Федеральное государственное образовательное бюджетное учреждение

высшего образования

**«Финансовый университет**

**при Правительстве Российской Федерации»**

**(Финансовый университет)**

Департамент анализа данных, принятия решений и финансовых технологий

**Курсовая работа по учебной дисциплине**

**«Операционные системы»**

**на тему:**

**Информационно-справочная система «Ноутбуки»**

Выполнил:

Студент группы ПИ 2-1

Матиив Роман Александрович

Научный руководитель:

Кишкович Юрий Павлович,

Доцент к/н

Москва

2017

Оглавление

[1. Условие задачи. Описание предметной области 3](#_Toc483706469)

[2. Описание класса предметной области 7](#_Toc483706470)

[3. Описание интерфейса пользователя на контрольном примере 13](#_Toc483706471)

[4. Состав приложения 21](#_Toc483706472)

[Список литературы 23](#_Toc483706473)

[Приложение. Исходный текст программы. 24](#_Toc483706474)

# **1. Условие задачи. Описание предметной области**

Требуется разработать информационно-справочную систему (далее ИСС) «Ноутбуки» для магазина, продающего ноутбуки. Магазину требуется вести учет имеющихся ноутбуков, а также оптимизировать деятельность персонала магазина по работе с ИСС.

Общие требования к программе следующие:

1. Разрабатываемое приложение должно представлять собой Windows приложение.
2. Приложение в обязательном порядке независимо от предметной области, указанной в задании, должно выполнять следующие операции:

- при первом запуске программы создать список в коде программы и отобразить его в DataGridView.

- реализовать добавление в список нового объекта, удаление объекта из списка, редактирование объекта в списке;

- вывести на форму сведения об объектах списка, удовлетворяющих введенному пользователем критерию;

- сохранить список объектов в указанном пользователем файле (используя сериализацию и стандартное диалоговое окно). Для сохранения файла приложение должно создать подкаталог в текущей папке приложения, если подкаталог не существует. Имя подкаталога и файла выбирает пользователь.

- создать копию файла, используя метод копирования.

- используя меню или панель инструментов, вывести информацию о пути к программе, а также файлах и вложенных папках текущего каталога.

- сохранить в реестре имя подкаталога и файла.

- в подкаталоге Manual с помощью Блокнота создать текстовый файл с инструкцией к программе. Для просмотра файла использовать меню или панель инструментов, по которому запустить Блокнот и загрузить инструкцию.

- при повторном запуске программы прочитать из реестра запомненные имена, создать список путем чтения (десериализации) данных из файла. Отобразить список в DataGridView. При повторном запуске программы список загружается из файла, создание списка в коде программы не использовать.

1. Список должен быть реализован в виде коллекции, например, динамического массива. Для сохранения файла использовать стандартное диалоговое окно.
2. Программа не должна завершаться аварийно: сообщения о некорректном вводе данных, противоречивых или недопустимых значениях данных, при отсутствии данных по функциональному запросу пользователя и других нештатных ситуациях отображать в окнах сообщений.
3. Программа должна быть читабельной и содержать полезные комментарии.
4. Список ноутбуков должен передаваться при, при первом запуске, в виде текстового файла с расширением txt. При последующих запусках приложение должно само считывать список из ранее сохраненного пользователем, или автоматически сохраненного, сериализованного списка, представленного в виде xml файла. В приложении данные должны отображаться в таблице (элемент управления DataGridView).

Работать с данной ИСС будут сотрудники магазина. ИСС должна позволять им решать следующие задачи:

1. Применение фильтра в ИСС по следующим параметрам:
   1. Название
   2. ID
   3. Цена
   4. Объем оперативной памяти
   5. Объем жесткого диска
   6. Диагональ дисплея
   7. Процессор
2. Изменение товара в ИСС
3. Добавление товара в ИСС
4. Удаление товара из ИСС

Для выполнения поставленной задачи будет использоваться интегрированная среда разработки (далее IDE) Visual Studio от компании Microsoft. Данная IDE будет использоваться, так как она включает в себя редактор форм для упрощения создания графического интерфейса приложения, поддерживает технологию авто дополнения кода и обладает возможностью простейшего рефакторинга.

Языком программирования выбран высокоуровневый язык C#. Так, как он довольно прост в использовании и отлично подходит для создания Windows приложений.

Объектом предметной области является товар ноутбук.

Для выполнения поставленной задачи будет разработан класс Ноутбук(Laptop) и класс товар(Product), который наследует класс Ноутбук(Laptop).

Также для визуализации ИСС будут разработаны следующие формы:

1. MainForm
2. AddFrom
3. FormForSave
4. Edit\_Show
5. CheckUsers
6. Registration

MainForm – главная форма, из которой работники магазина получают иформацию о товаре, применяют фильтр к полому списку товаров а также переходят к другим формам.

AddForm – служит для добавления нового товара в общий список.

FormForSave – позволяет сохранить список ноутбуков, сериализуя его.

Edit\_Show – форма, позволяющая редактировать запись о ноутбуке или просматривать полную информацию о конкретном ноутбуке.

CheckUsers – форма, проверяющая зарегистрирован пользователь или нет.

Registration – форма, для добавления пользователя в систему.

# **2. Описание класса предметной области**

Приложение реализовано с помощью классов, представленных на Рис 2.1 в виде диаграммы классов. Более подробная информация о классах представлена в таблицах далее.

