|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

**КУРСОВАЯ РАБОТА**

по дисциплине «Объектно-ориентрованное программирование»

**Тема курсовой работы:** Проектирование и реализация программной системы сбора сведений о писателях

и их литературных произведениях с использованием объектного подхода

**Студент группы** ИКБО-07-18 , Петров Анатолий Валерьевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель курсовой работы** ассистент Калашникова Е.С

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена к защите «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Допущен к защите «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Утверждаю

Заведующий кафедрой МОСИТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Головин С.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2019 г.

**ЗАДАНИЕ**

**на выполнение курсовой работы по дисциплине**

«Объектно-ориентрованное программирование»

Студент Петров Анатолий Валерьевич Группа ИКБО-07-18

**Тема работы:** Проектирование и реализация программной системы сбора сведений о писателях и их литературных произведениях с использованием объектного подхода

**Исходные данные:** задание на курсовую работу,тема №1, вариант №31. 1.Разработать приложение, позволяющее собирать и накапливать сведения о писателях и их литературных произведениях. Структура приложения обязательно должна включать следующие классы: писатель (автор), произведение, жанр,страна (где проживал автор), язык (на котором написано или было переведено

произведение), издание произведения (тираж), издательство и др.

**Перечень вопросов, подлежащих разработке, и обязательного графического материала:**

Моделирование объектной системы с использованием обозначений языка моделирования UML

Реализация программной системы средствами языка С++

Отчет по курсовой работе в виде расчетно-пояснительной записки .

**Срок представления к защите курсовой работы:** до «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

**Задание на курсовую работу выдал** \_\_\_\_\_\_\_\_\_\_\_\_ (Калашникова Е.С.)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

**Задание на курсовую работу получил**  \_\_\_\_\_\_\_\_\_\_\_\_ (Петров А.В.)

Оглавление

1.Введение 4

2.Теоритическое введение 6

2.1 Основные понятия объектно-ориентированного программирования 6

2.2 UML в объектно-ориентированном программировании 7

2.3 Подходы к сбору и анализу требований к ПО 9

2.4 Методы создания интерфейса 10

3.Проектная часть 11

3.1 Разработка спецификаций 11

3.2 Проектирование программного обеспечения 12

4.Экспериментальная часть 14

4.1 Тестирование 11

5.Заключение 17

6.Список литературы 18

7.Приложение 19

# **Введение**

Существует определенный стиль программирования, основанный на использовании таких понятий, как объект[[1]](#footnote-1) и класс[[2]](#footnote-2), который называется объектно-ориентированным. На данный момент именно этот подход является основополагающим в современном программировании, вне зависимости от языка написания кода, исходя из чего, данное направление темы курсовой работы особенно актуально.

Целью курсовой работы является получение навыка проектирования и реализации программной системы с применением объектно-ориентированного подхода.

Для достижения данной цели были поставлены следующие задачи:

1. Выявить особенности объектно-ориентированного программирования;
2. Определить основные понятия: объект, класс, инкапсуляция, наследование, полиморфизм;
3. Разработать этапы построения модели;
4. Определить список функций программы;
5. Выполнить построение концептуальной модели предметной области и представить диаграмму классов и отношений между ними;
6. Определить атрибуты и методы класса;
7. Составить текстовые спецификации интерфейса классов;
8. Спроектировать методы классов;
9. Создать диаграмму объектов;
10. Реализовать программу на основе проектных решений;
11. Обеспечить обработку исключительных ситуаций;
12. Разработать графический пользовательский интерфейс;
13. Выполнить тестирование программы и проверить корректность выполнения команд;
14. Составить отчет по курсовой работе;

Объектом курсовой работы является объектно-ориентированная система сбора сведений о писателях и их литературных произведениях с использованием объектного подхода. Были использованы такие методы исследования как: системно-информационный анализ, моделирование, индукция, а также изучение и обобщение.

Сама курсовая работа состоит из введения, теоретической части, практической части, экспериментальной части и заключения. В теоретической части были описаны ключевые особенности объектно-ориентированного программирования, в практической части была разработана система сбора сведений, в экспериментальной части было проведено тестирование программы.

# **2. Теоретическое введение**

## 2.1 Основные понятия объектно-ориентированного программирования

Объектно-ориентированное программирование, прежде всего, возникло для более облегченной и логически структурированной работы программиста. В программном коде реализация объектов осуществляется при помощи классов и структур. Говоря об объектно-ориентированном программировании, следует отметить, что, в отличии от процедурного программирования, программа представляет собой не целый последовательный набор данных, а некоторое количество объектов связанных каким-либо образом или взаимодействующих между собой. Исходя из этого, можно сказать, что каждый объект имеет свои определенные данные, набор команд, и логику выполнения, поставленных перед этим объектом, конкретных задач. Объекты, которые можно объединить по определенному свойству, образуют собственные классы, классы определяют методы и свойства объектов.

Одним из основных понятий объектно-ориентированного программирования является инкапсуляция. Оно и характеризует объединение методов и самих данных в единые объекты.

Наследование – еще одно, и не менее важное понятие объектно-ориентированного программирования. Именно эта способность связывает инкапсулированные классы между собой. Наследование, с логической точки зрения, можно рассматривать как пример индукции/дедукции.

Полиморфизм, а точнее полиморфизм подтипов, также является основной концепцией объектно-ориентированного программирования и он непосредственно связан с инкапсуляцией. Данная способность характеризует возможность объектов, имеющих одинаковую спецификацию, иметь различную реализацию. Таким образом, классы потомки, несмотря на отличие в реализации методов, можно обрабатывать как однотипный набор элементов.

## 2.2 UML[[3]](#footnote-3) в объектно-ориентированном программировании

Как уже было сказано ранее, программа в ООП состоит из некоторого количества подпрограмм. Разделение сложной и объемной программы на упрощенные и менее объемные компоненты называется декомпозицией.

Для абстрактного графического представления объектов моделирования системы используется язык UML. В свою очередь, диаграммы UML разделяются на два вида: структурные диаграммы, в которые входят диаграммы классов, компонентов и объектов; и диаграммы поведения, основными типами которого являются диаграммы последовательности и диаграммы взаимодействия. Подобное вспомогательное представление программы является весьма удобным способом структуризации и более точного определения взаимосвязей между компонентами программы.

Диаграмма классов демонстрирует классы, а также их атрибуты и взаимосвязи, описывает структуру какой-либо системы. Существует несколько вариантов взаимосвязи объектов: если какой-либо объект, создающий другой объект, определяет время существования зависимого от него объекта, то подобная связь будет называться композицией, если же один из объектов связан с другим по средствам указателя, то такое взаимодействие называется агрегацией, в этом случае, время существования существования классов не зависят друг от друга, если контейнер был удален, его содержимое остается. Примеры других взаимосвязей, которые могут использоваться при построении графических моделей системы, представлены в таблице 1.1.

