Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Институт информационных технологий и анализа данных

наименование института

Допускаю к защите

Руководитель:

А.Б. Столбов

И.О. Фамилия

Обработка файлов на языке C#

Наименование темы

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
к курсовой работе по дисциплине

Программирование на языке высокого уровня

1.001.00.00 ПЗ

обозначение документа

Выполнил студент группы ИСТб 19–2 Солопов Д.Д

шифр подпись Фамилия И.О.

Нормоконтроль Столбов А.Б.

подпись ФамилияИ.О.

Курсовая работа защищена с оценкой:

Иркутск 2020г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

По курсу Программирование на языке высокого уровня

Студенту Солопову Даниилу Дмитриевичу

(фамилия, инициалы)

Тема работы Обработка файлов на языке C#

Исходные данные Вариант 11

Рекомендуемая литература

1. Сосинская С.С. «Обработка файлов»: Методические указания по выполнению курсового проекта» Электронный каталог кафедры.

Графическая часть на – листах.

Дата выдачи задания 09 02 /2020 г.

Задание получил студент Солопов Д.Д.

подпись Фамилия И.О.

Дата представления работы руководителю 21 / 05 /2020 г.

Руководитель курсовой работы Столбов А.Б

подпись Фамилия И.О.

Содержание

[Введение 4](#_Toc41425297)

[Индивидуальное задание 5](#_Toc41425298)

[1 Иерархия форм и функций с описанием их назначения 6](#_Toc41425299)

[2 События форм 9](#_Toc41425300)

[3 Алгоритмы основных событий 11](#_Toc41425301)

[4 Формы 38](#_Toc41425302)

[5 Коды основных событий 41](#_Toc41425303)

[6 Таблица тестов 71](#_Toc41425304)

[7 Результаты тестирования 75](#_Toc41425305)

[Заключение 87](#_Toc41425306)

[Список использованных источников 88](#_Toc41425307)

Введение

Visual C# является реализацией языка C# корпорацией Майкрософт. Visual Studio поддерживает Visual C# с полнофункциональным редактором кода, компилятором, шаблонами проектов, конструкторами, мастерами кода, мощным и простым в использовании отладчиком и многими другими средствами. Библиотека классов .NET Framework предоставляет доступ ко многим службам операционной системы и другим полезным, правильным классам, что существенно ускоряет цикл разработки.

C#–это язык программирования, предназначенный для разработки самых разнообразных приложений. Язык C# прост, строго типизирован и объектно–ориентирован. Благодаря множеству нововведений C# обеспечивает возможность быстрой разработки приложений, но при этом сохраняет выразительность и элегантность, присущую языкам C.

Требуется на основании индивидуального задания разработать иерархию форм и функций с описанием их назначения, Windows–приложение на языке С#, таблицу тестов и оттестировать программу для создания файла, добавления записи, удаления записи, изменения записи, просмотра файла и выполнения двух запросов. Удаление и изменение записи происходит по ее номеру. Способы выполнения действий – согласно индивидуальному заданию. При ошибочных ситуациях (отсутствие записи, отсутствие файла и т.д.) необходимо выдавать сообщения.

Индивидуальное задание

Вариант 11, 1, 1, 1

11. Файл **“Игрушки”** (название игрушки, ее стоимость в руб. и возрастные границы детей, для которых игрушка предназначена – левая и правая).

Обработка следующих запросов:

1) выдать названия самых дорогих игрушек, цена которых не отличается от цены самой дорогой не более чем на заданную величину;

2) цены всех игрушек с заданным названием;

3) имеется ли мяч с заданной ценой, предназначенный детям от 3 до 8 лет;

4) возрастные границы игрушки с заданным названием.

1 Иерархия форм и функций с описанием их назначения

Иерархия показана на рис. 1 и рис. 2.

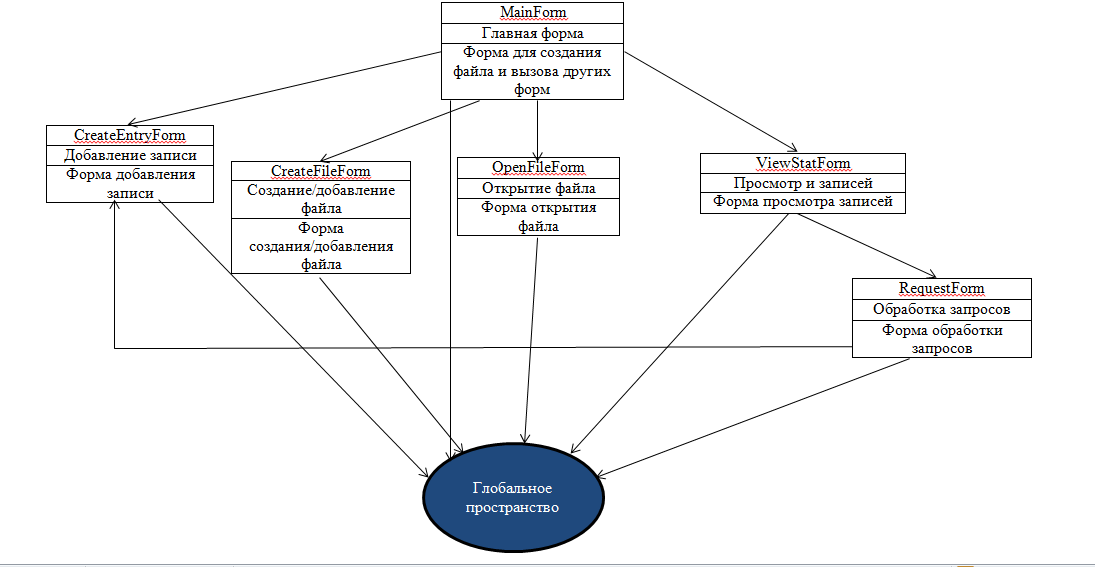


Рисунок 1 – Иерархия форм и функций с описанием их назначения



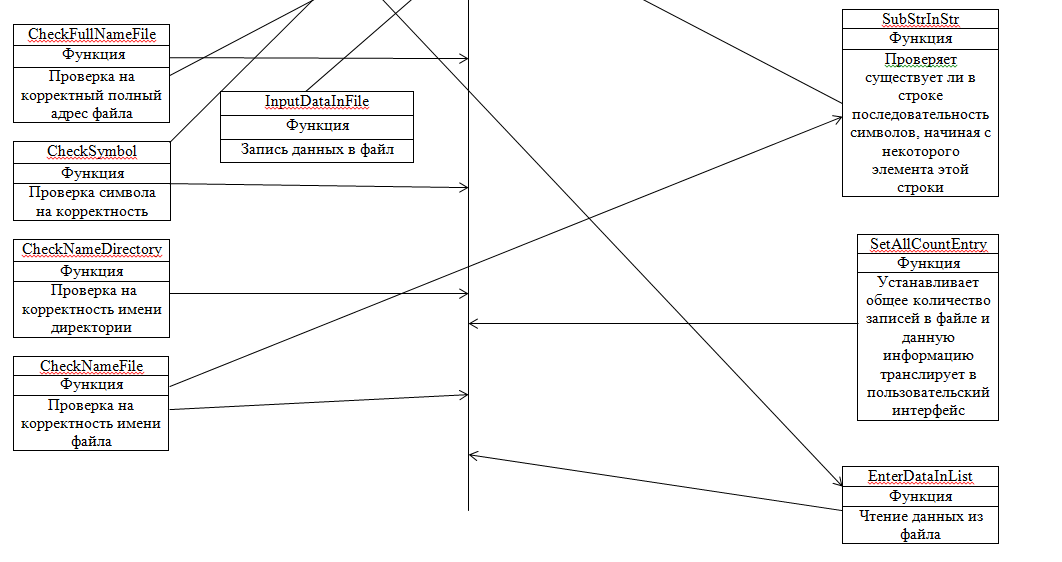


Рисунок 2 – Продолжение иерархии описания функций и их назначения

**CheckFullNameFile()** предназначена для полной проверки входной строки, которая является полным адресом файла, на корректность:

str – полный адрес файла

ext – тип файла

Функция возвращает булевое значение (true/false)

**CheckSymbol()** предназначена для проверки символа на корректность. Символ должен быть взят из латинского алфавита, кириллицы, может быть цифрой и символом ‘\_’ (замена пробела):

symbol – символ для проверки

Функция возвращает булевое значение (true/false)

**CheckNameDirectory()** предназначена для проверки пути к директории файла на корректность:

str – путь к директории

Функция возвращает булевое значение (true/false)

**CheckNameFile()** предназначена для проверки имени файла на корректность:

str – имя файла

ext – тип файла

Функция возвращает булевое значение (true/false)

**SubStrInStr()** предназначена для проверки входной строки на существования в ней подстроки, начинающейся с некоторого элемента строки и имеющего определённый размер:

base\_string – строка, в которой осуществляет поиск функция

substr – подстрока, которую нужно найти

point – индекс base\_string с которой нужно начинать поиск

Функция возвращает булевое значение (true/false)

**SetAllCountEntry()** устанавливает общее количество записей в файле и осуществляет проверку на корректность файла и на принадлежность файла данной программе, путём проверки записей в нём:

Функция возвращает булевое значение (true/false)

**DefineMaxMinPriceToy()** предназначена для поиска наибольшей или наименьшей по цене игрушки:

c – флаг, который управляет работой функции. При c равной true, функция осуществляет поиск игрушки с наибольшей ценой, при c равной false, функция осуществляет поиск игрушки с наименьшей ценой.

Функция ничего не возвращает

**EnterDataInList()** осуществляет запись данных из файла в множество структур типа UserEntry:

fullNameFile – полный путь к файлу

Функция возвращает булевое значение (true/false)

**EnterInBaseData()** предназначена для ввода в таблицу пользовательского интерфейса, для просмотра записей, данных полученных из файла и записанных в множество структур тип UserEntry:

fullNameFile – полный путь к файлу

Функция ничего не возвращает.

**InputDataInFile()** предназначена для записи в файл данных, скорректированных на определённых этапах работы с базой данных записей через пользовательский интерфейс:

fullNameFile – полный путь к файлу.

Функция ничего не возвращает.

2 События форм

В табл. 1 перечислены события форм.

Таблица 1 – События форм

|  |  |
| --- | --- |
| Название события | Назначение |
| MainForm | |
| addButton\_Click | Открытие формы CreateFileForm, закрытие формы MainForm |
| openButton\_Click | Открытие формы OpenFileForm, закрытие формы MainForm |
| deleteButton\_Click | Удаление файла из списка базы данных или полностью |
| CloseFileButton\_Click | Закрытие открытого файла |
| SetDataButton\_Click | Добавить запись в файл |
| ViewEntryButton\_Click | Просмотр записей файла |
| “Файл” ToolStripMenuItem\_Click | Выпадающий список “Добавить файл”, “Открыть файл” |
| “Добавить файл” ToolStripMenuItem\_Click | Открытие формы CreateFileFom, закрытие формы MainForm |
| “О программе” ToolStripMenuItem\_Click | Вывести сообщение описывающее программу |
| CreateFileForm | |
| CreateButton\_Click | Добавление полного пути файла с анализом на корректность |
| ViewButton\_Click | Добавление полного пути файла с помощью обозревателя директорий “Обзор” |
| exitButton\_Click | Закрытие формы CreateFileForm, открытие формы MainForm |
| OpenFileForm | |
| OpenButton\_Click | Добавление полного пути файла с анализом на корректность |
| ViewButton\_Click | Добавление полного пути файла с помощью обозревателя директорий “Обзор” |
| exitButton\_Click | Закрытие формы OpenFileForm, открытие формы MainForm |
| CreateEntryForm | |
| AddEntryButton\_Click | Добавление записи в конец файла |
| ExitButton\_Click | Закрытие формы CreateEntryFrom, открытие MainForm |

Продолжение таблицы 1

|  |  |
| --- | --- |
| valueFrom\_Press | Ввод данных характеризующих возрастное ограничение “от” игрушки |
| priceToy\_Press | Ввод цены игрушки в поле для ввода |
| valueBefore\_Press | Ввод данных характеризующих возрастное ограничение “до” игрушки |
| ViewStatForm | |
| ProcButton\_Click | Открытие формы RequestForm |
| SetEntryButton\_Click | Закрытие формы ViewStatForm, RequestForm(если открыта), открытие формы CreateEntryForm |
| DeleteButton\_Click | Удаление записи из файла(любой записи на выбор) |
| ExitButton\_Click | Закрывает форму ViewStatForm, RequestForm(если открыта), открывает форму MainForm |
| RequestForm | |
| AllMaxPriceButton\_Click | Обработка 1-го запроса(согласно индивидуальному заданию) |
| AllNameButton\_Click | Обработка 2-го запроса(согласно индивидуальному заданию) |
| HaveButton\_Click | Обработка 3-го запроса(согласно индивидуальному заданию) |
| AgeLimitButton\_Click | Обработка 4-го запроса(согласно индивидуальному заданию) |
| ExitButton\_Click | Закрытие формы RequestForm |
| havePriceBox\_Press | Ввод данных в поле для цены 3-го запроса |
| haveValueFromBox\_Press | Ввод данных в поле для значения характеризующего возрастное ограничение “от”(3-й запрос) |
| haveValueBeforeBox\_Press | Ввод данных в поле для значения характеризующего возрастное ограничение “до”(3-й запрос) |
| allMaxPriceBox\_Press | Ввод данных в поле для цены 1-го запроса |

3 Алгоритмы основных событий

**Функция CheckFullNameFile():**

Таблица 2 – Спецификация функции CheckFullNameFile

|  |  |
| --- | --- |
| Входные данные | Выходные данные |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Str | Полный путь к файлу | Строка | Check | Результат проверки | true/false |
| Ext | Расширение файла | Строка |  |  |  |

**Алгоритм:**

1.check = true;

2. ЕСЛИ ((str.Length <= 0) || (ext.Length <= 0)

|| (str.Length < ext.Length)

|| (!CheckSymbol(str[0])) || (!CheckSymbol(str[str.Length - 1]))

|| (str.IndexOf('.') != str.LastIndexOf('.'))) РАВНО true ТО

* 1. check = false
  2. Вернуть check

3. nameDirectory = “ ”

4. nameFile = “ ”

5. Присвоим переменной point индекс элемента строки str, в которой находится точка(‘.’):

point = str.IndexOf('.');

6. i = 0

7. ПОКА i < point ВЫПОЛНИТЬ

7.1. nameDirectory += str[i]

7.2. i = i + 1

ПОКА ВСЕ

8. Присвоим переменной point индекс элемента строки str, в которой находится символ “\”

point = str.LastIndexOf('\\');

9. i = (point+1)

10. ПОКА i < str.Length ВЫПОЛНИТЬ

10.1. nameFile += str[i]

10.2. i = i+1

ПОКА ВСЕ

11. ЕСЛИ ((nameDirectory.Length <= 0) || (nameFile.Length <= 0)

|| (!FileString.CheckNameDirectory(nameDirectory))

|| (!FileString.CheckNameFile(nameFile, ".txt"))) РАВНО true ТО

11.1. check = false

11.2. Вернуть check

12. Вернуть check

**Функция CheckSymbol():**

Таблица 3 – Спецификация функции CheckSymbol

|  |  |
| --- | --- |
| Входные данные | Выходные данные |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Symbol | Символ, который нужно проверить | Строка | check | Результат проверки | true/false |

**Алгоритм:**

1. check = true

2. ЕСЛИ (((symbol >= '0') && (symbol <= '9'))

|| ((symbol >= 'A') && (symbol <= 'Z'))

|| ((symbol >= 'a') && (symbol <= 'z'))

|| ((symbol >= 'А') && (symbol <= 'п'))

|| ((symbol >= 'р') && (symbol <= 'я'))

|| (symbol == '\_')) РАВНО true

2.1. Вернуть check

ИНАЧЕ

2.2. check = false

2.3. Вернуть check

**Функция CheckNameDirectory():**

Таблица 4 – Спецификация функции CheckNameDirectory

|  |  |
| --- | --- |
| Входные данные | Выходные данные |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Str | Путь к директории файла | Строка | check | Результат проверки | true/false |

**Алгоритм:**

1. check = true

2. ЕСЛИ ((str.Length <= 0) || (str.IndexOf('.') >= 0)

|| (str.IndexOf(':') != str.LastIndexOf(':')) || (str.IndexOf(':') < 0)

|| (!CheckSymbol(str[0]))

|| ((!CheckSymbol(str[str.Length - 1])) && (str[str.Length - 1] != '\\')))

2.1. check = false

2.2. Вернуть check

3. Присвоим переменной point индекс элемента строки str, в которой находится “:” :

point = str.IndexOf(“:”);

4. ЕСЛИ (str[point + 1] НЕ РАВНО '\\') ТО

4.1. check = false

4.2. Вернуть check

5. i = 0

6. ПОКА i < point ВЫПОЛНИТЬ

6.1. ЕСЛИ (CheckSymbol(str[i]) НЕ РАВНО true)

6.1.1. check = false

6.1.2. Вернуть check

6.2. i = i+1

ПОКА ВСЕ

7. count = 0

8. i = (point + 1)

9. ПОКА i < str.Length ВЫПОЛНИТЬ

9.1. ЕСЛИ (str[i] РАВНО '\\') ТО

9.1.1. count = count + 1

ИНАЧЕ ЕСЛИ (count > 1) ТО

9.1.2. check = false

9.1.3. Вернуть check

ИНАЧЕ

9.1.4. count = 0

9.2. i = i + 1

ПОКА ВСЕ

10. Вернуть check

**Функция CheckNameFile():**

Таблица 5 – Спецификация функции CheckNameFile

|  |  |
| --- | --- |
| Входные данные | Выходные данные |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Str | Имя файла | Строка | check | Результат проверки | true/false |
| Ext | Расширение файла | Строка |  |  |  |

