



Министерство науки и высшего образования Российской Федерации  
Калужский филиал  
федерального государственного бюджетного  
образовательного учреждения высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ, информационные технологии»

## О Т Ч Е Т

### УЧЕБНАЯ ПРАКТИКА

#### «Учебно-технологический практикум»

Студент гр. ИУК4-22Б \_\_\_\_\_ (подпись) ( Шлыков Т.Д. )  
(Ф.И.О.)

Руководитель \_\_\_\_\_ (подпись) ( Глебов С.А. )  
(Ф.И.О.)

Оценка руководителя \_\_\_\_\_ баллов \_\_\_\_\_  
30-50 (дата)

Оценка защиты \_\_\_\_\_ баллов \_\_\_\_\_  
30-50 (дата)

Оценка практики \_\_\_\_\_ баллов \_\_\_\_\_  
(оценка по пятибалльной шкале)

Комиссия: \_\_\_\_\_ ( Глебов С.А. )  
(подпись) (Ф.И.О.)

\_\_\_\_\_ ( Пчелинцева Н.И. )  
(подпись) (Ф.И.О.)

\_\_\_\_\_ ( Амеличева К.А. )  
(подпись) (Ф.И.О.)

Калуга, 2021

*Калужский филиал федерального государственного бюджетного образовательного  
учреждения высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(КФ МГТУ им. Н.Э. Баумана)*

---

УТВЕРЖДАЮ  
Заведующий кафедрой ИУК4  
\_\_\_\_\_ (Гагарин Ю.Е.)  
« 05 » июля 2021 г.

**З А Д А Н И Е**  
**на УЧЕБНУЮ ПРАКТИКУ, Учебно-технологический практикум**

За время прохождения практики студенту необходимо:

1. Осуществить и обосновать выбор предметной области, сформулировать тему проекта и определить область его применения.
2. Осуществить выбор технологий разработки и вспомогательного программного обеспечения.
3. Определить сроки, в рамках которых будет осуществляться контроль за промежуточным состоянием проекта.
4. Разработать или реализовать проект разработка приложения «Магазин мебели»
5. Подготовить отчет и защитить результаты практики.

Дата выдачи задания «05» июля 2021 г.

Руководитель практики \_\_\_\_\_ Амеличев Г.Э.

Задание получил \_\_\_\_\_ Шлыков Т.Д.

## Оглавление

ВВЕДЕНИЕ .....	5
1. МЕТОДЫ И ИНСТРУМЕНТЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ.....	6
1.1. Основание разработки.....	6
1.2 Назначение разработки .....	6
1.3 Технические требования .....	6
1.3.1 Требования к функциональным характеристикам. ....	6
1.3.1.1 Состав выполняемых функций .....	6
1.3.1.2 Организация входных и выходных данных .....	7
1.3.2. Требования к надежности. ....	7
1.3.3. Условия эксплуатации и требования к составу и параметрам технических средств. ....	7
1.3.4. Требования к транспортировке и хранению. ....	7
1.4. Требования к программной документации .....	8
1.5. Техничко-экономические показатели .....	8
1.5.1. Требования к информационной и программной совместимости. ....	8
1.6. Порядок контроля и приёмки .....	9
1.7. Календарный план.....	9
2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА ..	10
2.1. Принцип работы приложения.....	10
2.2. Разработка структурной схемы интерфейса .....	10
2.3. Описание логической структуры.....	12
2.3.1. Вход в программу.....	12
2.3.2. Добавление товара.....	12
2.3.3. Удаление товара .....	13
2.3.4. Сортировка товаров.....	13
2.3.5. Добавление товаров в корзину.....	13
2.3.6. Очистка корзины и покупка.....	13
2.3.7. Выход из аккаунта.....	14
2.4. Разработка плана тестирования .....	14
ЗАКЛЮЧЕНИЕ .....	15
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	16
<i>Основная литература</i> .....	16
<i>Дополнительная литература</i> .....	16

Приложение 1 .....	19
Приложение 2 .....	51

## ВВЕДЕНИЕ

Целью учебной практики является формирование практических навыков, а именно:

- выбор предметной области на основе имеющихся знаний в сфере разработки программного обеспечения, умение грамотно сформулировать тему проекта, область его применения и главные отличия относительно уже имеющихся проектов;
- подбор и оценка технологий разработки и вспомогательного программного обеспечения для реализации итогового проекта в соответствии с выбранной темой;
- оценка функциональности проекта, которая будет реализована к соответствующему сроку, умение грамотно распределить роли и задачи внутри команды;
- представление и оценке качества разработанного приложения.

Для достижения поставленной цели решаются следующие задачи:

- разделение студентов на группы, в которых они будут реализовывать итоговые проекты согласно предметным областям, согласовать выбранные темы, назначить сроки промежуточных встреч и консультаций;
- выбор языка программирования, библиотек, среды разработки, систему контроля версий, сервис хостинга проектов, обоснование своего выбора;
- подготовка доклада с использованием инструментов визуализации (презентации, графики, блок-схемы, UML-диаграммы и пр.), в котором будет представлен отчет команды о проделанной работе, общая информация о разработанном проекте, его преимущества, недостатки и перспективы дальнейшего развития.

# **1. МЕТОДЫ И ИНСТРУМЕНТЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ**

## **1.1. Основание разработки**

Мебельный магазин в первую очередь преследует цель продажи мебели и извлечении прибыли. Приложение «Магазин мебели» способствует увеличению доходности; облегчает жизнь как сотрудникам магазина, так и потенциальным - клиентам и заказчикам.

Во-первых, это приложение как позволяет уменьшить необходимый стартовый капитал для организации нового мебельного магазина, так уменьшить затраты на оплату труда обслуживающему персоналу. Путём сокращения рабочего коллектива

Во-вторых, приложение способствует привлечению новых клиентов. Оно заинтересует людей, которые по тем или иным причинам не могут или не хотят тратить время на поход магазин и долгий самостоятельный выбор мебели. Приложение позволяет заказчикам быстро и удобно выбрать и отсортировать перечень товаров по множеству критериев, не выходя из дома. Расширение клиентской базы приводит к непосредственному увеличению прибыли.

Основанием для данной работы служит обеспечение пользователей возможностью отыскать нужные сведения об ассортименте за оптимально короткий срок.

Наименование работы: «Магазин мебели».

Исполнители: Шлыков Т.Д., Игумнов А.И.

## **1.2 Назначение разработки**

Файл проекта содержит информацию о товарах: их название, цена, вес, рейтинг качества, компания производитель и доп. сведения о товаре.

## **1.3 Технические требования**

### **1.3.1 Требования к функциональным характеристикам.**

#### **1.3.1.1 Состав выполняемых функций.**

Разрабатываемое ПО должно обеспечивать:

- отображение каталога товаров магазина;
- возможность сортировки перечня товаров по всем его категориям;
- возможность просмотра корзины покупателя;
- возможность оформления заказа и его покупки;

Программа для управления информацией о товарах и покупателях должна такие поля иметь, как: Добавить товар, удалить товар, очистить корзину.

Основной режим использования системы - ежедневная работа.

### **1.3.2 Требования к надежности.**

Для обеспечения надежности необходимо проверять обеспечение устойчивого функционирования.

### **1.3.3 Условия эксплуатации и требования к составу и параметрам технических средств.**

Для работы системы необходим оператор.

Требования к составу и параметрам технических средств уточняются на этапе эскизного проектирования системы.

### **1.3.4 Требования к транспортировке и хранению**

Программа поставляется на установочном носителе информации с расширением .exe.

### **1.3.5 Программная документация поставляется в электронном и печатном виде.**

### **1.3.6 Специальные требования.**

Программное обеспечение должно иметь дружелюбный интерфейс, рассчитанный на пользователя (в плане компьютерной грамотности) средней квалификации.

Ввиду объемности проекта задачи предполагается решать поэтапно, при этом модули ПО, созданные в разное время, должны предполагать возможность наращивания системы и быть совместимы друг с другом, поэтому документация на принятое эксплуатационное ПО должна содержать полную информацию, необходимую для работы программистов с ним.

Язык программирования – по выбору исполнителя, должен обеспечивать возможность интеграции программного обеспечения с некоторыми видами периферийного оборудования.

#### **1.4 Требования к программной документации**

Состав технической документации должен включать в себя:

1. Техническое задание (ГОСТ 34.602-89);
2. Руководство программиста (ГОСТ 19.504-79);
3. Руководство пользования (ГОСТ 19.505-79);
4. Демонстрационный чертеж (UML).

#### **1.5 Техничко-экономические показатели**

Эффективность системы определяется удобством использования системы клиентами доставки, а также экономической выгодой, полученной от внедрения программного комплекса.

##### **1.5.1 Требования к информационной и программной совместимости.**

Программа должна работать на платформах ОС Win 7/8/10.

Для нормального функционирования данного программного продукта необходимо наличие:

- 1) IBM совместимый процессор 1ГГц и выше;
- 2) Не менее 10 Мб свободного места в памяти;
- 3) Не менее 1 Гб оперативной памяти (ОЗУ);

Для нормального функционирования программы у клиента необходимы:

- 1) Доступное бесперебойное интернет или локальное соединение;
- 2) .NET Framework Version 4.5 и выше
- 3) Операционная система Windows 7/8/10;
- 4) Не менее 10 Мб свободного места в памяти.



## 1.6 Порядок контроля и приёмки

После передачи Исполнителем отдельного функционального модуля программы Заказчику, последний имеет право тестировать модуль в течение 3 дней. После тестирования Заказчик должен принять работу по данному этапу или в письменном виде изложить причину отказа от принятия. В случае обоснованного отказа Исполнитель в праве доработать модуль.

## 1.7 Календарный план

№ Этапов	Название этапа	Сроки этапа	Чем оканчивается этап
1	Изучение предметной области.	5.07.2021 – 7.07.2021	Предложение по работе системы.
2	Проектирование системы. Разработка программы для работы с базами данных.	8.07.2021–15.07.2021	Акт сдачи-приёмки. Программного продукта
3	Тестирование и отладка программы. Внедрение программы для ежедневной работы.	16.07.2021–17.07.2021	Готовая программа, для удобной работы с реализованным графическим интерфейсом Программная документация. Акт сдачи - приёма работ.

Таблица 1. Календарный план работы

## 2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА

### 2.1 Принцип работы приложения

Программа представляет собой базу данных о товарах. Она позволяет осуществлять покупку. В программе реализованы следующие запросы:

- просмотр всех услуг и информации о них;
- добавление услуг в корзину, ее просмотр и очистка;
- расчёт стоимости услуг;
- добавление нового товара
- удаление выбранного товара
- сортировка товаров по различным критериям

### 2.2 Разработка структурной схемы интерфейса

Заказчик сформулировал следующие потребности: клиент оформляет заказ на некоторое количество услуг, оператор передает информацию о заказе сотрудникам мастерской.

На основании вышесказанного была построена диаграмма вариантов использования для системы оформления услуг по каталогу Рис. 1.

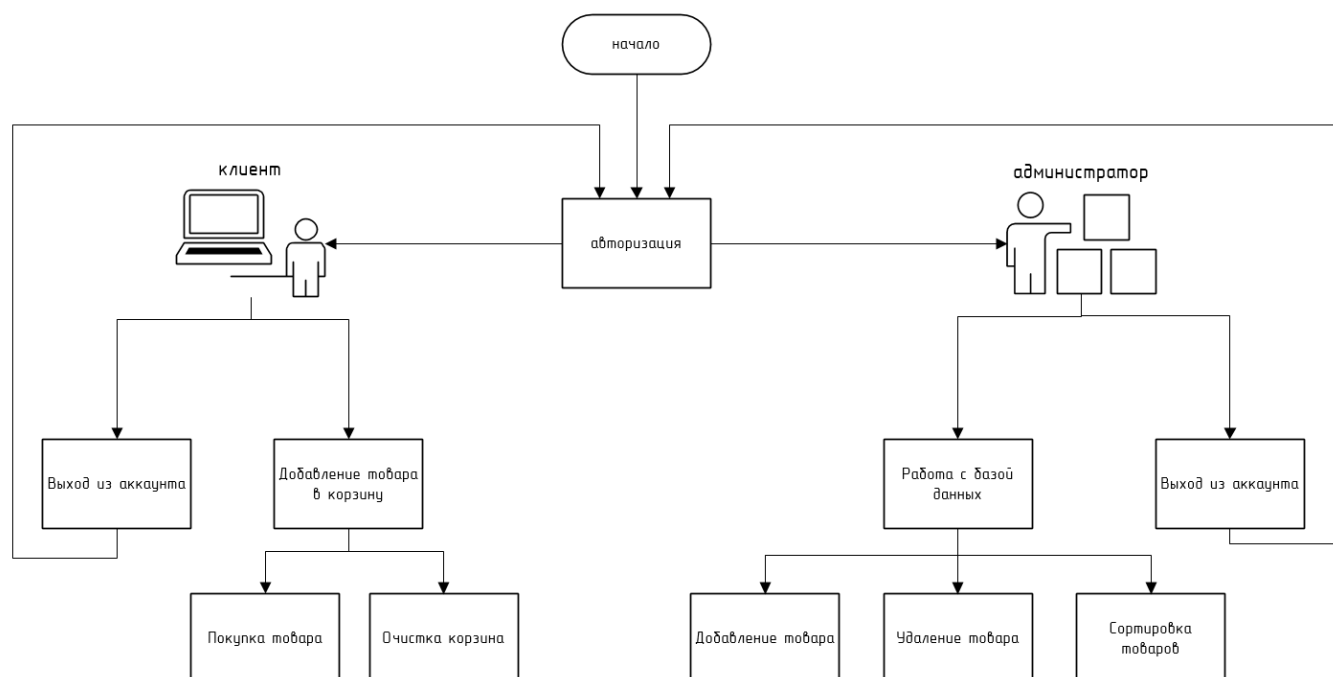


Рис.1. UML диаграмма для системы «Магазин мебели»

Разработанная программа состоит из следующих модульных частей:

Класс Goods – предназначен для работы с информацией, содержит следующие поля: путь для хранения базы данных с товарами, название товара, стоимость, вес, рейтинг, компания производитель, дополнительная информация.