Таблица 1 – Графическое представление взаимосвязей

|  |  |
| --- | --- |
| Наименование связи | Изображение |
| наследование | Macintosh HD:Users:macbook:Desktop:Снимок экрана 2018-04-22 в 20.32.44.png |
| агрегация | Macintosh HD:Users:macbook:Desktop:Снимок экрана 2018-04-22 в 20.33.48.png |
| композиция | Macintosh HD:Users:macbook:Desktop:Снимок экрана 2018-04-22 в 20.34.47.png |
| привязка данных | Macintosh HD:Users:macbook:Desktop:Снимок экрана 2018-04-22 в 20.35.44.png |
| привязка данных | Macintosh HD:Users:macbook:Desktop:Снимок экрана 2018-04-22 в 20.36.25.png |
| зависимость | Macintosh HD:Users:macbook:Desktop:Снимок экрана 2018-04-22 в 20.36.56.png |

## 2.3 Подходы к сбору и анализу требований к ПО

Формирование и анализ требований – это сложный процесс, который проходит через ряд этапов:

* Анализ предметной области. Необходимо изучить предметную область, где будет использоваться данная система.
* Сбор требований. Данный этап подразумевает общение с людьми, которые формируют требования. Так же продолжается анализ предметной области.
* Классификация требований. На этом этапе все требования формируются ы логически связанные группы.
* Назначение приоритетов. Этот этап позволяет определить, какие из требований являются наиболее важными.
* Проверка требований. Проверяется полнота, последовательность, непротиворечивость требований к реализации ПО.

## 2.4 Методы создания интерфейса

Интерфейс – это важная составляющая любой программной системы. Для того, чтобы пользователю было удобно пользоваться программой мы создаем форму, на которой создаем кнопки, поля для ввода, название полей и т. д. Для того, чтобы создать пользовательский интерфейс следует придерживаться некоторых правил, таких как:

* Должен быть удобен для чтения пользователя.
* При выполнении действий пользователем, компьютер должен выполнять правильные действия

Также существуют различные виды пользовательского интерфейса такие как:

* Визуальный
* Текстовый
* Графический
* Жестовый
* Голосовой
* Материальный (осязательный)

Таким образом, разработчик, опираясь на условие задачи разрабатывает наиболее удобный и правильный интерфейс.

# **3. Проектная часть**

## 3.1 Разработка спецификаций

Цель данной курсовой работы состоит в том, чтобы разработать систему сбора сведений об авторах и их книгах.

В этой системе мы разрабатываем программу, в которой пользователь может просматривать сведения о писателях, книгах и т.д. из базы данных, которая представляет собой таблицу, а также добавлять новые сведения, которые будут проверяться на корректный ввод и выводиться в таблицу. Визуальные функции показаны на рисунке 1.

Спецификация – точное формализованное описание функций и организаций разрабатываемого ПО.

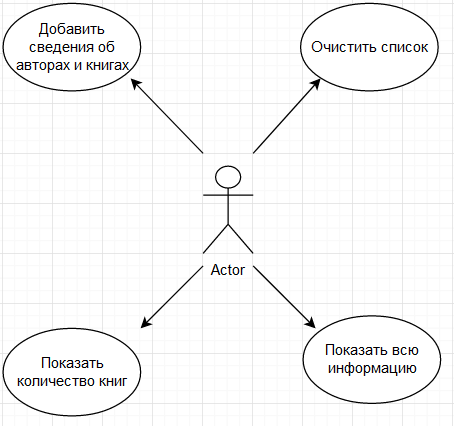


Рисунок 1 – Визуализация функций пользователя

Концептуальная модель предметной области представлена на рисунке 2.

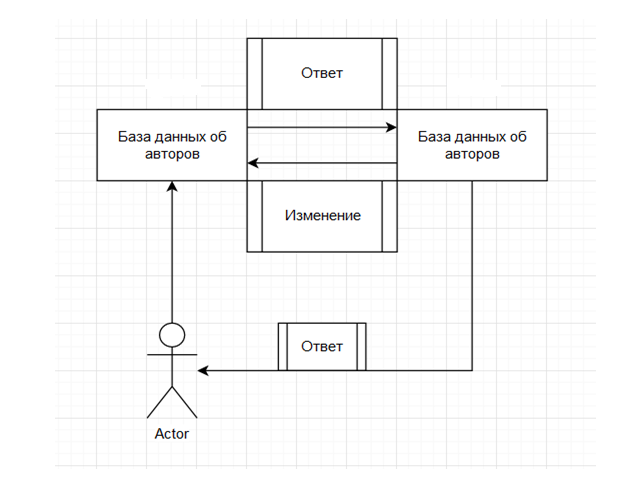


Рисунок 2 – Концептуальная модель предметной области.

## 3.2 Проектирование программного обеспечения

В итоге получается, что нам потребуется 9 классов. В MyForm происходят основные действия, а именно: реализация графического интерфейса (кнопки, таблица и прочее), обращение к текстовому файлу(базе данных) и выполнения методов базы данных. Поэтому я подключил к MyForm заголовки прочих классов, которые нужны для заполнения таблицы. Помимо этого, остальные классы проверяют введенные данные на корректность перед тем, как поместить в таблицу (см. рисунок 3).

Кроме того, было использовано наследование. Класс Books является родителем классов Style и Edition, а также от класса Author есть производный класс Country. Это было сделано для удобства и соблюдения логики.

Класс Helptxt помогает базе данных считывать информацию из таблицы и заполнять её в начале работы.

Перед завершением работы программа сохраняет все изменения в текстовый файл.

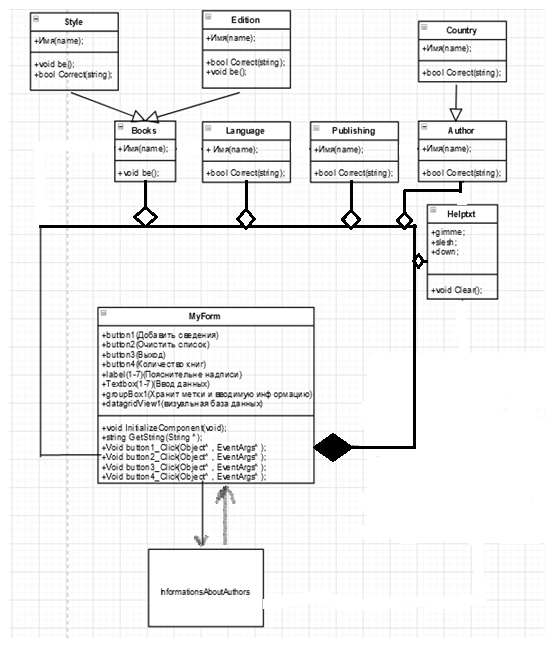


Рисунок 3 – Диаграмма классов

## 4. Экспериментальная часть

## 4.1 Тестирование

При запуске программы появляется таблица со сведениями (если они были в текстовом файле). В таблице видно количество сведений, авторы, литературные произведения, жанры, страны, языки, тиражи и издательства (см. рисунок 4).

