

Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина: Объектно-ориентированное программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
к курсовому проекту  
на тему

**ПРОГРАММА ВЕДЕНИЯ СКЛАДСКОГО УЧЁБА И РЕГИСТРАЦИИ  
ЗАКАЗОВ**

БГУИР КП 1-40 05 01-12 053 ПЗ

Студент гр. 024402

Котлярова Я.И.

Руководитель

Бич Н.А.

Минск 2022

## СОДЕРЖАНИЕ

Введение.....	5
1 Обзор ведения складского учёта и регистрации заказов, программных аналогов, методов и алгоритмов решения поставленной задачи.....	6
1.1 Обзор ведения складского учёта и регистрации заказов .....	6
1.2 Обзор программных аналогов.....	7
1.3 Обзор методов и алгоритмов решения поставленной задачи .....	9
2 Функциональное моделирование на основе стандарта IDEF0.....	12
3 Структура используемых данных.....	17
4 Описание созданных программных конструкций .....	20
5 Разработка и описание диаграммы классов приложения .....	21
5.1 Класс Authorization.....	21
5.2 Класс Admin.....	23
5.3 Класс Client .....	24
5.4 Класс Supplier .....	25
5.5 Класс Good .....	26
5.6 Класс Prod .....	27
5.7 Класс NonProd .....	28
5.8 Класс SmartPointer.....	29
6 Разработка и описание диаграммы вариантов использования приложения	31
7 Блок-схема алгоритма работы всей программы и двух и более основных методов .....	32
7.1 Назначение блок-схем .....	32
7.2 Схема алгоритма работы всей программы и двух основных методов ..	32
8 Описание алгоритма запуска приложения, его использования, результаты работы программы, тестирования обработки ошибок .....	35
Заключение .....	42
Список использованных источников .....	43
Приложение А Диаграмма классов .....	44
Приложение Б Диаграмма вариантов использования .....	45
Приложение В Схема алгоритма .....	46
Приложение Г Листинг кода.....	52

## ВВЕДЕНИЕ

Автоматизация является логичным направлением научно-технического процесса, подразумевающим использование автоматических и автоматизированных устройств, работающих частично или полностью без участия человека.

Человечество давно решило перейти от работы, выполняемой вручную, на использование машинной техники, так как автоматизация позволяет повысить производительность труда, улучшить качество продукции, оптимизировать процессы управления, отстранить человека от производств, опасных для здоровья. Сейчас создаётся всё больше программ, сервисов, интернет-магазинов, различных систем и подобного программного обеспечения для упрощения жизни людей и их удобства.

Причиной этого становится превосходство машинных программ над человеком в объёмах обрабатываемой информации за одну и ту же единицу времени, точности расчётов, скорости обработки информации.

Поэтому целью курсового проекта является улучшение качества работы и удобства использования программы ведения складского учёта и регистрации заказов путём разработки объектной модели и её программной реализации на языке C++.

Для реализации цели курсового проекта были сформулированы следующие задачи:

- написать программу для ведения складского учёта и регистрации заказов на языке программирования C++;
- спроектировать функциональную модель обозреваемой области;
- структурировать используемые данные;
- разработать и подробно описать программные конструкции;
- описать варианты использования приложения, используя диаграммы;
- описать схемы работы всей программы и разбор нескольких её методов используя блок-схемы;
- описать алгоритм запуска программы, алгоритм использования приложения, тестирование обработки ошибок и вывода результатов программы.

Конечная программа позволит автоматизировать и оптимизировать работу с системой ведения складского учёта и регистрацией заказов, позволяя работникам получать информацию о поступающих и отправляемых со склада товарах. Клиенты смогут делать заказы, а поставщики добавлять товары.

У разрабатываемой программы имеется довольно большое количество аналогов. Актуальность проекта подтверждается высоким спросом на программное обеспечение, подобное разрабатываемому.

# **1 ОБЗОР ВЕДЕНИЯ СКЛАДСКОГО УЧЁТА И РЕГИСТРАЦИИ ЗАКАЗОВ, ПРОГРАММНЫХ АНАЛОГОВ, МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ**

## **1.1 Обзор ведения складского учёта и регистрации заказов**

Предметной областью курсового проекта является «автоматизация ведения складского учёта и регистрации заказов».

Цель данной предметной области – это улучшение и упрощение учёта товаров и оформления и принятия заказов для работников складского помещения. Пользователи смогут заказывать товары со склада. Поставщики смогут подавать данные о товарах, поставляемых на склад.

Для подачи данных о товарах на склад поставщику надо будет зарегистрироваться в системе, указав следующие пункты:

- логин;
- пароль;
- наименование организации;
- ФИО;
- номер телефона;
- адрес электронной почты.

Для составления заказа пользователю надо будет зарегистрироваться в системе, указав следующие пункты:

- логин;
- пароль;
- наименование организации;
- ФИО;
- номер телефона;
- адрес электронной почты.

Для регистрации администратора нужно ввести правильно информацию по следующим пунктам:

- логин;
- пароль;
- ФИО;
- должность.

Если поставщик, пользователь или администратор уже регистрировался в системе, то он сможет зайти в систему снова, указав логин и пароль при входе в систему.

Администратор в автоматизируемой системной области будет обладать следующими основными возможностями:

1. Просмотр личной информации.
2. Изменение личной информации.
3. Просмотр информации об администраторах.
4. Просмотр информации о поставщиках.

5. Просмотр информации о клиентах.
6. Удаление администраторов из базы.
7. Удаление поставщиков из базы.
8. Удаление клиентов из базы.
9. Просмотр товаров на складе.
10. Сортировка товаров.

Учет товаров в системе будет определяться правильностью введенных данных по всем пунктам, описанным ниже:

- наименование товара;
- необходимые характеристики;
- фирма-изготовитель;
- поставляемое количество;
- цена за единицу товара.

Упрощенная схема работы складского помещения представлена на рисунке 1.1.



Рисунок 1.1 – Упрощенная схема работы складского помещения

## 1.2 Обзор программных аналогов

Для обзора программных аналогов был произведен поиск и рассмотрение некоторых похожих по сути программ в сети Интернет. Было найдено достаточно много программ, выполняющих подобные функции: «1С:Торговля и Склад», «МойСклад», «Антисклад» и другие.

«1С:Торговля и Склад» – известная программа для товарного учета, рассчитанная на крупный бизнес: промышленные предприятия, сетевые магазины и заведения общепита.

«1С:Торговля и Склад» предназначена для учета любых видов торговых операций. Благодаря гибкости и настраиваемости, система

способна выполнять все функции учета – от ведения справочников и ввода первичных документов до получения различных ведомостей и аналитических отчетов.

В программе «1С:Торговля и Склад» предоставлены следующие функции:

- вести отдельный управленческий и финансовый учет;
- вести учет от имени нескольких юридических лиц;
- вести партионный учет товарного запаса с возможностью выбора метода списания себестоимости (FIFO, LIFO, по средней);
- вести отдельный учет собственных товаров и товаров, взятых на реализацию;
- оформлять закупку и продажу товаров;
- производить автоматическое начальное заполнение документов на основе ранее введенных данных;
- вести учет взаиморасчетов с покупателями и поставщиками, детализировать взаиморасчеты по отдельным договорам;
- формировать необходимые первичные документы;
- оформлять счета-фактуры, автоматически строить книгу продаж и книгу покупок, вести количественный учет в разрезе номеров ГТД;
- выполнять резервирование товаров и контроль оплаты;
- вести учет денежных средств на расчетных счетах и в кассе;
- вести учет товарных кредитов и контроль их погашения;
- вести учет переданных на реализацию товаров, их возврат и оплату [1].

«МойСклад» – облачное ПО с широким набором функций. Войти в личный кабинет можно с любого устройства, в том числе с мобильного телефона.

Программа подходит для оптовых и розничных магазинов, для небольших предприятий. Малому бизнесу подходит бесплатный тариф, неограниченный по времени. Тариф рассчитан на торговые точки с одним сотрудником. Платные версии рассчитаны на любое количество точек и персонала.

В программе «МойСклад» предоставлены следующие функции:

- складской учёт;
- обработка заказов;
- розничные продажи;
- сборка и производство;
- работа с клиентами;
- управление финансами [2].

«Антисклад» – программное обеспечение для комплексного управления торговыми точками. В личном кабинете можно контролировать приемку и отгрузку, следить за оприходованием, перемещениями, списанием и инвентаризацией.

Базы данных синхронизируются в режиме реального времени. Сотрудники могут видеть реальное число остатков в магазине или на складе. Система отслеживает выручку, оборот, прибыль. Отчет по финансовым показателям доступен для руководителя.

«Антисклад» позволяет контролировать работу сотрудников и общаться с покупателями. В личном кабинете можно создавать и отправлять SMS-рассылки. Для отправки сообщений нужно собрать и загрузить базу телефонных номеров.

В программе «Антисклад» предоставлены следующие функции:

- синхронизация;
- инвентаризация;
- Интернет-продажи;
- настройка системы скидок;
- хранение данных;
- размерные сетки;
- мультивалютность;
- маркировка;
- печать этикеток и ценников [3].

### **1.3 Обзор методов и алгоритмов решения поставленной задачи**

Поставленная задача заключается в реализации автоматизированной системы ведения складского учёта и регистрации заказов. В зависимости от предоставляемого уровня доступа в программе предусмотрены ограничения на выполнение некоторых функций. То есть пользователь или поставщик не должны иметь прав, таких, какие имеет администратор, и наоборот.

В ходе разработки автоматизированной системы нужно реализовать следующие методы для администратора:

- авторизация администратора, которая включает в себя регистрацию администратора, вход в систему при наличии данных о регистрации;
- просмотр информации об администраторах;
- просмотр информации о поставщиках;
- просмотр информации о клиентах;
- удаление информации об администраторах;
- удаление информации о поставщиках;
- удаление информации о клиентах;
- просмотр товаров на складе;
- сортировка товаров по возрастанию количества;
- просмотр личной информации;
- изменение личной информации.

Метод просмотра информации о клиентах и поставщиках выводит на экран информацию обо всех имеющихся клиентах или поставщиках в базе.

Метод удаления информации о клиенте или поставщике выводит на экран информацию обо всех клиентах или поставщиках, после дается выбор, какого клиента или поставщика из списка удалить, после выводится сообщение об успешном удалении.

Метод просмотра товаров на складе выводит список товаров с их наименованиями, фирмой-производителем, датой изготовления, сроком годности, размерными характеристиками, количеством и ценой за единицу.

В ходе разработки автоматизированной системы нужно реализовать следующие методы для пользователя:

- авторизация пользователя, которая включает в себя регистрацию пользователя, вход в систему при наличии данных о регистрации;
- изменение личной информации;
- просмотр личной информации;
- просмотр товаров, хранящихся на складе;
- добавление товара в корзину;
- удаление товара из корзины;
- просмотр корзины;
- заказ товаров, находящихся в корзине.

Регистрация происходит путем ввода личных данных в программу, производится запись информации о клиенте и его логин и пароль заносятся в базу с клиентами.

Авторизация происходит путем ввода личного логина и пароля, при некорректной введенной информации программа выдаст ошибку и попросит ввести логин и пароль снова.

Метод изменения личной информации о клиенте предоставляет возможность изменить личную информацию. После изменения выводится сообщение об успешном изменении личной информации.

Метод просмотра личной информации выводит на экран всю личную информацию о клиенте.

Метод просмотра хранящихся на складе товаров выводит пользователю таблицу товаров со всеми их характеристиками.

Метод добавления товара в корзину позволяет пользователю первоначально просмотреть либо все хранящиеся на складе товары, либо отфильтрованные по максимальной цене, либо найти необходимый товар по названию. При этом перед характеристиками товара выводится его номер, введя который пользователь сможет добавить желаемый товар в корзину.

Метод удаления товара из корзины выводит пользователю все товары, находящиеся в его корзине, и запрашивает номер товара, который подлежит удалению.

Метод заказа товара подготавливает клиенту чек о его покупках и выводит его на экран, а также сохраняет данные в базе. При оформлении заказа также изменяется количество товаров в базе, которые были куплены.

В ходе разработки автоматизированной системы нужно реализовать следующие методы для поставщика:



- авторизация поставщика, которая включает в себя регистрацию поставщика, вход в систему при наличии данных о регистрации;
- изменение личной информации;
- просмотр личной информации;
- просмотр информации о поставленных товарах;
- добавление новых товаров;
- удаление товара со склада;
- изменения информации о товаре;
- сортировка товаров по возрастанию количества.

Метод изменения личной информации о поставщике предоставляет возможность изменить личную информацию. После изменения выводится сообщение об успешном изменении личной информации.

Метод просмотра личной информации выводит на экран всю личную информацию о поставщике.

Метод просмотра информации о поставленных товарах выводит список товаров с их наименованиями, заданными характеристиками, количеством и ценой за единицу. Поставщику выводятся только поставленные им товары.

Все методы используют для записи, удаления и хранения файлы, а для редактирования – контейнеры.

## 2 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0

Для функционального описания модели работы программы была выбрана нотация IDEF0. Данная функциональная модель представлена в виде набора блоков, каждый из которых в свою очередь представлен в виде чёрных прямоугольных областей с входами и выходами, управлением и механизмами, которые уточняются и раскрываются до необходимого уровня сложности и детальности. Данная модель даёт возможность описать все основные виды процессов, представленные в данной программе [4].

