Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра экономической информатики

К защите допустить:
Руководитель курсового проекта
ассистент кафедры ЭИ
_____ А.С. Купрейчик

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту
на тему

«АВТОМАТИЗИРОВАННАЯ СИСТЕМА ОЦЕНКИ ОБЪЕКТОВ НЕДВИЖИМОСТИ»

БГУИР КП 6-05-0611-01 102 ПЗ

Студент А. В. Еднач
Руководитель: А. С. Купрейчик

Минск 2024

РЕФЕРАТ

БГУИР КР 6-05-0611-01 102 ПЗ

ТЕМА: Автоматизированная система оценки объектов недвижимости / А.В. Еднач. – Минск : БГУИР, 2024, – п.з. – 52 с., рисунков – 37, источников – 15, приложений – 2.

Ключевые слова: автоматизация, оценка, система оценки, недвижимость.

Объект исследования: оценка объектов недвижимости.

Предмет исследования: приложение для оценки объектов недвижимости.

Цель курсовой работы: автоматизация процесса оценки объектов недвижимости. Основной задачей является повышение эффективности сбора и анализа данных по недвижимости через систематизацию процесса оценки и обработки данных о свойствах объектов.

Методология проведения работы: в процессе разработки системы использованы методы анализа, классификации, обобщения данных, функциональный анализ процессов, а также моделирование системы с использованием UML-диаграмм. В рамках работы была разработана программная и графическая составляющие системы, а также применены современные методы автоматизации и проектирования информационных систем.

Результаты работы: выполнена постановка задачи и определены основные методы её решения; в ходе моделирования системы построены UML-диаграммы, которые отразили структуру и функциональность программы. Были разработаны модели бизнес-процессов предметной области на основе нотации IDEF0 и IDEF1X для описания структуры данных. Также были описаны основные алгоритмы работы программы, создано руководство пользователя и проведено тестирование программного средства, показавшее его соответствие всем функциональным требованиям. Программный продукт разработан на языке C++ с использованием MS Visual Studio 2022.

Область применения результатов: разработанная система может быть применена в различных компаниях, занимающихся оценкой недвижимости, как частных, так и государственных. Она способствует эффективному управлению информацией об объектах, поддерживает процесс оценки и принятия решений на основе расчёта стоимости недвижимости. Разработанное программное средство полностью соответствует функциональным требованиям и может быть использовано для автоматизации процессов оценки недвижимости.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 6
1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ОЦЕНКИ ОБЪЕКТОВ НЕДВИЖИМОСТИ 8
1.1 Описание системы оценки объектов недвижимости 8
1.2 Построение функциональной модели системы оценки объектов недвижимости 10
2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ОЦЕНКИ НЕДВИЖИМОСТИ 14
2.1 Информационная модель системы и ее описание 14
2.2 Модели представления системы и их описание 17
2.2.1 Диаграмма вариантов использования 17
2.2.2 Диаграмма последовательностей 20
2.2.3 Диаграмма классов 22
2.2.4 Диаграмма состояний 24
2.3 Описание алгоритмов, реализующих бизнес-логику системы 25
2.4 Описание созданных программных конструкций 28
3 ОПИСАНИЕ АЛГОРИТМА ЗАПУСКА ПРИЛОЖЕНИЯ, ЕГО ИСПОЛЬЗОВАНИЯ, РЕЗУЛЬТАТЫ РАБОТЫ, ТЕСТИРОВАНИЯ ОБРАБОТКИ ОШИБОК 32
3.1 Алгоритм запуска приложения 32
3.2 Руководство пользователя 32
3.3 Тестирование работы приложения 39
ЗАКЛЮЧЕНИЕ 42
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 43
ПРИЛОЖЕНИЕ А (обязательное) Отчет о проверке на заимствования в системе «Антиплагиат» 44
ПРИЛОЖЕНИЕ Б (обязательное) Листинг кода алгоритмов, реализующих основную бизнес-логику 45

ВВЕДЕНИЕ

Автоматизация процессов оценки объектов недвижимости является важной задачей для современных организаций, занимающихся управлением и продажей недвижимости, а также для частных оценщиков и потенциальных покупателей [1]. Внедрение таких систем позволяет значительно упростить процесс оценки стоимости объектов, повысить точность расчётов и сократить время на проведение анализа. Это особенно актуально в условиях растущего спроса на быструю и точную оценку недвижимости, когда необходимость в автоматизированных средствах анализа возрастает. Такие системы предоставляют возможность оперативного получения данных о стоимости объектов недвижимости, что способствует улучшению процессов управления и продаж недвижимости [2].

