;

object o = MainTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value;

if (o != null && !int.TryParse(o.ToString(), out i) && i >= 0)

{

MainTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value = "Error";

o = "Error";

}

else if (o == null) o = "NA";

string s = o.ToString();

if (y > csv.data.Count - 1)

switch (x)

{

case 0:

csv.data.Add(new Chick(s, "NA", "NA", "NA", "NA", coop));

break;

case 1:

csv.data.Add(new Chick("NA", s, "NA", "NA", "NA", coop));

break;

case 2:

csv.data.Add(new Chick("NA", "NA", s, "NA", "NA", coop));

break;

case 3:

csv.data.Add(new Chick("NA", "NA", "NA", s, "NA", coop));

break;

case 4:

csv.data.Add(new Chick("NA", "NA", "NA", "NA", s, coop));

break;

}

else csv.data[y].ChickChange(s, x);

if (x == 4 && s != "Error" && !Diet.Items.Contains(s)) Diet.Items.Add(s);

}

/// <summary>

/// Событие изменения диеты.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void DietChanged(object sender, EventArgs e)

{

if (Diet.SelectedItem.ToString() == "Все") RewriteTable();

else RewriteTable(int.Parse(Diet.SelectedItem.ToString()));

}

/// <summary>

/// Событие расчёта мод, медиан и средних значений.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void Calculate(object sender, EventArgs e)

{

coop.Send(csv.data);

MiddleValues.Rows.Clear();

MiddleValues.Rows.Add(coop.Moda(Diet));

MiddleValues.Rows.Add(coop.Mediana(Diet));

MiddleValues.Rows.Add(coop.Middle(Diet));

}

/// <summary>

/// Событие очищения таблицы и удаления данных.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void Clear(object sender, EventArgs e)

{

MainTable.Rows.Clear();

MiddleValues.Rows.Clear();

MiddleValues.RowCount = 3;

MiddleValues.Rows[0].Cells[0].Value = "Мода";

MiddleValues.Rows[1].Cells[0].Value = "Медиана";

MiddleValues.Rows[2].Cells[0].Value = "Среднее значение";

Diet.Items.Clear();

Diet.Items.Add("Все");

Diet.SelectedItem = Diet.Items[0];

csv.Dispose();

}

/// <summary>

/// Событие выхода по нажатию "X".

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void Exit(object sender, FormClosingEventArgs e)

{

DialogResult ans = MessageBox.Show("Вы хотите сохранить данные перед выходом из приложения?", "Выход.", MessageBoxButtons.YesNoCancel);

if (ans == DialogResult.Cancel)

return;

else if (DialogResult.No == ans)

Application.ExitThread();

else

{

saveFileDialog1.Filter = "CSV Files (\*.csv)|\*.csv";

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

return;

string filename = saveFileDialog1.FileName;

csv.Save(filename);

Application.ExitThread();

}

}

/// <summary>

/// Событие выхода по нажатию пункта меню "Выход".

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void Exit(object sender, EventArgs e)

{

DialogResult ans = MessageBox.Show("Вы хотите сохранить данные перед выходом из приложения?", "Выход.", MessageBoxButtons.YesNoCancel);

if (ans == DialogResult.Cancel)

return;

else if (DialogResult.No == ans)

Application.ExitThread();

else

{

saveFileDialog1.Filter = "CSV Files (\*.csv)|\*.csv";

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

return;

string filename = saveFileDialog1.FileName;

csv.Save(filename);

Application.ExitThread();

}

}

/// <summary>

/// Событие сортировки таблицы.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void SortTable(object sender, DataGridViewSortCompareEventArgs e)

{

if (e.CellValue1.ToString() == "" || e.CellValue1.ToString() == "Error") e.SortResult = 1;

else if (e.CellValue2.ToString() == "" || e.CellValue2.ToString() == "Error") e.SortResult = -1;

else if (int.Parse(e.CellValue2.ToString()) < int.Parse(e.CellValue1.ToString())) e.SortResult = 1;

else if (int.Parse(e.CellValue2.ToString()) > int.Parse(e.CellValue1.ToString())) e.SortResult = -1;

else e.SortResult = 0;

e.Handled = true;

}

}

}