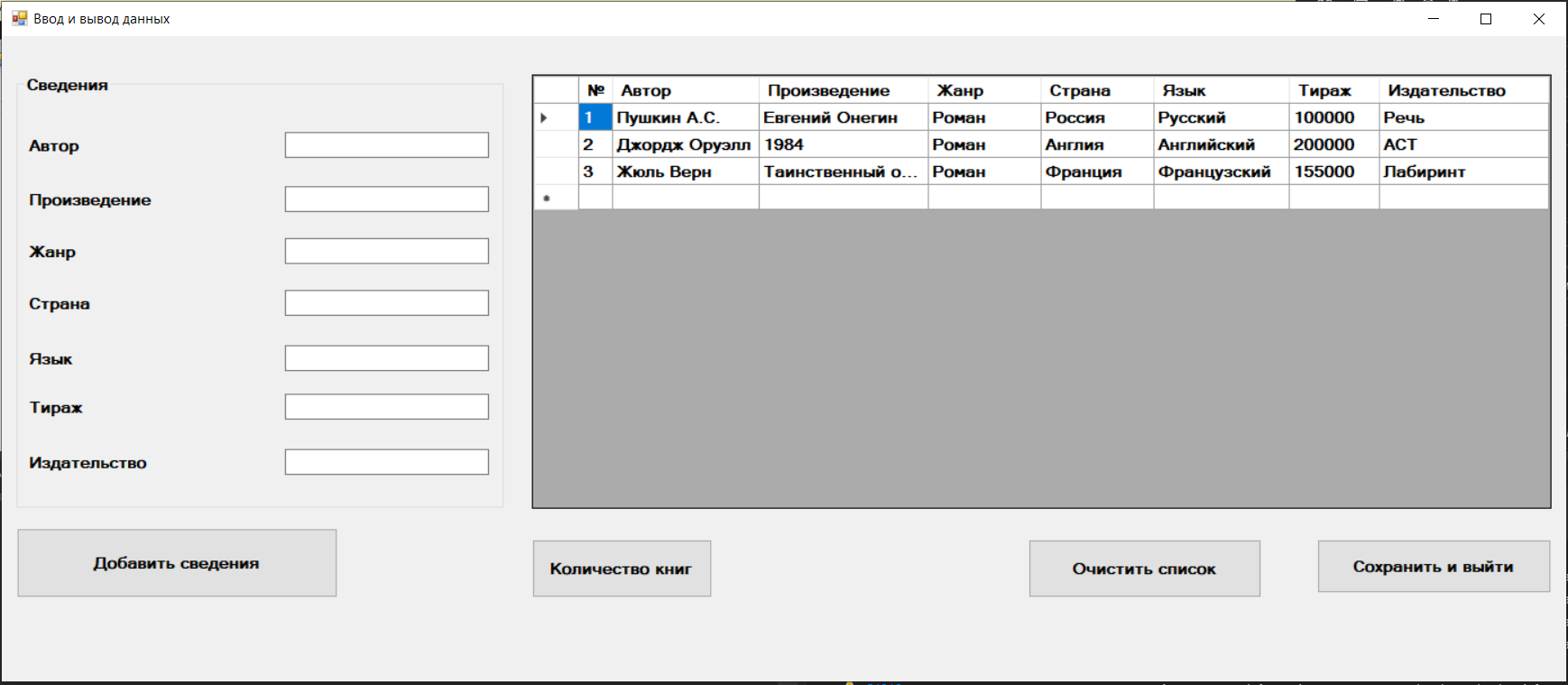


Рисунок 4 – База данных

При нажатии на кнопку количество книг появляется сообщение с соответствующим результатом (см. рисунок 5).

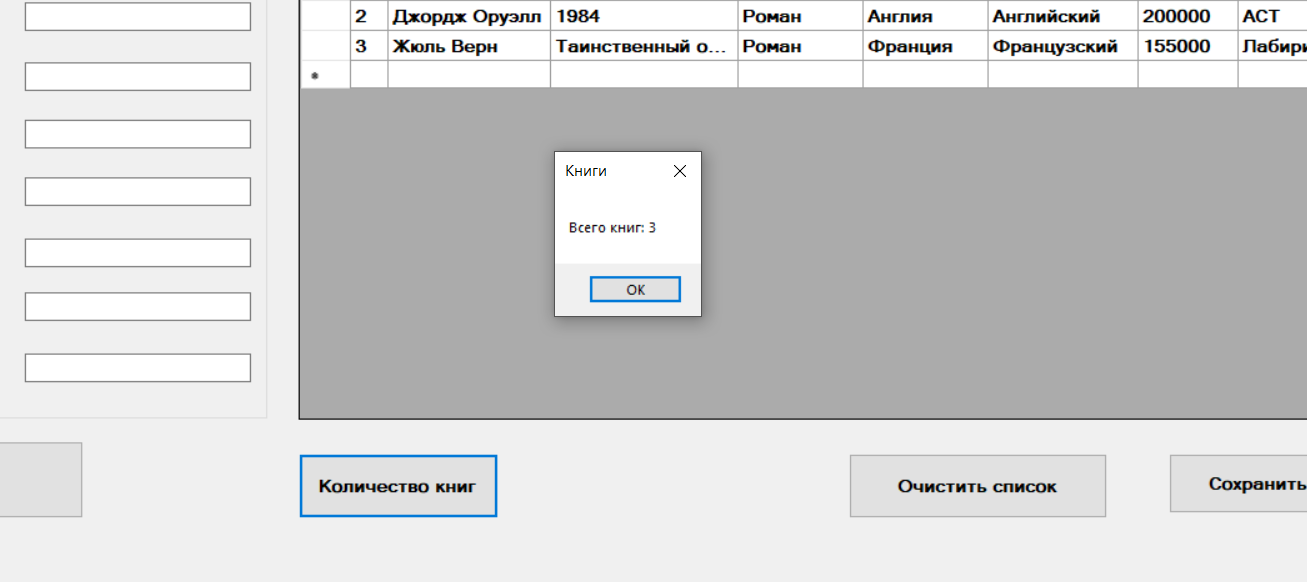


Рисунок 5 – Количество книг

При корректном вводе значений в поля, сведение попадает в таблицу и появляется уведомление (см. рисунок 6).

