**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ВТ**

отчет

**по учебной практике**

Тема: Графическое приложение (GUI)

|  |  |  |
| --- | --- | --- |
| Студент гр. 7307 |  | Торопов В. А. |
| Руководитель |  | Разумовский Г.В. |

**ЗАДАНИЕ**

**на учебную практику**

|  |  |  |
| --- | --- | --- |
| Студент Торопов В. А. | | |
| Группа 7307 | | |
| Тема практики: Графическое приложение (GUI) | | |
| Задание на практику:  Написать программу с пользовательским интерфейсом со свойствами ООП. Программа должна уметь сохранять и загружать данные. Пользователь должен иметь возможность взаимодействовать с интерфейсом. Задание: разработать приложение для работников библиотеки. В приложении должны храниться сведения об имеющихся в библиотеке книгах и о читателях библиотеки. Библиотекарю могут потребоваться следующие сведения: 1) Названия книг, закреплённых за читателем; 2) Фамилия автора и название книги с заданным шифром.  Библиотекарь может вносить следующие изменения:   * Запись нового читателя в библиотеку * Пополнение библиотеки * Списывание старой книги * Изменение шифра книги * Выдавать и списывать книги у читателя * Удаление читателя из библиотеки * Получать сведения о работе библиотеки (Количество читателей библиотеки, количество выданных книг за месяц, количество записавшихся читателей за месяц) | | |
| Сроки прохождения практики: 01.07.2019 – 12.07.2019 | | |
| Дата сдачи отчета: 10.07.2019 | | |
| Дата защиты отчета: 10.07.2019 | | |
|  | | |
| Студент |  | Торопов В. А. |
| Руководитель |  | Разумовский Г.В. |

**Аннотация**

Данный отчет содержит описание и примеры работы приложения, созданного при прохождении практики. В отчете описывается приложение для работников библиотеки.

**содержание**

[Введение 5](#_Toc13637254)

[Возможности приложения 6](#_Toc13637255)

[Особенности реализации интерфейса 6](#_Toc13637256)

[Особенности реализации внутренней структуры 6](#_Toc13637257)

[Описание пользовательского интерфейса 9](#_Toc13637258)

[Заключение 17](#_Toc13637259)

[Список использованных источников 18](#_Toc13637260)

[Приложения 19](#_Toc13637261)

# Введение

Цель учебной практики: закрепить навыки создания приложений с пользовательским интерфейсом с использованием языка любого программирования

Задачи:

* 1. Реализовать приложение удовлетворяющее условиям ООП с пользовательским интерфейсом.

# Возможности приложения

Приложение выполняет все функции данные в задании.

# Особенности реализации интерфейса

Приложение написано на языке C# с использованием Windows Form. В ходе данной работы использовались такие элементы, как button(Кнопки в программе), label(Надписи в программе), textbox(Текстовые поля для ввода и вывода информации в программе), comboBox(Вывод списка элементов с последующим поиском и выбором элементов в программе), pictureBox(Хранение и вывод изображения на главной странице в программе), ToolTip (Вывод подсказок при вводе данных в textbox), ErrorProvide(Обработка исключений и ошибок, а также вывод уведомлений о них пользователю). Также в программе использованы сами формы(Forms) для расположения элементов с целью взаимодействия с пользователем, а также для вывода информации пользователю.

# Особенности реализации внутренней структуры

Внутренняя структура приложения состоит из основного файла Program.cs(Содержащего main), а также из 3-х файлов классов: book.cs, reader.cs, library.cs. В основном файле находится код для запуска форм, а также класс переменной, хранящей в себе данные и методы для работы с библиотекой. Все данные хранятся в 3-х текстовых файлах: readers.txt, books.txt и statistic.txt. Подробное описание классов и переменных приведено ниже:

|  |  |
| --- | --- |
| Класс Program (Файл Library.cs) | |
| Методы | Назначение |
| Main() | Запуск программы и её визуальной части |

|  |  |
| --- | --- |
| Класс Library\_main (Файл Library.cs) | |
| Переменные | Назначение |
| Main\_library | Переменная, хранящая в себе данные библиотеки и методы работы с ней |
| Методы | Назначение |
| Update\_library() | Обновляет данные в файлах библиотеки согласно изменениям, произошедшим в программе. |

|  |  |
| --- | --- |
| Класс book (Файл book.cs) | |
| Переменные | Назначение |
| Name | Название книги |
| Autor | Автор книги |
| Publisher | Издатель книги |
| Year | Год издания книги |
| Code | Код книги |
| Available | Доступность книги(Можно ли её отдать читателю или она уже находится на руках) |
| Методы | Назначение |
| book() | Конструктор класса |
| Change\_code(string Code) | Меняет текущий Код книги на Code |
| Change\_Available() | Меняет доступность книги |
| Get\_name() | Предоставляет значение параметра Name |
| Get\_information() | Предоставляет информацию о книге(Значения параметров Name, Autor, Publishe, Year, Code) |
| Get\_Available() | Предоставляет значение параметра Available |