На рисунке 2.1 изображена контекстная диаграмма верхнего уровня модели «Учёт товаров на складе и регистрация заказов», а также определены входные и выходные потоки данных, механизмы ограничения и управления данными.

Входной поток включает в себя запрос от поставщика, накладную от поставщика на товар, сам товар, а также заказ клиента. Выходной поток включает в себя предоставление данных для клиента при покупке им товаров со склада. Механизмами управления являются поставщик, клиент и администратор. Механизмами ограничения являются нормативные и сопроводительные документы, а также учёт сроков годности товаров.

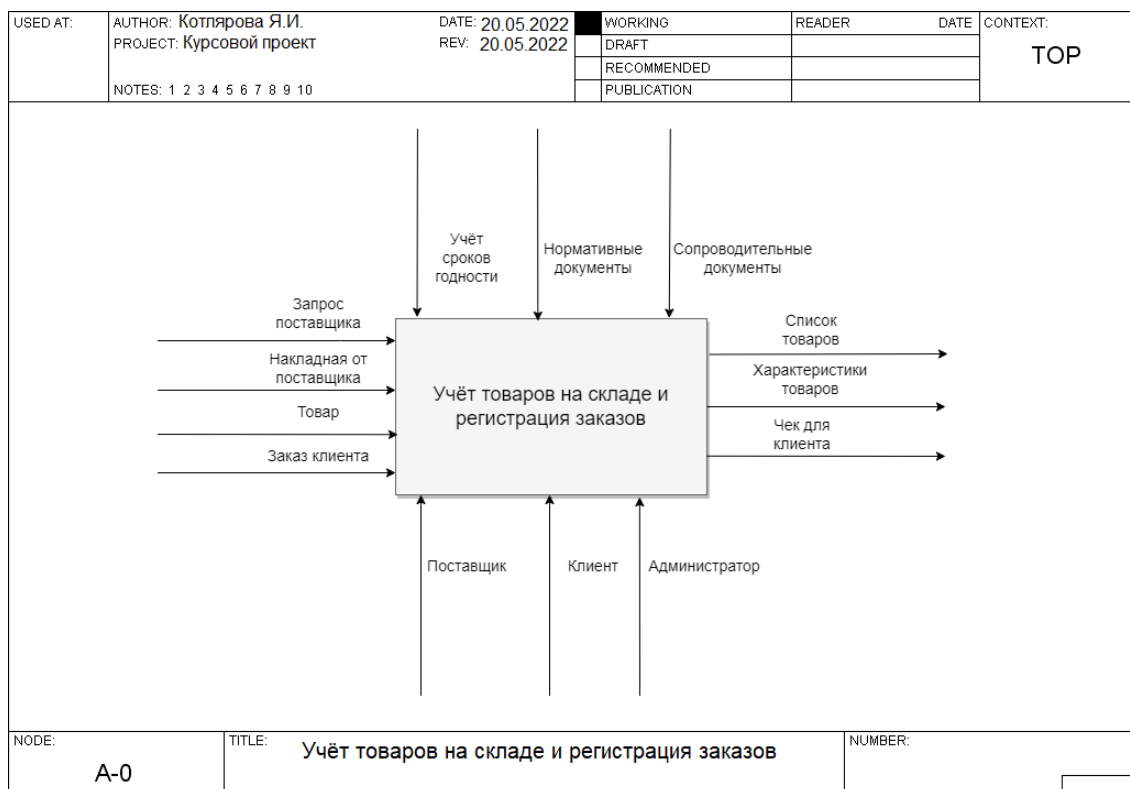


Рисунок 2.1 – Контекстная диаграмма верхнего уровня

На рисунке 2.2 изображена декомпозиция контекстной диаграммы верхнего уровня, состоящая из трёх блоков: «Поставка товара на склад», «Занесение данных в список товаров», «Реализация товара».

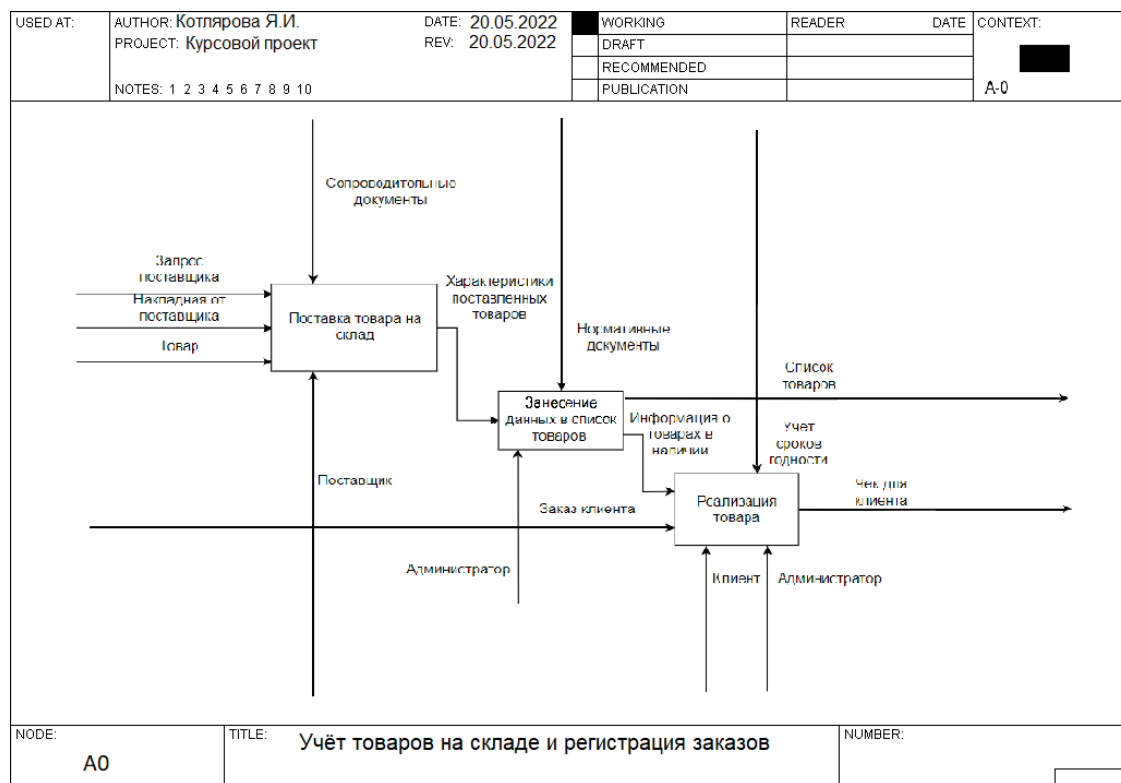


Рисунок 2.2 – Декомпозиция контекстной диаграммы

Первый компонент данной декомпозиции, «Поставка товара на склад», подразумевает получение товаров на склад от поставщиков. Данный процесс разделяется на создание поставщиков запроса на хранение товаров на складе, оформление необходимых документов и привоз товара на склад. Декомпозиция блока представлена на рисунке 2.3.

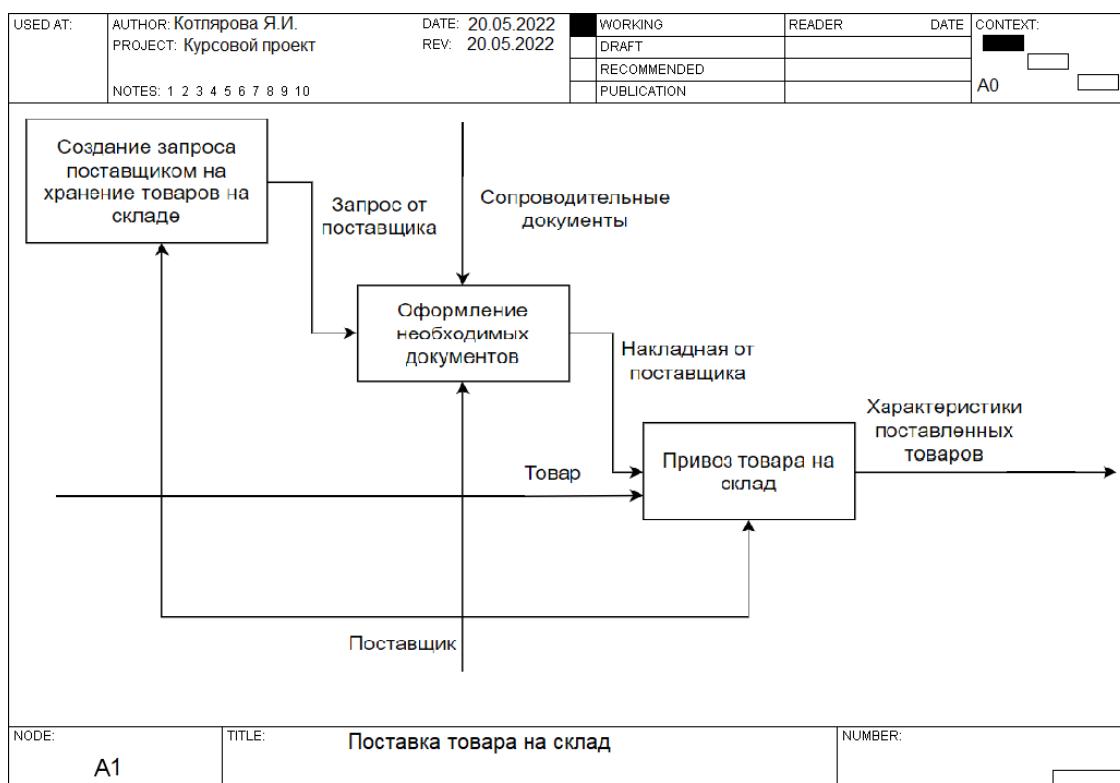


Рисунок 2.3 – Декомпозиция блока «Поставка товара на склад»

Второй компонент декомпозиции, «Занесение данных в список товаров», подразумевает собой процесс, в котором происходит сбор информации о товаре, затем проверка полученных данных на правильность и подлинность и занесение проверенных данных в базу. Декомпозиция блока представлена на рисунке 2.4.



Рисунок 2.4 – Декомпозиция блока «Занесение данных в список товаров»

Третий компонент представлен блоком «Реализация товара». В этом процессе происходит составление заказа клиентов с использованием информации о товарах в наличии, а также проверки сроков годности. Затем происходит оформления заказа клиента с использованием необходимых данных, полученных от клиента. Последним этапом является реализация заказа. Блок в качестве входных данных получает заказ клиента, а в качестве выходных выдает чек.

Декомпозиция блока «Реализация товара» представлена на рисунке 2.5.

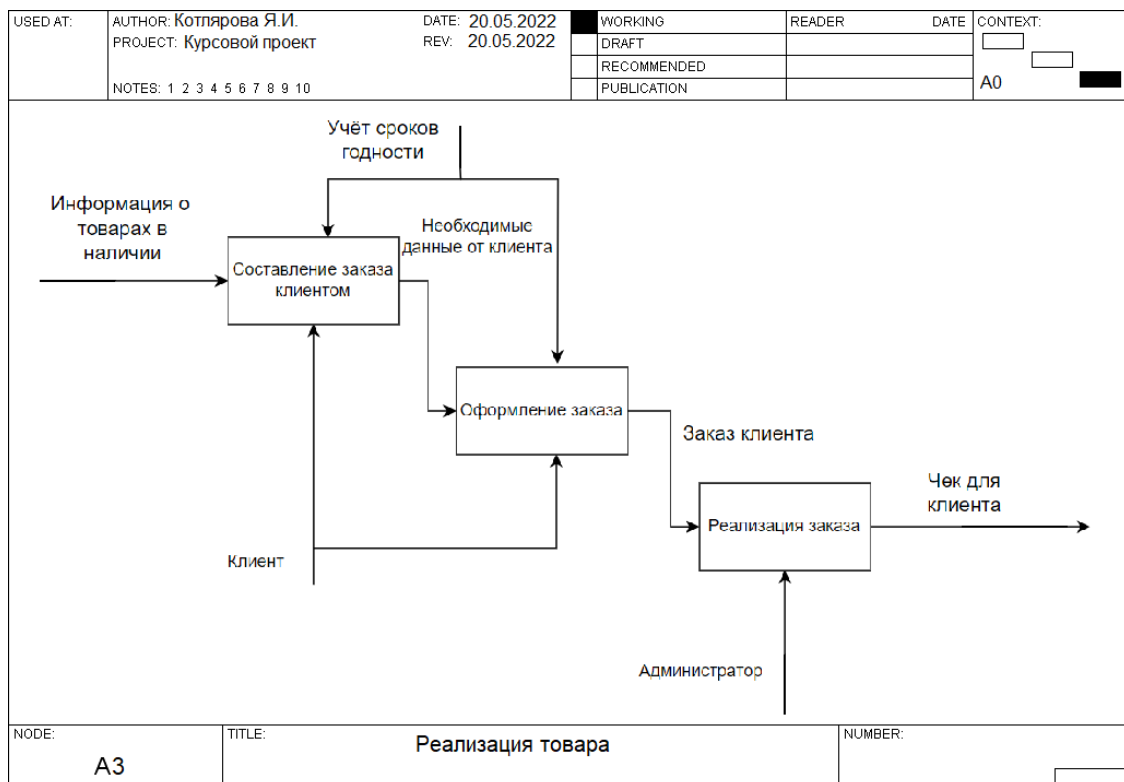


Рисунок 2.5 – Декомпозиция блока «Реализация товара»

Данный подход моделирования позволяет наглядно видеть, на каком этапе происходит приём товара на склад, сбор информации о товаре и запись информации о нём в базу данных. Также, наглядно видно, как происходит реализация товаров со склада.