**Алгоритм:**

1. check = true

2. ЕСЛИ ((str.Length <= 0) || (str.IndexOf('.') != str.LastIndexOf('.'))

|| (!CheckSymbol(str[0])) || (!CheckSymbol(str[str.Length - 1]))

|| (ext[0] != '.')) РАВНО true ТО

2.1. check = false

2.2. Вернуть check

3. Присвоим переменной point индекс элемента строки str, в которой находится точка(‘.’):

point = str.IndexOf('.');

4. ЕСЛИ (point > 0) ТО

4.1. ЕСЛИ SubStrInStr(str, ext, point) РАВНО false ТО

4.1.1. check = false

4.1.2. Вернуть check

4.2. i = 0

4.3. ПОКА i < point ВЫПОЛНИТЬ

4.3.1. ЕСЛИ CheckSymbol(str[i]) РАВНО false ТО

4.3.1.1. check = false

4.3.1.2. Вернуть check

ПОКА ВСЕ

4.4. ЕСЛИ ((str.Length - point) НЕ РАВНО (ext.Length)) ТО

4.4.1. check = false

4.5. Вернуть check

ИНАЧЕ

4.6. i = 0

4.7. ПОКА i < str.Length ВЫПОЛНИТЬ

4.7.1. ЕСЛИ CheckSymbol(str[i]) РАВНО false ТО

4.7.1.1. check = false

4.7.1.2. Вернуть check

5. Вернуть check

**Функция SubStrInStr():**

Таблица 6 – Спецификация функции SubStrInStr

|  |  |
| --- | --- |
| Входные данные | Выходные данные |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| base\_string | Строка, в которой будет осуществляться поиск другой строки | Строка | check | Результат проверки | true/false |
| Substr | Строка, которую нужно найти в строке base\_string с индекса point | Строка |  |  |  |
| Point | Индекс, с которой нужно искать строку | Целое число > 0 |  |  |  |

**Алгоритм:**

1. check = true

2. ЕСЛИ ((base\_string.Length <= substr.Length) ||

(base\_string.Length <= 0) || (substr.Length <= 0)

|| (point >= base\_string.Length) || (point < 0)) РАВНО true ТО

2.1. check = false

2.2. Вернуть check

3. count = 0

4. i = 0

5. j = 0

6. ПОКА (i < base\_string.Length) И (j < substr.Length) ВЫПОЛНИТЬ

6.1. ЕСЛИ base\_string[i] РАВНО substr[j] ТО

6.1.1. count = count + 1

6.2. i = i + 1

6.3. j = j + 1

ПОКА ВСЕ

7. ЕСЛИ count НЕ РАВНО substr.Length ТО

7.1. check = false

8. Вернуть check

**MainForm**

**Функция addButton\_Click():**

1. Вызов функции createFileClick()

**Функция createFileClick():**

1. Присвоить переменной formCreateFile созданную новую память для объекта CreateFileForm:

formCreateFile = new CreateFileForm

2. Присвоить свойству Extension объекта formCreateFile значение “.txt”:

formCreateFile.Extension = ".txt";

3. Присвоить свойству Click объекта CreateButton\_CreateFileForm объекта formCreateFile зачение this.createFile\_Create:

formCreateFile.CreateButton\_CreateFileForm.Click += this.createFile\_Create;

4. Присвоить свойству Click объекта ViewButton\_CreateFileForm объекта formCreateFile зачение this.createFile\_Create:

formCreateFile.ViewButton\_CreateFileForm.Click += this.createFile\_Create;

5. Присвоить свойству Left объекта formCreateFile значение this.Left:

formCreateFile.Left = this.Left;

6. Присвоить свойству Top объекта formCreateFile значение this.Top:

formCreateFile.Left = this.Top;

7. Открыть форму formCreateFile:

formCreateFile.Activate()

formCreateFile.Show();

8. Закрыть форму MainForm:

this.Hide();

9. Завершить функцию

**Функция createFile\_Create():**

1. ЕСЛИ ((formCreateFile.NameFile == null) || (formCreateFile.NameFile == "")) РАВНО true ТО

1.1. Завершить функцию

2. i = 0

3. ПОКА i< this.checkedListFile.Items.Count ВЫПОЛНИТЬ

3.1. ЕСЛИ (formCreateFile.NameFile == this.checkedListFile.Items[i].ToString()) ТО

3.1.1. MessageBox.Show("Файл с таким именем уже существует или добавлен", "Ошибка");

3.1.2. Завершить функцию

3.2. i=i+1

4. ЕСЛИ (!File.Exists(formCreateFile.NameFile)) РАВНО true

4.1. name\_directory = “ “

4.2. i=0

4.3. point = formCreateFile.NameFile.LastIndexOf('\\')

4.4. ПОКА i < point ВЫПОЛНИТЬ

4.4.1. name\_directory += formCreateFile.NameFile[i];

4.4.2. i=i+1

ПОКА ВСЕ

4.5. ЕСЛИ (!Directory.Exists(name\_directory)) РАВНО true ТО

4.5.1. БЛОК TRY

4.5.1.1. Directory.CreateDirectory(name\_directory);

КОНЕЦ БЛОКА TRY

БЛОК CATCH(Exception)

4.5.1.2. MessageBox.Show("Введены не корректные данные!", "Ошибка");

4.5.1.3. formCreateFile.NameFile = "";

4.5.1.4. Завершить функцию

КОНЕЦ БЛОКА CATCH

4.6. File.Create(formCreateFile.NameFile).Close();

4.7. ЕСЛИ (!File.Exists(formCreateFile.NameFile)) РАВНО true ТО

4.7.1. MessageBox.Show("Введены не корректные данные!", "Ошибка");

4.7.2. formCreateFile.NameFile = "";

4.7.3. Завершить функцию

5. this.checkedListFile.Items.Add(formCreateFile.NameFile);

6. Завершить функцию

**Функция openButton\_Click(sender, e):**

1. ЕСЛИ (this.checkedListFile.CheckedItems.Count <= 0) ТО

1.1. Вызвать функцию openFile\_Click(sender, e);

1.2. Завершить функцию

2. label3.Text = this.checkedListFile.CheckedItems[0].ToString();

3. ЕСЛИ SetAllCountEntry() РАВНО false ТО

3.1. MessageBox.Show("Данный файл не может быть обработан! " +

"Файл либо не принадлежит базе данных, либо данные в файле испорчены", "Ошибка");

3.2. Завершить функцию

4. this.unCheckedListFile\_Full();

5. Завершить функцию

**Функция unCheckedListFile\_Full():**

1. ЕСЛИ (this.checkedListFile.CheckedItems.Count == 0) ТО

1.1. Завершить функцию

2. listString – массив строк:

listString = new List<string>();

3. i = 0

4. ПОКА i < this.checkedListFile.Items.Count ВЫПОЛНИТЬ

4.1. listString.Add(this.checkedListFile.Items[i].ToString());

4.2.i=i+1

ПОКА ВСЕ

5. i=0

6. ПОКА i < listString.Count ВЫПОЛНИТЬ

6.1. this.checkedListFile.Items.Add(listString[i]);

6.2.i=i+1

ПОКА ВСЕ

7.Завершить функцию

**Функция openFile\_Click():**

1. Присвоить переменной formOpenFile созданную новую память для объекта OpenFileForm:

formOpenFile = new OpenFileForm();

2. Присвоить свойству Click объекта OpenButton объекта formOpenFile зачение this.openFileButton\_Click:

formOpenFile.OpenButton.Click += this.openFileButton\_Click;

3. Присвоить свойству Click объекта ViewButton объекта formOpenFile зачение this.openFileButton\_Click:

formOpenFile.ViewButton.Click += this.openFileButton\_Click;

4. Присвоить свойству Left объекта formOpenFile значение this.Left:

formOpenFile.Left = this.Left;

5. Присвоить свойству Top объекта formOpenFile значение this.Top:

formOpenFile.Left = this.Top;

6. Открыть форму formOpenFile:

formOpenFile.Activate()

formOpenFile.Show();

8. Закрыть форму MainForm:

this.Hide();

9. Завершить функцию

**Функция openFileButton\_Click():**

1. ЕСЛИ ((!FileString.CheckFullNameFile(formOpenFile.NameFile, ".txt"))) РАВНО true ТО

1.1. Завершить функцю

2. ЕСЛИ ((!File.Exists(formOpenFile.NameFile))) РАВНО true ТО

2.1. MessageBox.Show("Такого файла не существует!", "Ошибка");

2.2. Завершить функцию

3. label3.Text = formOpenFile.NameFile;

4. ЕСЛИ (!SetAllCountEntry()) РАВНО true ТО

4.1. MessageBox.Show("Данный файл не может быть обработан! " +

"Файл либо не принадлежит базе данных, либо данные в файле испорчены", "Ошибка");

5. Завершить функцию

**Функция CloseFileButton\_Click():**

1.Очищаем значение label5.Text:

label5.Text = “ “

2. Очищаем значение label3.Text:

label3.Text = “ “

**Функция SetDataButton\_Click():**

1. ЕСЛИ label3.Text.Length <= 0 ТО

1.1. MessageBox.Show("Нет открытого файла для добавления записи!", "Ошибка");

1.2. Завершить функцию

2. Создать форму CreateEntryForm:

formEntry = new CreateEntryForm()

3. Присвоить свойству Click объекта AddEntryButton объекта formEntry зачение this.addEntryButton\_Click:

formEntry.AddEntryButton.Click += this.addEntryButton\_Click;

4. Присвоить свойству Left объекта formEntry значение this.Left:

formEntry.Left = this.Left;

5. Присвоить свойству Top объекта formEntry значение this.Top:

formEntry.Top = this.Top;

6. Открыть форму formEntry:

formEntry.Activate()

formEntry.Show()

7. Закрыть форму MainForm:

this.Hide()

**Функция addEntryButton\_Click():**

1.ЕСЛИ ((!File.Exists(label3.Text)) || (!formEntry.Entry.CheckAllAttribute())) РАВНО true ТО

1.1. Заверишть функцию

2. Откроем поток для записи в файл:

StreamWriter fileWriter = new StreamWriter(label3.Text, true);

3.Запишем в файл данные:

fileWriter.WriteLine(label3.Text + " "

+ formEntry.Entry.name\_toy + " "

+ formEntry.Entry.price\_toy.ToString() + " "

+ formEntry.Entry.valueFrom + " "

+ formEntry.Entry.valueBefore);

4. Закроем поток для записи в файл:

fileWriter.Close()

5. ЕСЛИ SetAllCountryEntry() РАВНО false ТО

5.1. MessageBox.Show("Данный файл не может быть обработан! " +

"Файл либо не принадлежит базе данных, либо данные в файле испорчены", "Ошибка");

5.2. Завершить функцию

6. MessageBox.Show("Запись добавлена!", "Уведомление");

7. formEntry.Entry.SetStandartValues() (вызов метода структуры UserEntry SetStandartValues() ).

8. Завершить функцию

**Функция SetAllCountEntry():**

1.ЕСЛИ ((!File.Exists(label3.Text)) || (label3.Text.Length <= 0)) ТО

1.1. Вернуть false

2. Открыть поток для чтения из файла:

StreamReader fileReader = new StreamReader(label3.Text);

3. Запишем в массив строк allEntry все записи, которые пользователь добавил в файл:

allEntry = fileReader.ReadToEnd().Split('\n');

4. Закрыть поток для чтения из файла:

fileReader.Close()

5. i = 0

6. ПОКА i < (allEntry.Length – 1) ВЫПОЛНИТЬ

6.1. ЕСЛИ ((!(allEntry[i].Split(' ')[0].Equals(label3.Text)))

|| ((allEntry[i].Split(' ').Length != 5) && (allEntry[i].Split(' ').Length > 0))) РАВНО true ТО

6.1.1. Вернуть false

6.2.i=i+1

ПОКА ВСЕ

7. label5.Text = (allEntry.Length – 1).ToString()

8. Завершить функцию

**Функция ViewEntryButton\_Click():**

1. ЕСЛИ label3.Text.Length <= 0 ТО

1.1. MessageBox.Show("Нет открытого файла для добавления записи!", "Ошибка");

1.2. Завершить функцию

2. Создать форму ViewStatForm:

formStatView = new ViewStatForm(label3.Text);

3. Присвоить свойству Left объекта formStatView значение this.Left:

formStatView.Left = this.Left;

5. Присвоить свойству Top объекта formStatView значение this.Top:

formStatView.Top = this.Top;

6. Открыть форму formEntry:

formStatView.Activate()

formStatView.Show()

7. Закрыть форму MainForm:

this.Hide()

**Функция “Добавить” ToolStripMenuItem\_Click(sender, e):**

1. Вызов this.createFile\_Click(sender, e)

**Функция “Открыть” ToolStripMenuItem\_Click(sender, e):**

1. Вызов this.openButton\_Click(sender, e)

**Функция “О программе” ToolStripMenuItem\_Click(sender, e):**

1. Вызов AboutProgram(sender, e)

Функция AboutProgram():

1. MessageBox.Show("Автор программы: Солопов Даниил Дмитриевич\nНазвание программы: База данных игрушек" +

"(курсовой проект)\n\nЧто делает программа: создаёт/добавляет файл для работы, открывает файл" +

"осуществляет запись в файл данных, чтение данных из файла, просмотр данных.\n" +

"Осуществляет возможность изменения содержимого файла путём удаления/добавления записей.");

**Функция deleteButton\_Click():**

1. ЕСЛИ (this.checkedListFile.CheckedItems.Count == 0) ТО

1.1. Завершить функцию

2. ЕСЛИ (MessageBox.Show("Удалить файл(-ы)?", "Действие", MessageBoxButtons.YesNo) == DialogResult.Yes) ТО

2.1. ПОКА (this.checkedListFile.CheckedItems.Count > 0) ВЫПОЛНИТЬ

2.1.1. ЕСЛИ (this.checkedListFile.CheckedItems[0].ToString() == label3.Text) ТО

2.1.1.1. Очищаем label3.Text:

label3.Text = “ “

2.1.1.2. Очищаем label5.Text:

label5.Text = “ “

2.1.2. ЕСЛИ File.Exists(this.checkedListFile.CheckedItems[0].ToString()) РАВНО true ТО

2.1.2.1. File.Delete(this.checkedListFile.CheckedItems[0].ToString());

2.1.3. this.checkedListFile.Items.Remove(this.checkedListFile.CheckedItems[0]);

ПОКА ВСЕ

3. ПОКА (this.checkedListFile.CheckedItems.Count > 0) ВЫПОЛНИТЬ

3.1. this.checkedListFile.Items.Remove(this.checkedListFile.CheckedItems[0]);

4. Завершить функцию

**Функция MainForm():**

1. Инициализируем компонененты :

InitializeComponent();

2. Создать верхнее меню:

fileItem = new ToolStripMenuItem("Файл");

3. createFile = new ToolStripMenuItem("Добавить файл");

4. openFile = new ToolStripMenuItem("Открыть файл")

5. Задать реакцию программы на действия с панелью меню:

createFile.Click += this.createFile\_Click;

6. openFile.Click += this.openButton\_Click;

7. Добавить в кнопку “Файл” выпадающее меню, состоящее из “Добавить файл” и "Открыть файл” :

fileItem.DropDownItems.Add(createFile);

8. fileItem.DropDownItems.Add(openFile);

9. menuStrip1.Items.Add(fileItem);

10. Добавить в меню “О программе”:

dataAbout = new ToolStripMenuItem("О программе");

11. Задать реакцию программы на действия с кнопкой “О программе”:

dataAbout.Click += AboutProgram;

12. menuStrip1.Items.Add(dataAbout);

**CreateFileForm**

**Функция CreateFileForm():**

1. InitializeComponent()

2. Задать реакцию программы на щелчёк кнопки закрытия формы:

this.FormClosed += this.exitButton\_Click;

3. Создать объект openFile:

openFile = new OpenFileDialog();

4. openFile.Title = "Обзор";

5. openFile.Filter = "Текстовые файлы (\*.txt)|\*txt";

**Функция exitButton\_Click():**

1. Присвоить formMainWindow объект характеризующий форму MainWindow:

formMainWindow = Application.OpenForms[0];

2. formMainWindow.StartPosition = FormStartPosition.Manual;

3. formMainWindow.Left = this.Left;

4. formMainWindow.Top = this.Top;

5. Открыть форму formMainWindow:

formMainWindow.Show();

6. Закрыть форму CreateFileForm:

this.Hide()

**Функция CreateButton\_Click():**

1. nameDirectory.Text = nameDirectory.Text.Trim(' ').Replace(' ', '\_');

2. nameFile.Text = nameFile.Text.Trim(' ').Replace(' ', '\_');

3.ЕСЛИ ((nameDirectory.TextLength <= 0) || (nameFile.TextLength <=0)

|| (!FileString.CheckNameDirectory(nameDirectory.Text))

|| (!FileString.CheckNameFile(nameFile.Text, ".txt"))) ТО

3.1. MessageBox.Show("Введены не корректные данные", "Ошибка");

3.2. Завершить функцию

4. openFile.FileName = nameDirectory.Text;

5. ЕСЛИ (openFile.FileName[openFile.FileName.Length - 1] != '\\') ТО

5.1. openFile.FileName += "\\" + nameFile.Text;

