**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc165050057)

[ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ И ОБЩАЯ ПОСТАНОВКА ЗАДАЧ 5](#_Toc165050058)

[1.1 История и развитие гостиничного бизнеса 5](#_Toc165050059)

[1.2 Организационная структура гостиницы 8](#_Toc165050060)

[1.3 Функции, выполняемые администратором гостиницы 12](#_Toc165050061)

[1.4 Постановка задачи 15](#_Toc165050062)

[ГЛАВА 2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ 17](#_Toc165050063)

[2.1 Требования к программе 17](#_Toc165050064)

[2.2 Руководство пользователя 18](#_Toc165050065)

[2.3 Руководство программиста 27](#_Toc165050066)

[ЗАКЛЮЧЕНИЕ 31](#_Toc165050067)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 32](#_Toc165050068)

[ПРИЛОЖЕНИЕ А 33](#_Toc165050069)

# ВВЕДЕНИЕ

В настоящее время автоматизация затрагивает все более широкие области. Новейшие компьютерные программы помогают выполнять большую часть работы, эффективно управляя основными и вспомогательными производственными процессами.

Создание специальной информационной системы деятельности администратора гостиницы является актуальным и важным в современном мире гостиничного бизнеса.

Информационные системы в гостиничном бизнесе помогают управлять бронированием номеров, вести учет о гостях, управлять данными о персонале гостиницы. Также информационная система может вести учет финансовых операций, включая оплату номеров, услуг, налогов и других расходов или доходов.

Целью курсовой работы будет являться разработка информационной системы деятельности администратора гостиницы.

Для реализации поставленной цели, необходимо решить следующие задачи:

1. Изучить и описать предметную область.
2. Определить основные бизнес – процессы.
3. Автоматизировать процесс управления гостиничным бизнесом, обеспечивающий:

* возможность быстрого и удобного бронирования номеров;
* учет гостей и персонала;
* финансовый учет и отчетность;
* улучшение клиентского опыта;
* безопасность и конфиденциальность информации о клиентах.

Объектом исследования является гостиничное дело.

Предметом исследования являются процесс разработки информационной системы деятельности администратора гостиницы.

Для решения поставленных задач использовались методы: анализ (при описании предметной области); синтез (при реализации информационной системы); формализации (при проектировании информационной системы).

Разрабатываемая информационная система деятельности администратора гостиницы позволит координировать потоки информации, ускорить процессы её обработки, облегчит ввод и корректирование обрабатываемой информации, повысит производительность труда администратора гостиницы.

В первой главе курсовой работы исследуются теоретические основы и основные принципы автоматизации бизнес-процессов гостиничного бизнеса, а также постановка задач к реализации информационной системы.

Во второй главе выполняется практическая реализация разработки информационной системы деятельности администратора гостиницы.

В приложении представлен листинг программы.

# ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ И ОБЩАЯ ПОСТАНОВКА ЗАДАЧ

## 1.1 История и развитие гостиничного бизнеса

Понятие гостеприимство появилось еще на заре человеческой цивилизации. Веками совершенствуясь и обретая новые функции, понятие гостеприимство дошло и до наших дней, превратившись в сильную и значительную отрасль, приносящую существенный доход, как владельцам, так и государству. Сегодня гостиничный бизнес не только способствует экономическому росту отдельных государств, но и играет важную роль в улучшении быта людей, становящихся все более мобильными в условиях глобализации.

Само понятие «гостиница» происходит от латинского эквивалента этому слову «hostel», а также англосаксонского «hospitality», что значит гостеприимство (от старофранцузского «хоспис», что значит странно-приемный дом). Позднее данное название модернизировалось уже в более привычное нам, имеющее французские корни-«hotel».

Появление первых прообразов гостиниц, как и самой профессии по обслуживанию людей, останавливающихся на ночлег, уходит своими корнями в далекое прошлое. Гостиные дома располагались в городах и на дорогах, ведущих из одних стран в другие. Помимо путешественников услугами гостиных домов пользовались гонцы, курьеры, правительственные служащие. Здесь людям предоставляли ночлег, убежище, пищу, там же кормили и меняли лошадей.

В средние века в Европе постоялые дворы начинали создаваться при монастырях. Церковь обязывала организовывать «госпиции» для путешественников, паломников, священников, путешествующих по святым местам. Со временем из бесплатного приюта они становятся предприятиями, рассчитанными на получение дохода.

Прообразы гостиниц имелись и на Ближнем Востоке, и в Средней Азии, и в Закавказье. Купцы с караванами товаров путешествовали по пустыням и предгорьям. Ночевали они обычно в шатрах, но иногда останавливались в караван-сараях – своеобразных гостиничных комплексах, включавших в себя загон для верблюдов и помещения для ночлега людей, окруженные крепостной стеной.

В 18-19 веках с ростом экономических и политических связей между государствами начинается бурное развитие гостиничного хозяйства, особенно в городах Европы. Гостиничное дело превращается в важную отрасль, приносящую большую прибыль.

Развитие предприятий гостеприимства 19 века связано с развитием туризма. На морских побережьях, возле источников с минеральной водой, в живописных местах развертывается строительство крупных и мелких гостиниц. Постепенно совершенствуется их техническое оборудование, создаются комфортабельные условия для гостей, меняются формы и методы обслуживания.

В этой области экономики возникают компании, акционерные общества, корпорации и синдикаты. Такие крупные объединения стали руководить гостиничным хозяйством своих стран, а также строить гостиницы в других государствах. В Лондоне создается синдикат гостиниц, во Франции – «Союз хозяев гостиниц». Эти организации частных владельцев устанавливали цены на номера, готовили кадры гостиничных работников, способствовали развитию туризма. В 1906 году был создан «Международный союз владельцев гостиниц», объединивший владельцев 1700 гостиниц в различных странах мира.

В крупных европейских городах гостиницы начинают использовать и для других целей. В них устраивают казино, проводят пресс-конференции, приемы.

Бурное развитие гостиничного хозяйства продолжалось в 20 веке. Этому способствовало резкое качественное и количественное увеличение автомобильного, авиационного и железнодорожного транспорта, оживление торговых, культурных, научно технических и спортивных контактов между городами.

На Руси постоялые дворы, предшественники первых гостиниц, появляются в 12-13 веке. В них отдыхали и меняли лошадей. Эти постоялые дворы – «ямы», как их называли, располагались один от другого на расстоянии конного перехода.

В 15 веке создаются многочисленные почтовые станции, которыми ведает Ямской приказ. По велению Ямского приказа открывались новые станции, ему же подчинялись и ямщики. В это же время строятся многочисленные гостиные и постоялые дворы. В гостиных дворах не только торговали, в них жили и совершали коммерческие операции.

В связи с ростом промышленного производства и расширением торговых связей в 18-19 веке растет население городов, открываются новые гостиницы. В 1818 году в Москве функционирует 7 гостиниц. В Петербурге в 1900 году насчитывалось уже 325 гостиниц. В 1910 году в России было 4685 гостиниц, не считая постоялых дворов и трактиров с номерами. Все они принадлежали частным лицам и были чисто коммерческими предприятиями.

После Октябрьской революции декретом Советского правительства все гостиницы были национализированы, гостиничное хозяйство подверглось коренной перестройке. К 1940 году гостиницы были построены в 669 городах. В годы Великой Отечественной войны был нанесен огромный ущерб всему народному хозяйству, в том числе и гостиничному. В послевоенные годы развернулись большие работы по восстановлению, реконструкции и строительству новых гостиниц. Уже к 1960 году в 1364 городах Советского Союза гостей обслуживали в 1476 гостиницах.