Класс содержит следующие методы:

- getTitle, получающий информацию о названии товара
- getCompanySeller, получающий информацию о компании производителе;
- getAddInfo, получающий информацию о доп. описании товара;
- getCount, получающий информацию о количестве товара;
- getWeight, получающий информацию о массе товара;
- getRating, получающий информацию о рейтинге товара;
- getPrice, получающий информацию о стоимости товара
- setTitle, устанавливающий название товара;
- setCompanySeller, устанавливающий имя компании производителя;
- setAddInfo, устанавливающий доп. описание товара;
- setCount, устанавливающий количество товаров внутри одной позиции;
- setWeight, устанавливающий значение веса товара;
- setRating, устанавливающий значение рейтинга товара;
- setPrice, устанавливающий значение стоимости товара;
- printNote, выводящий на экран всю информацию о товаре;

Класс Money – предназначен для работы с информацией о стоимости товаров, и участвующий при добавлении товара в корзину и при оплате покупки. содержит следующие поля: ID, рубли, копейки.

Класс содержит следующие методы:

- getRubs, получающий информацию о стоимости товара в рублях;
- getCops, получающий информацию о количестве копеек в цене товара;
- getFullPrice, который возвращает полную стоимость товара;
- setRubs, устанавливающий стоимость товара в рублях;
- setCops, устанавливающий количество копеек в цене товара.

Модуль MyForm – содержит методы работы с графическим интерфейсом формы «Магазин мебели». Содержит следующие методы:

- tbAuthLogin, окно ввода логина;
- tbAuthPassword, окно ввода пароля;
- btnAuthEnter, выполняют вход под указанными логином и паролем;
- btnBack, выполняет выход из аккаунта;
- tbDelete, окно ввода названия удаляемого товара;
- btnDeleteNote, удаляет выбранный товар;
- btnSelectGoods, выбирает указанный товар;
- btnNoteAddNote, добавляет новый товар;
- tbNoteTitle, окно ввода названия товара;
- tbNotePrice, окно ввода стоимости товара;
- tbNoteWeight, окно ввода массы товара;
- rb1/rb2/rb3/rb4/rb5, выбор рейтинга товара по 5-бальной шкале;
- tbNoteCompanyName, окно ввода названия компании производителя;
- rtbNoteAddInfo, окно ввода дополнительной информации о товаре;
- btnSortCmpName, сортирует товары по имени компании производителя;
- btnSortRating, сортирует товары по рейтингу товара;
- btnSortWeight, сортирует товары по массе товара;
- btnSortPrice, сортирует товары по стоимости товара;
- btnSortName, сортирует товары по имени товара;
- tbCartName, окно ввода названия товара, добавляемого в корзину;
- tbCartCount, окно ввода количества покупаемых товаров одной позиции;
- btnCartClear, очищает корзину;
- btnCartBuyGoods, осуществляет покупку товаров из корзины.

## **2.3 Описание логической структуры**

### **2.3.1 Вход в программу**

Первой открывается форма MyForm(см. Приложение 2. Рис.1). Форма содержит 2 окна ввода TextBox для логина и пароля соответственно. И кнопку Button «Войти».

### **2.3.2 Добавление товара**

При входе в программу в роли администратора(см. Приложение 2. Рис.2)., видны следующие 4 формы Textbox для ввода названия товара, его цены, веса в кг., компании производителя; 1 форма RichTextbox для описания дополнительной информации о товаре; 1 форма RadioButton, состоящая из 5

кнопок на выбор, определяющих рейтинг товара по 5-бальной шкале; и одна форма Button, для добавления товара с уже введенной о нём информацией.

### **2.3.3 Удаление товара**

У администратора также есть возможность удаления товара. Для этого необходимо в поле ввода TextBox ввести название удаляемого товара и нажать кнопку Button «Удалить». После нажатия этой кнопки появляется форма Label в виде красной надписи «Запись удалена».

### **2.3.4 сортировка товаров**

При необходимости администратор может отсортировать перечень товаров по 5 критериям, а именно: Названию товара, цене, весу, рейтингу, имени компании. Для этого нужно нажать одну из 5 кнопок формы Button с соответствующими названиями, располагающихся вертикально сверху вниз.

### **2.3.5 Добавление товаров в корзину**

Для добавления товара в корзину нужно выбрать ее название в поле формы TextBox, над которым написано «Название товара:», количество в поле ввода TextBox. Добавленные товары отображаются в поле формы Panelbox(см. Приложение 2. Рис.7, Рис.8). Ниже надпись формы Label указывает на итоговую стоимость всех товаров в корзине.

### **2.3.6 Очистка корзины и покупка**

Для очистки корзины от имеющихся в ней товаров надо нажать кнопку «Очистить корзину», выполненной в форме Button (см. Приложение 2. Рис.9). Покупка реализована аналогичным образом кнопкой Button подписанной «Купить»

### **2.3.7 Выход из аккаунта.**

После входа и у администратора, и у клиента есть возможность выйти из аккаунта, нажав кнопку Button «Выйти».

## **2.4 Разработка плана тестирования**

- Запустите исполняемый файл «FurnitureShop.exe»;
- Произведите вход в систему заполнив поля «Логин:», «Пароль:», и нажав «Войти»;
- Войдите в систему как администратор, введя в поля «Логин» и «Пароль:» следующие данные: «admin», «admin1» и нажав на «Войти»;
- Добавьте несколько товаров, задав информацию о них в соответствующих пустых полях и нажав кнопку «Добавить товар».
- Отсортируйте по интересующему критерию
- Выйдите из аккаунта
- Войдите в систему как администратор, введя в поля «Логин» и «Пароль:» следующие данные: «user», «user1» и нажав на «Войти»;
- Добавьте интересующие вас товары в корзину, написав их название в предложенном поле, указав количество каждого из товаров, и, нажав на кнопку «Выбрать товар»;
- Очистите корзину.

## ЗАКЛЮЧЕНИЕ

В результате разработки приложения согласно заданию учебной практики были сформированы умения и навыки выбора предметной области на основе имеющихся знаний о сфере разработки программного обеспечения; по подбору и оценки технологий разработки и вспомогательного программного обеспечения для реализации итогового проекта в соответствии с выбранной темой; оценки функциональности проекта, реализованной к соответствующему сроку, умение грамотно распределить роли и задачи внутри команды; по представлению и оценки качества разработанного приложения.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

### *Основная литература*

1. Моделирование информационных ресурсов [Электронный ресурс]: учебно-методический/ Составитель Огнев Э.Н. - Кемерово : Кемеровский государственный университет культуры и искусств, 2013. - 36 с. : ил., табл. - URL: <http://biblioclub.ru/index.php?page=book&id=274218>
2. Коваленко, Ю.В. Информационно-поисковые системы [Электронный ресурс]: учебно-методическое пособие / Ю.В. Коваленко, Т.А. Сергиенко. — Омск: Омская юридическая академия, 2017. — 38 с.— Режим доступа: <http://www.iprbookshop.ru/66817.html>
3. Маюрникова, Л. А. Основы научных исследований в научно-технической сфере [Электронный ресурс] : учебно-методическое пособие / Л. А. Маюрникова, С. В. Новосёлов. — Кемерово : Кемеровский технологический институт пищевой промышленности, 2009. — 123 с. — Режим доступа: <http://www.iprbookshop.ru/14381.html>
4. Вайнштейн, М. З. Основы научных исследований [Электронный ресурс] : учебное пособие / М. З. Вайнштейн, В. М. Вайнштейн, О. В. Кононова. — Йошкар-Ола : Марийский государственный технический университет, Поволжский государственный технологический университет, ЭБС АСВ, 2011. — 216 с. — Режим доступа: <http://www.iprbookshop.ru/22586.html>
5. Мокий, М.С. Методология научных исследований [Текст]: учебник / М.С. Мокий, А.Л. Никифоров, В.С. Мокий. - М.: Юрайт, 2015. - 255 с.
6. Рогов, В.А. Методика и практика технических экспериментов [Текст]: учеб.пособие / В.А. Рогов, А.В. Антонов, Г.Г. Поздняк. – М.: Академия, 2005. – 288 с.

### *Дополнительная литература*

7. Щербаков, А. Интернет-аналитика [Электронный ресурс]: поиск и оценка информации в web-ресурсах: практическое пособие / А. Щербаков. - М.:



Книжный мир, 2012. - 78 с. - URL: <http://biblioclub.ru/index.php?page=book&id=89693> .

8. Моделирование систем [Текст]: учебник для вузов / С.И. Дворецкий, Ю.Л. Муромцев, В.А. Погонин, А.Г. Схиртладзе. – М.: Академия, 2009. – 320 с.

9. Порсев, Е. Г. Организация и планирование экспериментов [Электронный ресурс] : учебное пособие / Е. Г. Порсев.— Новосибирск : Новосибирский государственный технический университет, 2010. — 155 с. — Режим доступа: <http://www.iprbookshop.ru/45415.html>

10. Новиков, Б. А. Основы технологий баз данных / Б. А. Новиков ; под редакцией Е. В. Рогова. — Москва : ДМК Пресс, 2019. — 240 с. — ISBN 978-5-94074-820-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/123699>

11. Ревунков, Г. И. Базы и банки данных : учебное пособие / Г. И. Ревунков. — Москва : МГТУ им. Н.Э. Баумана, 2011. — 68 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/52425>

12. Кирнос, В. Н. Информатика 2. Основы алгоритмизации и программирования на языке C++ : учебно-методическое пособие / В. Н. Кирнос. — Томск : Томский государственный университет систем управления и радиоэлектроники, Эль Контент, 2013. — 160 с. — ISBN 2227-8397. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/14011.html>

13. Шелупанов, А. А. Информатика. Базовый курс. Часть 3. Основы алгоритмизации и программирования в среде Visual C++ 2005 : учебник / А. А. Шелупанов, В. Н. Кирнос. — Томск : Томский государственный университет систем управления и радиоэлектроники, В-Спектр, 2008. — 216 с. — ISBN 978-5-91191-091-4. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/14013.html>

14. Иванова, Г. С. Средства процедурного программирования Microsoft Visual C++ 2008 : учебное пособие / Г. С. Иванова, Т. Н. Ничушкина, Р. С. Самарев. — Москва : Московский государственный технический университет имени Н.Э.

Баумана, 2012. — 140 с. — ISBN 2227-8397. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/31263.html>

## Приложение 1

### Текст программы:

```
Goods.cpp
#include "Goods.h"
#include "Money.h"

using namespace GoodsInterface;

Goods::Goods() {
    strcpy_s(_title, "");
    strcpy_s(_companyseller, "");
    strcpy_s(_addinfo, "");
    _count = 0;
    _weight = 0;
    _rating = 0;
    _price.setCops(0);
    _price.setRubs(0);
}

Goods::Goods(const Goods& obj) {
    this->setTitle(obj.getTitle());
    this->setCompanySeller(obj.getCompanySeller());
    this->setAddInfo(obj.getAddInfo());
    this->setCount(obj.getCount());
    this->setWeight(obj.getWeight());
    this->setRating(obj.getRating());
    this->setPrice(obj.getPrice());
}

const char* Goods::getTitle() const {
    return _title;
}

const char* Goods::getCompanySeller() const {
    return _companyseller;
}

const char* Goods::getAddInfo() const {
    return _addinfo;
}

const unsigned long int Goods::getCount() const {
    return _count;
}

const unsigned long int Goods::getWeight() const {
    return _weight;
}

const unsigned long int Goods::getRating() const {
    return _rating;
}

const MoneyInterface::Money Goods::getPrice() const {
    return _price;
}

void Goods::setTitle(const char* str) {
    strcpy_s(_title, str);
}

void Goods::setCompanySeller(const char* str) {
    strcpy_s(_companyseller, str);
}

void Goods::setAddInfo(const char* str) {
```

```

        strcpy_s(_addinfo, str);
    }

    void Goods::setCount(const unsigned long int value) {
        _count = value;
    }

    void Goods::setWeight(const unsigned long int value) {
        _weight = value;
    }

    void Goods::setRating(const unsigned long int value) {
        _rating = value;
    }

    void Goods::setPrice(const MoneyInterface::Money& price) {
        _price = price;
    }

    std::fstream& printNote(std::fstream& out, Goods& obj) {
        if (out.tellp() != 0) {
            out << '\n';
        }
        if (obj.getCount() != 0) {
            out << obj.getTitle() << "|"
                << obj.getPrice() << "|"
                << obj.getWeight() << "|"
                << obj.getRating() << "|"
                << obj.getCompanySeller() << "|"
                << obj.getAddInfo() << "|"
                << obj.getCount() << "|";
        }
        else
        {
            out << obj.getTitle() << "|"
                << obj.getPrice() << "|"
                << obj.getWeight() << "|"
                << obj.getRating() << "|"
                << obj.getCompanySeller() << "|"
                << obj.getAddInfo() << "|";
        }

        return out;
    }

    std::ostream& operator<<(std::ostream& out, const Goods& obj) {
        if (obj.getCount() != 0) {
            out << obj.getTitle() << "|"
                << obj.getPrice() << " руб." << "|"
                << obj.getWeight() << " кг." << "|"
                << obj.getRating() << "|"
                << obj.getCompanySeller() << "|"
                << obj.getAddInfo() << "|"
                << obj.getCount() << " шт." << "|"
                << '\n';
        }
        else
        {
            out << obj.getTitle() << "|"
                << obj.getPrice() << " руб." << "|"
                << obj.getWeight() << " кг." << "|"
                << obj.getRating() << "|"
                << obj.getCompanySeller() << "|"
                << obj.getAddInfo() << "|"
                << '\n';
        }
    }