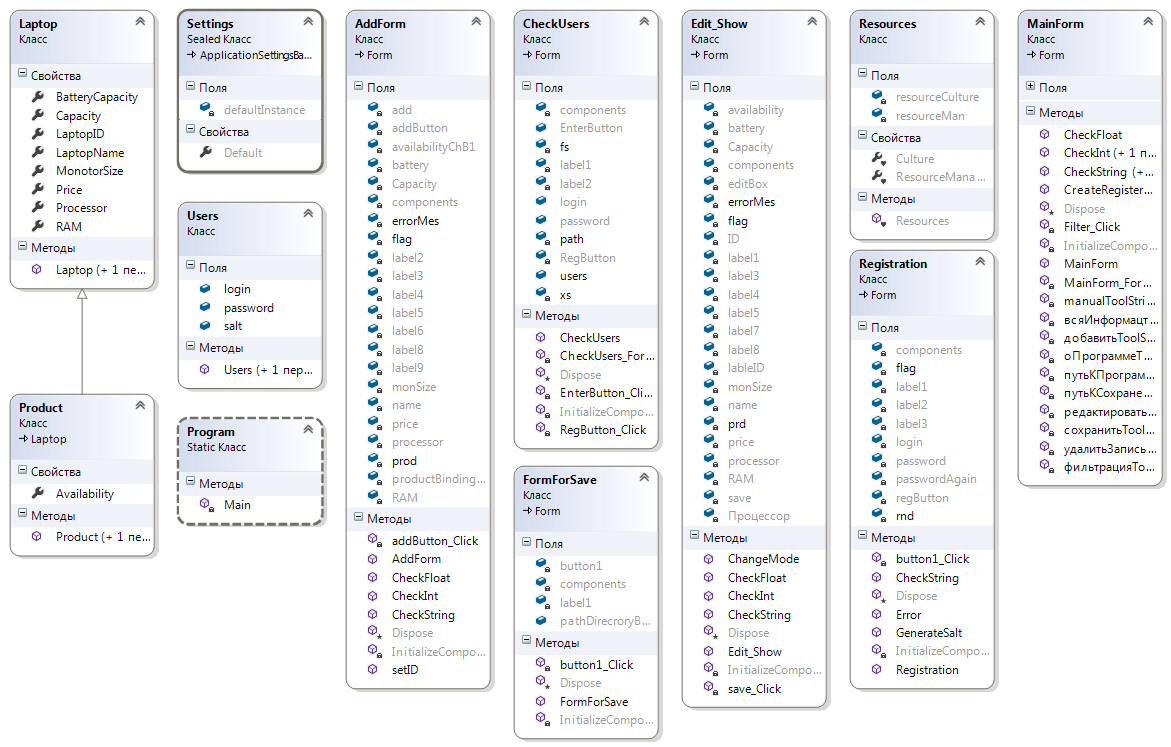


Рис. 2.1 Диаграмма классов

Таблица 2.1

Описание классов программы

|  |  |
| --- | --- |
| Класс | Функция |
| Laptop | Класс, содержащий в себе автоматические свойства и конструктор, реализующий объект ноутбук. |
| Product | Класс описывающий ноутбук как продукт. Содержит автоматическое поле и конструктор. |
| MainForm | Главная форма, позволяющая просматривать список ноутбуков в магазине а также применять фильтрацию по различным параметрам. |
| AddForm | Форма, добавляющая новую запись в общий список. |
| Edid\_Show | Форма для редактирования и просмотра записи. |
| FormForSave | Форма позволяющая выбрать название папки для сохранения сериализованного файла, содержащего список ноутбуков. |
| Program | Класс входа в программу. |
| Users | Реализует объект пользователь. |
| CheckUsers | Форма проверяющая зарегистрирован ли пользователь. |
| Registration | Форма для регистрации пользователя в системе. |

Таблица 2.2

Класс Laptop

|  |  |  |
| --- | --- | --- |
| Название | Тип | Обозначение |
| laptopID | int | Ид ноутбука |
| laptopName | string | Название ноутбука |
| manufacturer | string | Производитель |
| ram | int | Объем оперативной памяти |
| сapacity | int | Объем жесткого диска |
| monitorSize | int | Размер монитора |
| processor | string | Процессор |
| batteryCapacity | int | Кол-во мАч в батарее |
| laptopType | string | Тип ноутбука |
| price | int | Цена |

Поле laptopID является уникальным

Таблица 2.3

Класс Product

|  |  |  |
| --- | --- | --- |
| Название | Тип | Обозначение |
| laptop | Laptop | Ноутбук |
| availability | bool | Наличие |

Таблица 2.4

Методы AddForm

|  |  |
| --- | --- |
| Название | Функция |
| addButton\_Click | При нажатии на кнопку проверяет веденные значения и ели они не противоречивы добавляет запись в общий список. |
| CheckInt | Проверяет вводимые значения типа int |
| CheckString | Проверяет вводимые значения типа string |
| CheckFloat | Проверяет вводимые значения типа float |
| SetID | Отвечает за авто-инкремент поля LaptopID |

Таблица 2.5

Методы Edit\_Show

|  |  |
| --- | --- |
| Название | Функция |
| save\_Click | Проверяет введенные данные, вызывая другие методы и сохраняет измененные данные. |
| ChangeMode | Устанавливает режим просмотра или редактирования |
| CheckInt | Проверяет поля с типом данных int |
| CheckString | Проверяет поля с типом данных string |
| CheckFloat | Проверяет поля с типом данных float |

Таблица 2.6

Методы FormForSave

|  |  |
| --- | --- |
| Название | Функция |
| button1\_Click | Запоминает название папки для сохранения |

Таблица 2.7

Методы MainForm

|  |  |
| --- | --- |
| Название | Функция |
| MainForm | Происходит работа программы и вызов других методов |
| добавитьToolStripMenuItem\_Click | Вызывает AddForm |
| Filter\_Click | Фильтрует в зависимости от заданых параметров |
| CheckInt | Проверяет поля с типом int. Имеет 2 перегрузки. |
| CheckString | Проверяет поля с типом string. Имеет 2 перегрузки. |
| CheckFloat | Проверяет поля с типом float. Имеет 2 перегрузки. |
| фильтрацияToolStripMenuItem\_Click | Отображает/Убирает функционал фильтрации |
| удалитьЗаписьToolStripMenuItem\_Click | Удаляет запись |
| manualToolStripMenuItem\_Click | Открывает инструкцию |
| сохранитьToolStripMenuItem\_Click | Сохраняет список в файл (и создает копию) |
| CreateRegister | Создаю регистр для записи в реест и запичываю в него путь до программы и путь до сохраненного файла.  Имеет 2 перегрузки. |
| редактироватьToolStripMenuItem\_Click | Позволяет изменить информацию о продукте |
| всяИнформацтяToolStripMenuItem\_Click | Выводит всю информацию о продукте |
| путьКПрограммеToolStripMenuItem\_Click | Выводит путь к программе |
| путьКСохраненномуФайлуToolStripMenuItem\_Click | Выводит путь к сохраненному файлу |

Таблица 2.8

Users

|  |  |  |
| --- | --- | --- |
| Название | Тип | Обозначение |
| login | string | Логин |
| Password | int | Зашифрованный пароль |
| salt | string | Соль |

*Таблица 3.9*

Методы CheckUsers

|  |  |
| --- | --- |
| Название | Функция |
| RegButton\_Clicl | Перенаправляет к форме регистрации |
| EnterButton\_Click | Если логи и пароль вверны перенаправляет на главную форму |

# **3. Описание интерфейса пользователя на контрольном примере**

При входе в приложение, система запросит у пользователя логин и пароль (см. рис. 3.1). Также с данного окна возможно перейти к форме для регистрации, показанной на рис. 3.2. После того, как пользователь введет имеющуюся в системе связку логин-пароль, он попадет на главную форму (см рис 3.3).