ИНАЧЕ

5.2. openFile.FileName += nameFile.Text;

6. ЕСЛИ (!FileString.SubStrInStr(openFile.FileName, this.extension, openFile.FileName.IndexOf('.'))) ТО

6.1. openFile.FileName += this.extension;

7. Вызов функции exitButton\_Click:

this.exitButton\_Click(sender, e);

8. Завершить функцию

**Функция ViewButton\_Click():**

1. ЕСЛИ (openFile.ShowDialog() != DialogResult.OK) ТО

1.1. Завершить функцию

2. ЕСЛИ (!FileString.CheckFullNameFile(openFile.FileName, ".txt")) РАВНО true ТО

2.1. MessageBox.Show("Введены не корректные данные", "Ошибка");

2.2. Завершить функцию

3. Вызов функции exitButton\_Click:

this.exitButton\_Click(sender, e);

4. Завершить функцию

**OpenFileForm**

**Функция OpenFileForm():**

1. InitializeComponent()

2. this.FormClosed += exitButton\_Click;

3. openFile = new OpenFileDialog();

4. openFile.Title = "Обзор";

5. openFile.FileName = "";

6. openFile.Filter = "Текстовые файлы (\*.txt)|\*.txt";

**Функция exitButton\_Click():**

1. Form formMainWindow = Application.OpenForms[0];

2. formMainWindow.StartPosition = FormStartPosition.Manual;

3. formMainWindow.Left = this.Left;

4. formMainWindow.Top = this.Top;

5. Открыть форму formMainWindow:

formMainWindow.Show();

6. Закрыть форму CreateFileForm:

this.Hide()

**Функция OpenButton\_Click():**

1. openFile.FileName = this.textBox1.Text.Trim(' ').Replace(' ', '\_');

2. ЕСЛИ (!FileString.CheckFullNameFile(openFile.FileName, ".txt")) РАВНО true ТО

2.1. MessageBox.Show("Введены не корректные данные", "Ошибка");

2.1. Завершить функцию

3. Вызов функции exitButton\_Click:

this.exitButton\_Click(sender, e);

4. Завершить функцию

**Функция ViewButton\_Click():**

1. ЕСЛИ (openFile.ShowDialog() != DialogResult.OK) ТО

1.1. Завершить функцию

2. ЕСЛИ (!FileString.CheckFullNameFile(openFile.FileName, ".txt")) РАВНО true ТО

2.1. MessageBox.Show("Введены не корректные данные", "Ошибка");

2.2. Завершить функцию

3. Вызов функции exitButton\_Click:

this.exitButton\_Click(sender, e);

4. Завершить функцию

**CreateEntryForm**

**Функция CreateEntryForm():**

1. InitializeComponent()

2. Задать реакцию программы на закрытие формы:

this.FormClosed += this.ExitButton\_Click;

3. Задать стандартные значения переменной структуры dynamicEntry:

dynamicEntry.SetStandartValues();

4. Установить максимальную длину числа, характеризующего цену игрушки:

this.priceToyBox.MaxLength = 10;

5. Установить максимальную длину числа, характеризующего возрастное ограничение “от”:

this.valueFromBox.MaxLength = 2;

6. Установить максимальную длину числа, характеризующего возрастное ограничение “до”: this.valueBeforeBox.MaxLength = 2;

7. Завершить функцию

**Функция ExitButton\_Click():**

1. Присвоить переменной formMainWindow объект MainForm:

formMainWindow = Application.OpenForms[0];

2. formMainWindow.StartPosition = FormStartPosition.Manual;

3. formMainWindow.Left = this.Left;

4. formMainWindow.Top = this.Top;

5. Открыть форму formMainWindow:

formMainWindow.Show();

6. Закрыть форму CreateFileForm:

this.Hide()

**Функция AddEntryButton\_Click():**

1. dynamicEntry.SetStandartValues();

2. this.nameToyBox.Text = this.nameToyBox.Text.Trim(' ');

3. ЕСЛИ ((this.nameToyBox.Text.Length <= 0) || (this.priceToyBox.Text.Length <= 0)

|| (this.valueFromBox.Text.Length <= 0) || (this.valueBeforeBox.Text.Length <= 0)) РАВНО true ТО

3.1. MessageBox.Show("Необходимо заполнить все поля ввода!", "Ошибка");

3.2. Завершить функцию

4. this.nameToyBox.Text = this.nameToyBox.Text.Replace(' ', '\_');

5. dynamicEntry.SetValues(this.nameToyBox.Text, System.Convert.ToDouble(this.priceToyBox.Text),

System.Convert.ToInt32(this.valueFromBox.Text), System.Convert.ToInt32(this.valueBeforeBox.Text));

6. ЕСЛИ ((dynamicEntry.valueFrom > dynamicEntry.valueBefore) || (!dynamicEntry.CheckAllAttribute())) ТО

6.1. dynamicEntry.SetStandartValues();

6.2. MessageBox.Show("Введены не корректные данные!", "Ошибка");

7. this.flagPoint\_Press = false;

8. Завершить функцию

**Функция priceToy\_Press(sender, e):**

1. Присвоить number численный код события, которое нужно обработать:

number = e.KeyChar;

2. ЕСЛИ (this.priceToyBox.Text.IndexOf(',') < 0) ТО

2.1. Флаг существования точки в тексте содержащего число поставить в неактивное положение this.flagPoint\_Press = false;

3. ЕСЛИ (((!Char.IsDigit(number)) && (number != ',') && (number != 8))

|| ((number == ',') && (this.flagPoint\_Press))) ТО

3.1. Проигнорировать запись в блок TextBox:

e.Handled = true;

4. ЕСЛИ ((number == ',') && (!this.flagPoint\_Press)) ТО

4. Флаг существования точки в тексте содержащего число поставить в неактивное положение this.flagPoint\_Press = false;

5. Завершить функцию

**Функция valueFrom\_Press(sender, e):**

1. Присвоить number численный код события, которое нужно обработать:

number = e.KeyChar;

2. ЕСЛИ (!Char.IsDigit(number) && (number != 8)) ТО

2.1. Проигнорировать запись в блок TextBox:

e.Handled = true;

3. Завершить функцию

**Функция valueBeforePress(sender, e):**

1. Присвоить number численный код события, которое нужно обработать:

number = e.KeyChar;

2. ЕСЛИ (!Char.IsDigit(number) && (number != 8)) ТО

2.1. Проигнорировать запись в блок TextBox:

e.Handled = true;

3. Завершить функцию

**ViewStatForm**

**Функция ViewStatForm(string fullNameFile):**

1. InitializeComponent();

2. Задать реакцию программы на закрытие формы:

this.FormClosed += ExitButton\_Click;

3. Вызов функции EnterBaseData():

EnterInBaseData(fullNameFile);

4. this.full\_name\_file = fullNameFile;

5. Флаг активной формы поставить в активное положение:

this.openRequestForm = false;

6. Завершить функцию

**Функция DefineMaxMinPriceToy(bool c):**

Таблица 7 – Спецификация функции DefineMaxMinPriceToy

|  |  |
| --- | --- |
| Входные данные | Выходные данные |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| c | Какое значение искать: true – максимальное, false - минимальное | Строка |  |  |  |

**Алгоритм:**

1. ЕСЛИ ((this.userEntries == null) || (this.userEntries.Count <= 0)) ТО

1.1. Завершить функцию

2. index = 0

3. i=1

4. ПОКА i < this.userEntries.Count ВЫПОЛНИТЬ

4.1. ЕСЛИ (((this.userEntries[index].price\_toy > this.userEntries[i].price\_toy) && (!c))

|| ((this.userEntries[index].price\_toy < this.userEntries[i].price\_toy) && (c))) ТО

4.1.1.index = i

4.2. i=i+1

ПОКА ВСЕ

5. ЕСЛИ c РАВНО false ТО

5.1. this.nameMinPriceToy.Text = this.userEntries[index].name\_toy;

5.2. this.minPrice.Text = this.userEntries[index].price\_toy.ToString();

ИНАЧЕ

5.3. this.nameMaxPriceToy.Text = this.userEntries[index].name\_toy;

5.4. this.maxPrice.Text = this.userEntries[index].price\_toy.ToString();

6. Завершить функцию

**Функция EnterDataInListFile():**

Таблица 8 – Спецификация функции EnterDataInListFile

|  |  |
| --- | --- |
| Входные данные | Выходные данные |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| fullNameFile | Полный путь к файлу | Строка | check | Результат проверки | true/false |

**Алгоритм:**

1. check = true;

2. ЕСЛИ ((!File.Exists(fullNameFile)) || (fullNameFile.Length <= 0)) ТО

2.1. check = false

2.2. Вернуть check

3. Открыть поток для чтения из файла:

StreamReader fileReader = new StreamReader(fullNameFile);

4. Читать из файла все данные и последовательно их разместить в массиве структур allEntry:

allEntry = fileReader.ReadToEnd().Split('\n');

5. Закрыть поток чтения данных:

fileReader.Close()

6. userEntries – массив структур типа UserEntry:

this.userEntries = new List<UserEntry>();

7. i=0

8.ПОКА i<(allEntry.Length-1) ВЫПОЛНИТЬ

8.1. entry = allEntry[i].Split(' ');

8.2. entry[1] = entry[1].Replace('\_', ' ');

8.3. us – структура типа UserEntry:

us = new UserEntry();

8.4. Вызов метода SetValues объекта us:

us.SetValues(entry[1], System.Convert.ToDouble(entry[2]),

System.Convert.ToInt32(entry[3]), System.Convert.ToInt32(entry[4]));

8.5. Вызов метода Add объекта userEntries:

this.userEntries.Add(us);

8.6. i=i+1

ПОКА ВСЕ

9. Вернуть check

**Функция EnterDataInListFile():**

Таблица 9 – Спецификация функции EnterDataInListFile

|  |  |
| --- | --- |
| Входные данные | Выходные данные |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| fullNameFile | Полный путь к файлу | Строка |  |  |  |

**Алгоритм:**

1. ЕСЛИ (EnterDataInList(fullNameFile)) РАВНО false ТО

1.1. Завершить функцию

2. i=0

3. ПОКА i < this.userEntries.Count ВЫПОЛНИТЬ

3.1.Вызвать метод Add объекта dataBaseView: this.dataBaseView.Rows.Add(this.userEntries[i].name\_toy,

(this.userEntries[i].price\_toy.ToString() + " руб."), ("от " + this.userEntries[i].valueFrom.ToString() +

" до " + this.userEntries[i].valueBefore.ToString()));

3.2. this.dataBaseView.Rows[i].ReadOnly = true;

3.3. this.dataBaseView.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

3.4. i=i+1

ПОКА ВСЕ

4. Вызов функции DefineMaxMinPriceToy():

this.DefineMaxMinPriceToy(false);

5. Вызов функции DefineMaxMinPriceToy():

this.DefineMaxMinPriceToy(true);

6. Завершить функцию

**Функция ExitButton\_Click():**

1. ЕСЛИ (this.openRequestForm) РАВНО true ТО

1.1. this.formRequest.Hide()

2. Присвоить formMainWindow значение объекта MainForm:

formMainWindow = Application.OpenForms[0];

3. formMainWindow.StartPosition = FormStartPosition.Manual;

4. formMainWindow.Left = this.Left;

5. formMainWindow.Top = this.Top;

6. Открыть форму formMainWindow:

formMainWindow.Show();

7. Закрыть форму CreateFileForm:

this.Hide()

**Функция dbPrePaint(sender, e):**

1. index = e.RowIndex;

2. indexStr = (index+1).ToString()

3. header = this.dataBaseView.Rows[index].HeaderCell.Value;

4. ЕСЛИ ((header == null) || (!header.Equals(indexStr))) ТО

4.1. this.dataBaseView.Rows[index].HeaderCell.Value = indexStr;

5. Завершить функцию

**Функция SetEntryButton\_Click():**

1.ЕСЛИ (this.openRequestForm) РАВНО true ТО

1.1. this.formRequest.Hide()

2.Закрыть форму ViewStatForm:

this.Hide()

3. Вызвать метод SetData() объекта MainForm:

((MainForm)Application.OpenForms[0]).SetData();

4. Завершить функцию

**Функция InputDataInFile():**

Таблица 10 – Спецификация функции InputDataInFile

|  |  |
| --- | --- |
| Входные данные | Выходные данные |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| fullNameFile | Полный путь к файлу | Строка |  |  |  |

**Алгоритм:**

1. ЕСЛИ ((this.userEntries == null) || (!File.Exists(fullNameFile)) || (this.userEntries.Count <= 0)) ТО

1.1. Завершить функцию

2. Открыть поток для записи в файл:

fileWriter = new StreamWriter(fullNameFile, false);

3.buffer – массив структур типа UserEntry:

buffer = new List<UserEntry>()

4.i=0

5. ПОКА i < this.userEntries.Count ВЫПОЛНИТЬ

5.1. us – структура типа UserEntry:

us = new UserEntry()

5.2. Вызов метода SetValuels() структуры UserEntry:

us.SetValues(this.userEntries[i].name\_toy.Replace(' ', '\_'),

this.userEntries[i].price\_toy,

this.userEntries[i].valueFrom,

this.userEntries[i].valueBefore);

5.3. Вызов метода Add объекта buffer, с последующей передачей ей в качестве параметра переменной us:

buffer.Add(us);

5.4. i = i+1

ПОКА ВСЕ

6. i = 0

7. ПОКА i < this.userEntries ВЫПОЛНИТЬ

7.1. Запись в файл через поток данных:

fileWriter.WriteLine(fullNameFile + " " + this.userEntries[i].name\_toy + " " +

this.userEntries[i].price\_toy.ToString() + " " +

this.userEntries[i].valueFrom.ToString() + " " +

this.userEntries[i].valueBefore.ToString());

7.2. i=i+1

ПОКА ВСЕ

8. Закрыть поток записи в файл:

fileWriter.Close()

9. Завершить функцию

**Функция DeleteButton\_Click():**

1. ЕСЛИ ((this.userEntries == null) || (this.userEntries.Count <= 0) || (this.dataBaseView.CurrentRow.Index < 0)) ТО

1.1.Завершить функцию

2. Вызвать метод RemoveAt объекта this.userEntries с передачей ей параметров:

this.userEntries.RemoveAt(this.dataBaseView.CurrentRow.Index);

3. ЕСЛИ ((this.dataBaseView.CurrentRow.Index == 0) && (this.dataBaseView.Rows.Count <= 1)) ТО

3.1. Удалить файл:

File.Delete(this.full\_name\_file);

3.2. Создать файл:

File.Create(this.full\_name\_file).Close();

3.3. ((MainForm)Application.OpenForms[0]).SetAllEntry();

4. Вызвать функцию InputDataInFile():

this.InputDataInFile(this.full\_name\_file);

5. Вызвать функцию EnterInBaseData():

this.EnterInBaseData(this.full\_name\_file);

6. Вызвать метод SetAllEtry() объекта MainForm:

((MainForm)Application.OpenForms[0]).SetAllEntry();

7. Завершить функцию

**Функция ProcButton\_Click():**

1. ЕСЛИ this.openRequestForm РАВНО true ТО

1.1.Завершить функцию

2. formRequest.Left = this.Left

3. formRequest.Top = this.Top

4. formRequest.Activate()

5. Открыть форму:

formRequest.Show()

6. this.openRequestForm = true;

7. Завершить функцию

**RequestForm**

**Функция RequestForm():**

1. Инициализация компонентов:

InitializeComponent()

2.Задать реакцию программы после закрытия формы:

this.FormClosed += ExitButton\_Click;

3.Установить максимальную длину строки характеризующей число, которое устанавливает возрастное ограничение “от”:

this.haveValueFromBox.MaxLength = 2;

4. .Установить максимальную длину строки характеризующей число, которое устанавливает возрастное ограничение “до”:

this.haveValueBeforeBox.MaxLength = 2;

5. Установить максимальную длину строки характеризующей число, которое используется в 3-ем запросе и является ценой игрушки:

this.haveValueBeforeBox.MaxLength = 2;

6. Установить максимальную длину строки характеризующей число, которое используется в 1-ом запросе и характеризует цену:

this.allMaxPriceBox.MaxLength = 10;

7.Завершить функцию

**Функция ExitButton\_Click():**

1. Закрыть форму RequestForm:

this.Hide()

2. Установить флаг активной формы RequestForm в неактивное положение:

((MainForm)(Application.OpenForms[0])).ViewStateForm.OpenFormRequest = false;

3.Завершить функцию

**Функция AgeLimitButton\_Click():**

1. Убираем все начальные и конечные пробелы ageLimitBox.Text:

this.ageLimitBox.Text = this.ageLimitBox.Text.Trim(' ');

2. ЕСЛИ (this.ageLimitBox.Text.Length <= 0) ТО

2.1. MessageBox.Show("Необходимо заполнить поле для имени игрушки!", "Ошибка");

2.2. Завершить функцию

3.listUser – массив структур типа UserEntry:

listUser = ((MainForm)(Application.OpenForms[0])).ViewStateForm.AllEntry;

4.index = (-1)

5.i=0

6. ПОКА (i < listUser.Count) && (index < 0) ВЫПОЛНИТЬ

6.1. ЕСЛИ (listUser[i].name\_toy.Equals(this.ageLimitBox.Text)) РАВНО true ТО

6.1.1. index = i;

6.2. i = i+1

ПОКА ВСЕ

7. ЕСЛИ index < 0 ТО

7.1. MessageBox.Show("Игрушки с таким именем в базе данных нет", "Уведомление");

7.2. Завершить функцию

8. indexList – массив типа int:

indexList = new List<int>();

