

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ  
РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Департамент анализа данных,  
принятия решений и финансовых технологий**

**Пояснительная записка к междисциплинарной курсовой работе**

на тему:

**Разработка  
Информационно-справочной системы "Принтеры"**

Выполнил:

студент группы ПИ18-2

Никитин Роман Андреевич

*Никитин*

(Подпись)

Телефон: 8-917-922-30-89

Научный руководитель:

доцент, к.т.н. Горелов Сергей Витальевич

---

(Подпись)

2020

# Оглавление

<b>Введение .....</b>	<b>2</b>
<b>1. Постановка задачи .....</b>	<b>2</b>
<b>2. Описание предметной области .....</b>	<b>4</b>
<b>3. Актуальность автоматизации .....</b>	<b>4</b>
<b>4. Описание программы .....</b>	<b>5</b>
4.1. Алгоритмические решения .....	5
4.2. Описание интерфейса программы .....	6
4.3. Состав приложения .....	13
<b>5. Назначение и состав классов программы .....</b>	<b>14</b>
5.1. Диаграмма классов .....	14
5.2. Класс Printer (товара магазина) .....	14
5.3. Служебные классы .....	15
5.4. Формы .....	17
<b>Заключение .....</b>	<b>17</b>
<b>Список литературы .....</b>	<b>18</b>
<b>Приложение. Исходный код программы .....</b>	<b>18</b>

# Введение

Целью курсовой работы является приобретение навыков в анализе предметной области и в разработке сложного Windows-приложения на объектно-ориентированном языке программирования C# в среде Visual Studio. Разработка приложения позволит улучшить мои знания этого языка, а также методов визуального проектирования и анализа предметной области.

Анализ предметной области позволяет разработать информационную модель задачи. Важным этапом этого анализа является выбор из предметной области наиболее важных объектов и процессов автоматизации, а также разработка новых IT-подходов и функций для решения проблем предметной области.

Второй целью курсовой работы, и не менее важной, является разработка пояснительной записки, отвечающей таким требованиям, как полнота, лаконичность, техническая корректность и т.д. Техническая документация является составной частью программного продукта и, следовательно, определяет его качество. Поэтому приобретение навыков в разработке технической документации, а в данном случае пояснительной записки, также является немаловажной задачей.

## 1. Постановка задачи

В соответствии с выбранной темой требуется разработать Windows-приложение, представляющее из себя ПО для просмотра информации, продажи и учёта принтеров (техники в целом) в магазине, возможности которого включают вход под разными аккаунтами и типами пользователей, отображение статистики по заказам, запись в историю всех действий как в персональные файлы, так и в один большой, общий, составление списка недостающей техники, которую необходимо купить, а так же динамическое изменение цены, рейтинга товара или полное его удаление/добавление нового. Так же для удобства пользователя предусмотрена сортировка техники по различным параметрам, настройка цвета окна, кнопок на нем и защита от неправильных действий.

Приложение должно содержать средства манипулирования объектами, сохранения их в файле и загрузки из файла. Должны быть реализованы сервисные операции с файлами клиентов – копирование и удаление.

*Требования:*

1. Разрабатываемое приложение должно представлять собой Windows приложение.

2. Должны быть разработаны пользовательские классы (один или несколько), описывающие объекты предметной области, и несколько форм пользовательского интерфейса.

3. Разработчик самостоятельно определяет интерфейс программы и ее функциональность, однако для получения максимальной оценки приложение

в обязательном порядке независимо от предметной области, указанной в задании, должно выполнять следующие операции:

- ☐ Отображать в сетках DataGridView данные предметной области:
- ☐ Для информационной модели, основанной на списках класса List<T>, на момент первого запуска программы допускается отсутствие файла. В этом случае списки объектов должны быть созданы в программе на этапе разработки.
- ☐ Для информационной модели, основанной на БД (Access, SQL Server), таблицы должны быть предварительно заполнены записями.

☐ Реализовать добавление в источник данных нового объекта, удаление объекта из источника данных, редактирование объекта источника данных.

☐ Реализовать фильтрацию записей источника данных, удовлетворяющих введенному пользователем сложному критерию.

☐ Реализовать сортировку записей источника данных, включая многоуровневую.

☐ Сохранять источник данных:

☐ для информационной модели, основанной на списках, в файле, используя XML-сериализацию (или Json).

☐ для информационной модели, основанной на таблицах БД, в базе данных.

☐ При запуске программы загрузить сохраненные данные из файла или базы данных.

☐ Используя меню или панель инструментов, вызвать приложение Блокнот для

просмотра справки о программе: файл Help.txt текущего каталога программы.

☐ Создать пункт меню «Об авторе» с выводом соответствующей информации (MessageBox).

☐ Разработать несколько полезных пользователю функций для отображения статистических данных, например, средних, максимальных или минимальных

значений, данных для построения гистограммы или графика и т.п.

3. Список должен быть реализован в виде коллекции, например, динамического массива типа List<T> или BindingList<T>. 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.

4. Программа должна быть читабельной и содержать полезные комментарии.

## **2. Описание предметной области**

В разрабатываемом приложении предметной областью автоматизации является магазин по продаже принтеров (техники в целом). С точки зрения заказчика, магазин является источником дохода.

Деятельность бизнеса организована следующим образом: продавец может продать необходимое устройство покупателю, докупить недостающее, подсказать информацию о той или иной технике. Возможности программы достаточно обширны и многообразны. Есть все минимально необходимое для успешного ведения бизнеса и учета товаров на складе.

Каждый вид деятельности магазина может быть обеспечен современными IT-технологиями. Однако продажа товаров(принтеров)- один из наиболее важных аспектов существования магазина как такового и самое первое, с чем сталкиваются продавцы в работе, поэтому этот вид деятельности необходимо автоматизировать в первую очередь.

Продавцы и администратор- важнейшая ячейка в иерархическом устройстве магазина, поэтому программа максимально на них ориентирована. Каждый сотрудник важен, а вам важно обеспечить ему удобство в работе. Больше не нужно возиться с таблицами, заполнять вручную и проводить реинвентаризацию, все можно сделать в пару кликов!

Обрабатывая историю продаж и действий сотрудников магазина, можно получить различные статистические данные, направленные на улучшение качества обслуживания клиентов и принятие организационно-финансовых решений руководством магазина.

## **3. Актуальность автоматизации**

Клиенты магазина являются основными источниками дохода, поэтому возможность быстро и удобно с ними коммуницировать- залог успеха любого бизнеса. Учет позволит следить за рентабельностью, а, следовательно, и прибыльностью магазина. Учет товаров позволит получать статистику по наличию их на складе.

Поскольку программа позволит сократить непроизводительные, ручные затраты работников отеля по учету товаров, поднять уровень и скорость обслуживания и доходы магазина, автоматизация этой предметной области

является **актуальной** задачей. Для решения этой актуальной задачи разработано приложение, предназначенное для учета товаров и продаж в магазине. Также программа предназначена для выполнения ряда сервисных функций по работе с Базой Данных, реализованной с помощью MDB. Программа позволяет вводить, изменять, сохранять и удалять товары и пользователей из файла, а так же полностью откатывать все данные в программе.

## **4. Описание программы**

### **4.1. Алгоритмические решения**

Хранение списка клиентов, принтеров и информации о них организовано в базе данных MDB. Файл отображается на список типа `List<Class>`. Пользователь может динамически изменять файл с товарами, сортировать его, дополнять, а администратор имеет полный контроль над всем, включая и пользователей. Программа позволяет сохранять любые действия над товарами в базе данных и при новой загрузке данные не теряются и подгружаются.

Чтобы пользователь не заполнял все изменения вручную предусмотрено автоматическое сохранение, а при критических неисправностях в БД- полный откат системы до дефолтных значений.

Программа пишет логи (историю действий) для каждого пользователя отдельно и для всех вместе, если файлов для записей логов еще не существует- создает новые.

В любой момент пользователь может создать новый аккаунт в случае утери пароля или попросить у администратора его сменить/ напомнить. Программа предоставляет для этого специальные возможности в окне регистрации или окне возможностей администратора соответственно. Файл с логом будет создан автоматически при осуществлении первых важных действий нового пользователя, например, таких, как продажа товара или пополнение склада новой техникой.

Каждый товар имеет свои характеристики (поля), по которым можно сортировать или проводить аналитику, а так же советовать клиенту то, что ему необходимо.

Для отбора товаров в меню отображается ряд полей, создающих условие для сортировки. Отобраны будут все товары, удовлетворяющие условиям сортировки (одноуровневой).

Для работы с таблицей товаров используется интерфейс элемента DataGridView. Базовая функциональность DataGridView поддерживается специальными обработчиками событий таблицы, в которых осуществляется проверка полей таблицы на допустимость значений.

В программе реализована сортировка товаров по любому атрибуту. Для сортировки необходимо перейти в меню сверху и кликнуть на необходимое условие. Сортировка позволяет расположить строки с искомым признаком в нужном вам порядке. То есть сортировка в какой-то степени может заменить отбор товаров по одному заданному критерию. Многокритериальный отбор сортировка заменить не может.

Программа является задокументированной, пункт меню Помощь выдает полное описание программы.

В программе реализовано два режима работы: режим продавца и режим администратора, обладающего полным доступом. В режим полного доступа, в котором разрешаются все запрограммированные функции, можно попасть, только введя логин и пароль.

Для сохранения списка работников и товаров используется MDB. Разработан класс и множество методов, избавляющие разработчика от знания деталей ввода-вывода, а так же предохраняющих от введения некорректных данных. Использование этого класса повышает надежность программы и производительность разработчика.

В программе реализована оптимизация вывода списка товаров на диск. Если за время сеанса не произошло изменения какого-либо товара из списка, либо всего списка в целом (редактирование, удаление, добавление), то при завершении программы такой список не нуждается в сохранении в файле.

Для хранения объектов в оперативной памяти выбран список типа List<T>. В пользу такого решения выступает наличие метода сортировки Sort и OrderBy выполняющего упорядочивание объектов списка на месте. У списка BindingList<T> такого метода нет, а его метод расширения OrderBy создает новый список, что, на самом деле, не принципиально.

## **4.2. Описание интерфейса программы**

В приложении для реализации интерфейса используется 9 элементов управления и компонентов:

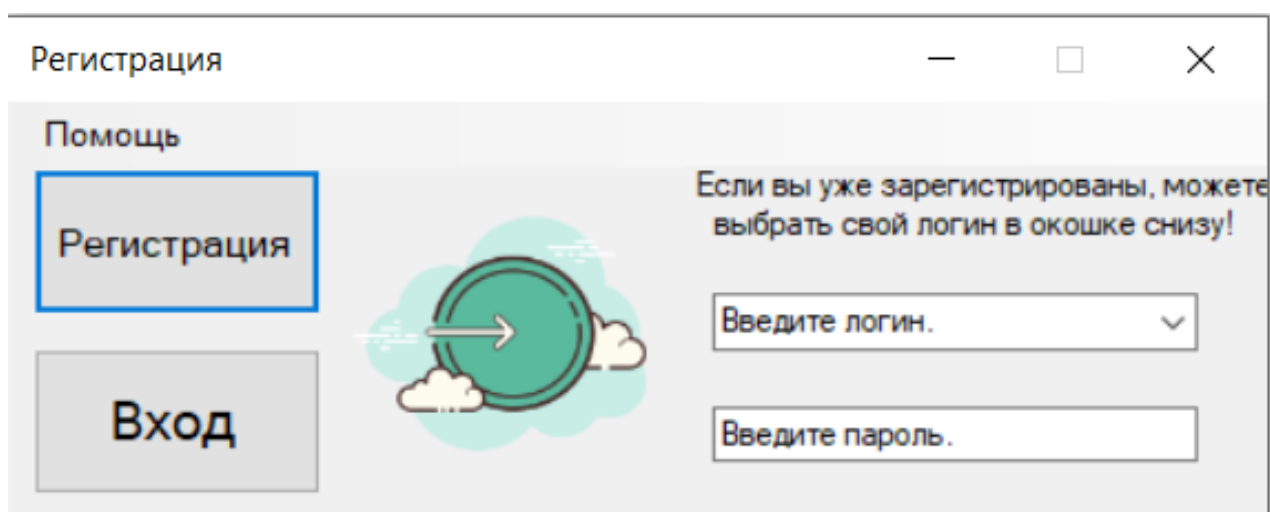
- DataGridView;
- Button;
- TextBox;
- Label;
- MessageBox;
- MenuStrip;
- OpenFileDialog;
- BindingSource

- PictureBox(для красоты, а так же реализации небольшой, но приятной «Пасхалки»).

Наиболее сложным элементом управления является таблица (сетка) DataGridView. Загрузка в таблицу записей клиентов осуществляется благодаря привязке таблицы к коллекции List<T> объектов типа Class. Привязка осуществляется с использованием посредника BindingSource.

Элемент управления DataGridView отображает список товаров, а так же список товаров и расширенной информации о них.

Запуская Windows приложение, мы попадаем в форму, в которой требуется зарегистрироваться для определения статуса сотрудника (для режима работы программы): Пользователь или Администратор». Уровень доступа «Пользователь» не позволяет каким-либо образом полностью удалить товар из списка или изменить о нем ключевую информацию, а так же не располагает функциями, требующими уровень доступа «Администратор». Уровень доступа «Администратор» не имеет каких-либо ограничений, поэтому войти в этот режим можно, только введя правильные логин и пароль от аккаунта, статус которого «admin» (рис. 4.1). Этот режим – для администраторов магазина.



*Рис. 4.1. Выбор режима работы.*

Если Вы- новый сотрудник или забыли логин и пароль от вашего аккаунта, вы можете зарегистрироваться, введя новые логин и пароль, нажав кнопку «Регистрация». Далее просто нажать «Вход» и Вы В ЛЮБОМ СЛУЧАЕ ВОЙДЕТЕ С ПРАВАМИ ПОЛЬЗОВАТЕЛЯ!

Введя правильные данные и нажав кнопку «Вход», мы попадаем в главную форму. В зависимости от типа аккаунта формы разнятся. Окно аккаунта с уровнем доступа «Пользователь(user)» на рис. 4.2.