```

```

    }

    return out;
}

Goods.h

#pragma once
#include "Money.h"
#include <string>
#include <iostream>
#include <fstream>

namespace GoodsInterface {
    class Goods {
        char _title[128] = "";
        char _companyseller[128] = "";
        char _addinfo[128] = "";
        unsigned long int _count{};
        unsigned long int _weight{};
        unsigned long int _rating{};
        MoneyInterface::Money _price{};

    public:
        Goods();
        Goods(const Goods& obj);
        const char* getTitle() const;
        const char* getCompanySeller() const;
        const char* getAddInfo() const;
        const unsigned long int getCount() const;
        const unsigned long int getWeight() const;
        const unsigned long int getRating() const;
        const MoneyInterface::Money getPrice() const;

        void setTitle(const char* str);
        void setCompanySeller(const char* str);
        void setAddInfo(const char* str);
        void setCount(const unsigned long int value);
        void setWeight(const unsigned long int value);
        void setRating(const unsigned long int value);
        void setPrice(const MoneyInterface::Money& price);
    };
}

std::ostream& operator<<(std::ostream& out, const GoodsInterface::Goods& obj);
std::fstream& printNote(std::fstream& out, GoodsInterface::Goods& obj);

Money.cpp

#include "Money.h"

using namespace MoneyInterface;

Money::Money() {
    _rubles = 0;
    _copecks = 0;
}

Money::Money(unsigned long int rub, unsigned long int cop) {
    while (cop > 99) {
        rub += 1;
        cop -= 100;
    }
    _rubles = rub;
    _copecks = cop;
}

```

```

}
Money::Money(const Money& sum) {
    _rubles = sum.getRubs();
    _copecks = sum.getCops();
}

const float Money::getFullPrice() const {
    return _rubles + _copecks / 100.0;
}
const unsigned long int Money::getRubs() const {
    return _rubles;
}
const unsigned long int Money::getCops() const {
    return _copecks;
}
void Money::setRubs(const unsigned long int rub) {
    _rubles = rub;
}
void Money::setCops(unsigned long int cop) {
    while (cop > 99) {
        _rubles += 1;
        cop -= 100;
    }
    _copecks = cop;
}
Money& Money::operator=(const Money& obj) {
    this->setRubs(obj.getRubs());
    this->setCops(obj.getCops());
    return *this;
}

std::ostream& operator<<(std::ostream& out, const Money& obj) {
    out << obj.getRubs() << '.' << obj.getCops();
    return out;
}

```

### Money.h

```

#pragma once
#include <fstream>

namespace MoneyInterface {
    class Money {
        unsigned long int _rubles{};
        unsigned long int _copecks{};
    public:
        Money();
        Money(unsigned long int rub, unsigned long int cop);
        Money(const Money& sum);
        const float getFullPrice() const;
        const unsigned long int getRubs() const;
        const unsigned long int getCops() const;
        void setRubs(const unsigned long int rub);
        void setCops(unsigned long int cop);
        Money& operator=(const Money& obj);
    };
}

std::ostream& operator<<(std::ostream& out, const MoneyInterface::Money&
obj);

```

```

MyForm.cpp
#include "MyForm.h"

```

```
using namespace System;
using namespace System::Windows::Forms;
[STAThreadAttribute]
```

```
void main(array<String>^ args) {
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);
    FurnitureShop::MyForm form;
    Application::Run(% form);
}
```

MyForm.h

```
#pragma once
#include <iostream>
#include <string>
#include <vector>
#include <ctime>
#include "Goods.h"
#include "Sort.h"
```

```
namespace FurnitureShop {
```

```
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
```

```
    static const char* GOODS_SHEET = "database/GoodsSheet.txt";
    static const char* CART_SHEET = "database/CartSheet.txt";
    static const char* SALES_REPORT = "database/SalesReport.txt";
```

```
    unsigned long int charToInt(const char* x, int k) {
        unsigned long int number = 0;
        int digit = 1, deg = k - 1;
        for (int i = 0; i < k; i++) {
            if (isdigit(x[i])) {
                digit = pow(10, deg) * (int(x[i]) - 48);
                number += digit;
                deg--;
            }
        }
        if (int(x[0]) == 45) {
            number /= -10;
        }
        return number;
    }
```

```
    std::fstream& operator>>(std::fstream& file, GoodsInterface::Goods&
obj) {
        file.clear();
        std::string buffer_string{};
        char buffer_word[128] = "";
        bool tempBool = false;

        int j{}, k{};
        char title[128] = "";
        char companyseller[128] = "";
        char addinfo[128] = "";
        unsigned long int count{};
        unsigned long int weight{};
```

```

unsigned long int rating{};
MoneyInterface::Money price{};

getline(file, buffer_string);
if (buffer_string.empty()) {
    return file;
}
else
{
    for (int i = 0; buffer_string[i] != '\0'; i++) {
        if (buffer_string[i] == '|') {
            j++;
            if (j == 7) {
                count = charToInt(buffer_word, k);
            }
        }
        else
            switch (j) {
                case 0:
                    title[i] = buffer_string[i];
                    break;
                case 1:
                    if (buffer_string[i] == '.') {
                        price.setRubs(charToInt(buffer_word, k));
                        tempBool = true;
                        for (int l = 0; l < k; l++) {
                            buffer_word[l] = '\0';
                        }
                        k = -1;
                    }
                    else {
                        if (!tempBool) {
                            buffer_word[k] =
buffer_string[i];
                        }
                        else {
                            buffer_word[k] =
buffer_string[i];
                        }
                    }
                    k++;
                    break;
                case 2:
                    if (tempBool) {
                        price.setCops(charToInt(buffer_word, k));
                        for (int l = 0; l < k; l++) {
                            buffer_word[l] = '\0';
                        }
                        k = 0;
                        tempBool = false;
                    }
                    buffer_word[k] = buffer_string[i];
                    k++;
                    break;
                case 3:
                    if (!tempBool) {
                        weight = charToInt(buffer_word,
k);
                        for (int l = 0; l < k; l++) {
                            buffer_word[l] = '\0';
                        }
                        k = 0;
                        tempBool = true;
                    }
            }
    }
}

```



```

        }
        buffer_word[k] = buffer_string[i];
        k++;
        break;
    case 4:
        if (tempBool) {
            rating = charToInt(buffer_word,

k);

            for (int l = 0; l < k; l++) {
                buffer_word[l] = '\0';
            }
            k = 0;
            tempBool = false;
        }
        companyseller[k] = buffer_string[i];
        k++;
        break;
    case 5:
        if (!tempBool) {
            for (int l = 0; l < k; l++) {
                buffer_word[l] = '\0';
            }
            k = 0;
            tempBool = true;
        }
        addinfo[k] = buffer_string[i];
        k++;
        break;
    case 6:
        if (tempBool) {
            for (int l = 0; l < k; l++) {
                buffer_word[l] = '\0';
            }
            k = 0;
            tempBool = false;
        }
        buffer_word[k] = buffer_string[i];
        k++;
        break;

    default:
        std::cout << "Error 404";
        break;
    }

}

}

obj.setTitle(title);
obj.setPrice(price);
obj.setWeight(weight);
obj.setRating(rating);
obj.setCompanySeller(companyseller);
obj.setAddInfo(addinfo);
obj.setCount(count);

return file;
}

std::istream& operator>>(std::istream& in, GoodsInterface::Goods&
obj) {

    MoneyInterface::Money* price = new MoneyInterface::Money;
    char* buffer = new char[128];
    unsigned long int value;

```

```

        in.ignore();

        std::cout << "Введите название товара: ";
        in.getline(buffer, 128);
        obj.setTitle(buffer);

        std::cout << "Цена: \n";
        std::cout << "Рубли: ";
        in >> value;
        price->setRubs(value);
        std::cout << "Копейки: ";
        in >> value;
        price->setCops(value);
        obj.setPrice(*price);
        delete price;

        std::cout << "Вес (кг.): ";
        in >> value;
        obj.setWeight(value);

        std::cout << "Рейтинг (от 0 до 5): ";
        in >> value;
        obj.setRating(value);

        in.ignore();

        std::cout << "Компания-производитель: ";
        in.getline(buffer, 128);
        obj.setCompanySeller(buffer);

        std::cout << "Доп. информация: ";
        in.getline(buffer, 128);
        obj.setAddInfo(buffer);

        return in;
    }

    /// <summary>
    /// Сводка для MyForm
    /// </summary>
    public ref class MyForm : public System::Windows::Forms::Form
    {
    public:
        MyForm(void)
        {
            InitializeComponent();

            //
            //TODO: добавьте код конструктора
            //
        }

    protected:
        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        ~MyForm()
        {
            if (components)
            {
                delete components;
            }
        }
    }

```

```

private: System::Windows::Forms::Label^ lbAuthLogin;
private: System::Windows::Forms::Label^ lbAuthPassword;
private: System::Windows::Forms::Button^ btnAuthEnter;
protected:

private: System::Windows::Forms::TextBox^ tbAuthLogin;
private: System::Windows::Forms::TextBox^ tbAuthPassword;
private: System::Windows::Forms::Label^ lbAuthError;

public: System::Windows::Forms::Button^ btnBack;
private:

private: System::Windows::Forms::Button^ btnNoteAddNote;
public:

private: System::Windows::Forms::Button^ btnDeleteNote;
private: System::Windows::Forms::Button^ btnSelectGoods;
private: System::Windows::Forms::Button^ btnCartBuyGoods;

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

private: System::Windows::Forms::Label^ lbNoteTitle;
private: System::Windows::Forms::Label^ lbNotePrice;
private: System::Windows::Forms::TextBox^ tbNotePrice;
private: System::Windows::Forms::Label^ lbNoteWeight;
private: System::Windows::Forms::TextBox^ tbNoteWeight;
private: System::Windows::Forms::Label^ lbNoteCompanyName;
private: System::Windows::Forms::TextBox^ tbNoteCompanyName;
private: System::Windows::Forms::Label^ lbNoteAddInfo;
private: System::Windows::Forms::RichTextBox^ rtbNoteAddInfo;

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

private: System::Windows::Forms::Panel^ NoteAddPanel;

private: System::Windows::Forms::Panel^ AuthPanel;

```

```

private: System::Windows::Forms::Panel^ SortPanel;
private: System::Windows::Forms::Panel^ DeletePanel;
private: System::Windows::Forms::TextBox^ tbDelete;

private: System::Windows::Forms::Label^ lbDelete;
private: System::Windows::Forms::Label^ lbSort;

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

private: System::Windows::Forms::Button^ btnSortPrice;
private: System::Windows::Forms::Button^ btnSortWeight;

private: System::Windows::Forms::Button^ btnSortCmpName;
private: System::Windows::Forms::Button^ btnSortRating;
private: System::Windows::Forms::Label^ lbDeleteInfo;
private: System::Windows::Forms::ListBox^ lbCartBuyGoods;

private: System::Windows::Forms::Panel^ lbCartPanel;
private: System::Windows::Forms::Label^ lbCartSelectInfo;

private: System::Windows::Forms::Label^ lbCartTitle;
private: System::Windows::Forms::TextBox^ tbCartName;
private: System::Windows::Forms::Label^ lbCartCount;

private: System::Windows::Forms::TextBox^ tbCartCount;
private: System::Windows::Forms::Button^ btnCartClear;

private: System::Windows::Forms::ErrorProvider^ errorProvider1;
private: System::Windows::Forms::ListBox^ lbGoods;
private: System::Windows::Forms::RadioButton^ rb5;
private: System::Windows::Forms::RadioButton^ rb1;
private: System::Windows::Forms::RadioButton^ rb4;
private: System::Windows::Forms::RadioButton^ rb3;
private: System::Windows::Forms::RadioButton^ rb2;
private: System::Windows::Forms::Label^ lbSum;

private: System::ComponentModel::IContainer^ components;

public:

public:

public:

```

```

        protected:

        protected:

        protected:

        private:
            /// <summary>
            /// Обязательная переменная конструктора.
            /// </summary>

#pragma region Windows Form Designer generated code
            /// <summary>
            /// Требуемый метод для поддержки конструктора — не изменяйте
            /// содержимое этого метода с помощью редактора кода.
            /// </summary>
            void InitializeComponent(void)
            {
                this->components = (gcnew
System::ComponentModel::Container());
                System::ComponentModel::ComponentResourceManager^
resources = (gcnew
System::ComponentModel::ComponentResourceManager(MyForm::typeid));
                this->lbAuthLogin = (gcnew
System::Windows::Forms::Label());
                this->lbAuthPassword = (gcnew
System::Windows::Forms::Label());
                this->btnAuthEnter = (gcnew
System::Windows::Forms::Button());
                this->tbAuthLogin = (gcnew
System::Windows::Forms::TextBox());
                this->tbAuthPassword = (gcnew
System::Windows::Forms::TextBox());
                this->lbAuthError = (gcnew
System::Windows::Forms::Label());
                this->btnBack = (gcnew
System::Windows::Forms::Button());
                this->btnNoteAddNote = (gcnew
System::Windows::Forms::Button());
                this->btnDeleteNote = (gcnew
System::Windows::Forms::Button());
                this->btnSelectGoods = (gcnew
System::Windows::Forms::Button());
                this->btnCartBuyGoods = (gcnew
System::Windows::Forms::Button());
                this->tbNoteTitle = (gcnew
System::Windows::Forms::TextBox());
                this->lbNoteTitle = (gcnew
System::Windows::Forms::Label());
                this->lbNotePrice = (gcnew
System::Windows::Forms::Label());
                this->tbNotePrice = (gcnew
System::Windows::Forms::TextBox());
                this->lbNoteWeight = (gcnew
System::Windows::Forms::Label());
                this->tbNoteWeight = (gcnew
System::Windows::Forms::TextBox());
                this->lbNoteCompanyName = (gcnew
System::Windows::Forms::Label());
                this->tbNoteCompanyName = (gcnew
System::Windows::Forms::TextBox());