Дальнейший рост материальной и технической гостиничной базы в стране определили следующие факторы: развитие существующих городов и появление новых; рост промышленности, науки, культуры и искусства; увеличение материального благосостояния людей. Все это создало предпосылки для развития внутреннего туризма, обмена делегациями, увеличения числа командированных и отпускников. В 1980 году в канун московской Олимпиады, гостиничное хозяйство СССР насчитывало 7000 гостиниц общей вместимостью 700 тысяч мест. Было построено много крупных, комфортабельных гостиниц.

Достигнув определенных высот и занимая одну из важных ролей в экономике и повседневной жизни людей, гостиничный бизнес не стоит на месте. Ежегодно открываются новые отели, идет реконструкция старых, обновляются номера, совершенствуются стандарты обслуживания. По данным Федеральной службы государственной статистики на конец 2008 года в Санкт-Петербурге насчитывалось 110 гостиниц, в то время как в 2005 году эта цифра равнялась 72 объектам.

Количество мини-гостиниц на конец 2007 года по данным Центра развития малых отелей составило 500 объектов, тогда как в 2003 году их было всего 25. Эти цифры довольно красноречиво говорят о том, что рынок гостиничных услуг продолжает развиваться и при чем довольно активно. Мировой экономический кризис немного притормозил строительство многих крупных сетевых объектов в крупных городах России. Но, по мнению многих отельеров, это дает мини-отелям шанс использовать свой потенциал. Поэтому в ближайшие годы прогнозируется рост мини-отелей в городах России и увеличение роли этого типа гостиничных предприятий в гостиничной отрасли.

За все время истории и развития гостиничного дела удалось выявить наиболее эффективные виды организационных структур, которые позволяют продуктивно функционировать гостинице, о чем будет сказано ниже.

## 1.2 Организационная структура гостиницы

Организационная структура гостиницы определяется в первую очередь масштабами и назначением гостиницы, ее местоположением, спецификой гостей, используемыми технологиями и другими факторами. Она является отражением полномочий и обязанностей, возложенных на каждого работника. Эффективная организация работы нужна каждой гостинице. Единой, идеальной для всех структуры управления гостиницей структуры не существует. На старте бизнеса и маленьких объектах она может быть сильно сокращена и сведена до нескольких человек-генеральный директор, он же бухгалтер, администраторы, горничная и ремонтник по договору. В крупных и сетевых гостиницах штат структурируется и разделяется на сегменты.

Но в любом случае выстраивание организационной структуры преследует одну цель-удовлетворение потребностей клиентов при лучшей управляемости, меньших финансовых и трудозатратах. Качество обслуживания гостей напрямую зависит от грамотно выстроенной организационной структуры управления отелем. Попытки свести количество персонала к минимуму, чтобы оптимизировать расходы, могут привести к ухудшению сервиса, потере репутации и прибыли.

Организационная структур любой организации будет зависеть от того, кто является ее участниками, какие задачи она решает и как далеко организация зашла в своем развитии. Независимо от того, какую организационную структуру вы выбираете, три элемента всегда будут присутствовать в ней:

1. Управление: конкретный человек или группа людей, которые принимают решения в организации.
2. Правила, по которым работает организация: многие из этих правил могут быть заявлены явно, в то время как другие могут быть скрытыми, но при этом не менее обязательными для исполнения.
3. Распределение труда: распределение труда может быть формальным или неформальным, временным или постоянным, но в каждой организации непременно будет определены место в ней и конкретные обязанности.

По сути, любая организационная структура-это иерархия управления и делегирования полномочий. Для каждого сотрудника должны быть четко определены место в ней и конкретные обязанности.

На практике используются три вида классических систем управления:

1. Линейная структура: самая простая структура, при которой персонал конкретного подразделения подчиняется единственному человеку-линейному руководителю. Команды спускаются по одному каналу, что исключает противоречивость распоряжений, позволяет быстро ставить задачу и контролировать ее выполнение. Каждый линейный менеджер компетентен во всех вопросах своей сферы. Например, руководитель службы питания знает в деталях ресторанное дело, логистику и хранение продуктов, основы бухучета. Такая структура часто применяется в мини-отелях и небольших гостиницах эконом-класса, которые предлагают небольшой набор услуг.
2. Функциональная структура: система с двумя уровнями управления, в которой каждое подразделение специализируется на конкретных бизнес-процессах. Например, организация приема и размещения, питания, кадрового обеспечения, финансового учета. В качестве главного координатора выступает старший менеджер. За процессы в каждом сегменте отвечает функциональный руководитель, который профессионально разбирается только в своей области. Это повышает эффективность работы конкретного подразделения, но может снизить скорость решения задач в целом из-за слабого взаимодействия отделов между собой. Такая структура традиционна для средних и крупных гостиниц, с широким спектром дополнительных услуг.
3. Линейно-функциональная структура: комбинация двух систем управления, которая объединяет их преимущества. У линейных руководителей внутри подразделения присутствуют функциональные сегменты. Например, у директора номерного фонда есть менеджеры службы бронирования, приема и размещения, эксплуатации, расчетной группы. Каждый функциональный руководитель отвечает за результат работы в целом. Это улучшает контроль и эффективность деятельности, но увеличивает численность штата и расходы на зарплату. Подобная структура характерна для крупных независимых и сетевых отелей, который могут позволить себе несколько слоев управления. Любая организационная система управления гостиницей должна быть гибкой. Это дает возможность манипулировать штатом, сокращать или расширять его при необходимости.

Основные функциональные подразделения и система менеджмента:

1. Руководители высшего звена или управляющий персонал: линейные и функциональные руководители-директора служб и менеджеры решают вопросы на конкретном участке, координируют непосредственных исполнителей и контролируют качество исполнения работ.
2. Низший персонал: слабое звено в бизнес-схеме из-за уровня ответственности, добросовестности, собранности, грубых ошибок и отсутствия профессиональной ориентации. Его нужно обучать, проверять, поддерживать и мотивировать.

Службы гостиницы по степени контакта персонала гостиницы с гостями делятся на две основные группы: 1) Front of the house-Фронт-офис-служащие напрямую контактируют с клиентами; 2) Back of the house-Бэк-офис-служащие не контактируют с клиентами. В гостиничной индустрии подобное разграничение является очень важным, так как влияет на требования, предъявляемые к персоналу. Обучение сотрудников проводится по разным программам, их служебные помещения четко разделены.

Отличается и форменная одежда. После изучения основных видов организационных структур необходимо перейти к функциям, которые выполняет администратор гостиницы.

## 1.3 Функции, выполняемые администратором гостиницы

В современном мире, где туризм и гостеприимство играют важную роль в экономике многих стран, профессия администратора гостиницы стоит на передовой обслуживания клиентов в индустрии. Это не просто лицо, которое встречает гостей при входе в отель, но и ключевой специалист, обеспечивающий эффективное управления и качество. От их профессионализма и умения общаться с людьми зависит впечатление клиентов о пребывании в номерах, что в свою очередь оказывает прямое влияние на репутацию и успешность гостиничного бизнеса.

Профессия администратора гостиницы имеет долгую и разнообразную историю, тесно связанную с развитием индустрии гостеприимства. Самые ранние формы этого бизнеса возникли в древних цивилизациях, таких как Римская империя и Китай, где путешественникам предлагались места для отдыха и ночлега. Однако концепция администратора гостиницы, как мы её знаем сегодня, начала формироваться в средние века в Европе, с ростом торговли и паломничества. Именно тогда начали появляться первые гостиницы и таверны.

В эпоху Возрождения и во время промышленной революции, когда путешествия стали более доступными, роль администратора стала более сложной и многофункциональной. Это был период, когда отели начали предлагать не только ночлег, но и различные услуги. Организация питания и развлечений потребовало более квалифицированного управления.

