Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Основы алгоритмизации и программирования»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  Доцент кафедры ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Н. А. Кириенко |
|  | \_\_\_.\_\_\_\_.2025 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему:

**«Разработка программы учёта издательств для книжного магазина»**

Вариант 1

|  |  |
| --- | --- |
|  | Выполнил студент группы 373601  Садовский Денис Юрьевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2025  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2025

**РЕФЕРАТ**

БГУИР КР 11-40 05 01-02 001 ПЗ

**Садовский, Д.Ю.** Разработка программы учёта издательств для книжного магазина /Д.Ю.Садовский. – Минск: БГУИР, 2025. – 44 с.

Пояснительная записка 44 с., 33 рис., 5 табл., 10 источников, 2 приложения

учёт издательств для книжного магазина, модели *UML*, *IDEF*0, *BPMN*, схемы алгоритмов, программное средство

*Цель* *курсовой работы*: учесть издательства для книжного магазина. Программа позволяет вносить новые и удалять старые данные, сохранять отредактированные записи, осуществлять сортировку данных по заданному условию, выводить информацию в необходимом формате.

*Методология проведения работы*: в процессе решения поставленных задач использованы принципы системного подхода, аналитические методы, методы компьютерной обработки экспериментальных данных и компьютерного моделирования.

*Результаты работы*: выполнена постановка задачи и определены основные методы ее решения; в ходе объектного моделирования системы построен ряд *UML*-диаграмм; разработаны модели бизнес-процессов предметной области на основе нотаций IDEF0 и BPMN; описаны основные алгоритмы работы программного средства; разработано руководство пользователя; выполнено тестирование программного средства, показавшее его соответствие функциональным требованиям, поставленным в задании на разработку.

Программный продукт разработан на языке *C++* с применением *MS Visual Studio 2022*.

*Область применения результатов*: c помощью разработанного программного средства можно учесть издательства для книжного магазина.

Разработанное программное средство полностью отвечает всем функциональным требованиям, необходимым для учёта издательств для книжного магазина.

**ЗАДАНИЕ ПО КУРСОВОМУ ПРОЕКТУ**

**Вариант 1**

Разработать приложение управления списком объектов структурного типа, отражающего данные учета издательств для книжного магазина. В разрабатываемом приложении обеспечить ввод структуры данных список из файла, операции добавления, редактирования, удаления элементов списка с определенными значениями полей, сортировку списка по заданному полю, вывод на экран отсортированного списка. Количество полей в структурной переменной не менее 4-х, типы полей разные.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc135579363)

[1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ  
 ПРОГРАММННОГО СРЕДСТВА 5](#_Toc135579364)

[1.1 Описание предметной области 5](#_Toc135579365)

[1.2 Разработка функциональной модели предметной области 8](#_Toc135579366)

[1.3 Разработка BPMN – модели основного процесса предметной области. 13](#_Toc135579367)

[1.4 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований. 14](#_Toc135579368)

[1.5 UML-модель представления программного обеспечения и ее описание. 16](#_Toc135579369)

[2 Проектирование и конструирование программного средства 22](#_Toc135579370)

[2.1 Постановка задачи 22](#_Toc135579371)

[2.2 Разработка модульной структуры 22](#_Toc135579372)

[2.3 Выбор способа организации данных 23](#_Toc135579373)

[2.4 Разработка перечня пользовательских функций 25](#_Toc135579374)

[2.5 Разработка схем алгоритмов работы программы 27](#_Toc135579375)

[3 Тестирование и проверка работоспособности программного средства 31](#_Toc135579376)

[4 Инструкция по развёртыванию приложения и сквозной пример 33](#_Toc135579377)

[4.1 Авторизация 33](#_Toc135579378)

[4.2 Модуль администратора 34](#_Toc135579379)

[4.3 Модуль пользователя 35](#_Toc135579380)

[Заключение 37](#_Toc135579381)

[Список используемой литературы 38](#_Toc135579382)

[Приложение А Отчет о проверке на заимствование в системе «Антиплагиат» 39](#_Toc135579383)

[Приложение Б Листинг кода алгоритмов, реализующих основную бизнес-логику 40](#_Toc135579384)

**ВВЕДЕНИЕ**

В современном мире книжные магазины играют важную роль в распространении знаний и культурных ценностей. Они являются не только местом продажи книг, но и центром обмена информацией, где читатели могут найти литературу на любой вкус и интерес. Однако с ростом ассортимента и увеличением количества издательств, сотрудничающих с книжными магазинами, возникает необходимость в эффективном управлении данными об издательствах и ассортименте книг.

Актуальность данной темы обусловлена тем, что многие книжные магазины до сих пор используют устаревшие методы учета издательств, такие как бумажные журналы. Внедрение специализированной программы учета издательств может значительно упростить процесс управления данными, повысить точность учета и улучшить анализ сотрудничества с издательствами.

Целью данной курсовой работы является разработка программы учета издательств для книжного магазина, которая позволит автоматизировать процесс учета, упростить взаимодействие с поставщиками и повысить эффективность управления ассортиментом.

Предметом исследования является программа учета издательств, которая будет использоваться для автоматизации процессов учета и анализа данных о поставщиках книжной продукции.

Объектом исследования выступают книжные магазины, которые сотрудничают с множеством издательств и нуждаются в эффективном инструменте для учета и управления данными.

Для достижения цели данной работы были поставлены следующие задачи:

* изучение предметной области учета издательств для книжного магазина и существующих методов управления информацией об издательствах;
* определение требований к разрабатываемой программе учета издательств для книжного магазина, включая функциональные и нефункциональные требования;
* разработка программного средства учета издательств с использованием языка программирования C++;
* тестирование и отладка разработанной программы для обеспечения соответствия заявленным требованиям;
* оценка эффективности и практической ценности разработанной системы учета издательств для книжного магазина.

**1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММННОГО СРЕДСТВА**

**1.1 Описание предметной области**

Книжные магазины являются важным звеном в цепочке распространения литературы, обеспечивая доступ читателей к широкому ассортименту книг от различных издательств. В условиях растущего рынка и увеличения количества издательств, сотрудничающих с книжными магазинами, возникает необходимость в эффективном управлении данными о поставщиках, ассортименте и наличии книг. Одной из ключевых задач в этом процессе является учет издательств, которые поставляют литературу в магазин.

Предметная область программы учета издательств для книжного магазина включает в себя процессы, связанные с управлением информацией об издательствах, их продукции, условиях сотрудничества и поставок. Основные характеристики предметной области:

* издательства: каждое издательство имеет уникальные данные, такие как название, контактная информация, реквизиты, условия сотрудничества и список поставляемой литературы;
* жанры книг: издательства могут специализироваться на различных жанрах (художественная литература, научная литература, учебники и т.д.), что влияет на ассортимент книжного магазина;
* книги: каждая книга имеет свои характеристики: название, автор, жанр, год издания, ISBN, количество экземпляров на складе и цена;
* происхождение книг: книги могут поставляться от разных издательств, что требует учета их происхождения;
* условия сотрудничества: книжные магазины заключают договоры с издательствами, которые определяют условия поставок, скидки, сроки оплаты и другие параметры;
* аналитика и отчетность: для эффективного управления ассортиментом необходимы инструменты анализа данных, такие как отчеты по продажам, остаткам и популярности книг, что помогает принимать обоснованные решения о закупках и сотрудничестве с издательствами.

Актуальность данной темы заключается в том, что многие книжные магазины продолжают использовать устаревшие способы учета издательств, такие как ведение бумажных журналов. Разработка и внедрение специализированной программы для учета издательств позволит существенно упростить процесс управления данными, повысить их точность и улучшить анализ взаимодействия с издательскими компаниями.

На рынке существует множество программных решений для учета издательств и управления ассортиментом книжных магазинов. Рассмотрим некоторые из них:

* “1С:Управление торговлей”: популярная система для автоматизации торговых операций, включая учет поставщиков, управление ассортиментом и анализ продаж, подходит для крупных книжных сетей, но может быть избыточной для небольших магазинов;
* “МойСклад”: облачное решение для учета товаров, управления заказами и анализа продаж; простое в использовании, но имеет ограниченные возможности для специализированного учета издательств;
* “ЛитРес:Управление”: специализированное решение для книжных магазинов, включающее учет издательств, управление ассортиментом и интеграцию с электронными книгами; подходит для магазинов, работающих как с печатной, так и с электронной литературой;
* “Библиотекарь”: программа для учета книг и издательств, разработанная специально для библиотек и книжных магазинов, включает функции учета поставок, управления ассортиментом и генерации отчетов;
* “BookManager”: специализированное решение для управления книжным ассортиментом, включающее учет издательств, управление заказами и анализ продаж; подходит для небольших и средних книжных магазинов.

Эти программные аналоги предоставляют различные функции для учета издательств и управления ассортиментом, но могут не полностью соответствовать потребностям конкретного книжного магазина. Разработка специализированной программы учета издательств позволит учесть все особенности работы магазина и повысить эффективность управления данными.

Разработка программы учета издательств для книжного магазина требует применения современных методов и алгоритмов, обеспечивающих эффективное управление данными, удобство взаимодействия с пользователем и высокую производительность системы. В рамках решения задачи можно использовать методы хранения данных, включая массивы структур и файловую систему, что позволит сохранять информацию между сеансами работы программы.

Для упорядочивания данных можно добавить алгоритмы сортировки, такие как пузырьковая сортировка, которая, несмотря на свою простоту, эффективна для небольших объемов данных. Если же программа будет работать с большим количеством информации, то предпочтительнее использовать более быстрые алгоритмы сортировки, такие как быстрая сортировка или сортировка слиянием. Поиск данных в программе будет реализован с помощью линейного поиска, который подходит для небольших объемов информации, а для повышения производительности при работе с отсортированными данными можно применить бинарный поиск.

Обработка текстовой информации, такой как названия книг и издательств, будет осуществляться с использованием методов форматирования строк и сравнения. Это обеспечит корректное отображение данных в таблицах и удобство восприятия информации пользователем. Взаимодействие с пользователем будет организовано через консольный интерфейс, который предоставит меню для выбора действий и выведет данные в структурированном виде. Для предотвращения ошибок ввода будет реализована валидация данных, что повысит надежность программы. Кроме того, для удобства пользователя можно добавить систему подсказок и инструкций.

Анализ данных об издательствах и книгах включает статистические расчеты, такие как определение среднего рейтинга издательств и количества книг по различным категориям. Эти расчеты помогут принимать обоснованные управленческие решения, например, касающиеся закупок новых книг или изменения условий сотрудничества с издательствами. Для этого можно реализовать соответствующие алгоритмы обработки данных и формирования отчетов.

Учитывая, что эта система ориентирована на упрощение учёта издательств для книжного магазина с использованием файлов, алгоритмы могут быть реализованы в виде функций и методов языка программирования C++, предназначенных для работы с файлами, обработки данных и взаимодействия с пользователем.