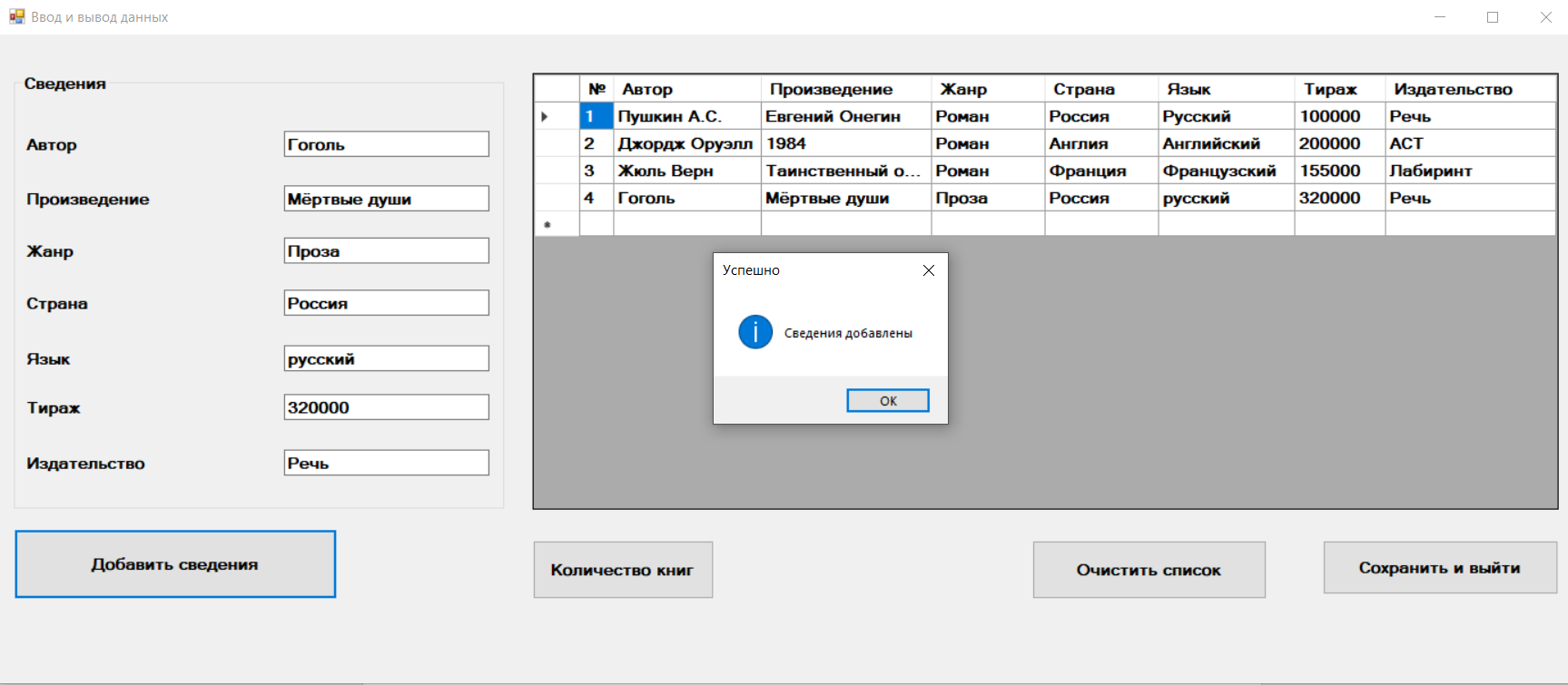


Рисунок 6 – Добавление сведений

«Очистить список» также работает правильно (см. рисунок 7)

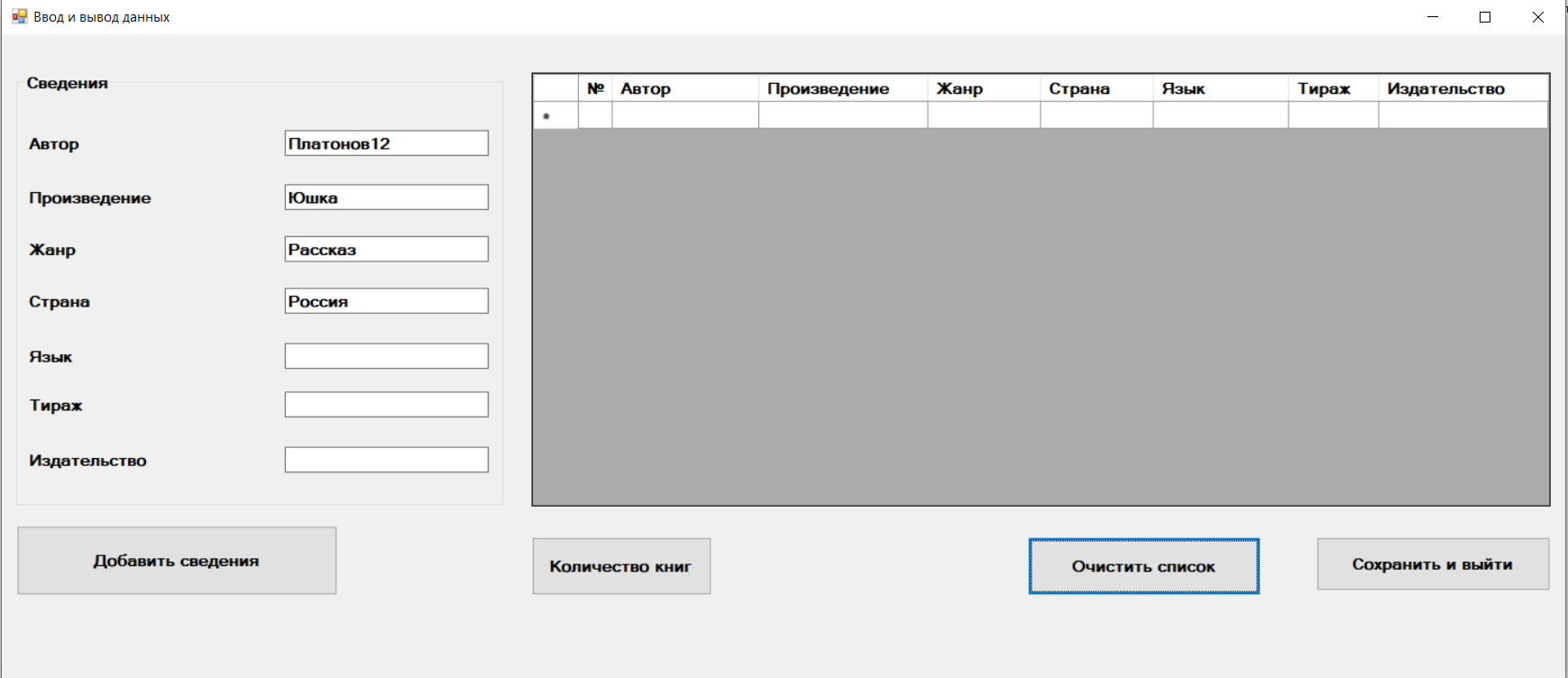


Рисунок 7 – Очищение список

При завершении работы и сохранении результатов программа работает без ошибок.

Предусмотренные ошибки: Заполнены не все поля (см. рисунок 8), введены не корректные данные (см. рисунок 9), вывод сообщение о неправильном открытии сервера.