В 20-м веке, с развитием транспорта и туризма, гостиничный бизнес и роль администратора претерпели значительные изменения. Возросла потребность в профессионализме, внедрении технологий и умении управлять потребностями гостей. Современные администраторы хостелов, гостиниц и отелей являются не только хранителями ключей, но и менеджерами, умеющими обеспечивать высококачественное обслуживание клиентов, управлять персоналом и поддерживать финансовое здоровье учреждения.

Виды деятельности:

1. Дежурный администратор гостиницы-это ключевая фигура в ежедневной работе учреждения. Он отвечает за общее управление отеля в свою смену, обеспечивает качественное обслуживание гостей, решает возникающие проблемы и координирует работу другого персонала. Также дежурный администратор контролирует процессы бронирования и заселения, поддерживает порядок и безопасность.

2. Администратор горничных в гостинице отвечает за управление персоналом уборщиц и обеспечение высокого уровня чистоты и порядка в номерах и общественных зонах. Он контролирует качество уборки, распределяет задачи между сотрудниками, следит за наличием необходимых расходных материалов и обеспечивает соблюдение стандартов гигиены.

3. Администратор хостела занимается управлением недорогими и часто более неформальным жильем для путешественников. Ключевые обязанности включают регистрацию гостей, управление бронированиями, обеспечение общего порядка и комфорта. В хостелах администратор часто взаимодействует непосредственно с гостями, помогает с информацией о местности и создает дружелюбную атмосферу.

4. Администратор гостиницы и сауны управляет заведением, которое предлагает дополнительные услуги. Он отвечает за обеспечение качественного обслуживания в этих зонах, контролирует соблюдение санитарных и технических норм, а также занимается организацией бронирования и предоставлением соответствующих услуг.

5. Ночной администратор гостиницы выполняет многие из тех же функций, что и дневной, но во вторую смену. Он контролирует процесс регистрации и выписки гостей, решает возникающие вопросы и проблемы, а также обеспечивает безопасность и спокойствие в номерах в ночное время.

6. Главный или старший администратор отвечает за общее управление персоналом. Он разрабатывает политику обслуживания, контролирует финансовые показатели и координирует работу различных отделов. Старший администратор, как правило, занимается более конкретными задачами и подчиняется главному администратору.

7. Администратор гостиницы на ресепшене-это первое лицо, с которым встречаются гости гостиницы. Он отвечает за регистрацию, информирование их об услугах, решение вопросов, связанных с проживанием, а также обеспечение общего комфорта во время их пребывания.

8. Администратор-кассир сочетает обе эти функции. Он занимается не только обслуживанием гостей, но и финансовыми операциями, такими как прием платежей, выставление счетов и управление кассовым аппаратом.

Администратор гостиницы играет ключевую роль в управлении отелем и обеспечении высокого уровня обслуживания клиентов. Его обязанности, принципы работы, функции и задачи охватывают широкий спектр деятельности.

1. Регистрация гостей: встреча при заезде, регистрация и распределение номеров. Предоставление информации о доступных услугах, правилах пребывания и особенностях гостиницы.
2. Обслуживание клиентов: ответы на запросы и жалобы постояльцев, предложение решений для улучшения их пребывания. Организация дополнительных услуг, таких как трансфер, бронирование билетов, заказ еды и прочее.
3. Управление бронированиями: контроль за системой заполнения номеров. Обработка онлайн и телефонных запросов, изменений и отмен.
4. Финансовые операции: обработка платежей, выставление счетов и управление кассовым аппаратом. Контроль за транзакциями и составление отчетов.
5. Управление персоналом: координация работы сотрудников гостиницы (горничные, охрана, технический персонал). Обеспечение эффективного взаимодействия между различными отделами.
6. Поддержание порядка и безопасности: контроль за соблюдением правил и порядка в гостинице. Взаимодействие с органами власти при необходимости.

Исходя из описанных выше функций, выполняемых администратором гостинице можно перейти к постановке задачи для разработки информационной системы.

## 1.4 Постановка задачи

Общая задача для курсовой работы предполагает разработку информационной системы для администратора гостиницы, которая будет обладать следующим функционалом:

1. Учет номеров.
2. Учет прибывших клиентов со следующей информацией о них: ФИО, номер паспорта, дата приезда, город откуда приехал клиент, номер который клиент заселил.
3. Учет служащих.
4. Финансовые операции.
5. Различные виды сортировки информации и ее удаление из базы данных при необходимости.

Задачи работы охватывают:

1. Проведение исследования потребностей рынка и определение функциональных требований к информационной системе, исходя из специфики процессов обмена недвижимости.

2. Определение оптимальных технологий и инструментов разработки, включая выбор языка программирования и базы данных.

3. Написание программного кода для реализации ключевых функций системы, включая регистрацию и учет клиентов.

4. Организация тестирования системы для выявления и исправления ошибок, обеспечение ее надежности и удобства использования.

5. Оценка эффективности работы системы информационной системы на основе тестовых сценариев и отзывов потенциальных пользователей, разработка рекомендаций по дальнейшему развитию и совершенствованию функционала.

В итоге работы предполагается получение полностью функциональной информационной системы администратора гостиницы, которая будет способствовать оптимизации процесса работа администратора, повышению уровня обслуживания клиентов.

# ГЛАВА 2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## Требования к программе

Для корректного функционирования программы необходимо как минимум 120 Мб свободного места на жестком диске. Операционная система должна быть не младше Microsoft Windows 7, включая Windows 8 и Windows 10, поддерживающая 64-битные версии. Программа должна стабильно работать при длительной эксплуатации компьютеров. В случае сбоев в работе аппаратной части необходимо обеспечить возможность восстановления функциональности программы без потери данных, в том числе и их автоматическое резервное копирование.

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

* Процессор: AMD 3020е с тактовой частотой до 2.6 ГГц.
* Оперативная память: 8 ГБ LPDDR4.
* Видеокарта: Интегрированная AMD Radeon Graphics Vega 3.
* Операционная система: Windows 10 Pro 64-битная версия.
* Накопитель: SSD на 256 ГБ.
* Дисплей: 14-дюймовый Full HD (1920 x 1080).

Эта конфигурация обеспечила разработчикам необходимую производительность для эффективной разработки, тестирования и дебаггинга программного обеспечения.

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 1.8 ГГц;
* ОЗУ: не менее 4 ГБ;
* Свободное место на жестком диске: не менее 1 ГБ;
* Минимальное разрешение экрана: 1024 x 768 пикселей.

Такие требования к аппаратной и программной части обеспечивают комфортное и бесперебойное использование программы, а также позволяют максимально использовать её функциональные возможности.

## 2.2 Руководство пользователя

Руководство пользователю по использованию консольного приложения информационной системы по управлению гостиницей предназначено для ознакомления пользователя с функциональными возможностями приложения, поможет научится использовать его.

Руководство пользователя обеспечивает полную информативность по структуре интерфейса приложения, описывает все его доступные функции.

Запуск приложения

Чтобы запустить приложение, нужно дважды кликнуть левой кнопкой мыши (ЛКМ) по исполняемому файлу — файл с расширением .exe. Данное действие показано на рисунке 1.