Тестирование программы включает модульное и интеграционное тестирование, а также проверку граничных случаев, что обеспечит корректность работы всех компонентов системы. Модульное тестирование будет проверять работоспособность отдельных функций программы, таких как добавление, редактирование и удаление данных, а интеграционное тестирование проверит взаимодействие различных частей программы. Особое внимание будет уделено тестированию на граничных значениях данных, чтобы убедиться в надежности программы при работе с экстремальными данными.

Таким образом, применение описанных методов и алгоритмов позволит создать эффективное программное решение для учета издательств в книжном магазине, обеспечивая высокую производительность, удобство работы с данными и надежность системы.

**1.2 Разработка функциональной модели предметной области**

Рисунок 1.1 – “Продать товары книжного магазина”.

Контекстная диаграмма содержит следующие параметры:

* входные параметры: спрос, клиенты, деньги;
* выходные параметры: данные о продажах, прибыль, данные о списании товаров;
* управляющие параметры: издательство, администратор магазина, сотрудники магазина;
* исполнительные параметры: условия рынка, законодательство, политики организации.

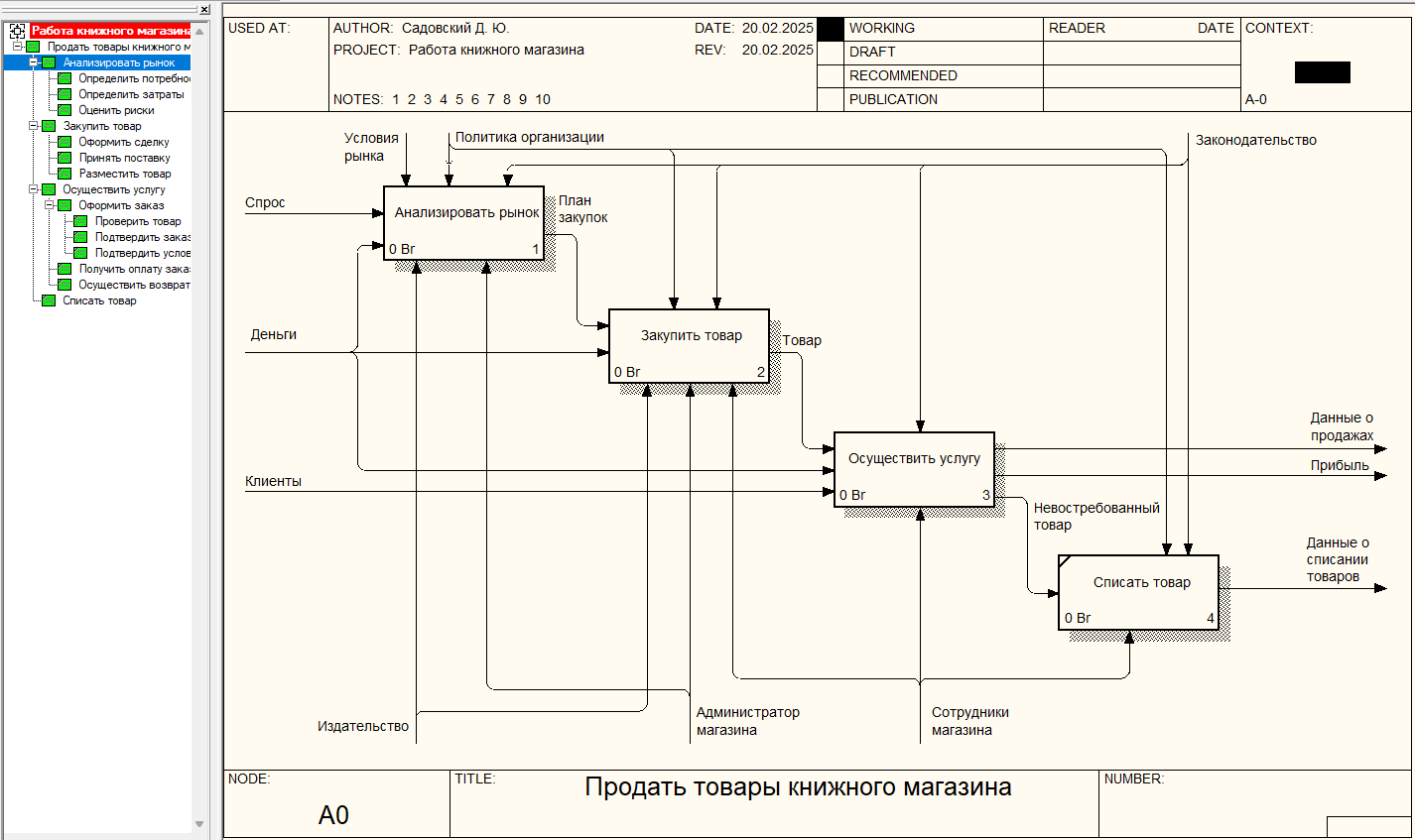


Рисунок 1.2 – Диаграмма декомпозиции.

Сначала необходимо произвести анализ рынка, затем закупить товар. После покупки происходит продажа выбранных товаров и списание невостребованных либо бракованных товаров для освобождения места.

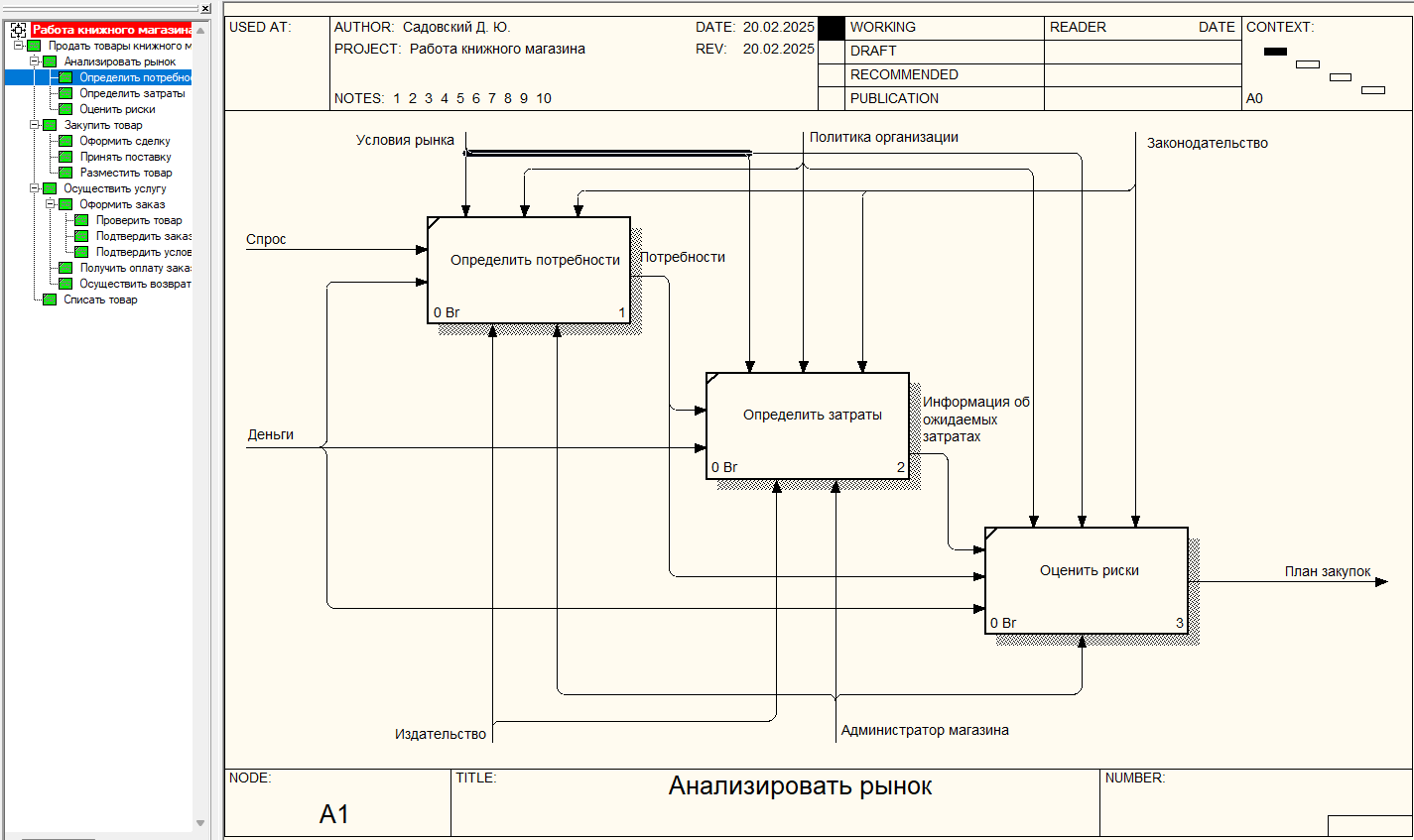


Рисунок 1.3 – Диаграмма декомпозиции процесса “Анализировать рынок”.

Во время анализа рынка происходит определение потребностей в данный момент времени, просчитываются ожидаемые затраты и происходит оценка рисков.

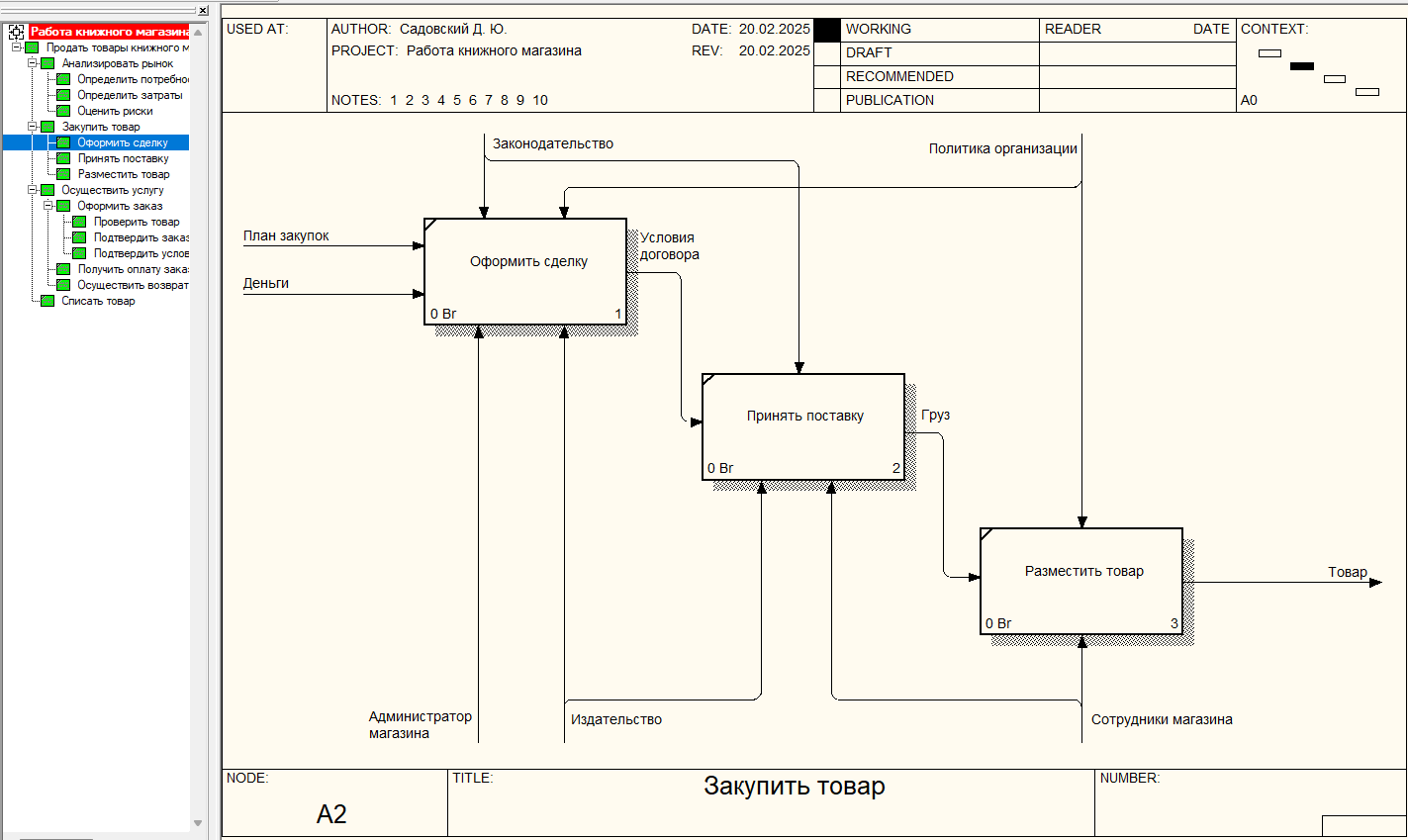


Рисунок 1.4 – Диаграмма декомпозиции процесса “Закупить товар”.