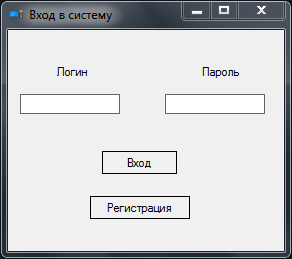


Рис.3.1 Форма авторизации

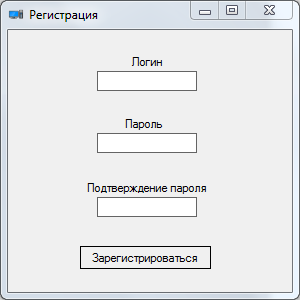


Рис.3.2 Форма для регистрации

Из главной формы (рис 3.3) пользователь может реализовать весь функционал программы, посредством вызова подчиненных форм из главной. Действия при вызове подчинённых форм описаны далее.

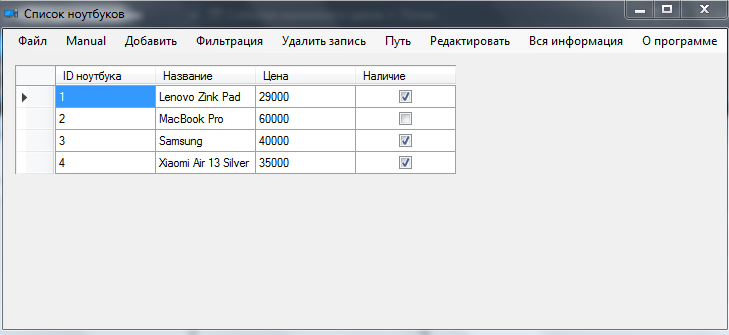


Рис. 3.3 Главная форма

После редактирования списка, внесения в него ноутбуков, удаления ноутбуков из него мы можем сохранить список в файл через меню «Файл» затем «Сохранить».

Если запуск первый, у пользователя будет спрошено название желаемой папки для сохранения, если нет, то отобразиться диалоговое окно, для сохранения файла, показанное на рис 3.4. Здесь пользователь должен ввести название файла и нажать «Сохранить».

Если файл с таким названием уде существует, система проинформирует пользователя об этом (см. Рис. 3.5) и, в зависимости от действий пользователя, либо заменит старый файл новым, либо вернет пользователя обратно на диалоговое окно для сохранения файла (см. Рис. 3.4).

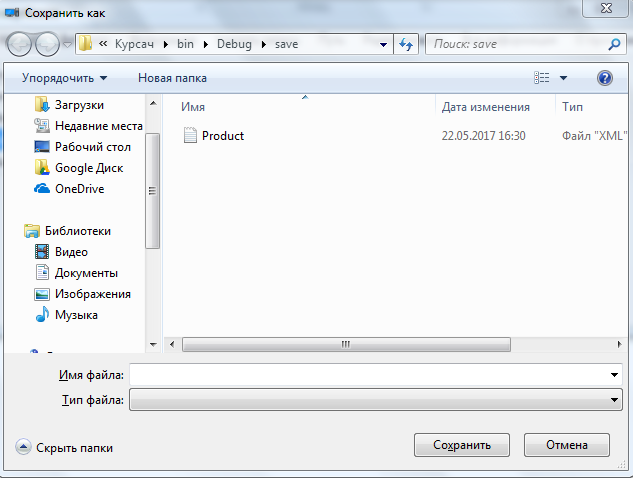


Рис. 3.4 Диалоговое окно сохранения

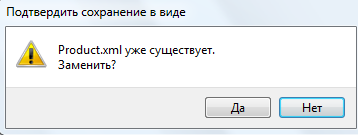


Рис. 3.5 Message Box с предупреждением

При нажатии на кнопку «Manual» в панели меню, откроется блокнот с инструкцией к программе представленной на рис. 3.6.

Файл с инструкцией доступен только для чтения, так что изменить его пользователь не сможет.

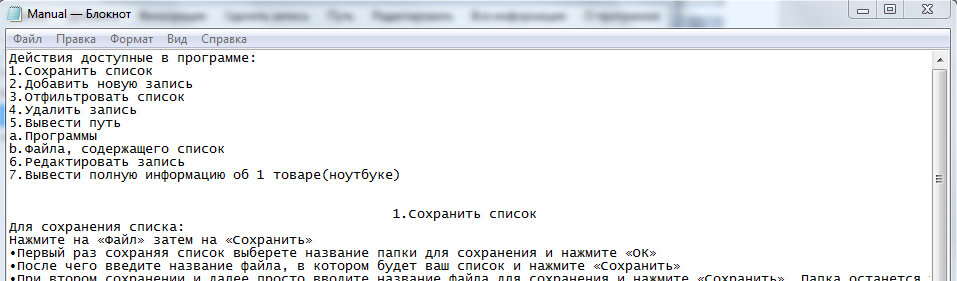


Рис. 3.6 Инструкция к программе

При нажатии на кнопку «Добавить» откроется форма для добавления представленная на рис. 3.7. Она позволяет пользователю добавлять новый ноутбук в список. При этом введенные значения проверяются на правильность и логичность (например, цена не может быть отрицательной).

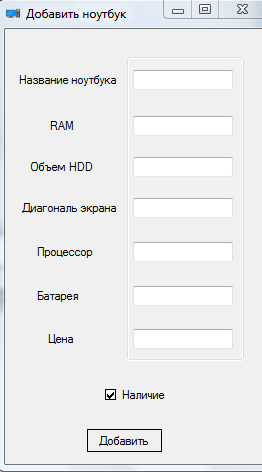


Рис. 3.7 Форма добавления ноутбуков

Форма до активации функционала фильтрации представлена на рис. 3.8.