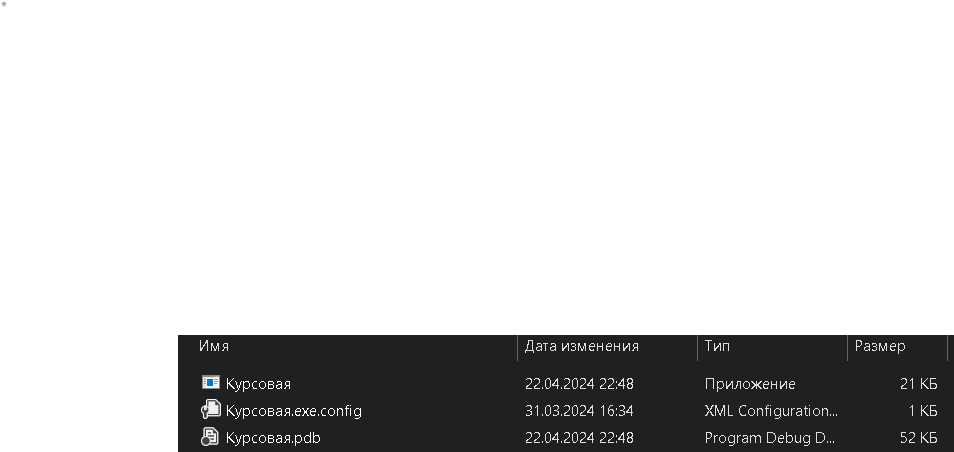


Рис. 1 Запуск приложения

Первое меню

После запуска приложения откроется первое меню, его можно увидеть на рисунке 2, на нем располагается функционал для переключения, реализованный при помощи клавиш. Нажав на клавишу на клавиатуре возможен переход на действие, которое необходимо пользователю.

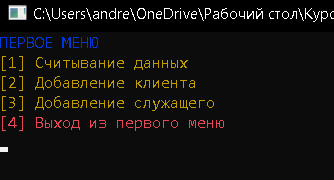


Рис. 2 Первое меню программы

Считывание данных

После перехода на первый пункт, который осуществляет считывание данных из файла в программу пользователь может увидеть надпись в случае успешного считывания данных, и в случае неудачного считывания данных как показано на рисунке 3.

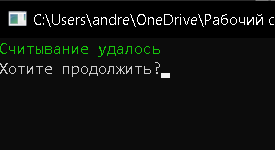


Рис. 3 Успешное считывание данных

Добавление клиента

Если пользователю необходимо добавить нового клиента в базу данных ему необходимо перейти на второй пункт и ввести все необходимые данные, данное действие показано на рисунке 4.

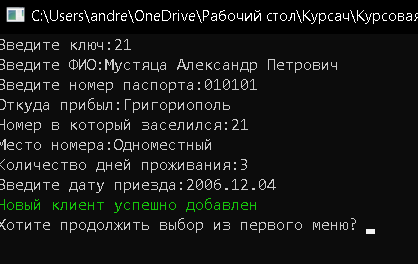


Рис. 4 Добавление клиента

Добавление служащего

После перехода на этот пункт пользователь может добавить нового служащего в файл. Данное действие показано на рисунке 5.

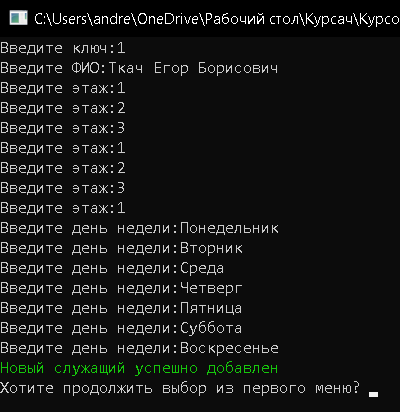


Рис. 5 Добавление служащего

Выход из первого меню

В случае если пользователь произвел все необходимые действия он может перейти в главное меню, что показано на рисунке 6.

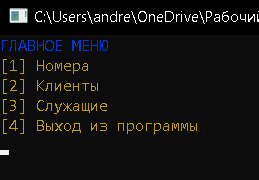


Рис. 6 Переход в главное меню

Главное меню

В главном меню пользователь может выбрать необходимые ему разделы: номера, клиенты, служащие.

Раздел Номера

Перейдя из главного меню на этот раздел, пользователь может увидеть несколько пунктов, которые привязаны к клавиатуре (рис. 7.1).

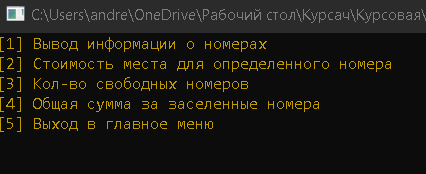


Рис. 7.1 Раздел номер

Вывод информации о номерах

Перейдя по этому пункту, пользователь может вывести на экран информацию обо всех номерах в гостинице (рис. 7.2).

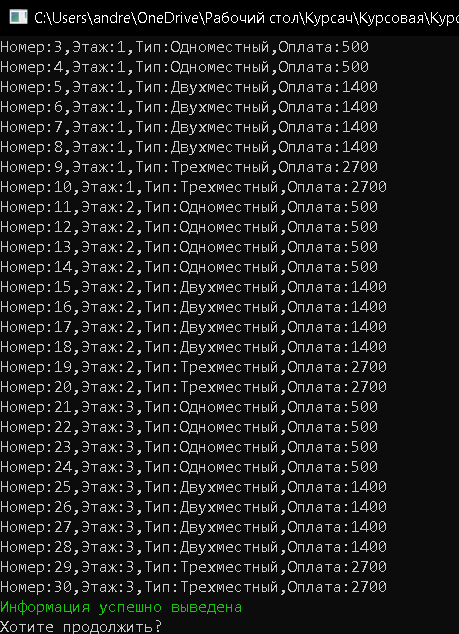


Рис. 7.2 Вывод информации о номерах

Стоимость определенного номера

Этот пункт может понадобиться пользователю в том случае, если необходимо узнать стоимость одной ночи в определенном номере для клиента, который хочет заселиться (рис 7.3).

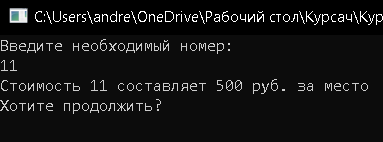


Рис. 7.3 Стоимость определенного номера

Количество свободных номеров

Нажав на клавиатуре клавишу 3, администратор сможет увидеть информацию о незаселенных номерах (рис 7.4).

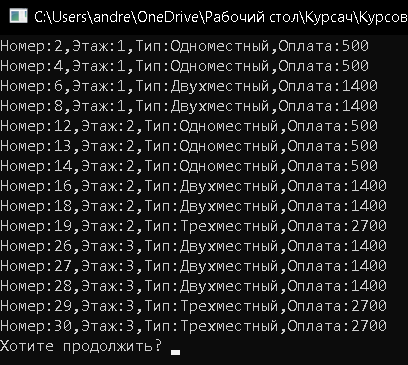


Рис. 7.4 Количество свободных номеров

Сумма, которую выплачивают за заселенные номера

Перейдя на данный пункт можно увидеть надпись, которая сообщает о сумме, которую оплатили все клиенты за оплату своих номеров (рис. 7.5).

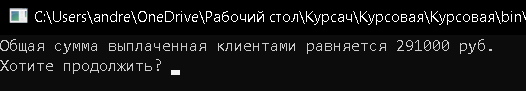


Рис. 7.5 Сумма, выплаченная клиентами

Выход в главное меню

Данный пункт реализует возвращение пользователя в главное меню для дальнейшего выбора.

Раздел Клиенты

В данном разделе пункты привязаны к клавишам нажимая на которые, пользователь переходит на различные пункты (рис. 8.1).

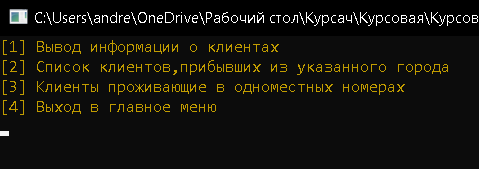


Рис. 8.1 Раздел Клиенты

Вывод информации о клиентах

Переход по этому пункту позволяет администратору увидеть полную информацию о клиентах (рис. 8.2).