9. Вызвать метод Add() объекта indexList с передачей ей параметра:

indexList.Add(index)

10. i=index+1

11. ПОКА i<listUser.Count ВЫПОЛНИТЬ

11.1.ЕСЛИ (listUser[i].name\_toy.Equals(this.ageLimitBox.Text)) РАВНО true ТО

11.1.1. Вызвать метод Add() объекта indexList с передачей ей параметра:

indexList.Add(i);

11.2.i=i+1

ПОКА ВСЕ

12. i=0

13. ПОКА i < indexList.Count ВЫПОЛНИТЬ

13.1. data\_toys += "от " + listUser[indexList[i]].valueFrom.ToString()

+ " до " + listUser[indexList[i]].valueBefore.ToString() + "\n";

13.2.i=i+1

ПОКА ВСЕ

14. MessageBox.Show("Возрастные границы игрушки " + this.ageLimitBox.Text + ":\n\n"

+ data\_toys + "\n", "Информация");

15. Завершить функцию

**Функция HaveButton\_Click():**

1. this.haveNameToyBox.Text = this.haveNameToyBox.Text.Trim(' ');

2. ЕСЛИ ((this.haveNameToyBox.Text.Length <= 0) || (this.havePriceToyBox.Text.Length <= 0)

|| (this.haveValueFromBox.Text.Length <= 0) || (this.haveValueBeforeBox.Text.Length <= 0)) ТО

2.1. MessageBox.Show("Необходимо заполнить поле для имени игрушки!", "Ошибка");

2.2. Завершить функцию

3. listUser – массив структур типа UserEntry:

listUser = ((MainForm)(Application.OpenForms[0])).ViewStateForm.AllEntry;

4.counter = 0

5.i=0

6. ПОКА i < listUser.Count ВЫПОЛНИТЬ

6.1. (listUser[i].name\_toy.Equals(this.haveNameToyBox.Text)

&& listUser[i].price\_toy.Equals(System.Convert.ToDouble(this.havePriceToyBox.Text))

&& listUser[i].valueFrom.Equals(System.Convert.ToInt32(this.haveValueFromBox.Text))

&& listUser[i].valueBefore.Equals(System.Convert.ToInt32(this.haveValueBeforeBox.Text)))

6.1.1. counter = counter+1

6.2.i=i+1

ПОКА ВСЕ

7. MessageBox.Show("Количество записей удовлетворяющих условию запроса: " + counter.ToString(), "Информация");

8. Завершить функцию

**Функция AllNameButton\_Click():**

1. this.allNameToyBox.Text = this.allNameToyBox.Text.Trim(' ');

2. ЕСЛИ (this.allNameToyBox.Text.Length <= 0) ТО

2.1. MessageBox.Show("Необходимо заполнить поле для имени игрушки!", "Ошибка");

2.2. Завершить функцию

3. listUser – массив структур типа UserEntry:

listUser = ((MainForm)(Application.OpenForms[0])).ViewStateForm.AllEntry;

4.index = (-1)

5.i=0

6. ПОКА (i < listUser.Count) && (index < 0) ВЫПОЛНИТЬ

6.1. (listUser[i].name\_toy.Equals(this.allNameToyBox.Text)) РАВНО true ТО

6.1.1. index = i

6.2. i=i+1

ПОКА ВСЕ

7. ЕСЛИ (index < 0) ТО

7.1. MessageBox.Show("Игрушки с таким именем в базе данных нет", "Уведомление");

7.2. Завершить функцию

8.i=0

9. ПОКА (i < listUser.Count) ВЫПОЛНИТЬ

9.1.ЕСЛИ (listUser[i].name\_toy.Equals(this.allNameToyBox.Text)) ТО

9.1.1. allPrice += listUser[i].price\_toy.ToString() + " руб.\n";

9.2.i=i+1

ПОКА ВСЕ

10. MessageBox.Show("Цены всех игрушек с именем " + this.allNameToyBox.Text + ": \n\n" + allPrice + "\n", "Информация");

11. Завершить функцию

**Функция AllMaxPriceButton\_Click():**

1. ЕСЛИ (this.allMaxPriceBox.Text.Length <= 0) ТО

1.1. MessageBox.Show("Необходимо заполнить поле для имени игрушки!", "Ошибка");

1.2. Завершить функцию

2. listUser – массив структур UserEntry:

listUser = ((MainForm)(Application.OpenForms[0])).ViewStateForm.AllEntry;

3. max\_price = ((MainForm)(Application.OpenForms[0])).ViewStateForm.MaxPrice;

4. def\_price = System.Convert.ToDouble(this.allMaxPriceBox.Text);

5.i=0

6. ПОКА i < listUser.Count ВЫПОЛНИТЬ

6.1.ЕСЛИ ((listUser[i].price\_toy + def\_price).Equals(max\_price)) ТО

6.1.1. data\_toys += listUser[i].name\_toy + " " +

listUser[i].price\_toy.ToString() + " .руб\n";

6.2. i=i+1

ПОКА ВСЕ

7.ЕСЛИ (data\_toys.Length <= 0) ТО

7.1. MessageBox.Show("Записей удовлетворяющих данному запросу в базе данных нет", "Уведомление");

7.2. Завершить функцию

8. MessageBox.Show("Записи удовлетворяющие данному запросу: \n\n" + data\_toys + "\n\n");

9. Завершить функцию

**Функция havePriceBox\_Press():**

1. Присвоить переменной number численный номер события, которое произошло:

number = e.KeyChar;

2. ЕСЛИ (this.havePriceToyBox.Text.IndexOf(',') < 0) ТО

2.1. this.flagPointHaveBox\_Press = false;

3. ЕСЛИ (((!Char.IsDigit(number)) && (number != ',') && (number != 8))

|| ((number == ',') && (this.flagPointHaveBox\_Press))) ТО

3.1. e.Handled = true;

4. ЕСЛИ ((number == ',') && (!this.flagPointHaveBox\_Press)) ТО

4.1. this.flagPointHaveBox\_Press = true;

5. Завершить программу

**Функция haveValueFromBox\_Press():**

1. Присвоить переменной number численный номер события, которое произошло:

number = e.KeyChar;

2. ЕСЛИ (!Char.IsDigit(number) && (number != 8)) ТО

2.1. e.Handled = true;

3.Завершить функцию

**Функция haveValueBeforeBox\_Press ():**

1. Присвоить переменной number численный номер события, которое произошло:

number = e.KeyChar;

2. ЕСЛИ (!Char.IsDigit(number) && (number != 8)) ТО

2.1. e.Handled = true;

3.Завершить функцию

**Функция allMaxPriceBox\_Press():**

1. Присвоить переменной number численный номер события, которое произошло:

number = e.KeyChar;

2. ЕСЛИ (this. allMaxPriceBox.Text.IndexOf(',') < 0) ТО

2.1. this. flagPointPriceBox\_Press= false;

3. ЕСЛИ (((!Char.IsDigit(number)) && (number != ',') && (number != 8))

|| ((number == ',') && (this.flagPointPriceBox\_Press))) ТО

3.1. e.Handled = true;

4. ЕСЛИ ((number == ',') && (!this.flagPointPriceBox\_Press)) ТО

4.1. this.flagPointPriceBox\_Press = true;

5. Завершить программу

4 Формы

1. Главная форма (Рис. 3)

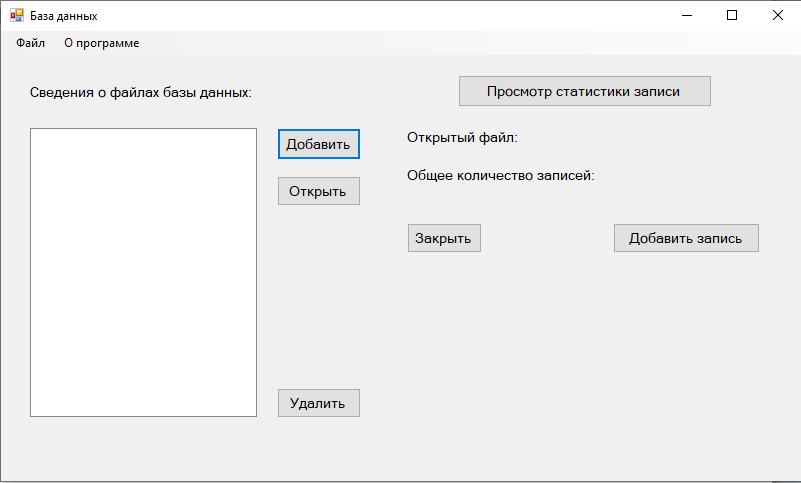


Рисунок 3 – Главная форма

2. Форма добавления/создания файла для базы данных (Рис. 4)

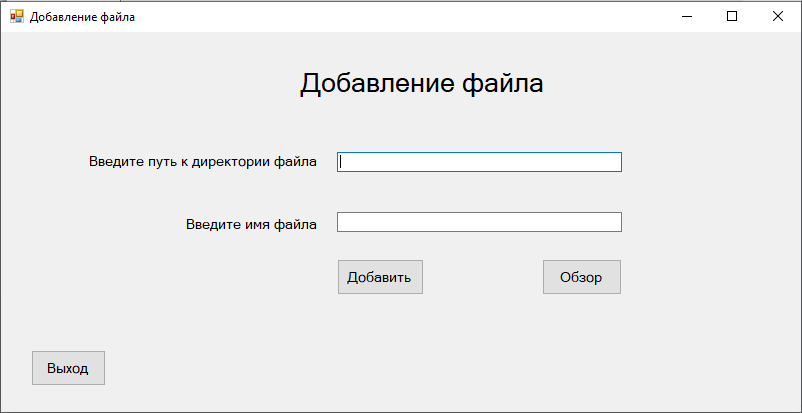


Рисунок 4 – Форма добавления/создания файла для базы данных

3. Форма открытия файла (Рис. 5)

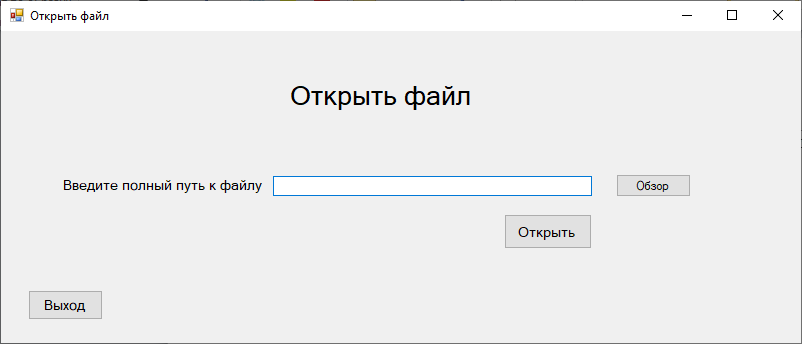


Рисунок 5 – Форма открытия файла

4. Форма добавления записи (Рис. 6)

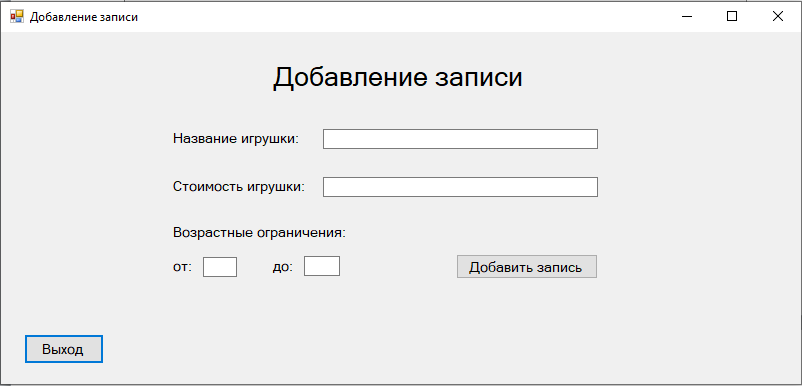


Рисунок 6 – Форма добавления записи

5. Форма просмотра статистики записей в файле (Рис. 7)

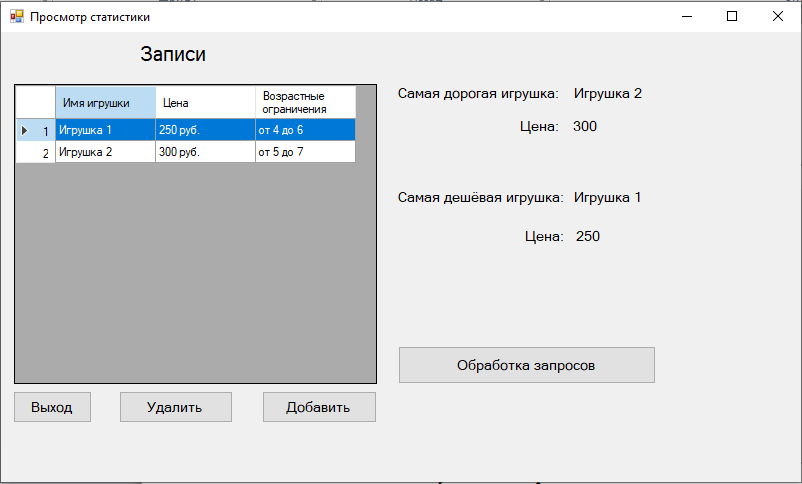


Рисунок 7 – Форма просмотра статистики записей в файле

6. Форма обработки запросов (Рис. 8)

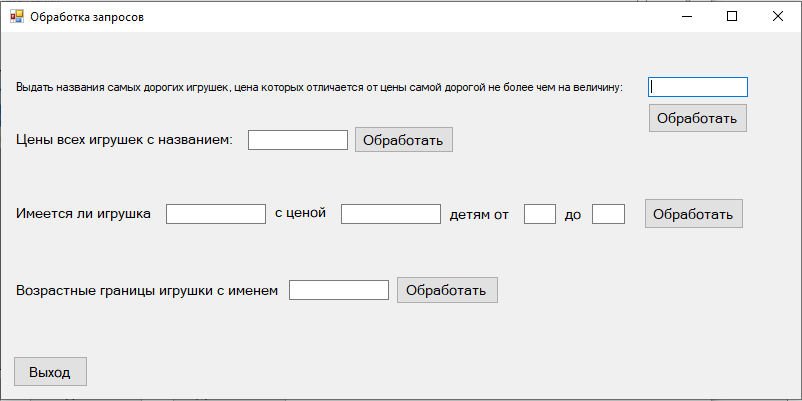


Рисунок 8 – Форма обработки запросов

5 Коды основных событий

Функция CheckFullNameFile():

public static bool CheckFullNameFile(string str, string ext)

{

if ((str.Length <= 0) || (ext.Length <= 0)

|| (str.Length < ext.Length)

|| (!CheckSymbol(str[0])) || (!CheckSymbol(str[str.Length - 1]))

|| (str.IndexOf('.') != str.LastIndexOf('.')))

return false;

string nameDirectory = "", nameFile = "";

int point = str.IndexOf('.');

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

nameDirectory += str[i];

point = str.LastIndexOf('\\');

for (int i = (point + 1); (i < str.Length); i++)

nameFile += str[i];

if ((nameDirectory.Length <= 0) || (nameFile.Length <= 0)

|| (!FileString.CheckNameDirectory(nameDirectory))

|| (!FileString.CheckNameFile(nameFile, ".txt")))

return false;

return true;

}

Функция CheckSymbol():

public static bool CheckSymbol(char symbol)

{

// Проверка символа на корректность (цифра, символ из латинского алфавита, кириллица и альтернатива пробелу

return (((symbol >= '0') && (symbol <= '9'))

|| ((symbol >= 'A') && (symbol <= 'Z'))

|| ((symbol >= 'a') && (symbol <= 'z'))

|| ((symbol >= 'А') && (symbol <= 'п'))

|| ((symbol >= 'р') && (symbol <= 'я'))

|| (symbol == '\_'));

}

Функция CheckNameDirectory():

public static bool CheckNameDirectory(string str)

{

//Проверка строки, содержащая название директории файла

if ((str.Length <= 0) || (str.IndexOf('.') >= 0)

|| (str.IndexOf(':') != str.LastIndexOf(':')) || (str.IndexOf(':') < 0)

|| (!CheckSymbol(str[0]))

|| ((!CheckSymbol(str[str.Length - 1])) && (str[str.Length - 1] != '\\')))

return false;

int point = str.IndexOf(':');

if (str[point + 1] != '\\')

return false;

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

if (!CheckSymbol(str[i]))

return false;

for (int i = (point + 1); i < str.Length; i++)

if ((str[i] != '\\') && (!CheckSymbol(str[i])) && (str[i] != ' '))

return false;

int count = 0; //длина последовательности символов \\ в строке str

for (int i = (point + 1); i < str.Length; i++)

{

if (str[i] == '\\')

count++;

else if (count > 1)

return false;

else

count = 0;

}

return true;

}

Функция CheckNameFile():

public static bool CheckNameFile(string str, string ext)

{

//Проверка имени файла

if ((str.Length <= 0) || (str.IndexOf('.') != str.LastIndexOf('.'))

|| (!CheckSymbol(str[0])) || (!CheckSymbol(str[str.Length - 1]))

|| (ext[0] != '.'))

return false;

int point = str.IndexOf('.');

if (point > 0)

{

if (!SubStrInStr(str, ext, point))

return false;

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

if (!CheckSymbol(str[i]))

return false;

return ((str.Length - point) == (ext.Length));

}

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

if (!CheckSymbol(str[i]))

return false;

return true;