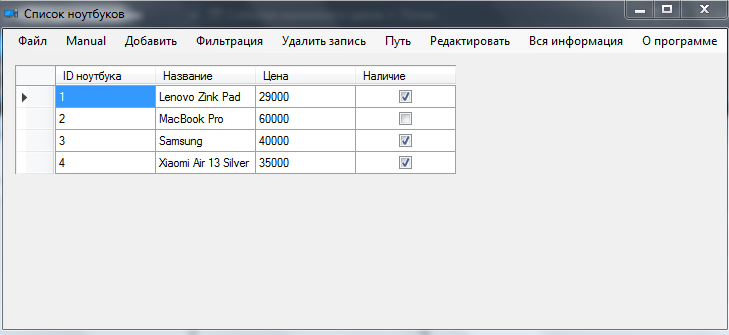


Рис. 3.8 форма без фильтрации

После нажатия на «Фильтрация» активируется функционал фильтрации что отчетливо видно из рис. 3.9. Благодаря чему, пользователь может фильтровать список по своему усмотрению и быстрее искать нужный ему ноутбук.

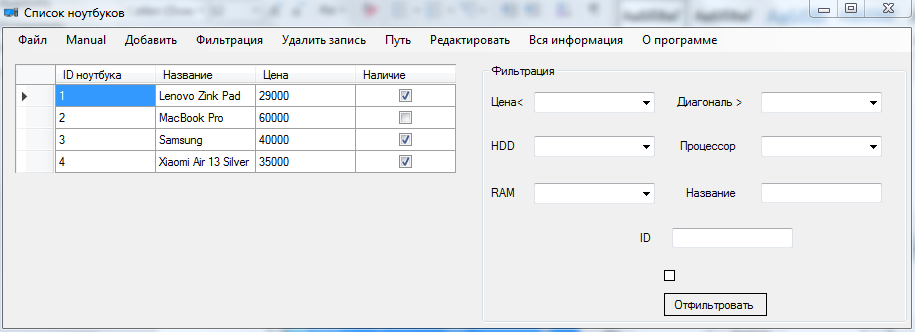


Рис. 3.9 Форма с фильтрацией

Программа также, при нажатии «Путь» а затем «Путь к программе» выведет информацию о полном пути к программе (см. Рис. 3.10), а также о полном пути к файлу, содержащему сериализованный список (см. Рис. 3.11), при нажатии «Путь» а затем «Путь к сохранённому файлу».

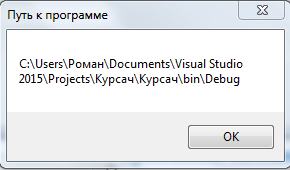


Рис. 3.10 Отображение пути к программе

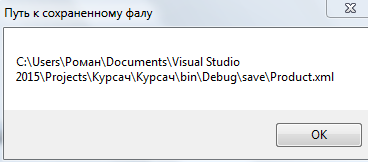


Рис. 3.11 Отображение пути к сохраненному файлу

Редактирование существующего списка осуществляется при помощи формы, представленной на рис. 3.12, запуск которой осуществляется посредством нажатия кнопки «Редактировать» на панели меню.

После внесения изменений система проверит корректность внесенных изменений и если они не противоречат логике, изменит выбранную пользователем запись.

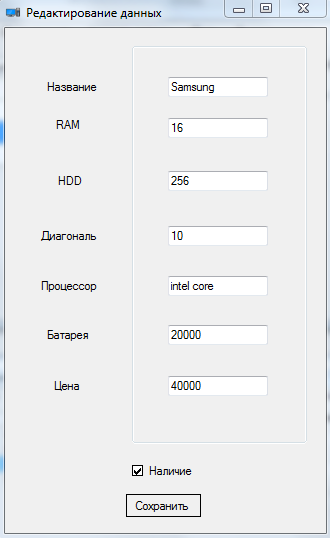


Рис. 3.12 Форма для редактирования данных

Форма, представленная на рис. 3.13, вызываемая через кнопку «Вся информация», позволяет пользователю просмотреть полную характеристику выбранного из списка ноутбука.

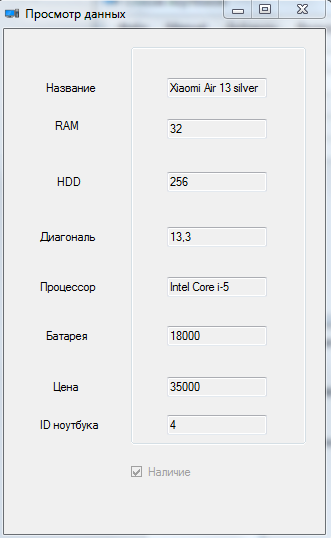


Рис. 3.13 Форма просмотра данных

По кнопке «О программе» вызывается Message Box, позволяющий увидеть информацию о программе и о ее создателе(см. Рис. 3.14).

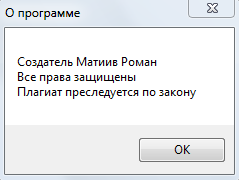


Рис. 3.14 MessageBox « О программе»

# **4. Состав приложения**

Папка проекта имеет следующую иерархию:

* Классы приложения
* Папка object
* Папка Properties
* Иконка приложения
* Папка bin
  + Папка Release
  + Папка Debug
    - Файл с исходным списком(для чтения при 1-ом запуске)
    - Файл с срериализованным списком
    - Файл с расширением exe
    - Папку save
      * Сохраненный сериализованный список
    - Папку Backup
      * Копия сераилизованного списка
    - Папку Manual
      * Инструкцию к программе
    - Папку Users
      * Данные о пользователях

Представленная выше иерархия содержит не все файлы, которые содержит проект, но тем не менее в наглядной форме визуализирует важные файлы и папки проекта.

В папке Debug изначально находиться файл со списком, это необходимо для того чтобы выполнить одно из требований.

В папку Backup помешаются копии сериализованных списков, для того, чтобы в случаи повреждения или потери исходного файла со списком его можно было восстановить.

Папка Manual содержит файл с инструкцией, которая доступна только на чтение, для того чтобы инструкцию к программе нельзя было несанкционированно изменить или удалить.