```

```

        this->lbNoteAddInfo = (gcnew
System::Windows::Forms::Label());
        this->rtbNoteAddInfo = (gcnew
System::Windows::Forms::RichTextBox());
        this->lbNoteRating = (gcnew
System::Windows::Forms::Label());
        this->NoteAddPanel = (gcnew
System::Windows::Forms::Panel());
        this->rb5 = (gcnew
System::Windows::Forms::RadioButton());
        this->rb1 = (gcnew
System::Windows::Forms::RadioButton());
        this->rb4 = (gcnew
System::Windows::Forms::RadioButton());
        this->rb3 = (gcnew
System::Windows::Forms::RadioButton());
        this->rb2 = (gcnew
System::Windows::Forms::RadioButton());
        this->AuthPanel = (gcnew
System::Windows::Forms::Panel());
        this->SortPanel = (gcnew
System::Windows::Forms::Panel());
        this->btnSortCmpName = (gcnew
System::Windows::Forms::Button());
        this->btnSortRating = (gcnew
System::Windows::Forms::Button());
        this->btnSortWeight = (gcnew
System::Windows::Forms::Button());
        this->btnSortPrice = (gcnew
System::Windows::Forms::Button());
        this->lbSort = (gcnew System::Windows::Forms::Label());
        this->btnSortName = (gcnew
System::Windows::Forms::Button());
        this->DeletePanel = (gcnew
System::Windows::Forms::Panel());
        this->lbDeleteInfo = (gcnew
System::Windows::Forms::Label());
        this->tbDelete = (gcnew
System::Windows::Forms::TextBox());
        this->lbDelete = (gcnew
System::Windows::Forms::Label());
        this->lbCartBuyGoods = (gcnew
System::Windows::Forms::ListBox());
        this->lbCartPanel = (gcnew
System::Windows::Forms::Panel());
        this->lbSum = (gcnew System::Windows::Forms::Label());
        this->btnCartClear = (gcnew
System::Windows::Forms::Button());
        this->tbCartCount = (gcnew
System::Windows::Forms::TextBox());
        this->lbCartCount = (gcnew
System::Windows::Forms::Label());
        this->lbCartSelectInfo = (gcnew
System::Windows::Forms::Label());
        this->lbCartTitle = (gcnew
System::Windows::Forms::Label());
        this->tbCartName = (gcnew
System::Windows::Forms::TextBox());
        this->errorProvider1 = (gcnew
System::Windows::Forms::ErrorProvider(this->components));
        this->lbGoods = (gcnew
System::Windows::Forms::ListBox());
        this->NoteAddPanel->SuspendLayout();
        this->AuthPanel->SuspendLayout();
        this->SortPanel->SuspendLayout();

```

```

        this->DeletePanel->SuspendLayout();
        this->lbCartPanel->SuspendLayout();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>errorProvider1))->BeginInit();
        this->SuspendLayout();
        //
        // lbAuthLogin
        //
        resources->ApplyResources(this->lbAuthLogin,
L"lbAuthLogin");
        this->lbAuthLogin->Name = L"lbAuthLogin";
        //
        // lbAuthPassword
        //
        resources->ApplyResources(this->lbAuthPassword,
L"lbAuthPassword");
        this->lbAuthPassword->Name = L"lbAuthPassword";
        //
        // btnAuthEnter
        //
        resources->ApplyResources(this->btnAuthEnter,
L"btnAuthEnter");
        this->btnAuthEnter->Name = L"btnAuthEnter";
        this->btnAuthEnter->UseVisualStyleBackColor = true;
        this->btnAuthEnter->Click += gcnew
System::EventHandler(this, &MyForm::btnEnter_Click);
        //
        // tbAuthLogin
        //
        resources->ApplyResources(this->tbAuthLogin,
L"tbAuthLogin");
        this->tbAuthLogin->Name = L"tbAuthLogin";
        //
        // tbAuthPassword
        //
        resources->ApplyResources(this->tbAuthPassword,
L"tbAuthPassword");
        this->tbAuthPassword->Name = L"tbAuthPassword";
        this->tbAuthPassword->UseSystemPasswordChar = true;
        //
        // lbAuthError
        //
        resources->ApplyResources(this->lbAuthError,
L"lbAuthError");
        this->lbAuthError->Name = L"lbAuthError";
        //
        // btnBack
        //
        resources->ApplyResources(this->btnBack, L"btnBack");
        this->btnBack->Name = L"btnBack";
        this->btnBack->UseVisualStyleBackColor = true;
        this->btnBack->Click += gcnew System::EventHandler(this,
&MyForm::btBack_Click);
        //
        // btnNoteAddNote
        //
        resources->ApplyResources(this->btnNoteAddNote,
L"btnNoteAddNote");
        this->btnNoteAddNote->Name = L"btnNoteAddNote";
        this->btnNoteAddNote->UseVisualStyleBackColor = true;
        this->btnNoteAddNote->Click += gcnew
System::EventHandler(this, &MyForm::btAddNote_Click);
        //
        // btnDeleteNote

```

```

//
resources->ApplyResources(this->btnDeleteNote,
L"btnDeleteNote");
this->btnDeleteNote->Name = L"btnDeleteNote";
this->btnDeleteNote->UseVisualStyleBackColor = true;
this->btnDeleteNote->Click += gcnew
System::EventHandler(this, &MyForm::btDeleteNote_Click);
//
// btnSelectGoods
//
resources->ApplyResources(this->btnSelectGoods,
L"btnSelectGoods");
this->btnSelectGoods->Name = L"btnSelectGoods";
this->btnSelectGoods->UseVisualStyleBackColor = true;
this->btnSelectGoods->Click += gcnew
System::EventHandler(this, &MyForm::btnSelectGoods_Click);
//
// btnCartBuyGoods
//
resources->ApplyResources(this->btnCartBuyGoods,
L"btnCartBuyGoods");
this->btnCartBuyGoods->Name = L"btnCartBuyGoods";
this->btnCartBuyGoods->UseVisualStyleBackColor = true;
this->btnCartBuyGoods->Click += gcnew
System::EventHandler(this, &MyForm::btnCartBuyGoods_Click);
//
// tbNoteTitle
//
resources->ApplyResources(this->tbNoteTitle,
L"tbNoteTitle");
this->tbNoteTitle->Name = L"tbNoteTitle";
//
// lbNoteTitle
//
resources->ApplyResources(this->lbNoteTitle,
L"lbNoteTitle");
this->lbNoteTitle->Name = L"lbNoteTitle";
//
// lbNotePrice
//
resources->ApplyResources(this->lbNotePrice,
L"lbNotePrice");
this->lbNotePrice->Name = L"lbNotePrice";
//
// tbNotePrice
//
resources->ApplyResources(this->tbNotePrice,
L"tbNotePrice");
this->tbNotePrice->Name = L"tbNotePrice";
this->tbNotePrice->KeyPress += gcnew
System::Windows::Forms::KeyPressEventHandler(this,
&MyForm::tbNotePrice_KeyPress);
//
// lbNoteWeight
//
resources->ApplyResources(this->lbNoteWeight,
L"lbNoteWeight");
this->lbNoteWeight->Name = L"lbNoteWeight";
//
// tbNoteWeight
//
resources->ApplyResources(this->tbNoteWeight,
L"tbNoteWeight");
this->tbNoteWeight->Name = L"tbNoteWeight";

```



```

        this->tbNoteWeight->KeyPress += gcnew
System::Windows::Forms::KeyPressEventHandler(this,
&MyForm::tbNoteWeight_KeyPress);
        //
        // lbNoteCompanyName
        //
        resources->ApplyResources(this->lbNoteCompanyName,
L"lbNoteCompanyName");
        this->lbNoteCompanyName->Name = L"lbNoteCompanyName";
        //
        // tbNoteCompanyName
        //
        resources->ApplyResources(this->tbNoteCompanyName,
L"tbNoteCompanyName");
        this->tbNoteCompanyName->Name = L"tbNoteCompanyName";
        //
        // lbNoteAddInfo
        //
        resources->ApplyResources(this->lbNoteAddInfo,
L"lbNoteAddInfo");
        this->lbNoteAddInfo->Name = L"lbNoteAddInfo";
        //
        // rtbNoteAddInfo
        //
        resources->ApplyResources(this->rtbNoteAddInfo,
L"rtbNoteAddInfo");
        this->rtbNoteAddInfo->Name = L"rtbNoteAddInfo";
        //
        // lbNoteRating
        //
        resources->ApplyResources(this->lbNoteRating,
L"lbNoteRating");
        this->lbNoteRating->Name = L"lbNoteRating";
        //
        // NoteAddPanel
        //
        this->NoteAddPanel->Controls->Add(this->rb5);
        this->NoteAddPanel->Controls->Add(this->rb1);
        this->NoteAddPanel->Controls->Add(this->lbNoteTitle);
        this->NoteAddPanel->Controls->Add(this->rb4);
        this->NoteAddPanel->Controls->Add(this->btnNoteAddNote);
        this->NoteAddPanel->Controls->Add(this->rb3);
        this->NoteAddPanel->Controls->Add(this->tbNoteTitle);
        this->NoteAddPanel->Controls->Add(this->rb2);
        this->NoteAddPanel->Controls->Add(this->lbNotePrice);
        this->NoteAddPanel->Controls->Add(this->tbNotePrice);
        this->NoteAddPanel->Controls->Add(this->lbNoteWeight);
        this->NoteAddPanel->Controls->Add(this->lbNoteRating);
        this->NoteAddPanel->Controls->Add(this->tbNoteWeight);
        this->NoteAddPanel->Controls->Add(this->rtbNoteAddInfo);
        this->NoteAddPanel->Controls->Add(this->
>lbNoteCompanyName);
        this->NoteAddPanel->Controls->Add(this->lbNoteAddInfo);
        this->NoteAddPanel->Controls->Add(this->
>tbNoteCompanyName);
        resources->ApplyResources(this->NoteAddPanel,
L"NoteAddPanel");
        this->NoteAddPanel->Name = L"NoteAddPanel";
        //
        // rb5
        //
        resources->ApplyResources(this->rb5, L"rb5");
        this->rb5->Name = L"rb5";
        this->rb5->UseVisualStyleBackColor = true;
        //

```

```

// rb1
//
resources->ApplyResources(this->rb1, L"rb1");
this->rb1->Name = L"rb1";
this->rb1->UseVisualStyleBackColor = true;
//
// rb4
//
resources->ApplyResources(this->rb4, L"rb4");
this->rb4->Name = L"rb4";
this->rb4->UseVisualStyleBackColor = true;
//
// rb3
//
resources->ApplyResources(this->rb3, L"rb3");
this->rb3->Checked = true;
this->rb3->Name = L"rb3";
this->rb3->TabStop = true;
this->rb3->UseVisualStyleBackColor = true;
//
// rb2
//
resources->ApplyResources(this->rb2, L"rb2");
this->rb2->Name = L"rb2";
this->rb2->UseVisualStyleBackColor = true;
//
// AuthPanel
//
this->AuthPanel->BackColor =
System::Drawing::SystemColors::ActiveBorder;
resources->ApplyResources(this->AuthPanel,
L"AuthPanel");

this->AuthPanel->Controls->Add(this->lbAuthLogin);
this->AuthPanel->Controls->Add(this->lbAuthPassword);
this->AuthPanel->Controls->Add(this->btnAuthEnter);
this->AuthPanel->Controls->Add(this->tbAuthLogin);
this->AuthPanel->Controls->Add(this->tbAuthPassword);
this->AuthPanel->Controls->Add(this->lbAuthError);
this->AuthPanel->Name = L"AuthPanel";
//
// SortPanel
//
this->SortPanel->Controls->Add(this->btnSortCmpName);
this->SortPanel->Controls->Add(this->btnSortRating);
this->SortPanel->Controls->Add(this->btnSortWeight);
this->SortPanel->Controls->Add(this->btnSortPrice);
this->SortPanel->Controls->Add(this->lbSort);
this->SortPanel->Controls->Add(this->btnSortName);
resources->ApplyResources(this->SortPanel,
L"SortPanel");

this->SortPanel->Name = L"SortPanel";
//
// btnSortCmpName
//
resources->ApplyResources(this->btnSortCmpName,
L"btnSortCmpName");

this->btnSortCmpName->Name = L"btnSortCmpName";
this->btnSortCmpName->UseVisualStyleBackColor = true;
this->btnSortCmpName->Click += gnew
System::EventHandler(this, &MyForm::btnSortCmpName_Click);
//
// btnSortRating
//
resources->ApplyResources(this->btnSortRating,
L"btnSortRating");

```

```

        this->btnSortRating->Name = L"btnSortRating";
        this->btnSortRating->UseVisualStyleBackColor = true;
        this->btnSortRating->Click += gnew
System::EventHandler(this, &MyForm::btnSortRating_Click);
        //
        // btnSortWeight
        //
        resources->ApplyResources(this->btnSortWeight,
L"btnSortWeight");
        this->btnSortWeight->Name = L"btnSortWeight";
        this->btnSortWeight->UseVisualStyleBackColor = true;
        this->btnSortWeight->Click += gnew
System::EventHandler(this, &MyForm::btnSortWeight_Click);
        //
        // btnSortPrice
        //
        resources->ApplyResources(this->btnSortPrice,
L"btnSortPrice");
        this->btnSortPrice->Name = L"btnSortPrice";
        this->btnSortPrice->UseVisualStyleBackColor = true;
        this->btnSortPrice->Click += gnew
System::EventHandler(this, &MyForm::btSortPrice_Click);
        //
        // lbSort
        //
        resources->ApplyResources(this->lbSort, L"lbSort");
        this->lbSort->Name = L"lbSort";
        //
        // btnSortName
        //
        resources->ApplyResources(this->btnSortName,
L"btnSortName");
        this->btnSortName->Name = L"btnSortName";
        this->btnSortName->UseVisualStyleBackColor = true;
        this->btnSortName->Click += gnew
System::EventHandler(this, &MyForm::btnSortName_Click);
        //
        // DeletePanel
        //
        this->DeletePanel->Controls->Add(this->lbDeleteInfo);
        this->DeletePanel->Controls->Add(this->tbDelete);
        this->DeletePanel->Controls->Add(this->lbDelete);
        this->DeletePanel->Controls->Add(this->btnDeleteNote);
        resources->ApplyResources(this->DeletePanel,
L"DeletePanel");
        this->DeletePanel->Name = L"DeletePanel";
        //
        // lbDeleteInfo
        //
        resources->ApplyResources(this->lbDeleteInfo,
L"lbDeleteInfo");
        this->lbDeleteInfo->Name = L"lbDeleteInfo";
        //
        // tbDelete
        //
        resources->ApplyResources(this->tbDelete, L"tbDelete");
        this->tbDelete->Name = L"tbDelete";
        //
        // lbDelete
        //
        resources->ApplyResources(this->lbDelete, L"lbDelete");
        this->lbDelete->Name = L"lbDelete";
        //
        // lbCartBuyGoods
        //