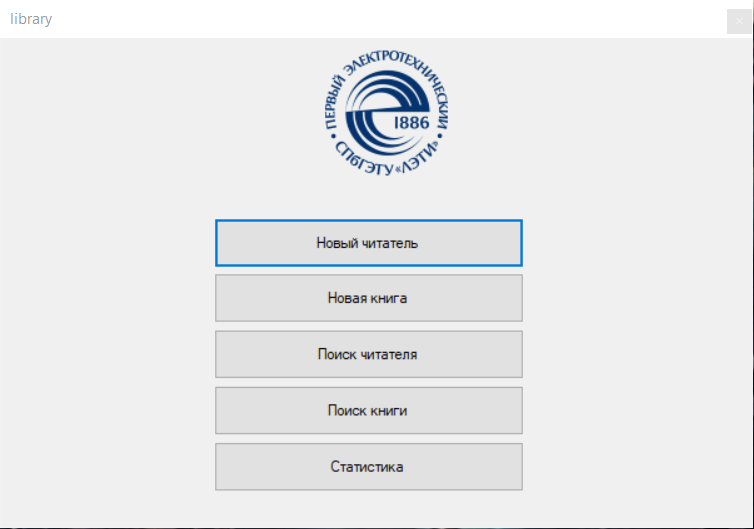
|  |  |
| --- | --- |
| Класс reader (Файл reader.cs) | |
| Переменные | Назначение |
| Name | Имя читателя |
| Surname | Фамилия читателя |
| Birthday | Дата рождения читателя |
| Registration | Дата регистрации читателя |
| Books\_reader | Список книг читателя на руках |
| Методы | Назначения |
| reader() | Конструктор класса |
| new\_reader\_book(book New\_book) | Добавляет книгу в список Books\_reader |
| Remove\_reader\_book(book Book) | Убирает книгу из списка Books\_reader |
| Get\_reader\_information() | Предоставляет информацию о читателе (Значения параметров Name, Surname, Birthday и Registration) |
| Get\_reader\_books() | Предоставляет данные списка Books\_reader |
| Get\_reader\_registration() | Предоставляет значение поля Registration |

|  |  |
| --- | --- |
| Класс library (Файл library.cs) | |
| Переменные | Назначение |
| readers | Список всех читателей |
| books | Список всех книг |
| Методы | Назначения |
| library() | Конструктор класса |
| New\_reader(string Name, string Surname, string Birthday) | Добавляет нового читателя в список readers |
| New\_book(string Autor, string Name, string Publisher, string Year, string Code) | Добавляет новую книгу в список books |
| Get\_reader(string Data) | Предоставляет информацию о читателе, данные которого совпадают с Data |
| Get\_books() | Предоставляет данные списка books |
| Get\_readers() | Предоставляет данные списка readers |
| New\_book\_reader(string Data\_reader, string Data\_book) | Добавляет книгу с данными Data\_book читателю с данными Data\_reader |
| Remove\_book\_reader(string Data\_reader, string Data\_book) | Удаляет книгу с данными Data\_book у читателя с данными Data\_reader |
| Remove\_reader(string Data\_reader) | Удаляет читателя с данными Data\_reader |
| Get\_reader\_book(string Data\_book) | Находит читателя, который содержит книгу с данными Data\_book. |
| New\_code\_book(String Data\_book, string New\_Code) | Меняет код книги с данными Data\_book с текущего на New\_code; |
| Delete\_book(string Data\_book, string Data\_reader) | Удаляет книгу с данными Data\_book из библиотеки |
| Proof\_Code(String Code) | Проверка существования кода в библиотеке |
| Add\_book\_month() | Добавляет к количеству книг выданных в этом месяце 1, если выдача произошла в текущем месяце и начинает счёт заново, если выдали книгу в новом. |
| Get\_count\_readers\_month() | Возвращает количество новых читателей за текущий месяц |
| Get\_count\_book\_month() | Возвращает количество выданных книг за текущий месяц |
| Get\_count\_readers() | Возвращает количество читателей в библиотеке |

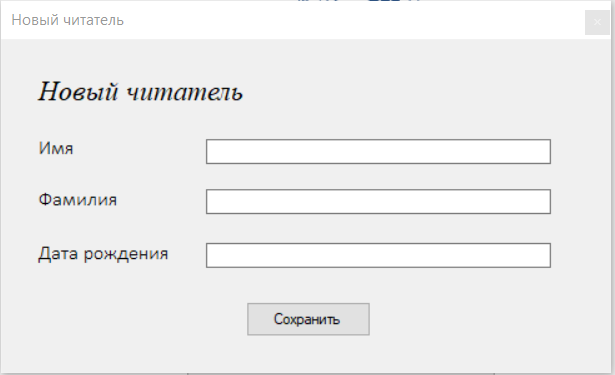
Приложение также способно сохранять и загружать данные из файла. Когда программа запускается, она загружает данные из файлов, а перед закрытием сохраняет все данные в файлы.

