**Федеральное государственное бюджетное профессиональное образовательное учреждение**

**Торжокский политехнический колледж**

**Федерального агентства по государственным резервам**

**ОТЧЁТ**

о зимней работе

**по профессиональному модулю МДК.01.01**

«**РАЗРАБОТКА ПРОГРАМНЫХ МОДУЛЕЙ**»

наименование профессионального модуля

Специальность ***09.02.07 «Информационные системы и программирование»***

Период с «28» декабря 2024г. по «12» января 2025г.

Выполнили студенты

(ФИО, подпись)

|  |  |
| --- | --- |
| Преподаватель  Бойков А.М  (ФИО, подпись) | оценка |

2024

Оглавление

[Введение 3](#_Toc189758319)

[1. Практическая значимость программного средства 4](#_Toc189758320)

[2. Функциональные требования к программному средству 6](#_Toc189758321)

[3. Модульная структура проектируемого программного средства 8](#_Toc189758322)

[4. Описание алгоритмов 11](#_Toc189758323)

[5. Разработка проекта в Visual Studio C# 15](#_Toc189758324)

# **Введение**

Цель работы – получить практический опыт парного программирования и разработки модульной структуры программы.

Задачи:

1) выявить практическую значимость проектируемого программного средства (далее – ПС) (с точки зрения использования, с точки зрения развития продуктов большего масштаба ~~и т.д. и т.п.~~);

2) обосновать функции разрабатываемого программного средства и варианты его использования (какие задачи позволит решить и кому);

3) спроектировать модульную структуру программного средства (структура классов: данные и методы);

4) описание нескольких алгоритмов методов ~~(как минимум – два)~~, отвечающие непосредственно за решение поставленной задачи;

5) разработать программное средство в среде Visual Studio.

# **1. Практическая значимость программного средства**

Для чего разрабатывается ПС? (тут что-то с отступами)

Данная программа предназначена для учета продаж мобильных телефонов в магазине. Она позволяет вести контроль за товарами, анализировать объемы продаж за различные периоды и формировать отчеты.

Использование программы упрощает процесс ведения учета, снижает вероятность ошибок, возникающих при ручном вводе данных, и ускоряет анализ продаж, что важно для эффективного управления магазином.

Система автоматически загружает данные из файлов, вычисляет остатки товаров, формирует отчеты о проданных устройствах и стоимости продаж за указанный период.

Кто пользователи данного ПС?

1. Менеджеры магазинов – анализируют продажи, выявляют популярные модели, оценивают спрос.

2. Продавцы – проверяют наличие товара перед продажей.

3. Владельцы бизнеса – оценивают прибыльность продаж, строят стратегию закупок.

Программа предназначена для небольших и средних магазинов, торгующих мобильными устройствами. Она помогает автоматизировать процессы учета, что особенно важно при большом количестве товаров.

Если бы данного ПС не было, то…

• Приходилось бы вести учет продаж вручную, например, в таблицах Excel. Это требует больше времени и повышает риск ошибок.

• Сложнее было бы быстро узнать остатки товара, особенно при большом ассортименте.

• Анализ продаж за период (день, неделю, месяц) потребовал бы дополнительных расчетов.

• Составление отчетов стало бы более трудоемким, а в случае ошибки

# **2. Функциональные требования к программному средству**

Программа для учета продаж мобильных телефонов должна обеспечивать выполнение следующих функций:

1. Основные возможности ПС:

• Загрузка данных о товарах

Программа загружает список товаров из файла. Каждый товар содержит:

• Марку (бренд)

• Модель

• Цену за единицу

• Категорию (например, “смартфон”, “раскладушка” и т. д.)

Требуется привести описание формата такого файла и пример

Требуется описать, как пользователь может воспользоваться данной функцией

• Загрузка данных о продажах

Загружается информация о продажах за разные даты. Каждая продажа содержит:

• Марку

• Модель

• Дату продажи

• Количество проданных единиц

• Отображение остатков товаров на складе

Программа анализирует загруженные данные и вычисляет, сколько единиц каждого товара осталось после продаж???. (это же значение описывается в файле или нет?). Если нет, тогда нужно убрать упоминание остатка из структуры файла, описать вычисление остатка в рамках отдельного требования

Требуется также привести описание формата такого исходного файла и пример

• Формирование отчета о продажах за заданный период

Пользователь может выбрать временной интервал (~~например~~ (требуются указать точные варианты периодов, которые может указать пользователь), за неделю или месяц), и программа выдаст список проданных товаров с указанием их количества и дат продаж.