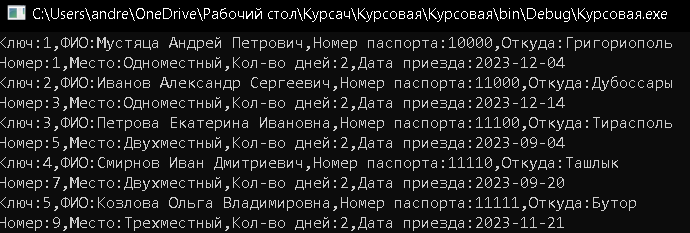


Рис. 8.2 Вывод информации о клиентах

Список клиентов, прибывших из указанного номера

Переход по данному пункту позволяет вывести информацию о клиентах, прибывших из введенного с клавиатуры города. В случае если клиентов из введенного города нет выводится соответствующая надпись (рис. 8.3).

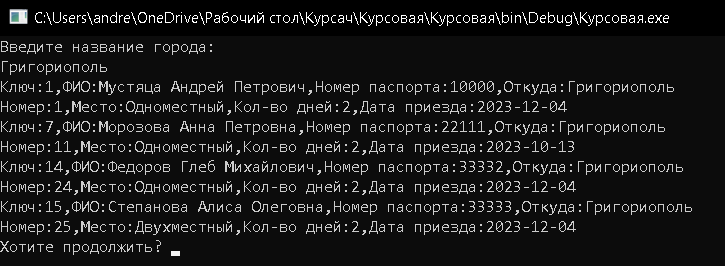


Рис. 8.3 Клиенты, прибывшие из определенного города

Клиенты, проживающие в одноместных номерах

Данный пункт необходим, если пользователю необходимо вывести список клиентов, проживающих в одноместных номерах (рис. 8.4).

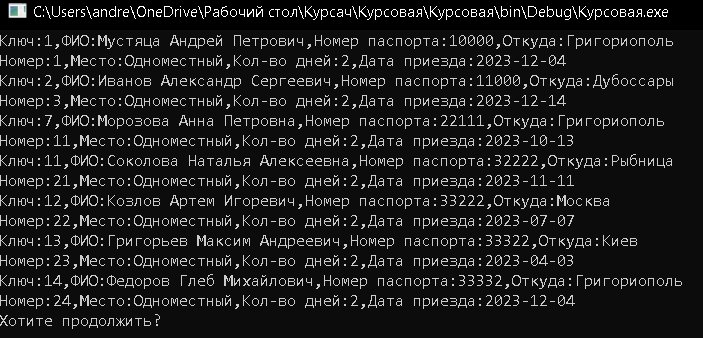


Рис. 8.4 Клиенты, проживающие в одноместных номерах

Выход в главное меню

Данный пункт реализует возвращение пользователя в главное меню для дальнейшего выбора.

Раздел Служащие

В данном разделе пользователь может выбрать действия, которые администратор может выполнять со служащими.

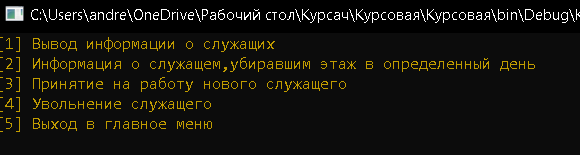


Рис. 9.1 Раздел Служащие

Вывод информации о служащих

Данный пункт позволяет вывести информацию о служащих, которые работают в гостинице (рис. 9.2).

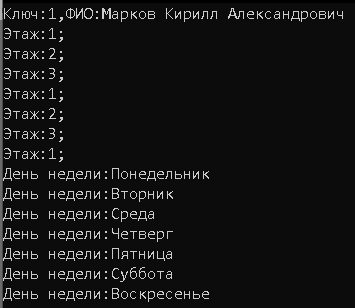


Рис. 9.2 Вывод информации о служащих

Информация о служащем, убиравшим этаж в определенный день недели

Данный пункт поможет пользователю выяснить какой именно служащий убирал определенный этаж в определенный день недели (рис. 9.3).

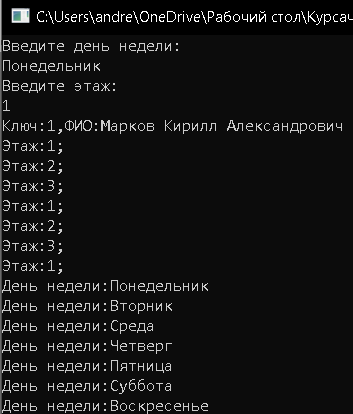


Рис. 9.3 Поиск служащего

Принятие на работу служащего

Перейдя по данному пункту администратор гостиницы может принять на работу нового служащего, введя его данные (рис 9.4).

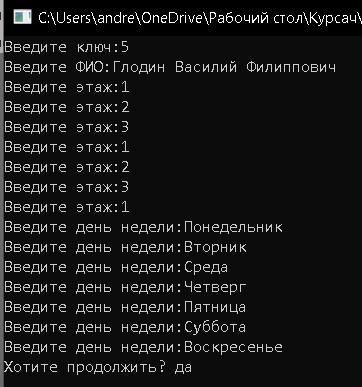


Рис. 9.4 Добавление нового служащего

Увольнение служащего

Переход по данному пункту позволяет в случае необходимости уволить служащего, что будет также отражено в базе данных (рис. 9.5).



Рис. 9.5 Увольнение служащего

Выход из программы

Данный пункт завершает программу.

Это основное руководство пользователю, которое поможет ознакомиться с функциональностью и использованием приложения

## Руководство программиста

Представленный программный продукт представляет собой консольное приложение. В разработке программного продукта реализовано множество алгоритмов.

Структура кода включает в себя:

1. Класс **Hotel** представляет собой класс, описывающий номера в гостинице. Данный класс содержит в себе следующие свойства:

* Number: свойство типа int, обозначающее номер отеля.
* Floor: свойство типа int, обозначающее этаж отеля.
* Type: свойство типа string, обозначающее тип номера (одноместный, двухместный или трехместный).
* Cost\_per\_day: свойство типа int, отвечающее за стоимость одной ночи в номере определенного типа.

Также в данном классе реализованы два поля:

* List1: поле, представляющее собой коллекцию объектов класса Client.
* List2: поле, представляющее собой коллекцию объектов класса Employee.

Также в классе **Hotel** представлены следующие методы:

* public Hotel: конструктор, позволяющий инициализировать все свойства и поля класса Hotel.
* public void SaveHotel: метод записи информации о номере в файл.
* public static Hotel ReaderNumbers: метод позволяющий считать с файла строку и добавить ее в коллекцию в качестве объекта класса Hotel.
* public void Print: метод вывода информации о номерах гостиницы.
* public int Cost: метод, вычисляющий стоимость одной ночи в номере определенного типа.
* public void Empty\_number: метод, позволяющий выяснить какой из номеров является не заселенным.

1. Класс **Client** представляет собой класс, описывающий клиентов гостиницы. Данный класс содержит следующие свойства:

* Key: свойство типа int, обозначающее ключ клиента.
* FIO: свойство типа string, обозначающее ФИО клиента.
* Number\_pasport: свойство типа int, обозначающее номер паспорта клиента.
* Number: свойство типа int, обозначающее номер в который клиент заселен.
* Place: свойство типа string, обозначающее тип номера в который клиент заселен.
* Kol\_vo\_days: свойство типа int, обозначающее количество дней, которые клиент планирует прожить в номере.
* Date: свойство типа DateTime, обозначающее дату приезда клиента.

Также в классе **Client** представлены следующие методы:

* public Client: конструктор, позволяющий инициализировать все свойства класса.
* public void SaveClients: метод для записи информации о клиенте в файл.
* public static Client ReaderClient: метод для считывания строки с файла и ее возвращения в виде объекта класса Client.
* public void PrintClient: метод для вывода информации о клиентах.
* public static void City: метод позволяющий выводить список клиентов, прибывших из определенного города.
* public void Single: метод позволяющий вывести выводить список клиентов, живущих в одноместных номерах.