Первая функция – “Оформить сделку”. Она заключается в процессе составления договора между издательством и книжным магазином. “Принять поставку” – это функция, в которой происходит привоз выбранных товаров на склад либо сразу в книжный магазин. Затем сотрудники размещают полученный товар на полки магазина.

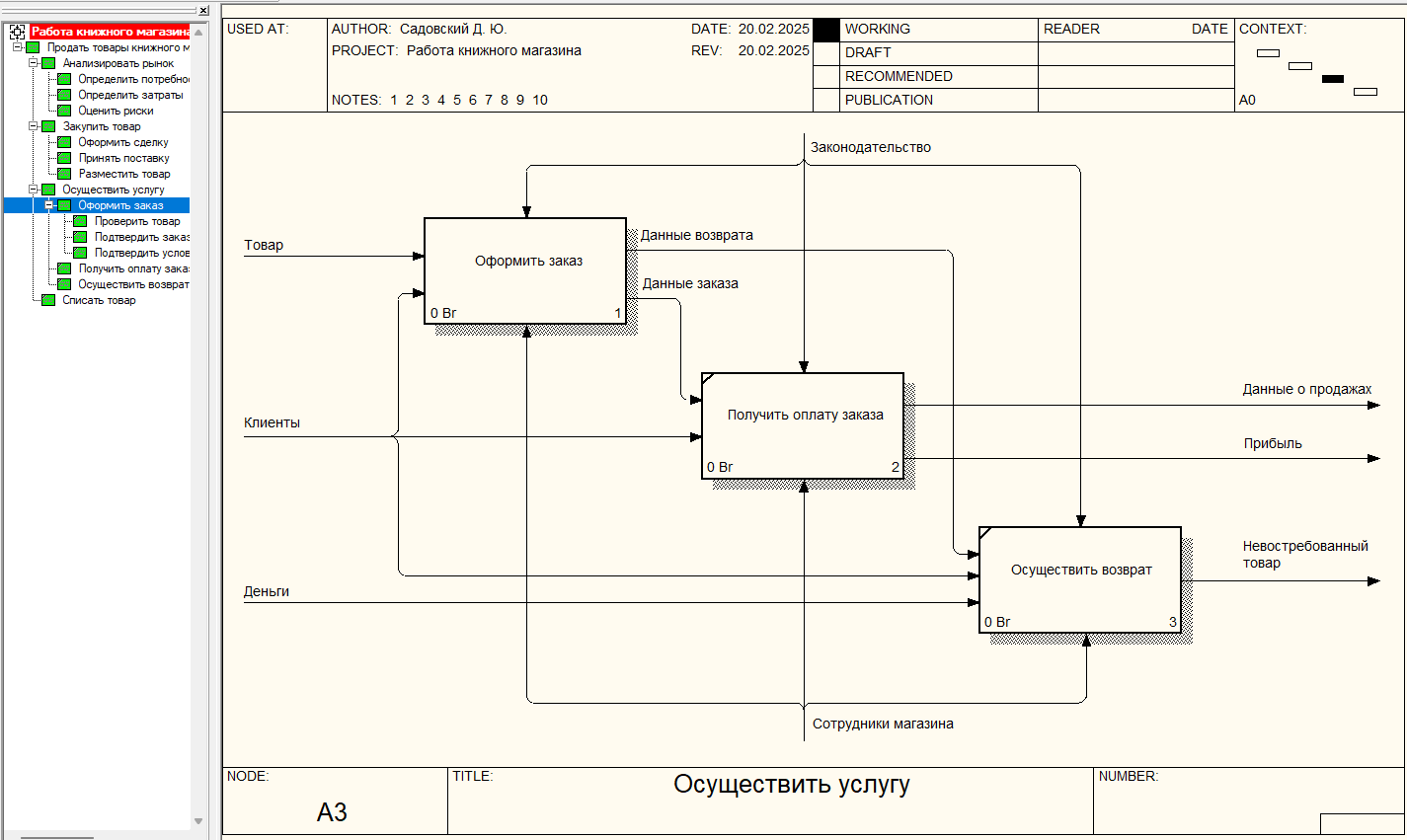


Рисунок 1.5 – Диаграмма декомпозиции процесса “Осуществить услугу”.

“Оформить заказ” – в зависимости от типа услуг происходит проверка товара для покупки, либо сотрудник проверяет возможность возврата. После осуществляется оплата выбранного товара. Если товар соответствует условиям, то происходит возврат товара.

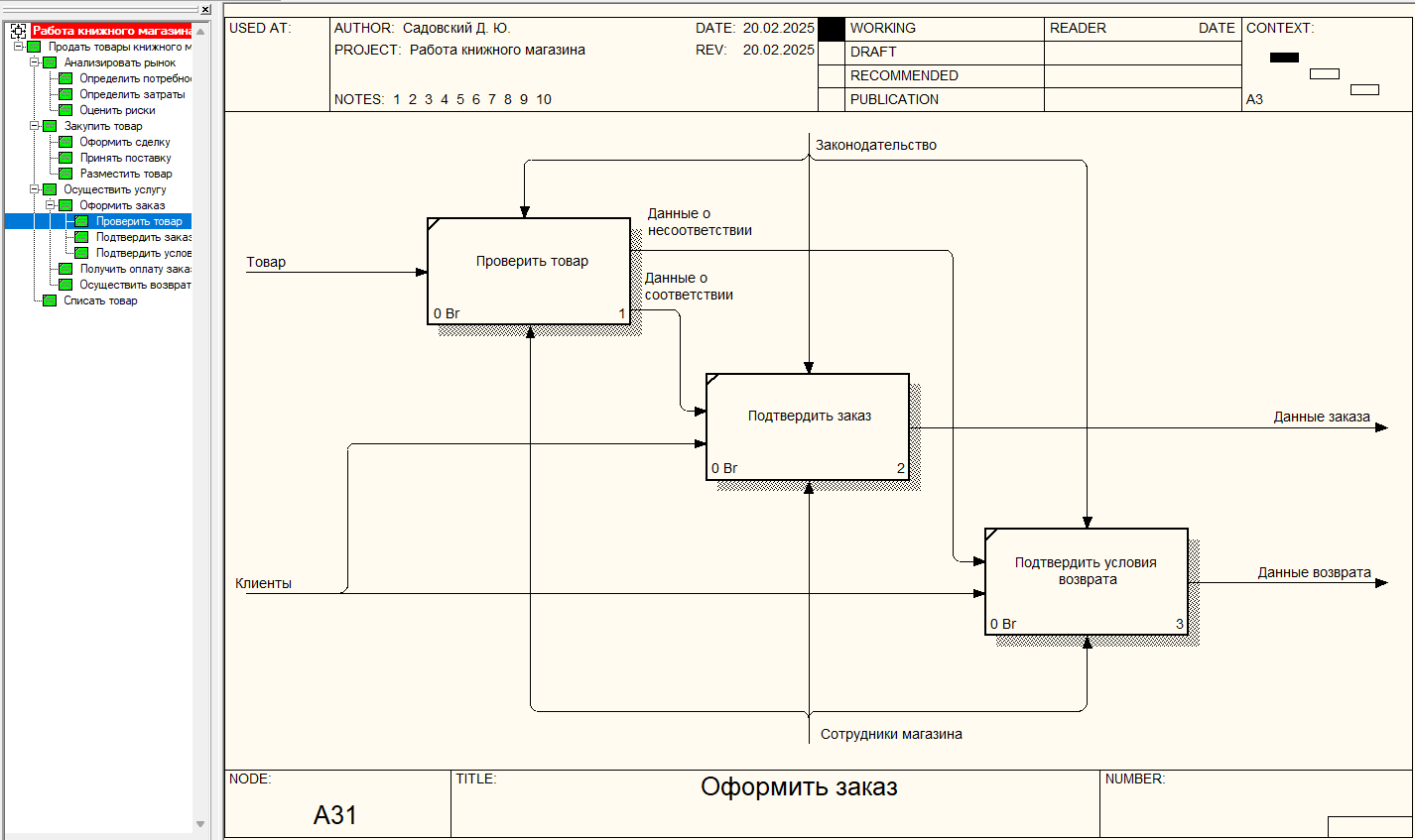


Рисунок 1.6 – Диаграмма декомпозиции процесса “Оформить заказ”.

Во время оформления заказа сотрудник магазина осуществляет проверку товара и данных клиента для продажи. Если клиент хочет оформить возврат, то происходит проверка условий для возврата товара.

**1.3 Разработка BPMN** – **модели основного процесса предметной области.**

BPMN (Business Process Model and Notation) – это стандартная нотация для моделирования бизнес-процессов. Она позволяет графически описывать потоки работы, взаимодействие между различными участниками и системами, а также управление данными в рамках конкретной предметной области.

Для предметной области покупки и заказа товара в BPMN можно создать следующую модель (рисунок 1.7):

* начальное событие: начало процесса покупки и заказа товара;
* задача: выбор товара – пользователь выбирает товар из предложенного ассортимента;
* задача: оформление заказа – пользователь заполняет необходимую информацию для оформления заказа, такую как адрес доставки, способ оплаты и т.д;
* эксклюзивное шлюзовое событие: выбор способа доставки – пользователь указывает способ получения товара: доставка или самовывоз;
* эксклюзивное шлюзовое событие: выбор способа оплаты – пользователь указывает способ оплаты товара: картой на сайте или при получении;
* эксклюзивное шлюзовое событие: проверка наличия товара – система проверяет наличие выбранного товара на складе;
* задача: отправка заявки – система отправляет заявку на товар компании;
* задача: получение ответа - пользователь получает ответ, от которого зависит, получит ли он товар;
* эксклюзивное шлюзовое событие: какой ответ на заявку – заявка может быть как принята, так и нет;
* задача: оплата и получение товара – если заявка на заказ принята клиент получит заказ;
* конечное событие: завершение процесса покупки и заказа товара.

