ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc199366867)

[ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc199366868)

[1.1. Понятие распределения продукции в типографиях 5](#_Toc199366869)

[1.2 Процесс отслеживания печатной продукции 6](#_Toc199366870)

[1.3 Причины необходимости автоматизации процессов распределения 9](#_Toc199366871)

[1.4 Особенности построения систем отслеживания распределения в типографиях 11](#_Toc199366872)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММНОЙ СИСТЕМЫ 15](#_Toc199366873)

[2.1 Требования к программе 15](#_Toc199366874)

[2.2 Руководство пользователя 16](#_Toc199366875)

[2.3 Руководство программиста 23](#_Toc199366876)

[ЗАКЛЮЧЕНИЕ 28](#_Toc199366877)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 29](#_Toc199366878)

[ПРИЛОЖЕНИЕ А 30](#_Toc199366879)

# ВВЕДЕНИЕ

В условиях стремительного развития информационных технологий автоматизация бизнес-процессов становится неотъемлемой частью эффективного управления любым предприятием. Типографии, занимающиеся выпуском печатной продукции, не являются исключением. Одной из важнейших задач в их деятельности является организация учёта и отслеживания распределения выпускаемой продукции между различными точками назначения — складами, почтовыми отделениями и конечными получателями.

Процесс распределения продукции в типографиях предполагает наличие больших объёмов данных, связанных с тиражами, наименованиями газет или журналов, сроками выпуска и маршрутами доставки. Отслеживание этих данных вручную или с использованием устаревших методов (например, таблиц Excel) приводит к ошибкам, задержкам и потере информации, что негативно сказывается на имидже организации и приводит к финансовым потерям. Для повышения эффективности работы типографий необходимо внедрение специализированных программных систем, способных автоматизировать процессы учёта, обработки и отслеживания информации о продукции.

Актуальность данной темы обусловлена необходимостью повышения прозрачности логистических процессов, минимизации человеческого фактора и оптимизации распределения ресурсов. Разработка программной системы для отслеживания распределения печатной продукции позволит типографиям получить точные данные о текущем состоянии заказов, своевременно вносить корректировки в маршруты доставки и быстрее реагировать на возникающие изменения.

Целью данной работы является разработка программной системы, предназначенной для учёта печатной продукции и отслеживания её распределения по типографиям и почтовым отделениям. Система должна быть удобной в использовании, обладать необходимым набором функций для управления данными и обеспечивать надёжное хранение информации.

Для достижения поставленной цели в работе будут решены следующие задачи:

провести анализ предметной области, определить ключевые процессы и выявить основные проблемы в существующих методах учёта, сформулировать функциональные и нефункциональные требования к разрабатываемой системе.

Разработать структуру данных и архитектуру программного решения, реализовать консольное приложение для управления процессами учёта и отслеживания продукции, провести тестирование разработанной системы и оценить её эффективность.

В первой главе работы будет рассмотрена предметная область, связанная с процессами распределения продукции в типографиях, проанализированы существующие подходы и выявлены основные недостатки. Это позволит чётко сформулировать требования к создаваемой системе и обосновать необходимость её разработки.

Вторая глава будет посвящена проектированию и реализации программной системы. Здесь будут детально рассмотрены функциональные требования, структура программы, используемые технологии и особенности реализации ключевых компонентов. Также будут приведены пояснения к основным частям кода, что позволит лучше понять внутреннюю логику работы системы.

В заключении будут подведены итоги проведённой работы, сделаны выводы о достигнутых результатах и предложены возможные направления для дальнейшего развития проекта.

Таким образом, разработка программной системы для отслеживания распределения печатной продукции является актуальной задачей, направленной на повышение эффективности работы типографий, улучшение качества обслуживания клиентов и оптимизацию внутренних процессов предприятия.

# ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1. Понятие распределения продукции в типографиях

Типографии играют ключевую роль в создании и распространении печатной продукции, обеспечивая не только её производство, но и своевременную доставку заказчикам, складам, почтовым отделениям и другим получателям. Распределение продукции — это сложный процесс, который начинается сразу после завершения печати и охватывает множество этапов: от учёта и формирования партий до планирования маршрутов, оформления документов и отслеживания поставок. Эффективная организация этого процесса позволяет сократить сроки доставки, снизить логистические затраты и повысить удовлетворённость клиентов. Ошибки же, такие как задержки, путаница в тиражах или лишние расходы, могут серьёзно навредить репутации типографии.

Учёт — основа успешного распределения. Без точной фиксации данных о наименовании продукции, количестве экземпляров, дате печати, пунктах назначения и ответственных лицах невозможно обеспечить порядок в поставках. При больших тиражах и множестве направлений ручной учёт становится неэффективным: ошибки в документах, потеря информации или человеческий фактор могут привести к значительным убыткам. Поэтому автоматизация учёта и отслеживания приоритет для современных типографий.

Печатная продукция имеет свои особенности, усложняющие распределение. Например, периодические издания, такие как газеты или журналы, требуют строгого соблюдения сроков доставки, иначе они теряют актуальность. Большие тиражи и разнообразие форматов (по размеру, весу, объёму) усложняют упаковку и транспортировку, а необходимость точной сортировки исключает возможность путаницы в заказах. Всё это требует чёткой координации, быстрой обработки данных и оперативного принятия решений.

Многие типографии до сих пор полагаются на устаревшие методы: ручное заполнение накладных, использование разрозненных электронных таблиц или отсутствие единой базы данных для отслеживания отгрузок. Такие подходы замедляют работу, увеличивают риск ошибок и мешают быстро реагировать на проблемы. Например, без актуальной информации о статусе поставок сложно оперативно устранить сбои в логистике.

Автоматизация процессов открывает новые перспективы. Специализированные информационные системы позволяют вести централизованный учёт, автоматически генерировать документы, контролировать перемещение партий и формировать отчёты в реальном времени. Такие решения повышают прозрачность работы и минимизируют ошибки. Создание собственной программной системы, адаптированной под конкретную типографию, помогает учесть её уникальные логистические процессы и требования к отчётности.