}

Функция SubStrInStr():

public static bool SubStrInStr(string base\_string, string substr, int point)

{

//Данная функция проверяет, присутствует ли строка substr в строке base\_string с индекса point включительно

if ((base\_string.Length <= substr.Length) ||

(base\_string.Length <= 0) || (substr.Length <= 0)

|| (point >= base\_string.Length) || (point < 0))

return false;

int count = 0;

for (int i = point, j = 0; (i < base\_string.Length) && (j < substr.Length); i++, j++)

if (base\_string[i] == substr[j])

count++;

return (count == substr.Length);

}

**MainForm**

public partial class MainForm : Form

{

private CreateFileForm formCreateFile = null;

private OpenFileForm formOpenFile = null;

private CreateEntryForm formEntry = null;

private ViewStatForm formStatView = null;

Функция MainForm()(конструктор):

public MainForm()

{

InitializeComponent();

//Инициализация пунктов меню

ToolStripMenuItem fileItem = new ToolStripMenuItem("Файл");

ToolStripMenuItem createFile = new ToolStripMenuItem("Добавить файл"),

openFile = new ToolStripMenuItem("Открыть файл");

createFile.Click += this.createFile\_Click;

openFile.Click += this.openButton\_Click;

fileItem.DropDownItems.Add(createFile);

fileItem.DropDownItems.Add(openFile);

menuStrip1.Items.Add(fileItem);

ToolStripMenuItem dataAbout = new ToolStripMenuItem("О программе");

dataAbout.Click += AboutProgram;

menuStrip1.Items.Add(dataAbout);

}

Функция AboutProgram():

private void AboutProgram(object sender, EventArgs e)

{

MessageBox.Show("Автор программы: Солопов Даниил Дмитриевич\nНазвание программы: База данных игрушек" +

"(курсовой проект)\n\nЧто делает программа: создаёт/добавляет файл для работы, открывает файл" +

"осуществляет запись в файл данных, чтение данных из файла, просмотр данных.\n" +

"Осуществляет возможность изменения содержимого файла путём удаления/добавления записей.");

}

Функция SetAllCountEntry():

private bool SetAllCountEntry()//устанавливает общее кол-во записей в файле

{

if ((!File.Exists(label3.Text)) || (label3.Text.Length <= 0))

return false;

StreamReader fileReader = new StreamReader(label3.Text);

string[] allEntry = fileReader.ReadToEnd().Split('\n');

fileReader.Close();

for (int i = 0; i < (allEntry.Length - 1); i++)

if ((!(allEntry[i].Split(' ')[0].Equals(label3.Text)))

|| ((allEntry[i].Split(' ').Length != 5) && (allEntry[i].Split(' ').Length > 0)))

return false;

this.label5.Text = (allEntry.Length - 1).ToString();

return true;

}

Обработка события нажатия на кнопку addEntryButton:

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

{

if ((!File.Exists(label3.Text)) || (!formEntry.Entry.CheckAllAttribute()))

return;

StreamWriter fileWriter = new StreamWriter(label3.Text, true);

fileWriter.WriteLine(label3.Text + " "

+ formEntry.Entry.name\_toy + " "

+ formEntry.Entry.price\_toy.ToString() + " "

+ formEntry.Entry.valueFrom + " "

+ formEntry.Entry.valueBefore);

fileWriter.Close();

if (!SetAllCountEntry())

{

label5.Text = "";

label3.Text = "";

MessageBox.Show("Данный файл не может быть обработан! " +

"Файл либо не принадлежит базе данных, либо данные в файле испорчены", "Ошибка");

return;

}

MessageBox.Show("Запись добавлена!", "Уведомление");

formEntry.Entry.SetStandartValues();

}

Фукнция createFile\_Create():

private void createFile\_Create(object sender, EventArgs e)

{

if ((formCreateFile.NameFile == null) || (formCreateFile.NameFile == ""))

return;

for (int i = 0; i < this.checkedListFile.Items.Count; i++)

if (formCreateFile.NameFile == this.checkedListFile.Items[i].ToString())

{

MessageBox.Show("Файл с таким именем уже существует!", "Ошибка");

return;

}

if (!File.Exists(formCreateFile.NameFile))

{

string name\_directory = "";

for (int i = 0, point = formCreateFile.NameFile.LastIndexOf('\\'); i < point; i++)

name\_directory += formCreateFile.NameFile[i];

if (!Directory.Exists(name\_directory))

{

try

{

Directory.CreateDirectory(name\_directory);

}

catch (Exception)

{

MessageBox.Show("Введены не корректные данные!", "Ошибка");

formCreateFile.NameFile = "";

return;

}

}

File.Create(formCreateFile.NameFile).Close();

if (!File.Exists(formCreateFile.NameFile))

{

MessageBox.Show("Введены не корректные данные!", "Ошибка");

formCreateFile.NameFile = "";

return;

}

}

this.checkedListFile.Items.Add(formCreateFile.NameFile);

formCreateFile.NameFile = "";

}

Обработка события нажатия на кнопку createFile\_Click:

void createFile\_Click(object sender, EventArgs e)

{

formCreateFile = new CreateFileForm();

formCreateFile.Extension = ".txt";

formCreateFile.CreateButton\_CreateFileForm.Click += this.createFile\_Create;

formCreateFile.ViewButton\_CreateFileForm.Click += this.createFile\_Create;

formCreateFile.Left = this.Left; // задаём открываемой форме позицию слева равную позиции текущей формы

formCreateFile.Top = this.Top; // задаём открываемой форме позицию сверху равную позиции текущей формы

formCreateFile.Show(); // отображаем CreateFileForm

this.Hide(); // скрываем MainForm (this - текущая форма)

}

Обработка события нажатия на кнопку openFile\_Click:

void openFile\_Click(object sender, EventArgs e)

{

label5.Text = "";

label3.Text = "";

formOpenFile = new OpenFileForm();

formOpenFile.OpenButton.Click += this.openFileButton\_Click;

formOpenFile.ViewButton.Click += this.openFileButton\_Click;

formOpenFile.Left = this.Left;

formOpenFile.Top = this.Top;

formOpenFile.Activate();

formOpenFile.Show();

this.Hide();

}

Обработка события нажатия на кнопку addButton\_Click:

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

{

//добавление файла

createFile\_Click(sender, e);

}

Обработка события нажатия на кнопку openFileButton\_Click:

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

{

label5.Text = "";

if ((!FileString.CheckFullNameFile(formOpenFile.NameFile, ".txt")))

{

formOpenFile.NameFile = "";

return;

}

if ((!File.Exists(formOpenFile.NameFile)))

{

MessageBox.Show("Такого файла не существует!", "Ошибка");

formOpenFile.NameFile = "";

return;

}

label3.Text = formOpenFile.NameFile;

if (!SetAllCountEntry())

{

label3.Text = "";

MessageBox.Show("Данный файл не может быть обработан! " +

"Файл либо не принадлежит базе данных, либо данные в файле испорчены", "Ошибка");

return;

}

formOpenFile.NameFile = "";

}

Обработка события нажатия на кнопку openButton\_Click:

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

{

//открытие файла

label3.Text = "";

if(this.checkedListFile.CheckedItems.Count <= 0)

{

openFile\_Click(sender, e);

return;

}

label3.Text = this.checkedListFile.CheckedItems[0].ToString();

if (!SetAllCountEntry())

{

label3.Text = "";

label5.Text = "";

MessageBox.Show("Данный файл не может быть обработан! " +

"Файл либо не принадлежит базе данных, либо данные в файле испорчены", "Ошибка");

return;

}

this.unCheckedListFile\_Full();

}

Функция unCheckedListFile\_Full():

private void unCheckedListFile\_Full()

{

if (this.checkedListFile.CheckedItems.Count == 0)

return;

List<string> listString = new List<string>();

for (int i = 0; i < this.checkedListFile.Items.Count; i++)

listString.Add(this.checkedListFile.Items[i].ToString());

this.checkedListFile.Items.Clear();

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

this.checkedListFile.Items.Add(listString[i]);

}

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

{

if (this.checkedListFile.CheckedItems.Count == 0)

return;

if(MessageBox.Show("Удалить файл(-ы)?", "Действие", MessageBoxButtons.YesNo) == DialogResult.Yes)

{

while (this.checkedListFile.CheckedItems.Count > 0)

{

if (this.checkedListFile.CheckedItems[0].ToString() == label3.Text)

{

label3.Text = "";

label5.Text = "";

}

if (File.Exists(this.checkedListFile.CheckedItems[0].ToString()))

File.Delete(this.checkedListFile.CheckedItems[0].ToString());

this.checkedListFile.Items.Remove(this.checkedListFile.CheckedItems[0]);

}

}

while(this.checkedListFile.CheckedItems.Count > 0)

this.checkedListFile.Items.Remove(this.checkedListFile.CheckedItems[0]);

}

Функция SetDataButton\_Click():

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

{

if(label3.Text.Length <= 0)

{

MessageBox.Show("Нет открытого файла для добавления записи!", "Ошибка");

return;

}

formEntry = new CreateEntryForm();

formEntry.AddEntryButton.Click += this.addEntryButton\_Click;

formEntry.Left = this.Left;

formEntry.Top = this.Top;

formEntry.Activate();

formEntry.Show();

this.Hide();

}

Функция CloseFileButton\_Click():

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

{

this.label5.Text = "";

this.label3.Text = "";

}

Обработка события нажатия на кнопку viewEntryButton:

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

{

if(label3.Text.Length <= 0)

{

MessageBox.Show("Нет открытого файла для просмотра статистики!", "Ошибка");

return;

}

formStatView = new ViewStatForm(label3.Text);

formStatView.Left = this.Left;

formStatView.Top = this.Top;

formStatView.Activate();

formStatView.Show();

this.Hide();

}

}

public partial class CreateFileForm : Form

{

public System.Windows.Forms.Button ViewButton\_CreateFileForm

{

get

{

return this.viewButton;

}

}

public System.Windows.Forms.Button CreateButton\_CreateFileForm

{

get

{

return this.createButton;

}

}

public System.Windows.Forms.Button ExitButton\_CreateFileForm

{

get

{

return this.exitButton;

}

}

public string Extension

{

set

{

this.extension = value;

}

get

{

return this.extension;

}

}

public string NameFile

{

get

{

return this.openFile.FileName;

}

set

{

this.openFile.FileName = value;

}

}

};

public partial class OpenFileForm : Form

{

public string NameFile

{

get

{

return this.openFile.FileName;

}

set

{

this.openFile.FileName = value;

}

}

public System.Windows.Forms.Button OpenButton

{

get

{

return this.openButton;

}

}

public System.Windows.Forms.Button ViewButton

{

get

{

return this.viewButton;

}

}

};

public partial class CreateEntryForm : Form

{

public UserEntry Entry

{

get

{

return this.dynamicEntry;

}

}

public System.Windows.Forms.Button AddEntryButton

{

get

{

return this.addEntryButton;

}

}

};

**CreateFileForm**

public partial class CreateFileForm : Form

{

private OpenFileDialog openFile = null;

Функция CreateFileForm():

public CreateFileForm()

{

InitializeComponent();

this.FormClosed += this.exitButton\_Click;

openFile = new OpenFileDialog();

openFile.Title = "Обзор";

openFile.FileName = "";

openFile.Filter = "Текстовые файлы (\*.txt)|\*txt";

}

Обработка события нажатия на кнопку exitButton:

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

{

Form formMainWindow = Application.OpenForms[0];

formMainWindow.StartPosition = FormStartPosition.Manual;

formMainWindow.Left = this.Left;

formMainWindow.Top = this.Top;

formMainWindow.Show();

this.Hide();

}

Обработка события нажатия на кнопку createButton:

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

{

nameDirectory.Text = nameDirectory.Text.Trim(' ').Replace(' ', '\_');

nameFile.Text = nameFile.Text.Trim(' ').Replace(' ', '\_');

if((nameDirectory.TextLength <= 0) || (nameFile.TextLength <=0)

|| (!FileString.CheckNameDirectory(nameDirectory.Text))

|| (!FileString.CheckNameFile(nameFile.Text, ".txt")))

{

MessageBox.Show("Введены не корректные данные", "Ошибка");

nameDirectory.Text = "";

nameFile.Text = "";

return;

}

openFile.FileName = nameDirectory.Text;

if (openFile.FileName[openFile.FileName.Length - 1] != '\\')

openFile.FileName += "\\" + nameFile.Text;

else

openFile.FileName += nameFile.Text;

if (!FileString.SubStrInStr(openFile.FileName, this.extension, openFile.FileName.IndexOf('.')))

openFile.FileName += this.extension;

nameDirectory.Text = "";

nameFile.Text = "";

this.exitButton\_Click(sender, e);

return;

}

Обработка события нажатия на кнопку viewButton:

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

{

if (openFile.ShowDialog() != DialogResult.OK)

{

openFile.FileName = "";

return;

}

if (!FileString.CheckFullNameFile(openFile.FileName, ".txt"))

{

MessageBox.Show("Введены не корректные данные", "Ошибка");

openFile.FileName = "";

return;

}

this.exitButton\_Click(sender, e);

}

}

**OpenFileForm**

public partial class OpenFileForm : Form

{

private OpenFileDialog openFile = null;

Функция OpenFileForm():

public OpenFileForm()

{

InitializeComponent();

this.FormClosed += exitButton\_Click;

openFile = new OpenFileDialog();

openFile.Title = "Обзор";

openFile.FileName = "";

openFile.Filter = "Текстовые файлы (\*.txt)|\*.txt";

}

Обработка события нажатия на кнопку exitButton:

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

{

Form formMainWindow = Application.OpenForms[0];

formMainWindow.StartPosition = FormStartPosition.Manual;

formMainWindow.Left = this.Left;

formMainWindow.Top = this.Top;

formMainWindow.Show();

this.Hide();

}

Обработка события нажатия на кнопку openButtonClick:

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

{

openFile.FileName = this.textBox1.Text.Trim(' ').Replace(' ', '\_');

if (!FileString.CheckFullNameFile(openFile.FileName, ".txt"))

{

MessageBox.Show("Введены не корректные данные", "Ошибка");

openFile.FileName = "";

return;

}

this.textBox1.Text = "";

this.exitButton\_Click(sender, e);

}

Обработка события нажатия на кнопку viewButton:

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

{

if (openFile.ShowDialog() != DialogResult.OK)

{

openFile.FileName = "";

return;

}

if (!FileString.CheckFullNameFile(openFile.FileName, ".txt"))

{

MessageBox.Show("Введены не корректные данные", "Ошибка");

openFile.FileName = "";

return;

}

this.exitButton\_Click(sender, e);

}

}

**CreateEntryForm**

Структура UserEntry:

public struct UserEntry

{

public string name\_toy;

public double price\_toy;

public int valueFrom;

public int valueBefore;

public bool CheckAllAttribute()

{

if ((name\_toy.Length <= 0) || (price\_toy <= 0.0)

|| (valueFrom <= 0) || (valueBefore <= 0))

return false;

bool f = false;

for (int i = 0; (i < name\_toy.Length) && (!f); i++)

if (((name\_toy[i] >= 'a') && (name\_toy[i] <= 'z'))

|| ((name\_toy[i] >= 'A') && (name\_toy[i] <= 'Z'))

|| ((name\_toy[i] >= '1') && (name\_toy[i] <= '9'))

|| ((name\_toy[i] >= 'А') && (name\_toy[i] <= 'п'))

|| ((name\_toy[i] >= 'р') && (name\_toy[i] <= 'я')))

f = true;

return f;

}

public void SetStandartValues()

{

name\_toy = "";

price\_toy = 0;

valueFrom = valueBefore = 0;

}

public void SetValues(string ntoy, double ptoy, int vFrom, int vBefore)

{

name\_toy = ntoy;

price\_toy = ptoy;

valueFrom = vFrom;

valueBefore = vBefore;

}

}

public partial class CreateEntryForm : Form

{

Функция CreateEntryForm():

public CreateEntryForm()

{

InitializeComponent();

this.FormClosed += this.ExitButton\_Click;

dynamicEntry.SetStandartValues();

this.priceToyBox.MaxLength = 10;

this.valueFromBox.MaxLength = 2;

this.valueBeforeBox.MaxLength = 2;

}

Обработка события нажатия на кнопку exitButton:

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

{

Form formMainWindow = Application.OpenForms[0];

formMainWindow.StartPosition = FormStartPosition.Manual;

formMainWindow.Left = this.Left;

formMainWindow.Top = this.Top;

formMainWindow.Show();

this.Hide();

}

Обработка события нажатия на кнопку addEntryButton:

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

{

dynamicEntry.SetStandartValues();

this.nameToyBox.Text = this.nameToyBox.Text.Trim(' ');

if ((this.nameToyBox.Text.Length <= 0) || (this.priceToyBox.Text.Length <= 0)

|| (this.valueFromBox.Text.Length <= 0) || (this.valueBeforeBox.Text.Length <= 0))

{

MessageBox.Show("Необходимо заполнить все поля ввода!", "Ошибка");

return;

}

this.nameToyBox.Text = this.nameToyBox.Text.Replace(' ', '\_');

dynamicEntry.SetValues(this.nameToyBox.Text, System.Convert.ToDouble(this.priceToyBox.Text),

System.Convert.ToInt32(this.valueFromBox.Text), System.Convert.ToInt32(this.valueBeforeBox.Text));

if((dynamicEntry.valueFrom > dynamicEntry.valueBefore) || (!dynamicEntry.CheckAllAttribute()))

{

dynamicEntry.SetStandartValues();