**Изображение выглядит как текст, снимок экрана, линия, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.**

Рисунок 1.7 – BPMN-модель процесса “Покупка товара”.

Это лишь пример базовой модели BPMN для предметной области покупки и заказа товара. Реальная модель может содержать более детальное описание шагов, включать дополнительные взаимодействия и проверки, а также учитывать конкретные требования и бизнес-процессы организации.

**1.4 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований.**

Необходимо разработать программу ассортимента игрушек в магазине. Программное приложение позволяет анализировать и управлять каталогом издателей, книг и учётными записями пользователей в процессе работы.

Согласно функциональному анализу процесса в качестве исходных данных выступают:

* список книг в магазине: название книги, автор, год публикации, издатель;
* информация об издательствах: название издательства, год основания, количество книг, рейтинг.

На выходе программное средство должно формировать:

* отчет о наличии книг в магазине: содержит информацию о том, какие издательства есть в магазине, какие книги принадлежат издательству;
* данные об аккаунтах пользователя: содержат информацию о пользователях.

Разрабатываемое программное средство должно обеспечивать выполнение следующих функций:

* учет книг: функция, позволяющая вести учет всех книг в магазине. Эта функция должна включать возможность добавления новых издательств и их книг, редактирования информации о книгах и удаления устаревших записей;
* учет пользователей: функция, позволяющая вести учет всех сотрудников магазина. Эта функция должна включать возможность добавления новых пользователей, редактирования информации о пользователях и удаления устаревших записей;
* поиск издательств и читателей: функция, позволяющая осуществлять поиск книг и читателей по различным параметрам;
* формирование отчетов: функция, позволяющая формировать различные отчеты на основе данных, содержащихся в программном средстве;
* система управления правами доступа: функция, обеспечивающая контроль доступа к программному средству и к данным, содержащимся в нем. Эта функция должна позволять управлять уровнями доступа пользователей и назначать им соответствующие права доступа.

Для разработки программного средства ассортимента учёта издательств для книжного магазина требуется использовать различные технологии и инструменты. В таблице 1.1 приведены примеры некоторых используемых технологий.

Таблица 1.1 – Различные технологии и инструменты, используемые в

программном средстве

|  |  |
| --- | --- |
| Технологии/инструменты | Значимость и польза |
| Языки программирования | Для разработки программного средства можно использовать различные языки программирования, в зависимости от требований к проекту. В этом проекте использован популярный язык программирования – С++. |
| СУБД | Для хранения данных, связанных с учетом книг и читателей, необходимо использовать СУБД (систему управления базами данных). В качестве СУБД в этом проекте выступают файлы. |
| Инструменты документирования | Для документирования программного средства и его функциональности необходимо использовать инструменты, |

Информационная система должна использоваться под управлением одной из семейства операционных систем Microsoft Windows NT, так как они в достаточной степени надежны и безопасны. Для клиентской части это может быть Windows 11. Не рекомендуется использовать операционные системы, разработанные ранее указанных, так как в них сетевая технология не была в полной мере отработана, что может привести к конфликтам внутри программного комплекса. Для печати отчетов требуется наличие принтера.

Примем следующие проектные решения:

* язык программирования – C++;
* СУБД – файлы.

Для хранения данных будут использованы файлы. Хранение данных таким образом позволяют легко получать необходимую информацию и управлять ей при необходимости, также структурированное хранение данных облегчает работу с ними: запись и перезапись данных в файл. Запись в файл является простым способом хранения данных, который не требует использования дополнительных библиотек или инструментов.

В качестве интегрированной среды разработки используем MS Visual Studio 2022. Visual Studio 2022 является одним из наиболее популярных интегрированных сред разработки (IDE) для создания приложений под платформы Windows, Android, iOS, web-приложения и многие другие.

В курсовой работе также необходимо предусмотреть: обработку исключительных ситуаций, возможность возврата назад (навигация), запрос на выполнение необратимых действий. Также для удобного пользования программой должны быть выведены подсказки и сообщения о невозможности совершения действия или о его выполнении.

**1.5 UML-модель представления программного обеспечения и ее описание.**

UML (Unified Modeling Language) – это язык для моделирования программных систем. Он позволяет визуализировать и описать различные аспекты системы, такие как ее структура, поведение и взаимодействия между компонентами. UML используется в разработке программного обеспечения для улучшения понимания системы, коммуникации между участниками проекта и документирования требований и проектных решений.

Диаграмма вариантов использования (Use Case Diagram) – это графическое представление того, как система взаимодействует с актерами (пользователями или другими системами). Она помогает понять функциональность системы и различные сценарии использования. На диаграмме отображаются актеры, варианты использования (действия или сценарии) и связи между ними. Это важный инструмент для анализа требований и проектирования системы.

Диаграмма вариантов использования представлена на рисунке 1.8. В ней участвуют 2 актера: администратор, сотрудник в качестве пользователя.

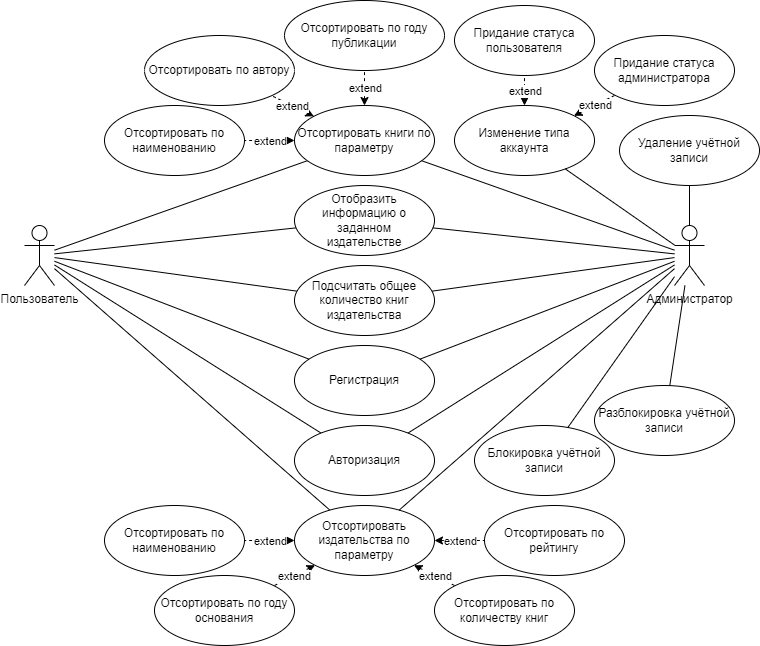
****

Рисунок 1.8 – Диаграмма вариантов использования.

Диаграмма последовательности – это диаграмма UML, которая показывает последовательность сообщений между объектами или компонентами в рамках конкретного сценария использования системы. Она представляет собой набор объектов, которые обмениваются сообщениями в течение определенного времени. Вертикальная ось диаграммы представляет время, горизонтальная ось – объекты или компоненты, между которыми передаются сообщения. Стрелки на диаграмме указывают направление передачи сообщений и порядок выполнения действий. Диаграммы последовательности обычно используются для моделирования поведения системы или процессов в системе, чтобы облегчить понимание и визуализацию сценариев использования.

Изображение выглядит как снимок экрана, диаграмма, текст, дизайн

Автоматически созданное описание

Рисунок 1.9 – Диаграмма последовательности

Диаграмма состояний – это графический инструмент, который иллюстрирует, как объект или система может находиться в различных состояниях и как они могут переходить между этими состояниями, используя вершины, дуги и другие символы для представления элементов автомата. Для более подробного описания элементов модели диаграммы состояний могут быть вложенными. Для правильного понимания смысла диаграммы состояний нужно знать теорию автоматов и уметь применять ее для моделирования поведения сущностей. Диаграмма состояний для объекта проверка заказа приведена на рисунке 1.10. Она помогает визуализировать и анализировать поведение системы в различных состояниях и условиях.

В начале процесса "Передача заказа на проверку" объект находится в начальном состоянии. Затем, он переходит в состояние "ожидание подтверждения получения заказа", где он ждет, пока клиент подтвердит свой заказ. Если заказ подтвержден, объект переходит в состояние "ожидание проверки заказа", где он начинает обрабатывать заказ, проверяя его наличие и готовность к доставке. Если же заказ не может быть выполнен, объект переходит в состояние "отклонение заказа". После подтверждения заказа процесс завершается.

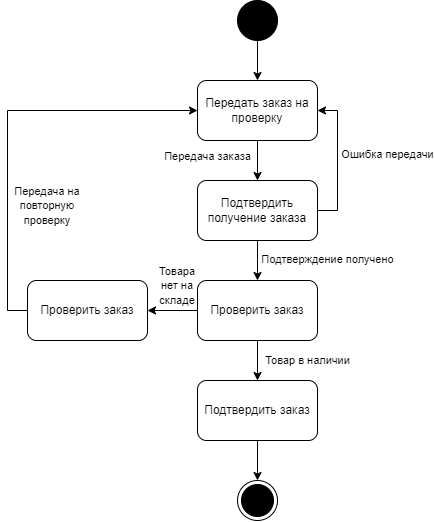


Рисунок 1.10 – Диаграмма состояний.

Диаграмма развертывания в UML используется для отображения элементов и компонентов системы, которые существуют на этапе ее исполнения. Она включает несколько типов фигур, которые помогают представить структуру и взаимосвязи между элементами системы.

На диаграмме развертывания используются трехмерные блоки или узлы, которые представляют базовые программные или аппаратные элементы системы. Линии, соединяющие узлы, указывают на связи между ними, а более мелкие фигуры, расположенные внутри блоков, представляют программные артефакты, развертываемые на этих узлах. Диаграмма развертывания системы приведена на рисунке 1.11.

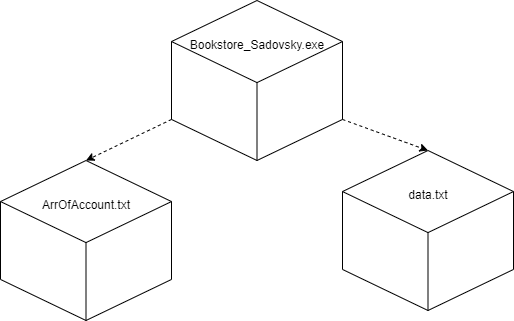


Рисунок 1.11 – Диаграмма развёртывания.

В данной диаграмме представлены файлы, которые в свою очередь образуют программу и зависят друг от друга:

* Bookstore\_Sadovsky.exe – сама программа;
* ArrOfAccount.txt – файл, в котором хранится информация с данными всех пользователей;
* data.txt – файл, в котором хранится информация с данными об издательствах и книгах.