1. Класс **Employee** представляет собой класс, описывающий работников гостиницы. Данный класс содержит следующие свойства и поля:

* Key: свойство типа int, представляющее собой ключ работника.
* FIO: свойство типа string, описывающее ФИО работника.
* floors: поле представляющее собой коллекцию типа int, описывающую этажи, которые убирает работник.
* days: поле представляющее собой коллекцию типа string, описывающую дни недели в которые работник работает.

Также в классе **Employee** представлены следующие методы:

* public Employee: конструктор, позволяющий инициализировать свойства класса.
* public void SaveEmployees: метод записи в файл.
* public static Employee ReaderEmployee: метод позволяющий считывать строку с файла и возвращать ее в качестве объекта класса **Employee**.
* public void PrintEmployee: метод, позволяющий вывести информацию о работнике.
* public void Cleaning: метод поиска работника, убирающего определенный этаж в определенный день недели.
* Public static void Delete: метод удаления (увольнения) работника.

Во время разработки были использованы следующие библиотеки объектно-ориентированного языка программирования С#: System, Sytem.Console, System.IO, System.Collections.Generic, System.Linq.

# ЗАКЛЮЧЕНИЕ

В данной курсовой работе была разработана информационная система для администратора гостиницы. Программа реализована на языке программирования C# и в качестве хранилища данных используется три файла.

Пользовательский интерфейс создан с применением консольных средств ввода-вывода, что обеспечивает простоту навигации и понятные команды для управления программой. Интерфейс позволяет удобно взаимодействовать с системой, вводить, осуществлять поиск и выводить информацию.

Информация о клиентах, номерах и служащих разделена и находится отдельно в трех текстовых файлах.

Основные преимущества разработанной системы включают в себя:

1. Интуитивно понятный и легко осваиваемый интерфейс обеспечивает комфортное взаимодействие пользователя с системой.

В процессе работы были выявлены некоторые области, где можно улучшить систему:

1. Расширение функциональности системы. Например, бронирование номеров.
2. Улучшение интерфейса консольного приложения для более удобного взаимодействия пользователя с системой, или же замена интерфейса на более презентабельный вид.
3. Интеграция с внешними сервисами или базами данных для получения дополнительной информации о клиентах, служащих, такой как номер телефона, страховой полис и т.д.

Цель работы была успешно достигнута - разработана информационная система, позволяющая управлять процессом управления гостиницей.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Албахари Дж., Албахари Б.: C# 8.0. Карманный справочник - Москва: ООО «И. Д. Вильямс», 2020. – 240 с.
2. Антон Марченко: Основы программирования на С# 2.0. Учебное пособие – 2017. – 557 с.
3. Шилдт, Г. C# 4.0: полное руководство. - Москва: ООО «И. Д. Вильямс», 2015. - 1056 с.
4. Лекция «Службы гостиницы»

(<https://ru.scribd.com/document/603037793/%D0%9B%D0%B5%D0%BA%D1%86%D0%B8%D1%8F-%D0%A1%D0%BB%D1%83%D0%B6%D0%B1%D1%8B-%D0%B3%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B8%D1%86%D1%8B>)

1. «История гостиничного дела»

(https://hoteladvisor.livejournal.com/32450.html)

# ПРИЛОЖЕНИЕ А

class Employee

{

/// <summary>

/// Свойства

/// </summary>

public int Key { set; get; }

public string FIO { set; get; }

public List<int> floors = new List<int>(7);

public List<string> days = new List<string>(7);

/// <summary>

/// Конструктор класса

/// </summary>

/// <param name="key"></param>

/// <param name="fio"></param>

/// <param name="floors"></param>

/// <param name="days"></param>

public Employee(int key,string fio,List<int>floors,List<string>days)

{

Key = key;

FIO = fio;

this.floors = floors;

this.days = days;

}

/// <summary>

/// Метод добавления

/// </summary>

/// <returns></returns>

public static Employee AddEmployee()

{

Console.Write("Введите ключ:");

int key = Convert.ToInt32(Console.ReadLine());

Console.Write("Введите ФИО:");

string fio = Console.ReadLine();

List<int> floors = new List<int>(7);

for(int i=0;i<floors.Capacity;i++)

{

Console.Write("Введите этаж:");

floors.Add(Convert.ToInt32(Console.ReadLine()));

if(floors[i]>3)

{

Console.Write("Попробуйте еще раз,ошибка:");

loors[i] = Convert.ToInt32(Console.ReadLine());

}

}

List<string> days = new List<string>(7);

for(int i=0;i<days.Capacity;i++)

{

Console.Write("Введите день недели:");

days.Add(Console.ReadLine());

}

return new Employee(key,fio,floors,days);

}

/// <summary>

/// Метод записи в файл

/// </summary>

public void SaveEmployees()

{

using (StreamWriter writer=new StreamWriter(@"C:\Users\andre\OneDrive\Рабочий стол\Курсач\Служащие.txt", true))

{

writer.WriteLine($"{Key};{FIO};{floors[0]};{floors[1]};{floors[2]};" +

$"{floors[3]};{floors[4]};{floors[5]};" +

$"{floors[6]};{days[0]};{days[1]};" +

$"{days[2]};{days[3]};{days[4]};{days[5]};{days[6]};");

}

}

/// <summary>

/// Метод чтения

/// </summary>

/// <param name="line"></param>

/// <returns></returns>

public static Employee ReaderEmployee(string line)

{

string[] parts = line.Split(';');

List<int> floors\_txt = new List<int>();

for(int i=2;i<=8;i++)

{

floors\_txt.Add(int.Parse(parts[i]));

}

List<string> days\_txt = new List<string>();

for(int i=9;i<=15;i++)

{

days\_txt.Add(parts[i]);

}

return new Employee(int.Parse(parts[0]),parts[1],floors\_txt,days\_txt);

}

/// <summary>

/// Метод вывода информации

/// </summary>

public void PrintEmployee()

{

Console.WriteLine($"Ключ:{Key},ФИО:{FIO}");

foreach(var elem in floors)

{

Console.WriteLine($"Этаж:{elem};");

}

foreach(var elem in days)

{

Console.WriteLine($"День недели:{elem}");

}

}

public void Cleaning(string day,int floor)

{

bool cleaner = false;

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

{

if(days[i]==day&&floors[i]==floor)

{

PrintEmployee();

cleaner = true;

}

}

if(cleaner==false)

{

Console.WriteLine("Работник не найден");

}

}

public static void Delete(List<Employee>employees)

{

Console.Write("Введите номер служащего,которого необходимо удалить:");

int delete = Convert.ToInt32(Console.ReadLine());

employees = employees.Where(x=>x.Key!=delete).ToList();

string[] lines = File.ReadAllLines(@"C:\Users\andre\OneDrive\Рабочий стол\Курсач\Служащие.txt");

if(delete-1>=0&&delete-1<lines.Length)

{

lines = lines.Where((line, index) => index != delete - 1).ToArray();

}

File.WriteAllLines(@"C:\Users\andre\OneDrive\Рабочий стол\Курсач\Служащие.txt", lines);

}

}

class Client

{

public int Key { set; get; }

public string FIO { set; get; }

public int Number\_pasport { set; get; }

public string Where { set; get; }

public int Number { set; get; }

public string Place { set; get; }

public int Kol\_vo\_days { set; get; }

public DateTime Date { set; get; }

/// <summary>

/// Конструктор

/// </summary>

/// <param name="key"></param>

/// <param name="fio"></param>

/// <param name="number\_pasport"></param>

/// <param name="where"></param>

/// <param name="number"></param>

/// <param name="place"></param>

/// <param name="kol\_vo\_days"></param>

public Client(int key,string fio,int number\_pasport,string where,int number,string place,int kol\_vo\_days,DateTime date)

{

Key = key;

FIO = fio;

Number\_pasport = number\_pasport;

Where = where;

Number = number;

Place = place;

Kol\_vo\_days = kol\_vo\_days;

Date = date;