Магазин Принтеров

Сортировка
Помощь

Меню сотрудника

Добавить принтер на склад

Продать принтер

Вывести список недостающего


Закупка недостающего

Посмотреть статистику

Записать статистику

Вывести текст из файла.

Информация о принтерах



Изменение цвета окна

Изменение цвета кнопок

Выполнить

	Name	Model	Type	Price	Rate	Kol_vo
►	SAMSUNG	SL-M4020ND/XEV	Laser-Poroshok	12000	9,1	10
	Samsung	Xpress_M2070	Laser	9990	7,7	4
	Xerox	Phaser_3020	Laser	7299,99	6,2	7
	HP	LaserJet_P1005	Laser	12490	7,9	20
	1	1	1	1	1	10

**Рис. 4.2.** Главное окно пользователя.

Окно аккаунта с уровнем доступа «Администратор(admin)» на рис. 4.3.

Магазин Принтеров

Сортировка Помощь

## Меню администратора

**Добавить товар в Price\_list**

Посмотреть статистику

Продать принтер

Записать статистику

Вывести список необходимого

Вывести текст из файла.

Закупка недостающего

Информация о принтерах

Список пользователей


Добавить нового пользователя

Удалить пользователя

Сменить статус аккаунта

Откатить все данные

Удалить принтер



Изменение цвета окна

Изменение цвета кнопок

Изменить информацию о принтере

Выполнить

	Name	Model	Type	Price	Rate	Kol_vo
▶	SAMSUNG	SL-M4020ND/XEV	Laser-Poroshok	12000	9,1	10
	Samsung	Xpress_M2070	Laser	9990	7,7	4
	Xerox	Phaser_3020	Laser	7299,99	6,2	7
	HP	LaserJet_P1005	Laser	12490	7,9	20
	1	1	1	1	1	10

**Рис. 4.3.** Главное окно администратора.

Для сохранения нового списка товаров в файле пользователь не должен делать ничего. Любые изменения, прошедшие необходимые проверки сохраняются автоматически. В случае ошибочно введенных данных пользователь получит сообщение от системы. При любом запуске программы автоматически подгружается полный список всей базы данных.

Для добавления товара по соответствующей кнопке заполняются textBoxЫ, в которых отображаются необходимые критерии для заполнения. Если вы не собираетесь работать с новыми данными(принтерами), вы можете выбрать необходимый в DataGridView снизу необходимый вам и поля автоматически перезаполнятся.

В программе реализована сортировка товаров по любому атрибуту. Для сортировки необходимо щелкнуть мышкой в меню, в разделе «Сортировка» и выбрать необходимую вам.

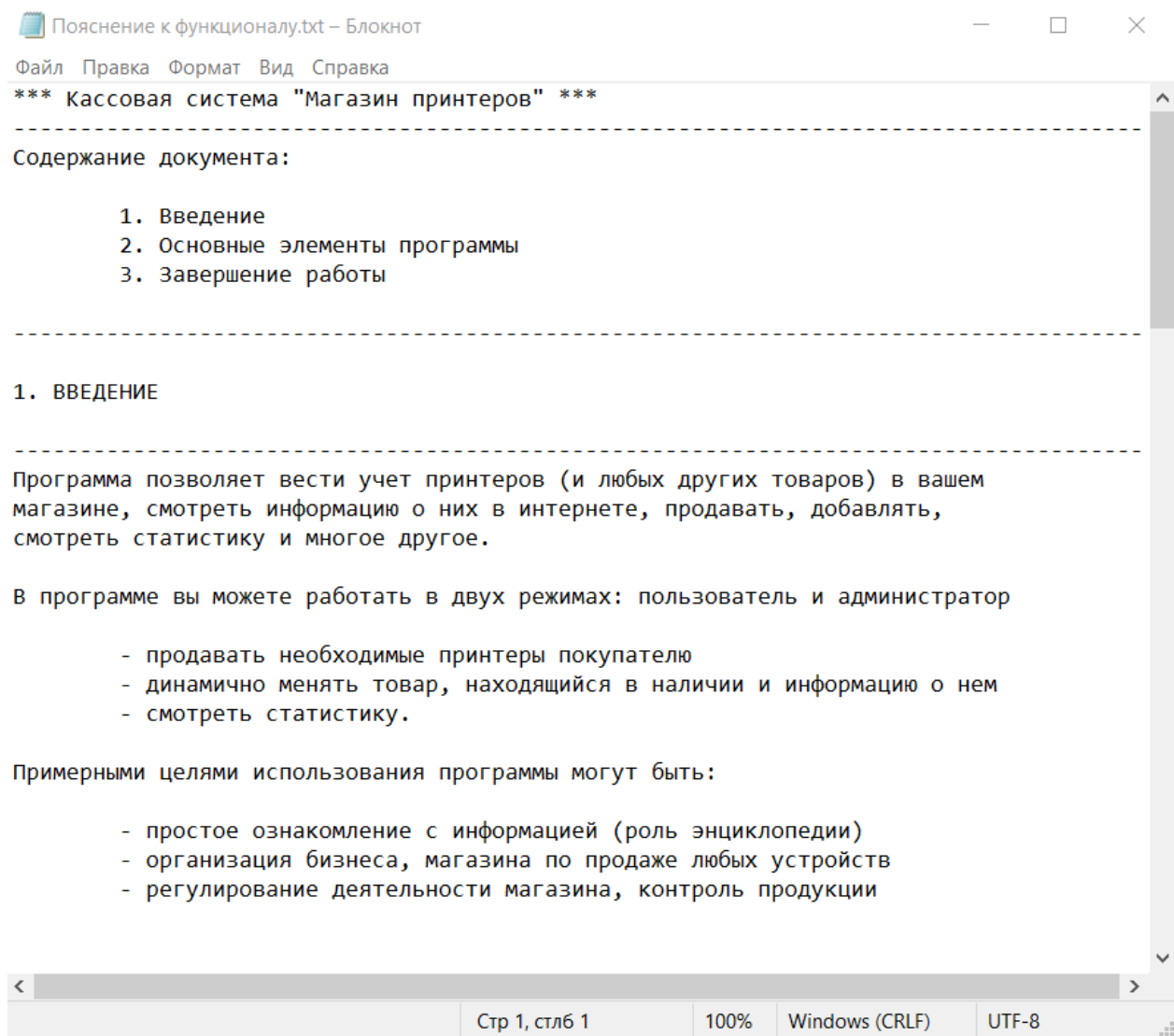
Для того, чтобы посмотреть подробную информацию о принтере(товаре) в интернете, вы можете перейти по вкладке «Информация о принтерах», где предложены таких функции, как просмотреть информацию, добавить информацию, а так же кастомизация внешнего вида формы. Вы можете выбрать нужный вам принтер в ComboBoxах или в списке, предложенном снизу и если о нем уже есть информация- в ваш буфер обмена будет скопирована короткая ссылка для перехода. Если информации нет- в ваш буфер обмена будет скопирована ссылка с сервисом сокращения ссылок и

предложено добавить информацию о принтере(товаре). Не забывайте, любые ваши действия записываются, ведь Большой брат следит за Вами! Так же если вы хотите посмотреть информацию о создателе и его научном руководителе- нажмите на соответствующую кнопку. Так же стоит отметить, что доступ к данной форме есть как у аккаунтов с уровнем доступа «Пользователь»(user), так и «Администратор»(admin). Рис.4.4.

	Name	Model	Info
▶	SAMSUNG	SL-M4020ND/XEV	<a href="https://is.gd/LqzmN8">https://is.gd/LqzmN8</a>
	Samsung	Xpress_M2070	<a href="https://is.gd/JCj12l">https://is.gd/JCj12l</a>
	Xerox	Phaser_3020	<a href="https://is.gd/vSN4xU">https://is.gd/vSN4xU</a>
	HP	LaserJet_P1005	<a href="https://is.gd/3yUSrh">https://is.gd/3yUSrh</a>
	1	1	1

**Рис. 4.4.** Окно информации о принтерах(товаре).

Для того, чтобы получить инструкцию по использованию приложения, нужно на панели меню нажать на «Помощь»: будет запущено приложение Блокнот с описанием программы (см. рис. 4.5).

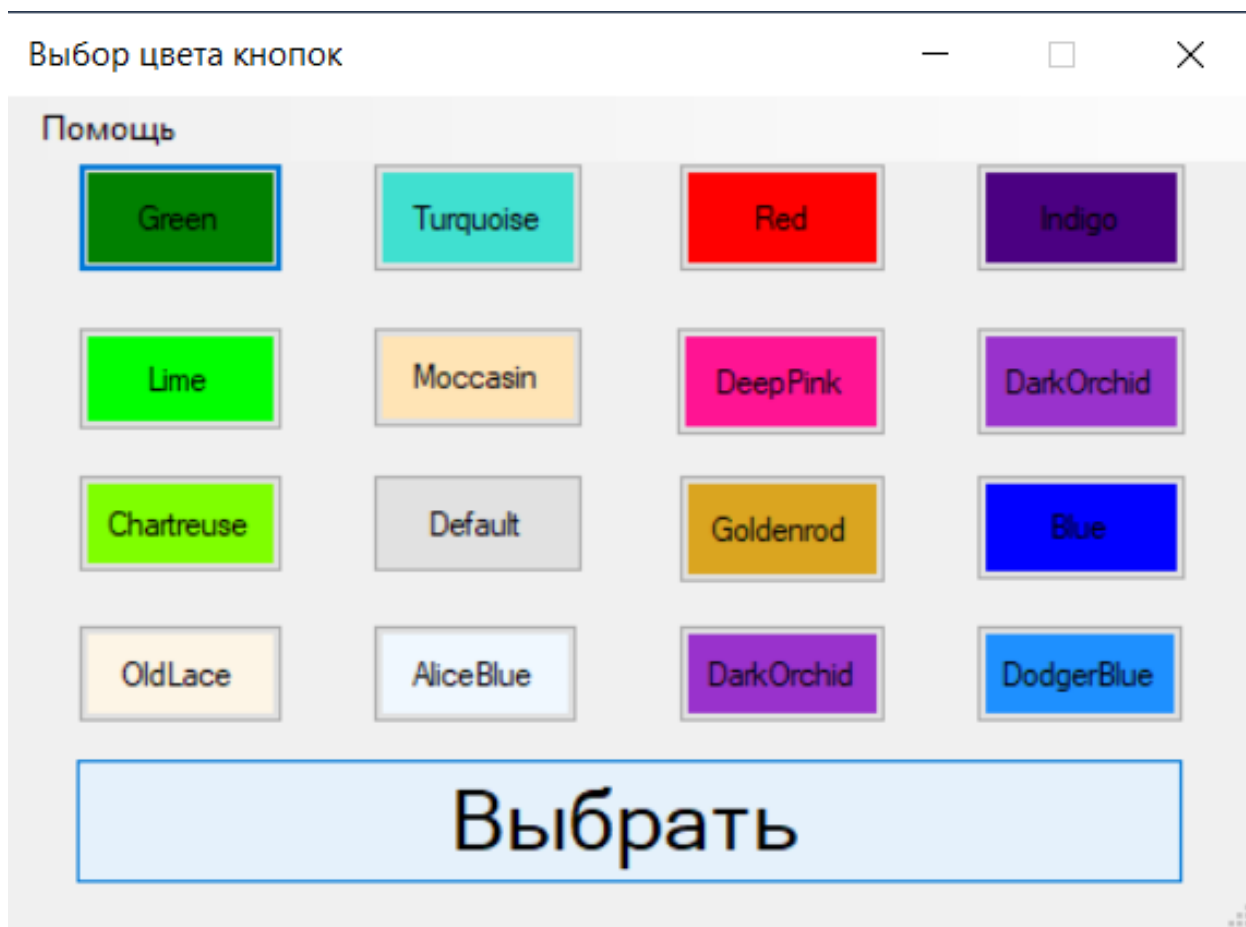


*Рис. 4.5. Окно Блокнота со справкой.*

При желании пользователь может настроить формы под свой вкус и цвет, в каждом окне предусмотрены отдельные настройки цвета фона и кнопок для создания своей кастомной формы внешнего вида. Рис 4.6 и Рис 4.7.