Диаграмма компонентов – это диаграмма UML, которая показывает компоненты программной системы и их взаимосвязи. Она помогает визуализировать структуру системы, показывает, какие компоненты существуют и как они взаимодействуют друг с другом.

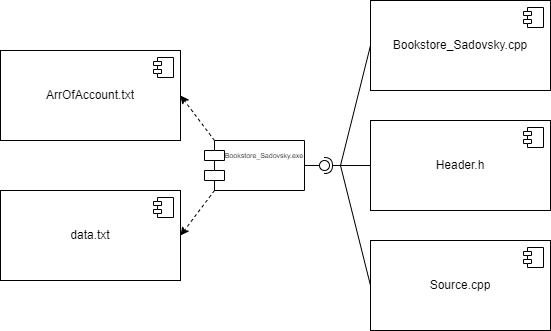


Рисунок 1.12 – Диаграмма компонентов.

Таким образом, UML-диаграммы играют важную роль в определении функциональных требований к системе и управлении всем процессом разработки. Они охватывают все этапы деятельности, такие как анализ, проектирование и тестирование. Варианты использования помогают понять, как желаемый результат для пользователя влияет на архитектуру системы и поведение ее компонентов, необходимых для реализации требуемой функциональности. Варианты использования позволяют определить, как система будет взаимодействовать с пользователем и другими системами, а также какие функции и возможности она должна предоставлять. Они являются важным инструментом для обеспечения соответствия разрабатываемой системы требованиям заказчика и пользователей, а также для улучшения ее качества и удобства использования. Кроме того, варианты использования помогают обнаруживать потенциальные проблемы и улучшать производительность системы. При изменении вариантов использования необходимо убедиться, что эти изменения не приведут к утере смысла, и система все еще будет соответствовать требованиям пользователей и заказчика.

**2 Проектирование и конструирование программного средства**

**2.1 Постановка задачи**

Целью данной курсовой работы является разработка программного средства для учета издательств в книжном магазине. О каждом издательстве будет известно следующее:

* наименование издательства;
* год основания;
* количество книг;
* рейтинг.

Также у каждого издательства будут опубликованные книги, о которых будет известно:

* название книги;
* автор;
* год публикации;
* издательство.

С данной информационной системой могут взаимодействовать следующие группы пользователей:

* пользователи;
* администраторы.

При работе с программой администратор должен иметь возможность решать следующие задачи:

* блокировать пользователя;
* разблокировать пользователя;
* изменять тип аккаунта;
* удалять любого пользователя.

Примем следующие проектные решения:

* платформа реализации приложения – Visual Studio 2022;
* язык программирования – C++;
* СУБД – файлы.

**2.2 Разработка модульной структуры**

Модульная структура помогает организовать систему на небольшие логические блоки, называемые модулями. Каждый модуль выполняет свои функции и может быть использован повторно в других проектах. Это делает систему гибкой, упрощает ее сопровождение и облегчает работу команды разработчиков. Кроме того, модульная структура повышает надежность системы и уменьшает влияние ошибок на другие модули.

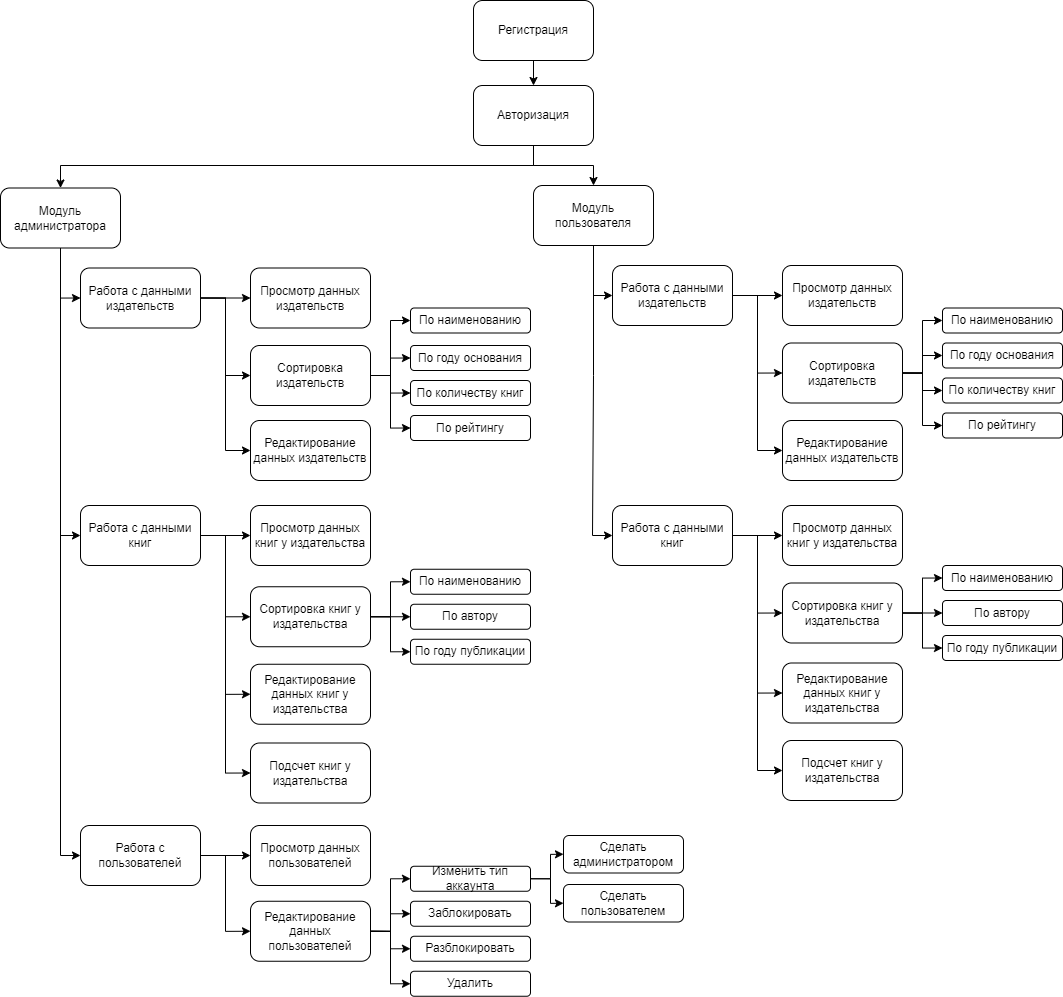


Рисунок 2.1 – Модульная структура программы

**2.3 Выбор способа организации данных**

В качестве способа организации данных в приложении были выбраны структуры. Выбор обусловлен несложной программной функциональностью. Структуры в C++ позволяют определить пользовательские типы данных, объединяющие несколько переменных различных типов в одну единую структуру. Описание используемых структур представлено ниже.

Структура “Account” для хранения учётных записей пользователей:

Таблица 2.1 – Описание структуры Account

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Назначение поля |
| username | string | Логин пользователя |
| password | string | Пароль пользователя |
| type | int | Должность пользователя |
| access | int | Статус доступа к аккаунту |

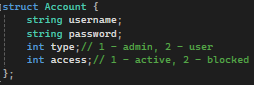


Рисунок 2.2 – Структура Account

Структура “Publisher” для хранения данных издательств:

Таблица 2.2 – Описание структуры Publisher

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Назначение поля |
| name | string | Название издательства |
| yearFounded | int | Год основания |
| booksPublished | int | Количество книг |
| rating | float | Рейтинг издательства |

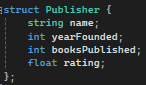


Рисунок 2.3 – Структура Publisher

Структура "Publisher" содержит несколько переменных различных типов данных, таких как строки для названия издательств, целое число для года основания и количества книг конкретного издательства. Тип данных float помогает уточнить рейтинг издательства в книжном магазине.

Структура “ Book” для хранения информации о книгах в магазине:

Таблица 2.3 – Описание структуры Book

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Назначение поля |
| title | string | Название книги |
| author | string | Автор |
| yearPublished | int | Год написания книги |
| publisherName | string | Издательство, которое опубликовало книгу |

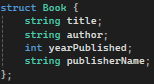


Рисунок 2.4 – Структура Book

Структура "Book" содержит информацию о названии, авторе, году написания и издательстве. У каждого издательства свой связанный список книг, количество которых автоматически подсчитывается в поле “booksPublished”.

**2.4 Разработка перечня пользовательских функций**

Функции, которые были использованы для разработки программы учета издательств в книжном магазине представлена ниже в таблице 2.4.

Таблица 2.4 – Перечень пользовательских функций программы

|  |  |
| --- | --- |
| Название функции | Назначение функции |
| void readFileAcc(Account\* arr\_of\_acc, int& number\_of\_acc) | Функция чтения пользователей из файла |
| void registerAcc(Account\* arr\_of\_acc, int& number\_of\_acc) | Функция регистрации пользователя |
| int logIn(Account\* arr\_of\_acc, int& number\_of\_acc) | Функция входа в аккаунт |
| void writeFileAcc(Account\* arr\_of\_acc, int number\_of\_acc) | Функция записи пользователей в файл |

Продолжение таблицы 2.4

|  |  |
| --- | --- |
| Название функции | Назначение функции |
| void delAcc(Account\* arr\_of\_acc, int& number\_of\_acc) | Функция удаления пользователя |
| void blockAcc(Account\* arr\_of\_acc, int& number\_of\_acc) | Функция блокировки пользователя |
| void unblockAcc(Account\* arr\_of\_acc, int& number\_of\_acc) | Функция разблокировки пользователя |
| void changeAccType(Account\* arr\_of\_acc, int& number\_of\_acc) | Функция изменения типа аккаунта |
| void showStartAccArr(Account\* arr\_of\_acc, int number\_of\_acc) | Функция меню управления администратора |
| void showAccArr(Account\* arr\_of\_acc, int number\_of\_acc) | Функция вывода всех пользователей |
| void addPublisher(Publisher publishers[], int& publisherCount) | Функция добавления издательства |
| void addBook(Book books[], int& bookCount, Publisher publishers[], int publisherCount, const string& publisherName) | Функция добавления книги |
| string formatBookList(const Book books[], int bookCount, const string& publisherName) | Функция объединения списка книг в строку |
| void printPublishersTable(const Publisher publishers[], int publisherCount, const Book books[], int bookCount) | Функция вывода таблицы издательств |
| void printBooksTable(const Book books[], int bookCount, const string& publisherName) | Функция вывода таблицы книг у издательства |
| void loadFromFile(Publisher publishers[], int& publisherCount, Book books[], int& bookCount, const string& filename) | Функция считывания издательств и связанных книг с файла |
| void saveToFile(const Publisher publishers[], int publisherCount, const Book books[], int bookCount, const string& filename) | Функция записи издательств и связанных книг в файл |
| void deleteBook(Book books[], int& bookCount, Publisher publishers[], int publisherCount, int bookOnePublisherIndex, string publisherName) | Функция удаления книги |
| void deletePublisher (Publisher publishers[], int& publisherCount, Book books[], int& bookCount, int index) | Функция удаления издательства |
| void sortBooksByTitleForPublisher(Book books[], int bookCount, const string& publisherName) | Функция сортировки книг издательства по названию |

Продолжение таблицы 2.4

|  |  |
| --- | --- |
| Название функции | Назначение функции |
| void sortBooksByAuthorForPublisher (Book books[], int bookCount, const string& publisherName) | Функция сортировки книг издательства по автору |
| void sortBooksByYearForPublisher(Book books[], int bookCount, const string& publisherName) | Функция сортировки книг издательства по году написания |
| void sortPublishersByName (Publisher publishers[], int publisherCount) | Функция сортировки издательств по названию |
| void sortPublishersByYear (Publisher publishers[], int publisherCount) | Функция сортировки издательств по году основания |
| void sortPublishersByBookCount (Publisher publishers[], int publisherCount, Book books[], int bookCount) | Функция сортировки издательств по количеству книг |
| void sortPublishersByRating (Publisher publishers[], int publisherCount) | Функция сортировки издательств по рейтингу |
| void editPublisher(Publisher publishers[], int& publisherCount, Book books[], int& bookCount, int index) | Функция вывода меню редактирования издательства и связанных книг |
| void main () | Функция мейн |

**2.5 Разработка схем алгоритмов работы программы**

Блок-схема – это графическое представление последовательности действий или процесса с использованием фигур и стрелок. Она помогает визуализировать и понять логику выполнения действий и алгоритмов. Блок-схемы используют прямоугольники для действий, ромбы для условий и стрелки для указания направления потока выполнения. Они широко используются в программировании, системном анализе и управлении проектами для анализа и оптимизации процессов.