Привести пример, какой должен быть вывод программы в ответ на запрос пользователя

Привести описание возможных запросов

Пропущен расчет и вывод стоимости проданного

• Сохранение отчета в файл

Программа позволяет сохранить отчет о продажах в текстовый файл для дальнейшего использования.

Требуется описать структуру информации и формат такого файла

Требуется описать, куда должно производиться сохранение

Требуется описать реакцию программы на ситуацию, когда файл не может быть успешно сохранён или не сохранился

2. Взаимодействие пользователя с программой:

Соотнести описание данного пункта с каждым отдельным требованиям и перенести их в соответствующее требование

Программа работает в консольном режиме. Пользователь взаимодействует с ней через меню:

1. Просмотр остатков товаров.

2. Формирование отчета о продажах за определенный период.

3. Сохранение отчета в файл.

4. Выход из программы.

При выборе формирования отчета система запрашивает у пользователя диапазон дат, после чего отображает найденные продажи в консоли и/или сохраняет их в файл.

3. Ошибки и валидация данных:

Соотнести описание данного пункта с каждым отдельным требованиям и перенести их в соответствующее требование

Программа должна обрабатывать ошибки ввода и некорректные данные:

• Если файл с товарами или продажами отсутствует, программа уведомляет пользователя. (привести конкретный текст сообщения)

• Если в файле с данными есть ошибки (неверный формат цены, даты, количества), программа пропускает такие строки и сообщает об этом.

Привести примеры некорректных строк и вывода сообщений об ошибках

• Если введены некорректные даты для отчета, программа просит повторить ввод.

Описать, что значит «некорректные» даты, привести описание поддерживаемых дат

4. Пример работы программы:

1. Пользователь запускает программу.

2. Выбирает пункт “Показать остатки товаров”.

3. Программа загружает данные, рассчитывает остатки и выводит их на экран.

4. Пользователь выбирает “Формирование отчета”, вводит начальную и конечную даты.

5. Программа находит все продажи за этот период и показывает их.

6. Если нужно, пользователь сохраняет отчет в файл.

7. При завершении работы программа корректно закрывается.

Программа обеспечивает удобство учета и анализа продаж, минимизируя ручные операции и повышая точность отчетов.

# **3. Модульная структура проектируемого программного средства**

**Раздел должен начинаться диаграммой классов!**

**Поля и методы должны быть представлены в нотации C#**

Программное средство построено на основе модульной структуры, включающей несколько классов, каждый из которых выполняет строго определенные функции. Это обеспечивает удобство расширения, поддержку кода и его повторное использование.

**Структура классов:**

Программа состоит из следующих классов:

• Класс Product – представляет собой модель товара. Он содержит поля для хранения информации о марке (бренде) товара, модели, цене за единицу и категории товара. Этот класс необходим для хранения и обработки данных о продукции.

• Класс Sale – хранит данные о продаже товара. В нем содержатся поля для хранения информации о марке товара, модели, дате продажи и количестве проданных единиц. Этот класс используется для работы с продажами, фильтрации данных по датам и формирования отчетов.

• Класс Store – основной управляющий класс, который объединяет списки товаров и продаж, ~~а также реализует методы загрузки данных, расчета остатков и формирования отчетов.~~  (в настоящий момент класс перегружен. Требуется выделить отдельные классы для загрузки данных, для сохранения отчета, классы для проверок исходных файлов и данных, вводимых пользователем с клавиатуры, класс для построения навигации пользователя – это может быть класс Program)

• Класс ReportGenerator – отвечает за создание и сохранение отчетов в файлы.

**Описание классов и их методов:**

Класс Product включает в себя следующие поля:

• Brand – строковое поле, представляющее марку (бренд) товара.

• Model – строковое поле, содержащее модель товара.

• Price – поле типа decimal, представляющее цену товара. Используется decimal, так как он позволяет точно работать с денежными значениями.

• Category – строковое поле, указывающее категорию товара (например, “смартфон”, “раскладушка”).

Класс Sale содержит следующие поля:

• Brand – строковое поле, содержащее марку товара.

• Model – строковое поле, в котором хранится модель товара.

• SaleDate – поле типа DateTime, представляющее дату продажи товара. Использование DateTime позволяет удобно фильтровать и анализировать продажи по временным диапазонам.

• Quantity – поле типа int, представляющее количество проданных единиц товара.