*Рис. 4.6. Окно изменения цвета фона формы.*



*Рис. 4.7. Окно изменения цвета кнопок.*

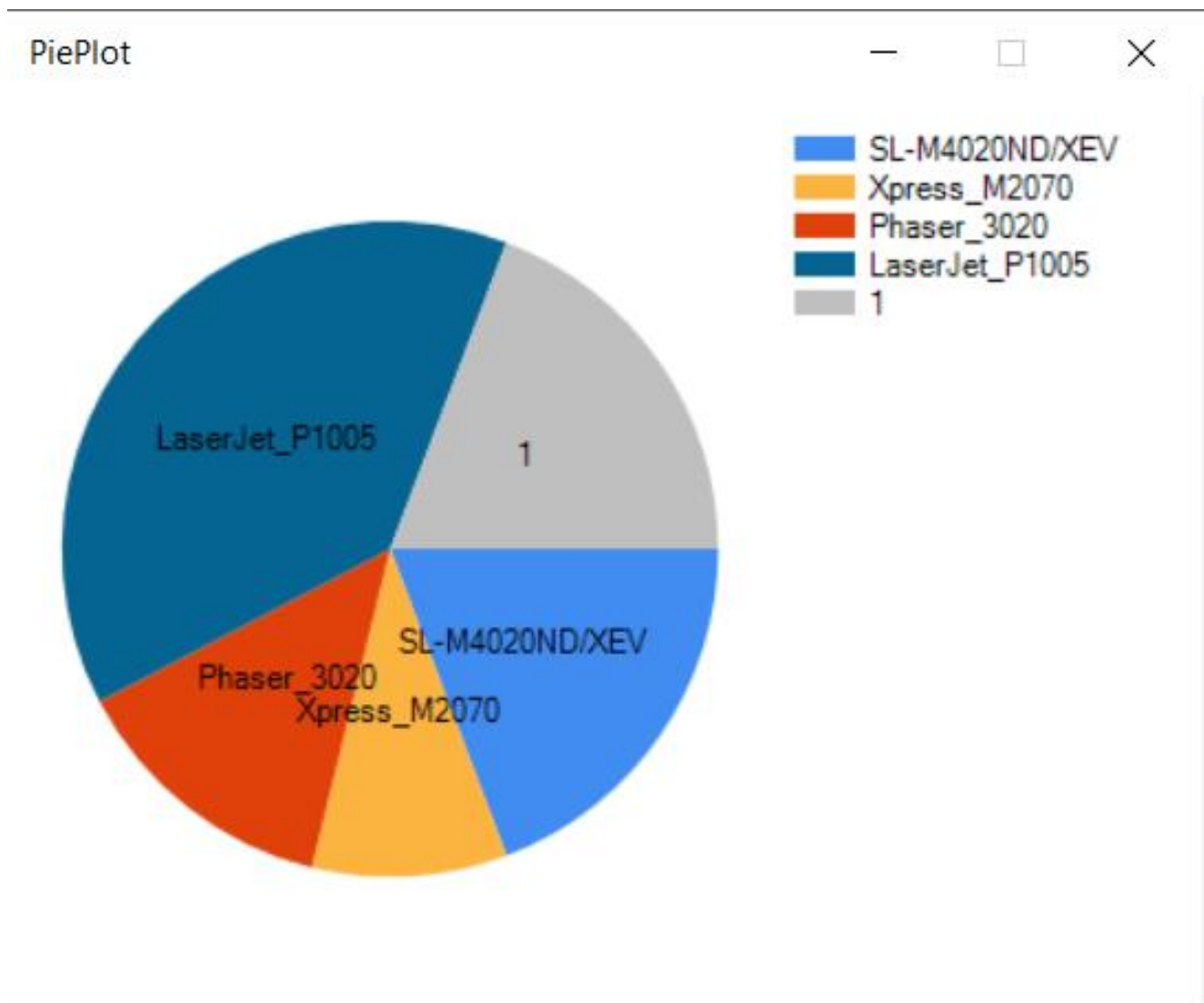


Рис. 4.8 Окно с графиком.

### 4.3. Состав приложения

В состав приложения входят следующие файлы и папки:

- Shop.sln – программа;
- WindowsFormsApp1– папка с со всеми файлами программы;
  - ✓ КР-Пояснительная записка– пояснительная записка к программе;
- База Данных–база клиентов и товаров:
  - ✓ Printer\_shop.mdb – большая бд, содержащая в себе все основные записи программы.
  - ✓ Так же предусмотрены дублиры в текстовом виде для удобного просмотра без подключения к бд.

## 5. Назначение и состав классов программы

### 5.1. Диаграмма классов

Разрабатываемая программа состоит из нескольких классов, содержащих множество полей, методов и свойств. Взаимосвязь основных классов приложения можно увидеть на диаграмме, изображенной на рис. 5.1. Стрелками изображены связи между классами и формами..

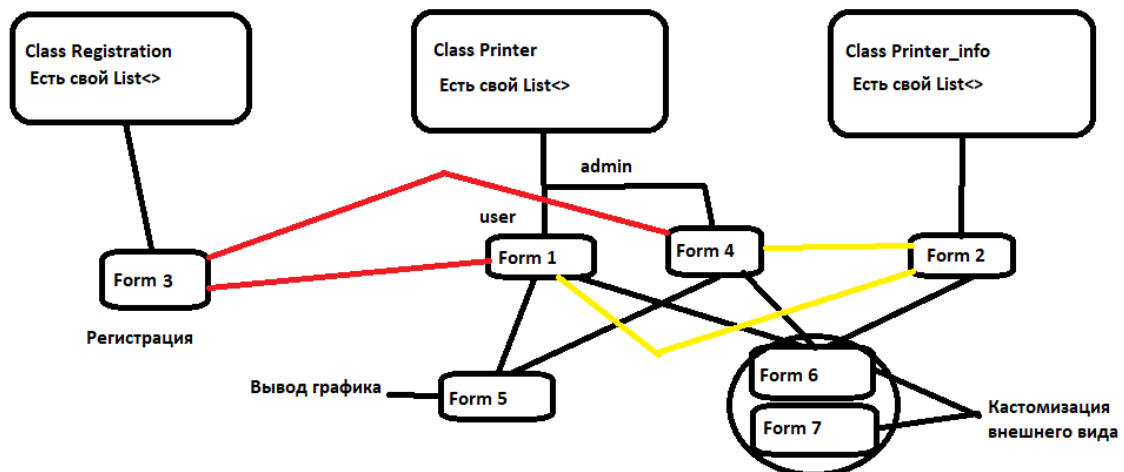


Рис. 5.1. Диаграмма классов

### 5.2. Класс Printer (товара магазина)

- ✓ Name – свойство, позволяющее возвращать и задавать фирму-производитель принтера(товара);
- ✓ Model – свойство, позволяющее возвращать и задавать модель принтера(товара);
- ✓ Type – свойство, позволяющее возвращать и задавать тип принтера(товара);
- ✓ Price– свойство, позволяющее возвращать и задавать цену принтера(товара);
- ✓ Rate – свойство, позволяющее возвращать и задавать рейтинг принтера(товара);
- ✓ Kol\_vo– свойство, позволяющее возвращать и задавать количество принтеров(товара) на складе;
- ✓ Printer(*параметры*) – параметрический конструктор, осуществляющий инициализацию свойств класса.

### 5.3. Служебные классы

К служебным классам относятся:

- **Registration** – класс для работы с разделом Регистрации (Form 3)  
Все открытые поля и методы класса **статические**.

*Поля:*

**string Name** – логин аккаунта, в последствии используется для написания логов.

**string Password** – пароль от аккаунта.

**string Status** – статус аккаунта (пользователь или администратор).

*Методы:*

**public static void Read\_users\_from\_mdb()** - Читает из базы данных уже зарегистрированных ранее пользователей.

**public static void Add\_user\_to\_mdb(string user, string password, string status)** Записывает пользователя в бд со всеми пользователями.

**public static void Delete\_user\_from\_mdb(string user, string password, string status)** – Удаляет пользователя из базы данных

**public static string Enter(string name, string password, string status)**  
Функция для входа в систему, проверяет логин, пароль, статус аккаунта и пропускает в систему.

**public static string Register(string name, string password, string status = "user")** Функция для регистрации нового пользователя из меню с регистрацией, по умолчанию статус – user.

**public static string Login(string name, string password)** Вход в систему, если введенные вами данные уже есть в списке.

#### **class Printer\_info**

*Поля:*

**string Name** – фирма производитель принтера(товара)

**string Model** – модель принтера(товара)

**string Info** – информация (ссылка на интернет- ресурс)

**public static void Write\_info\_to\_file()** - Записывает в файл информацию о принтерах

**public static string Change\_info\_about\_printer(string name, string model, string info)** – изменяет информацию о принтере в бд

**public static void Add\_info(string Name, string Model, string Info)** - Функция добавления информации о принтере

**public static void Load\_info\_from\_mdb()** Считывает информацию о принтерах и добавляет в список для последующей работы

**class Printer** Класс, по факту, самый главный в программе. Принтеры и практически вся работа с ними.

*Поля:*



**string Name** – фирма производитель принтера(товара)

**string Model** – модель принтера(товара)

**string Type** – тип принтера(товара)

**double Price** - цена

**double Rate** - рейтинг

**int Kol\_vo** – количество на складе

**public static string Change\_status\_of\_user**(string name, string password, string status, string new\_status) – функция для изменения типа(статуса) пользователя. В классе Printer потому что логически подходящий класс (Register) используется только на форме регистрации (Form3), и я решил не выносить его за пределы этой области, так что благодаря этому действию у меня каждый класс отвечает за отдельную форму.

**public static string Change\_price\_list\_items**(string name, string model, string type = "", double price = 0, double rate = 0, int kol\_vo = 0) - Функция изменения полей с данными о принтерах

**public static string Delete\_user**(string name, string password, string status) – функция удаления пользователя. В этом классе по той же причине, что и предыдущая.

**public static string Add\_user**(string name, string password, string status) – функция добавления пользователя. . В этом классе по той же причине, что и предыдущие.

**public static string Change\_price\_list\_items**(string name, string model, string type = "", double price = 0, double rate = 0, int kol\_vo = 0) – функция изменения данных в списке товаров.

**public static string Delete\_printer\_from\_price\_list**(string name, string model) – функция удаления принтера из списка товаров.

**public static void Reload()** – функция отката всех данных в проекте.

**public static void Read\_file\_with\_printers**(string filename) – функция для прочтения всех данных из файла и добавления в список. Нужна только для бэкапа данных.

**public static void Add\_Printer**(string Name, string Model, string Type, double Price, double Rate, int Kol\_vo) – функция для добавления нового принтера в список и в БД, либо для добавления существующего на склад.

**public static void Write\_log**(string username, string phrase, string name = "", string model = "", string kol\_vo = "") – функция записи логов ключевых действий пользователей.

**public static string Read\_from\_File**(string filename) функция чтения данных из файла и возвращение их из функции. **public static void Load\_printers\_from\_mdb()** - Функция для загрузки всех принтеров из бд.

**public static void Check\_list**(string Name, string Model, int Kol\_vo, double Price) – функция записи недостающих товаров в специальный файл.

**public static string Sell\_Printer**(string Name, string Model, int Kol\_vo, string Type = "", double Price = 0, double Rate = 0) – функция продажи принтера(товара), если он есть в списке.

**public static void purchase()** – функция закупки всех недостающих принтеров(товаров)

**public static void statistics()** – функция записи статистики по данным.

**public static string check\_statistics()** – функция вывода статистики на экран.

- **Program** – класс, содержащий метод Main, который позволяет запустить программу, открывая главную форму Form1.

## 5.4. Формы

Ниже представлены пользовательские классы, наследующие базовую функциональность от класса Form.

- **Form1** – класс, содержащий все основные обработчики событий и методы сотрудника. Рис. 4.2
- **Form2** – класс, содержащий всю дополнительную информацию проекта. Рис. 4.4
- **Form3** – класс, осуществляющий вход в основную форму (см. рис. 4.1).
- **Form4** – класс, содержащий все основные обработчики событий и методы администратора. Рис. 4.3
- **Form5** – класс, содержащий график, отображающий соотношение количества принтеров на складе (pieplot). Рис. 4.8
- **Form6** – класс, содержащий в себе метод выбора цвета фона окна, из которого был вызван. Рис. 4.6
- **Form7** – класс, содержащий в себе метод выбора цвета кнопок окна, из которого был вызван. Рис. 4.7

## Заключение

Разработав Windows приложение, был выполнен ряд поставленных задач. Справочно-информационная система написана на языке программирования C# с использованием классов, методов и свойств. Для реализации некоторых задач потребовалось обратиться к объектам операционной системы: системный реестр, файловая система, процессы.

Программа может модернизироваться и обновляться. К примеру, может быть расширен класс принтеров путем добавления новых объектов.

Программа может быть расширена статистическими и финансовыми функциями.

## Список литературы

1. Горелов С.В., Волков А.Г. Разработка Windows-приложений. Часть 1. Учебное пособие. Образовательный портал Финансового университета. 2018.
2. Горелов С.В. Разработка Windows-приложений. Часть 2. Учебное пособие. Образовательный портал Финансового университета. 2018.
3. Г. Шилдт. Полный справочник по C#. - М.: «Вильямс», 2004.
4. Официальный сайт Microsoft: [Интернет-ресурс]. URL: <https://msdn.microsoft.com org>.
5. Сайт <https://sergeygorelov.wixsite.com/projects>
6. Отличный канал на youtube «Выполняем домашние задания и курсовые работы». Моего преподавателя Горелова С.В
7. Учебник в 2 томах: Современные технологии программирования. Разработка Windows-приложений на языке C# Автор- Горелов С.В.

*Никитин*

## Приложение. Исходный код программы

### Файл Program.cs:

```
namespace WindowsFormsApp1
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form3());
        }
    }
}
```

### Файл Class1.cs

```
namespace WindowsFormsApp1
{
    class Registration /*Класс для регистрации нового пользователя*/
    {
```

```

        public string Name { get; set; }
        public string Password { get; set; }
        public string Status { get; set; }

        public static List<Registration> Registration_list = new List<Registration>();
/*Список со всеми пользователями*/
        public static string connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=printer_shop.mdb;";
        public Registration(string _name, string _password, string _status)
        {
            Name = _name;
            Password = _password;
            Status = _status;
        }
        public Registration() { }
        public static void Read_users_from_mdb() /*Функция читает файл с пользователями и
добавляет их в список*/
        {
            Registration_list.Clear();
            string query = "SELECT * FROM users";
            OleDbConnection myConnection = new OleDbConnection(connectionString);
            // открываем соединение с БД
            myConnection.Open();
            // создаем объект OleDbCommand для выполнения запроса к БД MS Access
            OleDbCommand command = new OleDbCommand(query, myConnection);
            // получаем объект OleDbDataReader для чтения табличного результата запроса
SELECT
            OleDbDataReader reader = command.ExecuteReader();
            // в цикле построчно читаем ответ от БД
            while (reader.Read())
            {
                Registration user = new Registration(reader[1].ToString(),
reader[2].ToString(), reader[3].ToString());
                Registration_list.Add(user);
            }
            // закрываем OleDbDataReader
            reader.Close();
            myConnection.Close();
        }

        public static void Add_user_to_mdb(string user, string password, string status)
        {
            OleDbConnection myConnection = new OleDbConnection(connectionString);
            myConnection.Open();
            OleDbTransaction trans = myConnection.BeginTransaction();
            string sql1 = $"SELECT MAX(id) FROM users";
            OleDbCommand cmd1 = new OleDbCommand(sql1, myConnection, trans);
            string x = cmd1.ExecuteScalar().ToString();
            int y = int.Parse(x) + 1;
            string text = string.Format("{0}, {1}, {2}, {3}", y, user, password, status);
            string sql2 = $"INSERT INTO users VALUES ({y}, '{user}', '{password}',
'{status}')}";
            OleDbCommand cmd2 = new OleDbCommand(sql2, myConnection, trans);
            cmd2.ExecuteNonQuery();
            trans.Commit();
            myConnection.Close();
        }
        public static void Delete_user_from_mdb(string user, string password, string
status)
        {
            OleDbConnection myConnection = new OleDbConnection(connectionString);
            myConnection.Open();
            OleDbTransaction trans = myConnection.BeginTransaction();