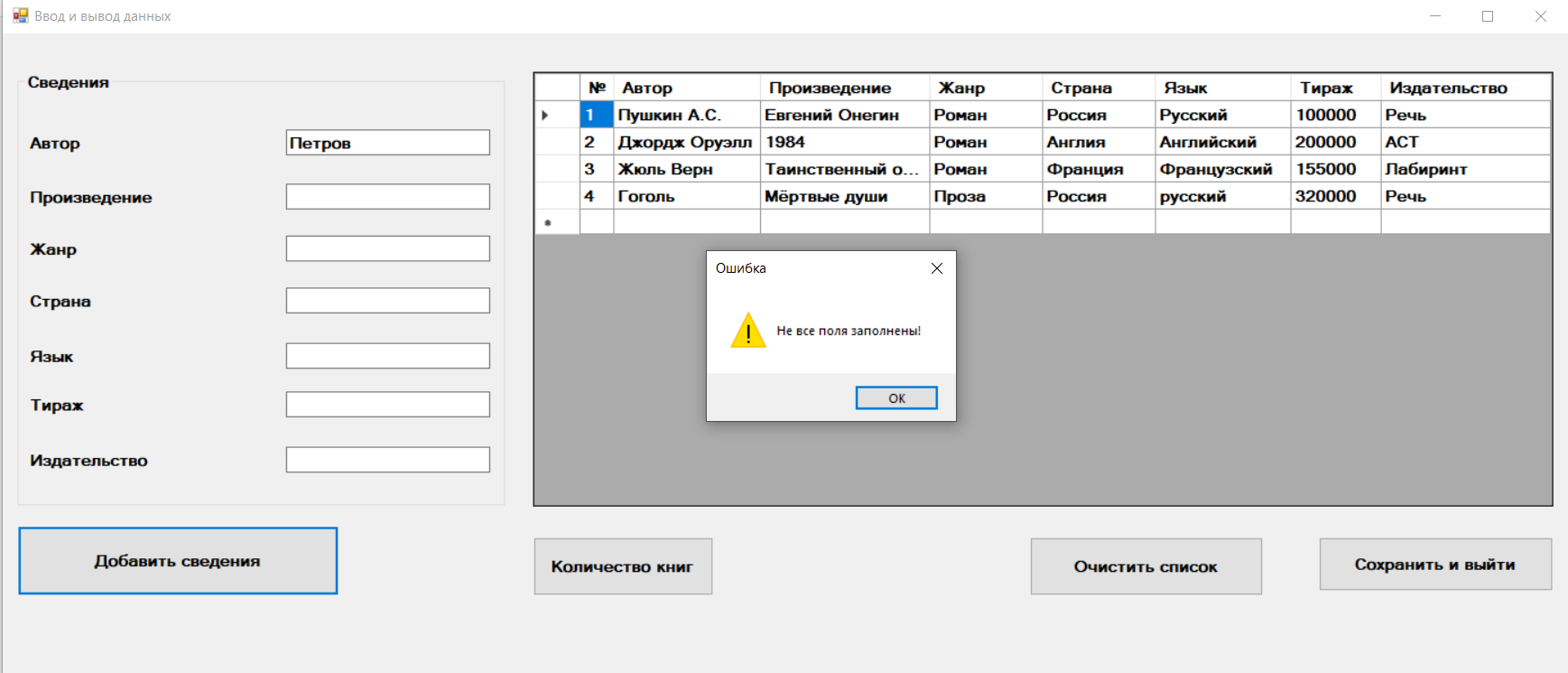


Рисунок 8 – Заполнены не все поля

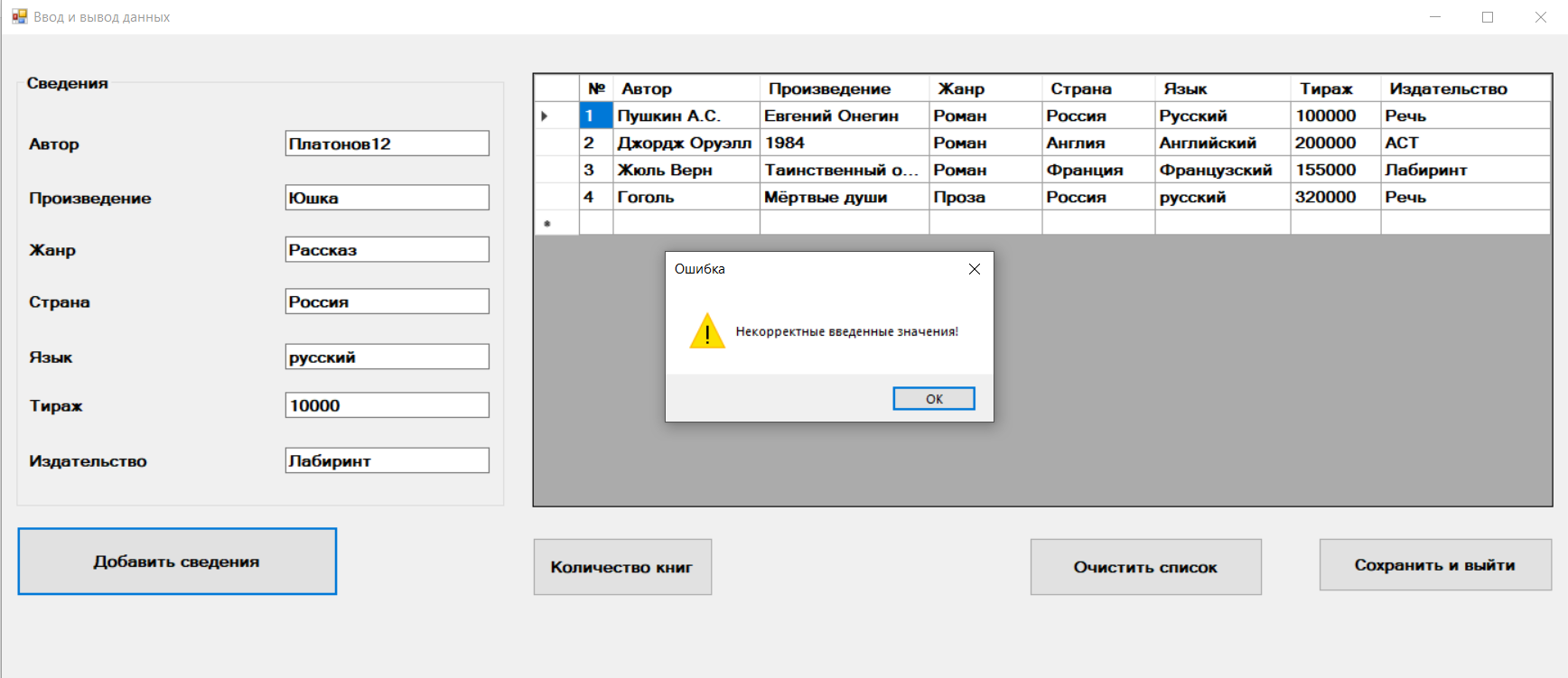


Рисунок 9 – Введены не корректные данные

# 

# **5. Заключение**

Ключевыми понятиями объектно-ориентированного программирования являются инкапсуляция – объединение логически связанных данных и методов в единые структуры и классы; наследование связывает классы между собой путем принятия новым объектом некоторых свойств объекта – родителя.