```

```

        this->lbCartBuyGoods->FormattingEnabled = true;
        resources->ApplyResources(this->lbCartBuyGoods,

L"lbCartBuyGoods");
        this->lbCartBuyGoods->Name = L"lbCartBuyGoods";
        //
        // lbCartPanel
        //
        this->lbCartPanel->Controls->Add(this->lbSum);
        this->lbCartPanel->Controls->Add(this->btnCartClear);
        this->lbCartPanel->Controls->Add(this->tbCartCount);
        this->lbCartPanel->Controls->Add(this->lbCartCount);
        this->lbCartPanel->Controls->Add(this->

>lbCartSelectInfo);
        this->lbCartPanel->Controls->Add(this->lbCartTitle);
        this->lbCartPanel->Controls->Add(this->tbCartName);
        this->lbCartPanel->Controls->Add(this->lbCartBuyGoods);
        this->lbCartPanel->Controls->Add(this->btnCartBuyGoods);
        this->lbCartPanel->Controls->Add(this->btnSelectGoods);
        resources->ApplyResources(this->lbCartPanel,

L"lbCartPanel");
        this->lbCartPanel->Name = L"lbCartPanel";
        //
        // lbSum
        //
        resources->ApplyResources(this->lbSum, L"lbSum");
        this->lbSum->Name = L"lbSum";
        //
        // btnCartClear
        //
        resources->ApplyResources(this->btnCartClear,

L"btnCartClear");
        this->btnCartClear->Name = L"btnCartClear";
        this->btnCartClear->UseVisualStyleBackColor = true;
        this->btnCartClear->Click += gcnew
System::EventHandler(this, &MyForm::btnCartClear_Click);
        //
        // tbCartCount
        //
        resources->ApplyResources(this->tbCartCount,

L"tbCartCount");
        this->tbCartCount->Name = L"tbCartCount";
        this->tbCartCount->KeyPress += gcnew
System::Windows::Forms::KeyPressEventHandler(this,
&MyForm::tbCartCount_KeyPress);
        //
        // lbCartCount
        //
        resources->ApplyResources(this->lbCartCount,

L"lbCartCount");
        this->lbCartCount->Name = L"lbCartCount";
        //
        // lbCartSelectInfo
        //
        resources->ApplyResources(this->lbCartSelectInfo,

L"lbCartSelectInfo");
        this->lbCartSelectInfo->ForeColor =
System::Drawing::Color::Red;
        this->lbCartSelectInfo->Name = L"lbCartSelectInfo";
        //
        // lbCartTitle
        //
        resources->ApplyResources(this->lbCartTitle,

L"lbCartTitle");
        this->lbCartTitle->Name = L"lbCartTitle";
        //

```

```

        // tbCartName
        //
        resources->ApplyResources(this->tbCartName,
L"tbCartName");
        this->tbCartName->Name = L"tbCartName";
        //
        // errorProvider1
        //
        this->errorProvider1->ContainerControl = this;
        //
        // lbGoods
        //
        this->lbGoods->FormattingEnabled = true;
        resources->ApplyResources(this->lbGoods, L"lbGoods");
        this->lbGoods->Name = L"lbGoods";
        //
        // MyForm
        //
        resources->ApplyResources(this, L"$this");
        this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
        this->Controls->Add(this->lbGoods);
        this->Controls->Add(this->lbCartPanel);
        this->Controls->Add(this->DeletePanel);
        this->Controls->Add(this->SortPanel);
        this->Controls->Add(this->AuthPanel);
        this->Controls->Add(this->NoteAddPanel);
        this->Controls->Add(this->btnBack);
        this->FormBorderStyle =
System::Windows::Forms::FormBorderStyle::FixedSingle;
        this->MaximizeBox = false;
        this->MinimizeBox = false;
        this->Name = L"MyForm";
        this->Load += gcnew System::EventHandler(this,
&MyForm::MyForm_Load);
        this->NoteAddPanel->ResumeLayout(false);
        this->NoteAddPanel->PerformLayout();
        this->AuthPanel->ResumeLayout(false);
        this->AuthPanel->PerformLayout();
        this->SortPanel->ResumeLayout(false);
        this->SortPanel->PerformLayout();
        this->DeletePanel->ResumeLayout(false);
        this->DeletePanel->PerformLayout();
        this->lbCartPanel->ResumeLayout(false);
        this->lbCartPanel->PerformLayout();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>errorProvider1))->EndInit();
        this->ResumeLayout(false);

    }
#pragma endregion

    void AuthTurnOn() {
        this->AuthPanel->Visible = true;
        this->lbGoods->Visible = false;
        this->btnBack->Visible = false;
    };
    void AuthTurnOff() {
        this->AuthPanel->Visible = false;
        this->lbGoods->Visible = true;
        this->btnBack->Visible = true;
    };
    void AdminPanelTurnOn() {
        this->DeletePanel->Visible = true;

```

```

        this->NoteAddPanel->Visible = true;
        this->SortPanel->Visible = true;
    };
    void AdminPanelTurnOff() {
        this->DeletePanel->Visible = false;
        this->NoteAddPanel->Visible = false;
        this->SortPanel->Visible = false;
    };
    void UserPanelTurnOn() {
        this->lbCartPanel->Visible = true;
    };
    void UserPanelTurnOff() {
        this->lbCartPanel->Visible = false;
    };

    void reload() {
        std::fstream fileGoods;

        fileGoods.open(GOODS_SHEET);
        this->lbGoods->Items->Clear();
        readSheet(fileGoods);
        fileGoods.close();
    }

    void reloadCart() {
        std::fstream fileCart;

        fileCart.open(CART_SHEET);
        this->lbCartBuyGoods->Items->Clear();
        readSheetCart(fileCart);
        fileCart.close();
    }

    void addNote(std::fstream& out) {
        MoneyInterface::Money* price = new
MoneyInterface::Money;
        GoodsInterface::Goods obj;
        if (this->tbNotePrice->Text != "" && this->tbNoteWeight-
>Text != "" && this->tbNoteTitle->Text != ""
            && tbNoteCompanyName->Text != "" && this-
>rtbNoteAddInfo->Text != "") {
            float TempPrice = float::Parse(this->tbNotePrice-
>Text);

            using namespace System::Runtime::InteropServices;
            obj.setTitle((const
char*)(void*)Marshal::StringToHGlobalAnsi(this->tbNoteTitle->Text));
            price->setRubs(int(TempPrice));
            price->setCops((TempPrice - int(TempPrice)) *
100.0);

            obj.setPrice(*price);
            if (rb1->Checked) {
                obj.setRating(1);
            }
            else if (rb2->Checked) {
                obj.setRating(2);
            }
            else if (rb3->Checked) {
                obj.setRating(3);
            }
            else if (rb4->Checked) {
                obj.setRating(4);
            }
            else if (rb5->Checked) {
                obj.setRating(5);
            }
        }
    }

```

```

    }
    else
    {
        obj.setRating(0);
    }
    obj.setWeight(Int64::Parse(this->tbNoteWeight->Text));
    obj.setCompanySeller((const
char*)(void*)Marshal::StringToHGlobalAnsi(this->tbNoteCompanyName->Text));
    obj.setAddInfo((const
char*)(void*)Marshal::StringToHGlobalAnsi(this->rtbNoteAddInfo->Text));
    out.seekp(0, std::ios_base::end);
    printNote(out, obj);
    out.close();
    out.open(SALES_REPORT, std::ios::app);
}

}

void readSheet(std::fstream& file) {
    System::String^ goodsinfo;
    GoodsInterface::Goods obj;
    std::vector<GoodsInterface::Goods> array_goods;
    if (file.peek() == EOF) {
        goodsinfo = "Данных нет";
        this->lbGoods->Items->Insert(0, goodsinfo);
    }
    else {
        while (!file.eof()) {
            file >> obj;
            if (obj.getTitle() != "") {
                array_goods.push_back(obj);
            }
        }
        for (int i{}; i < array_goods.size(); i++) {

            goodsinfo = gcnw
String(array_goods[i].getTitle()) + " | "
+ array_goods[i].getPrice().getRubs()
+ "." + array_goods[i].getPrice().getCops() + " руб. | "
+ array_goods[i].getWeight() + " кг. | "
+ array_goods[i].getRating() + " * | "
+ gcnw
String(array_goods[i].getCompanySeller()) + " | "
+ gcnw
String(array_goods[i].getAddInfo()) + " |";
            this->lbGoods->Items->Insert(i, goodsinfo);
        }
    }
}

void readSheetCart(std::fstream& file) {
    long double Cost{};
    long double FinalCost{};
    System::String^ goodsinfo;
    GoodsInterface::Goods obj;
    std::vector<GoodsInterface::Goods> array_goods;
    if (file.peek() == EOF) {
        goodsinfo = "Данных нет";
        this->lbSum->Text = "Итого: 0 руб.";
        this->lbCartBuyGoods->Items->Insert(0, goodsinfo);
    }
    else {
        while (!file.eof()) {
            file >> obj;

```

```

        if (obj.getTitle() != "") {
            array_goods.push_back(obj);
        }
    }
    for (int i{}; i < array_goods.size(); i++) {
        goodsinfo = gcnw
String(array_goods[i].getTitle()) + " | "
        + array_goods[i].getPrice().getRubs()
+ "." + array_goods[i].getPrice().getCops() + " руб. | "
        + array_goods[i].getWeight() + " кг. | "
"
        + array_goods[i].getRating() + " * | "
        + gcnw
String(array_goods[i].getCompanySeller()) + " | "
        + gcnw
String(array_goods[i].getAddInfo()) + " | "
        + array_goods[i].getCount() + " шт.
|";
        this->lbCartBuyGoods->Items->Insert(i,
goodsinfo);
    }
    for (int i{}; i < array_goods.size(); i++)
{ //исключение ненужного элемента
        Cost = (array_goods[i].getPrice().getCops())
/ 100.0 * array_goods[i].getCount() +
        array_goods[i].getPrice().getRubs() *
array_goods[i].getCount();
        FinalCost = FinalCost + Cost;
    }
    this->lbSum->Text = "Итого: " + FinalCost + "
руб.";
    Cost = 0;
    FinalCost = 0;
}

void deleteNote(std::fstream& file) {
    this->lbDeleteInfo->Visible = true;
    GoodsInterface::Goods obj;
    std::vector<GoodsInterface::Goods> array_goods;
    char* buffer = new char[128];
    bool fl{};
    using namespace System::Runtime::InteropServices;
    buffer =
(char*) (void*) Marshal::StringToHGlobalAnsi(this->tbDelete->Text);

    while (!file.eof()) {
        file >> obj;
        if (strcmp(obj.getTitle(), buffer)) {
            array_goods.push_back(obj);
        }
        else {
            fl = true;
        }
    }
    if (fl == false) {
        this->lbDeleteInfo->Text = "Запись не найдена";
    }
    else {
        file.close();
        file.open(GOODS_SHEET, std::ofstream::out |
std::ofstream::trunc);

        if (array_goods.empty()) {
            this->lbDeleteInfo->Text = "Данных нет";
        }
    }
}

```



```

        }
        else {
            for (GoodsInterface::Goods it : array_goods)

                printNote(file, it);
        }
    }
    this->lbDeleteInfo->Text = "Запись удалена";
}

void SelectGoods(std::fstream& fileGoods, std::fstream&
fileCart) {
    GoodsInterface::Goods obj;
    std::vector<GoodsInterface::Goods> array_goods;
    char* buffer = new char[128];
    bool fl{};
    long double Cost{};
    long double FinalCost{};
    using namespace System::Runtime::InteropServices;
    buffer =
(char*)(void*)Marshal::StringToHGlobalAnsi(this->tbCartName->Text);
    int selectedObj=-1;
    int goodsCount{};
    this->lbCartSelectInfo->Visible = true;

    if (fileGoods.peek() == EOF) {
        this->lbCartSelectInfo->Text = "Данных нет";
    }
    else {
        while (!fileGoods.eof()) {
            fileGoods >> obj;
            if (obj.getTitle() != "") {
                array_goods.push_back(obj);
            }
        }

        for (int i{}; i < array_goods.size(); i++)
            if (!strcmp(array_goods[i].getTitle(),
buffer)) {
                selectedObj = i;
            }

        if (selectedObj == -1) {
            this->lbCartSelectInfo->Text = "Запись не
найдена";
        }
        else
        {
            if (this->tbCartCount->Text != "") {
                goodsCount = int::Parse(this-
>tbCartCount->Text);

                array_goods[selectedObj].setCount(goodsCount);
                fileCart.seekp(0, std::ios_base::end);
                printNote(fileCart,
array_goods[selectedObj]);
                fileCart.close();
                fileCart.open(CART_SHEET,
std::ios::app);
                for (int i{}; i < array_goods.size();
i++) { //исключение ненужного элемента