}

/// <summary>

/// Добавление объекта

/// </summary>

/// <returns></returns>

public static Client AddClient()

{

Console.Write("Введите ключ:");

int key = Convert.ToInt32(Console.ReadLine());

Console.Write("Введите ФИО:");

string fio = Console.ReadLine();

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

int number\_pasport = Convert.ToInt32(Console.ReadLine());

Console.Write("Откуда прибыл:");

string where = Console.ReadLine();

Console.Write("Номер в который заселился:");

int number = Convert.ToInt32(Console.ReadLine());

Console.Write("Место номера:");

string place = Console.ReadLine();

Console.Write("Количество дней проживания:");

int kol\_vo\_days = Convert.ToInt32(Console.ReadLine());

Console.Write("Введите дату приезда:");

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

return new Client(key,fio,number\_pasport,where,number,place,kol\_vo\_days,date);

}

/// <summary>

/// Запись в файл Клиенты.txt

/// </summary>

public void SaveClients()

{

using (StreamWriter writer=new StreamWriter(@"C:\Users\andre\OneDrive\Рабочий стол\Курсач\Клиенты.txt",true))

{

writer.WriteLine($"{Key};{FIO};{Number\_pasport};{Where};{Number};{Place};{Kol\_vo\_days};{Date.ToString("yyyy-MM-dd")};");

}

}

/// <summary>

/// Считывание с файла Клиенты.txt и запись результатов считывания в коллекцию

/// </summary>

/// <param name="line"></param>

/// <returns></returns>

public static Client ReaderClient(string line)

{

string[] parts = line.Split(';');

return new Client(int.Parse(parts[0]),parts[1],int.Parse(parts[2]),parts[3],int.Parse(parts[4]),parts[5],int.Parse(parts[6]),DateTime.Parse(parts[7]));

}

/// <summary>

/// Вывод информации в консоль

/// </summary>

public void PrintClient()

{

Console.WriteLine($"Ключ:{Key},ФИО:{FIO},Номер паспорта:{Number\_pasport},Откуда:{Where}");

Console.WriteLine($"Номер:{Number},Место:{Place},Кол-во дней:{Kol\_vo\_days},Дата приезда:{Date.ToString("yyyy-MM-dd")}");

}

public static void City(List<Client>clients)

{

Console.WriteLine("Введите название города:");

string cities = Console.ReadLine();

bool citi = false;

foreach(var elem in clients)

{

if(elem.Where==cities)

{

elem.PrintClient();

citi = true;

}

}

if(citi==false)

{

Console.WriteLine("Клиенты из указанного города не найдены");

}

}

public void Single()

{

if(Place=="Одноместный")

{

PrintClient();

}

}

}

class Hotel

{

/// <summary>

/// Свойства класса

/// </summary>

public int Number { set; get; }

public int Floor { set; get; }

public string Type { set; get; }

public int Cost\_per\_day { set; get; }

public List<Client> list1 = new List<Client>();

public List<Employee> list2 = new List<Employee>();

/// <summary>

/// Конструктор класса

/// </summary>

/// <param name="number"></param>

/// <param name="floor"></param>

/// <param name="type"></param>

/// <param name="cost\_per\_day"></param>

/// <param name="clients"></param>

/// <param name="employees"></param>

public Hotel(int number,int floor,string type,int cost\_per\_day,List<Client>clients,List<Employee>employees)

{

Number = number;

Floor = floor;

Type = type;

Cost\_per\_day = cost\_per\_day;

this.list1 = clients;

this.list2 = employees;

}

/// <summary>

/// Метод добавления номера

/// </summary>

/// <param name="clients\_to\_number"></param>

/// <param name="employees"></param>

/// <returns></returns>

public static Hotel AddNumber(List<Client>clients\_to\_number,List<Employee>employees)

{

Console.Write("Номер:");

int number = Convert.ToInt32(Console.ReadLine());

Console.Write("Этаж:");

int floor = Convert.ToInt32(Console.ReadLine());

Console.Write("Тип номера:");

string type = Console.ReadLine();

Console.Write("Стоимость одной ночи:");

int cost\_per\_day = Convert.ToInt32(Console.ReadLine());

List<Client> list1 = new List<Client>();

for(int i=0;i<clients\_to\_number.Count;i++)

{

if(clients\_to\_number[i].Number==number&&clients\_to\_number[i].Place==type)

{

list1.Add(clients\_to\_number[i]);

}

}

List<Employee> list2 = new List<Employee>();

list2 = employees;

return new Hotel(number,floor,type,cost\_per\_day,list1,list2);

}

/// <summary>

/// Метод записи информации о номерах в файл Номера.txt

/// </summary>

public void SaveHotel()

{

using (StreamWriter writer=new StreamWriter(@"C:\Users\andre\OneDrive\Рабочий стол\Курсач\Номера.txt",true))

{

writer.WriteLine($"{Number};{Floor};{Type};{Cost\_per\_day}");

}

}

/// <summary>

/// Чтение и запись в объекты строки файла

/// </summary>

/// <param name="line"></param>

/// <param name="clients"></param>

/// <param name="employees"></param>

/// <returns></returns>

public static Hotel ReaderNumbers(string line,List<Client>clients,List<Employee>employees)

{

string[] parts = line.Split(';');

List<Client> clients\_txt = new List<Client>();

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

{

if(int.Parse(parts[0])==clients[i].Number&&parts[2]==clients[i].Place)

{

clients\_txt.Add(clients[i]);

}

}

List<Employee> employees\_txt = employees;

return new Hotel(int.Parse(parts[0]),int.Parse(parts[1]),parts[2],int.Parse(parts[3]),clients\_txt,employees\_txt);

}

/// <summary>

/// Вывод информации

/// </summary>

public void Print()

{

Console.WriteLine($"Номер:{Number},Этаж:{Floor},Тип:{Type},Оплата:{Cost\_per\_day}");

}

public int Cost()

{

if(Type=="Одноместный")

{

return Cost\_per\_day;

}

else if(Type=="Двухместный")

{

return Cost\_per\_day / 2;

}

else

{

return Cost\_per\_day / 3;

}

}

public void Empty\_number()

{

if(list1.Count==0)

{

Print();

}

}

}

class Program

{

static void Main(string[] args)

{

List<Hotel> list1 = new List<Hotel>();

List<Client> list2 = new List<Client>();

List<Employee> list3 = new List<Employee>();

bool flag = true;

//Добавление и запись

while (flag)

{

Console.Clear();

Console.ForegroundColor = ConsoleColor.DarkBlue;

Console.WriteLine("ПЕРВОЕ МЕНЮ");

Console.ResetColor();

Console.ForegroundColor = ConsoleColor.DarkYellow;

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

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

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

Console.ResetColor();

Console.ForegroundColor = ConsoleColor.Red;

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

Console.ResetColor();

switch (Console.ReadKey(true).KeyChar)

{

case '1':

try

{

Console.Clear();

using (StreamReader reader = new StreamReader(@"C:\Users\andre\OneDrive\Рабочий стол\Курсач\Служащие.txt"))

{

string line;

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

{

list3.Add(Employee.ReaderEmployee(line));

}

}

using (StreamReader reader = new StreamReader(@"C:\Users\andre\OneDrive\Рабочий стол\Курсач\Клиенты.txt"))

{

string line;

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

{

list2.Add(Client.ReaderClient(line));

}

}

using (StreamReader reader = new StreamReader(@"C:\Users\andre\OneDrive\Рабочий стол\Курсач\Номера.txt"))

{

string line;

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

{

list1.Add(Hotel.ReaderNumbers(line, list2, list3));

}

}

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Считывание удалось");

Console.ResetColor();

Console.Write("Хотите продолжить?");

string answer = Console.ReadLine();

if (answer == "Нет" || answer == "нет")

{

flag = false;