```

```

        string sql2 = $"DELETE FROM users WHERE user = '{user}' and password =
'{password}' and status = '{status}'";
        OleDbCommand cmd2 = new OleDbCommand(sql2, myConnection, trans);
        cmd2.ExecuteNonQuery();
        trans.Commit();
        myConnection.Close();
        for (int i = 0; i < Registration.Registration_list.Count; i++)
        {
            if (Registration.Registration_list[i].Name == user &
Registration.Registration_list[i].Password == password &
Registration.Registration_list[i].Status == status)
            {
                Registration.Registration_list.Remove(Registration.Registration_list[i]);
                Printer.Write_log(Form3.name_for_log, "удалил пользователя", user,
password, status);
                StreamWriter writer = new StreamWriter("users.txt", false,
Encoding.UTF8);
                for (int k = 0; k < Registration_list.Count; k++)
                {
                    if (k > 0)
                        writer.Write('\n' + Registration_list[k].Name + ' ' +
Registration_list[k].Password + ' ' + Registration_list[k].Status);
                    else
                        writer.Write(Registration_list[k].Name + ' ' +
Registration_list[k].Password + ' ' + Registration_list[k].Status);
                }
                writer.Close();
            }
        }
    }
    public static string Enter(string name, string password, string status) /*Функция
для входа в систему, проверяет логин, пароль, статус
аккаунта и пропускает в систему*/
    {
        int flag = 0;
        for (int i = 0; i < Registration_list.Count; i++)
        {
            if (Registration_list[i].Name == name & Registration_list[i].Password ==
password)
            {
                flag += 1;
            }
        }
        if (flag != 0)
        {
            if (status == "user")
                return "user";
            else return "admin";
        }
        else return "not_in_list";
    }
    public static string Register(string name, string password, string status =
"user") /*Функция для регистрации нового пользователя из
меню с регистрацией, по умолчанию статус - user*/
    {
        int flag = 0;
        for (int i = 0; i < Registration_list.Count; i++)
        {
            if (Registration_list[i].Name == name)
            {

```

```

        flag += 1;
    }
}
if (flag == 0)
{
    Registration person = new Registration(name, password, status);
    Registration_list.Add(person);
    Add_user_to_mdb(name, password, status);
    StreamWriter writer = new StreamWriter("users.txt", false,
Encoding.UTF8);
    for (int k = 0; k < Registration_list.Count; k++)
    {
        if (k > 0)
            writer.Write('\n' + Registration_list[k].Name + ' ' +
Registration_list[k].Password + ' ' + Registration_list[k].Status);
        else
            writer.Write(Registration_list[k].Name + ' ' +
Registration_list[k].Password + ' ' + Registration_list[k].Status);
    }
    writer.Close();
    return "added_to_list";
}
else return "already_in_list";

}

public static string Login(string name, string password) /*Вход в систему, если
введенные вами данные уже есть в списке*/
{
    Read_users_from_mdb();
    int flag = 0;
    string status = "";
    for (int i = 0; i < Registration_list.Count; i++)
    {
        if (Registration_list[i].Name == name & Registration_list[i].Password ==
password)
        {
            status = Registration_list[i].Status;
            flag += 1;
        }
    }
    if (flag != 0)
        return Enter(name, password, status);
    else
        return "not_in_list";
}

}

class Printer_info /*Класс информации о принтерах. Ранее, на зачете был
родительским.*/
{
    public string Name { get; set; }
    public string Model { get; set; }
    public string Info { get; set; }

    public static List<Printer_info> Printer_info_list = new List<Printer_info>();
    /*Список принтеров и информации о них*/
    public static string connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=printer_shop.mdb;";
    public Printer_info(string _name, string _model, string _info)
    {
        Name = _name;
        Model = _model;
        Info = _info;
    }
}

```

```

public Printer_info() { }
/*Изменение информации о принтере в списке и в файле и их перезапись*/
public static string Change_info_about_printer(string name, string model, string
info)
{
    for (int i = 0; i < Printer_info_list.Count; i++)
    {
        if (Printer_info_list[i].Name == name & Printer_info_list[i].Model ==
model)
        {
            Printer_info_list[i].Info = info;
            OleDbConnection myConnection = new OleDbConnection(connectString);
            myConnection.Open();
            OleDbTransaction trans = myConnection.BeginTransaction();
            string sql1 = $"UPDATE printer_info SET info = '{info}' WHERE (brand
= '{name}' and model = '{model}')";
            OleDbCommand cmd1 = new OleDbCommand(sql1, myConnection, trans);
            cmd1.ExecuteNonQuery();
            trans.Commit();
            myConnection.Close();
            return "Успешно!";
        }
    }
    return ":";
}
/*Прочтение информации о принтерах из файла и заполнение ею списка*/
public static void Read_info_from_file(string filename) /*Считывает информацию о
принтерах и добавляет в список для последующей работы*/
{
    string[] arStr = File.ReadAllLines(filename);
    for (int i = 0; i < arStr.Length; i++)
    {
        string Name = arStr[i].Split(' ')[0];
        string Model = arStr[i].Split(' ')[1];
        string Info = arStr[i].Split(' ')[2];
        Printer_info printer_info = new Printer_info(Name, Model, Info);
        Printer_info_list.Add(printer_info);
    }
}
public static void Load_info_from_mdb()
{
    Printer_info_list.Clear();
    string query = "SELECT * FROM printer_info";
    OleDbConnection myConnection = new OleDbConnection(connectString);
    // открываем соединение с БД
    myConnection.Open();
    // создаем объект OleDbCommand для выполнения запроса к БД MS Access
    OleDbCommand command = new OleDbCommand(query, myConnection);
    // получаем объект OleDbDataReader для чтения табличного результата запроса
SELECT
    OleDbDataReader reader = command.ExecuteReader();
    // в цикле построчно читаем ответ от БД
    while (reader.Read())
    {
        Printer_info user = new Printer_info(reader[1].ToString(),
reader[2].ToString(), reader[3].ToString());
        Printer_info_list.Add(user);
    }
    // закрываем OleDbDataReader
    reader.Close();
    myConnection.Close();
}
}

```

```

        public static void Add_info(string Name, string Model, string Info) /*Функция
добавления информации о принтере*/
        {
            int flag = 0;
            for (int i = 0; i < Printer_info_list.Count; i++)
            {
                if (Printer_info_list[i].Name == Name & Printer_info_list[i].Model ==
Model)
                    flag += 1;
            }
            if (flag == 0)
            {
                Printer_info printer_info = new Printer_info(Name, Model, Info);
                Printer_info_list.Add(printer_info);
                OleDbConnection myConnection = new OleDbConnection(connectString);
                myConnection.Open();
                OleDbTransaction trans = myConnection.BeginTransaction();
                string sql1 = $"SELECT MAX(id) FROM printer_info";
                OleDbCommand cmd1 = new OleDbCommand(sql1, myConnection, trans);
                string x = cmd1.ExecuteScalar().ToString();
                int y = int.Parse(x) + 1;
                string sql2 = $"INSERT INTO printer_info VALUES ({y}, '{Name}',
'{Model}', '{Info}')";
                OleDbCommand cmd2 = new OleDbCommand(sql2, myConnection, trans);
                cmd2.ExecuteNonQuery();
                trans.Commit();
                myConnection.Close();
            }
        }
    }

    class Printer /*Класс, по факту, самый главный в программе. Принтеры и практически
вся работа с ними*/
    {
        public string Name { get; set; }
        public string Model { get; set; }
        public string Type { get; set; }
        public double Price { get; set; }
        public double Rate { get; set; }
        public int Kol_vo { get; set; }

        public static List<Printer> Printer_list = new List<Printer>(); /*Список всех
принтеров и их характеристик*/
        public static string connectString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=printer_shop.mdb;";
        public Printer(string _name, string _model, string _type = "", double _price = 0,
double _rate = 0, int _kol_vo = 0)
        {
            Name = _name;
            Model = _model;
            Type = _type;
            Price = _price;
            Rate = _rate;
            Kol_vo = _kol_vo;
        }
    }

    /*Функция изменения статуса пользователя. Не знаю, почему здесь, по факту- должна
быть в регистрации.*/
    public static string Change_status_of_user(string name, string password, string
status, string new_status)
    {
        Registration.Read_users_from_mdb();
        for (int i = 0; i < Registration.Registration_list.Count; i++)
        {

```



```

        if (Registration.Registration_list[i].Name == name &
Registration.Registration_list[i].Password == password &
Registration.Registration_list[i].Status == status)
        {
            Registration.Registration_list[i].Status = new_status;
Printer.Write_log(Form3.name_for_log, "изменил статус пользователя",
name, password, status);
OleDbConnection myConnection = new OleDbConnection(connectString);
myConnection.Open();
OleDbTransaction trans = myConnection.BeginTransaction();
string sql1 = $"UPDATE users SET status = '{new_status}' WHERE (user
= '{name}' and password = '{password}')"
OleDbCommand cmd1 = new OleDbCommand(sql1, myConnection, trans);
cmd1.ExecuteNonQuery();
trans.Commit();
myConnection.Close();
return "Успешно";
        }
    }
    return "Неуспешно!";
}
/*Функция изменения данных в списке товаров*/
public static string Change_price_list_items(string name, string model, string
type = "", double price = 0, double rate = 0, int kol_vo = 0)
{
    for (int i = 0; i < Printer_list.Count; i++)
    {
        if (Printer_list[i].Name == name & Printer_list[i].Model == model)
        {
            int flag = 0;
            if (type != "" & type.Split().Length == 1)
            {
                Printer_list[i].Type = type;
                flag += 1;
            }

            if (price != 0 & price >= 0 & price <= 10000000)
            {
                Printer_list[i].Price = price;
                flag += 1;
            }
            if (rate != 0 & rate >= 0 & rate <= 10)
            {
                flag += 1;
                Printer_list[i].Rate = rate;
            }
            if (kol_vo != 0 & kol_vo >= 0)
            {
                flag += 1;
                Printer_list[i].Kol_vo = kol_vo;
            }
            if (flag != 0)
            {
                OleDbConnection myConnection = new
OleDbConnection(connectString);
myConnection.Open();
OleDbTransaction trans = myConnection.BeginTransaction();
string sql1 = $"UPDATE price_list SET type =
'{{Printer_list[i].Type}}', price = {{Printer_list[i].Price}}, rate = {{Printer_list[i].Rate}},
kol_vo = {{Printer_list[i].Kol_vo}} where brand = '{{name}}' and model = '{{model}}'";
OleDbCommand cmd1 = new OleDbCommand(sql1, myConnection, trans);
cmd1.ExecuteNonQuery();
trans.Commit();
            }
        }
    }
}

```

```

        myConnection.Close();
        Write_log(Form3.name_for_log, "изменил информацию о принтере",
name, model);
        return "Успешно!";
    }
    else return "Нечего изменять!";
}
}
return "Неуспешно!";
}
/*Функция для удаления принтера из списка товаров, если он есть в списке*/
public static string Delete_printer_from_price_list(string name, string model)
{
    for (int i = 0; i < Printer_list.Count; i++)
    {
        if (Printer_list[i].Name == name & Printer_list[i].Model == model)
        {
            Printer_list.Remove(Printer_list[i]);
            OleDbConnection myConnection = new OleDbConnection(connectString);
            myConnection.Open();
            OleDbTransaction trans = myConnection.BeginTransaction();
            string sql2 = $"DELETE FROM price_list WHERE brand = '{name}' and
model = '{model}'";
            OleDbCommand cmd2 = new OleDbCommand(sql2, myConnection, trans);
            cmd2.ExecuteNonQuery();
            trans.Commit();
            myConnection.Close();
            Write_log(Form3.name_for_log, "удалил принтер", name, model);
            return "Успешно!";
        }
    }
    return "Неуспешно!";
}
/*Функция удаления пользователя, если он есть в списке*/
public static string Delete_user(string name, string password, string status)
{
    for (int i = 0; i < Registration.Registration_list.Count; i++)
    {
        if (Registration.Registration_list[i].Name == name &
Registration.Registration_list[i].Password == password &
Registration.Registration_list[i].Status == status)
        {
            Printer.Write_log(Form3.name_for_log, "удалил пользователя", name,
password, status);
            Registration.Delete_user_from_mdb(name, password, status);
            return "Успешно!";
        }
    }
    return "Неуспешно!";
}
/*Функция добавления нового пользателя, если его нет в списке*/
public static string Add_user(string name, string password, string status)
{
    Registration.Read_users_from_mdb();
    int flag = 0;
    for (int i = 0; i < Registration.Registration_list.Count; i++)
    {
        if (Registration.Registration_list[i].Name == name)
        {
            flag += 1;
        }
    }
    if (flag == 0)