В итоге, понимание особенностей распределения и проблем традиционных подходов создаёт основу для разработки эффективных программных решений, которые могут значительно улучшить работу типографий.

## 1.2 Процесс отслеживания печатной продукции

Процесс отслеживания печатной продукции в типографиях представляет собой последовательность действий, направленных на контроль перемещения тиражей от момента завершения печати до доставки конечным получателям. Эффективное отслеживание позволяет своевременно выявлять отклонения от планов, минимизировать потери и обеспечить высокое качество обслуживания клиентов.

Отслеживание продукции включает в себя несколько ключевых этапов:

1. Формирование партий продукции.

После окончания печатного процесса продукция сортируется по партиям в зависимости от пунктов назначения, объёмов заказа и особенностей доставки. Каждая партия сопровождается комплектом учётных документов: спецификацией, актами приёмки и маршрутными листами.

2. Регистрация данных о продукции.

Все сведения о произведённых партиях фиксируются в информационной системе или в бумажных реестрах: наименование издания, тираж, дата выпуска, пункты отправления и назначения. На этом этапе важно обеспечить корректность ввода информации для предотвращения ошибок на следующих этапах.

3. Передача продукции в зону отгрузки.

Партии продукции передаются на склад временного хранения или непосредственно в зону отгрузки, где происходит дальнейшая подготовка к транспортировке. Здесь может производиться дополнительная упаковка, маркировка и распределение по маршрутам.

4. Организация транспортировки.

Продукция отгружается согласно заранее утверждённым маршрутам доставки. Ответственные лица контролируют процесс погрузки и передачу продукции транспортным компаниям или собственным логистическим подразделениям.

5. Контроль выполнения доставки.

На протяжении всего пути следования осуществляется мониторинг статуса доставки: отправка, движение, прибытие в пункт назначения. Информация о выполнении или задержке доставки фиксируется в системе.

6. Приёмка продукции в пункте назначения.

При поступлении продукции на склад или в почтовое отделение производится приёмка по количеству и качеству. В случае выявления несоответствий формируются акты расхождений, которые затем обрабатываются в системе учёта.

Методы отслеживания продукции

На практике для контроля распределения печатной продукции могут применяться различные методы, ручной контроль.

Использование бумажных документов, ведомостей, реестров. Метод прост в реализации, но подвержен ошибкам и требует значительных трудозатрат, Электронные таблицы. Применение программ вроде Microsoft Excel позволяет автоматизировать подсчёты, но не обеспечивает достаточного уровня надёжности и оперативности в обработке информации.

Информационные системы учёта**,** специализированные программные решения позволяют в реальном времени отслеживать движение партий, получать отчёты о состоянии доставки, фиксировать отклонения и корректировать маршруты, современные типографии всё чаще стремятся перейти от ручных методов к использованию автоматизированных систем, поскольку это позволяет существенно повысить точность учёта, ускорить обработку данных и обеспечить прозрачность всех операций. Проблемы и риски при отсутствии системного отслеживания.

При отсутствии налаженной системы отслеживания типографии сталкиваются с рядом проблем, потеря или задержка партий продукции, невозможность оперативного получения актуальной информации о местонахождении тиражей, увеличение издержек на логистику за счёт неэффективного планирования маршрутов, снижение качества обслуживания клиентов и партнёров, рост числа возвратов и перерасходов ресурсов.

Эти риски особенно критичны для типографий, выпускающих продукцию с жёсткими сроками распространения, таких как ежедневные газеты или еженедельные журналы.

Роль программной системы в процессе отслеживания

Внедрение специализированной программной системы позволяет автоматизировать все этапы отслеживания печатной продукции, централизовать хранение информации о партиях продукции, автоматически формировать сопроводительные документы, в реальном времени отслеживать этапы доставки, своевременно выявлять и устранять отклонения, обеспечивать прозрачную отчётность для всех участников процесса.

Такая система способствует повышению эффективности работы типографии, оптимизации логистических процессов и укреплению деловой репутации предприятия.Таким образом, понимание процесса отслеживания продукции и осознание его роли в общем технологическом цикле позволяет обоснованно подойти к проектированию и реализации эффективной программной системы, способной значительно упростить управление процессами распределения печатной продукции.

## 1.3 Причины необходимости автоматизации процессов распределения

На современном этапе развития типографской отрасли эффективность распределения печатной продукции становится решающим фактором конкурентоспособности. Традиционные методы учёта и отслеживания тиражей, такие как ручное ведение документации или использование простых электронных таблиц, уже не справляются с текущими задачами.

Они порождают множество проблем: высокую вероятность ошибок из-за опечаток, неверных расчётов объёмов или неправильного указания пунктов назначения; задержки в обработке данных, которые замедляют принятие решений; отсутствие прозрачности, усложняющее отслеживание статуса заказов; сложности с масштабированием при росте объёмов производства; а также повышенные риски потерь и недостач из-за недостаточного контроля. Эти недостатки подчёркивают острую необходимость перехода к автоматизированным системам.

Современные типографии сталкиваются с жёсткими требованиями: оперативная обработка больших объёмов данных, минимизация человеческого фактора, точность доставки, возможность быстрой корректировки маршрутов и заказов, а также предоставление клиентам актуальной информации о статусе их заказов. Эти задачи невозможно эффективно решать без использования специализированных информационных технологий.

Внедрение программных систем для управления распределением даёт типографиям значительные преимущества. Такие системы обеспечивают точный учёт данных о партиях продукции, пунктах назначения и сроках доставки, сокращают операционные затраты за счёт уменьшения ручной работы, ускоряют формирование отгрузочных документов и планирование маршрутов, повышают прозрачность процессов благодаря отчётности в реальном времени, укрепляют доверие клиентов через оперативное информирование и легко адаптируются к росту производства и расширению сети поставок.