### 3 СТРУКТУРА ИСПОЛЬЗУЕМЫХ ДАННЫХ

Вся информация, которая используется в проекте, хранится в текстовых файлах.

Информация об администраторах хранится в файлах «admin.txt» и «InfoAdmin.txt». В файле «admin.txt» хранятся логин и пароль каждого зарегистрированного администратора. Данный файл обновляется при регистрации новых администраторов. В файле «InfoAdmin.txt» хранятся следующие данные администраторов:

- логин;
- ФИО;
- должность.

Данный файл обновляется при правильной регистрации новых администраторов.

Клиентская информация хранится в двух файлах: «client.txt» и «InfoClient.txt». В «client.txt» хранятся логин и пароль каждого зарегистрированного клиента. Данный файл обновляется при регистрации новых клиентов и удалении существующих администратором. В файле «InfoClient.txt» хранятся следующие данные клиентов:

- логин;
- наименовании организации;
- ФИО;
- номер телефона;
- электронная почта.

Данный файл обновляется при правильной регистрации нового клиента и удалении зарегистрированного клиента администратором.

Информация о поставщиках хранится в двух файлах: «supplier.txt» и «InfoSupplier.txt». В «supplier.txt» хранятся логин и пароль каждого зарегистрированного поставщика. Данный файл обновляется при регистрации новых поставщиков и удалении существующих администратором. В файле «InfoSupplier.txt» хранятся следующие данные клиентов:

- логин;
- наименовании организации;
- ФИО;
- номер телефона;
- электронная почта.

Данный файл обновляется при правильной регистрации нового поставщика и удалении зарегистрированного администратором.

Пароли в файлах хранятся в зашифрованном виде. Шифрование проводится с помощью встроенной структуры библиотеки STL hash.

Информация о товарах, поставленных на склад, хранится в двух файлах: «ProdGoods.txt» и «NonProdGoods.txt», содержащие информацию о

производственных и непроеизводственных товарах соответственно. Данные файлы обновляются во время работы программы. В файле «ProdGoods.txt» хранятся следующие данные о товарах:

- наименование товара;
- фирма-производитель;
- дата изготовления;
- срок годности;
- поставленное количество;
- цена за единицу товара;
- логин поставщика, поставившего данный товар.

В файле «NonProdGoods.txt» хранятся следующие данные о товарах:

- наименование товара;
- фирма-производитель;
- дата изготовления;
- размерные характеристики;
- поставленное количество;
- цена за единицу товара;
- логин поставщика, поставившего данный товар.

Файлы «ClientBasket.txt» и «ClientBasket2.txt» хранят в себе информацию, представляющую собой корзину клиента. Данные файлы обновляются во время работы программы. В файле «ClientBasket.txt» хранятся данные, относящиеся к производственным товарам:

- наименование товара;
- фирма-производитель;
- дата изготовления;
- срок годности;
- покупаемое количество;
- цена за единицу товара;
- логин поставщика, поставившего данный товар;
- логин клиента, которому принадлежит заказ в корзине.

В файле «ClientBasket2.txt» хранятся данные, относящиеся к производственным товарам:

- наименование товара;
- фирма-производитель;
- дата изготовления;
- размерные характеристики;
- покупаемое количество;
- цена за единицу товара;
- логин поставщика, поставившего данный товар;
- логин клиента, которому принадлежит заказ в корзине.

Для работы с файлами необходимо подключить заголовочный файл <fstream>. В <fstream> определены несколько классов и подключены



заголовочные файлы `<ifstream>` – файловый ввод и `<ofstream>` – файловый вывод.

Файловый ввод-вывод аналогичен стандартному вводу-выводу, единственное отличие – это то, что ввод/вывод выполняется не на экран, а в файл. Если ввод/вывод на стандартные устройства выполняется с помощью объектов `cin` и `cout`, то для организации файлового ввода-вывода достаточно создать собственные объекты, которые можно использовать аналогично операторам `cin` и `cout`. В данном проекте они имеют название `fin` и `fout` соответственно.

Библиотека STL содержит много разных контейнерных классов, которые можно использовать в разных ситуациях. В проекте контейнеры используются для хранения данных в оперативной памяти для последующего взаимодействия с ними.

Контейнеры – это объекты, которые содержат в себе совокупность других объектов. Если говорить в общем, то контейнеры STL делятся на три основные категории: последовательные, ассоциативные, адаптеры.

Последовательные контейнеры – это контейнерные классы, элементы которых находятся в последовательности. Все есть шесть последовательных контейнеров: `std::vector`; `std::deque`; `std::array`; `std::list`; `std::forward_list`; `std::basic_string`.

Ассоциативные контейнеры – это контейнерные классы, которые автоматически сортируют все свои элементы. Всего четыре ассоциативных контейнера: `std::set`, `std::multiset`, `std::map`, `std::multimap`.

Адаптеры – это специальные предопределенные контейнерные классы, которые адаптированы для выполнения конкретных заданий. Всего есть три контейнера адаптера: `std::stack`, `std::queue`, `std::priority_queue`.

Итератор – это объект, который способен перебирать элементы контейнерного класса без необходимости пользователю знать реализацию определенного контейнерного класса. Все контейнеры предоставляют, как минимум, два типа итераторов: `container::iterator` – итератор для чтения/записи; `container::const_iterator` – итератор только для чтения.

Конкретно используемым в проекте контейнером является `<vector>`. Векторы представляют собой динамические массивы. Класс `<vector>` поддерживает динамический массив, который при необходимости может увеличивать свой размер.

В проекте контейнеры используются для редактирования данных, хранящиеся в файле. Данные считываются из файла и при соответствии поставленным условиям записываются в контейнер объектов класса с помощью функции `push_back`. Далее данные, находящиеся в контейнере, можно изменять, обращаясь к элементам контейнера подобно элементам массива. Также данные можно удалить как функцией `erase`, так и не записыванием в контейнер при считывании файла. После изменения данные помещаются обратно в файл с помощью перегруженного оператора записи в поток `<<`.

## 5 РАЗРАБОТКА И ОПИСАНИЕ ДИАГРАММЫ КЛАССОВ ПРИЛОЖЕНИЯ

Объектно-ориентированное программирование предполагает создание собственных типов данных, называемых классами. Такой подход использовался и при разработке данного проекта.

ООП строится на четырёх основных парадигмах:

1 Абстракция данных. Абстрагирование означает выделение значимой информации и исключение из рассмотрения незначимой. В ООП рассматривают лишь абстракцию данных, подразумевая набор наиболее значимых характеристик объекта, доступных остальной программе.

2 Инкапсуляция – свойство системы, позволяющее объединить данные и методы, работающие с ними, в оболочке – классе.

3 Наследование представляет собой механизм получения нового класса из существующего. Существующий класс должен быть дополнен для создания производного класса. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс – потомком, наследником, дочерним или производным классом.

4 Применительно к C++ полиморфизм представляет собой термин, используемый для описания процесса, в котором различные реализации функции могут быть доступны посредством одного и того же имени. По этой причине полиморфизм иногда характеризуется фразой "один интерфейс, много методов".

В данном проекте были разработаны следующие 8 классов:

- класс Authorization;
- класс Admin, который наследуется от Authorization;
- класс Client, который наследуется от Authorization;
- класс Supplier, который наследуется от Authorization;
- абстрактный класс Good;
- класс Prod, который наследуется от Good;
- класс NonProd, который наследуется от Good;
- шаблонный класс SmartPointer и структура Status.

В данном проекте были реализованы описанные выше принципы объектно-ориентированного программирования. Диаграмма классов проекта представлена в приложении А.

### 5.1 Класс Authorization

Класс Authorization является базовым классом для трёх классов: Admin, Client и Supplier. Он представляет собой функционал для регистрации новых пользователей и авторизации уже заданных в системе. В данном классе содержатся следующие поля:

- поля login и loginR, хранящие логин пользователя;

- поля password и passwordR, хранящие пароль пользователя;
- поле org, хранящее наименование организации пользователя;
- поле fio, хранящее ФИО пользователя;
- поле tel, хранящее телефон пользователя;
- поле email, хранящее электронную почту пользователя;
- поле complete, разрешающее или запрещающее вход в аккаунт;
- поле Check, являющееся проверкой правильности ввода логина и

пароля.

Для класса разработаны следующие методы:

- метод check;
- метод reg, позволяющий пользователю зарегистрироваться, путём добавление в подходящий файл логина и пароля;
- метод log, позволяющий выполнить вход в аккаунт в последующие разы использования приложения;
- метод set, позволяющий заполнить поля объектов класса;
- методы wtofa, wtofs, wtofc, позволяющие записать логины и пароли пользователей в файлы;
- метод RetFIO, возвращающий значение private поля fio.

Диаграмма класса Authorization представлена на рисунке 5.1.

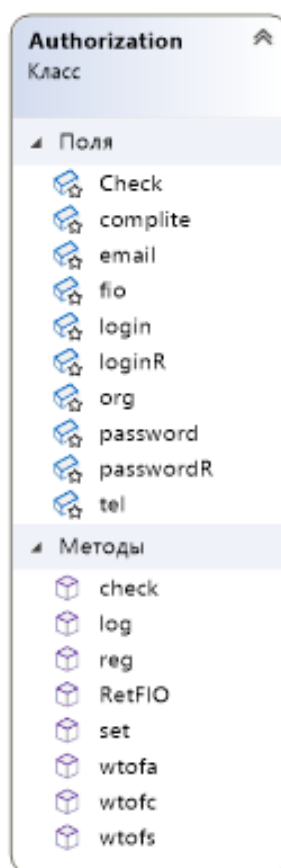


Рисунок 5.1 – Диаграмма класса Authorization

## 5.2 Класс Admin

Класс Admin является производным класса Authorization и наследует все его поля и методы. Добавленным полем в этом классе является поле post, хранящее информацию о должности администратора.

В классе были определены следующие методы:

- переопределённый метод set, позволяющий заполнить поля объектов класса;
- метод wtof, позволяющий записать информацию об администраторе в файл;
- метод rfromf, позволяющий считать информацию об администраторе из файла;
- метод viewInfo, позволяющий вывести информацию на консоль;
- методы viewAdmins, viewClients и viewSupliers, позволяющие просмотреть информацию обо всех администраторах, клиентах и поставщиках соответственно;
- методы deleteAdmin, deleteClient и deleteSupplier, позволяющий удалить одного из администраторов, клиентов или поставщиков из базы;
- шаблонный метод viewAll, позволяющий просмотреть все товары на складе;
- шаблонный метод sort, позволяющий отсортировать товары по возрастанию поставленного количества;
- метод change, позволяющий изменить информацию об администраторе.

Диаграмма класса Admin представлена на рисунке 5.2.

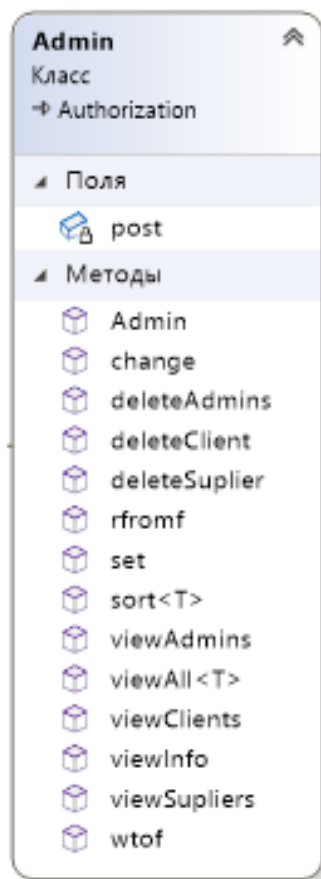


Рисунок 5.2 – Диаграмма класса Admin

### 5.3 Класс Client

Класс Client является производным класса Authorization и наследует все его поля и методы. Данный класс не имеет добавленных полей.

В классе были определены следующие методы:

- переопределённый метод **set**, позволяющий заполнить поля объектов класса;
- метод **wtof**, позволяющий записать информацию о клиенте в файл;
- метод **rfromf**, позволяющий считать информацию о клиенте из файла;
- метод **viewInfo**, позволяющий вывести информацию на консоль;
- метод **clearBasket**, позволяющие очистить корзину клиента;
- методы **deleteFromBasket** и **deleteFromBasket2**, позволяющие удалить товар из корзины производственных или непроизводственных товаров соответственно;
- шаблонный метод **viewAll**, позволяющий просмотреть все товары на складе;
- метод **change**, позволяющий изменить информацию о клиенте.

Диаграмма класса Client представлена на рисунке 5.3.

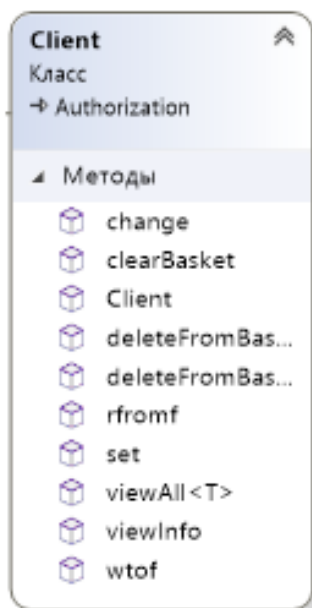


Рисунок 5.3 – Диаграмма класса Client

## 5.4 Класс Supplier

Класс Supplier является производным класса Authorization и наследует все его поля и методы. Данный класс не имеет добавленных полей.