Автоматизированные системы оценки недвижимости обеспечивают возможность многократного проведения расчётов с использованием различных критериев (например, площадь, расстояние до метро, стоимость за квадратный метр), что позволяет оценщикам гибко адаптировать методику оценки к различным условиям рынка. Кроме того, такие системы позволяют создавать базу данных объектов недвижимости, которая может использоваться для аналитических целей и прогнозирования рыночных трендов. Это снижает нагрузку на специалистов по оценке и способствует повышению объективности и прозрачности процесса, что в свою очередь помогает поддерживать доверие клиентов. Для пользователей автоматизация позволяет быстро находить и сортировать объекты по ключевым характеристикам, а также получать доступ к объективным оценкам недвижимости [3].

Разработка автоматизированной системы оценки недвижимости с использованием современных технологий и подходов, таких как объектно-ориентированное программирование, является важной задачей в условиях цифровой трансформации бизнеса. Это позволяет интегрировать различные методы расчёта стоимости, обработки данных, а также обеспечивает безопасность и удобство работы с системой для пользователей.

Цель данной курсовой работы — автоматизация оценки стоимости объектов недвижимости. Для достижения этой цели необходимо выполнить следующие задачи:

– изучить и проанализировать бизнес-процесс системы оценки объектов недвижимости визуализировать функциональную модель системы при помощи стандарта IDEF0;

– спроектировать архитектуру системы с использованием стандарта моделирования UML и блок-схем, разработать автоматизированную систему

оценки объектов недвижимости в соответствии с спроектированной архитектурой;

– отладить и протестировать систему для обеспечения корректной работы, описать алгоритм запуска разработанного программного обеспечения, написать легкое в освоении конечным пользователем руководство.

Объектом исследования является автоматизированная система оценки недвижимости. Предметом исследования – процесс разработки и внедрения программного обеспечения для автоматизации оценки объектов недвижимости.

Теоретической и методологической основой данной работы стали научные публикации и практические исследования в области оценки недвижимости и автоматизации бизнес-процессов. А также мой объёмный опыт в области оценки недвижимости и работы с ней.

Курсовой проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности составляет 88,71%. Цитирования обозначены ссылками на публикации, указанные в «Списке использованных источников». Скриншот приведен в приложении А (рисунок А.1).

1 АНАЛИЗ И МОДЕЛИРОВАНИЕ СИСТЕМЫ ОЦЕНКИ ОБЪЕКТОВ НЕДВИЖИМОСТИ

1.1 Описание системы оценки объектов недвижимости

Система автоматизированной оценки недвижимости представляет собой специализированное программное обеспечение, предназначенное для управления информацией об объектах недвижимости, автоматизации процессов их оценки и упрощения взаимодействия с различными категориями пользователей. Основная задача системы заключается в обеспечении эффективного хранения данных об объектах недвижимости, автоматическом расчёте их стоимости, а также в предоставлении инструментов для сортировки и поиска объектов по различным критериям. Программа охватывает широкий спектр задач, связанных с управлением объектами недвижимости, что делает её полезной для различных участников рынка недвижимости, включая пользователей и оценщиков.

Система автоматизирует ключевые процессы, такие как хранение данных об объектах недвижимости и выполнение оценки их стоимости. Алгоритмы оценки разработаны на основе значительного опыта в области оценки объектов недвижимости, что позволяет учесть множество факторов, влияющих на цену, включая площадь объекта, его расположение, удалённость от транспортных узлов, инфраструктуру и прочие параметры. Это даёт возможность пользователям получать точные и объективные результаты оценки, а также ускоряет процесс работы с базой данных недвижимости.