Для достижения максимальной эффективности автоматизация должна быть комплексной и учитывать особенности конкретной типографии. Простая замена бумажных ведомостей на электронные таблицы не решает ключевых проблем. Эффективная система должна включать следующие компоненты:

* Единое хранилище данных о всей продукции.
* Модули для планирования отгрузок и маршрутов.
* Инструменты для контроля выполнения доставки.
* Системы уведомлений об отклонениях и проблемах.
* Механизмы формирования и хранения отчётности.

Только при таком подходе автоматизация позволит значительно повысить эффективность распределения печатной продукции, минимизировать ошибки и укрепить позиции типографии на рынке.

## 1.4 Особенности построения систем отслеживания распределения в типографиях

Создание эффективной системы отслеживания распределения печатной продукции в типографиях требует глубокого анализа всех аспектов производственных и логистических процессов. Такая система должна быть не только функциональной и надёжной, но и максимально адаптированной к уникальным особенностям конкретной типографии, чтобы обеспечивать бесперебойную работу, минимизировать ошибки и повышать конкурентоспособность предприятия.

В условиях современного рынка, где скорость, точность и прозрачность доставки становятся определяющими факторами успеха, внедрение передовых информационных технологий выходит на первый план. Типографии, которые продолжают полагаться на устаревшие методы, рискуют потерять клиентов из-за задержек, путаницы в заказах или неконкурентоспособных затрат. Автоматизация процессов распределения позволяет не только устранить эти проблемы, но и открыть новые возможности для оптимизации работы и повышения удовлетворённости клиентов.

Особенности печатной продукции создают уникальные вызовы для проектирования системы отслеживания. Разнообразие форматов — от небольших листовок и газет до массивных книг, каталогов и рекламных буклетов — требует учёта различий в размерах, весе, способах упаковки и условиях хранения, чтобы предотвратить повреждения или ошибки при транспортировке.

Тиражи могут варьироваться от ограниченных партий для локальных заказчиков до огромных объёмов для общенациональных или международных клиентов, что подчёркивает необходимость гибкости системы.

Срочные и сезонные заказы, особенно в случае периодических изданий, таких как журналы или рекламные материалы, добавляют дополнительное давление: даже небольшая задержка может сделать продукцию неактуальной, что приводит к финансовым потерям и репутационным рискам. Точная сортировка по направлениям доставки критически важна, чтобы каждый клиент или пункт назначения получил именно тот тираж, который был заказан, без путаницы или недостач. Эти факторы требуют от системы не только высокой детализации учёта, но и способности оперативно обрабатывать большие объёмы данных, поддерживая ритм работы типографии и отвечая строгим ожиданиям клиентов.

Функциональность системы отслеживания должна охватывать все ключевые этапы распределения, чтобы обеспечить полный контроль над процессами. Регистрация партий продукции с указанием их характеристик, таких как наименование, количество экземпляров, дата печати и пункт назначения, создаёт основу для точного и прозрачного учёта. Гибкое управление маршрутами позволяет не только назначать оптимальные пути доставки, но и оперативно корректировать их в случае изменений, таких как срочные запросы клиентов или проблемы на маршруте, например, дорожные заторы или изменения в расписании.

Интеграция с производственными данными связывает информацию о печати с процессами распределения, создавая единую цепочку управления, которая исключает разрывы в информационном потоке. Автоматизация формирования сопроводительных документов, таких как накладные, акты передачи и маршрутные листы, устраняет необходимость ручного ввода данных, что снижает вероятность ошибок и значительно экономит время сотрудников.

Мониторинг статуса доставки в реальном времени позволяет отслеживать местоположение партий, оперативно реагировать на любые сбои и предоставлять клиентам актуальную информацию о статусе их заказов.

Генерация отчётов и аналитики даёт руководству возможность анализировать эффективность процессов, выявлять узкие места и разрабатывать стратегии для их устранения. Без этих функций система будет неполной и не сможет в полной мере отвечать требованиям современной типографии, особенно в условиях высоких производственных нагрузок и сложных логистических задач.

Архитектура системы должна быть построена на принципах масштабируемости, чтобы справляться с увеличением объёмов производства и расширением сети поставок без потери производительности. Надёжность и отказоустойчивость обеспечивают сохранность данных и стабильную работу даже при технических сбоях, что особенно важно в пиковые периоды, такие как предпраздничные сезоны или крупные рекламные кампании.

Удобный и интуитивно понятный интерфейс необходим для сотрудников склада, логистов и менеджеров, чтобы минимизировать время на обучение, повысить продуктивность и снизить вероятность ошибок из-за человеческого фактора. Безопасность данных требует строгого разграничения доступа, чтобы предотвратить несанкционированные изменения, утечки конфиденциальной информации или вмешательство в критически важные процессы.

Резервное копирование и механизмы быстрого восстановления после сбоев дополнительно усиливают надёжность системы, минимизируя риски простоев и потерь данных. Гибкость системы также важна, чтобы её можно было адаптировать под изменения в бизнес-процессах, внедрение новых видов продукции или требования новых клиентов, что делает её универсальным инструментом для долгосрочного использования.

Интеграция с другими системами предприятия — ключевой фактор для достижения максимальной эффективности. Связь с ERP-системами управления производством позволяет синхронизировать данные о печати и распределении, создавая целостную картину всех процессов.

Интеграция с бухгалтерскими и складскими программами упрощает учёт затрат, управление запасами и контроль остатков, что особенно важно для крупных типографий с большим ассортиментом продукции. Подключение к CRM-системам обеспечивает оперативное взаимодействие с клиентами, позволяя предоставлять им актуальную информацию о статусе заказов и повышая уровень их удовлетворённости.

Такое объединение создаёт единое информационное пространство, где все этапы от планирования печати до доставки продукции — работают как единое целое. Это не только ускоряет обработку заказов, но и даёт руководству полное представление о состоянии дел, что упрощает стратегическое планирование, прогнозирование и принятие обоснованных управленческих решений.