В классе были определены следующие методы:

- переопределённый метод set, позволяющий заполнить поля объектов класса;
- метод wtof, позволяющий записать информацию о клиенте в файл;
- метод rfromf, позволяющий считать информацию о клиенте из файла;
- метод viewInfo, позволяющий вывести информацию на консоль;
- метод change, позволяющий изменить информацию о клиенте;
- шаблонный метод viewForSup, позволяющий пользователю просмотреть товары, добавленные им.

Диаграмма класса Supplier представлена на рисунке 5.4.

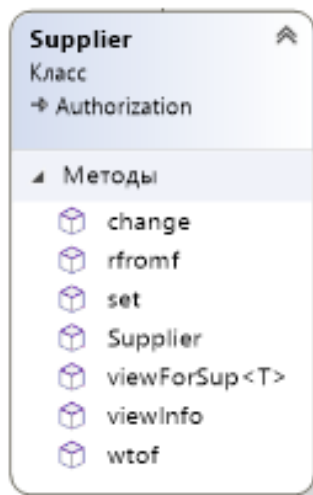


Рисунок 5.4 – Диаграмма класса Supplier

## 5.5 Класс Good

Класс Good является абстрактным классом, который служит основой для всех товаров, поставляемых на склад. В данном классе содержатся следующие поля:

- поле name, хранящее наименование товара;
- поле firm, хранящее фирму-производителя товара;
- поле supplierlog, хранящее логин поставщика;
- поле amount, хранящее количество поставленных товаров;
- поле cost, хранящее цену за единицу товара.

В классе были заданы следующие чисто виртуальные методы:

- метод add, позволяющий добавить товар в базу;
- метод set, позволяющий заполнить поля объектов класса;
- метод writeInBas, позволяющий записать выбранный пользователем товар в корзину;
- метод view, позволяющий вывести информацию о товаре на консоль в табличном виде;
- метод viewBusket, позволяющий клиенту просмотреть корзину;
- метод viewForSup, позволяющий поставщику просмотреть поставленные им товары;
- метод del, позволяющий удалить товар из базы;
- метод change, позволяющий изменить данные о товаре;
- метод sort, позволяющий отсортировать товары в базе по возрастанию поставленного количества;
- метод filter, позволяющий вывести товары отфильтрованные по максимальной цене;
- метод addToBusket, позволяющий клиенту добавить товар в корзину;
- метод search, позволяющий найти товар в базе по наименованию;



- метод `order`, позволяющий пользователю заказать товары, находящиеся в его корзине;
- метод `getOrder`, позволяющий получить копию чека заказа.

Наличие в классе чисто виртуальных функций делает его абстрактным, что запрещает создавать объекты этого типа и вынуждает переопределить эти функции в дочерних классах, реализуя одну из парадигм объектно-ориентированного программирования – полиморфизма.

Диаграмма класса `Good` представлена на рисунке 5.5.

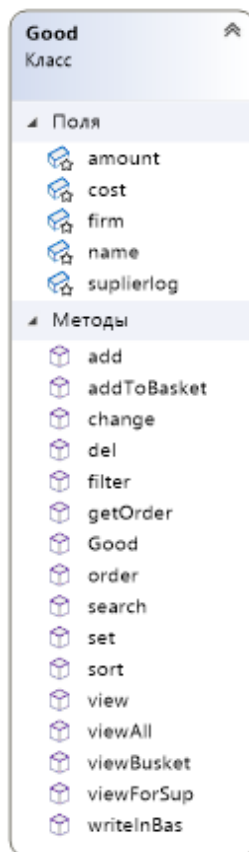


Рисунок 5.5 – Диаграмма класса `Good`

## 5.6 Класс `Prod`

Класс `Prod` является производным класса `Good` и наследует все его поля, а также переопределяет методы, так как они являются чисто виртуальными. Класс имеет добавленные поля:

- поле `date1`, хранящее дату изготовления производственного товара;
- поле `date2`, хранящее срок годности производственного товара.

Класс переопределяет все методы абстрактного базового класса, которые выполняют те же функции.

Диаграмма класса `Prod` представлена на рисунке 5.6.

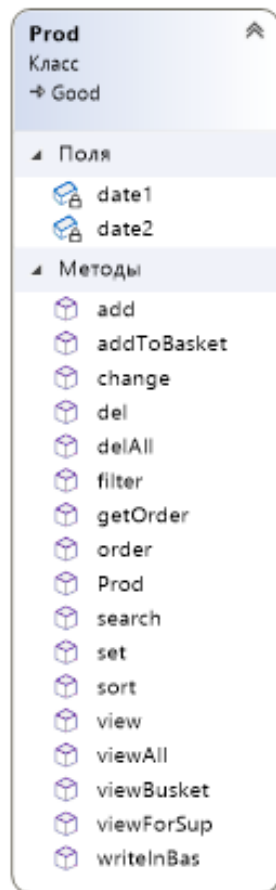


Рисунок 5.6 – Диаграмма класса Prod

## 5.7 Класс NonProd

Класс NonProd является производным класса Good и наследует все его поля, а также переопределяет методы, так как они являются чисто виртуальными. Класс имеет добавленные поля:

- поле date, хранящее дату изготовления производственного товара;
- поля l, w и h, используемые для задания длины, ширины и высоты;
- поле size, хранящее размерные характеристики товара.

Класс переопределяет все методы абстрактного базового класса, которые выполняют те же функции.

Диаграмма класса NonProd представлена на рисунке 5.7.

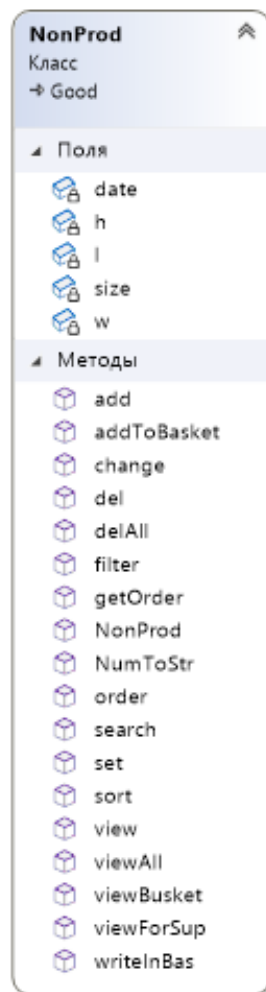


Рисунок 5.7 – Диаграмма класса NonProd

## 5.8 Класс SmartPointer

Класс SmartPointer предоставляет функционал для создания умных указателей на объекты, которые будут следить за объектами и когда объекты перестанут использоваться указатели удалят их. Вместе с классом SmartPointer используется структура Status, которая хранит в себе счетчик ссылок на объект (count) и указатель на объект (\*ptr).

Для данного класса определено поле \*smartPtr – указатель типа Status. В данном классе содержатся:

- конструктор с одним параметром;
- конструктор копирования;
- перегрузка оператора operator->;
- перегрузка оператора operator=;
- деструктор;
- метод просмотра числа ссылок на объект showCounter.

Диаграмма класса SmartPointer и структуры Status представлена на рисунке 5.8.

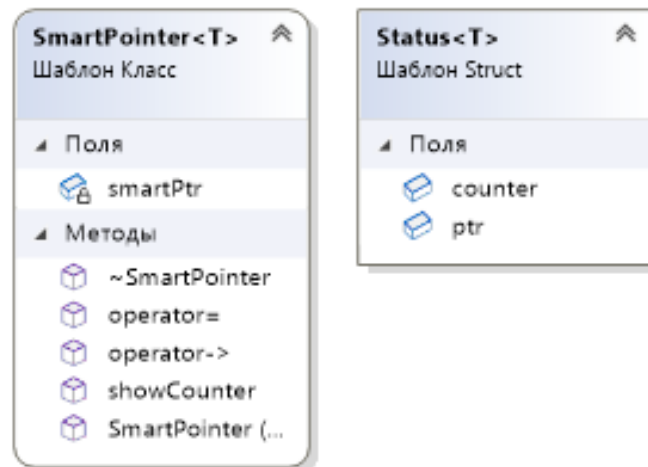


Рисунок 5.8 – Диаграммы класса `SmartPointer` и структуры `Status`

Класс `SmartPointer` нужен для того, чтобы не было следующих ошибок: утечки памяти, разыменовывание нулевого указателя, либо обращение к неинициализированной области памяти, удаление уже удаленного объекта.

## **6 РАЗРАБОТКА И ОПИСАНИЕ ДИАГРАММЫ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ ПРИЛОЖЕНИЯ**

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними. При построении диаграммы могут использоваться также общие элементы нотации: примечания и механизмы расширения.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером (действующим лицом, актантом, актёром) называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация сервисов (функций), которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий [5].

Для данного компьютерного приложения была разработана собственная диаграмма вариантов использования приложения. Данную диаграмму можно увидеть на рисунке А.1 приложения А. Действующими лицами в ней являются администратор, клиент и поставщик. Рассмотрим процессы, доступные каждому из них.

Администратор должен авторизоваться через свой собственный логин и пароль или зарегистрироваться. Далее он может товары, хранящиеся на складе, сортировать их для более удобного просмотра и учета. Администратор может просматривать информацию обо всех поставщиках, клиента и администраторах, а также удалять администраторов, клиентов и поставщиков из базы. Также он может просмотреть свою личную информацию и изменить её.

Клиент может войти в систему под своим логином и паролем, если уже регистрировался. Иначе может зарегистрироваться и потом авторизоваться. Далее клиент может просмотреть товары, находящиеся на складе, добавить товар в корзину, удалить товар из корзины и просмотреть корзину. Также клиент может сделать заказ и получить копию чека. Как и администратор, клиент может просмотреть свою личную информацию и изменить её.

Поставщик может войти в систему под своим логином и паролем или зарегистрироваться, введя личные данные. Далее поставщик может добавить товар на склад, просмотреть поставленные им товары, изменить данные о поставленном товаре, удалить товар со склада и отсортировать товары в базе по возрастанию поставленного количества. Также поставщик может просмотреть свою личную информацию и изменить её.

## **7 БЛОК-СХЕМА АЛГОРИТМА РАБОТЫ ВСЕЙ ПРОГРАММЫ И ДВУХ И БОЛЕЕ ОСНОВНЫХ МЕТОДОВ**

### **7.1 Назначение блок-схем**

Блок-схема представляет собой графическое представление алгоритма с помощью различных символов, форм и стрелок для демонстрации процесса или программы. С помощью алгоритмов мы можем легко понять программу. Основная цель блок-схемы – анализировать различные процессы.

Блок-схемы широко используются при написании программ, так как они:

- гораздо проще для понимания, чем запись в виде команд;
- упрощают процесс отладки;
- позволяют составить эффективную программную документацию;
- облегчают процесс демонстрации и обсуждения программы.

### **7.2 Схема алгоритма работы всей программы и двух основных методов**

Рассмотрим рисунок В.1 приложения В, на котором представлена схемы работы всей программы. При запуске программы пользователю предлагается выбрать, что он хочет сделать: зарегистрироваться, войти в аккаунт или выйти из программы. Как при выборе регистрации, так и при выборе входа в аккаунт у пользователя уточняется его роль, по которой определяются необходимые для работы файлы. После успешной регистрации или входа в аккаунт перед пользователем появляется меню для роли, которую он выбрал. Подробно различные меню пользователя можно рассмотреть на рисунках В.2-В.4. Пользователю предоставляется три роли: администратор, поставщик и клиент.

Администратор может выбрать из одиннадцати пунктов меню, которые позволят ему работать как с данными о товарах на складе, так и с данными других пользователей и личными данными. Администратор может просмотреть товары, хранящиеся на складе, а также отсортировать их по возрастанию количества. Он может просмотреть всех администраторов, поставщиков и клиентов, хранящихся в базе, а также удалить данные о них, если это необходимо. Администратор может просмотреть свою личную информацию, вводимую при регистрации, и, при необходимости, изменить её. Одиннадцатым пунктом меню администратора является пункт «Выход», возвращающий пользователя в меню авторизации.

Поставщик может выбирать из восьми предоставляемых ему функций. Среди них есть функции, позволяющий работать с данными товаров и с личными данными. Первые четыре пункта меню, кроме второго, содержат в себе ещё подменю, выбирая которые, поставщик совершает те или иные

действия. Функции, позволяющие работать с данными о товарах, включают возможность добавления новых товаров на склад, при этом выбирая тип добавляемых товаров: производственные или непроизводственные, информация о которых хранится в разных файлах. Возможность просмотра поставленных им товаров, изменение информации о них или удаление товаров из базы. Также пользователь может отсортировать товары по возрастанию количества. Следующими двумя функциями является просмотр личной информации, вводимой при регистрации и её изменение при необходимости. Последним доступным вариантом является опция «Выход», которая возвращает пользователя в меню авторизации.