```

```

    {
        string answer = Registration.Register(name, password, status);
        if (answer == "added_to_list")
        {
            Printer.Write_log(Form3.name_for_log, "добавил нового пользователя",
name, password, status);
            return "added_to_list";
        }
        else return "already_in_list";
    }
    else return "already_in_list";
}
/*Функция для отката всех данных из файла, который никак и никогда не
задействуется в других функциях*/
public static void Reload()
{
    Printer.Write_log(Form3.name_for_log, "Откатил все данные программы");
    Printer_list.Clear();
    Printer_info.Printer_info_list.Clear();
    Printer_list.Clear();
    Printer_info.Load_info_from_mdb();
    Read_file_with_printers("reload.txt");
    StreamWriter writer1 = new StreamWriter("check_list.txt", false,
Encoding.UTF8);
    writer1.Write("");
    writer1.Close();
    StreamWriter writer2 = new StreamWriter("stat_kol_vo.txt", false,
Encoding.UTF8);
    writer2.Write("");
    writer2.Close();
    StreamWriter writer3 = new StreamWriter("stat_price.txt", false,
Encoding.UTF8);
    writer3.Write("");
    writer3.Close();
    OleDbConnection myConnection = new OleDbConnection(connectString);
    myConnection.Open();
    OleDbTransaction trans = myConnection.BeginTransaction();
    string sql1 = "DELETE FROM price_list";
    OleDbCommand cmd1 = new OleDbCommand(sql1, myConnection, trans);
    cmd1.ExecuteNonQuery();
    string sql2 = "DELETE FROM printer_info";
    OleDbCommand cmd2 = new OleDbCommand(sql2, myConnection, trans);
    cmd2.ExecuteNonQuery();
    trans.Commit();
    myConnection.Close();
    int x = 0;
    string sql3 = "";
    string sql4 = "";

    for (int i = 0; i<Printer_list.Count; i++)
    {
        OleDbConnection my_new_Connection = new OleDbConnection(connectString);
        my_new_Connection.Open();
        OleDbTransaction transact = my_new_Connection.BeginTransaction();
        sql3 = $"INSERT INTO price_list VALUES ({x}, '{Printer_list[i].Name}',
'{{Printer_list[i].Model}}', '{{Printer_list[i].Type}}', '{{Printer_list[i].Price}}',
'{{Printer_list[i].Rate}}', '{{Printer_list[i].Kol_vo}}')";
        OleDbCommand cmd3 = new OleDbCommand(sql3, my_new_Connection, transact);
        cmd3.ExecuteNonQuery();
        transact.Commit();
        my_new_Connection.Close();
        x += 1;
    }
}

```

```

        x = 0;
        for (int i = 0; i < Printer_info.Printer_info_list.Count; i++)
        {
            OleDbConnection my_new_Connection_1 = new OleDbConnection(connectString);
            my_new_Connection_1.Open();
            OleDbTransaction transact_1 = my_new_Connection_1.BeginTransaction();
            sql4 = $"INSERT INTO printer_info VALUES ({x},
'{Printer_info.Printer_info_list[i].Name}', '{Printer_info.Printer_info_list[i].Model}',
'{Printer_info.Printer_info_list[i].Info}')";
            OleDbCommand cmd4 = new OleDbCommand(sql4, my_new_Connection_1,
transact_1);

            cmd4.ExecuteNonQuery();
            transact_1.Commit();
            x += 1;
            my_new_Connection_1.Close();
        }
    }
    /*Нужна только для бэкапа. Функция для прочтения файла с информацией о принтерах
и заполнения ей списка*/
    public static void Read_file_with_printers(string filename)
    {
        int flag = 0;
        string[] arStr = File.ReadAllLines(filename);
        for (int i = 0; i < arStr.Length; i++)
        {
            string Name = arStr[i].Split(' ')[0];
            string Model = arStr[i].Split(' ')[1];
            string Type = arStr[i].Split(' ')[2];
            double Price = double.Parse(arStr[i].Split(' ')[3]);
            double Rate = double.Parse(arStr[i].Split(' ')[4]);
            int Kol_vo = int.Parse(arStr[i].Split(' ')[5]);
            for (int j = 0; j < Printer_list.Count; j++)
            {
                if (Printer_list[j].Name == Name & Printer_list[j].Model == Model)
                {
                    flag += 1;
                }
            }
            if (flag == 0)
            {
                Printer printer = new Printer(Name, Model, Type, Price, Rate,
Kol_vo);
                Printer_list.Add(printer);
            }
        }
    }
    public static void Load_printers_from_mdb()
    {
        Printer_list.Clear();
        string query = "SELECT * FROM price_list";
        OleDbConnection myConnection = new OleDbConnection(connectString);
        // открываем соединение с БД
        myConnection.Open();
        // создаем объект OleDbCommand для выполнения запроса к БД MS Access
        OleDbCommand command = new OleDbCommand(query, myConnection);
        // получаем объект OleDbDataReader для чтения табличного результата запроса
        SELECT
        OleDbDataReader reader = command.ExecuteReader();
        // в цикле построчно читаем ответ от БД
        while (reader.Read())
        {

```

```

        Printer print = new Printer(reader[1].ToString(), reader[2].ToString(),
reader[3].ToString(), double.Parse(reader[4].ToString()),
double.Parse(reader[5].ToString()), int.Parse(reader[6].ToString()));
        Printer_list.Add(print);
    }
    // закрываем OleDbDataReader
    reader.Close();
    myConnection.Close();
}

/*Функция для добавления в список принтера, если его нет либо для добавления
количества принтеров, если уже есть в списке*/
public static void Add_Printer(string Name, string Model, string Type, double
Price, double Rate, int Kol_vo)
{
    int flag = 0;
    /*Добавление количества, когда принтер уже есть в списке*/
    for (int i = 0; i < Printer_list.Count; i++)
    {
        if (Printer_list[i].Name == Name & Printer_list[i].Model == Model)
        {
            Printer_list[i].Kol_vo += Kol_vo;
            Write_log(Form3.name_for_log, "добавил принтеры на склад", Name,
Model, Kol_vo.ToString());
            flag += 1;
            OleDbConnection myConnection = new OleDbConnection(connectString);
            myConnection.Open();
            OleDbTransaction trans = myConnection.BeginTransaction();
            string sql2 = $"UPDATE price_list SET kol_vo =
'{{Printer_list[i].Kol_vo}}' WHERE brand = '{{Name}}' and model = '{{Model}}'";
            OleDbCommand cmd2 = new OleDbCommand(sql2, myConnection, trans);
            cmd2.ExecuteNonQuery();
            trans.Commit();
            myConnection.Close();
        }
    }
    /*Добавление нового принтера*/
    if (flag == 0 & Type != " " & Price != 0 & Rate >= 0 & Rate <= 10)
    {
        Printer printer = new Printer(Name, Model, Type, Price, Rate, Kol_vo);
        Printer_list.Add(printer);
        Write_log(Form3.name_for_log, "добавил принтеры на склад", Name, Model,
Kol_vo.ToString());
        OleDbConnection myConnection = new OleDbConnection(connectString);
        myConnection.Open();
        OleDbTransaction trans = myConnection.BeginTransaction();
        string sql1 = $"SELECT MAX(id) FROM price_list";
        OleDbCommand cmd1 = new OleDbCommand(sql1, myConnection, trans);
        string x = cmd1.ExecuteScalar().ToString();
        int y = int.Parse(x) + 1;
        string sql2 = $"INSERT INTO price_list VALUES ({{y}}, '{{Name}}', '{{Model}}',
'{{Type}}', '{{Price}}', '{{Rate}}', '{{Kol_vo}}')";
        OleDbCommand cmd2 = new OleDbCommand(sql2, myConnection, trans);
        cmd2.ExecuteNonQuery();
        trans.Commit();
        myConnection.Close();
    }
}

/*Запись логов для каждого пользователя в отдельности общего для всех*/
public static void Write_log(string username, string phrase, string name = "",
string model = "", string kol_vo = "")
{

```

```

        if (name != "" & model != "" & kol_vo != "")
        {
            StreamWriter writer = new StreamWriter(username + "_log.txt", true,
Encoding.UTF8);
            writer.WriteLine("{0} {1} {2} {3} {4} {5}", DateTime.Now, username,
phrase, name, model, kol_vo);
            writer.Close();
            StreamWriter writer1 = new StreamWriter("big_log.txt", true,
Encoding.UTF8);
            writer1.WriteLine("{0} {1} {2} {3} {4} {5}", DateTime.Now, username,
phrase, name, model, kol_vo);
            writer1.Close();
        }
        else
        {
            StreamWriter writer = new StreamWriter(username + "_log.txt", true,
Encoding.UTF8);
            writer.WriteLine("{0} {1} {2}", DateTime.Now, username, phrase);
            writer.Close();
            StreamWriter writer1 = new StreamWriter("big_log.txt", true,
Encoding.UTF8);
            writer1.WriteLine("{0} {1} {2}", DateTime.Now, username, phrase);
            writer1.Close();
        }
    }
    /*Чтение данных из файла и возвращение их из функции*/
    public static string Read_from_File(string filename)
    {
        if (filename != "users.txt" & filename != "big_log.txt")
        {
            string text = "";
            StreamReader reader = new StreamReader(filename);
            string[] arStr = File.ReadAllLines(filename);
            for (int i = 0; i < arStr.Length; i++)
            {
                text = text + arStr[i] + "\n";
            }
            reader.Close();
            return text;
        }
        else return "Не получится, шалунишка!";
    }
    /*Функция записи недостающих товаров в специальный файл для их хранения*/
    public static void Check_list(string Name, string Model, int Kol_vo, double
Price)
    {
        StreamWriter writer = new StreamWriter("check_list.txt", true,
Encoding.UTF8);
        writer.WriteLine("{0} {1} {2} {3}", Name, Model, Kol_vo, Price);
        writer.Close();
    }
    /*Функция продажи товара(принтера), если такой есть в списке. Реализованы все
проверки, чтобы не крашился.*/
    public static string Sell_Printer(string Name, string Model, int Kol_vo, string
Type = "", double Price = 0, double Rate = 0)
    {
        double total = 0;
        for (int i = 0; i < Printer_list.Count; i++)
        {
            if (Printer_list[i].Name == Name & Printer_list[i].Model == Model)
            {
                if (Printer_list[i].Kol_vo >= Kol_vo)
                {

```

```

        total += Kol_vo * Printer_list[i].Price;
        Printer_list[i].Kol_vo -= Kol_vo;
        OleDbConnection myConnection = new
OleDbConnection(connectString);
        myConnection.Open();
        OleDbTransaction trans = myConnection.BeginTransaction();
        string sql2 = $"UPDATE price_list SET kol_vo =
'{Printer_list[i].Kol_vo}' WHERE brand = '{Name}' and model = '{Model}'";
        OleDbCommand cmd2 = new OleDbCommand(sql2, myConnection, trans);
        cmd2.ExecuteNonQuery();
        trans.Commit();
        myConnection.Close();
        Write_log(Form3.name_for_log, "продал принтеров", Name, Model,
total.ToString());
        return string.Format("Продан/ы принтер/ы модели {0} на сумму
{1}!", Name, total);
    }
    else
    {
        total += Printer_list[i].Kol_vo * Printer_list[i].Price;
        double total_money_needs = (Kol_vo - Printer_list[i].Kol_vo) *
Printer_list[i].Price;
        Write_log(Form3.name_for_log, "продал принтеров", Name, Model,
(Printer_list[i].Kol_vo).ToString());
        string deficit = string.Format("Не хватает принтеров {0}, модели
{3}, магазину нужно докупить ещё как минимум {1} на сумму {2}!", Name, Kol_vo -
Printer_list[i].Kol_vo, total_money_needs, Printer_list[i].Model);
        string sold = string.Format("В итоге принтеров фирмы: {1} ,
модели: {2} продано на сумму: {0}", total, Name, Printer_list[i].Model);
        Check_list(Name, Printer_list[i].Model, Kol_vo -
Printer_list[i].Kol_vo, total_money_needs);
        Printer_list[i].Kol_vo = 0;
        OleDbConnection myConnection = new
OleDbConnection(connectString);
        myConnection.Open();
        OleDbTransaction trans = myConnection.BeginTransaction();
        string sql2 = $"UPDATE price_list SET kol_vo =
'{Printer_list[i].Kol_vo}' WHERE brand = '{Name}' and model = '{Model}'";
        OleDbCommand cmd2 = new OleDbCommand(sql2, myConnection, trans);
        cmd2.ExecuteNonQuery();
        trans.Commit();
        myConnection.Close();
        return (deficit + '\n' + sold);
    }
}

return string.Format("Нет принтера фирмы {0} модели {1}!", Name, Model);
}
/*Функция закупки всех недостающих принтеров из списка.*/
public static void purchase()
{
    StreamReader reader = new StreamReader("check_list.txt", Encoding.UTF8);
    string[] arStr1 = File.ReadAllLines("check_list.txt", Encoding.UTF8);
    reader.Close();
    for (int i = 0; i < arStr1.Length; i++)
    {
        string Name = arStr1[i].Split(' ')[0];
        string Model = arStr1[i].Split(' ')[1];
        int Kol_vo = int.Parse(arStr1[i].Split(' ')[2]);
        Add_Printer(Name, Model, " ", 0, 0, Kol_vo);
    }
}

```

```

        StreamWriter writer = new StreamWriter("check_list.txt", false,
Encoding.UTF8);
        writer.Write("");
        writer.Close();
    }
    /*Функция для записи статистики по данным*/
    public static void statistics()
    {
        int max_kol_vo = 0;
        double max_price = 0;
        int k_p = 0;
        double tot_price = 0;
        int tot_kol_vo = 0;
        double mean = 0;
        for (int i = 0; i < Printer_list.Count; i++)
        {
            k_p += 1;
            tot_price += Printer_list[i].Price;
            tot_kol_vo += Printer_list[i].Kol_vo;
            if (max_price < Printer_list[i].Price)
                max_price = Printer_list[i].Price;
            if (max_kol_vo < Printer_list[i].Kol_vo)
                max_kol_vo = Printer_list[i].Kol_vo;
        }
        mean = tot_price / k_p;
        StreamWriter writer = new StreamWriter("stat_price.txt", true,
Encoding.UTF8);
        writer.Write("Средняя цена за принтер: " + mean.ToString() + "\n" + "Всего
принтеров на складе:" + tot_kol_vo.ToString() + "\n" +
        "На складе принтеров на сумму: " + tot_price.ToString() + "\n" + "Всего
различных моделей принтеров: " + k_p.ToString() + "\n" +
        "Самый дорогой принтер стоит: " + max_price.ToString() + "\n" +
        "Наибольшее количество принтеров одной модели: " + max_kol_vo.ToString());
        for (int i = 0; i < Printer_list.Count; i++)
        {
            writer.Write("Цена принтера на момент времени " + " " + DateTime.Now + "
" + Printer_list[i].Name + " " + Printer_list[i].Model + " " + Printer_list[i].Price +
"\n");
        }
        writer.Close();
        StreamWriter writer1 = new StreamWriter("stat_kol_vo.txt", true,
Encoding.UTF8);
        for (int i = 0; i < Printer_list.Count; i++)
        {
            writer1.Write("Количество принтеров на момент времени " + " " +
DateTime.Now + " " + Printer_list[i].Name + " " + Printer_list[i].Model + " " +
Printer_list[i].Kol_vo + "\n");
        }
        writer1.Close();
    }
    /*Функция для вывода статистики на экран*/
    public static string check_statistics()
    {
        int max_kol_vo = 0;
        double max_price = 0;
        int k_p = 0;
        double tot_price = 0;
        int tot_kol_vo = 0;
        double mean = 0;
        for (int i = 0; i < Printer_list.Count; i++)
        {
            k_p += 1;
            tot_price += Printer_list[i].Price;