Программа предоставляет пользователям следующие возможности:

1 Просмотр информации об объектах недвижимости, содержащихся в базе данных.

2 Поиск недвижимости по различным параметрам, таким как площадь объекта, расстояние до метро, рейтинг или стоимость.

3 Сортировка объектов по заданным критериям, например, по рейтингу, расстоянию до метро или цене.

4 Оценка объектов недвижимости с помощью встроенных алгоритмов.

Для оценки объектов недвижимости предусмотрены специализированные инструменты, которые позволяют профессиональным оценщикам работать с базой данных объектов, проводить анализ и вносить оценочные значения, которые влияют на расчёт итоговой стоимости недвижимости. Система учитывает введённые данные и автоматически пересчитывает цену на основе оценок, что упрощает работу оценщиков и позволяет обеспечить точность расчётов.

Система включает несколько уровней доступа для различных пользователей:

1 Обычные пользователи имеют возможность просматривать объекты недвижимости и выполнять поиск по заданным критериям.

2 Оценщики могут проводить оценку объектов, просматривать список недвижимости и вносить данные о её стоимости. Эта информация влияет на итоговую цену объекта и помогает создать более точную оценочную модель.

3 Администраторы имеют расширенные права: они могут управлять аккаунтами пользователей, добавлять или удалять объекты недвижимости, а также редактировать существующие записи. Администраторы отвечают за поддержание актуальности данных и корректность работы системы.

Для добавления нового объекта недвижимости в базу данных пользователю необходимо иметь права администратора. После авторизации в учётной записи администратора требуется ввести основные характеристики объекта, такие как его название, площадь, расстояние до ближайшего метро и другие параметры. Программа автоматически создаёт запись об объекте в базе данных, и после проверки данных объект становится доступным для поиска и оценки другими пользователями.

Для оценки объекта недвижимости оценщик проходит авторизацию, после чего может выбрать объект из списка и внести свою оценку, которая сохраняется в базе данных автоматически. Это позволяет обеспечить прозрачность и достоверность оценок, а также упростить процесс управления информацией о недвижимости.

На сегодняшний день существует множество аналогичных систем для автоматизированной оценки недвижимости, таких как "Недвижка.tech"[1] и решения от ООО «Группа Комплексных Решений».[2] Эти системы используют различные алгоритмы для расчёта стоимости объектов на основе множества параметров, таких как площадь, расположение, тип дома и другие. Однако большинство таких систем являются коммерческими и ограничены в доступе, предоставляя полный функционал только своим клиентам. В отличие от них, предлагаемая система распространяется по лицензии GNU GPL V3, что позволяет её свободное использование и модификацию под нужды конкретных пользователей. Это открывает широкие возможности для адаптации системы под различные сценарии использования и интеграции с другими инструментами. [3]

Система предлагает удобный интерфейс, который обеспечивает простой доступ к функциям поиска, сортировки и оценки объектов недвижимости. Пользователям не требуется глубоких технических знаний для работы с системой, что делает её доступной для широкого круга специалистов, включая оценщиков, риэлторов, администраторов баз данных и других участников рынка недвижимости.

Помимо основных возможностей, система автоматизированной оценки недвижимости может быть интегрирована с другими сервисами для улучшения функциональности, такими как карты для визуализации объектов, внешние базы данных для получения дополнительной информации о районе, статистические сервисы для анализа рынка недвижимости. Это позволяет

расширить круг возможностей для анализа данных и повысить точность оценки объектов.

Таким образом, система автоматизированной оценки недвижимости решает важные задачи в сфере управления информацией о недвижимости и расчёта её стоимости. Благодаря современным алгоритмам обработки данных, поддержке различных уровней пользователей и гибкости использования, она является мощным инструментом для профессионалов, работающих с объектами недвижимости.