Клиенту предоставляется возможность выбирать из девяти предоставляемых ему функций. Среди них есть функции, позволяющий работать с данными товаров, заказами и с личными данными. Первые три пункта меню содержат в себе ещё подменю, выбирая которые, клиент совершает те или иные действия. Функции, позволяющие работать с данными о товарах, включают возможность просмотра товаров, причём клиент может выбрать, желает он просмотреть все товары, либо только производственные, либо только непроизводственные. Пользователь может добавлять товары в корзину, при этом просматривая как полностью весь список товаров, так и отфильтрованный по цене, так и ища товар по наименованию. Пользователь может просмотреть свою корзину товаров, а также удалить товары из неё. Двумя функциями, связанными с заказом, являются оформление заказа и получения копии чека, который выводится пользователю при оформлении заказа, то есть покупки товаров, находящихся в корзине. Следующими двумя функциями является просмотр личной информации, вводимой при регистрации и её изменение при необходимости. Последним доступным вариантом является опция «Выход», которая возвращает пользователя в меню авторизации.

Одними из наиболее важных и часто используемых функций в данном проекте являются функция входа пользователя в аккаунт и функция вывода списка товаров, хранящихся на складе, которые можно увидеть на рисунках В.5 и В.6 приложения В. Функция авторизации используется при каждом запуске приложения и подразумевает собой ввод логина и пароля, для входа в аккаунт. Функция просмотра товаров в том или ином виде также доступна всем пользователям. При её вызове пользователю выводится список товаров, хранящихся на складе, причём функция имеет две перегрузки: для продовольственных и непродовольственных товаров.

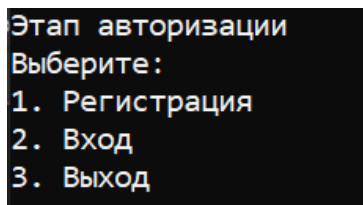
Функция авторизации пользователя имеет достаточно простую реализацию: пользователь вводит логин и пароль, пароль шифруется, и эти данные передаются в функцию, которая сравнивает считанные из файла данные с введёнными, и при совпадении логина и пароля возвращает значение true или false. При возвращении true функция возвращает в меню такое же значение, а в меню переменной присваивается значение, открывающее пользователю меню его роли.

Функция просмотра товаров на складе, открывает файл, хранящий необходимую информацию, при этом проверяя корректность открытия, после чего в цикле считываются данные, пока это возможно и выводятся на экран в табличном виде, после чего файл закрывается. Также при этом ведётся подсчёт товаров. Значения счётчика используются как при выводе, так и для проверки наличия товаров на складе.



## **8 ОПИСАНИЕ АЛГОРИТМА ЗАПУСКА ПРИЛОЖЕНИЯ, ЕГО ИСПОЛЬЗОВАНИЯ, РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ, ТЕСТИРОВАНИЯ ОБРАБОТКИ ОШИБОК**

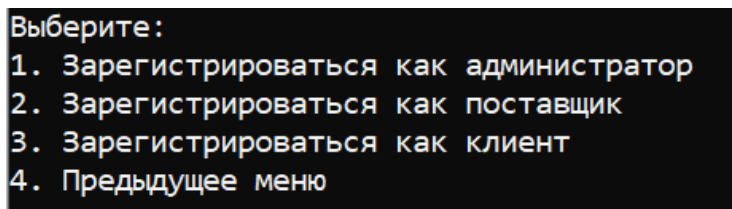
При запуске программы перед пользователем появляется начальное меню, которое можно увидеть из рисунка 8.1.



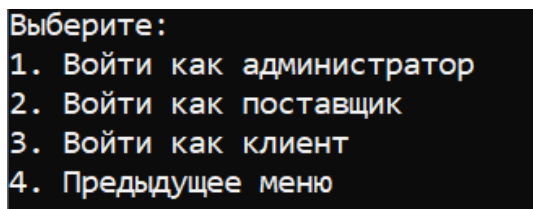
Этап авторизации  
Выберите:  
1. Регистрация  
2. Вход  
3. Выход

Рисунок 8.1 – Начальное меню

При выборе первого или второго пункта меню пользователю будет предложено выбрать роль, с которой он хочет зарегистрироваться в системе или войти в неё, что видно из рисунка 8.2.



Выберите:  
1. Зарегистрироваться как администратор  
2. Зарегистрироваться как поставщик  
3. Зарегистрироваться как клиент  
4. Предыдущее меню



Выберите:  
1. Войти как администратор  
2. Войти как поставщик  
3. Войти как клиент  
4. Предыдущее меню

Рисунок 8.2 – Выбор роли

При авторизации пользователя попросят ввести логин и пароль и при корректном вводе логина и пароля пользователь перейдёт в пользовательское меню, как показано на рисунке 8.3.

```

Введите логин: login1
Введите пароль: pass1

Добро пожаловать, Котко Дмитрий Анатольевич!
Выбирайте:
1. Просмотр товаров на складе
2. Сортировка товаров по количеству
3. Просмотр списка администраторов
4. Просмотр списка клиентов
5. Просмотр списка поставщиков
6. Удаление информации об администраторе из базы
7. Удаление информации о клиенте из базы
8. Удаление информации о поставщике из базы
9. Просмотреть личную информацию
10. Изменить личную информацию
11. Выход

```

Рисунок 8.3 – Авторизация пользователя

Первый пункт меню позволяет администратору просмотреть все товары, хранящиеся на складе. Причём данные выводятся в табличном виде, как показано на рисунке 8.4.

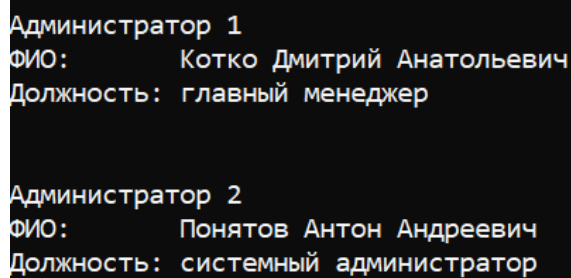
Производственные товары:						
Наименование товара	Фирма	Дата изготовления	Годен до	Количество	Цена	
1. Молоко 2.5%	Савушкин продукт	12.05.2022	22.05.2022	200	1.97	BYN
2. Сметана 20%	Минская марка	10.05.2022	30.05.2022	49	2.05	BYN
3. Трешок	Белакт	20.05.2022	20.06.2022	55	1.65	BYN
4. Сок берёзовый	Клёнки	13.04.2022	30.07.2022	40	1.99	BYN
5. Сок мультифруктовый	Добрый	10.05.2022	16.06.2022	50	2.5	BYN

Непроизводственные товары:						
Наименование товара	Фирма	Дата изготовления	Размер	Количество	Цена	
1. Платье летнее	8 марта	10.10.2021	1.2x0.4x0.7	23	35.99	BYN
2. Юбка-карандаш	Коминтерн	26.01.2021	0.6x0.4x0.6	35	23.55	BYN
3. Обои в цветочек	ГомельОбои	23.09.2021	10x0.5x1	200	58	BYN
4. Обои в линию	МинскОбои	01.09.2021	10x1x1	150	79	BYN

Рисунок 8.4 – Просмотр товаров, хранящихся на складе

Второй пункт позволяет отсортировать все товары по возрастанию поставленного количества. При выборе третьего, четвёртого или пятого пунктов на экран выводится список администраторов, поставщиков или клиентов, хранящихся в базе. Вывод администраторов представлен на рисунке 8.5.

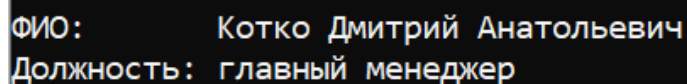


```
Администратор 1
ФИО:      Котко Дмитрий Анатольевич
Должность: главный менеджер

Администратор 2
ФИО:      Понятов Антон Андреевич
Должность: системный администратор
```

Рисунок 8.5 – Просмотр администраторов

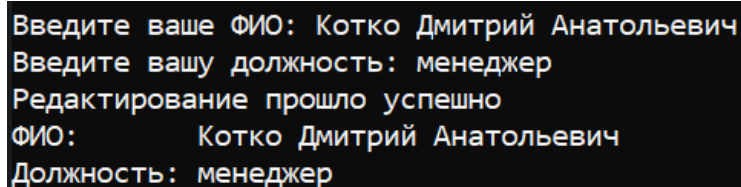
При выборе одного из пунктов шесть, семь или восемь выводится список администраторов, поставщиков или клиентов, подобный представленному на рисунке 8.5 и запрашивается номер для удаления. При выборе девятого пункта администратору выводится личная информация, как это показано на рисунке 8.6.



```
ФИО:      Котко Дмитрий Анатольевич
Должность: главный менеджер
```

Рисунок 8.6 – Вывод личной информации администратора

Выбор десятого пункта позволяет перезаписать эту информацию путём запроса ввода новых данных, как это показано на рисунке 8.7, а одиннадцатый пункт позволяет вернуться в меню авторизации.



```
Введите ваше ФИО: Котко Дмитрий Анатольевич
Введите вашу должность: менеджер
Редактирование прошло успешно
ФИО:      Котко Дмитрий Анатольевич
Должность: менеджер
```

Рисунок 8.7 – Изменение личной информации

Для регистрации необходимо ввести новые логин и пароль, а также необходимую личную информацию. Пример регистрации нового клиента показан на рисунке 8.8.

```
Введите логин: login33
Введите пароль: pass33
Введите наименование вашей организации: ОАО Задор
Введите ваше ФИО: Найда Алёна Игоревна
Введите ваш номер телефона: 375256308220
Введите вашу почту: naidahelena@mail.com

Добро пожаловать, Найда Алёна Игоревна!
Выбирайте:
1. Просмотр товаров на складе
2. Добавить товар в корзину
3. Удалить товар из корзины
4. Просмотреть корзину
5. Оформить заказ
6. Получить копию чека
7. Просмотреть личную информацию
8. Изменить личную информацию
9. Выход
```

Рисунок 8.8 – Регистрация нового клиента

Первый пункт меню «Просмотр товаров на складе» аналогичен просмотру товаров у администратора, за исключением того, что можно просмотреть как все товары, так и только производственные или только непроизводственные. Подменю, реализующее этот выбор показано на рисунке 8.9.

```
Выберите:
1. Просмотреть все товары
2. Просмотреть только производственные товары
3. Просмотреть только непроизводственные товары
4. Предыдущее меню
```

Рисунок 8.9 – Подменю для выбора вида просмотра товаров

Выбор второго меню позволяет пользователю просмотреть товары и добавить один в корзину. При этом просмотреть он может либо производственные, либо непроизводственные товары, а после выбрать между просмотром всех товаров выбранного типа, либо отфильтрованных по

максимальной цене, либо при поиске по наименованию. Пример добавления производственного товара с применение фильтрации и с применением поиска показан на рисунке 8.10.

Выберите вид товара для добавления в корзину:

1. Производственный
2. Непроизводственный
3. Предыдущее меню

1

Выберите:

1. Поиск товара по наименованию
2. Предварительно отфильтровать с указанием максимальной цены
3. Добавить без фильтрования

2

Введите максимальную цену для фильтрации товаров: 2

Фильтрация прошла успешно.

Ваш отфильтрованный список товаров:

Наименование товара	Фирма	Дата изготовления	Годен до	Количество	Цена
1. Сок берёзовый	Клёнки	13.04.2022	30.07.2022	37	1.99 BYN
4. Трвожок	Белакт	20.05.2022	20.06.2022	52	1.65 BYN
5. Молоко 2.5%	Савушкин продукт	12.05.2022	22.05.2022	199	1.97 BYN

Введите номер товара для добавления его в корзину

4

В каком количестве: 2

Добавление прошло успешно!

---

Введите наименование искомого товара: Сметана 20%

Поиск прошёл успешно.

Наименование товара	Фирма	Дата изготовления	Годен до	Количество	Цена
2. Сметана 20%	Минская марка	10.05.2022	30.05.2022	49	2.05 BYN

Введите номер товара для добавления его в корзину

2

В каком количестве: 1

Добавление прошло успешно!

Рисунок 8.10 – Пример добавления товара в корзину

Четвёртый пункт меню позволяет просмотреть корзину, а при выборе третьего пункта к этому выводу добавляется запрос номера товара, который следует удалить из корзины. Вывод содержимого корзины показан на рисунке 8.11.

Производственные товары:					
Наименование товара	Фирма	Дата изготовления	Годен до	Количество	Цена
1. Трвожок	Белакт	20.05.2022	20.06.2022	2	1.65 BYN
Непроизводственные товары:					
Наименование товара	Фирма	Дата изготовления	Размер	Количество	Цена
В корзине нет товаров.					

Рисунок 8.11 – Вывод содержимого корзины

Пятый пункт даёт возможность оформить заказ. При этом будет выведен чек, содержащий перечень товаров, находящихся в корзине клиента и их общую стоимость, как показано на рисунке 8.12. При оформлении заказана, количество товаров, которые покупаются пользователем, уменьшается на покупаемое количество. Шестой пункт меню выведет общий чек всех покупок пользователя.

Ваши покупки из производственных товаров:				
Трвожок	Белакт	20.05.2022	20.06.2022	2 1.65 BYN
Общей стоимостью: 3.3 BYN				
Ваши покупки из непроизводственных:				
Общей стоимостью: 0 BYN				

Рисунок 8.12 – Оформление заказа

Пункты семь и восемь аналогичны по функционалу подобным пунктам в меню администратора. Последний пункт меню возвращает пользователя в меню авторизации.