MessageBox.Show("Введены не корректные данные!", "Ошибка");

}

this.nameToyBox.Text = "";

this.priceToyBox.Text = "";

this.valueFromBox.Text = "";

this.valueBeforeBox.Text = "";

this.flagPoint\_Press = false;

}

private bool flagPoint\_Press = false;

Обработка события нажатия на клавишу при фокусе на priceToyBox:

private void priceToy\_Press(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (this.priceToyBox.Text.IndexOf(',') < 0)

this.flagPoint\_Press = false;

if (((!Char.IsDigit(number)) && (number != ',') && (number != 8))

|| ((number == ',') && (this.flagPoint\_Press)))

e.Handled = true;

if ((number == ',') && (!this.flagPoint\_Press))

this.flagPoint\_Press = true;

}

Обработка события нажатия на клавишу при фокусе на valueFromBox:

private void valueFrom\_Press(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!Char.IsDigit(number) && (number != 8))

e.Handled = true;

}

Обработка события нажатия на клавишу при фокусе на valueBeforeBox:

private void valueBefore\_Press(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!Char.IsDigit(number) && (number != 8))

e.Handled = true;

}

}

**ViewStatForm**

public partial class ViewStatForm : Form

{

private List<UserEntry> userEntries = null; //Все данные содержащиеся в файле

private string full\_name\_file; //полное имя файла

private RequestForm formRequest = null;

private bool openRequestForm;

Функция ViewStatForm():

public ViewStatForm()

{

InitializeComponent();

this.FormClosed += ExitButton\_Click;

this.openRequestForm = false;

}

Функция ViewStatForm():

public ViewStatForm(string fullNameFile)

{

InitializeComponent();

this.FormClosed += ExitButton\_Click;

EnterInBaseData(fullNameFile);

this.full\_name\_file = fullNameFile;

this.openRequestForm = false;

}

Функция DefineMaxMinPriceToy():

private void DefineMaxMinPriceToy(bool c)

{

if ((this.userEntries == null) || (this.userEntries.Count <= 0))

return;

int index = 0;

for (int i = 1; i < this.userEntries.Count; i++)

if (((this.userEntries[index].price\_toy > this.userEntries[i].price\_toy) && (!c))

|| ((this.userEntries[index].price\_toy < this.userEntries[i].price\_toy) && (c)))

index = i;

if (!c)

{

this.nameMinPriceToy.Text = this.userEntries[index].name\_toy;

this.minPrice.Text = this.userEntries[index].price\_toy.ToString();

}

else

{

this.nameMaxPriceToy.Text = this.userEntries[index].name\_toy;

this.maxPrice.Text = this.userEntries[index].price\_toy.ToString();

}

}

Функция EnterDataInList():

private bool EnterDataInList(string fullNameFile)

{

//Запись данных из файла в массив структур userEntries

if ((!File.Exists(fullNameFile)) || (fullNameFile.Length <= 0))

return false;

StreamReader fileReader = new StreamReader(fullNameFile);

string[] allEntry = fileReader.ReadToEnd().Split('\n'); //Удаление переводов строки

fileReader.Close();

this.userEntries = new List<UserEntry>();

for(int i = 0; i < (allEntry.Length-1); i++)

{

string[] entry = allEntry[i].Split(' ');

entry[1] = entry[1].Replace('\_', ' '); //Возврат пробелов

UserEntry us = new UserEntry();

us.SetValues(entry[1], System.Convert.ToDouble(entry[2]),

System.Convert.ToInt32(entry[3]), System.Convert.ToInt32(entry[4]));

this.userEntries.Add(us);

}

return true;

}

Функция EnterInBaseData():

private void EnterInBaseData(string fullNameFile)

{

//Вставляем в базу данных данные из userEntries соответствующие требованиям задачи

if (!EnterDataInList(fullNameFile))

return;

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

{

this.dataBaseView.Rows.Add(this.userEntries[i].name\_toy,

(this.userEntries[i].price\_toy.ToString() + " руб."), ("от " + this.userEntries[i].valueFrom.ToString() +

" до " + this.userEntries[i].valueBefore.ToString()));

this.dataBaseView.Rows[i].ReadOnly = true;

this.dataBaseView.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

}

this.DefineMaxMinPriceToy(false);

this.DefineMaxMinPriceToy(true);

return;

}

Обработка события нажатия на кнопку exitButton:

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

{

if (this.openRequestForm)

this.formRequest.Hide();

Form formMainWindow = Application.OpenForms[0];

formMainWindow.StartPosition = FormStartPosition.Manual;

formMainWindow.Left = this.Left;

formMainWindow.Top = this.Top;

formMainWindow.Show();

this.Hide();

}

Обработка события перерисовывания таблицы dataBaseView:

private void dbPrePaint(object sender, DataGridViewRowPrePaintEventArgs e)

{

//Нумерация строк таблицы

int index = e.RowIndex;

string indexStr = (index + 1).ToString();

object header = this.dataBaseView.Rows[index].HeaderCell.Value; //Обращение к заголовку строки

if ((header == null) || (!header.Equals(indexStr)))

this.dataBaseView.Rows[index].HeaderCell.Value = indexStr; //Присвоение заголовку значения

}

Обработка события нажатия на кнопку setEntryButton:

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

{

//Вызываем форму добавления записи

if (this.openRequestForm)

this.formRequest.Hide();

this.Hide();

((MainForm)Application.OpenForms[0]).SetData();

}

Функция InputDataInFile:

private void InputDataInFile(string fullNameFile)

{

if ((this.userEntries == null) || (!File.Exists(fullNameFile)) || (this.userEntries.Count <= 0))

return;

StreamWriter fileWriter = new StreamWriter(fullNameFile, false);

List<UserEntry> buffer = new List<UserEntry>();

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

{

UserEntry us = new UserEntry();

us.SetValues(this.userEntries[i].name\_toy.Replace(' ', '\_'),

this.userEntries[i].price\_toy,

this.userEntries[i].valueFrom,

this.userEntries[i].valueBefore);

buffer.Add(us);

}

this.userEntries.Clear();

this.userEntries = buffer;

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

{

fileWriter.WriteLine(fullNameFile + " " + this.userEntries[i].name\_toy + " " +

this.userEntries[i].price\_toy.ToString() + " " +

this.userEntries[i].valueFrom.ToString() + " " +

this.userEntries[i].valueBefore.ToString());

}

fileWriter.Close();

this.userEntries.Clear();

}

Обработка события нажатия на кнопку deleteButton:

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

{

if ((this.userEntries == null) || (this.userEntries.Count <= 0) || (this.dataBaseView.CurrentRow.Index < 0))

return;

this.userEntries.RemoveAt(this.dataBaseView.CurrentRow.Index);

if ((this.dataBaseView.CurrentRow.Index == 0) && (this.dataBaseView.Rows.Count <= 1))

{

this.userEntries.Clear();

this.dataBaseView.Rows.Clear();

File.Delete(this.full\_name\_file);

File.Create(this.full\_name\_file).Close();

((MainForm)Application.OpenForms[0]).SetAllEntry();

this.nameMinPriceToy.Text = "";

this.minPrice.Text = "";

this.nameMaxPriceToy.Text = "";

this.maxPrice.Text = "";

return;

}

this.dataBaseView.Rows.Clear();

this.InputDataInFile(this.full\_name\_file);

this.EnterInBaseData(this.full\_name\_file);

((MainForm)Application.OpenForms[0]).SetAllEntry();

}

Обработка события нажатия на кнопку procButton:

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

{

if (this.openRequestForm)

return;

formRequest = new RequestForm();

formRequest.Left = this.Left;

formRequest.Top = this.Top;

formRequest.Activate();

formRequest.Show();

this.openRequestForm = true;

}

}

**RequestForm**

public partial class RequestForm : Form

{

Функция RequestForm():

public RequestForm()

{

InitializeComponent();

this.FormClosed += ExitButton\_Click;

this.haveValueFromBox.MaxLength = 2;

this.haveValueBeforeBox.MaxLength = 2;

this.havePriceToyBox.MaxLength = 10;

this.allMaxPriceBox.MaxLength = 10;

}

Обработка события нажатия на кнопку ExitButton:

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

{

this.Hide();

((MainForm)(Application.OpenForms[0])).ViewStateForm.OpenFormRequest = false;

}

Обработка события нажатия на кнопку ageLimitButton:

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

{

this.ageLimitBox.Text = this.ageLimitBox.Text.Trim(' ');

if (this.ageLimitBox.Text.Length <= 0){

MessageBox.Show("Необходимо заполнить поле для имени игрушки!", "Ошибка");

return;

}

List<UserEntry> listUser = ((MainForm)(Application.OpenForms[0])).ViewStateForm.AllEntry;

int index = (-1);

for (int i = 0; (i < listUser.Count) && (index < 0); i++)

if (listUser[i].name\_toy.Equals(this.ageLimitBox.Text))

index = i;

if (index < 0)

{

MessageBox.Show("Игрушки с таким именем в базе данных нет", "Уведомление");

this.ageLimitBox.Text = "";

return;

}

List<int> indexList = new List<int>();

indexList.Add(index);

for (int i = (index + 1); (i < listUser.Count); i++)

if (listUser[i].name\_toy.Equals(this.ageLimitBox.Text))

indexList.Add(i);

string data\_toys = "";

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

data\_toys += "от " + listUser[indexList[i]].valueFrom.ToString()

+ " до " + listUser[indexList[i]].valueBefore.ToString() + "\n";

MessageBox.Show("Возрастные границы игрушки " + this.ageLimitBox.Text + ":\n\n"

+ data\_toys + "\n", "Информация");

this.ageLimitBox.Text = "";

}

Обработка события нажатия на кнопку haveButtonClick:

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

{

this.haveNameToyBox.Text = this.haveNameToyBox.Text.Trim(' ');

if ((this.haveNameToyBox.Text.Length <= 0) || (this.havePriceToyBox.Text.Length <= 0)

|| (this.haveValueFromBox.Text.Length <= 0) || (this.haveValueBeforeBox.Text.Length <= 0))

{

MessageBox.Show("Необходимо заполнить поле для имени игрушки!", "Ошибка");

return;

}

List<UserEntry> listUser = ((MainForm)(Application.OpenForms[0])).ViewStateForm.AllEntry;

int counter = 0;

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

if (listUser[i].name\_toy.Equals(this.haveNameToyBox.Text)

&& listUser[i].price\_toy.Equals(System.Convert.ToDouble(this.havePriceToyBox.Text))

&& listUser[i].valueFrom.Equals(System.Convert.ToInt32(this.haveValueFromBox.Text))

&& listUser[i].valueBefore.Equals(System.Convert.ToInt32(this.haveValueBeforeBox.Text)))

counter++;

MessageBox.Show("Количество записей удовлетворяющих условию запроса: " + counter.ToString(), "Информация");

this.haveNameToyBox.Text = "";

this.havePriceToyBox.Text = "";

this.haveValueFromBox.Text = "";

this.haveValueBeforeBox.Text = "";

}

Обработка события нажатия на кнопку allNameButton:

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

{

this.allNameToyBox.Text = this.allNameToyBox.Text.Trim(' ');

if (this.allNameToyBox.Text.Length <= 0)

{

MessageBox.Show("Необходимо заполнить поле для имени игрушки!", "Ошибка");

return;

}

List<UserEntry> listUser = ((MainForm)(Application.OpenForms[0])).ViewStateForm.AllEntry;

int index = (-1);

for (int i = 0; (i < listUser.Count) && (index < 0); i++)

if (listUser[i].name\_toy.Equals(this.allNameToyBox.Text))

index = i;

if (index < 0)

{

MessageBox.Show("Игрушки с таким именем в базе данных нет", "Уведомление");

this.allNameToyBox.Text = "";

return;

}

string allPrice = "";

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

if (listUser[i].name\_toy.Equals(this.allNameToyBox.Text))

allPrice += listUser[i].price\_toy.ToString() + " руб.\n";

MessageBox.Show("Цены всех игрушек с именем " + this.allNameToyBox.Text + ": \n\n" + allPrice + "\n", "Информация");

this.allNameToyBox.Text = "";

}

Обработка события нажатия на кнопку allMaxPrice:

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

{

if(this.allMaxPriceBox.Text.Length <= 0)

{

MessageBox.Show("Необходимо заполнить поле для имени игрушки!", "Ошибка");

return;

}

List<UserEntry> listUser = ((MainForm)(Application.OpenForms[0])).ViewStateForm.AllEntry;

double max\_price = ((MainForm)(Application.OpenForms[0])).ViewStateForm.MaxPrice,

def\_price = System.Convert.ToDouble(this.allMaxPriceBox.Text);

string data\_toys = "";

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

if ((listUser[i].price\_toy + def\_price).Equals(max\_price))

data\_toys += listUser[i].name\_toy + " " +

listUser[i].price\_toy.ToString() + " .руб\n";

this.allMaxPriceBox.Text = "";

if (data\_toys.Length <= 0)

{

MessageBox.Show("Записей удовлетворяющих данному запросу в базе данных нет", "Уведомление");

return;

}

MessageBox.Show("Записи удовлетворяющие данному запросу: \n\n" + data\_toys + "\n\n");

}

private bool flagPointHaveBox\_Press = false;

Обработка события нажатия на клавишу при фокусе на havePriceBox:

private void havePriceBox\_Press(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (this.havePriceToyBox.Text.IndexOf(',') < 0)

this.flagPointHaveBox\_Press = false;

if (((!Char.IsDigit(number)) && (number != ',') && (number != 8))

|| ((number == ',') && (this.flagPointHaveBox\_Press)))

e.Handled = true;

if ((number == ',') && (!this.flagPointHaveBox\_Press))

this.flagPointHaveBox\_Press = true;

}

Обработка события нажатия на клавишу при фокусе на haveValueFromBox:

private void haveValueFromBox\_Press(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!Char.IsDigit(number) && (number != 8))

e.Handled = true;

}

Обработка события нажатия на клавишу при фокусе на haveValueBeforeBox:

private void haveValueBeforeBox\_Press(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!Char.IsDigit(number) && (number != 8))

e.Handled = true;

}

private bool flagPointPriceBox\_Press = false;

Обработка события нажатия на клавишу при фокусе на allMaxPriceBox\_Press:

private void allMaxPriceBox\_Press(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (this.allMaxPriceBox.Text.IndexOf(',') < 0)

this.flagPointPriceBox\_Press = false;

if (((!Char.IsDigit(number)) && (number != ',') && (number != 8))

|| ((number == ',') && (this.flagPointPriceBox\_Press)))

e.Handled = true;

if ((number == ',') && (!this.flagPointPriceBox\_Press))

this.flagPointPriceBox\_Press = true;

}

}

public partial class MainForm : Form

{

public ViewStatForm ViewStateForm

{

get

{

return this.formStatView;

}

}

}

public partial class ViewStatForm : Form

{

public double MaxPrice

{

get

{

if (this.maxPrice.Text.Length <= 0)

return 0;

return System.Convert.ToDouble(this.maxPrice.Text);

}

}

public List<UserEntry> AllEntry

{

get

{

return this.userEntries;

}

}

public bool OpenFormRequest

{

get

{

return this.openRequestForm;

}

set

{

this.openRequestForm = value;

}

}

}

6 Таблица тестов

В этом разделе описаны примеры тестов для проверки работоспособности программы.

Таблица 11 – Тесты для проверки работоспособности программы

|  |  |  |  |
| --- | --- | --- | --- |
| **N теста** | **Назначение теста** | **Входные данные для теста** | **Выходные данные для теста** |
| 1 | Создание/добавление файла(успех) | C:\Directory  Name File.txt | Сообщение “Файл добавлен!” (Рис.9) |
| 2 | Создание/добавление файла(ошибка) |  | Сообщение “Введены не корректные данные” (Рис.10) |
| 3 | Создание/добавление файла(ошибка) | DISK\\:Restore  Game Toys.txt | Сообщение “Введены не корректные данные” (Рис.11) |
| 4 | Создание/добавление(ошибка) | C:\Directory  Name File | Сообщение “Файл с таким именем уже существует или добавлен” (Рис.12) |
| 5 | Открытие(успех) | C\Directory\Name File.txt | Сообщение “Файл открыт!” (Рис. 13) |
| 6 | Открытие(ошибка) | C:\DDDD\2,..1,4L12;5 | Сообщение “Введены не корректные данные” (Рис.14) |

Продолжение таблицы 11

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | Открытие(ошибка) | C:\No\_Directory\No\_File.txt | Сообщение “Такого файла не существует!” (Рис.15) |
| 8 | Добавление записи(при не открытом файле)(ошибка) |  | Сообщение “Нет открытого файла для добавления записи!” (Рис.16) |
| 9 | Добавление записи(при открытом файле(например C:\Directory\Name File.txt)) | Игрушка 1  250  5 8 | Сообщение “Запись добавлена” (Рис.17) |
| 10 | Добавление записи(при открытом файле(например C:\Directory\Name File.txt)) | Игрушка 2  240  9 5 | Сообщение “Введены не корректные данные!” (Рис. 18) |
| 11 | Добавление записи(при открытом файле(например C:\Directory\Name File.txt)) |  | Сообщение “Необходимо заполнить все поля ввода!” (Рис.19) |
| 12 | Добавление записи(при открытом файле(например C:\Directory\Name File.txt)) | Игрушка 1  1 5 | Сообщение “Необходимо заполнить все поля ввода!” (Рис.20) |
| 13 | Просмотр статистики файла(при не открытом файле)(ошибка) |  | Сообщение “Нет открытого файла для просмотра статистики!” (Рис.21) |

Продолжение таблицы 11

|  |  |  |  |
| --- | --- | --- | --- |
| 14 | Просмотр статистики файла(при открытом файле(например C:\Directory\Name File.txt)) |  | (Рис. 22) |
| 15 | Обработка 1-го запроса(успех) | 50 | Сообщение “Записи удовлетворяющие данному запросу: Игрушка 2 200 руб.” (Рис. 23) |
| 16 | Обработка 1-го запроса(успех) | 11 | Сообщение “Записей удовлетворяющих данному запросу в базе данных нет” (Рис. 24) |
| 17 | Обработка 1-го запроса(ошибка) |  | Сообщение “Необходимо заполнить поле для ввода дынных!” (Рис. 25) |
| 18 | Обработка 2-го запроса(успех) | Игрушка 1 | Сообщение “Цены всех игрушек с именем Игрушка 1:  250 руб.”(Рис. 26) |
| 19 | Обработка 2-го запроса(ошибка) |  | Сообщение “Необходимо заполнить поле для ввода данных” (Рис. 27) |
| 20 | Обработка 2-го запроса(успех) | Игрушка 4 | Сообщение “Игрушки с таким именем в базе данных нет”(Рис. 28) |
| 21 | Обработка 3-го запроса(успех) | Игрушка 1 250 5 8 | Сообщение “Количество записей удовлетворяющих условию запроса: 1” (Рис. 29) |
| 22 | Обработка 3-го запроса(ошибка) | Игрушка 4 9 | Сообщение “Необходимо заполнить поля для ввода данных!” (Рис. 30) |

Продолжение таблицы 11

|  |  |  |  |
| --- | --- | --- | --- |
| 23 | Обработка 4-го запроса(успех) | Игрушка 1 | Сообщение “Возрастные границы игрушки Игрушка 1:  от 5 до 8” (Рис. 31) |
| 24 | Обработка 4-го запроса(ошибка) |  | Сообщение “Необходимо заполнить поле для ввода данных!” (Рис. 32) |
| 25 | Обработка 4-го запроса(успех) | Игрушка 4 | Сообщение “Игрушки с таким именем в базе данных нет”(Рис. 33) |

7 Результаты тестирования

В этом разделе показан внешний вид и выполненное событие во всех формах при выполнении тестов.