Типичная система отслеживания может включать следующие функциональные модули:

* Модуль регистрации печатной продукции для фиксации всех данных о тиражах.
* Модуль планирования маршрутов и отгрузок для оптимизации логистики.
* Модуль отслеживания статуса партий для контроля доставки в реальном времени.
* Модуль формирования документации для автоматической генерации необходимых бумаг.
* Модуль генерации отчётов и статистики для анализа эффективности процессов.

Каждый из этих модулей должен быть тесно интегрирован с остальными, чтобы обеспечить целостность и слаженность работы системы. Такой подход позволяет типографии не только минимизировать затраты и повысить точность доставки, но и укрепить свои позиции на рынке за счёт высокого уровня сервиса, оперативности и надёжности.

# ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММНОЙ СИСТЕМЫ

## 2.1 Требования к программе

Для корректного функционирования программы «Система учёта газет» необходимо не менее 50 МБ свободного места на жестком диске. Программа разработана для работы на операционных системах Microsoft Windows 7, Windows 8, Windows 10 или более поздних версиях, поддерживающих 64-битные архитектуры.

Программа должна обеспечивать стабильную работу при длительном использовании, а также предусматривать сохранение данных в текстовых файлах для предотвращения их потери в случае сбоев аппаратной или программной части.

Разработка программного продукта осуществлялась с использованием следующей конфигурации аппаратного и программного обеспечения:

* Процессор: Intel Core i5 с тактовой частотой до 2.4 ГГц.
* Оперативная память: 8 ГБ DDR4.
* Видеокарта: Интегрированная Intel UHD Graphics.
* Операционная система: Windows 10 Pro 64-битная версия.
* Накопитель: SSD на 256 ГБ.
* Дисплей: 15.6-дюймовый Full HD (1920 x 1080).

Данная конфигурация обеспечивала достаточную производительность для разработки, тестирования и отладки программы, включая компиляцию кода и работу с текстовыми файлами данных.

Минимально необходимая конфигурация для использования программы:

* Операционная система: Windows 7, 8, 10 (64-битные версии).
* Процессор: Intel Core 2 Duo или аналогичный с частотой от 1.6 ГГц.
* Оперативная память: не менее 2 ГБ.
* Свободное место на жестком диске: не менее 50 МБ.
* Минимальное разрешение экрана: 1024 x 768 пикселей.

Такие требования к аппаратной и программной части позволяют обеспечить стабильную и бесперебойную работу программы, а также комфортное использование её функциональных возможностей, включая управление списками газет и почтовых отделений, расчёт стоимости и работу с файлами данных.

## 2.2 Руководство пользователя

Руководство пользователя по использованию консольного приложения «Система учёта газет» предназначено для ознакомления с функциональными возможностями программы и обучения работе с ней. Документ обеспечивает полную информативность о структуре интерфейса приложения и описывает все доступные функции.

Для запуска программы необходимо дважды кликнуть левой кнопкой мыши (ЛКМ) по исполняемому файлу с расширением .exe, созданному в среде Microsoft Visual Studio. Убедитесь, что файлы newspapers.txt и postoffices.txt находятся в той же директории, что и исполняемый файл, если требуется загрузка ранее сохранённых данных. Окно консоли после запуска показано на рисунке 1.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1 Запуск приложения

После запуска программы открывается первое меню, отображаемое в консоли (рис. 2). Меню предоставляет пользователю пять опций, выбор которых осуществляется нажатием соответствующей клавиши на клавиатуре.

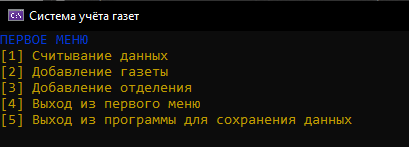


Рис. 2 Первое меню программы

Выбор пункта [1] в первом меню инициирует загрузку данных о газетах и почтовых отделениях из файлов newspapers.txt и postoffices.txt. При успешной загрузке отображается сообщение «Считывание удалось» (рис. 3). В случае ошибки, например, если файлы отсутствуют или повреждены, выводится сообщение «Считывание не удалось».

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3 Успешное считывание данных

Для добавления новой газеты в базу данных выберите пункт [2] в первом меню. Пользователю будет предложено ввести следующие данные:

* Название газеты (например, «Вечерние новости»).
* Индекс газеты (например, «12345»).
* ФИО редактора (например, «Иванов И.И.»).
* Цена в рублях (например, «45,50»).

После ввода данных отображается сообщение «Новая газета успешно добавлена» (рис. 4). Данные сохраняются в файл newspapers.txt.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 4 Добавление клиента

Для добавления нового почтового отделения выберите пункт [3] в первом меню. Пользователю необходимо ввести:

* Номер отделения (целое число, например, «1»).
* Адрес отделения (например, «ул. Ленина, 10»).

После ввода отображается сообщение «Новое отделение успешно добавлено» (рис. 5). Данные сохраняются в файл postoffices.txt.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 5 Добавление отделения

Если все необходимые действия в первом меню выполнены, пользователь может выбрать пункт [4] для перехода в главное меню (рис. 6). Пункт [5] завершает программу с сохранением всех данных.

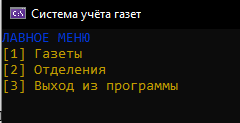


Рис. 6 Переход в главное меню

Главное меню предоставляет доступ к трём основным разделам: «Газеты», «Отделения» и «Выход из программы»

Перейдя в раздел «Газеты» ([1] в главном меню), пользователь видит подменю с четырьмя пунктами (рис. 8).

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 8 Подменю «Газеты»**0**

Выбор пункта [1] отображает список всех газет с их характеристиками: номер, название, индекс, редактор и цена (рис. 9). Если газет нет, выводится сообщение «Газет нет».