}

}

catch

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Считывание не удалось");

}

break;

case '2':

try

{

Console.Clear();

list2.Add(Client.AddClient());

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Новый клиент успешно добавлен");

Console.ResetColor();

Console.Write("Хотите продолжить выбор из первого меню? ");

string answer = Console.ReadLine();

foreach (var elem in list2)

{

elem.SaveClients();

}

if (answer=="Нет"||answer=="нет")

{

flag = false;

}

}

catch

{

Console.ForegroundColor = ConsoleColor.Red;

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

Console.ResetColor();

}

break;

case '3':

try

{

Console.Clear();

list3.Add(Employee.AddEmployee());

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Новый служащий успешно добавлен");

Console.ResetColor();

Console.Write("Хотите продолжить выбор из первого меню? ");

string answer = Console.ReadLine();

foreach (var elem in list3)

{

elem.SaveEmployees();

}

if (answer == "Нет" || answer == "нет")

{

flag = false;

}

}

catch

{

Console.ForegroundColor = ConsoleColor.Red;

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

Console.ResetColor();

}

break;

case '4':

flag = false;

break;

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

break;

}

}

bool flag\_main = true;

while (flag\_main)

{

Console.Clear();

Console.ForegroundColor = ConsoleColor.DarkBlue;

Console.WriteLine("ГЛАВНОЕ МЕНЮ");

Console.ResetColor();

Console.ForegroundColor = ConsoleColor.DarkYellow;

Console.WriteLine("[1] Номера");

Console.WriteLine("[2] Клиенты");

Console.WriteLine("[3] Служащие");

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

Console.ResetColor();

switch (Console.ReadKey(true).KeyChar)

{

case '1':

bool flag\_number = true;

while(flag\_number)

{

Console.Clear();

Console.ForegroundColor = ConsoleColor.DarkYellow;

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

Console.WriteLine("[2] Стоимость места для определенного номера");

Console.WriteLine("[3] Кол-во свободных номеров");

Console.WriteLine("[4] Общая сумма за заселенные номера");

Console.WriteLine("[5] Выход в главное меню");

Console.ResetColor();

switch(Console.ReadKey(true).KeyChar)

{

case '1':

try

{

Console.Clear();

foreach (var elem in list1)

{

elem.Print();

}

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Информация успешно выведена");

Console.ResetColor();

Console.Write("Хотите продолжить? ");

string answer = Console.ReadLine();

if (answer == "Нет" || answer == "нет")

{

flag\_number = false;

}

}

catch

{

Console.WriteLine("Ошибка");

}

break;

case '2':

try

{

Console.Clear();

Console.WriteLine("Введите необходимый номер:");

int n = Convert.ToInt32(Console.ReadLine());

foreach (var elem in list1)

{

if (elem.Number == n)

{

Console.WriteLine($"Стоимость {elem.Number} составляет {elem.Cost()} руб. за место");

}

}

Console.Write("Хотите продолжить? ");

string answer = Console.ReadLine();

if (answer == "Нет" || answer == "нет")

{

flag\_number = false;

}

}

catch

{

Console.WriteLine("Ошибка");

}

break;

case '3':

try

{

Console.Clear();

foreach (var elem in list1)

{

elem.Empty\_number();

}

Console.Write("Хотите продолжить? ");

string answer = Console.ReadLine();

if (answer == "Нет" || answer == "нет")

{

flag\_number = false;

}

}

catch

{

Console.WriteLine("Ошибка");

}

break;

case '4':

try

{

Console.Clear();

int sum = 0;

int sum\_days = 0;

foreach (var elem in list2)

{

sum\_days += elem.Kol\_vo\_days;

}

foreach (var elem in list1)

{

if (elem.list1.Count != 0)

{

sum += elem.Cost();

}

}

Console.WriteLine($"Общая сумма выплаченная клиентами равняется {sum \* sum\_days} руб.");

Console.Write("Хотите продолжить? ");

string answer = Console.ReadLine();

if (answer == "Нет" || answer == "нет")

{

flag\_number = false;

}

}

catch

{

Console.WriteLine("Ошибка");

}

break;

case '5':flag\_number = false;

break;

default:

Console.Clear();

Console.ForegroundColor = ConsoleColor.Red;

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

Console.ResetColor();

break;

}

}

break;

case '2':

bool flag\_client = true;

while(flag\_client)

{

Console.Clear();

Console.ForegroundColor = ConsoleColor.DarkYellow;

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

Console.WriteLine("[2] Список клиентов,прибывших из указанного города");

Console.WriteLine("[3] Клиенты проживающие в одноместных номерах");

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

Console.ResetColor();

switch(Console.ReadKey(true).KeyChar)

{

case '1':

try

{

Console.Clear();

foreach (var elem in list2)

{

elem.PrintClient();

}

Console.Write("Хотите продолжить? ");

string answer = Console.ReadLine();

if (answer == "Нет" || answer == "нет")

{

flag\_client = false;

}

}

catch

{

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

}

break;

case '2':

try

{

Console.Clear();

Client.City(list2);

Console.Write("Хотите продолжить? ");

string answer = Console.ReadLine();

if (answer == "Нет" || answer == "нет")

{

flag\_client = false;

}

}

catch

{

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

}

break;

case '3':

try

{

Console.Clear();

foreach (var elem in list2)

{

elem.Single();

}

Console.Write("Хотите продолжить? ");

string answer = Console.ReadLine();

if (answer == "Нет" || answer == "нет")

{

flag\_client = false;

}

}

catch

{

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

}

break;

case '4':flag\_client = false;

break;

default:

Console.Clear();

Console.ForegroundColor = ConsoleColor.Red;

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

Console.ResetColor();

break;

}

}

foreach (var elem in list3)

{

elem.PrintEmployee();

}

break;

case '3':

bool flag\_employee = true;

while(flag\_employee)

{

Console.Clear();

Console.ForegroundColor = ConsoleColor.DarkYellow;

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

Console.WriteLine("[2] Информация о служащем,убиравшим этаж в определенный день");

Console.WriteLine("[3] Принятие на работу нового служащего");

Console.WriteLine("[4] Увольнение служащего");

Console.WriteLine("[5] Выход в главное меню");

Console.ResetColor();

switch(Console.ReadKey(true).KeyChar)

{

case '1':

try

{

Console.Clear();

foreach(var elem in list3)

{

elem.PrintEmployee();

}

Console.Write("Хотите продолжить? ");

string answer = Console.ReadLine();

if (answer == "Нет" || answer == "нет")

{

flag\_employee = false;

}

}

catch

{

Console.WriteLine("Ошибка");

}

break;

case '2':

try

{

Console.Clear();

Console.WriteLine("Введите день недели:");

string day = Console.ReadLine();

Console.WriteLine("Введите этаж:");

int floor = Convert.ToInt32(Console.ReadLine());

foreach (var elem in list3)

{

elem.Cleaning(day,floor);

}

Console.Write("Хотите продолжить? ");

string answer = Console.ReadLine();

if (answer == "Нет" || answer == "нет")

{

flag\_employee = false;

}

}

catch

{

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

}

break;

case '3':

try

{

Console.Clear();

list3.Add(Employee.AddEmployee());

list3.Last().SaveEmployees();

Console.Write("Хотите продолжить? ");

string answer = Console.ReadLine();

if (answer == "Нет" || answer == "нет")

{

flag\_employee = false;

}

}

catch

{

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

}

break;

case '4':

try

{

Employee.Delete(list3);

Console.Write("Хотите продолжить? ");

string answer = Console.ReadLine();

if (answer == "Нет" || answer == "нет")

{

flag\_employee = false;

}

}

catch

{

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

}

break;

case '5':flag\_employee = false;

break;

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

break;

}

}

break;

case '4':return;

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

break;

}

}

Console.ReadKey();

}

}