```



```

        tot_kol_vo += Printer_list[i].Kol_vo;
        if (max_price < Printer_list[i].Price)
            max_price = Printer_list[i].Price;
        if (max_kol_vo < Printer_list[i].Kol_vo)
            max_kol_vo = Printer_list[i].Kol_vo;
    }
    mean = tot_price / k_p;
    return ("Средняя цена за принтер: " + mean.ToString() + "\n" + "Всего
принтеров на складе:" + tot_kol_vo.ToString() + "\n" +
        "На складе принтеров на сумму: " + tot_price.ToString() + "\n" + "Всего
различных моделей принтеров: " + k_p.ToString() + "\n" +
        "Самый дорогой принтер стоит: " + max_price.ToString() + "\n" +
        "Наибольшее количество принтеров одной модели: " + max_kol_vo.ToString());
    }
}

```

## Файл Form1.cs

```

{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            pictureBox1.Image = new Bitmap("10.png");
            Printer.Load_printers_from_mdb();
            printerBindingSource.DataSource = Printer.Printer_list;
        }
        int flag = 0;

        private void clear()
        {
            textBox4.Text = "";
            textBox5.Text = "";
            textBox6.Text = "";
            textBox7.Text = "";
            textBox8.Text = "";
            textBox9.Text = "";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            clear();
            textBox4.Text = "Введите фирму принтера";
            textBox5.Text = "Введите модель принтера";
            textBox6.Text = "Введите тип принтера";
            textBox7.Text = "Введите цену принтера";
            textBox8.Text = "Введите рейтинг принтера";
            textBox9.Text = "Введите количество принтеров";
            flag = 1;
        }
        /*За заполнение textboxes отвечают кнопки, на которых описаны возможности
        функций, для выполнения же используется отдельная кнопка "Выполнить"*/
        private void button2_Click(object sender, EventArgs e)
        {
            if (flag == 0)
                MessageBox.Show("Сначала выберите действие из меню!");
        }
    }
}

```

```

else if (flag == 1)
{
    if (textBox4.Text != "Введите фирму принтера" & textBox5.Text != "Введите
модель принтера" & textBox6.Text != "Введите тип принтера" &
        textBox7.Text != "Введите цену принтера" & textBox8.Text != "Введите
рейтинг принтера" & textBox9.Text != "Введите количество принтеров" &
        textBox4.Text.Split().Length == 1 & textBox5.Text.Split().Length == 1
& textBox6.Text.Split().Length == 1 & textBox7.Text.Split().Length == 1 &
        textBox8.Text.Split().Length == 1 & textBox9.Text.Split().Length ==
1)
    {
        try
        {
            {
                Printer.Add_Printer(textBox4.Text, textBox5.Text,
textBox6.Text, double.Parse(textBox7.Text), double.Parse(textBox8.Text),
int.Parse(textBox9.Text));
                clear();
                flag = 0;
            }
            else MessageBox.Show("Вы вводите данные не правильно! Возможно,
вам стоит поменять знаки пробела на нижнее подчеркивание '_', вводить наименования длиной
менее 20 символов или заполнить все предложенные поля!");
        }
        catch
        {
            MessageBox.Show("Вы вводите данные не правильно! Возможно, вам
стоит поменять знаки пробела на нижнее подчеркивание '_', вводить наименования длиной
менее 20 символов или заполнить все предложенные поля!");
        }
    }
    else MessageBox.Show("Вы вводите данные не правильно! Возможно, вам стоит
поменять знаки пробела на нижнее подчеркивание '_', вводить наименования длиной менее 20
символов или заполнить все предложенные поля!");
}
else if (flag == 2)
{
    if (textBox4.Text != "Введите фирму принтера" & textBox5.Text != "Введите
модель принтера" & textBox9.Text != "Введите количество принтеров")
    {
        try
        {
            {
                if (int.Parse(textBox9.Text) > 0)
                {
                    MessageBox.Show(Printer.Sell_Printer(textBox4.Text,
textBox5.Text, int.Parse(textBox9.Text)));
                    clear();
                    flag = 0;
                }
                else MessageBox.Show("Вы не можете продать меньше, чем 1
принтер!");
            }
            catch
            {
                MessageBox.Show("Вы неправильно ввели данные!");
            }
        }
    }
}
else if (flag == 3)
{
    if (textBox4.Text != "Введите фирму принтера")
    {

```

```

        try
        {
            MessageBox.Show(Printer.Read_from_File(textBox4.Text));
            clear();
            flag = 0;
        }
        catch
        {
            MessageBox.Show("Вы неправильно ввели данные!");
        }
    }
    printerBindingSource.ResetBindings(false);
}
private void button4_Click(object sender, EventArgs e)
{
    clear();
    Form2 form2 = new Form2();
    form2.Show();
}

private void button3_Click(object sender, EventArgs e)
{
    flag = 3;
    clear();
    textBox4.Text = "Введите название файла для прочтения";
}

private void button6_Click(object sender, EventArgs e)
{
    flag = 2;
    clear();
    textBox4.Text = "Введите фирму принтера";
    textBox5.Text = "Введите модель принтера";
    textBox9.Text = "Введите количество принтеров";
}

private void button7_Click(object sender, EventArgs e)
{
    string text = Printer.Read_from_File("check_list.txt");
    if (text == "")
    {
        MessageBox.Show("Пока ещё нечего закупать!");
    }
    else
    {
        Printer.purchase();
        MessageBox.Show("Закупка прошла успешно!");
        Printer.Write_log(Form3.name_for_log, "произвел закупку недостающих
принтеров");
        printerBindingSource.ResetBindings(false);
        clear();
    }
}

private void button8_Click(object sender, EventArgs e)
{
    clear();
    string text = Printer.Read_from_File("check_list.txt");
    if (text == "")
    {
        MessageBox.Show("Тут пока что пусто!");
    }
}

```

```

        else MessageBox.Show(text);
    }

    private void button9_Click(object sender, EventArgs e)
    {
        Printer.statistics();
        MessageBox.Show("Статистика по ценам принтеров записана в файлы
stat_price.txt и stat_kol_vo.txt");
        clear();
    }

    private void button10_Click(object sender, EventArgs e)
    {
        MessageBox.Show(Printer.check_statistics());
        Form5 form5 = new Form5();
        form5.Show();
        clear();
    }

    private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        if (flag == 1 | flag == 2)
        {
            textBox4.Text = dataGridView1[0,
dataGridView1.CurrentRow.Index].Value.ToString();
            textBox5.Text = dataGridView1[1,
dataGridView1.CurrentRow.Index].Value.ToString();
            textBox6.Text = dataGridView1[2,
dataGridView1.CurrentRow.Index].Value.ToString();
            textBox7.Text = dataGridView1[3,
dataGridView1.CurrentRow.Index].Value.ToString();
            textBox8.Text = dataGridView1[4,
dataGridView1.CurrentRow.Index].Value.ToString();
        }
    }
    /*Открываю формы в режиме диалога, потому что умею*/
    private void button11_Click(object sender, EventArgs e)
    {
        Form6 form6 = new Form6();
        form6.Owner = this;
        form6.ShowDialog();
    }

    private void button12_Click(object sender, EventArgs e)
    {
        Form7 form7 = new Form7();
        form7.Owner = this;
        form7.ShowDialog();
    }

    private void Form1_FormClosed(object sender, FormClosedEventArgs e)
    {
        Application.Exit();
    }
    /*Каждая сортировка так же и меняет структура данных в файле, что как я посчитал-
не особо важно, ведь лучше отсортированные
данные, чем случайно разбросанные*/
    private void toolStripMenuItem3_Click(object sender, EventArgs e)
    {
        printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderBy(x => x.Price).ToList<Printer>();
        printerBindingSource.ResetBindings(false);
    }

```

```

private void помощьToolStripMenuItem_Click(object sender, EventArgs e)
{
    Process.Start("Пояснение к функционалу.txt");
}

private void сортироватьПоУбываниюЦеныToolStripMenuItem_Click(object sender,
EventArgs e)
{
    printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderByDescending(x => x.Price).ToList<Printer>();
    printerBindingSource.ResetBindings(false);
}

private void сортироватьПоНазваниюToolStripMenuItem_Click(object sender,
EventArgs e)
{
    printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderBy(x => x.Name).ToList<Printer>();
    printerBindingSource.ResetBindings(false);
}

private void сортироватьПоТипуМоделиToolStripMenuItem_Click(object sender,
EventArgs e)
{
    printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderBy(x => x.Type).ToList<Printer>();
    printerBindingSource.ResetBindings(false);
}

private void сортироватьПоВозрастаниюКоличестваToolStripMenuItem_Click(object
sender, EventArgs e)
{
    printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderBy(x => x.Kol_vo).ToList<Printer>();
    printerBindingSource.ResetBindings(false);
}

private void сортироватьПоУбываниюКоличестваToolStripMenuItem_Click(object
sender, EventArgs e)
{
    printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderByDescending(x => x.Kol_vo).ToList<Printer>();
    printerBindingSource.ResetBindings(false);
}

private void сортироватьПоВозрастаниюРейтингаToolStripMenuItem_Click(object
sender, EventArgs e)
{
    printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderBy(x => x.Rate).ToList<Printer>();
    printerBindingSource.ResetBindings(false);
}

private void сортироватьПоУбываниюРейтингаToolStripMenuItem_Click(object sender,
EventArgs e)
{
    printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderByDescending(x => x.Rate).ToList<Printer>();
    printerBindingSource.ResetBindings(false);
}

```

```

        private void сортироватьПоМоделиToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderBy(x => x.Model).ToList<Printer>();
            printerBindingSource.ResetBindings(false);
        }
    }
}

```

## Файл Form2.cs

```

namespace WindowsFormsApp1
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();

            private void Form2_Load(object sender, EventArgs e)
            {
                pictureBox1.Image = new Bitmap("20.png");
                Printer_info.Load_info_from_mdb();
                printerBindingSource.DataSource = Printer.Printer_list;
                printerinfoBindingSource.DataSource = Printer_info.Printer_info_list;
            }

            private void button1_Click(object sender, EventArgs e)
            {
                MessageBox.Show("Создатель- Никитин Роман Андреевич, Научный руководитель-
Горелов Сергей Витальевич.");
            }

            private void button2_Click(object sender, EventArgs e)
            {
                int flag = 0;
                string info = "";
                for (int i = 0; i < Printer_info.Printer_info_list.Count; i++)
                {
                    if (Printer_info.Printer_info_list[i].Name == comboBox1.Text &
Printer_info.Printer_info_list[i].Model == comboBox2.Text)
                    {
                        flag = flag + 1;
                        info += Printer_info.Printer_info_list[i].Info;
                    }
                }
                if (flag == 0)
                {
                    Clipboard.SetDataObject("https://clck.ru/");
                    MessageBox.Show("Ссылки на информацию об этом принтере пока нет,
пожалуйста, добавьте её! Желательно воспользуйтесь сервисом для сокращения ссылок. Ссылка
на сервис уже в вашем буфере обмена!");
                    textBox1.Text = "Введите информацию о принтере!";
                }
                else
                {
                    Clipboard.SetDataObject(info);
                    MessageBox.Show("Ссылка на сайт с информацией в вашем буфере обмена,
просто вставьте её в строку поиска методом Ctrl+V!");
                    flag = 0;
                }
            }
        }
    }
}

```

```

    }

    private void button3_Click(object sender, EventArgs e)
    {
        if (textBox1.Text != "" & textBox1.Text != "Введите информацию о принтере!")
        {
            Printer_info.Add_info(comboBox1.Text, comboBox2.Text, textBox1.Text);
        }

        Printer.Write_log(Form3.name_for_log, " Добавил информацию о принтере ",
comboBox1.Text, comboBox2.Text, textBox1.Text);
        textBox1.Clear();
        printerinfoBindingSource.ResetBindings(false);
    }

    private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        comboBox1.Text = dataGridView1[0,
dataGridView1.CurrentRow.Index].Value.ToString();
        comboBox2.Text = dataGridView1[1,
dataGridView1.CurrentRow.Index].Value.ToString();
        Clipboard.SetDataObject(dataGridView1[2,
dataGridView1.CurrentRow.Index].Value.ToString());
        MessageBox.Show("Ссылка на сайт с информацией в вашем буфере обмена, просто
вставьте ее в строку поиска методом Ctrl+V!");
    }

    private void textBox1_Click(object sender, EventArgs e)
    {
    }
    /*Открываю формы в режиме диалога, потому что могу себе позволить*/
    private void button5_Click(object sender, EventArgs e)
    {
        Form6 form6 = new Form6();
        form6.Owner = this;
        form6.ShowDialog();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        Form7 form7 = new Form7();
        form7.Owner = this;
        form7.ShowDialog();
    }

    private void помощьToolStripMenuItem1_Click(object sender, EventArgs e)
    {
        Process.Start("Пояснение к функционалу.txt");
    }
    /*Каждая сортировка меняет структуру порядок элементов в списке, а так же в
файле*/
    private void сортироватьПоФирмеToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        printerinfoBindingSource.DataSource = Printer_info.Printer_info_list =
Printer_info.Printer_info_list.OrderBy(x => x.Name).ToList<Printer_info>();
        printerinfoBindingSource.ResetBindings(false);
    }

    private void сортироватьПоФирмеЯаToolStripMenuItem_Click(object sender, EventArgs
e)
    {