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 9 Вывод информации о газетах

Для редактирования данных газеты выберите пункт [2]. Программа отображает список газет, после чего пользователь вводит номер газеты. Затем можно изменить название, индекс, редактора или цену, нажимая Enter для сохранения текущих значений. После редактирования отображается сообщение «Газета успешно отредактирована» (рис. 10).

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 10 Редактирование газеты

Выбор пункта [3] позволяет удалить газету. После отображения списка газет пользователь вводит номер газеты для удаления. При успешном удалении выводится сообщение «Газета успешно удалена» (рис. 11).

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 11 Удаление газеты

Пункт [4] возвращает пользователя в главное меню.

Перейдя в раздел «Отделения» ([2] в главном меню), пользователь видит подменю с тремя пунктами (рис. 12).

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 12 Подменю «Отделения»

Для добавления газеты в почтовое отделение выберите пункт [1]. Пользователь вводит номер отделения; если отделение не существует, программа запрашивает его адрес. Затем отображается список газет, и пользователь выбирает номер газеты для добавления. После этого отображается информация об отделении и добавленной газете (рис. 13).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 13 Добавление газеты в отделение

Пункт [2] позволяет рассчитать суммарную стоимость всех газет в выбранном отделении. Пользователь выбирает отделение из списка, после чего программа выводит общую стоимость (рис. 14). Если в отделении нет газет, отображается соответствующее сообщение.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 14 Расчёт стоимости в отделении

Пункт [3] возвращает пользователя в главное меню.

Выбор пункта [3] в главном меню завершает работу программы, автоматически сохраняя все данные в файлы newspapers.txt и postoffices.txt (рис. 15).

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 15 Выход из программы

## 2.3 Руководство программиста

Данный раздел описывает программную реализацию консольного приложения «Система учёта газет», разработанного на языке C# в среде Microsoft Visual Studio. Программа предназначена для управления данными о газетах и почтовых отделениях, их хранения в текстовых файлах и выполнения операций, таких как добавление, редактирование, удаление и расчёт стоимости газет. Руководство адресовано программистам, которые будут поддерживать или модифицировать код. Здесь описаны структура программы, классы, методы, алгоритмы и рекомендации по доработке.

Программа организована в двух пространствах имён: Kursah (классы Newspaper, PostOffice, PrintingHouse) и Курсовая (класс Program, реализующий основную логику). Основной функционал сосредоточен в классе Program, который управляет пользовательским интерфейсом и взаимодействием с данными. Данные хранятся в текстовых файлах newspapers.txt (список газет) и postoffices.txt (список отделений и их газет). Программа использует объектно-ориентированный подход, разделяя функциональность на классы и методы.

Основные компоненты:

* Классы данных: Newspaper (газета), PostOffice (почтовое отделение), PrintingHouse (типография, не используется).
* Логика управления: Класс Program с методами для меню и операций с данными.
* Хранение данных: Текстовые файлы с разделителями.
* Интерфейс: Консольный интерфейс с многоуровневыми меню и цветовым оформлением.

Описание классов

Программа включает три класса для данных и один для управления. Ниже описаны их свойства и методы.

Класс Newspaper описывает сущность «газета» и хранит её характеристики.

Свойства:

* Name (тип string): Название газеты.
* Index (тип string): Уникальный индекс.
* Editor (тип string): ФИО редактора.
* Price (тип decimal): Цена в рублях.

Методы:

Print(): Выводит информацию о газете в консоль в формате: «Название: {Name}, Индекс: {Index}, Редактор: {Editor}, Цена: {Price} руб.».

Класс PostOffice описывает почтовое отделение и его газеты.

Свойства:

* Number (тип int): Номер отделения.
* Address (тип string): Адрес.
* Newspapers (тип List<Newspaper>): Список газет.

Конструктор:

PostOffice(): Инициализирует пустой список Newspapers.

Методы:

Print(): Выводит номер, адрес и названия газет.

Класс PrintingHouse описывает типографию, но не используется в текущей версии.

Свойства:

* Id (тип int): Идентификатор.
* Name (тип string): Название.
* Address (тип string): Адрес.
* Newspapers (тип List<Newspaper>): Список газет.

Конструктор:

PrintingHouse(): Инициализирует пустой список.

Класс Program реализует логику управления и пользовательский интерфейс.

Поля:

* newspaperList (тип List<Newspaper>): Список газет.
* postOffices (тип List<PostOffice>): Список отделений.
* newspaperFile (тип string): Путь к newspapers.txt.
* postOfficeFile (тип string): Путь к postoffices.txt.

Основной метод:

Основные методы класса Program:

Методы разделены на группы: отображение меню, управление газетами, управление отделениями и работа с файлами.

Методы отображения меню:

* ShowFirstMenu(): Показывает первое меню с опциями: считывание данных, добавление газеты, добавление отделения, выход из меню, выход из программы.
* ShowMainMenu(): Отображает главное меню: «Газеты», «Отделения», «Выход».
* ShowNewspapersMenu(): Подменю для газет: вывод, редактирование, удаление, выход.
* ShowPostOfficesMenu(): Подменю для отделений: добавление газеты, расчёт стоимости, выход.

Меню используют цветовое оформление (ConsoleColor.DarkBlue для заголовков, ConsoleColor.DarkYellow для пунктов) и Console.ReadKey(true) для обработки ввода.

Методы управления газетами:

* AddNewspaper(): Запрашивает данные (название, индекс, редактор, цена), добавляет газету в newspaperList, сохраняет в файл.
* ShowAllNewspapers(): Выводит список газет с номерами.
* EditNewspaper(): Редактирует газету по номеру, позволяя изменить данные или оставить текущие.
* RemoveNewspaper(): Удаляет газету по номеру, обновляя файл.