1.2 Построение функциональной модели системы оценки объектов недвижимости

IDEF0 (Integration Definition for Function Modeling) – это методология и язык моделирования, используемые для анализа и проектирования бизнес-процессов [7]. Она позволяет описывать функции, потоки данных и контроль в бизнес-процессах и представляет их в виде блок-схем.

Использование IDEF0 позволяет более эффективно проектировать бизнес-процессы, повышать качество их выполнения и уменьшать затраты на их выполнение. Она является надежным инструментом для анализа и проектирования систем, включая систему по оценке недвижимости.

Рассмотрим информационную систему, выполненную с помощью средств моделирования функций IDEF0. Для начала необходимо сделать контекстную модель информационной системы. Контекстная диаграмма – самая верхняя диаграмма, на которой объект моделирования представлен единственным блоком с граничными стрелками. Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой.

На рисунке 1.1 описан основной блок «Оценка объекта недвижимости». Входящие стрелки – «Данные об объекте», «Данные пользователей». Это то, что необходимо иметь для начала работы. Стрелки управления – «Требования к курсовому проекту», «Руководства к стандартным библиотекам C++», «правила оценки недвижимости». В роли механизмов выступают пользователь, администратор и программное обеспечение. После завершения процесса мы получаем «Стоимость объекта недвижимости».