Для абстрактного представления объектов используются диаграммы UML. Они помогают, как разработчику, так и пользователю более детально определить связи и взаимодействие между различными частями системы.

Пользовательский интерфейс – не менее важная часть каждого программного обеспечения, он должен быть удобен и прост в использовании, с его помощью происходит прием данных и запросов, и дальнейшее функционирование программы непосредственно с ним связано.

В данной курсовой работе были рассмотрены основные понятия и ключевые особенности объектно-ориентированного программирования, получены навыки проектирования и реализации программного обеспечения. Программа совмещает в себе работу с классами, файлами и пользовательский интерфейс; предусмотрена защита от некорректного ввода данных и уведомление пользователей об ошибках.

Реализация программной системы сбора сведений о писателях и их литературных произведениях прошло этапы тестирование и успешно работает в соответствии с заданием. Были выполнены все поставленные задачи и достигнута цель работы.

Программа предоставлена в приложении.

# **6. Список литературы**

1. Страуструп, Бьярне.

С83 Программирование: принципы и практика с использованием С++,

2-е изд. : Пер. с англ. - М . : ООО "И . Д. Вильяме", 20 1 6. - 1 328 с. :

ил . - Парал. тит. англ.ISBN 978-5-8459- 1 949-6 (рус)

1. Пахомов Б. И.
   1. П12 C/C++ и MS Visual C++ 2012 для начинающих. — СПб.: БХВ-Петербург, 2013. — 512 с.: ил. ISBN 978-5-9775-0881-0
2. G. Mori, F. Paterno, C. Santoro. CTTE: Support for Developing and Analyzing Task Models for Interactive Systems Design // IEEE Trans. Software Eng. 2002, vol. 28, N8, P. 797-813.
3. Разработка ПО. Документация по Visual Studio: [Электронный ресурс]

https://www.visualstudio.com/ru/

(Дата обращения: 2.05.2019).

# **7. Приложение**

**MyForm.h**

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include "Author.h"

#include "Helptxt.h"

#include "Books\_and\_other\_informations.h"

namespace Information {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace std;

int k = 0;

Helptxt clear;

Author author;

Country From;

Style style;

Language language;

Edition edition;

Publishing publishing;

Books book;

/// <summary>

/// Сводка для MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

ifstream fin;

fin.open("InformationsAboutAuthors.txt");

if (!fin.is\_open())

{

ofstream fin("InformationsAboutAuthors.txt");

}

else

{

char ch, chfull[100];

for (int l1 = 0; l1 < 100; l1++)

{

chfull[l1] = 0;

}

int l = 0, i = 0;

Helptxt \*help = new Helptxt;

while (!fin.eof())

{

dataGridView1->Rows->Add();

while (fin.get(ch))

{

if (ch == '\n')

{

chfull[l] = 0;

break;

}

else

{

if (ch != '/')

{

chfull[l] = ch;

l++;

}

else

{

help->gimme = chfull;

i++;

dataGridView1->Rows[k]->Cells[i]->Value = gcnew System::String(help->gimme.c\_str());

help->gimme = "";

for (int l1 = 0; l1 < 100; l1++)

{

chfull[l1] = 0;

}

l = 0;

}

}

}

if (i == 7)

{

dataGridView1->Rows[k]->Cells[0]->Value = k + 1;

k++;

l = 0;

i = 0;

}

else

{

for (int j = 0; j < i+1; j++)

{

dataGridView1->Rows[k]->Cells[j]->Value = "";

}

MessageBox::Show("Ошибка чтения файла!", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Warning);

break;

}

}

}

fin.close();

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

protected:

private: System::Windows::Forms::DataGridView^ dataGridView1;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::Label^ FIO;

private: System::Windows::Forms::Label^ Books;

private: System::Windows::Forms::Label^ Style;

private: System::Windows::Forms::Label^ Country;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::TextBox^ textBox3;

private: System::Windows::Forms::TextBox^ textBox4;

private: System::Windows::Forms::GroupBox^ groupBox1;

private: System::Windows::Forms::TextBox^ textBox7;

private: System::Windows::Forms::TextBox^ textBox6;

private: System::Windows::Forms::TextBox^ textBox5;

private: System::Windows::Forms::Label^ Publishing;

private: System::Windows::Forms::Label^ Edition;

private: System::Windows::Forms::Label^ Language;

private: System::Windows::Forms::Button^ button4;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column1;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column2;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column3;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column4;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column5;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column6;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column7;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column8;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::Button^ button3;

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());

this->Column1 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column2 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column3 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column4 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column5 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column6 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column7 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column8 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->FIO = (gcnew System::Windows::Forms::Label());

this->Books = (gcnew System::Windows::Forms::Label());

this->Style = (gcnew System::Windows::Forms::Label());

this->Country = (gcnew System::Windows::Forms::Label());

this->button1 = (gcnew System::Windows::Forms::Button());

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->textBox3 = (gcnew System::Windows::Forms::TextBox());

this->textBox4 = (gcnew System::Windows::Forms::TextBox());

this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());

this->textBox7 = (gcnew System::Windows::Forms::TextBox());

this->textBox6 = (gcnew System::Windows::Forms::TextBox());

this->textBox5 = (gcnew System::Windows::Forms::TextBox());

this->Publishing = (gcnew System::Windows::Forms::Label());

this->Edition = (gcnew System::Windows::Forms::Label());

this->Language = (gcnew System::Windows::Forms::Label());

this->button4 = (gcnew System::Windows::Forms::Button());

this->button2 = (gcnew System::Windows::Forms::Button());

this->button3 = (gcnew System::Windows::Forms::Button());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->BeginInit();

this->groupBox1->SuspendLayout();

this->SuspendLayout();

//

// dataGridView1

//

this->dataGridView1->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView1->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(8) {

this->Column1,

this->Column2, this->Column3, this->Column4, this->Column5, this->Column6, this->Column7, this->Column8

});

this->dataGridView1->Location = System::Drawing::Point(470, 34);

this->dataGridView1->Margin = System::Windows::Forms::Padding(4);

this->dataGridView1->Name = L"dataGridView1";

this->dataGridView1->RowTemplate->Height = 24;

this->dataGridView1->Size = System::Drawing::Size(904, 385);

this->dataGridView1->TabIndex = 1;

//

// Column1

//

this->Column1->HeaderText = L"№";

this->Column1->Name = L"Column1";

this->Column1->ReadOnly = true;

this->Column1->Width = 30;

//

// Column2

//

this->Column2->HeaderText = L"Автор";

this->Column2->Name = L"Column2";

this->Column2->ReadOnly = true;

this->Column2->Width = 130;

//

// Column3

//

this->Column3->HeaderText = L"Произведение";

this->Column3->Name = L"Column3";

this->Column3->ReadOnly = true;

this->Column3->Width = 150;