Пример AddNewspaper:

static void AddNewspaper()

{

Console.WriteLine("=== ДОБАВЛЕНИЕ ГАЗЕТЫ ===");

Newspaper n = new Newspaper();

Console.Write("Название: ");

n.Name = Console.ReadLine();

Console.Write("Индекс: ");

n.Index = Console.ReadLine();

Console.Write("ФИО редактора: ");

n.Editor = Console.ReadLine();

Console.Write("Цена (в руб.): ");

if (decimal.TryParse(Console.ReadLine(), out decimal price))

n.Price = price;

else

n.Price = 0;

newspaperList.Add(n);

SaveData();

}

Методы управления отделениями

* AddPostOffice(): Создаёт отделение, запрашивая номер и адрес.
* AddNewspaperToPostOffice(): Добавляет газету в отделение, создавая новое, если его нет.
* CalculateTotalCostInPostOffice(): Рассчитывает сумму цен газет в отделении с помощью LINQ (Sum).

Методы работы с файлами

* SaveData(): Сохраняет газеты в newspapers.txt (формат: Name,Index,Editor,Price) и отделения в postoffices.txt (номер, адрес, затем газеты с отступом).
* LoadData(): Читает файлы, создавая объекты Newspaper и PostOffice.

# ЗАКЛЮЧЕНИЕ

В данной курсовой работе разработана информационная система «Система учёта газет» для управления данными о газетах и их распределением по почтовым отделениям. Программа реализована на языке C# в среде Microsoft Visual Studio, с использованием текстовых файлов newspapers.txt и postoffices.txt для хранения данных.

Пользовательский интерфейс реализован через консольные средства ввода-вывода, обеспечивая простоту навигации и понятные команды. Он позволяет добавлять, редактировать, удалять данные о газетах и отделениях, а также рассчитывать их стоимость. Поддержка кодировки UTF-8 обеспечивает корректное отображение русского текста.

Данные о газетах и отделениях хранятся в отдельных текстовых файлах, что упрощает управление и резервное копирование.

Основные преимущества системы:

1. Простой и интуитивно понятный консольный интерфейс.
2. Надёжное хранение данных в текстовых файлах.
3. Эффективный расчёт стоимости с использованием LINQ.

Области для улучшения:

1. Добавление поиска и сортировки газет.
2. Переход на JSON или SQLite для хранения данных.
3. Интеграция класса PrintingHouse для учёта типографий.

Цель работы достигнута: создана информационная система, обеспечивающая управление данными о газетах и почтовых отделениях, с потенциалом для дальнейших доработок.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Семакин И.Г., Шеина Т.Ю., Загвоздкин В.А. Информатика и ИКТ. Базовый уровень. – М.: БИНОМ. Лаборатория знаний, 2020. – 240 с.

2. Куликов С.В. Разработка информационных систем. – СПб.: Питер, 2021. – 320 с.

3. Виноградов В.Н., Чернова Я.Г. Единая государственная информационная система мониторинга // Информатизация образования и науки. – 2018. – № 3. – С. 45–52.

4. Документация Microsoft Visual Studio [Электронный ресурс]. – Режим доступа: <https://docs.microsoft.com/en-us/visualstudio/>. – Дата доступа: 07.06.2025.

5. ГОСТ 19.401-78 Единая система программной документации. Тексты программ. Общие положения. – М.: Издательство стандартов, 1978. – 12 с.

6. ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчёт о научно-исследовательской работе. Общие требования. – М.: Стандартинформ, 2017. – 20 с.

7. Примеры курсовых работ по информационным системам [Электронный ресурс]. – Режим доступа: <https://infourok.ru/>. – Дата доступа: 07.06.2025.

8. Электронная библиотека Тольяттинского государственного университета [Электронный ресурс]. – Режим доступа: <https://dspace.tltsu.ru/>. – Дата доступа: 07.06.2025.

# ПРИЛОЖЕНИЕ А

using Kursah;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

namespace Курсовая

{

internal class Program

{

static List<Newspaper> newspaperList = new List<Newspaper>();

static List<PostOffice> postOffices = new List<PostOffice>();

static string newspaperFile = "newspapers.txt";

static string postOfficeFile = "postoffices.txt";

static void Main(string[] args)

{

Console.OutputEncoding = System.Text.Encoding.UTF8;

Console.Title = "Система учёта газет";

bool flag = true;

// Первое меню: загрузка и добавление данных

while (flag)

{

Console.Clear();

ShowFirstMenu();

switch (Console.ReadKey(true).KeyChar)

{

case '1':

try

{

Console.Clear();

LoadData();

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Считывание удалось");

Console.ResetColor();

Console.Write("Хотите продолжить? (Да/Нет): ");

string answer = Console.ReadLine();

if (answer.ToLower() == "нет")

flag = false;

}

catch

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Считывание не удалось");

Console.ResetColor();

Console.ReadKey();

}

break;

case '2':

try

{

Console.Clear();

AddNewspaper();

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Новая газета успешно добавлена");

Console.ResetColor();

Console.Write("Хотите продолжить выбор из первого меню? (Да/Нет): ");

string answer = Console.ReadLine();

if (answer.ToLower() == "нет")

flag = false;

}

catch

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Произошла ошибка, попробуйте еще раз");

Console.ResetColor();

Console.ReadKey();

}

break;

case '3':

try

{

Console.Clear();

AddPostOffice();

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Новое отделение успешно добавлено");

Console.ResetColor();

Console.Write("Хотите продолжить выбор из первого меню? (Да/Нет): ");

string answer = Console.ReadLine();

if (answer.ToLower() == "нет")

flag = false;

}

catch

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Произошла ошибка, попробуйте еще раз");

Console.ResetColor();

Console.ReadKey();

}

break;

case '4':

flag = false;

break;

case '5':

SaveData();

return;

default:

Console.Clear();

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Ошибка, попробуйте еще раз");

Console.ResetColor();

Console.ReadKey();

break;

}

}

// Главное меню

bool flagMain = true;

while (flagMain)

{

Console.Clear();

ShowMainMenu();

switch (Console.ReadKey(true).KeyChar)

{

case '1':

bool flagNewspapers = true;

while (flagNewspapers)

{

Console.Clear();