Папка Users содержит данные о пользователе, а именно: логин, зашифрованных пароль и соль, с помощью которой и осуществляется шифрование.

Папка проекта содержит также иконку программы.

Имя папки для сохранения, в данном случае save, пользователь может выбирать самостоятельно. В ней содержится сериализованный список ноутбуков.

# **Список литературы**

1. Павловская Т.А. C#. Программирование на языке высокого уровня. Учебник — СПб.: ПИТЕР, 2006-2011. — 432 с.
2. Шилдт Г. С#: учебный курс. — СПб.: Питер; К.: Издательская группа BHV, 2003. — 512 с.
3. Библиотека MSDN [Электронный ресурс] –   
   Электрон. дан. – Microsoft Corporation, 2017 – Режим доступа: http://msdn.microsoft.com свободный. – Загл. с экрана. – Яз. рус.
4. Cyberforum информационный портал [Электронный ресурс]. – Режим доступа – http://www.cyberforum.ru

# **Приложение. Исходный текст программы.**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсач

{

public partial class AddForm : Form

{

bool flag = true;

Product prod = new Product();

string errorMes = "";

public AddForm()

{

InitializeComponent();

}

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

{

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Проверка заполнены ли все поля

flag = true;

foreach (TextBox i in add.Controls)

{

if (i.Text.Length == 0)

flag = false;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

errorMes = "Заполните все поля";

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Проверка типов

CheckString(name);

CheckInt(RAM);

CheckInt(Capacity);

CheckFloat(monSize);

CheckString(processor);

CheckInt(battery);

CheckFloat(price);

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

if (flag)

{

prod.LaptopID = setID();

prod.LaptopName = add.Controls[0].Text;

prod.RAM = int.Parse(add.Controls[1].Text);

prod.Capacity = int.Parse(add.Controls[2].Text);

prod.MonotorSize = float.Parse(add.Controls[3].Text);

prod.Processor = add.Controls[4].Text;

prod.BatteryCapacity = int.Parse(add.Controls[5].Text);

prod.Price = float.Parse(add.Controls[6].Text);

prod.Availability = availabilityChB1.Checked;

MainForm.prodLst.Add(prod);//Добавление 1 обекта в BindingList

Close();

}

else

{

MessageBox.Show(errorMes);

}

}

public void CheckInt(TextBox tb)//Проверка на int

{

int tmp = 0;

if (!int.TryParse(tb.Text, out tmp))

{

flag = false;

errorMes = "Неправильный тип данных в поле " + tb.Name + " \nВведите целое число";

}

if (tmp < 0)

{

flag = false;

errorMes = "Поле " + tb.Name + " не может быть отрицательным";

}

}

public void CheckString(TextBox tb)//Проверка на string

{

int tmp1 = 0;

float tmp2;

if ((int.TryParse(tb.Text, out tmp1)) | (float.TryParse(tb.Text, out tmp2)))

{

flag = false;

errorMes = "Неправильный тип данных в поле " + tb.Name + " \nВведите строковое значение";

}

}

public void CheckFloat(TextBox tb)//Проверка на float

{

float tmp = 0;

if (!float.TryParse(tb.Text, out tmp))

{

flag = false;

errorMes = "Неправильный тип данных в поле " + tb.Name + " введите цисло \nПример: 2,75 (через запятую)";

}

if (tmp < 0)

{

flag = false;

errorMes = "Поле " + tb.Name + " не может быть отрицательным";

}

}

public int setID()

{

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ищет естьли пробелы в списке ID(удаленные ID)

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

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

lst.Add(MainForm.prodLst[i].LaptopID);

lst.Sort();

for (int i = 0; i < lst.Count-1; i++)

{

if (lst[i] + 1 != lst[i + 1])

return lst[i] + 1;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

return lst.Max()+1;//Если пробелов нет то вернет max+1

}//Реализует авто инкремент ID

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсач

{

public partial class Edit\_Show : Form

{

Product prd = new Product();

bool flag = true;

string errorMes = "";

public Edit\_Show()

{

InitializeComponent();

prd = MainForm.prodLst[MainForm.i];//Отображаю выбранный объект

name.Text = prd.LaptopName;

ID.Text = prd.LaptopID.ToString();

RAM.Text = prd.RAM.ToString();

Capacity.Text = prd.Capacity.ToString();

monSize.Text = prd.MonotorSize.ToString();

processor.Text = prd.Processor.ToString();

battery.Text = prd.BatteryCapacity.ToString();

price.Text = prd.Price.ToString();

availability.Checked = prd.Availability;

ChangeMode(MainForm.flagEditShow);//Режим просмотра/редактирования

}

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

{

flag = true;

foreach (TextBox i in editBox.Controls)

{

if (i.Text.Length == 0)

flag = false;

}

errorMes = "Заполните все поля";

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Проверяю типы данных

CheckString(name);

CheckInt(RAM);

CheckInt(Capacity);

CheckFloat(monSize);

CheckString(processor);

CheckInt(battery);

CheckFloat(price);

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Переношу изменения в главный список

if (flag)

{

prd.LaptopName = name.Text;

prd.RAM = int.Parse(RAM.Text);

prd.Capacity = int.Parse(Capacity.Text);

prd.MonotorSize = float.Parse(monSize.Text);

prd.Processor = processor.Text;

prd.BatteryCapacity = int.Parse(battery.Text);

prd.Price = float.Parse(price.Text);

prd.Availability = availability.Checked;

MainForm.prodLst[MainForm.i] = prd;

this.Close();

}

else

{

MessageBox.Show(errorMes);

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

public void ChangeMode (bool flag)

{

foreach (TextBox i in editBox.Controls)

{

i.ReadOnly = !flag;

}

availability.Enabled = flag;

save.Enabled = flag;

save.Visible = flag;

if (flag)

this.Text = "Редактирование данных";

else

this.Text = "Просмотр данных";

if (flag)

{

ID.Visible = false;

lableID.Visible = false;

}

else

{

ID.Visible = true;

lableID.Visible = true;

}

}//Переключает ражим просмтора/редакирования

public void CheckInt(TextBox tb)//Проверка на int

{

int tmp=0;

if(!int.TryParse(tb.Text, out tmp))

{

flag = false;

errorMes = "Неправильный тип данных в поле " + tb.Name + " \nВведите целое число";

}

if(tmp < 0)