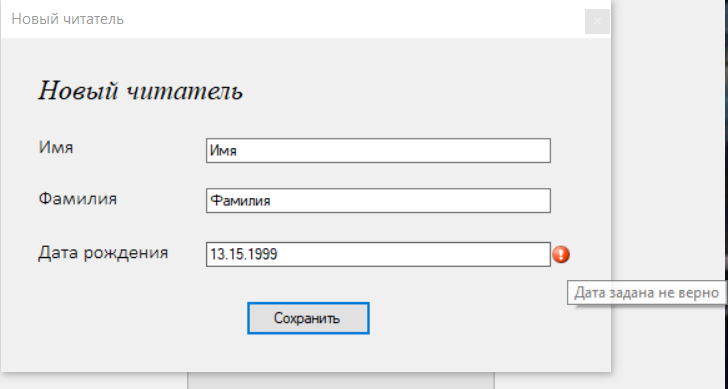
# Описание пользовательского интерфейса

При запуске программы появляется стартовое окно-меню, в котором можно выбрать интересующие нас функции.

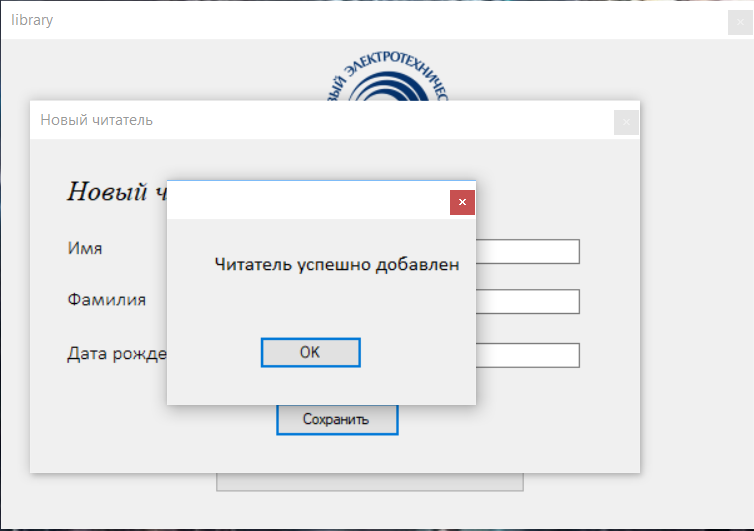
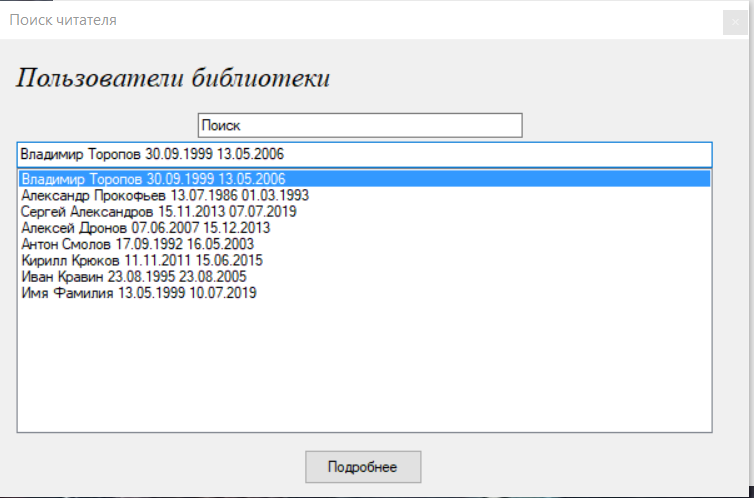
 При выборе определённой функции мы можем увидеть следующее диалоговое окно:

# 

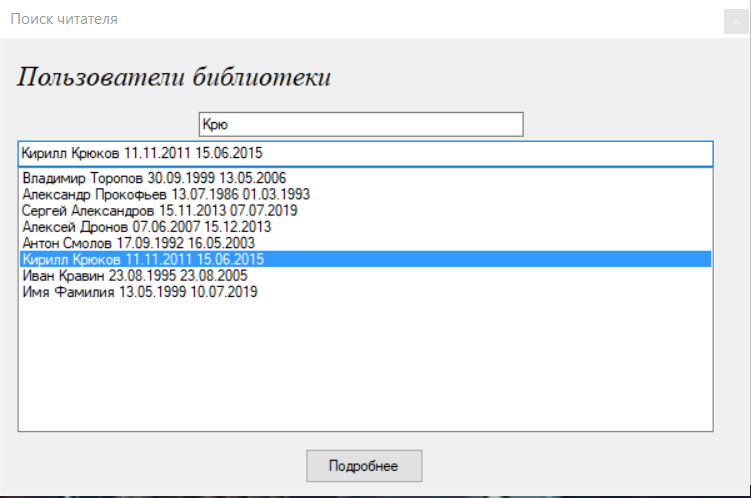
Введя некоторые данные попытаемся их сохранить, в следствии чего можем наткнуться на ошибку при вводе, о которой нас предупредит данное уведомление.

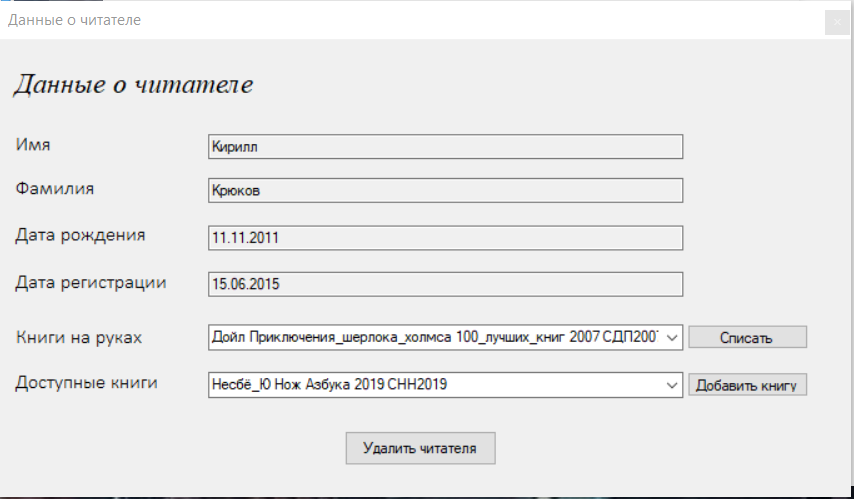


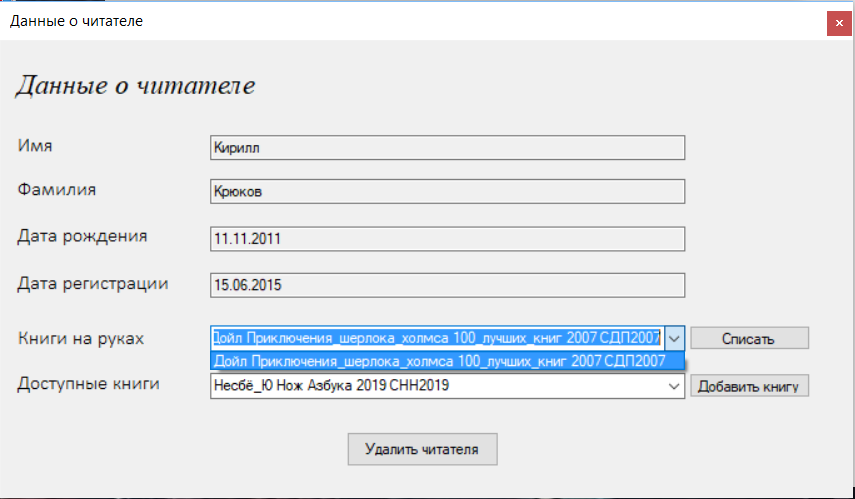
Введя корректные данные попытаемся вновь сохранить нового пользователя, на что получим обратную связь в виде диалогового окна о том, что добавление прошло успешно.