Результат тестирования (тест 1), когда файл был создан и добавлен:

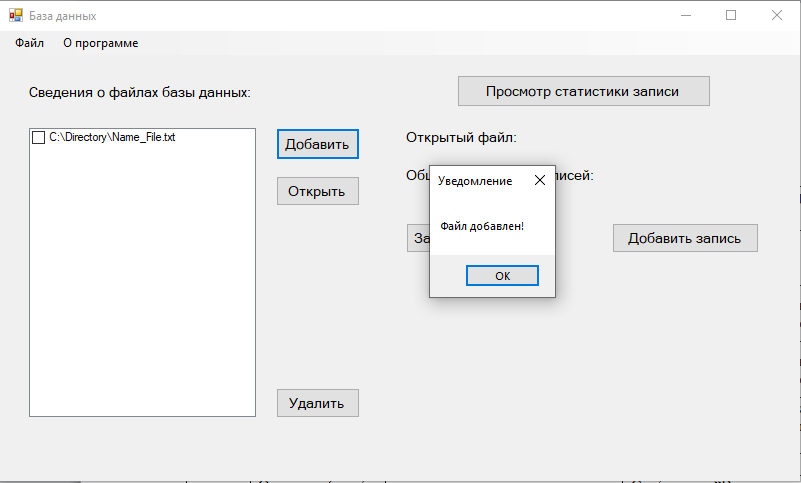


Рисунок 9 – Результат теста 1

Результат тестирования (тест 2), когда поля ввода данных оказались пустыми:

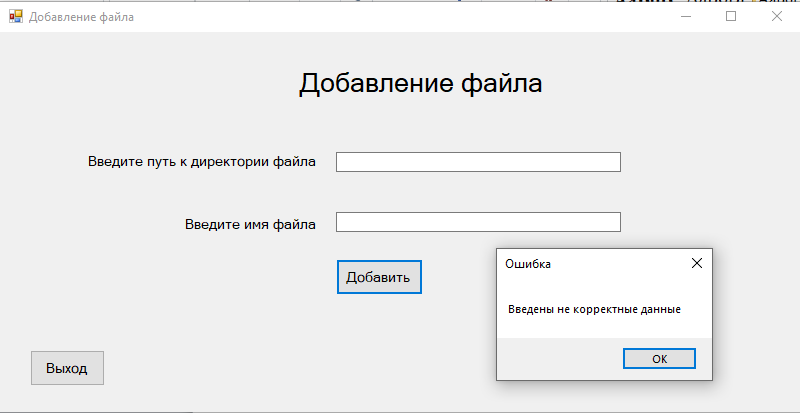


Рисунок 10 – Результат теста 2

Результат тестирования (тест 3), когда данные не корректны:

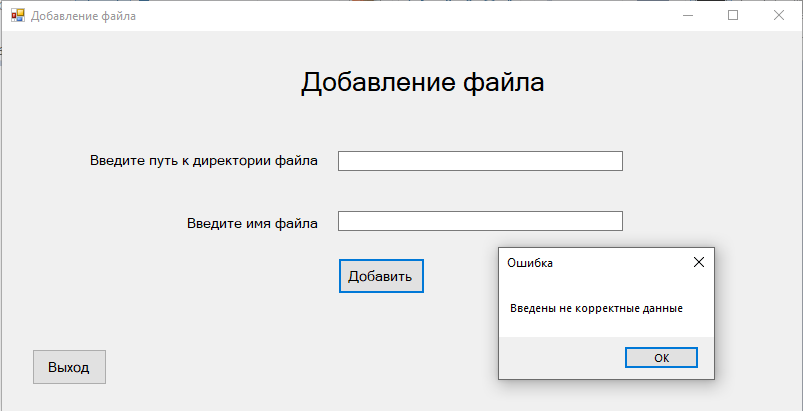


Рисунок 11 – Результат теста 3

Результат тестирования (тест 4), когда файл с таким именем уже добавлен:

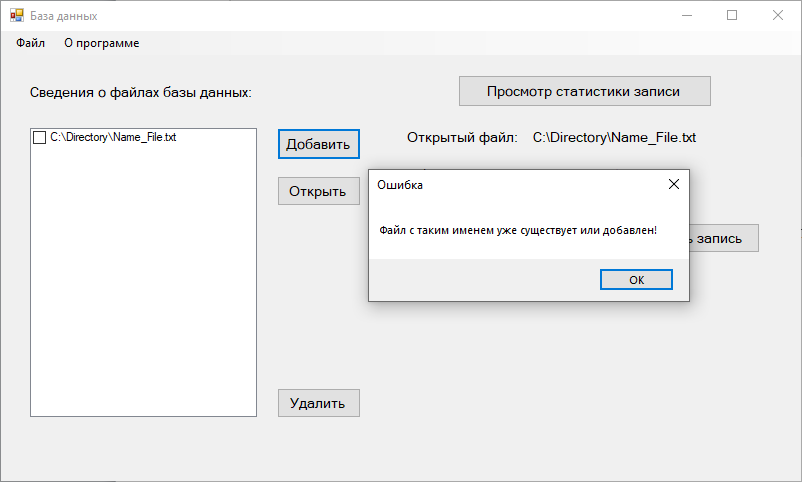


Рисунок 12 – Результат теста 4

Результат тестирования (тест 5), когда файл открыт:

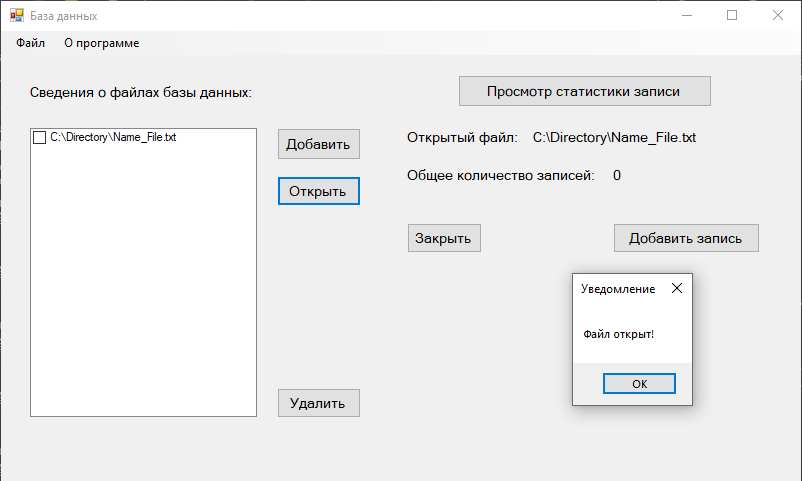


Рисунок 13 – Результат теста 5

Результат тестирования (тест 6), когда в поле ввода для открытия файла введены не корректные данные:

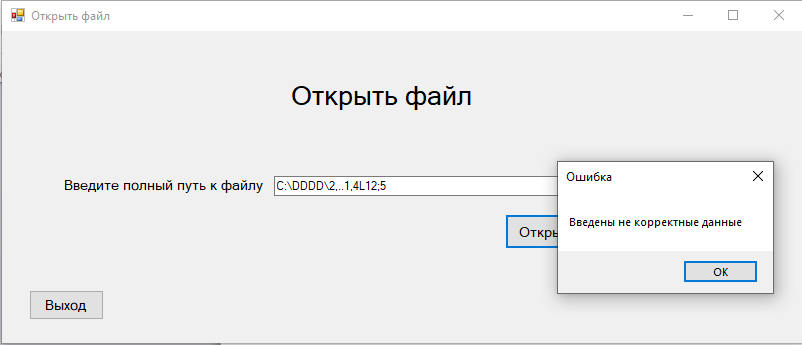


Рисунок 14 – Результат теста 6

Результат тестирования (тест 7), когда пользователь пытается открыть файл, которого не существует:

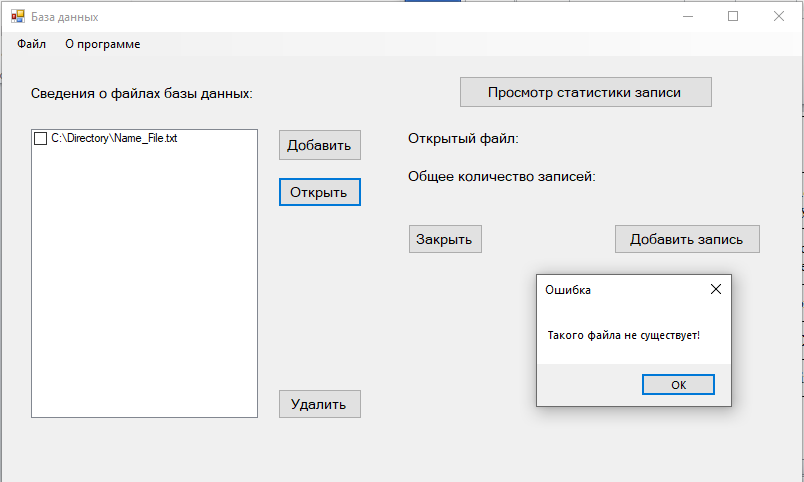


Рисунок 15 – Результат теста 7

Результат тестирования (тест 8), когда пользователь пытается добавить запись при не открытом файле:

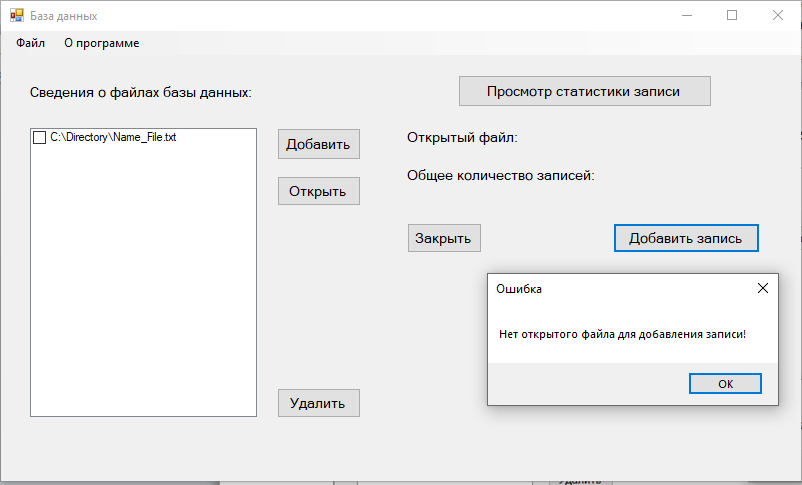


Рисунок 16 – Результат теста 8

Результат тестирования (тест 9), когда пользователь добавляет запись удачно(при открытом файле):

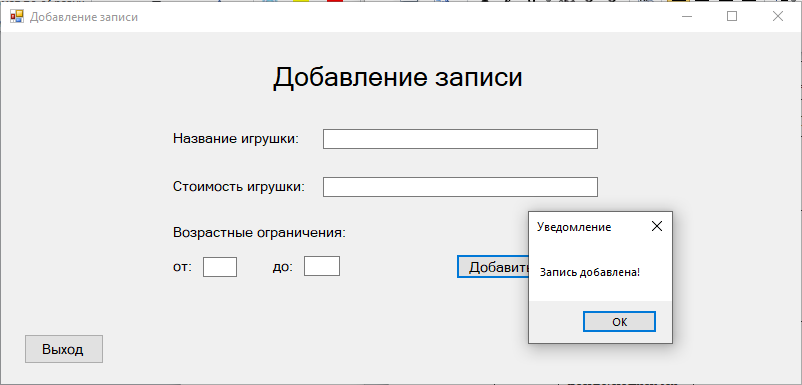


Рисунок 17 – Результат теста 9

Результат тестирования (тест 10), когда пользователь вводит некорректные данные:

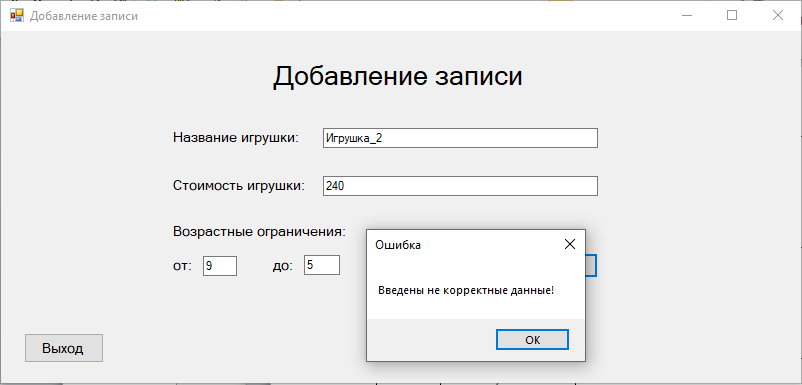


Рисунок 18 – Результат теста 10

Результат тестирования (тест 11), когда пользователь добавляет запись, но поля для ввода пусты:

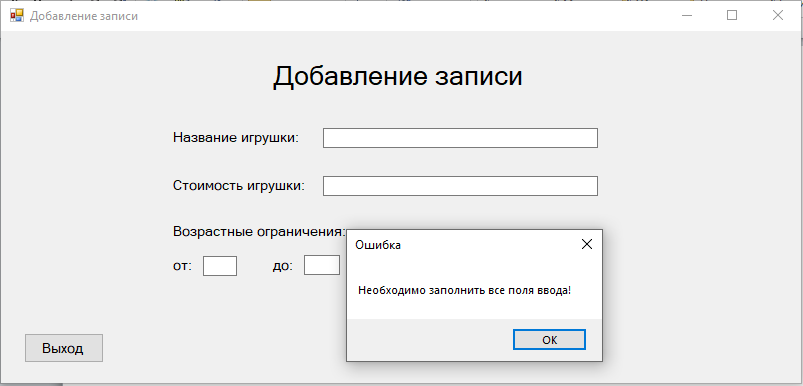


Рисунок 19 – Результат теста 11

Результат тестирования (тест 12), когда пользователь заполняет не все поля для ввода:

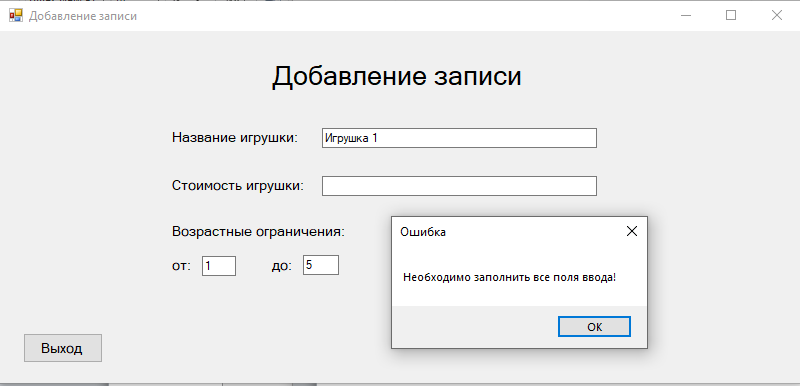


Рисунок 20 – Результат теста 12

Результат тестирования (тест 13), когда пользователь пытается просмотреть статистику не открытого файла:

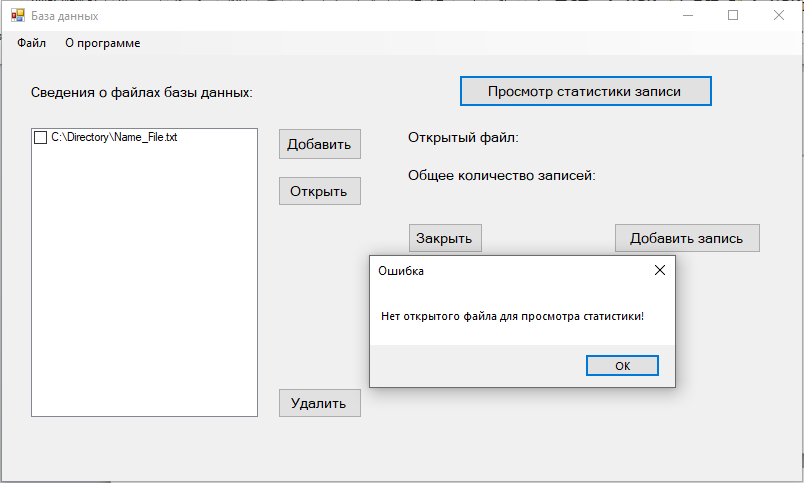


Рисунок 21 – Результат теста 13

Результат тестирования (тест 14), когда пользователь просматривает статистику открытого файла:

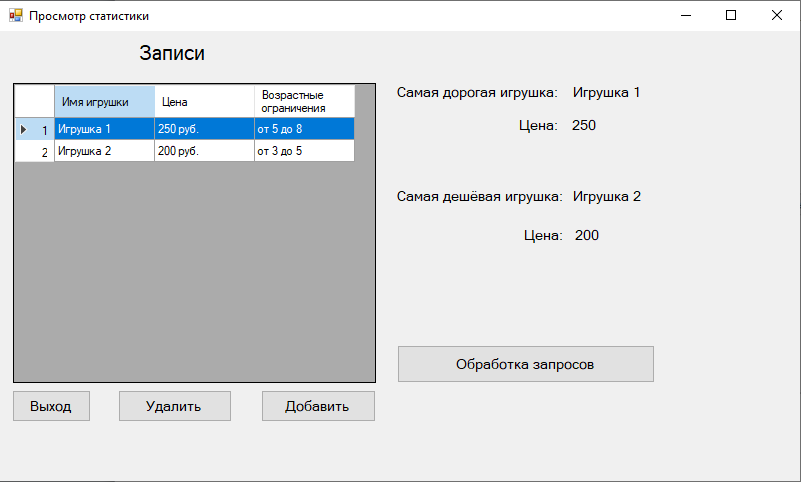


Рисунок 22 – Результат теста 14

Результат тестирования (тест 15), когда пользователь задаёт число для первого запроса:

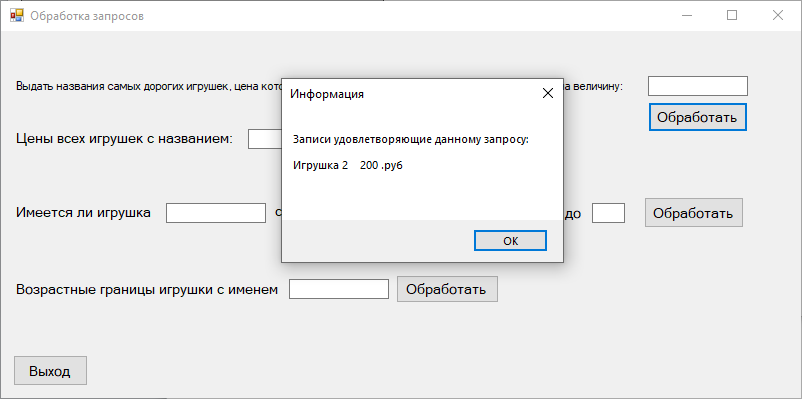


Рисунок 23 – Результат теста 15

Результат тестирования (тест 16), когда пользователь задаёт число для первого запроса:

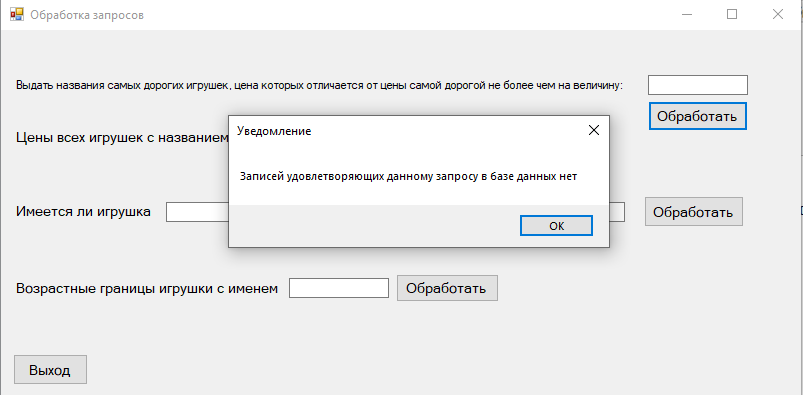


Рисунок 24 – Результат теста 16

Результат тестирования (тест 17), когда пользователь ведёт работу с первым запросом, но оставил поле для ввода данных пустым:

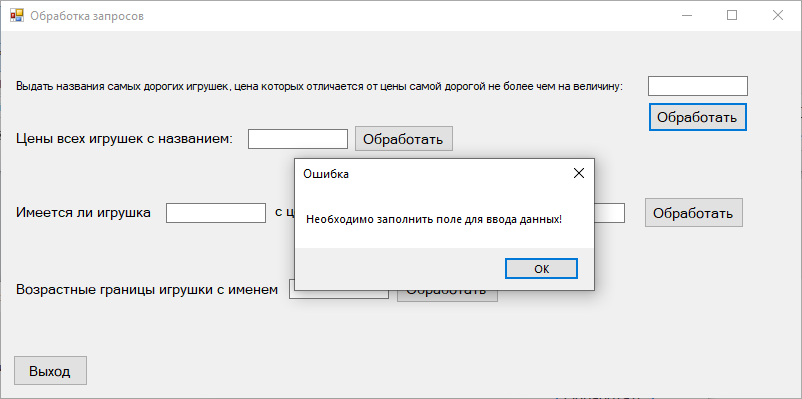


Рисунок 25 – Результат теста 17

Результат тестирования (тест 18), когда пользователь задаёт значение для работы со вторым запросом:

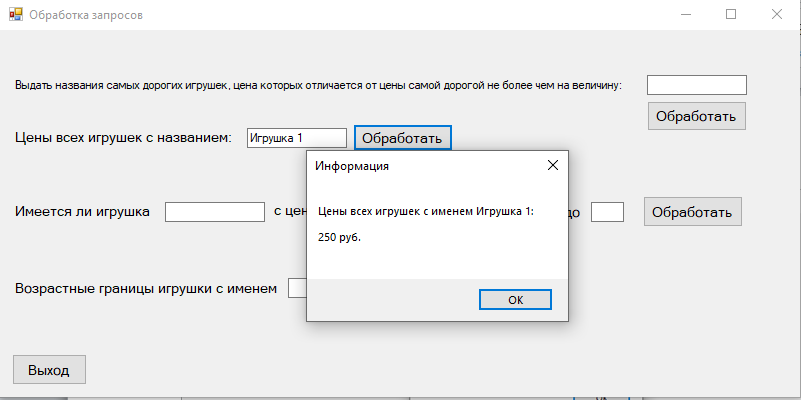


Рисунок 26 – Результат теста 18

Результат тестирования (тест 19), когда пользователь ведёт работу со вторым запросом, но оставил поле для ввода данных пустым:

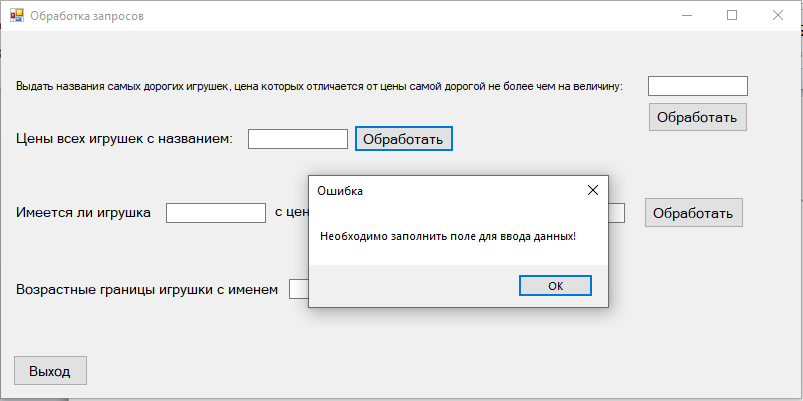


Рисунок 27 – Результат теста 19

Результат тестирования (тест 20), когда пользователь вводит имя игрушки, которой в базе данных нет:

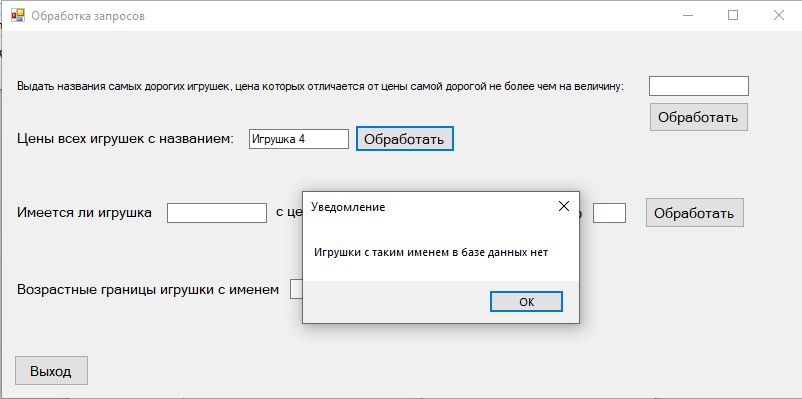


Рисунок 28 – Результат теста 20

Результат тестирования (тест 21), когда пользователь ведёт работу с 3-им запросом:

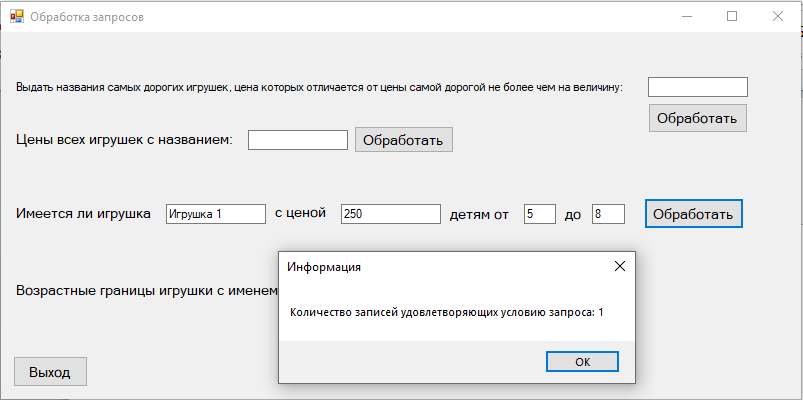


Рисунок 29 – Результат теста 21

Результат тестирования (тест 22), когда пользователь ведёт работу с 3-им запросом, но оставляет некоторые поля для ввода данных пустыми:

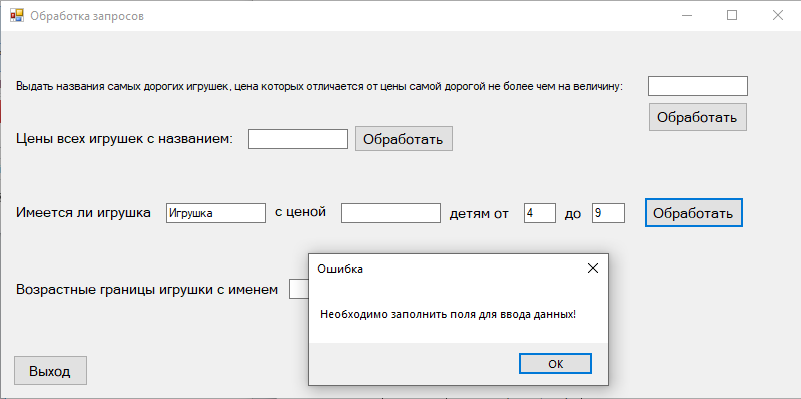


Рисунок 30 – Результат теста 22

Результат тестирования (тест 23), когда пользователь ведёт работу с 4-ым запросом:

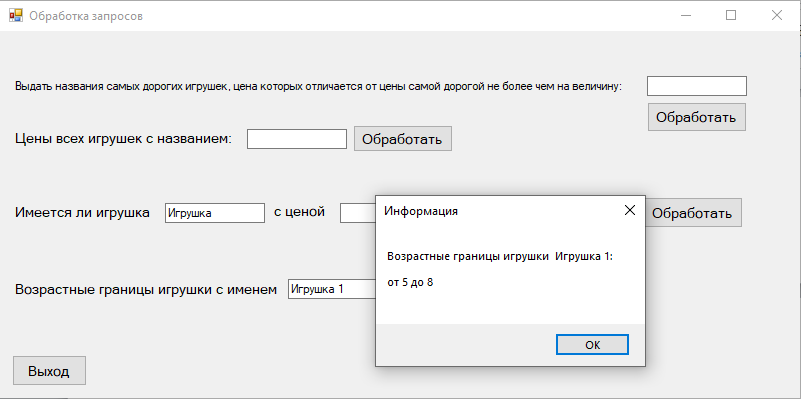


Рисунок 31 – Результат теста 23

Результат тестирования (тест 24), когда пользователь ведёт работу с 4-ым запросом, но оставляет поле для ввода данных пустым:

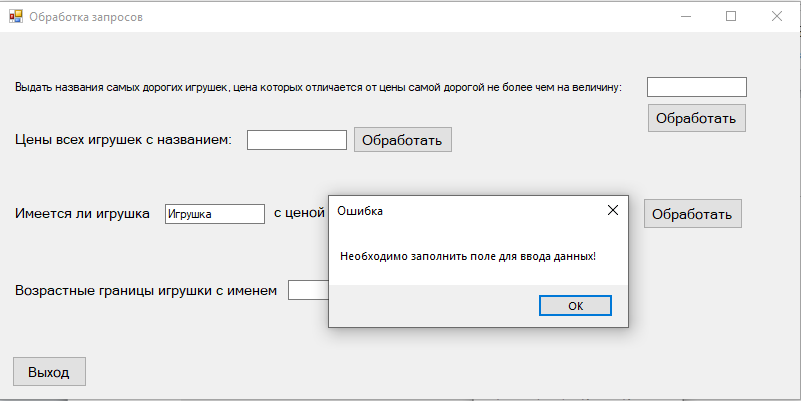


Рисунок 32 – Результат теста 24

Результат тестирования (тест 25), когда пользователь вводит название игрушки, которой в базе данных нет:

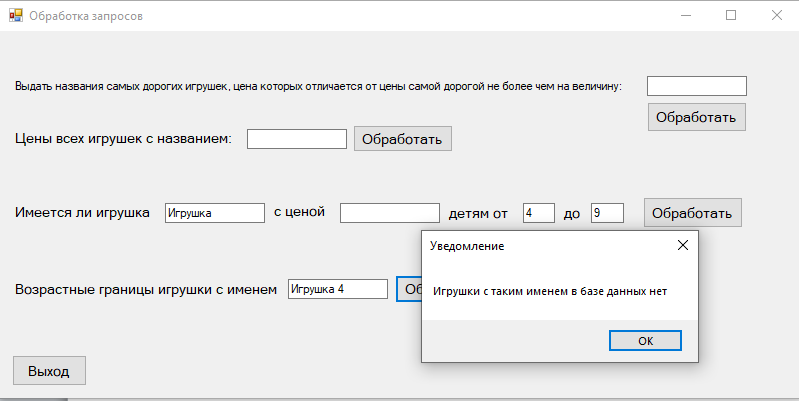


Рисунок 33 – Результат теста 25

Заключение

C# — простой, современный, объектно–ориентированный язык программирования. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java.

**В результате выполнения курсового проекта было разработано Windows–приложение, которое работает с текстовым файлом с информацией о игрушках.** Способы выполнения действий – согласно индивидуальному заданию. **В приложении можно создавать файлы, добавлять новые записи в файлы, удалять записи из файлов.** В приложение реализован просмотр файла **и выполнение четырёх запросов – согласно индивидуальному заданию.**

Приложение было протестировано на самые разнообразные ошибки, при разработке все они были исправлены.

Таким образом, цель работы достигнута, поставленные задачи выполнены. Полученные знания и навыки в создании оконного приложения на языке C# пригодятся в будущем для совершенствования в профессиональной сфере.

Список использованных источников

1. Сосинская С.С. «Обработка файлов»: Методические указания по выполнению курсового проекта» Электронный каталог кафедры.
2. Гуннерсон Э. Введение в C#. СПб: Питер, 2001.
3. Вагнер, Билл С# Эффективное программирование / Билл Вагнер. – М.: ЛОРИ, 2013. – 320 c.
4. Ишкова, Э. А. Самоучитель С#. Начала программирования / Э.А. Ишкова. – М.: Наука и техника, 2013. – 496 c.
5. Троелсен, Эндрю Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. – М.: Вильямс, 2015. – 486 c.
6. Биллинг В.А. Основы программирования на C#. – Т.: Интернет–университет информационных технологий, Бином, 2012. – 488 с.
7. Евдокимов П.В. C# на примерах. – М.: Наука и техника,2016. – 304 с.
8. Нейгел К., Ивьен Б., Глинн Д., Уотсон К., Скиннер М. C# 4.0 и платформа .NET 4, 2011.