Класс Store реализует методы для работы с товарами и продажами:

• LoadProducts(string filePath) – метод, загружающий список товаров из файла. Он проверяет корректность данных и создает объекты класса Product.

Необходимо проверять успешность загрузки, метод должен возвращать bool (true – данные успещно загружены; false – данные не удалось загрузить)

• LoadSales(string filePath) – метод, загружающий информацию о продажах. Он читает файл, проверяет формат данных (за проверку структуры файла и содержащейся в ней информации должен отвечать отдельный класс, условный SalesFileValidator) и создает объекты класса Sale.

• ShowStock() – метод, вычисляющий и отображающий остатки товаров. Он анализирует продажи каждого товара и рассчитывает, сколько единиц осталось??? (На основе какой информации? Остаток не описан ни в рамках класса Product, ни в рамках класса Sale) на складе.

• GetSalesReport(DateTime start, DateTime end) – метод, фильтрующий список продаж и возвращающий только те записи, которые соответствуют указанному диапазону дат.

Класс ReportGenerator содержит метод GenerateReport(List<Sale> sales, string filePath), который принимает список продаж и записывает их в файл. Это позволяет отделить логику формирования отчетов от бизнес-логики программы.

**Взаимосвязь классов:**

**Обновить описание с учетом комментариев, вынесенных выше**

Класс Store является центральным элементом системы, так как он управляет товарами и продажами. Он содержит списки объектов Product и Sale, а также предоставляет методы для их обработки.

Когда программа запускается, класс Store загружает информацию о товарах и продажах с помощью методов LoadProducts() и LoadSales().

Если пользователь хочет посмотреть остатки товаров, вызывается метод ShowStock(), который анализирует список продаж и рассчитывает оставшееся количество каждого товара.

Если необходимо сформировать отчет о продажах, пользователь вводит диапазон дат, после чего вызывается метод GetSalesReport(), который выбирает только те продажи, которые произошли в этот период. Эти данные передаются в ReportGenerator, который сохраняет их в файл.

**Логика работы классов:**

1. При запуске программы создается объект класса Store.

2. Вызываются методы LoadProducts() и LoadSales(), загружающие данные из файлов.

3. Пользователь выбирает действие из меню:

Если он хочет просмотреть остатки товаров, программа вызывает метод ShowStock(), который анализирует список продаж и выводит количество оставшихся товаров.

Если он хочет сформировать отчет о продажах за определенный период, программа запрашивает начальную и конечную даты, вызывает метод GetSalesReport(), а затем передает данные в ReportGenerator для сохранения в файл.

4. После выполнения выбранного действия программа возвращается в главное меню, предлагая выбрать следующее действие.

Такая структура обеспечивает удобство использования, легкость масштабирования и четкое разделение логики на отдельные компоненты.

# **4. Описание алгоритмов**

**Требуется внести к пояснениям алгоритмов блок-схемы алгоритмов**

В данном разделе описаны ключевые алгоритмы, реализованные в программном средстве. Минимальное требование – два алгоритма, непосредственно решающих поставленные задачи. В программе используются алгоритмы: алгоритм вычисления остатков товаров на складе, алгоритм формирования отчета о продажах за указанный период

Каждый алгоритм будет описан пошагово, с пояснением его работы.

**Алгоритм вычисления остатков товаров на складе:**

Назначение:

Данный алгоритм позволяет определить количество оставшихся товаров на складе, учитывая проданные единицы.

Исходные данные: список товаров, загруженный из файла (марка, модель, цена, категория). Список продаж, содержащий информацию о проданных товарах (марка, модель, дата, количество).

**Невозможно на основе представленной исходной информации произвести расчет остатка на складе!!! Алгоритм нереализуем. Требуется привести описание реализуемого алгоритма**

Описание алгоритма:

Алгоритм вычисления остатков товаров проходит по всему списку товаров, устанавливая начальное количество (как, если этой информации нет???) каждого товара. Затем он проверяет, имеются ли продажи этой модели в списке продаж, суммируя количество проданных единиц. После этого рассчитывается остаток путем вычитания проданного количества из общего запаса. Итоговые данные выводятся на экран, включая название товара, его цену и оставшееся количество.

Псевдокод алгоритма:

Для каждого товара в списке товаров:

Установить sold = 0 (количество проданных единиц)

Для каждой продажи в списке продаж:

Если модель товара совпадает с моделью в продаже:

Увеличить sold на количество проданных единиц

Вычислить остаток: stock = начальное количество - sold

Вывести марку, модель, цену и остаток

**Алгоритм формирования отчета о продажах за указанный период:**

Назначение:

Данный алгоритм фильтрует продажи по заданному диапазону дат и формирует отчет.