```

```

        Cost =
(array_goods[i].getPrice().getCops()) / 100.0 * array_goods[i].getCount() +
        array_goods[i].getPrice().getRubs() * array_goods[i].getCount();
        FinalCost = FinalCost + Cost;
    }
    this->lbSum->Text = "Итого: " +
FinalCost + " руб.";
    Cost = 0;
    FinalCost = 0;
}
}
}

void BuyGoods(std::fstream& fileCart, std::fstream& fileSales)
{
    fileCart.seekp(0, std::ios_base::end);
    fileCart.close();
    fileCart.open(CART_SHEET);
    GoodsInterface::Goods obj;
    std::vector<GoodsInterface::Goods> array_goods;
    int selectedObj{};
    long double Cost{};
    long double FinalCost{};
    char choose{};

    if (fileCart.peek() == EOF) {
        this->lbCartSelectInfo->Text = "Данных нет\n";
    }
    else {
        while (!fileCart.eof()) {
            fileCart >> obj;
            if (obj.getTitle() != "") {
                array_goods.push_back(obj);
            }
        }

        fileSales.seekp(0, std::ios_base::end);
        for (int i{}; i < array_goods.size(); i++) {
            printNote(fileSales, array_goods[i]);
            Cost =
(array_goods[i].getPrice().getCops()) / 100.0 * array_goods[i].getCount() +
            array_goods[i].getPrice().getRubs() * array_goods[i].getCount();
            fileSales << std::endl << "Цена за
единицу: " << Cost;
            FinalCost = FinalCost + Cost;
        }
        fileSales << std::endl << "Итого: " <<
FinalCost << std::endl;
        fileSales.close();
        fileSales.open(SALES_REPORT, std::ios::app);

        fileCart.close();
        fileCart.open(CART_SHEET, std::ios::out);
    }
}

private: System::Void btnEnter_Click(System::Object^ sender,
System::EventArgs^ e) {
    System::String^ ADMIN_LOGIN = "admin";
    System::String^ ADMIN_PASSWORD = "admin1";
}

```

```

System::String^ USER_LOGIN = "user";
System::String^ USER_PASSWORD = "user1";

std::fstream fileGoods;
std::fstream fileCart;
std::fstream fileSales;

setlocale(LC_ALL, "Russian");

int id{};
System::String^ buffer_login{};
System::String^ buffer_password{};

//авторизация

        buffer_login= this->tbAuthLogin->Text;
        buffer_password = this->tbAuthPassword->Text;
        if ((buffer_login == ADMIN_LOGIN) && (buffer_password ==
ADMIN_PASSWORD)) {
                id = 1;
        }
        else if ((buffer_login == USER_LOGIN) &&
(buffer_password == USER_PASSWORD)) {
                id = 2;
        }
        else {
                this->lbAuthError->Visible = true;
                this->lbAuthError->Text = "Неверный логин или
пароль";
        }

//интерфейсы
if (id == 1) {
        AuthTurnOff();
        UserPanelTurnOff();
        AdminPanelTurnOn();
}

if (id == 2) {
        AuthTurnOff();
        AdminPanelTurnOff();
        UserPanelTurnOn();
}
}

private: System::Void btBack_Click(System::Object^ sender,
System::EventArgs^ e) {
        AuthTurnOn();
        UserPanelTurnOff();
        AdminPanelTurnOff();
}

private: System::Void btDeleteNote_Click(System::Object^ sender,
System::EventArgs^ e) {
        std::fstream fileGoods;

        fileGoods.open(GOODS_SHEET);
        deleteNote(fileGoods);
        fileGoods.close();
        reload();
}

private: System::Void btReadSheet_Click(System::Object^ sender,
System::EventArgs^ e) {

```

```

    }

    private: System::Void btAddNote_Click(System::Object^ sender,
System::EventArgs^ e) {
        std::fstream fileGoods;

        fileGoods.open(GOODS_SHEET, std::ofstream::out | std::ios::app);
        addNote(fileGoods);
        fileGoods.close();
        reload();
    }

    private: System::Void btSortPrice_Click(System::Object^ sender,
System::EventArgs^ e) {
        std::fstream fileGoods;

        fileGoods.open(GOODS_SHEET);
        sortPrice(fileGoods);
        fileGoods.close();
        reload();
    }

    private: System::Void btnSortName_Click(System::Object^ sender,
System::EventArgs^ e) {
        std::fstream fileGoods;

        fileGoods.open(GOODS_SHEET);
        sortName(fileGoods);
        fileGoods.close();
        reload();
    }

    private: System::Void btnSortWeight_Click(System::Object^ sender,
System::EventArgs^ e) {
        std::fstream fileGoods;

        fileGoods.open(GOODS_SHEET);
        sortWeight(fileGoods);
        fileGoods.close();
        reload();
    }

    private: System::Void btnSortRating_Click(System::Object^ sender,
System::EventArgs^ e) {
        std::fstream fileGoods;

        fileGoods.open(GOODS_SHEET);
        sortRating(fileGoods);
        fileGoods.close();
        reload();
    }

    private: System::Void btnSortCmpName_Click(System::Object^ sender,
System::EventArgs^ e) {
        std::fstream fileGoods;

        fileGoods.open(GOODS_SHEET);
        sortCompanySeller(fileGoods);
        fileGoods.close();
        reload();
    }

    private: System::Void btnSelectGoods_Click(System::Object^ sender,
System::EventArgs^ e) {
        std::fstream fileGoods;

```

```

        std::fstream fileCart;

        fileGoods.open(GOODS_SHEET, std::ifstream::in);
        fileCart.open(CART_SHEET, std::ofstream::out | std::ios::app);
        SelectGoods(fileGoods, fileCart);
        fileCart.close();
        fileGoods.close();

        fileCart.open(CART_SHEET);
        this->lbCartBuyGoods->Items->Clear();
        readSheetCart(fileCart);
        fileCart.close();
    }

    private: System::Void MyForm_Load(System::Object^ sender,
System::EventArgs^ e) {
        UserPanelTurnOff();
        AdminPanelTurnOff();
        AuthTurnOn();

        std::fstream fileGoods;

        fileGoods.open(GOODS_SHEET);
        this->lbGoods->Items->Clear();
        readSheet(fileGoods);
        fileGoods.close();

        std::fstream fileCart;

        fileCart.open(CART_SHEET);
        this->lbCartBuyGoods->Items->Clear();
        readSheetCart(fileCart);
        fileCart.close();
    }

    private: System::Void btnCartClear_Click(System::Object^ sender,
System::EventArgs^ e) {
        std::fstream fileCart;
        fileCart.open(CART_SHEET, std::ios::out);
        fileCart.close();
        fileCart.open(CART_SHEET);
        this->lbCartBuyGoods->Items->Clear();
        readSheetCart(fileCart);
        fileCart.close();
    }

    private: System::Void btnCartBuyGoods_Click(System::Object^ sender,
System::EventArgs^ e) {
        std::fstream fileCart;
        std::fstream fileSales;
        fileCart.open(CART_SHEET, std::ifstream::in);
        fileSales.open(SALES_REPORT, std::ofstream::out | std::ios::app);
        BuyGoods(fileCart, fileSales);
        fileSales.close();
        fileCart.close();

        reloadCart();
    }

    private: System::Void tbCartCount_KeyPress(System::Object^ sender,
System::Windows::Forms::KeyPressEventArgs^ e) {
        if ((e->KeyChar<'0' || e->KeyChar>'9') && (e->KeyChar != 8)) {
            e->Handled = true;
        }
    }

```

```

    }
    private: System::Void tbNotePrice_KeyPress(System::Object^ sender,
System::Windows::Forms::KeyPressEventArgs^ e) {
        if ((e->KeyChar < '0' || e->KeyChar>'9') && (e->KeyChar != 8)&&(e-
>KeyChar!=',')) {
            e->Handled = true;
        }
    }

    private: System::Void tbNoteWeight_KeyPress(System::Object^ sender,
System::Windows::Forms::KeyPressEventArgs^ e) {
        if ((e->KeyChar < '0' || e->KeyChar>'9') && (e->KeyChar != 8)) {
            e->Handled = true;
        }
    }
};

}

```

### Sort.cpp

```

#include <iostream>
#include <vector>
#include "Goods.h"
#include "Money.h"
#include "Sort.h"

using namespace std;

static const char* GOODS_SHEET = "database/GoodsSheet.txt";
static const char* CART_SHEET = "database/CartSheet.txt";
static const char* SALES_REPORT = "database/SalesReport.txt";

unsigned long int charToInt(const char* x, int k) {
    unsigned long int number = 0;
    int digit = 1, deg = k - 1;
    for (int i = 0; i < k; i++) {
        if (isdigit(x[i])) {
            digit = pow(10, deg) * (int(x[i]) - 48);
            number += digit;
            deg--;
        }
    }
    if (int(x[0]) == 45) {
        number /= -10;
    }
    return number;
}

fstream& operator>>(fstream& file, GoodsInterface::Goods & obj) {
    file.clear();
    string buffer_string{};
    char buffer_word[128] = "";
    bool tempBool = false;

    int j{}, k{};
    char title[128] = "";
    char companyseller[128] = "";
    char addinfo[128] = "";
    unsigned long int count{};
    unsigned long int weight{};
    unsigned long int rating{};
    MoneyInterface::Money price{};
}

```

```

getline(file, buffer_string);
if (buffer_string.empty()) {
    return file;
}
else
{
    for (int i = 0; buffer_string[i] != '\0'; i++) {
        if (buffer_string[i] == '|') {
            j++;
            if (j == 7) {
                count = charToInt(buffer_word, k);
            }
        }
        else
        switch (j) {
            case 0:
                title[i] = buffer_string[i];
                break;
            case 1:
                if (buffer_string[i] == '.') {
                    price.setRubs(charToInt(buffer_word, k));
                    tempBool = true;
                    for (int l = 0; l < k; l++) {
                        buffer_word[l] = '\0';
                    }
                    k = -1;
                }
                else {
                    if (!tempBool) {
                        buffer_word[k] = buffer_string[i];
                    }
                    else {
                        buffer_word[k] = buffer_string[i];
                    }
                }
                k++;
                break;
            case 2:
                if (tempBool) {
                    price.setCops(charToInt(buffer_word, k));
                    for (int l = 0; l < k; l++) {
                        buffer_word[l] = '\0';
                    }
                    k = 0;
                    tempBool = false;
                }
                buffer_word[k] = buffer_string[i];
                k++;
                break;
            case 3:
                if (!tempBool) {
                    weight = charToInt(buffer_word, k);
                    for (int l = 0; l < k; l++) {
                        buffer_word[l] = '\0';
                    }
                    k = 0;
                    tempBool = true;
                }
                buffer_word[k] = buffer_string[i];
                k++;
                break;
            case 4:
                if (tempBool) {
                    rating = charToInt(buffer_word, k);
                    for (int l = 0; l < k; l++) {

```

```

        buffer_word[l] = '\\0';
    }
    k = 0;
    tempBool = false;
}
companyseller[k] = buffer_string[i];
k++;
break;
case 5:
    if (!tempBool) {
        for (int l = 0; l < k; l++) {
            buffer_word[l] = '\\0';
        }
        k = 0;
        tempBool = true;
    }
    addinfo[k] = buffer_string[i];
    k++;
    break;
case 6:
    if (tempBool) {
        for (int l = 0; l < k; l++) {
            buffer_word[l] = '\\0';
        }
        k = 0;
        tempBool = false;
    }
    buffer_word[k] = buffer_string[i];
    k++;
    break;

default:
    cout << "Error 404";
    break;
}
}

obj.setTitle(title);
obj.setPrice(price);
obj.setWeight(weight);
obj.setRating(rating);
obj.setCompanySeller(companyseller);
obj.setAddInfo(addinfo);
obj.setCount(count);

return file;
}

void sortName(fstream& file) {
    GoodsInterface::Goods obj;
    vector<GoodsInterface::Goods> array_goods;
    char* buffer = new char[128];

    if (file.peek() == EOF) {
        cout << "Данных нет\\n";
    }
    else {
        while (!file.eof()) {
            file >> obj;
            if (obj.getTitle() != "") {
                array_goods.push_back(obj);
            }
        }
        for (int j{}; j < array_goods.size(); j++)

```



```

        for (int i{}; i < array_goods.size() - 1; i++) {
            if (strcmp(array_goods[i].getTitle(), array_goods[i +
1].getTitle()) > 0) {
                obj = array_goods[i];
                array_goods[i] = array_goods[i + 1];
                array_goods[i + 1] = obj;
            }
        }
        file.close();
        file.open(GOODS_SHEET, ios::out);
        file.close();
        file.open(GOODS_SHEET, std::ofstream::out | ios::app);
        for (int i{}; i < array_goods.size(); i++) {
            printNote(file, array_goods[i]);
        }
    }
}

void sortPrice(fstream& file) {
    GoodsInterface::Goods obj;
    vector<GoodsInterface::Goods> array_goods;
    char* buffer = new char[128];

    if (file.peek() == EOF) {
        cout << "Данных нет\n";
    }
    else {
        while (!file.eof()) {
            file >> obj;
            if (obj.getTitle() != "") {
                array_goods.push_back(obj);
            }
        }
        for (int j{}; j < array_goods.size(); j++)
            for (int i{}; i < array_goods.size() - 1; i++) { //исключение
нужного элемента
                if (array_goods[i].getPrice().getRubs() > array_goods[i
+ 1].getPrice().getRubs()) {
                    obj = array_goods[i];
                    array_goods[i] = array_goods[i + 1];
                    array_goods[i + 1] = obj;
                }
                else if (array_goods[i].getPrice().getRubs() ==
array_goods[i + 1].getPrice().getRubs())
                    if (array_goods[i].getPrice().getCops() >
array_goods[i + 1].getPrice().getCops())
                    {
                        obj = array_goods[i];
                        array_goods[i] = array_goods[i + 1];
                        array_goods[i + 1] = obj;
                    }
            }
        file.close();
        file.open(GOODS_SHEET, ios::out);
        file.close();
        file.open(GOODS_SHEET, std::ofstream::out | ios::app);
        for (int i{}; i < array_goods.size(); i++) {
            printNote(file, array_goods[i]);
        }
    }
}

void sortWeight(fstream& file) {
    GoodsInterface::Goods obj;
    vector<GoodsInterface::Goods> array_goods;