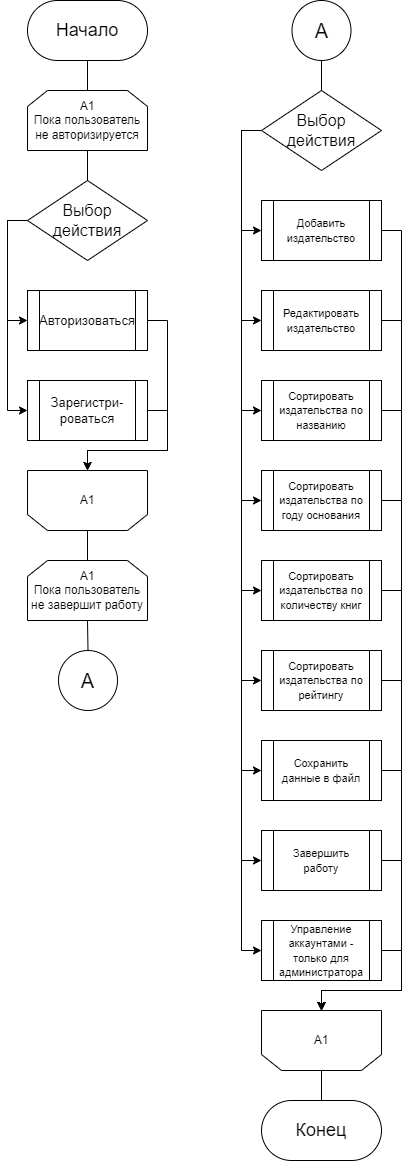


Рисунок 2.5 – Схема алгоритма работы программы

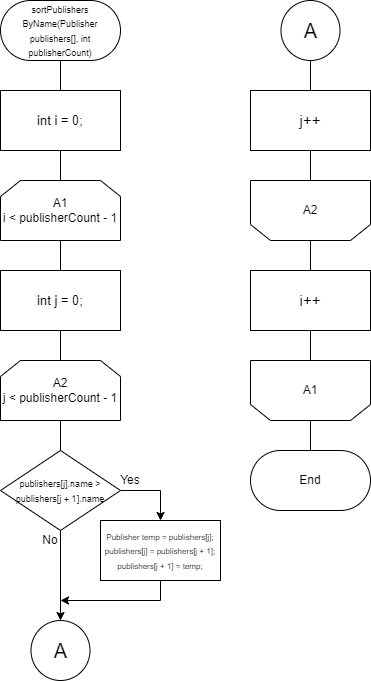


Рисунок 2.6 – Схема алгоритма sortPublishersByName

Алгоритм “formatBookList” позволяет объединить список книг в фиксированную строку для отображения в таблице с издательствами первых книг вида “Книга1, Книга2, Кни…”. Причем в алгоритме происходят несколько проверок, которые позволяют изменять строку в зависимости от количества книг и длин их названий. Например, если информация о книгах издательства отсутствует, то в таблице будет отображаться строка “Нет книг”. Также при совпадении полученной длины полных названий книг вида “Книга\_\_\_\_\_1, Книга\_\_\_\_\_2” с допустимой длиной символов в ячейке таблицы происходит дополнительная проверка на наличие книг у издательства, при наличии которых итоговая строка будет вида “Книга\_\_\_\_\_1, Книга\_\_\_...” означающая, что это не единственные книги у данного издательства. Блок-схема алгоритма “formatBookList” представлена на рисунке 2.7.

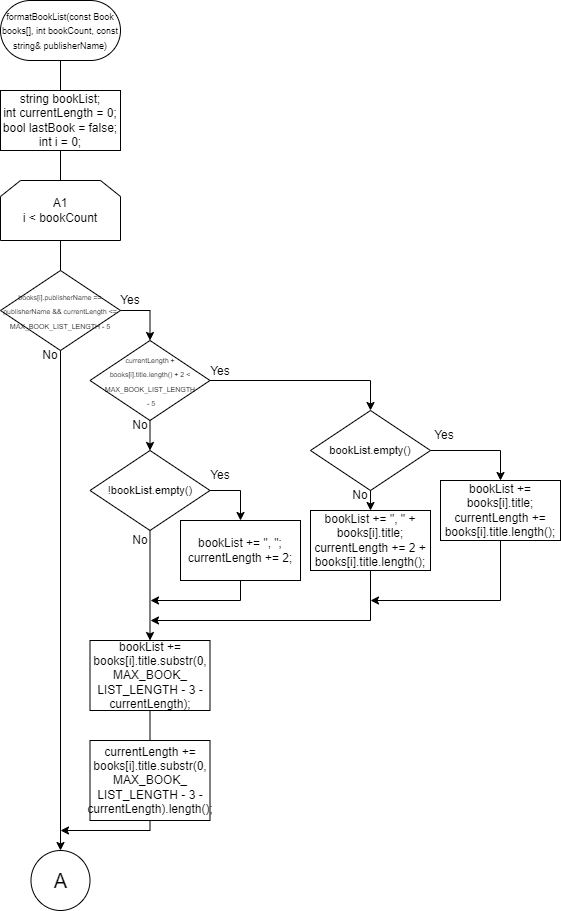
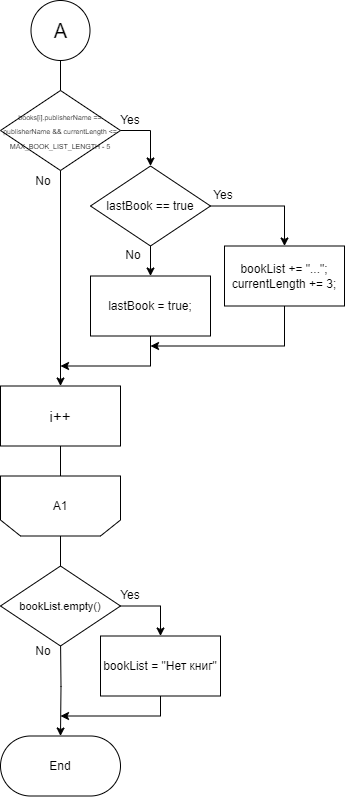


Рисунок 2.7 – Схема алгоритма formatBookList

**3 Тестирование и проверка работоспособности программного средства**

Тестирование программного средства – это метод проверки соответствия фактического программного продукта ожидаемым требованиям, который также необходим, чтобы убедиться, что продукт не содержит дефектов.

Результаты тестирования показали, что программа учета издательств для книжного магазина соответствует функциональным требованиям и все функции работают корректно.

Программа предупреждает пользователя об исключительных ситуациях, что помогает ему быстрее понять, в чем заключается проблема при работе с ней. В программе реализованы проверки на корректный ввод целочисленных значений и недопустимых символов. При некорректном вводе программа очищает строку ввода и запрашивает ввод снова. Повторный ввод повторяется до тех пор, пока пользователь не введет допустимое значение.

После запуска работы программы пользователю предлагают 2 действия: авторизация и регистрация (рисунок 3.1).

Изображение выглядит как Шрифт, текст, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 3.1- Начальное меню

При вводе несуществующего логина программа выдаст сообщение об ошибке (рисунок 3.2).

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 3.2 – Ввод несуществующего логина

При вводе некорректного значения в меню программа выдаст сообщение об ошибке (рисунок 3.3).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.3 – Ввод некорректного значения

При вводе существующего логина в регистрации программа выдаст сообщение об ошибке (рисунок 3.4).

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 3.4 – Ввод существующего логина в меню регистрации

При вводе некорректных значений данных издательств программа выдаст сообщение об ошибке (рисунки 3.5).

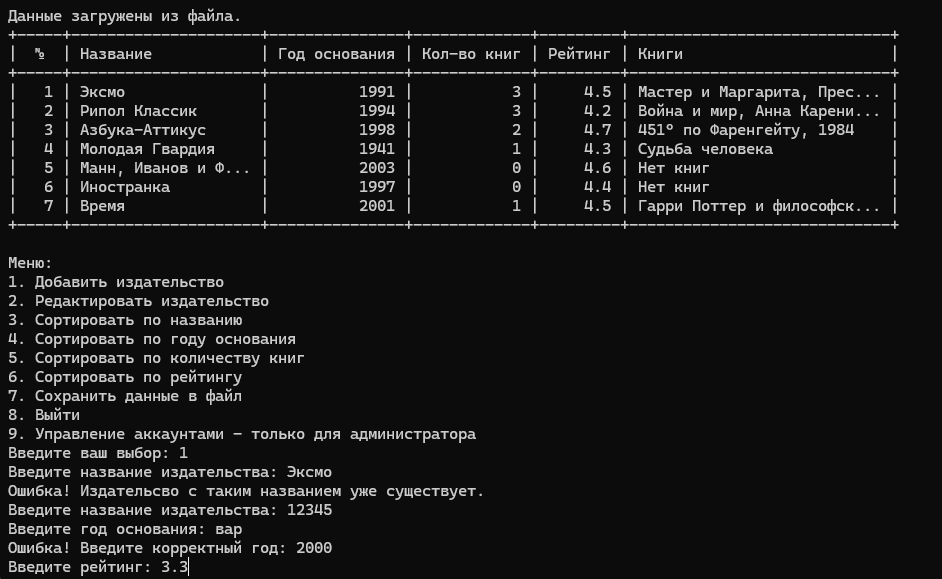


Рисунок 3.5 – Ввод некорректных значений

**4 Инструкция по развёртыванию приложения и сквозной пример**

**4.1 Авторизация**

При запуске программного средства пользователю предоставляется выбор авторизации (рисунок 4.1).

Изображение выглядит как Шрифт, текст, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 4.1 – Меню авторизации

При первом запуске приложения необходимо зарегистрироваться (рисунок 4.2).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 4.2 – Меню регистрации

После авторизации отображаются таблица с данными издательств и список вариантов использования (рисунок 4.3).