```

```

        printerinfoBindingSource.DataSource = Printer_info.Printer_info_list =
Printer_info.Printer_info_list.OrderByDescending(x => x.Name).ToList<Printer_info>();
        printerinfoBindingSource.ResetBindings(false);
    }

    private void сортироватьПоМоделиАяToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        printerinfoBindingSource.DataSource = Printer_info.Printer_info_list =
Printer_info.Printer_info_list.OrderBy(x => x.Model).ToList<Printer_info>();
        printerinfoBindingSource.ResetBindings(false);
    }

    private void сортироватьПоМоделиЯаToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        printerinfoBindingSource.DataSource = Printer_info.Printer_info_list =
Printer_info.Printer_info_list.OrderByDescending(x => x.Model).ToList<Printer_info>();
        printerinfoBindingSource.ResetBindings(false);
    }

    private void button6_Click(object sender, EventArgs e)
    {
        if (textBox1.Text != "Изменить инормацию о принтере" & textBox1.Text != "" &
textBox1.Text.Split().Length <= 1 & textBox1.Text.Length <= 25)
        {
            string answer = Printer_info.Change_info_about_printer(comboBox1.Text,
comboBox2.Text, textBox1.Text);
            if (answer == "Успешно!")
            {
                printerinfoBindingSource.ResetBindings(false);
                textBox1.Text = "Изменить инормацию о принтере";
                MessageBox.Show("Информация о принтере успешно изменена!");
            }
            else MessageBox.Show("Вы неправильно ввели данные, либо информации о таком
принтере пока нет, а значит и менять нечего!");
        }
    }
}

```

## Файл Form3.cs

```

namespace WindowsFormsApp1
{
    public partial class Form3 : Form
    {
        public Form3()
        {
            InitializeComponent();
        }
        private void Form3_Load(object sender, EventArgs e)
        {
            Registration.Read_users_from_mdb();
            comboBox1.Text = "Введите логин.";
            pictureBox1.Image = new Bitmap("30.png");
        }
        /*Регистрация пользователя, проверки на данные, введенные в textboxы или в
comboBox*/
        public static string name_for_log = "";
        private void button1_Click(object sender, EventArgs e)
        {

```



```

        if (textBox2.Text != "Введите пароль." & textBox2.Text != "" & comboBox1.Text
!= "Введите логин." & comboBox1.Text != "" & comboBox1.Text.Split().Length == 1 &
        textBox2.Text.Split().Length == 1)
        {
            try
            {
                string answer = Registration.Register(comboBox1.Text, textBox2.Text);
                if (answer == "added_to_list")
                {
                    MessageBox.Show("Отлично, вы зарегистрированы, можете
логиниться!");
                }
                else MessageBox.Show("Такой пользователь уже существует, можете
логиниться!");
            }
            catch
            {
                MessageBox.Show("Вы ввели некорректные данные!");
            }
        }
        else MessageBox.Show("Введите ваш логин и пароль! В логине и пароле не должно
быть пробелов! Если проблема не устраняется, попробуйте ввести другой логин/пароль");
    }

    private void button2_Click(object sender, EventArgs e)
    {
        if (textBox2.Text != "Введите пароль." & textBox2.Text != "")
        {
            try
            {
                string log = Registration.Login(comboBox1.Text, textBox2.Text);
                if (log == "user")
                {
                    Form1 form1 = new Form1();
                    name_for_log = comboBox1.Text;
                    this.Hide();
                    MessageBox.Show("Добро пожаловать! Вы вошли с правами
работника!");
                    form1.Show();
                }
                else if (log == "admin")
                {
                    Form4 form4 = new Form4();
                    name_for_log = comboBox1.Text;
                    this.Hide();
                    MessageBox.Show("Добро пожаловать! Не забывайте, что вы вошли с
правами администратора!");
                    form4.Show();
                }
                else
                    MessageBox.Show("Сначала залогиньтесь!");
            }
            catch
            {
                MessageBox.Show("Вы ввели неверные данные!");
            }
        }
        else MessageBox.Show("Введите ваш логин и пароль!");
    }

    private void textBox2_Click(object sender, EventArgs e)
    {

```

```

    }

    private void помощьToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Process.Start("Пояснение к функционалу.txt");
    }

    private void comboBox1_DropDown(object sender, EventArgs e)
    {
        Registration.Read_users_from_mdb();
        registrationBindingSource.DataSource = Registration.Registration_list;
        comboBox1.DataSource = Registration.Registration_list;
    }
}

```

## Файл Form4.cs

```

{
    public partial class Form4 : Form
    {
        public Form4()
        {
            InitializeComponent();
        }
        Printer print = new Printer("", "", "", 0, 0, 0);

        private void Form4_Load(object sender, EventArgs e)
        {
            pictureBox1.Image = new Bitmap("40.png");
            Printer.Load_printers_from_mdb();
            printerBindingSource.DataSource = Printer.Printer_list;
        }
        int flag = 0;
        private void clear()
        {
            textBox4.Text = "";
            textBox5.Text = "";
            textBox6.Text = "";
            textBox7.Text = "";
            textBox8.Text = "";
            textBox9.Text = "";
        }

        /*За заполнение textboxов отвечают кнопки, на которых описаны возможности
        функций, для выполнения же используется отдельная кнопка "Выполнить"*/
        private void button1_Click(object sender, EventArgs e)
        {
            textBox4.Text = "Введите фирму принтера";
            textBox5.Text = "Введите модель принтера";
            textBox6.Text = "Введите тип принтера";
            textBox7.Text = "Введите цену принтера";
            textBox8.Text = "Введите рейтинг принтера";
            textBox9.Text = "Введите количество принтеров";
            flag = 1;
        }
        private void button2_Click(object sender, EventArgs e)
        {
            if (flag == 0)
                MessageBox.Show("Сначала выберите действие из меню!");
            else if (flag == 1)
            {

```

```

        if (textBox4.Text != "Введите фирму принтера" & textBox5.Text != "Введите
модель принтера" & textBox6.Text != "Введите тип принтера" &
        textBox7.Text != "Введите цену принтера" & textBox8.Text != "Введите
рейтинг принтера" & textBox9.Text != "Введите количество принтеров" &
        textBox4.Text.Split().Length == 1 & textBox5.Text.Split().Length == 1
& textBox6.Text.Split().Length == 1 & textBox7.Text.Split().Length == 1 &
        textBox8.Text.Split().Length == 1 & textBox9.Text.Split().Length ==
1)
    {
        try
        {
            {
                Printer.Add_Printer(textBox4.Text, textBox5.Text,
textBox6.Text, double.Parse(textBox7.Text), double.Parse(textBox8.Text),
int.Parse(textBox9.Text));
                clear();
                flag = 0;
            }
            else MessageBox.Show("Вы вводите данные не правильно! Возможно,
вам стоит поменять знаки пробела на нижнее подчеркивание '_', вводить наименования длиной
менее 20 символов или заполнить все предложенные поля! Так же не забывайте о том, что
рейтинг может быть лишь в 10 бальной шкале, цена до 1000000!");
        }
        catch
        {
            MessageBox.Show("Вы вводите данные не правильно! Возможно, вам
стоит поменять знаки пробела на нижнее подчеркивание '_', вводить наименования длиной
менее 20 символов или заполнить все предложенные поля! Так же не забывайте о том, что
рейтинг может быть лишь в 10 бальной шкале, цена до 1000000!");
        }
    }
    else MessageBox.Show("Вы вводите данные не правильно! Возможно, вам стоит
поменять знаки пробела на нижнее подчеркивание '_', вводить наименования длиной менее 20
символов или заполнить все предложенные поля! Так же не забывайте о том, что рейтинг
может быть лишь в 10 бальной шкале, цена до 1000000!");
}
else if (flag == 2)
{
    if (textBox4.Text != "Введите фирму принтера" & textBox5.Text != "Введите
модель принтера" & textBox9.Text != "Введите количество принтеров")
    {
        try
        {
            if (int.Parse(textBox9.Text) > 0)
            {
                MessageBox.Show(Printer.Sell_Printer(textBox4.Text,
textBox5.Text, int.Parse(textBox9.Text)));
                clear();
                flag = 0;
            }
            else MessageBox.Show("Вы не можете продать меньше, чем 1
принтер!");
        }
        catch
        {
            MessageBox.Show("Вы неправильно ввели данные!");
        }
    }
}
}
else if (flag == 3)
{

```

```

        if (textBox4.Text != "Введите фирму принтера")
        {
            try
            {
                MessageBox.Show(Printer.Read_from_File(textBox4.Text));
                clear();
                flag = 0;
            }
            catch
            {
                MessageBox.Show("Вы неправильно ввели данные!");
            }
        }
    }
    else if (flag == 4)
    {
        if (textBox4.Text != "Введите логин" & textBox5.Text != "Введите пароль"
& textBox6.Text != "Введите тип аккаунта (user/admin)"
        & textBox4.Text != "" & textBox5.Text != "" & textBox6.Text != "" &
        (textBox6.Text == "user" | textBox6.Text == "admin") &
        textBox4.Text.Split().Length == 1 & textBox5.Text.Split().Length == 1
& textBox6.Text.Split().Length == 1 &
        textBox4.Text != "users" & textBox4.Text != "info" & textBox4.Text !=
"check_list" &
        textBox4.Text != "price_list" & textBox4.Text != "printer_info" &
textBox4.Text != "reload" & textBox4.Text != "stat_kol_vo" &
        textBox4.Text != "stat_price")
        {
            try
            {
                string answer = Printer.Add_user(textBox4.Text, textBox5.Text,
textBox6.Text);

                if (answer == "added_to_list")
                {
                    MessageBox.Show("Новый пользователь успешно добавлен!");
                    flag = 0;
                    clear();
                }
                else MessageBox.Show("Такой пользователь уже есть!");
                clear();
            }
            catch
            {
                MessageBox.Show("Вы неправильно ввели данные!");
            }
        }
        else MessageBox.Show("Правильно вводите данные! Вместо пробелов следует
использовать '_', если необходимо.");
    }
    else if (flag == 5)
    {
        if (textBox4.Text != "Введите логин" & textBox5.Text != "Введите пароль"
& textBox6.Text != "Введите тип аккаунта (user/admin)"
        & textBox4.Text != "" & textBox5.Text != "" & textBox6.Text != "" &
        (textBox6.Text == "user" | textBox6.Text == "admin"))
        {
            try
            {
                string answer = Printer.Delete_user(textBox4.Text, textBox5.Text,
textBox6.Text);

                if (answer == "Успешно")
                {
                    MessageBox.Show("Пользователь успешно удален!");
                }
            }
            catch
            {
                MessageBox.Show("Вы неправильно ввели данные!");
            }
        }
    }
}

```

```

        clear();
        flag = 0;
    }
    else MessageBox.Show("Ошибка! Похоже, такого пользователя нет!");
}
catch
{
    MessageBox.Show("Ошибка! Похоже, такого пользователя нет!");
}
}
else MessageBox.Show("В 3 графе должен быть тип аккаунта! Либо user либо
admin.");
}
else if (flag == 6)
{
    if (textBox4.Text != "" & textBox5.Text != "" & textBox6.Text != "" &
textBox7.Text != "" & textBox4.Text != "Введите логин" & textBox5.Text != "Введите тип
аккаунта (user/admin)" &
        textBox6.Text != "Введите тип аккаунта (user/admin)" & textBox7.Text
!= "Введите желаемый тип аккаунта (user/admin)" &
        (textBox6.Text == "user" | textBox7.Text == "admin" | textBox6.Text
== "admin" | textBox7.Text == "user"))
        try
        {
            string answer = Printer.Change_status_of_user(textBox4.Text,
textBox5.Text, textBox6.Text, textBox7.Text);
            if (answer == "Успешно")
            {
                MessageBox.Show("Статус аккаунта успешно изменен!");
                clear();
                flag = 0;
            }
            else MessageBox.Show("Выбранного вами аккаунта нет, либо вы
неверно заполнили данные!");
        }
        catch
        {
            MessageBox.Show("Выбранного вами аккаунта нет, либо вы неверно
заполнили данные!");
        }
        else MessageBox.Show("Выбранного вами аккаунта нет, либо вы неверно
заполнили данные!");
    }
    else if (flag == 7)
    {
        double price = double.TryParse(textBox7.Text, out _) ?
double.Parse(textBox7.Text) : 0;
        double rate = double.TryParse(textBox8.Text, out _) ?
double.Parse(textBox8.Text) : 0;
        int kol_vo = int.TryParse(textBox9.Text, out _) ?
int.Parse(textBox9.Text) : 0;
        try
        {
            string answer = Printer.Change_price_list_items(textBox4.Text,
textBox5.Text, textBox6.Text, price, rate, kol_vo);
            if (answer == "Успешно!")
            {
                clear();
                MessageBox.Show("Вы успешно изменили данные о принтере");
            }
            else if (answer == "Нечего изменять!")
                MessageBox.Show("Вы не ввели данные для замены значений полей или
ввели их неправильно!");
        }
    }
}
}

```

```

        else MessageBox.Show("Такого принтера нет или вы ввели неверные
данные.");
    }
    catch
    {
        MessageBox.Show("Такого принтера нет или вы ввели неверные данные.");
    }
}
else if (flag == 8)
{
    if (textBox4.Text.Split().Length == 1 & textBox5.Text.Split().Length ==
1)
    {
        string answer = Printer.Delete_printer_from_price_list(textBox4.Text,
textBox5.Text);
        if (answer == "Успешно!")
        {
            clear();
            MessageBox.Show("Принтер успешно удален из прайс-листа!");
        }
        else MessageBox.Show("Данного принтера нет в прайс-листе, либо вы
неверно ввели данные!");
    }
    else MessageBox.Show("Данного принтера нет в прайс - листе, либо вы
неверно ввели данные!");
}
printerBindingSource.ResetBindings(false);
}
private void button4_Click(object sender, EventArgs e)
{
    clear();
    Form2 form2 = new Form2();
    form2.Show();
}

private void button3_Click(object sender, EventArgs e)
{
    flag = 3;
    clear();
    textBox4.Text = "Введите название файла для прочтения";
}

private void button6_Click(object sender, EventArgs e)
{
    flag = 2;
    clear();
    textBox4.Text = "Введите фирму принтера";
    textBox5.Text = "Введите модель принтера";
    textBox9.Text = "Введите количество принтеров";
}

private void button7_Click(object sender, EventArgs e)
{
    string text = Printer.Read_from_File("check_list.txt");
    if (text == "")
    {
        MessageBox.Show("Пока ещё нечего закупать!");
    }
    else
    {
        Printer.purchase();
        MessageBox.Show("Закупка прошла успешно!");
    }
}