ShowNewspapersMenu();

switch (Console.ReadKey(true).KeyChar)

{

case '1':

try

{

Console.Clear();

ShowAllNewspapers();

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Информация успешно выведена");

Console.ResetColor();

Console.Write("Хотите продолжить? (Да/Нет): ");

string answer = Console.ReadLine();

if (answer.ToLower() == "нет")

flagNewspapers = false;

}

catch

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Ошибка, попробуйте еще раз");

Console.ResetColor();

Console.ReadKey();

}

break;

case '2':

try

{

Console.Clear();

EditNewspaper();

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Газета успешно отредактирована");

Console.ResetColor();

Console.Write("Хотите продолжить? (Да/Нет): ");

string answer = Console.ReadLine();

if (answer.ToLower() == "нет")

flagNewspapers = false;

}

catch

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Ошибка, попробуйте еще раз");

Console.ResetColor();

Console.ReadKey();

}

break;

case '3':

try

{

Console.Clear();

RemoveNewspaper();

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Газета успешно удалена");

Console.ResetColor();

Console.Write("Хотите продолжить? (Да/Нет): ");

string answer = Console.ReadLine();

if (answer.ToLower() == "нет")

flagNewspapers = false;

}

catch

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Ошибка, попробуйте еще раз");

Console.ResetColor();

Console.ReadKey();

}

break;

case '4':

flagNewspapers = false;

break;

default:

Console.Clear();

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Ошибка, попробуйте еще раз");

Console.ResetColor();

Console.ReadKey();

break;

}

}

break;

case '2':

bool flagPostOffices = true;

while (flagPostOffices)

{

Console.Clear();

ShowPostOfficesMenu();

switch (Console.ReadKey(true).KeyChar)

{

case '1':

try

{

Console.Clear();

AddNewspaperToPostOffice();

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Газета успешно добавлена в отделение");

Console.ResetColor();

Console.Write("Хотите продолжить? (Да/Нет): ");

string answer = Console.ReadLine();

if (answer.ToLower() == "нет")

flagPostOffices = false;

}

catch

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Ошибка, попробуйте еще раз");

Console.ResetColor();

Console.ReadKey();

}

break;

case '2':

try

{

Console.Clear();

CalculateTotalCostInPostOffice();

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Стоимость успешно рассчитана");

Console.ResetColor();

Console.Write("Хотите продолжить? (Да/Нет): ");

string answer = Console.ReadLine();

if (answer.ToLower() == "нет")

flagPostOffices = false;

}

catch

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Ошибка, попробуйте еще раз");

Console.ResetColor();

Console.ReadKey();

}

break;

case '3':

flagPostOffices = false;

break;

default:

Console.Clear();

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Ошибка, попробуйте еще раз");

Console.ResetColor();

Console.ReadKey();

break;

}

}

break;

case '3':

SaveData();

return;

default:

Console.Clear();

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Ошибка, попробуйте еще раз");

Console.ResetColor();

Console.ReadKey();

break;

}

}

}

static void ShowFirstMenu()

{

Console.ForegroundColor = ConsoleColor.DarkBlue;

Console.WriteLine("ПЕРВОЕ МЕНЮ");

Console.ResetColor();

Console.ForegroundColor = ConsoleColor.DarkYellow;

Console.WriteLine("[1] Считывание данных");

Console.WriteLine("[2] Добавление газеты");

Console.WriteLine("[3] Добавление отделения");

Console.WriteLine("[4] Выход из первого меню");

Console.WriteLine("[5] Выход из программы для сохранения данных");

Console.ResetColor();

}

static void ShowMainMenu()

{

Console.ForegroundColor = ConsoleColor.DarkBlue;

Console.WriteLine("ГЛАВНОЕ МЕНЮ");

Console.ResetColor();

Console.ForegroundColor = ConsoleColor.DarkYellow;

Console.WriteLine("[1] Газеты");

Console.WriteLine("[2] Отделения");

Console.WriteLine("[3] Выход из программы");

Console.ResetColor();

}

static void ShowNewspapersMenu()

{

Console.ForegroundColor = ConsoleColor.DarkYellow;

Console.WriteLine("[1] Вывод информации о газетах");

Console.WriteLine("[2] Редактировать газету");

Console.WriteLine("[3] Удалить газету");

Console.WriteLine("[4] Выход в главное меню");

Console.ResetColor();

}

static void ShowPostOfficesMenu()

{

Console.ForegroundColor = ConsoleColor.DarkYellow;

Console.WriteLine("[1] Добавить газету в отделение");

Console.WriteLine("[2] Посчитать стоимость в отделении");

Console.WriteLine("[3] Выход в главное меню");

Console.ResetColor();

}

static void AddNewspaper()

{

Console.WriteLine("=== ДОБАВЛЕНИЕ ГАЗЕТЫ ===");

Newspaper n = new Newspaper();

Console.Write("Название: ");

n.Name = Console.ReadLine();

Console.Write("Индекс: ");

n.Index = Console.ReadLine();

Console.Write("ФИО редактора: ");

n.Editor = Console.ReadLine();

Console.Write("Цена (в руб.): ");

if (decimal.TryParse(Console.ReadLine(), out decimal price))

n.Price = price;

else

n.Price = 0;

newspaperList.Add(n);

SaveData();

}

static void AddPostOffice()

{

Console.WriteLine("=== ДОБАВЛЕНИЕ ОТДЕЛЕНИЯ ===");

PostOffice po = new PostOffice();

Console.Write("Номер отделения: ");

if (int.TryParse(Console.ReadLine(), out int number))

po.Number = number;

Console.Write("Адрес отделения: ");

po.Address = Console.ReadLine();

po.Newspapers = new List<Newspaper>();

postOffices.Add(po);

SaveData();

}

static void EditNewspaper()

{

ShowAllNewspapers();

Console.Write("\nВведите номер газеты для редактирования: ");

if (int.TryParse(Console.ReadLine(), out int i) && i > 0 && i <= newspaperList.Count)