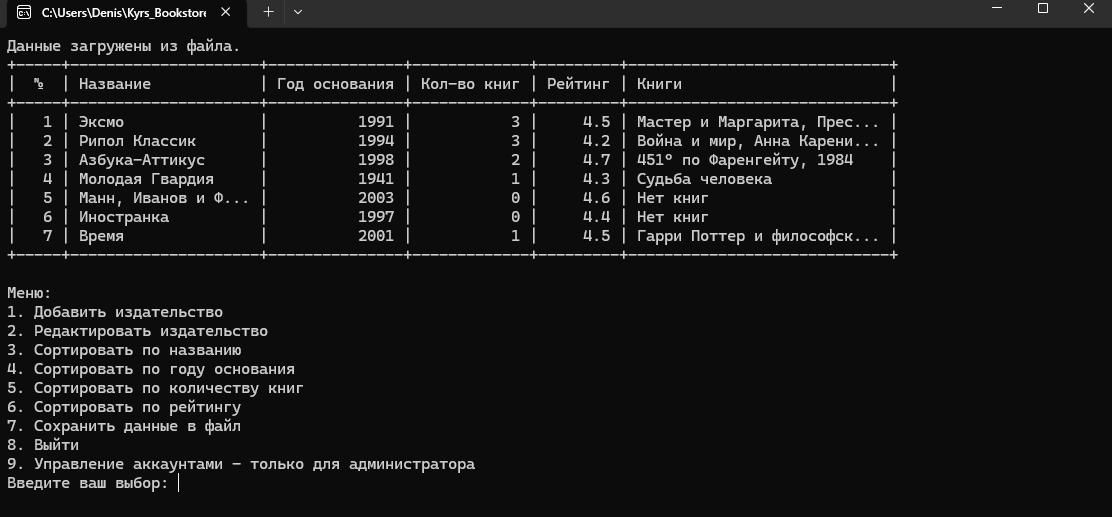


Рисунок 4.3 – Меню пользователя

При выборе редактировании издательства выводится меню редактирования (рисунок 4.4).

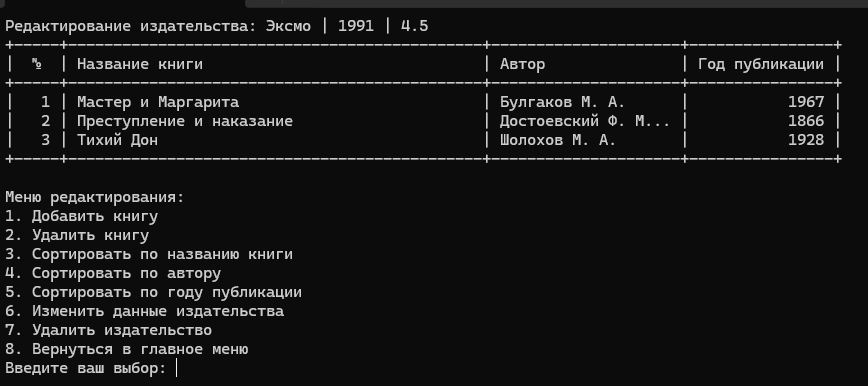


Рисунок 4.4 – Меню пользователя

**4.2 Модуль администратора**

После успешного входа под именем администратора пользователю становится доступно меню управления аккаунтами, в котором отображается список всех аккаунтов и их статус (рисунок 4.4).

**Изображение выглядит как текст, снимок экрана

Автоматически созданное описание**

Рисунок 4.4 – Меню администратора

При выборе изменения статуса аккаунта(пользователь-администратор), блокировки или разблокировки пользователя, администратору необходимо ввести порядковый номер пользователя (рисунок 4.6).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 4.5 – Блокировка пользователя

При выборе назад пользователя вернет в первоначальное меню.

**4.3 Модуль пользователя**

После авторизации отображаются таблица с данными издательств и список вариантов использования (рисунок 4.3).

При выборе сортировать по рейтингу, произойдет вывод измененной таблицы с сообщением об успешной сортировке (рисунок 4.6). Также данные можно сохранить в отсортированном виде.

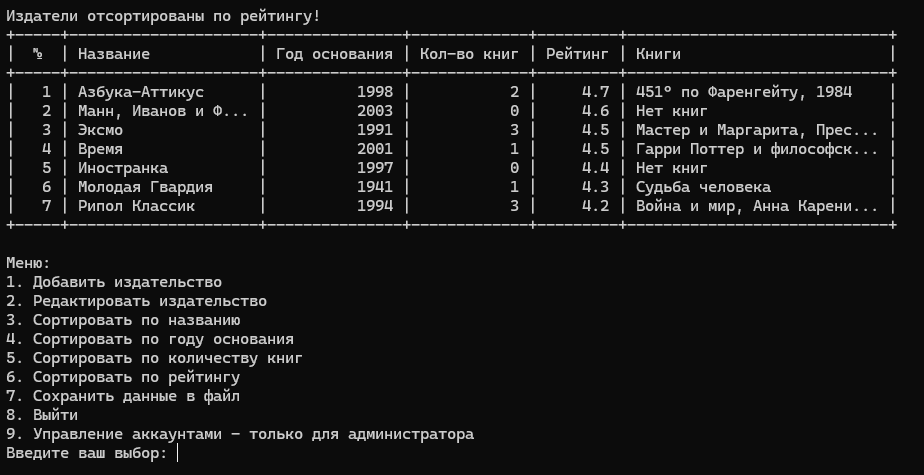


Рисунок 4.6 – Сортировка издательств по рейтингу

Кроме того, возможна сортировка книг отдельных издательств в меню редактирования издательства. Сортировка по году написания представлена на рисунке 4.7.

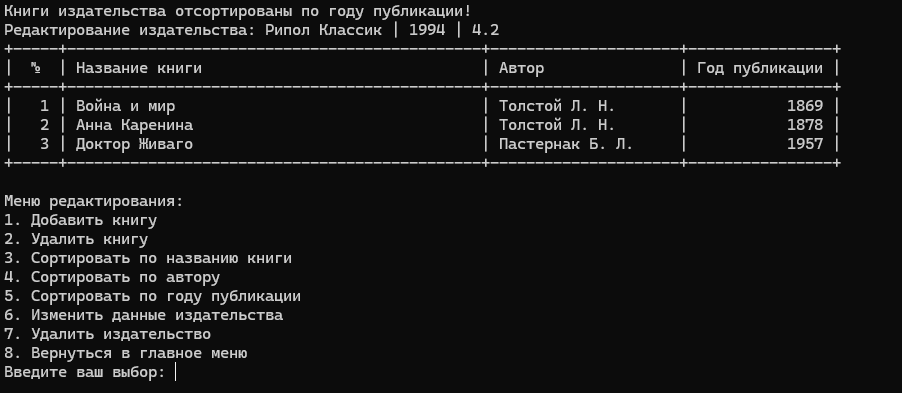


Рисунок 4.7 – Сортировка книг по году публикации

**Заключение**

Разработка программы учёта издательств для книжного магазина имеет важное значение для оптимизации процессов управления ассортиментом и повышения эффективности работы магазина. Программное решение позволяет систематизировать данные об издательствах, упрощает процесс учёта и анализа информации, что способствует более эффективному взаимодействию с поставщиками и управлению книжным ассортиментом. В разработанной программе предусмотрены функциональные возможности, позволяющие регистрировать, редактировать и отслеживать информацию об издательствах, а также анализировать их деятельность. Графический интерфейс программы обеспечивает удобство взаимодействия пользователей с системой, что минимизирует временные затраты на выполнение рутинных операций. Для понимания логики работы программного средства были разработаны модели бизнес-процессов в нотациях IDEF0 и BPMN, которые визуализируют основные этапы учёта издательств и взаимодействия с ними. Также были созданы проектные решения на основе языка UML, что способствует более глубокому пониманию структуры программы и оптимизации процессов её использования.

Таким образом, разработанное программное средство предоставляет инструменты для эффективного учёта издательств, упрощает процесс управления данными и способствует повышению общей эффективности работы книжного магазина. Внедрение данной программы позволит сократить временные затраты на обработку информации, улучшить качество обслуживания клиентов и повысить конкурентоспособность магазина на рынке.

**Список используемой литературы**

[1] Зачем нужна uml [Электронный ресурс]. – https://habr.com/ru/articles/458680/

[2] Как начать моделировать бизнес-процессы в BPMN [Электронный ресурс]. – Электронные данные. – Режим доступа: [Как начать моделировать бизнес-процессы в BPMN / Хабр (habr.com)](https://habr.com/ru/articles/697326/)

[3] Основы формальных методов описания бизнес-процессов: учеб. Пособие / К. Е. Самуйлов, К. Е. Серебренникова [и др.]. – М.: РУДН, 2008. – 130 с.: ил.

[4] Диаграмма состояний [Электронный ресурс]. – Режим доступа: <https://anylogic.help/ru/anylogic/statecharts/statecharts.html>.

[5] ТРПО – Курсовое проектирование (работа) - Пособие [Электронный ресурс]. – Режим доступа: <https://cloud.mail.ru/public/oaoe/asWuomL8y>.

[6] Документация по Visual Studio [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/visualstudio/ide/?view=vs-2022>.

[7] Основы алгоритмизации и программирования: конспект лекций для студентов всех специальностей и форм обучения БГУИР / В. Л. Бусько, А. Г. Корбит, Т. М. Кривоносова. - Минск: БГУИР, 2004. – 103 с.

[8] «СТАНДАРТ ПРЕДПРИЯТИЯ ДИПЛОМНЫЕ ПРОЕКТЫ (РАБОТЫ)ОБЩИЕ ТРЕБОВАНИЯ СТП 01–2017» [Электронный ресурс]. —

Режим доступа: https://www.bsuir.by/m/12\_100229\_1\_122976.pdf .

[9] Простое руководство к диаграммам развёртывания UML [Электронный ресурс]. – Режим доступа: [https://creately.com/blog/ru/uncategorized-ru/учебное-пособие-по-диаграмме-развёрт/](https://creately.com/blog/ru/uncategorized-ru/%D1%83%D1%87%D0%B5%D0%B1%D0%BD%D0%BE%D0%B5-%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%B8%D0%B5-%D0%BF%D0%BE-%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B5-%D1%80%D0%B0%D0%B7%D0%B2%D1%91%D1%80%D1%82/) .

[10] Полное руководство по диаграммам компонентов [Электронный ресурс]. – Режим доступа: https://creately.com/blog/ru/uncategorized-ru/учебное-пособие-по-компонентной-диаг/ .