{

flag = false;

errorMes = "Поле " + tb.Name + " не может быть отрицательным ";

}

}

public void CheckString(TextBox tb)//Проверка на string

{

int tmp1 =0;

float tmp2;

if((int.TryParse(tb.Text, out tmp1))|(float.TryParse(tb.Text, out tmp2)))

{

flag = false;

errorMes = "Неправильный тип данных в поле " + tb.Name + " \nВведите строковое значение";

}

}

public void CheckFloat(TextBox tb)//Проверка на float

{

float tmp = 0;

if (!float.TryParse(tb.Text, out tmp) & (tmp > 0))

{

flag = false;

errorMes = "Неправильный тип данных в поле " + tb.Name + " введите цисло \nПример: 2,75 (через запятую)";

}

if (tmp < 0)

{

flag = false;

errorMes = "Поле " + tb.Name + " не может быть отрицательным ";

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсач

{

public partial class FormForSave : Form

{

public FormForSave()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

if(pathDirecroryBox.Text.Trim(' ').Length!=0)

{

MainForm.pathDirectory= pathDirecroryBox.Text.Trim(' ');

Close();

}

else

{

MessageBox.Show("Введите название папки");

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Курсач

{

public class Laptop

{

public int LaptopID { get; set; }

public string LaptopName { get; set; }

public int RAM { get; set; }

public int Capacity { get; set; }

public float MonotorSize { get; set; }

public string Processor { get; set; }

public int BatteryCapacity { get; set; }

public float Price { get; set; }

public Laptop(int ID,string name,int RAM,int capacity,float monitorSize,string Proc,int battery,float price)

{

this.LaptopID = ID;

this.LaptopName = name;

this.RAM = RAM;

this.Capacity = capacity;

this.MonotorSize = monitorSize;

this.Processor = Proc;

this.BatteryCapacity = battery;

this.Price = price;

}

public Laptop() { }

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

using System.Xml.Serialization;

using System.Diagnostics;

using Microsoft.Win32;

namespace Курсач

{

public partial class MainForm : Form

{

public static BindingList<Product> prodLst = new BindingList<Product>();//Список ноутбуков магазина

//Для сериализации/десериализации

XmlSerializer xs = new XmlSerializer(typeof(BindingList<Product>));

FileStream fs;

//Путь к файлу и деректории

string path = "";

public static string pathDirectory="";

public string pathBackup = Directory.GetCurrentDirectory() + "\\Backup\\Backup.xml";

public static int i;//Передает индекс выделенной строки в другую форму

public static bool flagEditShow = true;//В зависимости от значения форма отображается в режие просмотра или редактирования

//Проверка на ошибки

string errorMes = "";

bool flag = true;

public MainForm()

{

InitializeComponent();

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Работа с регистрами

RegistryKey currentUserKey = Registry.CurrentUser;

//Если регистра с полным путем нет создаю его и задаю путь по умолчанию

RegistryKey pathk = currentUserKey.OpenSubKey("path");

if(pathk==null)

{

pathk= currentUserKey.CreateSubKey("path");

pathk.SetValue("path", "Product.xml");

}

pathk.Close();

//Считываю регистр полного пути

pathk = currentUserKey.OpenSubKey("path");

path = pathk.GetValue("path").ToString();

pathk.Close();

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

this.Width = 740;//Ширина окна по умолчанию

try

{

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//1-й запуск Создаю пустой xml файл куда потом и буду записывать и считаваю с txt файла

if(!File.Exists(path))//Если нет xml файла

{

fs = new FileStream(path, FileMode.Create);

string[] arr = File.ReadAllLines("tmp.txt");//Название указать вручную при первом запуске

for (int i = 0; i < arr.Length; i = i + 9)

prodLst.Add(new Product(int.Parse(arr[i]), arr[i + 1], int.Parse(arr[i + 2]), int.Parse(arr[i + 3]), float.Parse(arr[i + 4]), arr[i + 5], int.Parse(arr[i + 6]), float.Parse(arr[i + 7]), bool.Parse(arr[i + 8])));

xs = new XmlSerializer(typeof(BindingList<Product>));//Сериализую для 2-го и далее запуска

xs.Serialize(fs, prodLst);//Сериализую

fs.Close();

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Считывает сериализованный файл при 2-ом и далее запусках

fs = new FileStream(path, FileMode.Open);

if(fs.Length!=0)

prodLst = (BindingList<Product>)xs.Deserialize(fs);

fs.Close();

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

catch

{

MessageBox.Show("Файл не может быть правильно считат или не существует");

}

productBindingSource.DataSource = prodLst;//Главный список

}

private void добавитьToolStripMenuItem\_Click(object sender, EventArgs e)//Добавить новый ноутбук(product)

{

AddForm addForm = new AddForm();

addForm.ShowDialog();

}

private void Filter\_Click(object sender, EventArgs e) //Фильтрует в зависимости от заданых параметров

{

for (int i = 0; i < DGV.RowCount; i++)//Сброс фильтра(показывает все)

DGV.Rows[i].Visible = true;

flag = true;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Проверка типов данных

CheckFloat(priceBox);

CheckInt(ramBox);

CheckInt(hddBox);

CheckFloat(displayBox);

CheckString(processorBox);

CheckInt(IDBox);

CheckString(nameBox);

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

if (flag)

{

DGV.CurrentCell = null;//Убирает выделение ячейки

for (int i = 0; i < DGV.RowCount; i++)

{

//Цена

if (priceBox.Text.Trim(' ').Length != 0)

if (float.Parse(DGV["Price", i].Value.ToString()) > float.Parse(priceBox.Text.Trim(' ')))

DGV.Rows[i].Visible = false;

//RAM

if (ramBox.Text.Trim(' ').Length != 0)

if (int.Parse(DGV["RAM", i].Value.ToString()) != int.Parse(ramBox.Text.Trim(' ')))

DGV.Rows[i].Visible = false;

//HDD

if (hddBox.Text.Trim(' ').Length != 0)

if (int.Parse(DGV["Capacity", i].Value.ToString()) != int.Parse(hddBox.Text.Trim(' ')))

DGV.Rows[i].Visible = false;

//Диагональ

if (displayBox.Text.Trim(' ').Length != 0)

if (float.Parse(DGV["MonotorSize", i].Value.ToString()) > float.Parse(displayBox.Text.Trim(' ')))