Меню поставщика имеет вид, представленный на рисунке 8.13.

```
Добро пожаловать, Котлярова Яна Игоревна!  
Выбирайте:  
1. Поставка товаров на склад  
2. Просмотр товаров на складе  
3. Изменить информацию о товаре  
4. Удалить информацию о товаре  
5. Сортировка товаров по количеству  
6. Просмотреть личную информацию  
7. Изменить личную информацию  
8. Выход
```

Рисунок 8.13 – Меню поставщика

Первый пункт меню позволяет поставщику добавить товары на склад, предварительно выбрав вид добавляемых товаров, как показано на рисунке 8.14.

```

Выберите вид товаров:
1. Производственные
2. Непроизводственные
3. Предыдущее меню
2

Сколько товаров вы хотите добавить?
1

Введите данные:
Товар 1
Наименование товара: Брюки мужские
Фирма: Коминтерн
Дата изготовления: 10.05.2022
Размер:
Длина: 1.2
Ширина: 0.4
Высота: 1.2
Количество: 60
Цена: 56.50

```

Рисунок 8.14 – Добавление товара

Выбор второго пункта меню позволяет пользователю просмотреть все товары, поставленные им. Вывод товаров для поставщика представлен на рисунке 8.15.

Производственные товары:						
Наименование товара	Фирма	Дата изготовления	Годен до	Количество	Цена	
1. Молоко 2.5%	Савушкин продукт	12.05.2022	22.05.2022	200	1.97	BYN
2. Сметана 20%	Минская марка	10.05.2022	30.05.2022	49	2.05	BYN
3. Творожок	Белакт	20.05.2022	20.06.2022	53	1.65	BYN

Непроизводственные товары:						
Наименование товара	Фирма	Дата изготовления	Размер	Количество	Цена	
1. Платье летнее	8 марта	10.10.2021	1.2x0.4x0.7	23	35.99	BYN
2. Юбка-карандаш	Коминтерн	26.01.2021	0.6x0.4x0.6	35	23.55	BYN
3. Брюки мужские	Коминтерн	10.05.2022	1.2x0.4x1.2	60	56.5	BYN

Рисунок 8.15 – Вывод товаров для поставщика

При выборе третьего или четвёртого пунктов меню поставщику будет предложено выбрать тип товара, а после выведен список товаров и запрошен номер товара, который подлежит изменению или удалению. При изменении данных о товаре просится заново ввести данные о товаре, подобно тому, как это представлено на рисунке 8.14.

Пункты шесть и семь по использованию аналогичным подобным пунктам в меню администратора и клиента. Последний пункт возвращает пользователя в меню авторизации.

## ЗАКЛЮЧЕНИЕ

Основной причиной автоматизации является превосходство машин над человеком при обработке информации, объемах обрабатываемой информации за один промежуток времени и скорости этой обработки. Автоматизация позволяет, если не полностью исключить человеческие ошибки, то, по крайней мере, свести их к минимуму, поэтому всё больше областей жизни человека автоматизируется.

В результате разработки данного проекта была создана программа на языке C++, обеспечивающая автоматизированную систему ведения складского учёта и регистрации заказов, которая позволяет ускорить процесс и сделать его удобнее и проще.

Была спроектирована функциональная модель по стандарту IDEF0, позволяющая подробно рассмотреть функционирование складского помещения и учет его деятельности.

Для хранения данных были использованы несколько текстовых файлов, информация в которых была структурирована для быстрого и удобного поиска, для разделения информации по блокам и устранения возможностей потери информации.

Диаграммы классов были разработаны на основе UML, благодаря чему можно легко разобрать функционал разработанной программы, улучшать его, если это потребуется, и быстро найти реализацию необходимых методов.

Была разработана диаграмма вариантов использования приложения, которая позволяет рассказать про возможности разработанной программы по учету деятельности складского помещения без объяснения реализации.

Создана и описана схема работы всей программы, чтобы можно было посмотреть функционал без реализации, чтобы найти методы для улучшения функционала. Также были приведены две схемы основных алгоритмов программы, чтобы была возможность быстро проанализировать функционал.

Был описан алгоритм работы приложения за все роли, которые оно предоставляет, а также разобраны все методы и их результаты для каждой роли.

В результате разработки были достигнуты все поставленные задачи.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] 1С [Электронный ресурс]. – Обзор автоматизируемой предметной области, программных аналогов, методов и алгоритмов решения поставленной задачи – Режим доступа: <https://1c.ru/rus/products/1c/predpr/torg77.htm>

[2] moysklad [Электронный ресурс]. – Обзор автоматизируемой предметной области, программных аналогов, методов и алгоритмов решения поставленной задачи – Режим доступа: <https://www.moysklad.ru/vozmozhnosti/>

[3] antisklad [Электронный ресурс]. – Обзор автоматизируемой предметной области, программных аналогов, методов и алгоритмов решения поставленной задачи – Режим доступа: <https://antisklad.ru/>

[4] Методология IDEF0 – Учебная и научная деятельность Анисимова Владимира Викторовича [Электронный ресурс]. – Функциональное моделирование на основе стандарта IDEF0 – Режим доступа: [https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6/tema6\\_2](https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6/tema6_2)

[5] intuit [Электронный ресурс] – Разработка и описание диаграммы вариантов использования приложения – Режим доступа: <https://intuit.ru/studies/courses/32/32/lecture/1004>

