МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Тверской государственный технический университет» (ТвГТУ)

Кафедра “Программного обеспечения”

ЛАБОРАТОРНАЯ РАБОТА №1.

Контракты в интерфейсе

по дисциплине «Основы программирования»

Разработали: студенты группы

Б.ПИН.РИС.24.06

Думчев Данила Сергеевич

Маматов Михаил Георгиевич

Ратхоре Александр Шарванович

Балабуев Никита Сергеевич

Проверила:

Лисничук Арина Бахытжановна

Тверь, 2025

Оглавление

[Введение 3](#_Toc212744407)

[Часть I. 4](#_Toc212744408)

[Операция 1: Добавить новый товар 4](#_Toc212744409)

[Операция 2: "Отгрузить товар со склада" 6](#_Toc212744410)

[Операция 3: Переместить товар между складами 9](#_Toc212744411)

[Реализация 11](#_Toc212744412)

[Вывод 15](#_Toc212744413)

# Введение

Цели:

1. Научиться формулировать спецификации операций в виде предусловий (Pre) и постусловий (Post) и показывать их выполнение прямо в интерфейсе.
2. Сделатьвизуальный калькулятор wp (слабейшего предусловия).
3. Построить интерактивную проверку корректности цикла. Формулировка инвариантов, задавание и контроль вариант-функции, проверка инварианта через движок wp
4. Сделать инструмент, который задаёт булеву функцию **по номеру** и/или **по формуле**, строит таблицу истинности, генерирует DNF и KNF (в базисе ¬, ∧, ∨), проверяет **эквивалентность** двух функций (по таблицам).

Задачи:

1. Создать настольное приложение с использованием WPF и архитектурного подхода MVVM.
2. Создать вкладку «Операции». В ней реализовать 3 операции согласно «сюжету».
3. Создать движок калькулятора слабейшего предусловия (wp).
4. Создать вкладку «Калькулятор wp», где будет использовать движок калькулятора.
5. Создать вкладку «Циклы», где будут задаваться инварианты, вариант-функции, проверка через wp движок.
6. Создать вкладку «Логические функции». В этой вкладке будут еще 3 вкладки: «По номеру», «По формуле», «Сравнение».

# Часть I.

## Операция 1: Добавить новый товар

**Назначение:** Добавить на склад новый товар с указанным количеством.

**Параметры:** Название товара (строка), Количество (целое число).

**1. Предусловие (Pre-condition)**

Набор условий, которые должны быть истинными **до** начала выполнения операции. Если любое из них нарушено, операция должна быть прервана.

1. **Название не пустое:** Введенная строка, содержащая название товара, не является пустой или состоит только из пробельных символов.
2. **Количество положительное:** Значение количества (Q) должно строго превышать ноль (Q>0).
3. **Уникальность:** На складе **отсутствует** товар с таким же названием.

**2. Постусловие (Post-condition)**

Набор утверждений, которые должны стать истинными **после** успешного выполнения операции.

1. **Появление товара:** В основном списке товаров склада появился новый элемент с заданным названием и количеством.
2. **Инвариант уникальности:** Общее количество уникальных товаров на складе (размер списка) увеличилось ровно на 1.
3. **Сохранность данных:** Введенные название и количество совпадают с данными в новом элементе списка.

**3. Эффекты и Исключения**

| Сценарий | Эффект/Действие | Исключение/Сообщение | Индикатор Pre | Индикатор Post |
| --- | --- | --- | --- | --- |
| **Успешное выполнение** (Pre выполнено) | Вызывается метод Warehouse.AddNewGood(), товар добавляется в список. | Успешное сообщение (MessageBox.Show("Товар добавлен")). | **Зеленый** | **Зеленый** |
| **Неудача** (Pre не выполнено) | Операция прерывается (товар не добавляется). | Генерируется исключение ArgumentException или другое сообщение об ошибке. | **Красный** | **Красный** |

**4. Граничные примеры**

Граничные примеры демонстрируют правильное поведение системы как при номинальном (стандартном) вводе, так и на критических границах, где Pre-условие нарушается.

| Тип | Входные данные | Ожидаемое поведение | Результат (Индикаторы) |
| --- | --- | --- | --- |
| **Валидный (Номинальный)** | "Компьютерная мышь", 10 шт | Операция должна быть выполнена. Товар **добавляется** на склад. | **Pre: Зеленый** (Pre-условие выполнено) **Post: Зеленый** (Post-условие выполнено) |
| **Невалидный (Граничный)** | "Карандаш", 0 шт (Проверка Q>0) | Операция должна быть **отклонена**, так как количество не является положительным числом. | **Pre: Красный** (Количество ≤0) **Post: Красный** (Операция не выполнялась) |
| **Невалидный (Граничный)** | "Монитор", 5 шт (После того, как "Монитор", 100 шт уже добавлен) | Операция должна быть **отклонена**, так как товар с названием "Монитор" уже существует на складе. | **Pre: Красный** (Нарушено условие уникальности) **Post: Красный** (Операция не выполнялась) |