DGV.Rows[i].Visible = false;

//Процессор

if (processorBox.Text.Trim(' ').Length != 0)

if (DGV["Processor", i].Value.ToString().ToLower() != processorBox.Text.ToLower().Trim(' '))

DGV.Rows[i].Visible = false;

//В наличае

if (availBox.Checked)

if (DGV["availability", i].Value.ToString() == "False")

DGV.Rows[i].Visible = false;

//ID

if (IDBox.Text.Trim(' ').Length != 0)

if (int.Parse(DGV["laptopID", i].Value.ToString()) != int.Parse(IDBox.Text.Trim(' ')))

DGV.Rows[i].Visible = false;

//Название

if (nameBox.Text.Trim(' ').Length != 0)

if (DGV["laptopName", i].Value.ToString().ToLower() != nameBox.Text.ToLower().Trim(' '))

DGV.Rows[i].Visible = false;

}

}

else

{

MessageBox.Show(errorMes);

}

}

public void CheckInt(ComboBox cb)//Проверка на int(ComboBox)

{

if (cb.Text.Length == 0) return;

int tmp = 0;

if (!int.TryParse(cb.Text, out tmp))

{

flag = false;

errorMes = "Неправильный тип данных в поле " + cb.Name + " \nВведите целое число";

}

if (tmp < 0)

{

flag = false;

errorMes = "Поле " + cb.Name + " не может быть отрицательным ";

}

}

public void CheckInt(TextBox cb)//Проверка на int(TexBox)

{

if (cb.Text.Length == 0) return;

int tmp = 0;

if (!int.TryParse(cb.Text, out tmp))

{

flag = false;

errorMes = "Неправильный тип данных в поле " + cb.Name + " \nВведите целое число";

}

if (tmp < 0)

{

flag = false;

errorMes = "Поле " + cb.Name + " не может быть отрицательным ";

}

}

public void CheckString(ComboBox cb)//Проверка на string(ComboBox)

{

if (cb.Text.Length == 0) return;

int tmp1 = 0;

float tmp2;

if ((int.TryParse(cb.Text, out tmp1)) | (float.TryParse(cb.Text, out tmp2)))

{

flag = false;

errorMes = "Неправильный тип данных в поле " + cb.Name + " \nВведите строковое значение";

}

}

public void CheckString(TextBox cb)//Проверка на string(TexBox)

{

if (cb.Text.Length == 0) return;

int tmp1 = 0;

float tmp2;

if ((int.TryParse(cb.Text, out tmp1)) | (float.TryParse(cb.Text, out tmp2)))

{

flag = false;

errorMes = "Неправильный тип данных в поле " + cb.Name + " \nВведите строковое значение";

}

}

public void CheckFloat(ComboBox cb)//Проверка на float

{

if (cb.Text.Length == 0) return;

float tmp = 0;

if (!float.TryParse(cb.Text, out tmp) & (tmp > 0))

{

flag = false;

errorMes = "Неправильный тип данных в поле " + cb.Name + " введите цисло \nПример: 2,75 (через запятую)";

}

if (tmp < 0)

{

flag = false;

errorMes = "Поле " + cb.Name + " не может быть отрицательным ";

}

}

private void фильтрацияToolStripMenuItem\_Click(object sender, EventArgs e)//Отображает/Убирает функционал фильтрации

{

for (int i = 0; i < DGV.RowCount; i++)//Сброс фильтра(показывает все)

DGV.Rows[i].Visible = true;

if (!Filter.Visible)

Filter.Visible = true;

else

Filter.Visible = false;

if (Filter.Visible)

this.Width = 924;

else

this.Width = 740;

priceBox.Text = "";

ramBox.Text = "";

hddBox.Text = "";

displayBox.Text = "";

processorBox.Text = "";

availBox.Text = "";

nameBox.Text = "";

IDBox.Text = "";

}

private void удалитьЗаписьToolStripMenuItem\_Click(object sender, EventArgs e)//Удаляет запись

{

foreach (DataGridViewRow i in DGV.SelectedRows)

DGV.Rows.Remove(i);

}

private void manualToolStripMenuItem\_Click(object sender, EventArgs e)//Инструкция к программме (Manual)

{

if (File.Exists("Manual\\Manual.txt"))

{

Process myProcess = null;

try

{myProcess = Process.Start("notepad.exe", "Manual\\Manual.txt"); }

catch

{MessageBox.Show("Программа notepad не найденна"); }

myProcess.Close();

}

else

{

MessageBox.Show("Файл с инструкцией не найден");

}

}

private void сохранитьToolStripMenuItem\_Click(object sender, EventArgs e)//Сохраняет список в файл (и создает копию)

{

RegistryKey currentUserKey = Registry.CurrentUser;

RegistryKey pathD = currentUserKey.OpenSubKey("pathD");

if (pathD == null)//Сохраняет название папки которую ввел пользователь

{

pathD = currentUserKey.CreateSubKey("pathD");

pathD.Close();

FormForSave f = new FormForSave();

f.ShowDialog();

SaveFileDialog saveFD = new SaveFileDialog();

Directory.CreateDirectory(Directory.GetCurrentDirectory().ToString() + String.Format("\\")+pathDirectory);

saveFD.InitialDirectory = Directory.GetCurrentDirectory().ToString() + String.Format("\\")+pathDirectory;

saveFD.DefaultExt = "xml";

if (saveFD.ShowDialog() == DialogResult.OK & saveFD.FileName.Length > 0)

{

fs = new FileStream(saveFD.FileName, FileMode.Create);

xs = new XmlSerializer(typeof(BindingList<Product>));

xs.Serialize(fs, prodLst);

//Создает директорию для backup-а и копирует туда файл при 1 запуске

Directory.CreateDirectory(Directory.GetCurrentDirectory() + "\\Backup");

File.Copy(saveFD.FileName, pathBackup);

CreateRegister(currentUserKey, saveFD);//Записываю в реестр полный путь с SaveFileDialog

}

CreateRegister(currentUserKey, pathDirectory);//Записываю в реестр имя папки для сохранения

fs.Close();

}

else//Если имя папки дял сохранение уже было введено

{

SaveFileDialog saveFD = new SaveFileDialog();

saveFD.InitialDirectory = path;

saveFD.DefaultExt = "xml";

if (saveFD.ShowDialog() == DialogResult.OK & saveFD.FileName.Length > 0)