**Приложение А   
Отчет о проверке на заимствование в системе «Антиплагиат»**

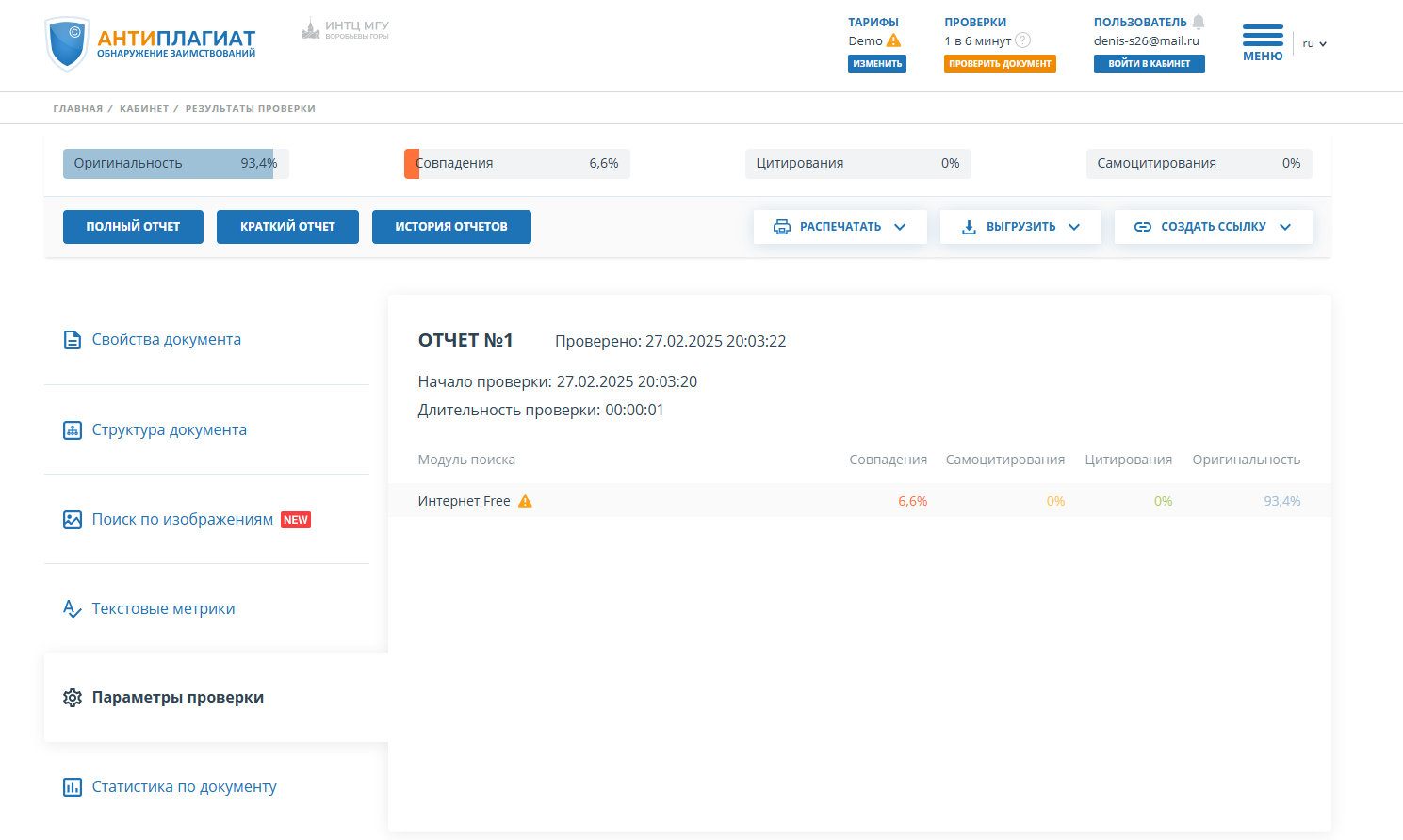


Рисунок А.1 – Проверка на заимствование

**Приложение Б   
Листинг кода алгоритмов, реализующих основную бизнес-логику**

Листинг кода файла Bookstore\_Sadovsky.cpp:

#include "Header.h"

int number\_of\_acc = 0;

int current\_acc = -1;

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Account arr\_of\_acc[MAX\_NUMBER\_OF\_ARR];

Publisher publishers[MAX\_PUBLISHERS];

int publisherCount = 0;

Book books[MAX\_BOOKS];

int bookCount = 0;

int ch = 0;

int choice = 0;

readFileAcc(arr\_of\_acc, number\_of\_acc);

while (ch == 0) {

if (number\_of\_acc == 0) {

registerAcc(arr\_of\_acc, number\_of\_acc);

}

cout << "\nАвторизоваться(1)\nЗарегистрироватьcя(2)\n";

if (scanf\_s("%d", &choice) == 1 && choice < 3 && choice > 0) {

switch (choice) {

case 1:

current\_acc = logIn(arr\_of\_acc, number\_of\_acc);

if (current\_acc > -1) {

ch = 1;

}

break;

case 2:

registerAcc(arr\_of\_acc, number\_of\_acc);

break;

}

}

else {

scanf\_s("%\*[^\n]");

printf(" Ошибка.\n");

}

}

system("cls");

string filename = "data.txt";

loadFromFile(publishers, publisherCount, books, bookCount, filename);

choice = 0;

do {

printPublishersTable(publishers, publisherCount, books, bookCount);

cout << "\nМеню:\n"

<< "1. Добавить издательство\n"

<< "2. Редактировать издательство\n"

<< "3. Сортировать по названию\n"

Продолжение приложения Б

<< "4. Сортировать по году основания\n"

<< "5. Сортировать по количеству книг\n"

<< "6. Сортировать по рейтингу\n"

<< "7. Сохранить данные в файл\n"

<< "8. Выйти\n"

<< "9. Управление аккаунтами - только для администратора\n"

<< "Введите ваш выбор: ";

while (scanf\_s("%d", &choice) != 1 || choice < 1 || choice > 9) {

cout << "Ошибка! Введите число от 1 до 8: ";

scanf\_s("%\*[^\n]");

}

switch (choice) {

case 1:

addPublisher(publishers, publisherCount);

break;

case 2:

int index;

cout << "Введите номер издательства для редактирования: ";

while (scanf\_s("%d", &index) != 1 || index < 1 || index > publisherCount) {

cout << "Ошибка! Введите корректный номер: ";

scanf\_s("%\*[^\n]");

}

system("cls");

editPublisher(publishers, publisherCount, books, bookCount, index - 1);

break;

case 3:

sortPublishersByName(publishers, publisherCount);

system("cls");

cout << "Издатели отсортированы по названию!" << endl;

break;

case 4:

sortPublishersByYear(publishers, publisherCount);

system("cls");

cout << "Издатели отсортированы по году основания!" << endl;

break;

case 5:

sortPublishersByBookCount(publishers, publisherCount, books, bookCount);

system("cls");

cout << "Издатели отсортированы по количеству книг!" << endl;

break;

case 6:

sortPublishersByRating(publishers, publisherCount);

system("cls");

cout << "Издатели отсортированы по рейтингу!" << endl;

break;

case 7:

system("cls");

saveToFile(publishers, publisherCount, books, bookCount, filename);

break;

case 8:

cout << "Выход из программы." << endl;

break;

case 9:

system("cls");

if (arr\_of\_acc[current\_acc].type == 1) {

showStartAccArr(arr\_of\_acc, number\_of\_acc);

}

else {

cout << "\nДанный вариант доступен только администратору.\n";

}

break;

Продолжение приложения Б

}

} while (choice != 8);

}

Листинг кода функции добавления издательства:

void addPublisher(Publisher publishers[], int& publisherCount) {

if (publisherCount >= MAX\_PUBLISHERS) {

cout << "Достигнуто максимальное количество издательств!" << endl;

}

else {

Publisher newPublisher;

bool isUnique;

do {

cout << "Введите название издательства: ";

cin.ignore();

getline(cin, newPublisher.name);

// Проверка на уникальность имени издательства

isUnique = true;

for (int i = 0; i < publisherCount; i++) {

if (publishers[i].name == newPublisher.name) {

cout << "Ошибка! Издательсво с таким названием уже существует.\n";

isUnique = false;

}

}

} while (!isUnique);

cout << "Введите год основания: ";

while (scanf\_s("%d", &newPublisher.yearFounded) != 1 || newPublisher.yearFounded < 0) {

cout << "Ошибка! Введите корректный год: ";

scanf\_s("%\*[^\n]");

}

newPublisher.booksPublished = 0;

cout << "Введите рейтинг: ";

while (scanf\_s("%f", &newPublisher.rating) != 1 || newPublisher.rating < 0 || newPublisher.rating > 10) {

cout << "Ошибка! Введите рейтинг от 0 до 10: ";

scanf\_s("%\*[^\n]");

}

publishers[publisherCount] = newPublisher;

publisherCount++;

system("cls");

cout << "Издательство добавлено!" << endl;

}

}

Листинг кода функции добавления книги:

void addBook(Book books[], int& bookCount, Publisher publishers[], int publisherCount, const string& publisherName) {

if (bookCount >= MAX\_BOOKS) {

cout << "Достигнуто максимальное количество книг!" << endl;

Продолжение приложения Б

}

else {

Book newBook;

cout << "Введите название книги: ";

cin.ignore();

getline(cin, newBook.title);

cout << "Введите автора книги: ";

getline(cin, newBook.author);

cout << "Введите год публикации: ";

while (scanf\_s("%d", &newBook.yearPublished) != 1 || newBook.yearPublished < 0) {

cout << "Ошибка! Введите корректный год: ";

scanf\_s("%\*[^\n]");

}

newBook.publisherName = publisherName;

// Поиск издательства и обновление количества книг

for (int i = 0; i < publisherCount; i++) {

if (publishers[i].name == publisherName) {

publishers[i].booksPublished++;

}

}

books[bookCount] = newBook;

bookCount++;

system("cls");

cout << "Книга добавлена!" << endl;

}

}

Листинг кода функции объединения списка книг:

string formatBookList(const Book books[], int bookCount, const string& publisherName) {

string bookList;

int currentLength = 0;

bool lastBook = false;

string prevName;

for (int i = 0; i < bookCount; i++) {

if (books[i].publisherName == publisherName && currentLength <= MAX\_BOOK\_LIST\_LENGTH - 5) {

if (currentLength + books[i].title.length() + 2 < MAX\_BOOK\_LIST\_LENGTH - 5) {

if (bookList.empty()) {

bookList += books[i].title;

currentLength += books[i].title.length();

}

else {

bookList += ", " + books[i].title;

currentLength += 2 + books[i].title.length();

}

}

else {

if (!bookList.empty()) {

bookList += ", ";

currentLength += 2;

}

Продолжение приложения Б

bookList += books[i].title.substr(0, MAX\_BOOK\_LIST\_LENGTH - 3 - currentLength);

prevName = books[i].title.substr(0, MAX\_BOOK\_LIST\_LENGTH - 3 - currentLength);

currentLength += books[i].title.substr(0, MAX\_BOOK\_LIST\_LENGTH - 3 - currentLength).length();

}

}

if (books[i].publisherName == publisherName && currentLength == MAX\_BOOK\_LIST\_LENGTH - 3 && books[i].title.length() > 1) {

if (lastBook || books[i].title != prevName) {

bookList += "...";

currentLength += 3;

}

else {

lastBook = true;

}

}

}

if (bookList.empty()) {

bookList = "Нет книг";

}

return bookList;

}