```

```

char* buffer = new char[128];

if (file.peek() == EOF) {
    cout << "Данных нет\n";
}
else {
    while (!file.eof()) {
        file >> obj;
        if (obj.getTitle() != "") {
            array_goods.push_back(obj);
        }
    }
    for (int j{}; j < array_goods.size(); j++)
        for (int i{}; i < array_goods.size() - 1; i++) { //исключение
ненужного элемента
            if (array_goods[i].getWeight() > array_goods[i +
1].getWeight()) {
                obj = array_goods[i];
                array_goods[i] = array_goods[i + 1];
                array_goods[i + 1] = obj;
            }
        }
    file.close();
    file.open(GOODS_SHEET, ios::out);
    file.close();
    file.open(GOODS_SHEET, std::ofstream::out | ios::app);
    for (int i{}; i < array_goods.size(); i++) {
        printNote(file, array_goods[i]);
    }
}

}

void sortRating(fstream& file) {
    GoodsInterface::Goods obj;
    vector<GoodsInterface::Goods> array_goods;
    char* buffer = new char[128];

    if (file.peek() == EOF) {
        cout << "Данных нет\n";
    }
    else {
        while (!file.eof()) {
            file >> obj;
            if (obj.getTitle() != "") {
                array_goods.push_back(obj);
            }
        }
        for (int j{}; j < array_goods.size(); j++)
            for (int i{}; i < array_goods.size() - 1; i++) { //исключение
ненужного элемента
                if (array_goods[i].getRating() > array_goods[i +
1].getRating()) {
                    obj = array_goods[i];
                    array_goods[i] = array_goods[i + 1];
                    array_goods[i + 1] = obj;
                }
            }
        file.close();
        file.open(GOODS_SHEET, ios::out);
        file.close();
        file.open(GOODS_SHEET, std::ofstream::out | ios::app);
        for (int i{}; i < array_goods.size(); i++) {
            printNote(file, array_goods[i]);
        }
    }
}

```

```

}

void sortCompanySeller(fstream& file) {
    GoodsInterface::Goods obj;
    vector<GoodsInterface::Goods> array_goods;
    char* buffer = new char[128];

    if (file.peek() == EOF) {
        cout << "Данных нет\n";
    }
    else {
        while (!file.eof()) {
            file >> obj;
            if (obj.getTitle() != "") {
                array_goods.push_back(obj);
            }
        }
        for (int j{}; j < array_goods.size(); j++)
            for (int i{}; i < array_goods.size() - 1; i++) { //исключение
нужного элемента
                if (strcmp(array_goods[i].getCompanySeller(),
array_goods[i + 1].getCompanySeller()) > 0) {
                    obj = array_goods[i];
                    array_goods[i] = array_goods[i + 1];
                    array_goods[i + 1] = obj;
                }
            }
        file.close();
        file.open(GOODS_SHEET, ios::out);
        file.close();
        file.open(GOODS_SHEET, std::ofstream::out | ios::app);
        for (int i{}; i < array_goods.size(); i++) {
            printNote(file, array_goods[i]);
        }
    }
}

```

Sort.h

```

#pragma once
void sortName(std::fstream& file);
void sortPrice(std::fstream& file);
void sortWeight(std::fstream& file);
void sortRating(std::fstream& file);
    void sortCompanySeller(std::fstream& file);

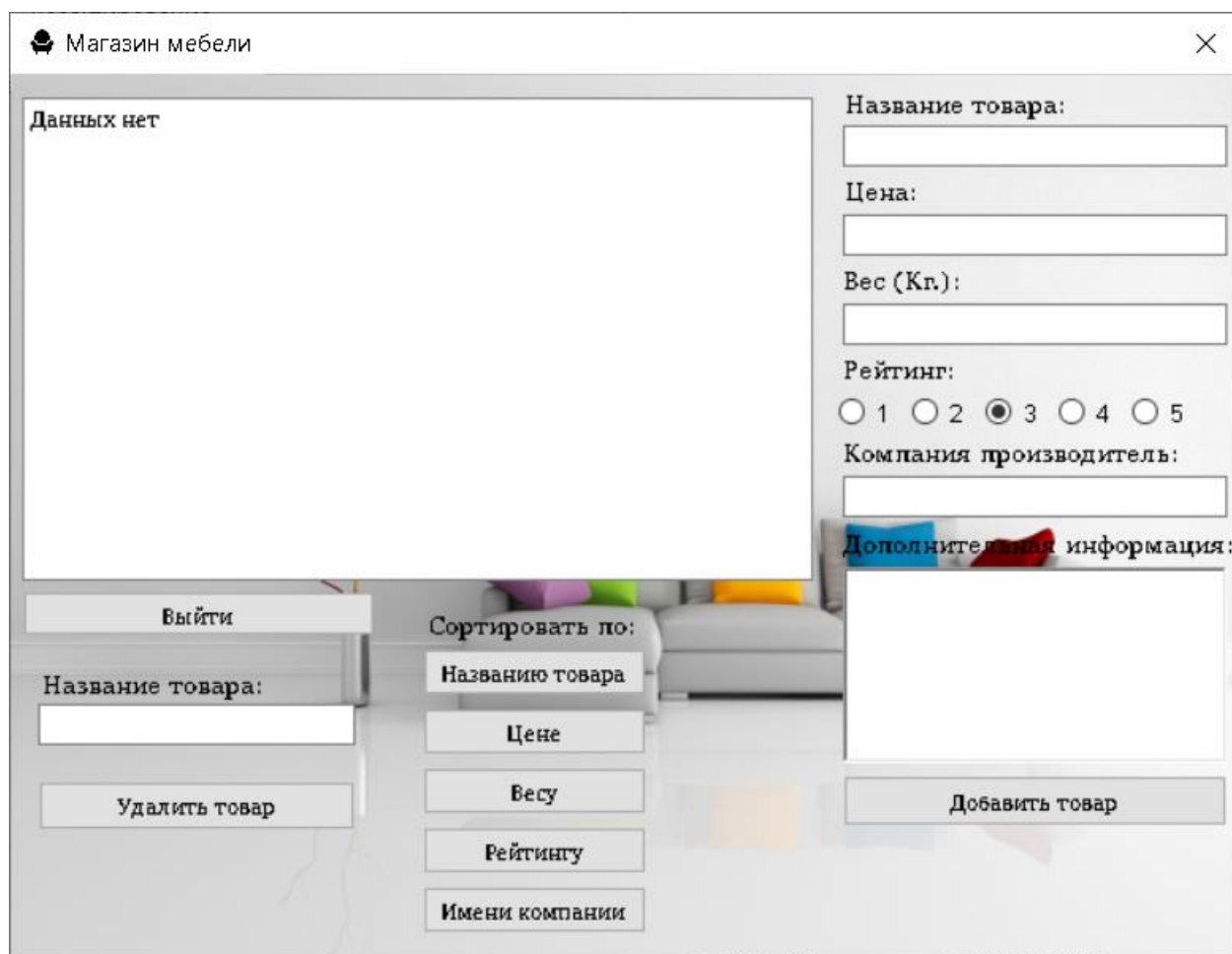
```

## Приложение 2



The initial login window titled "Магазин мебели" (Furniture Store) features a background image of a red and yellow sofa. It contains two input fields labeled "Логин:" (Login) and "Пароль:" (Password), and a "Войти" (Login) button at the bottom.

Рис.1 Начальное окно



The administrator's working window titled "Магазин мебели" (Furniture Store) displays a management interface. On the left, a large area shows "Данных нет" (No data). Below this are buttons for "Выйти" (Logout), "Название товара:" (Product name), and "Удалить товар" (Delete product). In the center, a "Сортировать по:" (Sort by) section includes buttons for "Названию товара" (Product name), "Цене" (Price), "Весу" (Weight), "Рейтингу" (Rating), and "Имени компании" (Company name). On the right, a form for adding a new product includes fields for "Название товара:" (Product name), "Цена:" (Price), "Вес (Кг.):" (Weight in kg), and "Компания производитель:" (Manufacturer company), followed by a 5-point rating scale (radio buttons 1-5, with 3 selected). Below these is a "Дополнительная информация:" (Additional information) field and a "Добавить товар" (Add product) button. The background of the window shows a modern living room with a grey sofa and colorful cushions.

Рис.2 Рабочее окно администратора

Магазин мебели

Шкаф | 20000.0 руб. | 200 кг. | 4 \* | Cabinet inc. | Двудверный шкаф купе

Раскладной диван | 5690.0 руб. | 72 кг. | 4 \* | Pushe Hoff | Берёзовый кар

Тумбочка | 899.0 руб. | 20 кг. | 2 \* | Vox company | Дёшево, ненадёжно |

Итальянский стол | 40000.0 руб. | 100 кг. | 5 \* | Lacasa | Шикарный стол

Двухъярусная кровать | 26990.0 руб. | 80 кг. | 5 \* | Polkamoda | Этажа 2,

Табуретка | 1190.0 руб. | 2 кг. | 2 \* | Г. Табуреткинск | Отсутствует спинка

Выйти

Название товара:

Удалить товар

Сортировать по:

Названию товара

Цене

Весу

Рейтингу

Имени компании

Название товара:

Табуретка

Цена:

1190

Вес (Кг.):

2

Рейтинг:

☐ 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5

Компания производитель:

Г. Табуреткинск

Дополнительная информация:

Отсутствует спинка

Добавить товар

Рис.3 Результат добавления товаров

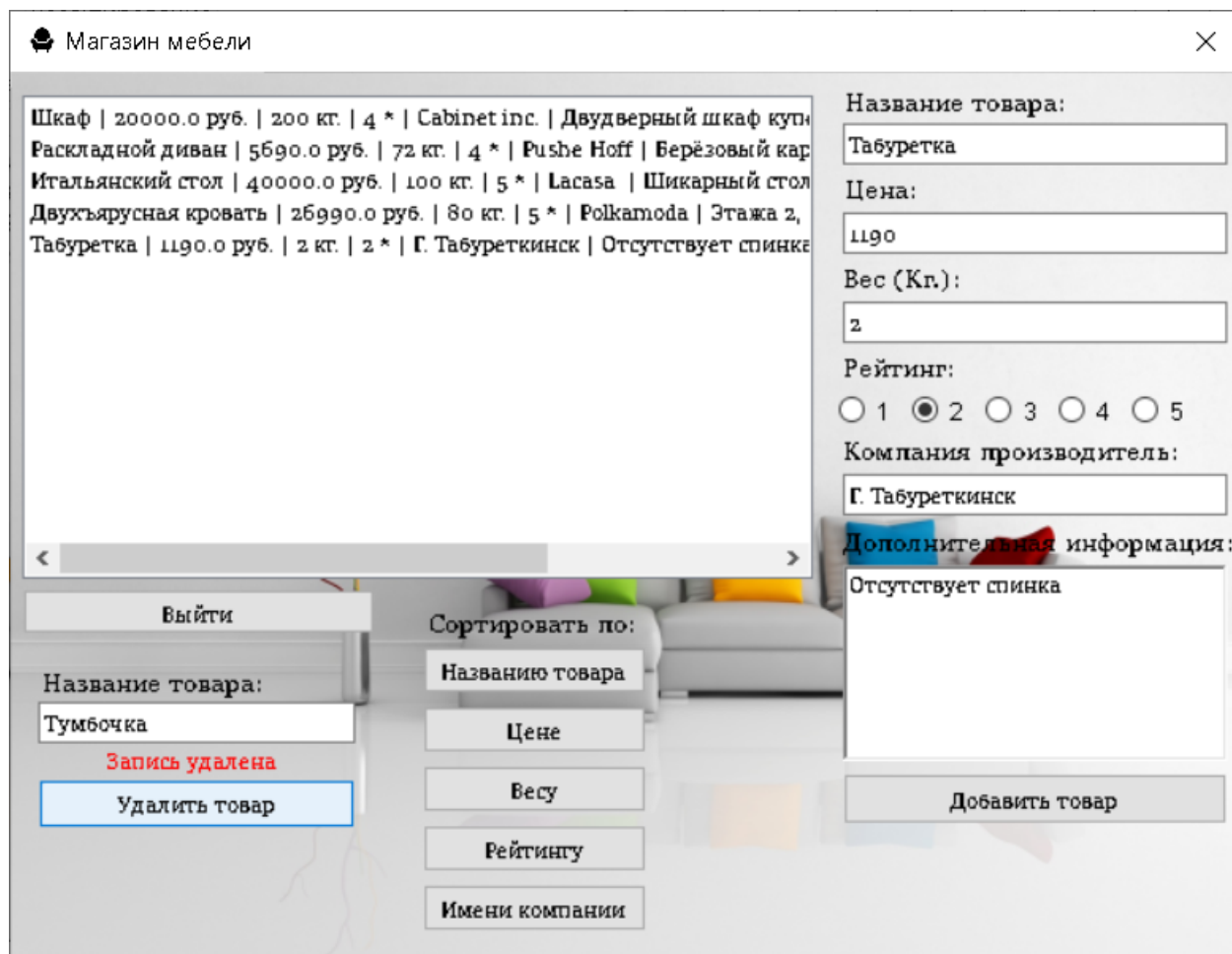


Рис.4 Удаление из списка товаров тумбочки

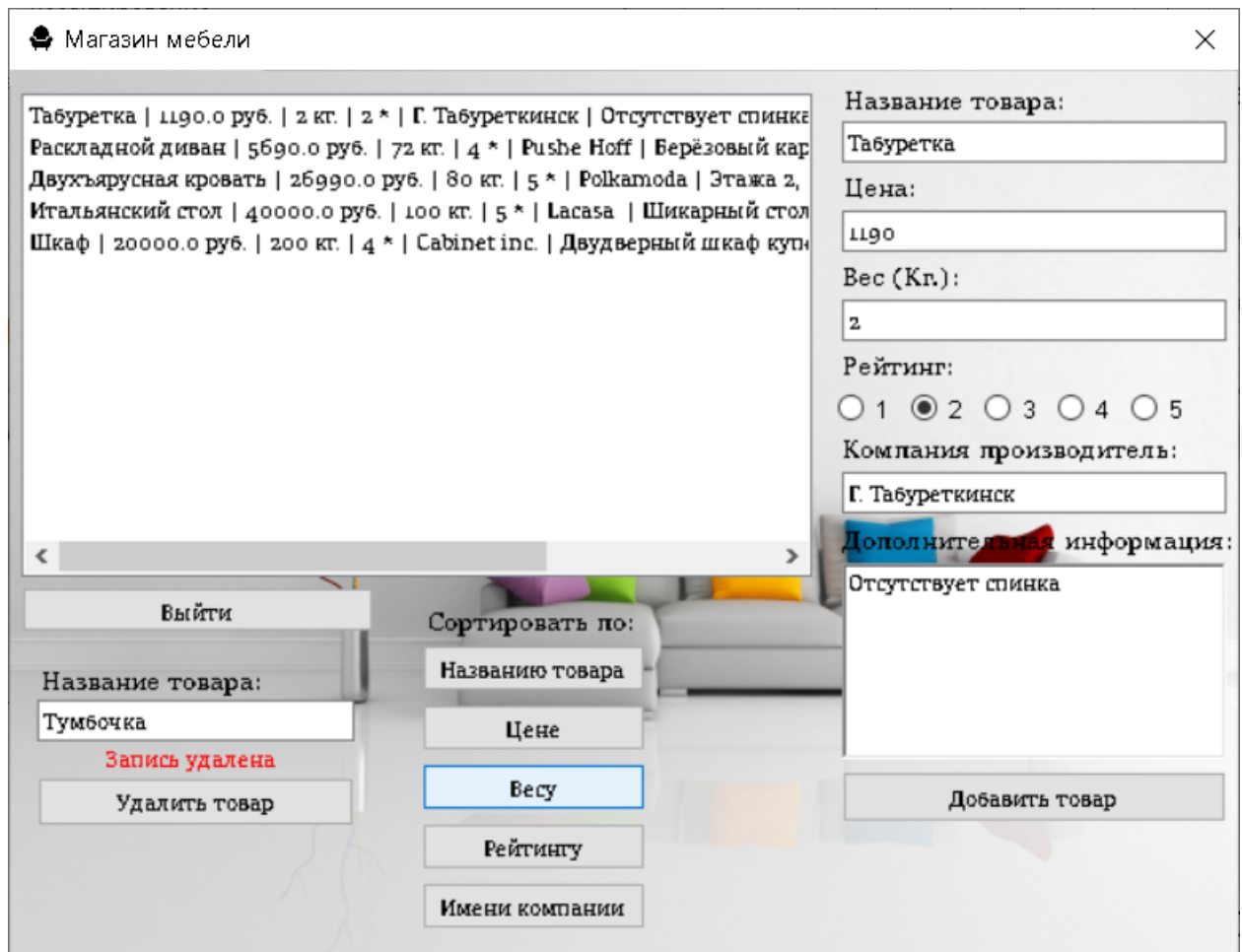


Рис.5 Сортировка товаров по весу



Рис.6 Выход в окно авторизации

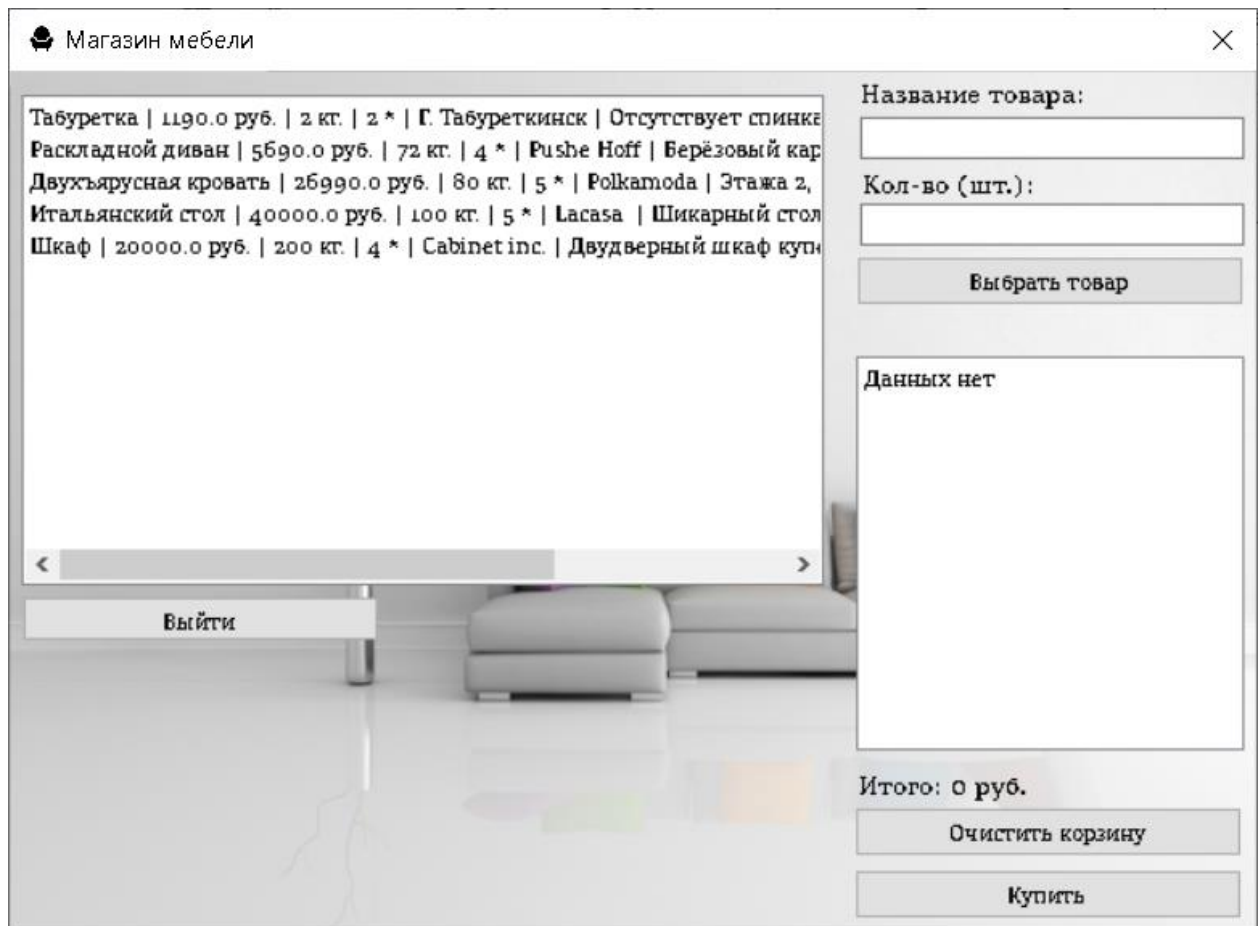


Рис.7 Рабочее окно клиента.

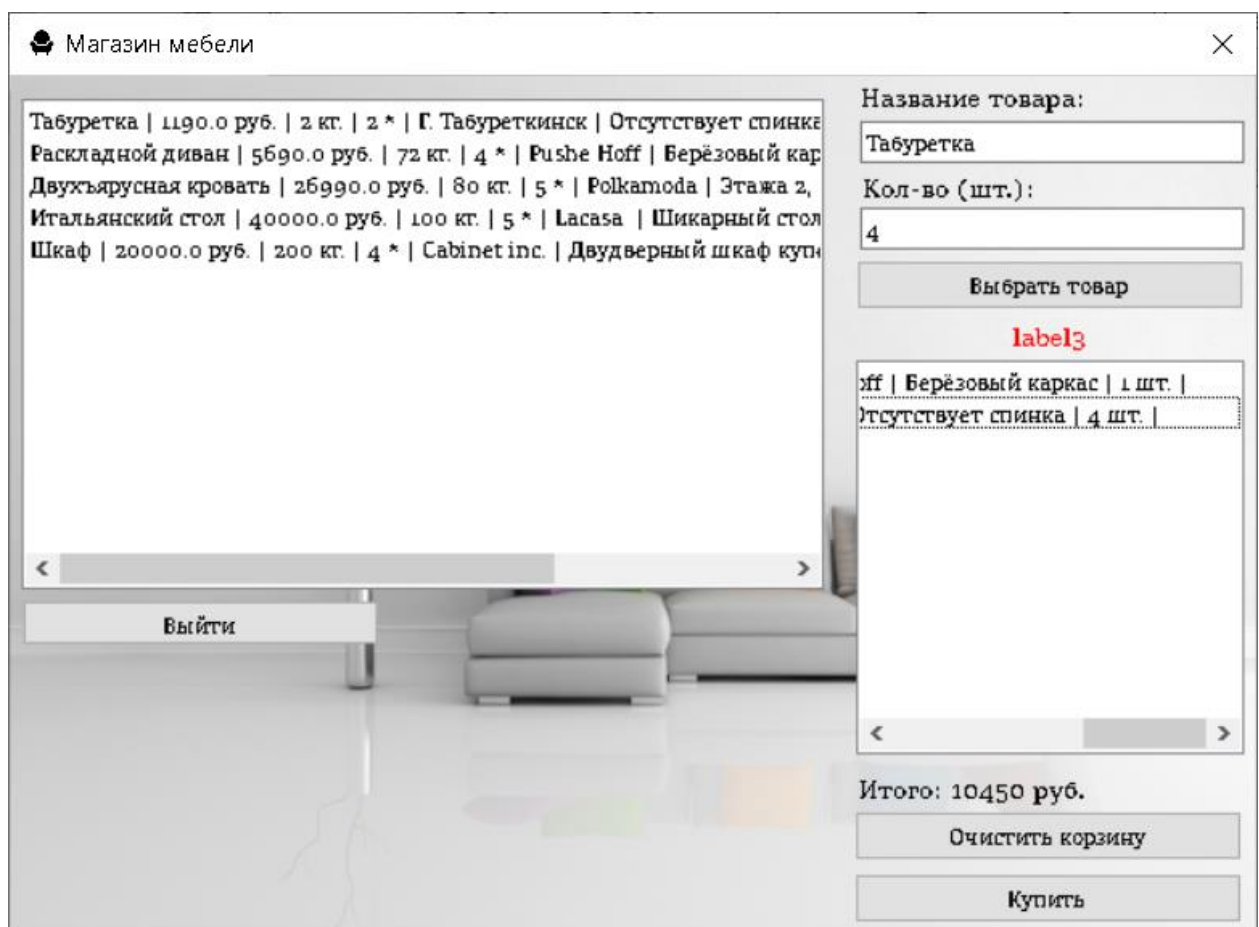


Рис.8 Добавление товаров в корзину



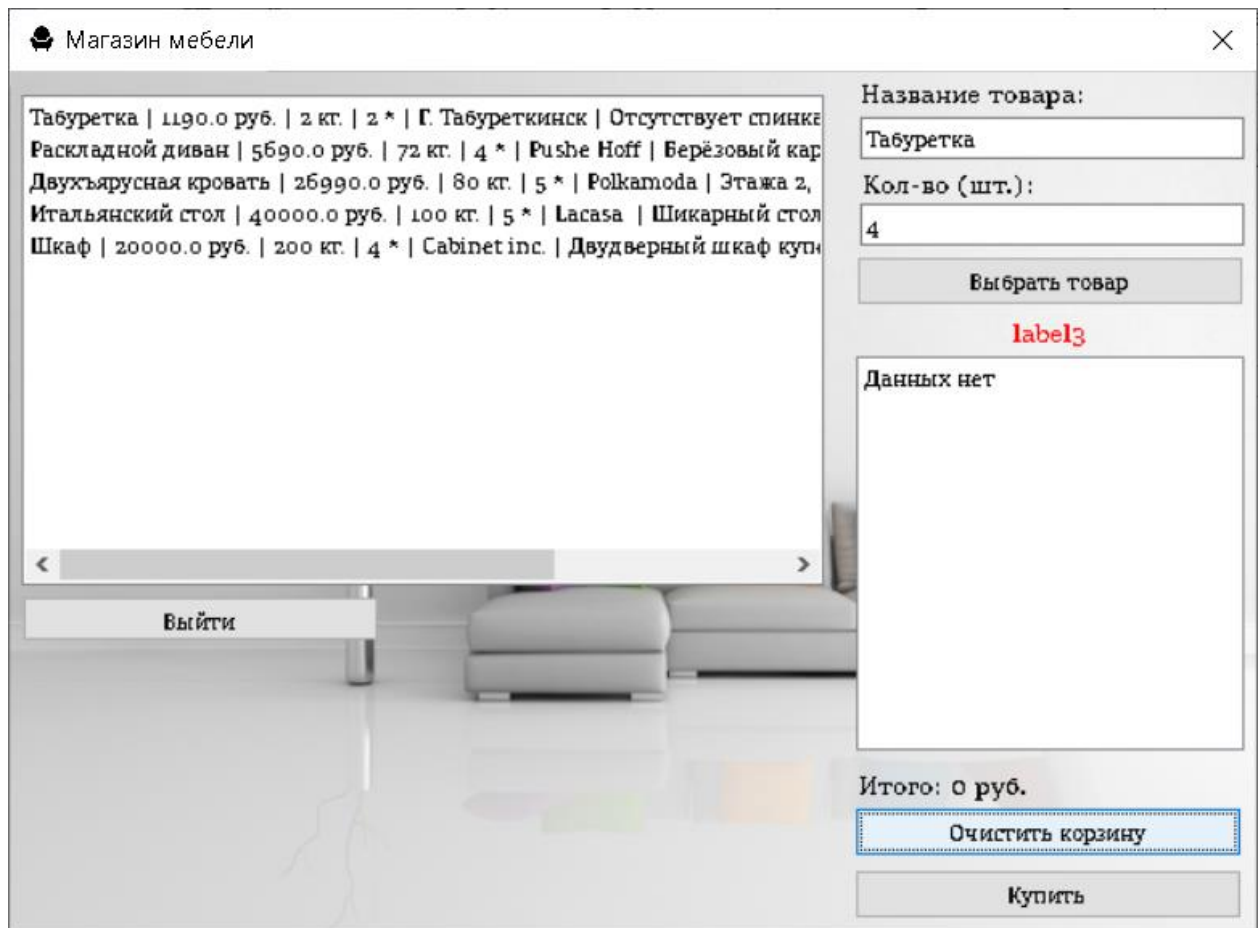


Рис.9 Корзина после отчистки