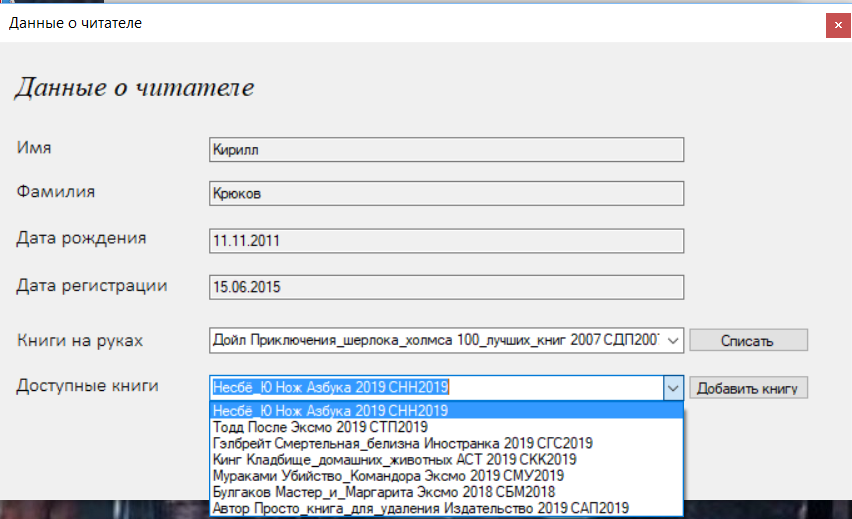
Вернувшись в меню выберем пункт «Поиск читателя» и перед нами возникнет следующее окно:

На нём мы видим Список всех читателей, полосу поиска, а также кнопку «Подробнее» которая позволяет нам ознакомится с читателем подробнее, а также добавить или убрать книгу из его профиля.

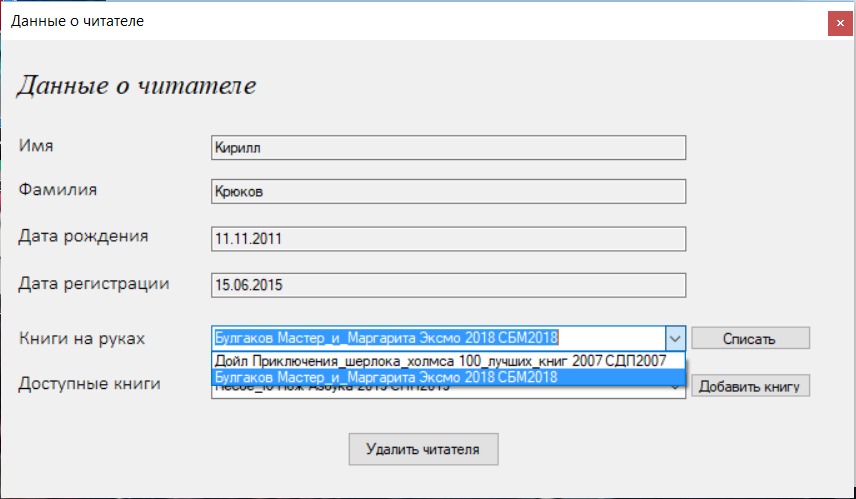


Найдя нужного нам читателя перемещаемся на его профиль и видим его данные, а также можем посмотреть книги на руках и доступные книги, которые мы можем выдать данному читателю.