Исходные данные: список продаж (марка, модель, дата продажи, количество). Две даты, введенные пользователем (начальная и конечная).

Описание алгоритма:

Алгоритм формирования отчета сначала запрашивает у пользователя начальную и конечную дату, после чего проверяет корректность введенных данных. Затем он перебирает список продаж и отбирает те, что попадают в указанный диапазон. Если такие продажи есть, их список выводится в консоль или передается в модуль сохранения отчета, иначе программа уведомляет пользователя о том, что за данный период продаж не было.

Псевдокод алгоритма:

Запросить начальную и конечную дату у пользователя

Если начальная дата больше конечной:

Вывести сообщение об ошибке

Запросить ввод заново

Создать пустой список reportSales

Для каждой продажи в списке продаж:

Если дата продажи находится в заданном диапазоне:

Добавить продажу в reportSales

Если reportSales не пуст:

Вывести продажи в консоль

Иначе: вывести сообщение "Продаж за указанный период нет"

**Графическое представление алгоритмов:**

Блок-схема алгоритма вычисления остатков товаров.

Конец

Обрабатываем следующий товар

Выводим информацию

Вычисляем остаток

Ищем продажи данного товара и суммируем

Перебираем товары

Загружаем список товаров и продаж

Рисунок 1 - Блок-схема алгоритма вычисления остатков товаров

Блок-схема не соответствует стандарту. В рамках блок-схемы должны быть выделенные структурные элементы программы: начало, конец, блоки операций, блоки условий и циклов. Перебор товаров, очевидно, должен быть отображен на блок-схеме циклом

Блок схема алгоритма формирования отчета о продажах.

Вывести отчет в консоль

Вывести сообщение, что продаж нет

Нет

Да ДаДДДадапвоарворпаоравполрваопрварповалравпДа

Проверить, есть ли продажи в отчете?

Добавить продажу в отчет, если они в диапазоне дат

Вывести сообщение об ошибке

Перебрать продажи за указанный период

Нет

Да

Даты корректны?

Проверить корректность введённых данных

Запросить начальную и конечную дату

Рисунок 2 - Блок схема алгоритма формирования отчета о продажах

Блок-схема не соответствует стандарту. В рамках блок-схемы должны быть выделенные структурные элементы программы: начало, конец, блоки операций, блоки условий и циклов. Условия нарисованы неверно

# **5. Разработка проекта в Visual Studio C#**

**Буду изучать раздел после того, как внесем изменения по комментариям в разделах 2 – 3 – 4. Он должен соотноситься с тем, что описано выше**

Разрабатываемое программное средство было реализовано в среде Visual Studio C#. Оно представляет собой консольное приложение, которое позволяет загружать данные о товарах и продажах, отображать остатки товаров на складе, формировать отчеты о продажах за заданный период и сохранять их в текстовые файлы.

**Описание структуры проекта**

Проект состоит из следующих файлов:

• Program.cs – основной файл, содержащий точку входа и консольное меню для взаимодействия с пользователем.

• Product.cs – класс, описывающий товар (марка, модель, цена, категория).

• Sale.cs – класс, содержащий информацию о продаже (марка, модель, дата продажи, количество).

• Store.cs – основной класс для управления товарами и продажами, содержащий методы загрузки данных, расчета остатков и формирования отчетов.

• ReportGenerator.cs – класс, отвечающий за сохранение отчетов о продажах в файл.

**Реализация ключевых методов**

Основные функции программы реализованы в классе Store. Ниже приведены фрагменты кода, демонстрирующие работу ключевых методов.

Метод загрузки товаров из файла:

public void LoadProducts(string filePath)

{ if (!File.Exists(filePath)) { Console.WriteLine($"Ошибка: Файл {filePath} не найден!"); return; } foreach (var line in File.ReadLines(filePath)) { var parts = line.Split(';'); if (parts.Length != 4 || !decimal.TryParse(parts[2], out decimal price))

{ Console.WriteLine($"Ошибка в строке: {line}"); continue; } Products.Add(new Product { Brand = parts[0], Model = parts[1], Price = price, Category = parts[3] });

} }

Этот метод проверяет существование файла, загружает данные, обрабатывает ошибки ввода и формирует список товаров.