# ПРИЛОЖЕНИЕ А (обязательное) Диаграмма классов

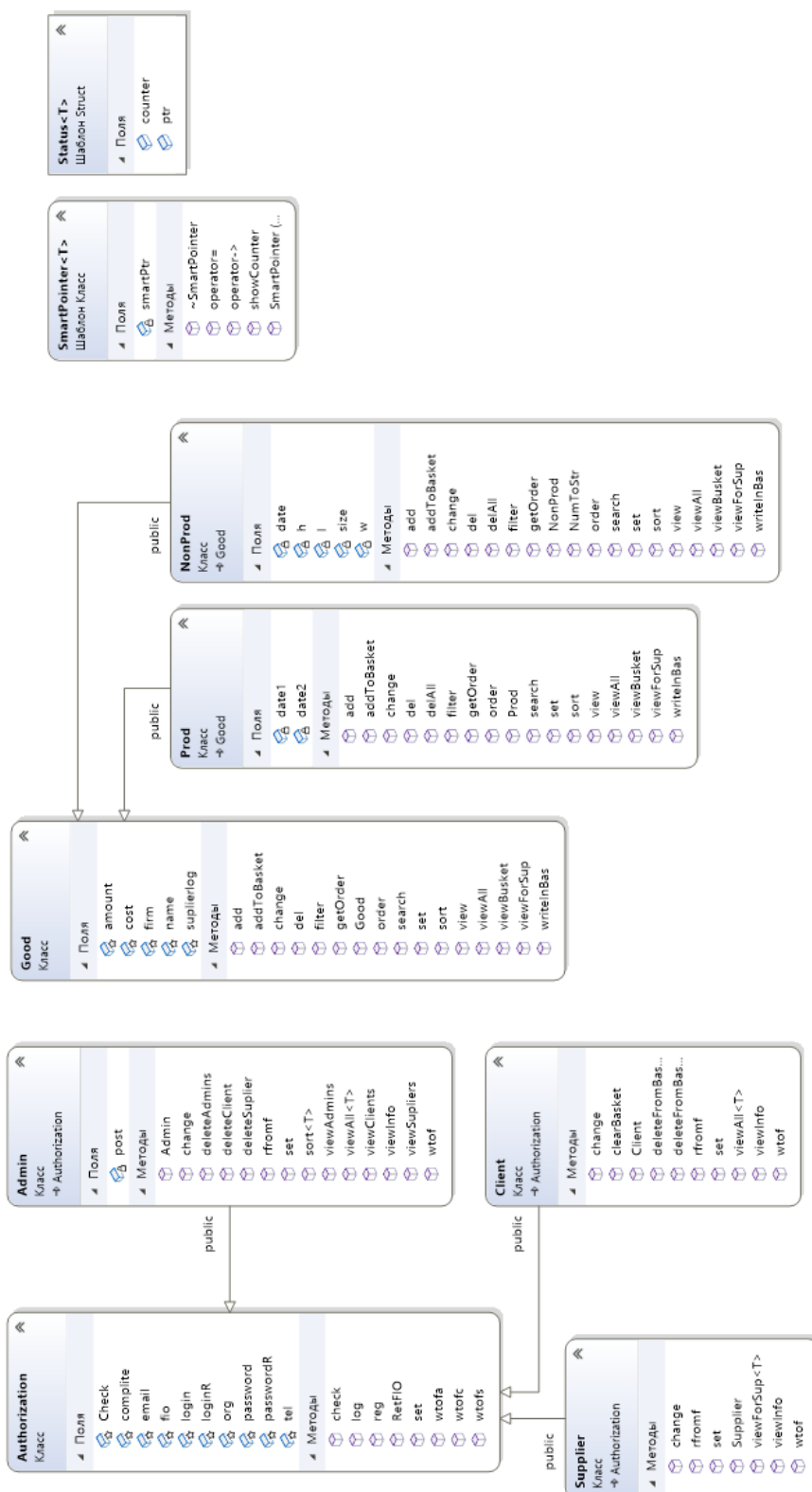


Рисунок А.1 – Диаграмма классов

## Диаграмма вариантов использования

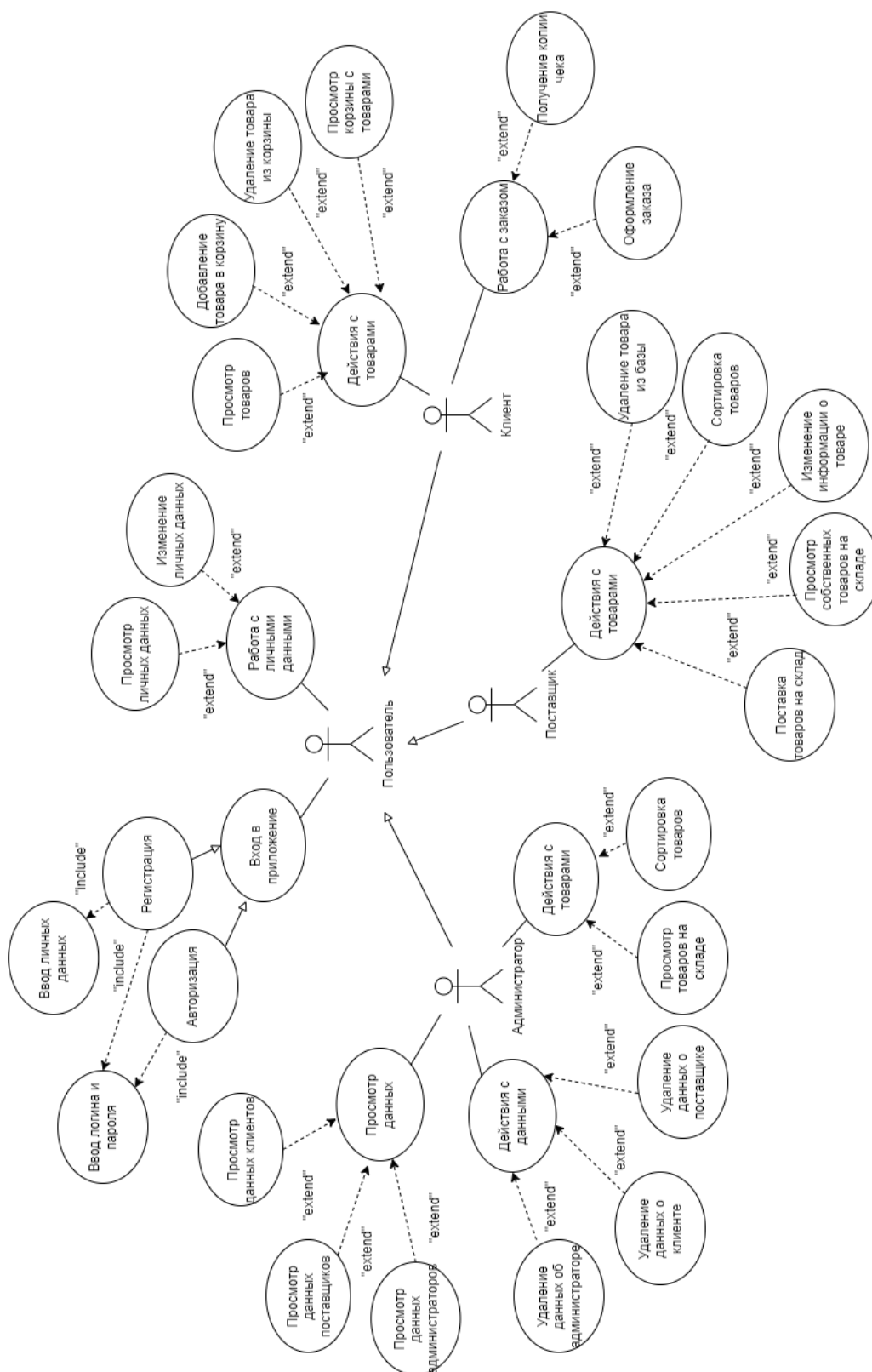


Рисунок Б.1 – Диаграмма вариантов использования

**ПРИЛОЖЕНИЕ В**  
**(обязательное)**  
**Схема алгоритма**

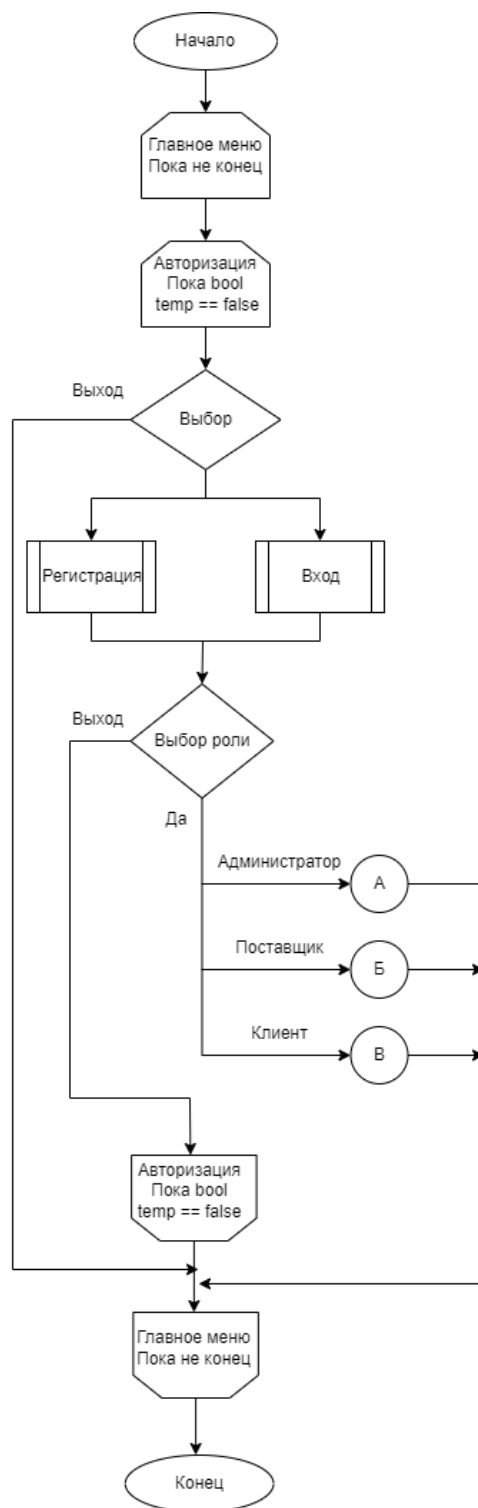


Рисунок В.1 – Блок-схема алгоритма работы программы

## Продолжение приложения В



Рисунок В.2 – Блок-схема алгоритма работы меню администратора

## Продолжение приложения В

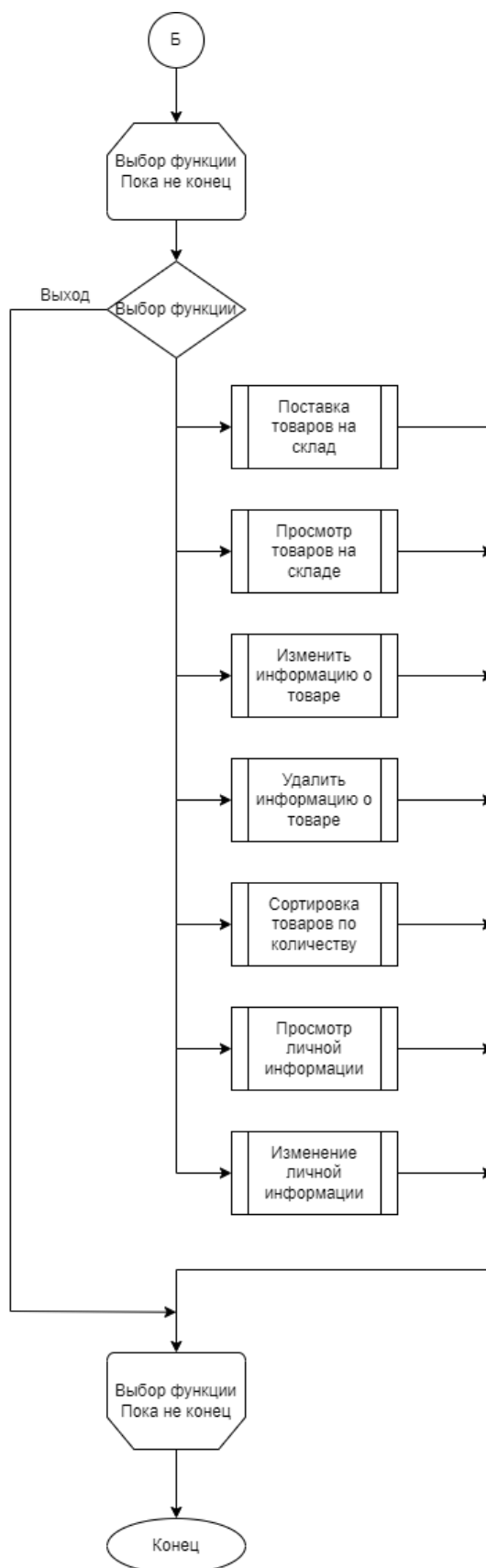


Рисунок В.3 – Блок-схема алгоритма работы меню поставщика

## Продолжение приложения В

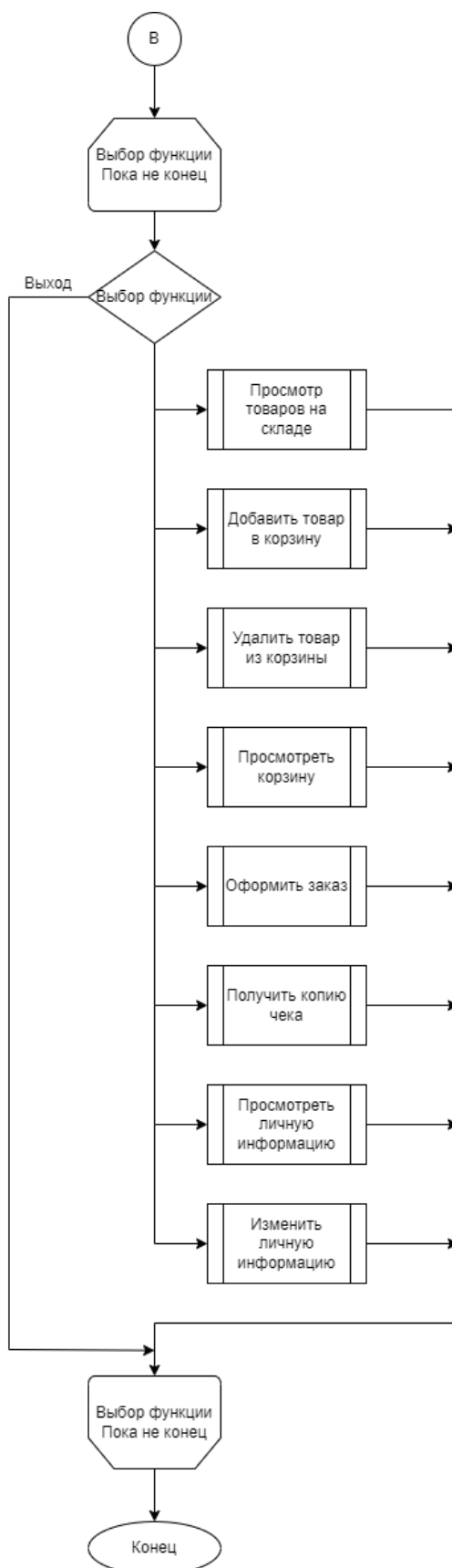


Рисунок В.4 – Блок-схема алгоритма работы меню клиента

## Продолжение приложения В

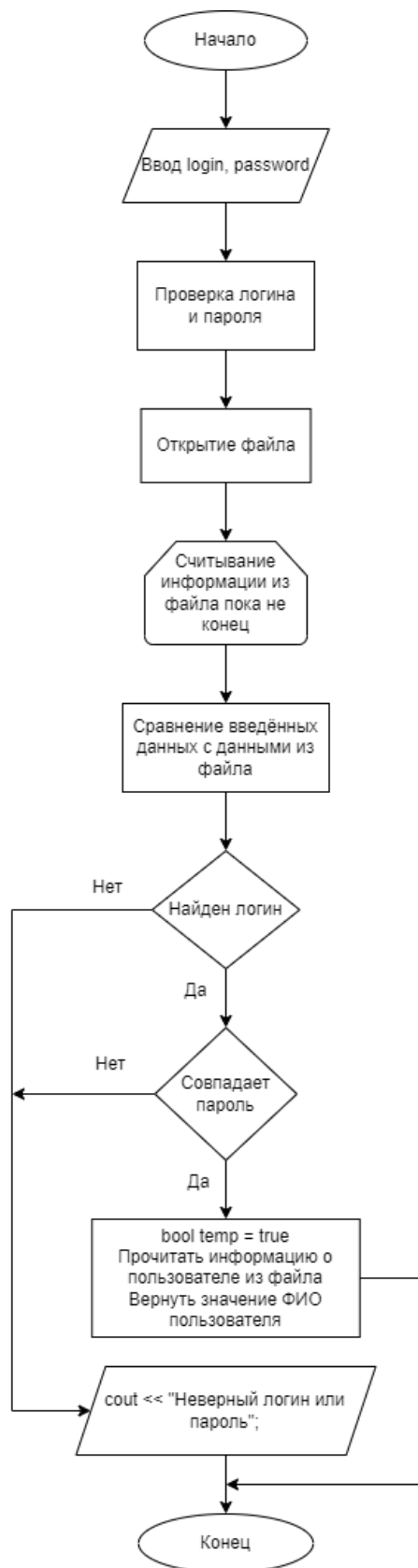


Рисунок В.5 – Блок-схема алгоритма работы функции авторизации пользователя



## Продолжение приложения В



Рисунок В.6 – Блок-схема алгоритма работы функции просмотра товаров на складе

## ПРИЛОЖЕНИЕ Г

### (обязательное)

### Листинг кода

#### 1. Реализация пользовательского меню:

##### Разделение ролей:

```
// объявление переменных
// ...
cout << "\nДобро пожаловать, " << fio << "!\n";
while (temp)
{
    switch (type)
    {
        case 1:
            switch (AdminMenu()) {...}
            break;
        case 2:
            switch (SupplierMenu()) {...}
            break;
        case 3:
            switch (ClientMenu()) {...}
            break;
        default: exit(0);
    }
}
```

##### Авторизация:

```
cin.ignore();
while (log[0] == '\0')
{
    cout << "Введите логин: ";
    log = FunctionForSpaces();
}
while (pass[0] == '\0')
{
    cout << "Введите пароль: ";
    pass = FunctionForSpaces();
}
hash = hasher(pass);
pass = to_string(hash);
temp = admin->log(log, pass, 1);
if (temp == false) cout << "Ошибка! Неверный логин или пароль.\n";
type = 1;
admin->rfromf(log);
fio = admin->RetFIO();

// реализация функции log
bool log(string login, string password, int type)
{
    string File_name;
    this->loginR = login;
    this->passwordR = password;
    if (type == 1) { File_name = "admin.txt"; }
    else if (type == 2) { File_name = "suplier.txt"; }
    else if (type == 3) { File_name = "client.txt"; }
    ifstream fin(File_name);
    while (getline(fin, Check))
```

## Продолжение приложения Г

```
{
    if (check(Check))
    {
        complite = true;
        break;
    }
    else
    {
        complite = false;
    }
}
return complite;
}
```

### Регистрация и шифрация пароля:

```
cin.ignore();
type = 1;
while (log[0] == '\0')
{
    cout << "Введите логин (без пробелов, они будут удалены): ";
    log = FunctionForSpaces();
}
while (pass[0] == '\0')
{
    cout << "Введите пароль (без пробелов, они будут удалены): ";
    pass = FunctionForSpaces();
}
hash = hasher(pass);
pass = to_string(hash);
temp = authorization->reg(log, pass, 1);
while (fio[0] == '\0')
{
    cout << "Введите ваше ФИО: ";
    getline(cin, fio);
}
while (post[0] == '\0')
{
    cout << "Введите вашу должность: ";
    getline(cin, post);
}
admin->set(log, fio, post);
admin->wtof();
break;
// реализация функций reg и check
bool reg(string login, string password, int type)
{
    ofstream fout;
    string File_name;
    if (type == 1) { File_name = "admin.txt"; }
    else if (type == 2) { File_name = "suplier.txt"; }
    else if (type == 3) { File_name = "client.txt"; }
    fout.open(File_name, ios::app);
    if (fout.is_open())
    {
        fout << login << " " << password << endl;
        fout.close();
        return true;
    }
}
```

## Продолжение приложения Г

```
else
{
    cout << "\nФайл пуст!\n";
    return false;
}

bool check(string Check)
{
    if (Check.find(loginR) >= 0 && Check.find(loginR) < 1000)
    {
        if (Check.find(passwordR) >= 0 && Check.find(passwordR) < 1000) {
return true; }
    }
    return false;
}
```

### Просмотр информации:

```
void viewInfo()
{
    cout.setf(ios::left);
    cout << setw(11) << "ФИО: " << this->fio << endl;
    cout << setw(11) << "Должность: " << this->post << endl;
}

void viewAdmins()
{
    Admin a;
    ifstream fin;
    fin.open("InfoAdmin.txt");
    if (fin.is_open())
    {
        string clientlog, clientFIO, clientpost;
        int count = 1;
        while (fin >> clientlog && fin.ignore() && getline(fin,
clientFIO, '\t') && getline(fin, clientpost, '\n'))
        {
            a.set(clientlog, clientFIO, clientpost);
            cout << endl;
            cout << "Администратор " << count++ << endl;
            a.viewInfo();
            cout << endl;
        }
        fin.close();
    }
}
```