{

Newspaper n = newspaperList[i - 1];

Console.WriteLine("=== РЕДАКТИРОВАНИЕ ===");

Console.Write("Новое название (Enter — оставить): ");

string input = Console.ReadLine();

if (input != "") n.Name = input;

Console.Write("Новый индекс: ");

input = Console.ReadLine();

if (input != "") n.Index = input;

Console.Write("Новый редактор: ");

input = Console.ReadLine();

if (input != "") n.Editor = input;

Console.Write("Новая цена: ");

input = Console.ReadLine();

if (decimal.TryParse(input, out decimal p)) n.Price = p;

SaveData();

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Неверный номер.");

Console.ResetColor();

Console.ReadKey();

}

}

static void ShowAllNewspapers()

{

Console.WriteLine("=== СПИСОК ВСЕХ ГАЗЕТ ===\n");

if (newspaperList.Count == 0)

{

Console.WriteLine("Газет нет.");

}

else

{

for (int i = 0; i < newspaperList.Count; i++)

{

Console.WriteLine($"[{i + 1}]");

newspaperList[i].Print();

Console.WriteLine();

}

}

Console.WriteLine();

}

static void RemoveNewspaper()

{

ShowAllNewspapers();

Console.Write("\nВведите номер газеты для удаления: ");

if (int.TryParse(Console.ReadLine(), out int i) && i > 0 && i <= newspaperList.Count)

{

newspaperList.RemoveAt(i - 1);

SaveData();

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Неверный номер.");

Console.ResetColor();

Console.ReadKey();

}

}

static void AddNewspaperToPostOffice()

{

Console.WriteLine("=== ДОБАВЛЕНИЕ ГАЗЕТЫ В ОТДЕЛЕНИЕ ===");

Console.Write("Введите номер отделения: ");

int.TryParse(Console.ReadLine(), out int num);

PostOffice po = postOffices.Find(x => x.Number == num);

if (po == null)

{

po = new PostOffice();

po.Number = num;

Console.Write("Введите адрес отделения: ");

po.Address = Console.ReadLine();

po.Newspapers = new List<Newspaper>();

postOffices.Add(po);

}

ShowAllNewspapers();

Console.Write("Введите номер газеты: ");

if (int.TryParse(Console.ReadLine(), out int i) && i > 0 && i <= newspaperList.Count)

{

po.Newspapers.Add(newspaperList[i - 1]);

SaveData();

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Неверный номер.");

Console.ResetColor();

Console.ReadKey();

}

Console.WriteLine();

po.Print();

}

static void CalculateTotalCostInPostOffice()

{

Console.WriteLine("=== СУММАРНАЯ СТОИМОСТЬ ГАЗЕТ В ОТДЕЛЕНИИ ===");

if (postOffices.Count == 0)

{

Console.WriteLine("Нет отделений для расчёта.");

Console.ReadKey();

return;

}

Console.WriteLine("Список отделений:");

for (int i = 0; i < postOffices.Count; i++)

{

Console.WriteLine($"{i + 1}. Отделение №{postOffices[i].Number}, Адрес: {postOffices[i].Address}");

}

Console.Write("Выберите отделение (введите номер): ");

if (!int.TryParse(Console.ReadLine(), out int postOfficeIndex) || postOfficeIndex < 1 || postOfficeIndex > postOffices.Count)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Неверный выбор отделения.");

Console.ResetColor();

Console.ReadKey();

return;

}

var selectedPostOffice = postOffices[postOfficeIndex - 1];

if (selectedPostOffice.Newspapers.Count == 0)

{

Console.WriteLine($"В отделении №{selectedPostOffice.Number} нет газет.");

Console.ReadKey();

return;

}

decimal totalCost = selectedPostOffice.Newspapers.Sum(newspaper => newspaper.Price);

Console.WriteLine($"Суммарная стоимость газет в отделении №{selectedPostOffice.Number}: {totalCost} руб.");

Console.ReadKey();

}

static void SaveData()

{

using (StreamWriter sw = new StreamWriter(newspaperFile))

{

foreach (var newspaper in newspaperList)

{

sw.WriteLine($"{newspaper.Name},{newspaper.Index},{newspaper.Editor},{newspaper.Price}");

}

}

using (StreamWriter sw = new StreamWriter(postOfficeFile))

{

foreach (var postOffice in postOffices)

{

sw.WriteLine($"{postOffice.Number},{postOffice.Address}");

foreach (var newspaper in postOffice.Newspapers)

{

sw.WriteLine($" {newspaper.Name},{newspaper.Index},{newspaper.Editor},{newspaper.Price}");

}

}

}

}

static void LoadData()

{

if (File.Exists(newspaperFile))

{

using (StreamReader sr = new StreamReader(newspaperFile))

{

string line;

while ((line = sr.ReadLine()) != null)

{

var parts = line.Split(',');

var newspaper = new Newspaper

{

Name = parts[0],

Index = parts[1],

Editor = parts[2],

Price = decimal.TryParse(parts[3], out decimal price) ? price : 0

};

newspaperList.Add(newspaper);

}

}

}

if (File.Exists(postOfficeFile))

{

using (StreamReader sr = new StreamReader(postOfficeFile))

{

string line;

PostOffice postOffice = null;

while ((line = sr.ReadLine()) != null)

{

if (line.StartsWith(" "))

{

var parts = line.Trim().Split(',');

var newspaper = new Newspaper

{

Name = parts[0],

Index = parts[1],

Editor = parts[2],

Price = decimal.TryParse(parts[3], out decimal price) ? price : 0

};

postOffice.Newspapers.Add(newspaper);

}

else

{

var parts = line.Split(',');

postOffice = new PostOffice

{

Number = int.Parse(parts[0]),

Address = parts[1],

Newspapers = new List<Newspaper>()

};

postOffices.Add(postOffice);

}

}

}

}

}

}

}