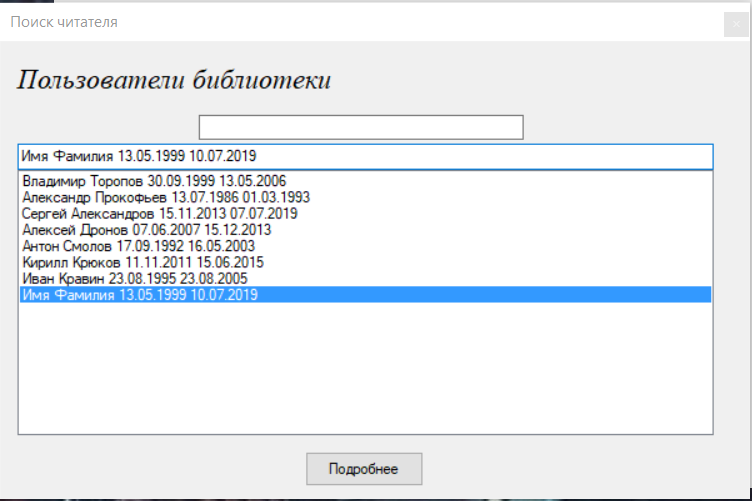


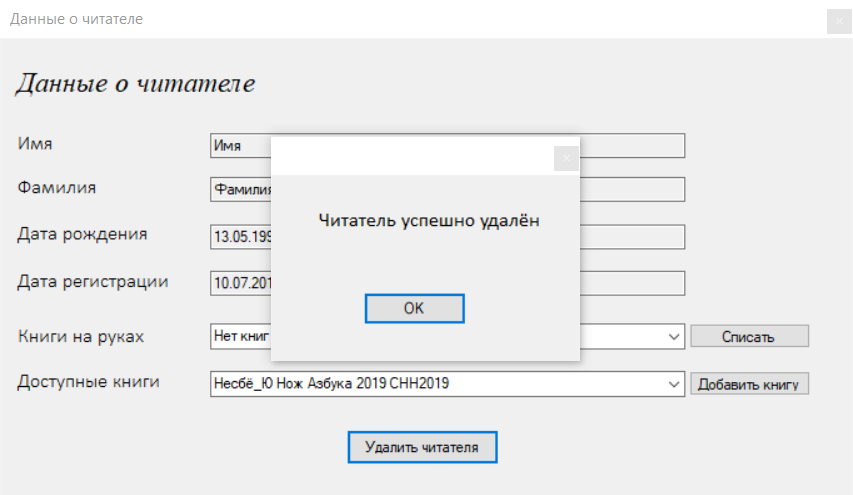


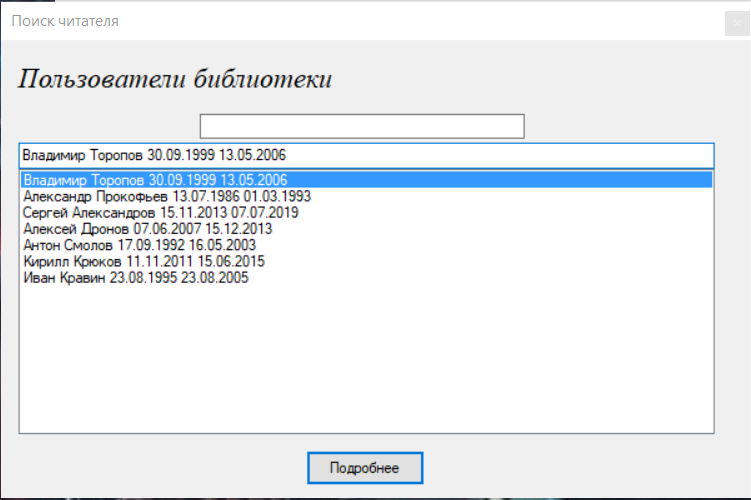
Данный читатель хочет взять себе домой книгу Булгакова «Мастер и маргарита», поэтому выбираем эту книгу в списке доступных и нажимаем кнопку «Добавить книгу» и книга появляется в списке «Книги на руках»



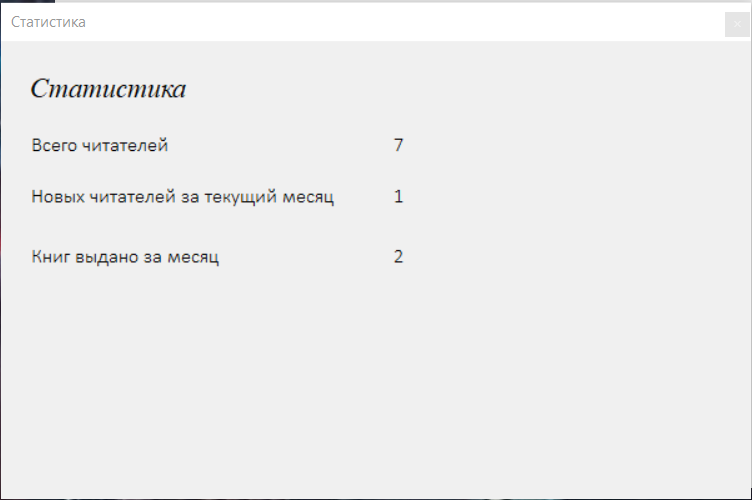
Вернувшись к списку читателей выбираем другого пользователя и также смотрим подробнее его профиль. Этого читателя необходимо удалить из нашей базы, поэтому мы нажимаем на кнопку «Удалить читателя» и получаем в ответ диалоговое окно, сообщающее нам об успехе проведённой операции. Нажав на кнопку «ОК» мы вернулись к списку пользователей и там действительно нет удалённого пользователя.







Вернувшись в меню мы решаем посмотреть статистику библиотеки, нажав на соответствующую кнопку мы можем увидеть требующиеся нам данные



# Заключение

При выполнении заданий как общих, так и индивидуальных поставленные цели были достигнуты.

Таким образом, в результате прохождения учебной практики были получены знания языка c#, навыки при работе со средой разработки visual studio, а также windows forms.