//

// Column4

//

this->Column4->HeaderText = L"Жанр";

this->Column4->Name = L"Column4";

this->Column4->ReadOnly = true;

//

// Column5

//

this->Column5->HeaderText = L"Страна";

this->Column5->Name = L"Column5";

this->Column5->ReadOnly = true;

//

// Column6

//

this->Column6->HeaderText = L"Язык";

this->Column6->Name = L"Column6";

this->Column6->ReadOnly = true;

this->Column6->Width = 120;

//

// Column7

//

this->Column7->HeaderText = L"Тираж";

this->Column7->Name = L"Column7";

this->Column7->ReadOnly = true;

this->Column7->Width = 80;

//

// Column8

//

this->Column8->HeaderText = L"Издательство";

this->Column8->Name = L"Column8";

this->Column8->ReadOnly = true;

this->Column8->Width = 150;

//

// textBox1

//

this->textBox1->Location = System::Drawing::Point(238, 51);

this->textBox1->Margin = System::Windows::Forms::Padding(4);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(181, 27);

this->textBox1->TabIndex = 2;

//

// FIO

//

this->FIO->AutoSize = true;

this->FIO->Location = System::Drawing::Point(8, 54);

this->FIO->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->FIO->Name = L"FIO";

this->FIO->Size = System::Drawing::Size(66, 20);

this->FIO->TabIndex = 3;

this->FIO->Text = L"Автор";

//

// Books

//

this->Books->AutoSize = true;

this->Books->Location = System::Drawing::Point(8, 102);

this->Books->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->Books->Name = L"Books";

this->Books->Size = System::Drawing::Size(145, 20);

this->Books->TabIndex = 4;

this->Books->Text = L"Произведение";

//

// Style

//

this->Style->AutoSize = true;

this->Style->Location = System::Drawing::Point(8, 148);

this->Style->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->Style->Name = L"Style";

this->Style->Size = System::Drawing::Size(57, 20);

this->Style->TabIndex = 5;

this->Style->Text = L"Жанр";

//

// Country

//

this->Country->AutoSize = true;

this->Country->Location = System::Drawing::Point(8, 194);

this->Country->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->Country->Name = L"Country";

this->Country->Size = System::Drawing::Size(77, 20);

this->Country->TabIndex = 6;

this->Country->Text = L"Страна";

//

// button1

//

this->button1->Location = System::Drawing::Point(13, 436);

this->button1->Margin = System::Windows::Forms::Padding(4);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(285, 62);

this->button1->TabIndex = 7;

this->button1->Text = L"Добавить сведения";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// textBox2

//

this->textBox2->Location = System::Drawing::Point(238, 99);

this->textBox2->Margin = System::Windows::Forms::Padding(4);

this->textBox2->Name = L"textBox2";

this->textBox2->Size = System::Drawing::Size(181, 27);

this->textBox2->TabIndex = 10;

//

// textBox3

//

this->textBox3->Location = System::Drawing::Point(238, 145);

this->textBox3->Margin = System::Windows::Forms::Padding(4);

this->textBox3->Name = L"textBox3";

this->textBox3->Size = System::Drawing::Size(181, 27);

this->textBox3->TabIndex = 11;

//

// textBox4

//

this->textBox4->Location = System::Drawing::Point(238, 191);

this->textBox4->Margin = System::Windows::Forms::Padding(4);

this->textBox4->Name = L"textBox4";

this->textBox4->Size = System::Drawing::Size(181, 27);

this->textBox4->TabIndex = 12;

//

// groupBox1

//

this->groupBox1->Controls->Add(this->textBox7);

this->groupBox1->Controls->Add(this->textBox6);

this->groupBox1->Controls->Add(this->textBox5);

this->groupBox1->Controls->Add(this->Publishing);

this->groupBox1->Controls->Add(this->Edition);

this->groupBox1->Controls->Add(this->Language);

this->groupBox1->Controls->Add(this->textBox1);

this->groupBox1->Controls->Add(this->textBox4);

this->groupBox1->Controls->Add(this->FIO);

this->groupBox1->Controls->Add(this->textBox3);

this->groupBox1->Controls->Add(this->Books);

this->groupBox1->Controls->Add(this->textBox2);

this->groupBox1->Controls->Add(this->Style);

this->groupBox1->Controls->Add(this->Country);

this->groupBox1->Location = System::Drawing::Point(13, 34);

this->groupBox1->Margin = System::Windows::Forms::Padding(4);

this->groupBox1->Name = L"groupBox1";

this->groupBox1->Padding = System::Windows::Forms::Padding(4);

this->groupBox1->Size = System::Drawing::Size(432, 385);

this->groupBox1->TabIndex = 13;

this->groupBox1->TabStop = false;

this->groupBox1->Text = L"Сведения";

//

// textBox7

//

this->textBox7->Location = System::Drawing::Point(238, 332);

this->textBox7->Name = L"textBox7";

this->textBox7->Size = System::Drawing::Size(181, 27);

this->textBox7->TabIndex = 18;

//

// textBox6

//

this->textBox6->Location = System::Drawing::Point(238, 283);

this->textBox6->Name = L"textBox6";

this->textBox6->Size = System::Drawing::Size(181, 27);

this->textBox6->TabIndex = 17;

//

// textBox5

//

this->textBox5->Location = System::Drawing::Point(238, 240);

this->textBox5->Name = L"textBox5";

this->textBox5->Size = System::Drawing::Size(181, 27);

this->textBox5->TabIndex = 16;

//

// Publishing

//

this->Publishing->AutoSize = true;

this->Publishing->Location = System::Drawing::Point(8, 335);

this->Publishing->Name = L"Publishing";

this->Publishing->Size = System::Drawing::Size(142, 20);

this->Publishing->TabIndex = 15;

this->Publishing->Text = L"Издательство";

//

// Edition

//

this->Edition->AutoSize = true;

this->Edition->Location = System::Drawing::Point(8, 286);

this->Edition->Name = L"Edition";

this->Edition->Size = System::Drawing::Size(66, 20);

this->Edition->TabIndex = 14;

this->Edition->Text = L"Тираж";

//

// Language

//

this->Language->AutoSize = true;

this->Language->Location = System::Drawing::Point(8, 243);

this->Language->Name = L"Language";

this->Language->Size = System::Drawing::Size(56, 20);

this->Language->TabIndex = 13;

this->Language->Text = L"Язык";

//

// button4

//

this->button4->Location = System::Drawing::Point(910, 446);

this->button4->Name = L"button4";

this->button4->Size = System::Drawing::Size(207, 52);

this->button4->TabIndex = 15;

this->button4->Text = L"Очистить список";

this->button4->UseVisualStyleBackColor = true;

this->button4->Click += gcnew System::EventHandler(this, &MyForm::button4\_Click);

//

// button2

//

this->button2->Location = System::Drawing::Point(470, 446);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(160, 52);

this->button2->TabIndex = 17;

this->button2->Text = L"Количество книг";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2\_Click);