{

fs = new FileStream(saveFD.FileName, FileMode.Create);

xs = new XmlSerializer(typeof(BindingList<Product>));

xs.Serialize(fs, prodLst);

fs.Close();

File.Copy(saveFD.FileName, pathBackup,true);//Создаю копию файла

CreateRegister(currentUserKey, saveFD); //Записываю в реестр полный путь с SaveFileDialog

}

}

}

public void CreateRegister(RegistryKey currentUserKey, SaveFileDialog saveFD) //Записываю в реестр полный путь с SaveFileDialog

{

currentUserKey = Registry.CurrentUser;

RegistryKey pathK = currentUserKey.CreateSubKey("path");

pathK.SetValue("path", saveFD.FileName);

pathK.Close();

}

public void CreateRegister(RegistryKey currentUserKey,string pathDirectory)//Записываю в реестр имя папки для сохранения

{

RegistryKey pathDD = currentUserKey.CreateSubKey("pathD");

pathDD.SetValue("pathD", pathDirectory.ToString());

pathDD.Close();

}

private void редактироватьToolStripMenuItem\_Click(object sender, EventArgs e)//Позаоляеи изменить информацию о продукте

{

flagEditShow = true;

i= DGV.CurrentRow.Index;

Edit\_Show f = new Edit\_Show();

f.ShowDialog();

}

private void всяИнформацтяToolStripMenuItem\_Click(object sender, EventArgs e)//Выводит всю информаумю о продукте

{

if (DGV.CurrentRow != null)

{

flagEditShow = false;

i = DGV.CurrentRow.Index;

Edit\_Show f = new Edit\_Show();

f.ShowDialog();

}

else

MessageBox.Show("Выберите конкретную ячейку для просмотра");

}

private void путьКПрограммеToolStripMenuItem\_Click(object sender, EventArgs e)//Выводит путь к программе

{

MessageBox.Show(Directory.GetCurrentDirectory(), "Путь к программе");

}

private void путьКСохраненномуФайлуToolStripMenuItem\_Click(object sender, EventArgs e)//Выводит путь к сохраненному файлу

{

MessageBox.Show(path, "Путь к сохраненному фалу");

}

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

{

MessageBox.Show("Создатель Матиив Роман\nВсе права защищены\nПлагиат преследуется по закону", "О программе", MessageBoxButtons.OK);

}//О программе

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

//using System.Xml.Serialization;

namespace Курсач

{

// [Serializable]

public class Product:Laptop

{

public bool Availability { get; set; }

public Product(int ID, string name, int RAM, int capacity, float monitorSize, string Proc, int battery, float price,bool availability) :base(ID,name,RAM,capacity,monitorSize,Proc,battery,price)

{

this.Availability = availability;

}

public Product() { }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсач

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Курсач

{

public class Users

{

public int password;

public string login;

public string salt;

public Users(string login,string password,string salt)

{

this.login = login;

this.salt = salt;

this.password = password.GetHashCode() + salt.GetHashCode();

}

public Users() { }

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсач

{

public partial class Registration : Form

{

bool flag = true;

Random rnd = new Random();

public Registration()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

flag = true;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Проверки

if (password.Text != passwordAgain.Text)

Error("Пароли не совпадают");

CheckString(login);

CheckString(password);

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

{

if (login.Text == CheckUsers.users[i].login)

Error("Пользователь с таким логином уже есть");

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

if(flag)//Если ошибок нет то регистрирует

{

CheckUsers.users.Add(new Users(login.Text, password.Text, GenerateSalt()));//Добавление

MessageBox.Show("Пользователь " + login.Text + " успешно зарегистрирован");

Close();

}

login.Text = "";

password.Text = "";

passwordAgain.Text = "";

}

public void CheckString(TextBox tb)//Проверка на полей

{

if (tb.Text.Trim().Length == 0)

{

Error("Все поля должны быть заполенны");

return;

}

int tmp1 = 0;

float tmp2;

if ((int.TryParse(tb.Text, out tmp1)) | (float.TryParse(tb.Text, out tmp2)))

{

Error("В логине и пароле должны присутстоввать буквы");

}

}

public string GenerateSalt()

{

string salt="";

string chars = "123456789!@#$%^&\*()\_?";

for (int i = 0; i < 3; i++)

{

salt+= chars.Substring(rnd.Next(0, chars.Length), 1);

}

return salt;

} // Возвращает соль

public void Error(string str)

{

if(flag)

{

flag = false;

MessageBox.Show(str);

}

} // Если есть ошибка выводит сообщение и ставит флаг

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Serialization;

using System.IO;

namespace Курсач

{

public partial class CheckUsers : Form

{

public static List<Users> users = new List<Users>();//Информация о пользователях

string path = Directory.GetCurrentDirectory() + "\\Users\\Users.xml";

XmlSerializer xs = new XmlSerializer(typeof(List<Users>));

FileStream fs;

public CheckUsers()

{

InitializeComponent();

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//Создаю файл пользователями если его нет

if(!File.Exists(path))

{

users.Add(new Users("admin", "admin", "123"));//Пользователь по умолчанию

fs = new FileStream(path, FileMode.Create);

xs.Serialize(fs, users);

fs.Close();

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

else

{

fs = new FileStream(path, FileMode.Open);

if (fs.Length != 0)

users = (List<Users>)xs.Deserialize(fs);

fs.Close();

}

}

private void EnterButton\_Click(object sender, EventArgs e)//Вход если пара логин-пароль существует

{

MainForm form = new MainForm();

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Проверяю есть ли такой пользователь

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

{

if (login.Text == users[i].login)

if (password.Text.GetHashCode() + users[i].salt.GetHashCode() == users[i].password)

{

this.Enabled = false;

this.Visible = false;

form.ShowDialog();

return;

}

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MessageBox.Show("Такого пользователя не существует");

login.Text = "";

password.Text = "";

}

private void CheckUsers\_FormClosing(object sender, FormClosingEventArgs e)//Осуществляет выход

{

Application.Exit();

}

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

{

Registration form = new Registration();

form.ShowDialog();

//Сериализует список пользователей

fs = new FileStream(path, FileMode.Create);

xs.Serialize(fs, users);

fs.Close();

} //Переходит на форму для регистрации

}

}