# Список использованных источников

1. [https://docs.microsoft.com](https://docs.microsoft.com/) – Официальный сайт документации Microsoft
2. [http://www.cyberforum.ru](http://www.cyberforum.ru/) – Форум программистов и сисадминов Киберфорум.
3. [https://metanit.com](https://metanit.com/) – Сайт о программировании

# Приложения

Приложение 1:

Код программы:

Файл Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace library\_project

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

Library\_main.Update\_library();

}

}

public class Library\_main

{

public static library Main\_library = new library();

public static void Update\_library()

{

System.IO.StreamWriter file = new System.IO.StreamWriter("../readers.txt", false, System.Text.Encoding.Default);

List<reader> readers\_main = new List<reader>();

readers\_main = Main\_library.Get\_readers();

List<book> books\_main = new List<book>();

books\_main = Main\_library.Get\_books();

foreach (reader i in readers\_main)

{

file.Write(i.Get\_reader\_information() + ' ');

List<book> reader\_books\_main = new List<book>();

reader\_books\_main = i.Get\_reader\_books();

if(reader\_books\_main != null)

{

foreach (book j in reader\_books\_main)

{

file.Write(j.Get\_Name() + ' ');

}

}

file.WriteLine();

}

file.Close();

file = new System.IO.StreamWriter("../books.txt", false, System.Text.Encoding.Default);

foreach (book i in books\_main)

{

file.WriteLine(i.Get\_information());

}

file.Close();

}

}

}

Файл book.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace library\_project

{

public class book

{

string Name;

string Year;

string Autor;

string Publisher;

string Code;

bool Available;

public book()

{

}

public book(string Autor, string Name, string Publisher, string Year, string Code)

{

this.Name = Name;

this.Year = Year;

this.Autor = Autor;

this.Publisher = Publisher;

this.Code = Code;

this.Available = true;

}

public void Change\_code(string Code)

{

this.Code = Code;

}

public void Change\_Available()

{

if (this.Available)

{

this.Available = false;

}

else this.Available = true;

}

public string Get\_Name()

{

return this.Name;

}

public string Get\_information()

{

return Autor + ' ' + Name + ' ' + Publisher + ' ' + Year + ' ' + Code;

}

public bool Get\_Available()

{

return this.Available;

}

}

}

Файл reader.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace library\_project

{

public class reader

{

string Name;

string Surname;

string Birthday;

string Registration;

List<book> books\_reader;

public reader()

{

books\_reader = new List<book>();

}

public reader(string Name, string Surname, string Birthday, string Registration)

{

this.Name = Name;

this.Surname = Surname;

this.Birthday = Birthday;

this.Registration = Registration;

books\_reader = new List<book>();

}

public void new\_reader\_book(book New\_book)

{

books\_reader.Add(New\_book);

}

public void remove\_reader\_book(book Book)

{

books\_reader.Remove(Book);

}

public string Get\_reader\_information()

{

string result = this.Name + ' ' + this.Surname + ' ' + this.Birthday + ' ' + this.Registration;

return result;

}

public List<book> Get\_reader\_books()

{

return books\_reader;

}

public string Get\_reader\_Registration()

{

return this.Registration;

}

}

}

Файл library.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace library\_project

{

public class library

{

List<reader> readers = new List<reader>();

List<book> books = new List<book>();

public library()

{

System.IO.StreamReader file = new System.IO.StreamReader("../books.txt", Encoding.Default);

string line = file.ReadLine();

while(line != null && line != ""){

string[] words = line.Split(new char[] {' '});

book New\_book = new book(words[0], words[1], words[2], words[3], words[4]);

books.Add(New\_book);

line = file.ReadLine();

}

file.Close();

file = new System.IO.StreamReader("../readers.txt", System.Text.Encoding.Default);

line = file.ReadLine();

while (line != null && line != "")

{

string[] words = line.Split(new char[] { ' ' });

reader New\_reader = new reader(words[0], words[1], words[2], words[3]);

for(int i = 4; i < words.Length; i++)

{

foreach(book k in books)

{

if(k.Get\_Name() == words[i])

{

New\_reader.new\_reader\_book(k);

k.Change\_Available();

}

}

}

readers.Add(New\_reader);

line = file.ReadLine();

}

file.Close();

}

public void New\_Reader(string Name, string Surname, string Birthday)

{

Name = Name.Replace(" ", "\_");

Surname = Surname.Replace(" ", "\_");

string DataTime = Convert.ToString(DateTime.Today).Substring(0, 10);

reader New\_reader = new reader(Name, Surname, Birthday, DataTime);

readers.Add(New\_reader);

Library\_main.Update\_library();

}

public void New\_Book(string Autor, string Name, string Publisher, string Year, string Code)

{

Autor = Autor.Replace(" ", "\_");

Name = Name.Replace(" ", "\_");

Publisher = Publisher.Replace(" ", "\_");

book New\_book = new book(Autor, Name, Publisher, Year, Code);

books.Add(New\_book);

Library\_main.Update\_library();

}

public reader Get\_reader(string Data)

{

reader result = null;

foreach (reader i in readers)

{

if(i.Get\_reader\_information() == Data)

{

result = i;

}

}

return result;

}

public List<book> Get\_books()

{

return books;

}

public List<reader> Get\_readers()

{

return readers;