//

// button3

//

this->button3->Location = System::Drawing::Point(1166, 446);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(208, 48);

this->button3->TabIndex = 18;

this->button3->Text = L"Сохранить и выйти";

this->button3->UseVisualStyleBackColor = true;

this->button3->Click += gcnew System::EventHandler(this, &MyForm::button3\_Click);

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(11, 20);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(1387, 572);

this->Controls->Add(this->button3);

this->Controls->Add(this->button2);

this->Controls->Add(this->button4);

this->Controls->Add(this->groupBox1);

this->Controls->Add(this->button1);

this->Controls->Add(this->dataGridView1);

this->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->Margin = System::Windows::Forms::Padding(4);

this->Name = L"MyForm";

this->Text = L"Ввод и вывод данных";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->EndInit();

this->groupBox1->ResumeLayout(false);

this->groupBox1->PerformLayout();

this->ResumeLayout(false);

}

#pragma endregion

private: std::string GetString(String ^ s)

{

std::string os;

using namespace Runtime::InteropServices;

const char\* chars =

(const char\*)(Marshal::StringToHGlobalAnsi(s)).ToPointer();

os = chars;

Marshal::FreeHGlobal(IntPtr((void\*)chars));

return os;

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

if ((textBox1->Text == "") || (textBox2->Text == "") || (textBox3->Text == "") || (textBox4->Text == "") || (textBox5->Text == "")

|| (textBox6->Text == "") || (textBox7->Text == ""))

{

MessageBox::Show("Не все поля заполнены!", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

else

{

int t = 0;

author.name = GetString(textBox1->Text);

book.name = GetString(textBox2->Text);

book.be();

style.name = GetString(textBox3->Text);

From.name = GetString(textBox4->Text);

language.name = GetString(textBox5->Text);

edition.name = GetString(textBox6->Text);

publishing.name = GetString(textBox7->Text);

if (author.Correct(author.name))

{

if (style.Correct(style.name))

{

if (From.Correct(From.name))

{

if (language.Correct(language.name))

{

if (edition.Correct(edition.name))

{

if (publishing.Correct(publishing.name))

{

t = 1;

dataGridView1->Rows->Add();

dataGridView1->Rows[k]->Cells[0]->Value = k + 1;

dataGridView1->Rows[k]->Cells[1]->Value = textBox1->Text;

dataGridView1->Rows[k]->Cells[2]->Value = textBox2->Text;

dataGridView1->Rows[k]->Cells[3]->Value = textBox3->Text;

dataGridView1->Rows[k]->Cells[4]->Value = textBox4->Text;

dataGridView1->Rows[k]->Cells[5]->Value = textBox5->Text;

dataGridView1->Rows[k]->Cells[6]->Value = textBox6->Text;

dataGridView1->Rows[k]->Cells[7]->Value = textBox7->Text;

MessageBox::Show("Сведения добавлены", "Успешно", MessageBoxButtons::OK, MessageBoxIcon::Asterisk);

ofstream fout;

fstream file("InformationsAboutAuthors.txt");

fout.open("InformationsAboutAuthors.txt", ofstream ::app);

Helptxt sl;

if ( file.peek() != EOF)

fout << sl.down << author.name << sl.slesh << book.name << sl.slesh << style.name << sl.slesh

<< From.name << sl.slesh << language.name << sl.slesh << edition.name << sl.slesh

<< publishing.name << sl.slesh;

else

fout << author.name << sl.slesh << book.name << sl.slesh << style.name << sl.slesh

<< From.name << sl.slesh << language.name << sl.slesh << edition.name << sl.slesh

<< publishing.name << sl.slesh;

fout.close();

textBox1->Text = "";

textBox2->Text = "";

textBox3->Text = "";

textBox4->Text = "";

textBox5->Text = "";

textBox6->Text = "";

textBox7->Text = "";

}

}

}

}

}

}

if (t == 0)

{

MessageBox::Show("Некорректные введенные значения!", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Warning);

k--;

}

k++;

}

}

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e) {

dataGridView1->Rows->Clear();

clear.Clear();

k = 0;

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageBox::Show("Всего книг: " + k, "Книги", MessageBoxButtons::OK);

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

exit(0);

}

};

}

**MyForm.cpp**

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

void main(array<String^>^ args)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Information::MyForm form;//здесь должно быть имя вашего проекта

Application::Run(%form);

}

**Author.h**

#pragma once

#include <string>

class Author

{

public:

char s;

std::string name;

bool Correct(std::string s)

{

bool t = true;

for (int i = 0; i < s.length(); i++)

if (int(s[i]) > 47 && int(s[i]) < 59)//48-58 это ASCII-коды цифр

{

t = false;

break;

}

return t;

}

};

class Country : public Author

{

};

**Author.cpp**

#include "Author.h"

**Books\_and\_other\_informations.h**

#pragma once

#include <string>

class Books

{

public:

std::string name;

void be()

{

}

};

class Style : public Books

{

public:

bool Correct(std::string s)

{

bool t = true;

for (int i = 0; i < s.length(); i++)

if (int(s[i]) > 47 && int(s[i]) < 59)//48-58 это ASCII-коды цифр

{

t = false;

break;

}

return t;

}

};

class Language

{

public:

std::string name;

bool Correct(std::string s)

{

bool t = true;

for (int i = 0; i < s.length(); i++)

if (int(s[i]) > 47 && int(s[i]) < 59)//48-58 это ASCII-коды цифр

{

t = false;

break;

}

return t;

}

};

class Edition : public Books

{

public:

bool Correct(std::string s)

{

bool t = true;

for (int i = 0; i < s.length(); i++)

if (int(s[i]) < 48 || int(s[i]) > 58)//48-58 это ASCII-коды цифр

{

t = false;

break;

}

return t;

}

};

class Publishing

{

public:

std::string name;

bool Correct(std::string s)

{

bool t = true;

for (int i = 0; i < s.length(); i++)

if (int(s[i]) > 47 && int(s[i]) < 59)//48-58 это ASCII-коды цифр

{

t = false;

break;

}

return t;

}

};

**Books\_and\_other\_informations.cpp**

#include "Books\_and\_other\_informations.h"

**Helptxt.h**

#pragma once

#include <string>

#include <fstream>

class Helptxt {

public:

std::string gimme;

char slesh = '/';

char down = '\n';

void Clear()

{

remove("InformationsAboutAuthors.txt");

}

};

**Helptxt.cpp**

#include "Helptxt.h"

1. Объект – это место в памяти, имеющее тип, который определяет вид информации, которая может храниться в этом месте. [↑](#footnote-ref-1)
2. Класс – это пользовательский тип, состоящий из встроенных типов других пользовательских типов и функций. [↑](#footnote-ref-2)
3. Сокращение от английского Unified Modeling Language, что в переводе обозначает – унифицированный язык моделирования [↑](#footnote-ref-3)