Метод вычисления остатков товаров:

public void ShowStock(){ Console.WriteLine("Остатки товаров:"); foreach (var product in Products) { int sold = Sales.Where(s => s.Model == product.Model).Sum(s => s.Quantity); Console.WriteLine($"{product.Brand} {product.Model} - Цена: {product.Price} руб. Остаток: {sold} шт."); } }

Метод анализирует продажи и рассчитывает, сколько единиц каждого товара осталось в наличии.

Метод формирования отчета о продажах:

public List<Sale> GetSalesReport(DateTime start, DateTime end) { return Sales.Where(s => s.SaleDate >= start && s.SaleDate <= end).ToList(); }

Метод фильтрует список продаж, выбирая только те, которые произошли в заданный период.

Метод сохранения отчета в файл:

public static void GenerateReport(List<Sale> sales, string filePath) { using (StreamWriter sw = new StreamWriter(filePath)) { sw.WriteLine("Отчет о продажах:"); foreach (var sale in sales) { sw.WriteLine($"{sale.Brand} {sale.Model} - {sale.Quantity} шт. Дата: {sale.SaleDate}"); } } Console.WriteLine($"Отчет сохранен в файл {filePath}"); }

Этот метод записывает список продаж в текстовый файл.

**Работа с консольным меню**

Пользователь взаимодействует с программой через консольное меню в Program.cs. Пример интерфейса:

while (true)

{

Console.WriteLine("\nВыберите действие:");

Console.WriteLine("1 - Показать остатки товаров");

Console.WriteLine("2 - Сформировать отчет о продажах");

Console.WriteLine("3 - Сохранить отчет в файл");

Console.WriteLine("4 - Выход");

string choice = Console.ReadLine();

if (choice == "1") { store.ShowStock(); } else if (choice == "2") {

Console.Write("Введите начальную дату (гггг-мм-дд): ");

DateTime start = DateTime.Parse(Console.ReadLine());

Console.Write("Введите конечную дату (гггг-мм-дд): ");

DateTime end = DateTime.Parse(Console.ReadLine());

var report = store.GetSalesReport(start, end);

foreach (var sale in report)

{

Console.WriteLine($"{sale.Brand} {sale.Model} - {sale.Quantity} шт. Дата: {sale.SaleDate}");

}

}

else if (choice == "3")

{

Console.Write("Введите начальную дату (гггг-мм-дд): ");

DateTime start = DateTime.Parse(Console.ReadLine());

Console.Write("Введите конечную дату (гггг-мм-дд): ");

DateTime end = DateTime.Parse(Console.ReadLine());

Console.Write("Введите имя файла: ");

string filePath = Console.ReadLine();

ReportGenerator.GenerateReport(store.GetSalesReport(start, end), filePath);

} else if (choice == "4") { break; } else { Console.WriteLine("Ошибка: неверный ввод!"); } }

Это консольное меню позволяет пользователю выбирать действия: просмотр остатков, формирование отчета или его сохранение в файл.

**Тестирование программы**

Программа была протестирована в нескольких сценариях:

1. Корректная загрузка данных – проверена обработка файлов с товарами и продажами.

2. Вывод остатков – проверено вычисление количества оставшихся товаров.

3. Фильтрация по дате – успешно сформированы отчеты за разные периоды.

4. Обработка ошибок – протестированы случаи отсутствия файлов, некорректного ввода данных и пустых продаж.

Пример работы программы в консоли:

Выберите действие:

1 - Показать остатки товаров

2 - Сформировать отчет о продажах

3 - Сохранить отчет в файл

4 - Выход

> 1

Остатки товаров:

Samsung Galaxy S21 - Цена: 49999 руб. Остаток: 5 шт.

Apple iPhone 13 - Цена: 79999 руб. Остаток: 3 шт.

Выберите действие:

> 2

Введите начальную дату (гггг-мм-дд): 2024-01-01

Введите конечную дату (гггг-мм-дд): 2024-01-31

Samsung Galaxy S21 - 2 шт. Дата: 2024-01-10

Apple iPhone 13 - 1 шт. Дата: 2024-01-15

5. Вывод

Разработанное программное средство позволяет автоматизировать учет продаж мобильных телефонов. Оно загружает данные из файлов, отображает остатки товаров, формирует отчеты по продажам за любой период и сохраняет их в текстовые файлы. Благодаря модульной структуре программа легко расширяется, а использование консольного интерфейса делает ее простой в использовании.