}

public void New\_book\_reader(string Data\_reader, string Data\_book)

{

foreach(reader i in readers)

{

if(i.Get\_reader\_information() == Data\_reader)

{

foreach(book j in books)

{

if(j.Get\_information() == Data\_book)

{

readers[readers.IndexOf(i)].new\_reader\_book(books[books.IndexOf(j)]);

books[books.IndexOf(j)].Change\_Available();

}

}

}

}

Library\_main.Main\_library.Add\_book\_month();

Library\_main.Update\_library();

}

public void Remove\_book\_reader(string Data\_reader, string Data\_book)

{

foreach (reader i in readers)

{

if (i.Get\_reader\_information() == Data\_reader)

{

foreach (book j in books)

{

if (j.Get\_information() == Data\_book)

{

readers[readers.IndexOf(i)].remove\_reader\_book(books[books.IndexOf(j)]);

books[books.IndexOf(j)].Change\_Available();

}

}

}

}

Library\_main.Update\_library();

}

public void Remove\_reader(string Data\_reader)

{

foreach(reader i in readers)

{

if (i.Get\_reader\_information() == Data\_reader)

{

List<book> Data\_books = new List<book>();

Data\_books = i.Get\_reader\_books();

foreach(book j in Data\_books)

{

foreach(book k in books)

{

if(j.Get\_information() == k.Get\_information())

{

books[books.IndexOf(k)].Change\_Available();

break;

}

}

}

readers.Remove(i);

break;

}

}

Library\_main.Update\_library();

}

public reader Get\_reader\_book(string Data\_book)

{

reader result = new reader();

result = null;

foreach(book i in books)

{

if(i.Get\_information() == Data\_book)

{

if (i.Get\_Available())

{

return result;

}

else

{

foreach(reader j in readers){

List<book> Data\_reader\_book = new List<book>();

Data\_reader\_book = j.Get\_reader\_books();

foreach(book k in Data\_reader\_book)

{

if (k.Get\_information() == Data\_book)

{

result = readers[readers.IndexOf(j)];

return result;

}

}

}

}

}

}

return result;

}

public void New\_code\_book(string Data\_book, string New\_code)

{

foreach(book i in books)

{

if(i.Get\_information() == Data\_book)

{

books[books.IndexOf(i)].Change\_code(New\_code);

}

}

Library\_main.Update\_library();

}

public void Delete\_book(string Data\_book, string Data\_reader)

{

if(Data\_reader == null)

{

foreach(book i in books)

{

if(i.Get\_information() == Data\_book)

{

books.Remove(books[books.IndexOf(i)]);

break;

}

}

}

else

{

this.Remove\_book\_reader(Data\_reader, Data\_book);

foreach (book i in books)

{

if (i.Get\_information() == Data\_book)

{

books.Remove(books[books.IndexOf(i)]);

break;

}

}

}

Library\_main.Update\_library();

}

public bool Proof\_Code(string Code)

{

foreach(book i in books)

{

if (i.Get\_information().Contains(Code))

{

return true;

}

}

return false;

}

public void Add\_book\_month()

{

int month = DateTime.Today.Month;

System.IO.StreamReader file = new System.IO.StreamReader("../statistic.txt", Encoding.Default);

string Data = file.ReadLine();

string[] words = Data.Split(new char[] { ' ' });

int count = int.Parse(words[1]);

if(int.Parse(words[0]) == month)

{

count++;

}

else

{

count = 1;

}

file.Close();

System.IO.StreamWriter file\_write = new System.IO.StreamWriter("../statistic.txt", false, System.Text.Encoding.Default);

file\_write.WriteLine(Convert.ToString(month) + ' ' + Convert.ToString(count));

file\_write.Close();

}

public string Get\_count\_readers\_month()

{

System.IO.StreamReader file = new System.IO.StreamReader("../statistic.txt", Encoding.Default);

string Data = file.ReadLine();

file.Close();

string[] words = Data.Split(new char[] { ' ' });

int month\_current = int.Parse(words[0]);

int counter = 0;

foreach (reader i in readers)

{

string reader\_registration = i.Get\_reader\_Registration();

DateTime dt = DateTime.Parse(reader\_registration);

if(month\_current == dt.Month && DateTime.Today.Year == dt.Year)

{

counter++;

}

}

return Convert.ToString(counter);

}

public string Get\_count\_book\_month()

{

System.IO.StreamReader file = new System.IO.StreamReader("../statistic.txt", Encoding.Default);

string Data = file.ReadLine();

file.Close();

string[] words = Data.Split(new char[] { ' ' });

return words[1];

}

public string Get\_count\_readers()

{

return Convert.ToString(readers.Count);

}

}

}