## Операция 2: "Отгрузить товар со склада"

**Pre-условия**

1. Название товара не пустое - обязательное указание существующего товара
2. Количество товара > 0 - должна быть указана положительная сумма отгрузки
3. Товар существует на складе - наименование должно присутствовать в системе
4. Достаточное количество для отгрузки - текущий запас ≥ запрашиваемому количеству

**Post-условия**

1. Количество товара уменьшено - значение уменьшено на указанную величину
2. Товар удален при нулевом остатке - если количество ≤ 0, товар исключается из списка
3. Целостность данных сохранена - остальные товары не затронуты

**Эффекты**

* Успешное выполнение: количество уменьшено, интерфейс обновлен
* Частичное выполнение: товар удален при достижении нуля
* Исключения:
  + ArgumentException - некорректный формат ввода
  + InvalidOperationException - недостаточно товара или товар не найден

**Граничные примеры**

**Пример 1**: Частичная отгрузка товара

Текущее состояние: "Мышь", 10  
Входные данные: "Мышь", 3  
Ожидаемое поведение:

* Pre-условия выполнены (товар есть, количество достаточно)
* Количество уменьшается до 7
* Товар остается в списке
* Post-условия выполняются успешно

**Пример 2**: Полная отгрузка с удалением товара

Текущее состояние: "Клавиатура", 2  
Входные данные: "Клавиатура", 2  
Ожидаемое поведение:

* Pre-условия выполнены
* Количество уменьшается до 0
* Товар удаляется из списка (количество ≤ 0)
* Post-условие "удаление при нуле" выполняется
* Список товаров обновляется

Обработка некорректного формата

Входные данные: аааааааа, о  
Поведение:

* ArgumentException с сообщением о ожидаемом формате
* Операция прерывается до исправления ввода

Нарушение бизнес-правил

Поведение:

* Pre-условия не выполняются
* Операция отклонена на этапе валидации
* Пользователь уведомлен о конкретной причине отказа

Системные исключения

Поведение:

* Отображение пользователю понятного сообщения об ошибке
* Сохранение целостности данных системы
* Возможность повтора операции после исправления ошибок

## Операция 3: Переместить товар между складами

**Назначение:** Переместить заданное количество товара с одного склада на другой.

**Параметры:** Название товара (строка), Количество (целое число), Склад-источник (не реализован в текущей версии, используется как концепт), Склад-получатель (не реализован в текущей версии, используется как концепт).

**1. Предусловие (Pre-condition)**

Набор условий, которые должны быть истинными **до** начала выполнения перемещения.

1. **Различие складов:** Склад-источник и Склад-получатель не должны совпадать.
2. **Количество положительное:** Количество (Q), которое нужно переместить, должно быть положительным числом (Q>0).
3. **Наличие товара:** Товар с указанным названием должен существовать на **Складе-источнике**.
4. **Достаточность количества:** Количество, которое нужно переместить, не должно превышать текущий остаток этого товара на **Складе-источнике**.

**2. Постусловие (Post-condition)**

Утверждения, которые должны стать истинными **после** успешного выполнения операции.

1. **Изменение источника:** Количество товара на **Складе-источнике** уменьшилось на заданную величину.
2. **Изменение получателя:** Количество товара на **Складе-получателе** увеличилось на заданную величину.
3. **Инвариант:** Общее количество товара в системе (сумма по всем складам) **не изменилось**.

**3. Эффекты и Исключения**

| Сценарий | Эффект/Действие | Исключение/Сообщение | Индикатор Pre | Индикатор Post |
| --- | --- | --- | --- | --- |
| **Успешное выполнение** (Pre выполнено) | Значения количества товара на складах-источнике и получателе обновлены. | Успешное сообщение (например, "Перемещение выполнено"). | **Зеленый** | **Зеленый** |
| **Неудача** (Pre не выполнено) | Операция прерывается, состояние складов **не меняется**. | Генерируется исключение или сообщение об ошибке (например, "Недостаточно товара"). | **Красный** | **Красный** |

**4. Граничные примеры**

Граничные примеры особенно важны для этой операции, так как они проверяют логику баланса и существования.

| Тип | Входные данные | Нарушенное Pre-условие | Ожидаемое поведение |
| --- | --- | --- | --- |
| **Валидный (Номинальный)** | Товар: "Стул", Кол-во: 5 (Источник: 10 стульев; Получатель: 0 стульев) | — | Операция должна быть выполнена. Исход: 5 стульев; Получатель: 5 стульев. |
| **Невалидный (Граничный)** | Товар: "Стол", Кол-во: 10 (Источник: 8 столов) | **Достаточность количества:** На складе-источнике нет 10 столов. | Операция должна быть **отклонена**. Склады не меняются. |
| **Невалидный (Критический)** | Товар: "Ковер", Кол-во: 1 (Источник: 0 ковров) | **Наличие товара:** Товар отсутствует на складе-источнике. | Операция должна быть **отклонена**. Склады не меняются. |

# Реализация

**1. Архитектура и разделение ответственности**

Проект реализован как настольное приложение на платформе **WPF** (Windows Presentation Foundation) с использованием языка **C#**. В качестве основы использован паттерн **Model-View-ViewModel (MVVM)**, который обеспечивает четкое разделение предметной области, бизнес-логики и пользовательского интерфейса.