### Добавление, редактирование и удаление записей:

```
void wtof()
{
    ofstream fout;
    fout.open("InfoAdmin.txt", ios::app);
    if (fout.is_open())
    {
        fout << endl;
        fout << login << '\t' << fio << '\t' << post;
        fout.close();
    }
}
```

## Продолжение приложения Г

```
    }

void change(string log)
{
    ifstream fin;
    Admin ToChange;
    vector<Admin> pr;
    fin.open("InfoAdmin.txt", ios::in);
    if (fin.is_open())
    {
        while (fin >> login && fin.ignore() && getline(fin, fio,
'\t') && getline(fin, post, '\n'))
        {
            if (login != log)
            {
                ToChange.set(login, fio, post);
                pr.push_back(ToChange);
            }
            else
            {
                fio[0] = '\\0'; post[0] = '\\0';
                cin.ignore();
                while (fio[0] == '\\0')
                {
                    cout << "Введите ваше ФИО: ";
                    getline(cin, fio);
                }
                while (post[0] == '\\0')
                {
                    cout << "Введите вашу должность: ";
                    getline(cin, post);
                }
                ToChange.set(login, fio, post);
                pr.push_back(ToChange);
            }
        }
        fin.close();
    }

    ofstream fout;
    fout.open("InfoAdmin.txt", ios::trunc);
    if (fout.is_open())
    {
        for (auto i = 0; i < pr.size(); i++)
        {
            pr[i].wtof();
        }
        pr.clear();
        fout.close();
        cout << "Редактирование прошло успешно" << endl;
    }
}

void deleteAdmins()
{
    ifstream fin;
    Admin clientDel;
    vector<Admin> admins;
    int countToDel = 0, i;
    string LogToDel;
```

## Продолжение приложения Г

```
string Log, Pass;
string client_log, client_fio, client_post;
viewAdmins();
cout << "Введите номер администратора для удаления: ";
cin >> i;
i--;
fin.open("InfoAdmin.txt");
if (fin.is_open())
{
    while (fin >> client_log && fin.ignore() && getline(fin,
client_fio, '\t') && getline(fin, client_post, '\n'))
    {
        if (countToDel++ != i)
        {
            clientDel.set(client_log, client_fio,
client_post);
            admins.push_back(clientDel);
        }
        else { LogToDel = client_log; }
    }
    fin.close();
    fin.clear();
}

ofstream fout;
fout.open("InfoAdmin.txt", ios::trunc);
if (fout.is_open())
{
    for (auto j = 0; j < admins.size(); j++)
    {
        admins[j].wtof();
    }
    fout.close();
    fout.clear();
    admins.clear();
}

vector<Authorization> vecA;
Authorization admin;
fin.open("admin.txt");
if (fin.is_open())
{
    while (fin >> Log >> Pass)
    {
        if (Log != LogToDel)
        {
            admin.set(Log, Pass);
            vecA.push_back(admin);
        }
    }
    fin.close();
}

fout.open("admin.txt", ios::trunc);
if (fout.is_open())
{
    for (auto i = 0; i < vecA.size(); i++)
    {
        vecA[i].wtofa();
    }
}
```

## Продолжение приложения Г

```
        cout << "Удаление прошло успешно!" << endl;
    }
}
```

### Поиск, сортировка и фильтрация записей:

```
bool search()
{
    ifstream fin;
    Prod ToSearch;
    vector<Prod> pr, temp;
    int i = 0;
    bool t = false;
    string nameForSearch;
    cout << "Введите наименование искомого товара: ";
    cin.ignore();
    getline(cin, nameForSearch);
    fin.open("ProdGoods.txt", ios::in);
    if (fin.is_open())
    {
        while (fin.ignore() && getline(fin, name, '\\t') &&
        getline(fin, firm, '\\t') && fin >> datel >> date2 >> amount >> cost >>
        supplierlog)
        {
            ToSearch.set(name, firm, datel, date2, amount, cost,
            supplierlog);
            pr.push_back(ToSearch);
        }
        fin.close();
    }
    for (auto i = 0; i < pr.size(); i++)
    {
        if (pr[i].name == nameForSearch)
        {
            t = true;
            cout << "Поиск прошёл успешно.\n";
            cout.setf(ios::left);
            cout << setw(28) << "Наименование товара " << setw(25)
            << "Фирма " << setw(19) << "Дата изготовления "
            << setw(11) << "Годен до " << setw(12) <<
            "Количество " << setw(4) << "Цена" << endl;
            cout.setf(ios::left);
            cout << i + 1 << ". " << setw(25) << pr[i].name <<
            setw(25) << pr[i].firm << setw(19) << pr[i].datel << setw(11)
            << pr[i].date2 << setw(10) << right <<
            pr[i].amount << setw(6) << pr[i].cost << left << " BYN " << endl;
        }
        else if (i == pr.size()) cout << "Подходящих товаров на
        складе нет.\n";
    }
    pr.clear();
    return t;
}

void sort()
{
    ifstream fin;
    Prod ToChange;
    vector<Prod> pr;
```

## Продолжение приложения Г

```
    fin.open("ProdGoods.txt", ios::in);
    if (fin.is_open())
    {
        while (fin.ignore() && getline(fin, name, '\t') &&
getline(fin, firm, '\t') && fin >> datel >> date2 >> amount >> cost >>
supplierlog)
        {
            ToChange.set(name, firm, datel, date2, amount, cost,
supplierlog);
            pr.push_back(ToChange);
        }
        fin.close();
    }
    for (auto i = 1; i < pr.size(); i++)
        for (auto j = pr.size() - 1; j >= i; j--)
            if (pr[j - 1] > pr[j])
            {
                Prod temp;
                temp = pr[j - 1];
                pr[j - 1] = pr[j];
                pr[j] = temp;
            }
    ofstream fout;
    fout.open("ProdGoods.txt");
    if (fout.is_open())
    {
        for (auto i = 0; i < pr.size(); i++)
        {
            pr[i].add();
        }
        fout.close();
    }
    pr.clear();
    cout << "Сортировка производственных товаров прошла успешно" <<
endl;
}

void filter(double maxcost)
{
    ifstream fin;
    Prod ToFilt;
    vector<Prod> pr, temp;
    short i = 0;
    string tmp;
    fin.open("ProdGoods.txt", ios::in);
    if (fin.is_open())
    {
        while (fin.ignore() && getline(fin, name, '\t') &&
getline(fin, firm, '\t') && fin >> datel >> date2 >> amount >>
supplierlog)
        {
            ToFilt.set(name, firm, datel, date2, amount, cost,
supplierlog);
            pr.push_back(ToFilt);
        }
        fin.close();
    }
    cout << "Фильтрация прошла успешно.\nВаш отфильтрованный список
товаров:\n";
    cout.setf(ios::left);
```



## Продолжение приложения Г

```
cout << setw(28) << "Наименование товара " << setw(25) << "Фирма " <<
setw(19) << "Дата изготовления "
    << setw(11) << "Годен до " << setw(12) << "Количество " <<
setw(6) << "Цена" << endl;
for (auto i = 0; i < pr.size(); i++)
{
    if (pr[i].cost <= maxcost)
    {
        cout.setf(ios::left);
        cout << i+1 << ". " << setw(25) << pr[i].name <<
setw(25) << pr[i].firm << setw(19) << pr[i].date1 << setw(11)
            << pr[i].date2 << setw(10) << right <<
pr[i].amount << setw(6) << pr[i].cost << left << " BYN " << endl;
    }
    else if (i == pr.size() - 1) cout << "Подходящих товаров на
складе нет.\n";
}
pr.clear();
}
```

2. Исходная и итоговая информация хранится в бинарных файлах. Для хранения в оперативной памяти используются контейнеры STL:

### Хранение в файлах:

```
void wtof()
{
    ofstream fout;
    fout.open("InfoAdmin.txt", ios::app);
    if (fout.is_open())
    {
        fout << endl;
        fout << login << '\t' << fio << '\t' << post;
        fout.close();
    }
}

void rfromf(string l)
{
    ifstream fin;
    fin.open("InfoAdmin.txt");
    if (fin.is_open())
    {
        while (fin >> login && fin.ignore() && getline(fin, fio,
'\t') && getline(fin, post, '\n'))
        {
            if (login == l) break;
        }
        fin.close();
    }
}
```

### Использование контейнеров STL:

```
// ...
vector<Admin> pr;
vector<Supplier> suppliers;
vector<Client> clients;
// ...
```

## Продолжение приложения Г

### 3. Предусмотрены следующие аспекты:

#### Реализация базовых принципов ООП:

##### Абстракция:

```
class Good
{
protected:
    string name, firm, supplierlog;
    int amount = 0;
    double cost = 0;
public:
    Good() {}
    virtual void set(string, string, string, string, int, double, string) =
0;
    virtual void add() = 0;
    virtual void writeInBas(string) = 0;
    virtual void view(int) const = 0;
    virtual void viewAll() = 0;
    virtual void viewBasket(string) = 0;
    virtual void viewForSup(string) = 0;
    virtual void del(int, string) = 0;
    virtual void change(int, string) = 0;
    virtual void sort() = 0;
    virtual void filter(double) = 0;
    virtual void addToBasket(string) = 0;
    virtual void order(string) = 0;
    virtual bool search() = 0;
    virtual void getOrder(string) = 0;
};
```

##### Наследование:

```
class Prod : public Good
{...}
```

##### Инкапсуляция:

```
class Authorization
{
protected:
    //...
    string fio;
public:
    //...
    string RetFIO() { return fio; }
}
```

##### Использование пространств имён (стандартных и собственных):

```
using namespace std;

namespace Menu
{
    int AutorizationMenu1() {...}
    int AutorizationMenu2() {...}
    int AutorizationMenu3() {...}
    int AdminMenu() {...}
}
```

## Продолжение приложения Г

```
int SupplierMenu() {...}  
int ClientMenu() {...}  
}
```

```
using namespace Menu;
```

**Использование стандартных, пользовательских, дружественных, виртуальных функций:**

**Дружественные:**

```
class Authorization  
{  
    // ...  
    friend ostream& operator<<(ostream& stream, Authorization obj)  
    {  
        stream << obj.login << "\t";  
        stream << obj.password << "\t";  
        return stream;  
    }  
}
```

**Виртуальные:**

```
class Good  
{  
    //...  
    virtual void add() = 0;  
    //...  
}
```

Пользовательские и стандартные функции используются везде выше.

**Реализация обработки ошибок:**

```
int FunctionForCheck()  
{  
    int i = 0;  
    double d;  
    while (i == 0)  
    {  
        try  
        {  
            cin >> d;  
            i = d;  
            if (i != d) throw 0.1;  
            if (i <= 0) throw 10;  
            if (cin.fail()) throw "aaa";  
            if (!cin) throw "aaa";  
        }  
        catch (double)  
        {  
            cin.clear();  
            cin.ignore(10, '\n');  
            cout << "Вы ввели не целое число. Повторите ввод." << endl;  
            i = 0;  
        }  
        catch (int)  
        {  

```

## Продолжение приложения Г

```
        cin.clear();
        cin.ignore(10, '\n');
        cout << "Вы ввели отрицательное число или 0. Повторите
ввод." << endl;
        i = 0;
    }
    catch (exception&)
    {
        cin.clear();
        cin.ignore(10, '\n');
        cout << "Вы ввели символ вместо числа. Повторите ввод." <<
endl;
        i = 0;
    }
    catch (...)
    {
        cin.clear();
        cin.ignore(10, '\n');
        cout << "Ошибка ввода. Повторите ввод." << endl;
        i = 0;
    }
}
return i;
}
```

### Использование перегрузки методов:

```
class Authorization
{
    //...
    void set(string login, string password)
    {
        loginR = login;
        passwordR = password;
    }
    //...
}
class Admin : public Authorization
{
    //...
    void set(string l, string f, string p)
    {
        this->login = l;
        this->fio = f;
        this->post = p;
    }
    //...
}
```

### Использование перегрузки операторов:

```
class Prod : public Good
{
    //...
    friend bool operator>(const Prod& obj1, const Prod& obj2)
    {
        return obj1.amount > obj2.amount;
    }
    friend bool operator==(Prod a, Prod b)
```

## Продолжение приложения Г

```
        { return ((a.name) == (b.name) && ((a.cost) == (b.cost)) &&
((a.supplierlog) == (b.supplierlog))); }
//...
}
```

### Использование шаблонных классов и методов:

```
template<class T> class SmartPointer
{
private:
    Status<T>* smartPtr;
public:
    SmartPointer(T* ptr = 0){...}
    SmartPointer(const SmartPointer& obj){...}
    ~SmartPointer() {...}
    //...
    T* operator->() const {...}
    //...
}
```

### Использование динамической памяти и умных указателей:

```
int main()
{
//...
    SmartPointer<Client> client = new Client();
    SmartPointer<Admin> admin = new Admin();
    SmartPointer<Supplier> supplier = new Supplier();
    SmartPointer<Authorization> authorization = new Authorization();
    SmartPointer<Prod> prod = new Prod();
    SmartPointer<NonProd> nonprod = new NonProd();
//...
}
```

Использование потоков C++, перегрузки операторов ввода/вывода, контролирования работы с потоком приведено в функциях выше.