```

```

        Printer.Write_log(Form3.name_for_log, "произвел закупку недостающих
принтеров");
        printerBindingSource.ResetBindings(false);
        clear();
    }
}

private void button8_Click(object sender, EventArgs e)
{
    clear();
    string text = Printer.Read_from_File("check_list.txt");
    if (text == "")
    {
        MessageBox.Show("Тут пока что пусто!");
    }
    else MessageBox.Show(text);
}

private void button9_Click(object sender, EventArgs e)
{
    clear();
    Printer.statistics();
    MessageBox.Show("Статистика по ценам принтеров записана в файлы
stat_price.txt и stat_kol_vo.txt");
}

private void button10_Click(object sender, EventArgs e)
{
    clear();
    MessageBox.Show(Printer.check_statistics());
    Form5 form5 = new Form5();
    form5.Show();
}

private void button11_Click(object sender, EventArgs e)
{
    clear();
    Printer.Reload();
    Printer.Read_file_with_printers("price_list.txt");
    printerBindingSource.ResetBindings(false);
    MessageBox.Show("Данные успешно откачены до начальных!");
}

private void button12_Click(object sender, EventArgs e)
{
    clear();
    flag = 4;
    textBox4.Text = "Введите логин";
    textBox5.Text = "Введите пароль";
    textBox6.Text = "Введите тип аккаунта (user/admin)";
}

private void button13_Click(object sender, EventArgs e)
{
    clear();
    flag = 5;
    textBox4.Text = "Введите логин";
    textBox5.Text = "Введите пароль";
    textBox6.Text = "Введите тип аккаунта (user/admin)";
}

private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{

```

```

        if (flag == 1 | flag == 2)
        {
            textBox4.Text = dataGridView1[0,
dataGridView1.CurrentRow.Index].Value.ToString();
            textBox5.Text = dataGridView1[1,
dataGridView1.CurrentRow.Index].Value.ToString();
            textBox6.Text = dataGridView1[2,
dataGridView1.CurrentRow.Index].Value.ToString();
            textBox7.Text = dataGridView1[3,
dataGridView1.CurrentRow.Index].Value.ToString();
            textBox8.Text = dataGridView1[4,
dataGridView1.CurrentRow.Index].Value.ToString();
        }
        else if (flag == 7)
        {
            textBox4.Text = dataGridView1[0,
dataGridView1.CurrentRow.Index].Value.ToString();
            textBox5.Text = dataGridView1[1,
dataGridView1.CurrentRow.Index].Value.ToString();
        }
        else if (flag == 8)
        {
            textBox4.Text = dataGridView1[0,
dataGridView1.CurrentRow.Index].Value.ToString();
            textBox5.Text = dataGridView1[1,
dataGridView1.CurrentRow.Index].Value.ToString();
        }
    }

    private void pictureBox2_Click(object sender, EventArgs e)
    {
        pictureBox2.Image = new Bitmap("C_Sharp_logo.png");
    }
    /*При выборе textbox автоматически выделяю весь текст в нем для удобства*/

    private void button14_Click(object sender, EventArgs e)
    {
        Form6 form6 = new Form6();
        form6.Owner = this;
        form6.ShowDialog();
    }

    private void button15_Click(object sender, EventArgs e)
    {
        Form7 form7 = new Form7();
        form7.Owner = this;
        form7.ShowDialog();
    }

    private void button16_Click(object sender, EventArgs e)
    {
        Process.Start("users.txt");
    }

    private void button17_Click(object sender, EventArgs e)
    {
        flag = 6;
        textBox4.Text = "Введите логин";
        textBox5.Text = "Введите пароль";
        textBox6.Text = "Введите тип аккаунта (user/admin)";
        textBox7.Text = "Введите желаемый тип аккаунта (user/admin)";
    }

```



```

private void Form4_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Exit();
}

private void button18_Click(object sender, EventArgs e)
{
    textBox4.Text = "Введите фирму принтера, который вы хотите изменить";
    textBox5.Text = "Введите модель принтера, который вы хотите изменить";
    textBox6.Text = "Введите новый тип принтера, если хотите изменить";
    textBox7.Text = "Введите новую цену принтера, если хотите изменить";
    textBox8.Text = "Введите новый рейтинг принтера, если хотите изменить";
    textBox9.Text = "Введите новое количество принтеров, если хотите изменить";
    flag = 7;
}

private void button19_Click(object sender, EventArgs e)
{
    flag = 8;
    textBox4.Text = "Введите фирму принтера, который вы хотите удалить";
    textBox5.Text = "Введите модель принтера, который вы хотите удалить";
}

private void button20_Click(object sender, EventArgs e)
{
    Process.Start("Пояснение к функционалу.txt");
}

/*Каждая сортировка меняет структуру порядок элементов в списке, а так же в
файле*/
private void сортировкаПоВозрастаниюЦеныToolStripMenuItem_Click(object sender,
EventArgs e)
{
    printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderBy(x => x.Price).ToList<Printer>();
    printerBindingSource.ResetBindings(false);
}

private void помощьToolStripMenuItem_Click(object sender, EventArgs e)
{
    Process.Start("Пояснение к функционалу.txt");
}

private void сортироватьПоНазваниюToolStripMenuItem_Click(object sender,
EventArgs e)
{
    printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderBy(x => x.Name).ToList<Printer>();
    printerBindingSource.ResetBindings(false);
}

private void сортироватьПоТипуМоделиToolStripMenuItem_Click(object sender,
EventArgs e)
{
    printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderBy(x => x.Type).ToList<Printer>();
    printerBindingSource.ResetBindings(false);
}

private void сортироватьПоВозрастаниюКоличестваToolStripMenuItem_Click(object
sender, EventArgs e)
{

```

```

        printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderBy(x => x.Kol_vo).ToList<Printer>();
        printerBindingSource.ResetBindings(false);
    }

    private void сортироватьПоУбываниюКоличестваToolStripMenuItem_Click(object
sender, EventArgs e)
    {
        printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderByDescending(x => x.Kol_vo).ToList<Printer>();
        printerBindingSource.ResetBindings(false);
    }

    private void сортироватьПоВозрастаниюРейтингаToolStripMenuItem_Click(object
sender, EventArgs e)
    {
        printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderBy(x => x.Rate).ToList<Printer>();
        printerBindingSource.ResetBindings(false);
    }

    private void сортироватьПоУбываниюРейтингаToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderByDescending(x => x.Rate).ToList<Printer>();
        printerBindingSource.ResetBindings(false);
    }

    private void сортироватьПоМоделиToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderBy(x => x.Model).ToList<Printer>();
        printerBindingSource.ResetBindings(false);
    }

    private void сортировкаПоУбываниюЦеныToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        printerBindingSource.DataSource = Printer.Printer_list =
Printer.Printer_list.OrderByDescending(x => x.Price).ToList<Printer>();
        printerBindingSource.ResetBindings(false);
    }
}
}

```

## Файл Form5.cs

```

namespace WindowsFormsApp1
{
    public partial class Form5 : Form
    {
        public Form5()
        {
            InitializeComponent();
        }
        public static List<string> list_of_models = new List<string>();
        public static List<int> list_of_numbers = new List<int>();

        public static void pieplot()
        {

```

```

        for (int i = 0; i < Printer.Printer_list.Count; i++)
        {
            list_of_models.Add(Printer.Printer_list[i].Model);
            list_of_numbers.Add(Printer.Printer_list[i].Kol_vo);
        }

private void Form5_Load_1(object sender, EventArgs e)
{
    pieplot();
    chart1.Series[0].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Pie;
    chart1.Series[0].Points.DataBindXY(list_of_models, list_of_numbers);
    list_of_models.Clear();
    list_of_numbers.Clear();
}
}
}

```

## Файл Form6.cs

```

namespace WindowsFormsApp1
{
    public partial class Form6 : Form
    {
        public Form6()
        {
            InitializeComponent();
        }
        /*Просто меняю цвет родительской формы для диалоговой.*/
        private void button3_Click(object sender, EventArgs e)
        {
            Owner.BackColor = System.Drawing.Color.Green;
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Owner.BackColor = System.Drawing.Color.Red;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Owner.BackColor = SystemColors.Control;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Owner.BackColor = System.Drawing.Color.Yellow;
        }

        private void button12_Click(object sender, EventArgs e)
        {
            Owner.BackColor = System.Drawing.Color.Cyan;
        }

        private void button5_Click(object sender, EventArgs e)
        {
            Owner.BackColor = System.Drawing.Color.Aquamarine;
        }

        private void button6_Click(object sender, EventArgs e)
        {
            Owner.BackColor = System.Drawing.Color.Ivory;
        }
    }
}

```

```

    }

    private void button7_Click(object sender, EventArgs e)
    {
        Owner.BackColor = System.Drawing.Color.Maroon;
    }

    private void button11_Click(object sender, EventArgs e)
    {
        Owner.BackColor = System.Drawing.Color.OliveDrab;
    }

    private void button9_Click(object sender, EventArgs e)
    {
        Owner.BackColor = System.Drawing.Color.PaleVioletRed;
    }

    private void button14_Click(object sender, EventArgs e)
    {
        Owner.BackColor = System.Drawing.Color.PeachPuff;
    }

    private void button16_Click(object sender, EventArgs e)
    {
        Owner.BackColor = System.Drawing.Color.SlateGray;
    }

    private void button10_Click(object sender, EventArgs e)
    {
        Owner.BackColor = System.Drawing.Color.Tan;
    }

    private void button8_Click(object sender, EventArgs e)
    {
        Owner.BackColor = System.Drawing.Color.Sienna;
    }

    private void button13_Click(object sender, EventArgs e)
    {
        Owner.BackColor = System.Drawing.Color.PowderBlue;
    }

    private void button15_Click(object sender, EventArgs e)
    {
        Owner.BackColor = System.Drawing.Color.Tomato;
    }

    private void button17_Click(object sender, EventArgs e)
    {
    }

    private void помощьToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Process.Start("Пояснение к функционалу.txt");
    }
}
}

```

## Файл Form7.cs

```

namespace WindowsFormsApp1
{
    public partial class Form7 : Form
    {

```

```

public Form7()
{
    InitializeComponent();
}
/*Просто меняю цвет кнопок родительско для диалоговой формы*/
private void button1_Click(object sender, EventArgs e)
{
    for (int i = 0; i < Owner.Controls.Count; i++)
    {
        if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
            Owner.Controls[i].BackColor = Color.Green;
    }
}

private void button2_Click(object sender, EventArgs e)
{
    for (int i = 0; i < Owner.Controls.Count; i++)
    {
        if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
            Owner.Controls[i].BackColor = Color.Red;
    }
}

private void button3_Click(object sender, EventArgs e)
{
    for (int i = 0; i < Owner.Controls.Count; i++)
    {
        if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
            Owner.Controls[i].BackColor = Color.Blue;
    }
}

private void button4_Click(object sender, EventArgs e)
{
    for (int i = 0; i < Owner.Controls.Count; i++)
    {
        if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
            Owner.Controls[i].BackColor = Color.Lime;
    }
}

private void button5_Click(object sender, EventArgs e)
{
    for (int i = 0; i < Owner.Controls.Count; i++)
    {
        if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
            Owner.Controls[i].BackColor = Color.AliceBlue;
    }
}

private void button6_Click(object sender, EventArgs e)
{
    for (int i = 0; i < Owner.Controls.Count; i++)
    {

```

```

        if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
            Owner.Controls[i].BackColor = Color.DarkOrchid;
    }

    private void button7_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < Owner.Controls.Count; i++)
        {
            if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
                Owner.Controls[i].BackColor = Color.OldLace;
        }
    }

    private void button8_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < Owner.Controls.Count; i++)
        {
            if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
                Owner.Controls[i].BackColor = Color.Goldenrod;
        }
    }

    private void button9_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < Owner.Controls.Count; i++)
        {
            if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
                Owner.Controls[i].BackColor = Color.Turquoise;
        }
    }

    private void button10_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < Owner.Controls.Count; i++)
        {
            if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
                Owner.Controls[i].BackColor = Color.Indigo;
        }
    }

    private void button11_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < Owner.Controls.Count; i++)
        {
            if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
                Owner.Controls[i].BackColor = Color.Moccasin;
        }
    }

    private void button12_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < Owner.Controls.Count; i++)
        {
            if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
                Owner.Controls[i].BackColor = Color.DeepPink;
        }
    }

```

```

    }
}

private void button13_Click(object sender, EventArgs e)
{
    for (int i = 0; i < Owner.Controls.Count; i++)
    {
        if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
            Owner.Controls[i].BackColor = Color.DarkOrchid;
    }
}

private void button14_Click(object sender, EventArgs e)
{
    for (int i = 0; i < Owner.Controls.Count; i++)
    {
        if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
            Owner.Controls[i].BackColor = Color.Chartreuse;
    }
}

private void Form7_Load(object sender, EventArgs e)
{
}

private void button16_Click(object sender, EventArgs e)
{
    for (int i = 0; i < Owner.Controls.Count; i++)
    {
        if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
            Owner.Controls[i].BackColor = SystemColors.Control;
    }
}

private void button15_Click(object sender, EventArgs e)
{
    for (int i = 0; i < Owner.Controls.Count; i++)
    {
        if (Owner.Controls[i].GetType().ToString() ==
"System.Windows.Forms.Button")
            Owner.Controls[i].BackColor = Color.DodgerBlue;
    }
}

private void помощьToolStripMenuItem_Click(object sender, EventArgs e)
{
    Process.Start("Пояснение к функционалу.txt");
}

private void button17_Click(object sender, EventArgs e)
{
}
}
}

```