* **Model (Класс предметной области):** Представлен статическим классом **Warehouse** (в проекте ClassLibrary). Этот класс инкапсулирует все данные о складе (список товаров) и содержит ключевую бизнес-логику: методы для добавления (AddNewGood), проверки (CheckAddValid) и получения текстовых формулировок контрактов (GetPreConditionText, GetPostConditionText).
* **View (Пользовательский интерфейс):** Представлен файлами **MainWindow.xaml** (главное окно операций) и **ContractWindow.xaml** (модальное окно контракта). Отвечает исключительно за отображение данных и реакцию на действия пользователя.
* **ViewModel (Логика отображения):** В текущей реализации логика представления и управления состоянием интерфейса (динамические индикаторы, обработка нажатий) сосредоточена в файле **MainWindow.xaml.cs** (Code-Behind). *Для дальнейшего развития рекомендуется вынести эту логику в отдельные классы ViewModel, по одному на каждую операцию, согласно принципам MVVM.*

**2. Реализация контрактов в коде**

**A. Проверка Предусловия (Pre-condition)**

Проверка Pre-условия выполнена в два этапа:

1. **В классе Model (Warehouse.cs):** Для каждой операции (например, Добавить новый товар) создан метод-валидатор (public static bool CheckAddValid(Good good)). Этот метод содержит все логические проверки, соответствующие формализованным Pre-условиям (например, good.quantity > 0 и уникальность).
2. **В View-логике (MainWindow.xaml.cs):**
   * Реализован обработчик события **OperationText\_TextChanged()**, который при каждом изменении текста в поле ввода вызывает соответствующий метод Check\*Valid() из Warehouse.
   * Результат этой проверки (true или false) немедленно преобразуется в цвет **индикатора Pre-условия** (элемент PreConditionIndicator) с использованием Brushes.Green или Brushes.Red.

**B. Проверка Постусловия (Post-condition)**

1. **Во время выполнения (MainWindow.xaml.cs):**
   * Метод **Button\_Click()** инициирует выполнение операции.
   * Если операция (Warehouse.AddNewGood()) была успешно завершена без выброса исключений, то есть считается, что **Post-условие выполнено**, индикатор **Post-условия** (PostConditionIndicator) окрашивается в **зеленый** цвет.
   * В случае, если Pre-условие было нарушено или возникло исключение во время выполнения, индикатор Post-условия остается **красным**, что сигнализирует о несоблюдении контракта или ошибке.

**3. Механизм визуализации контракта**

Для обеспечения прозрачности контракта, как того требует задание, реализовано модальное окно **ContractWindow**.

* Класс **Warehouse** содержит статические методы (GetPreConditionText(int index) и GetPostConditionText(int index)), которые возвращают **полный текстовый вид** контракта для выбранной операции.
* При нажатии на кнопку **«Показать контракт»** (ShowContractButton\_Click), MainWindow.xaml.cs открывает ContractWindow, передавая ей индекс выбранной операции.
* **ContractWindow.xaml.cs** использует этот индекс для получения и отображения полного текста Pre- и Post-условий, а также граничных примеров.

# Вывод

В ходе выполнения лабораторной работы была успешно **реализована концепция «Контракты в интерфейсе»** применительно к системе складского учёта. Были разработаны и документированы ключевые операции, включая **«Добавить новый товар»** и **«Переместить товар между складами»**, с чёткой формализацией их спецификаций.

**Основные результаты работы:**

1. **Формализация контрактов:** Для каждой операции были сформулированы исчерпывающие **Предусловия (Pre-conditions)**, описывающие требования к входным данным и состоянию системы, а также **Постусловия (Post-conditions)**, гарантирующие правильность конечного состояния.
2. **Визуализация Pre-условий:** Был реализован **динамический индикатор Pre-условия**, который в реальном времени меняет цвет с красного на зеленый при вводе валидных данных. Этот механизм позволяет пользователю **мгновенно предотвращать ошибки**, обеспечивая соответствие входных данных требованиям контракта ещё до запуска операции.
3. **Подтверждение Post-условий:** Индикатор **Post-условия** (зеленый цвет после выполнения) подтверждает, что операция успешно завершена и **система перешла в гарантированное контрактом состояние**.
4. **Прозрачность документации:** Создано модальное окно **«Показать контракт»**, обеспечивающее полную прозрачность логики работы системы для пользователя, включая детальное описание Pre/Post-условий, эффектов и граничных примеров.