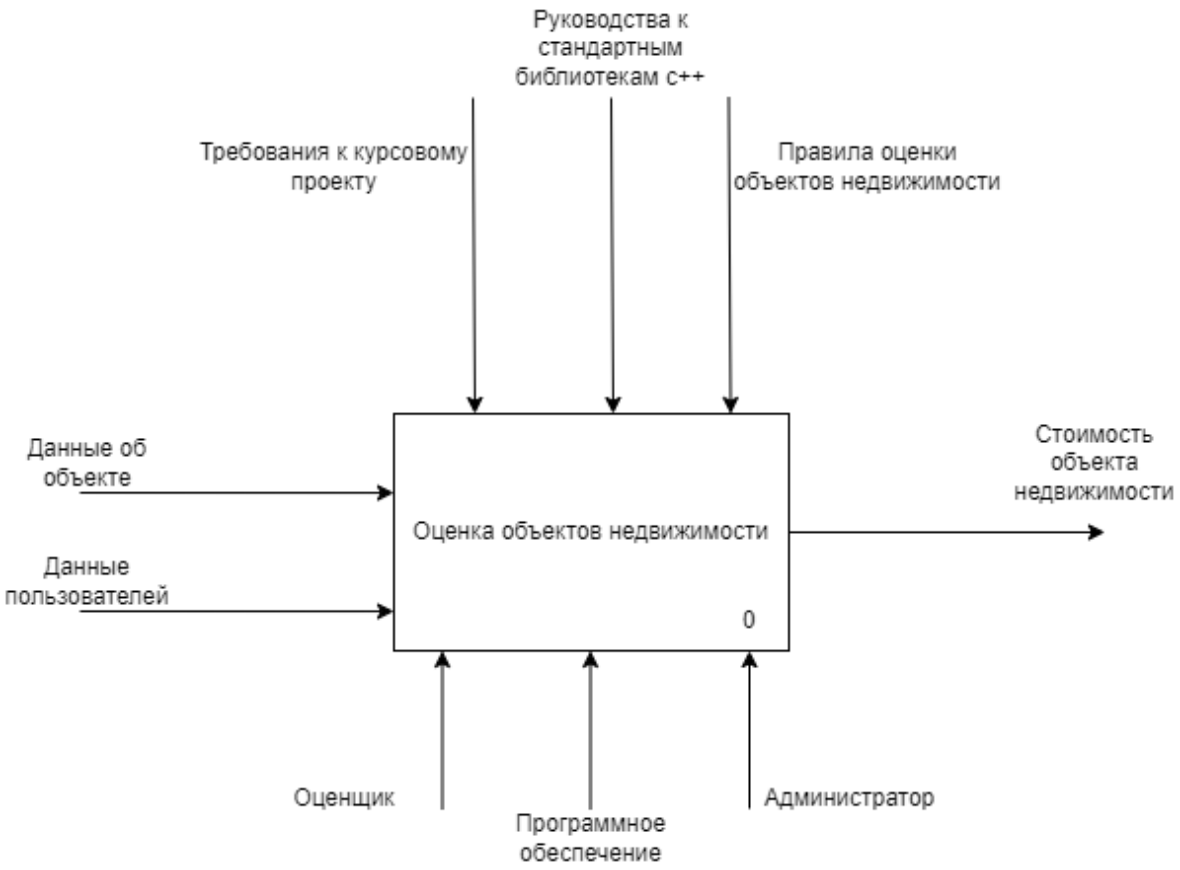

Рисунок 1.1 – Контекстная диаграмма модели A-0

На рисунке 1.2 представлена декомпозиция процесса «Оценка объектов недвижимости». В данном случае мы получили диаграмму, состоящую из четырех процессов:

1 Авторизация в программе – процесс авторизации оценщика или администратора.

2 Добавление недвижимости – процесс занесения в базу данных недвижимости, для последующей оценки оценщиками.

3 Запрос данных об объекте недвижимости в базу данных – процесс предоставления пользователю итоговых данных об объекте недвижимости.

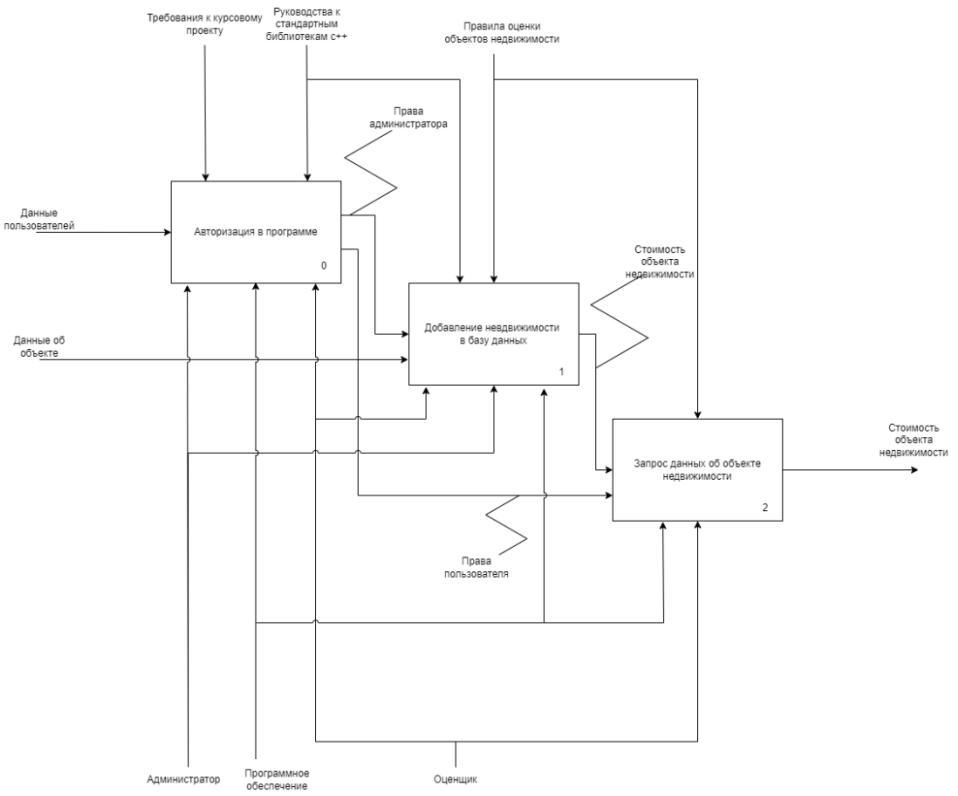

Рис. 1.2 – Декомпозиция блока «Оценка объектов недвижимости»

На рисунке 1.3 предоставлена декомпозиция процесса «Добавление недвижимости в базу данных». Для этого нужно иметь данные об объекте, внести их в базу, рассчитать его стоимость и проверить итоговые данные.

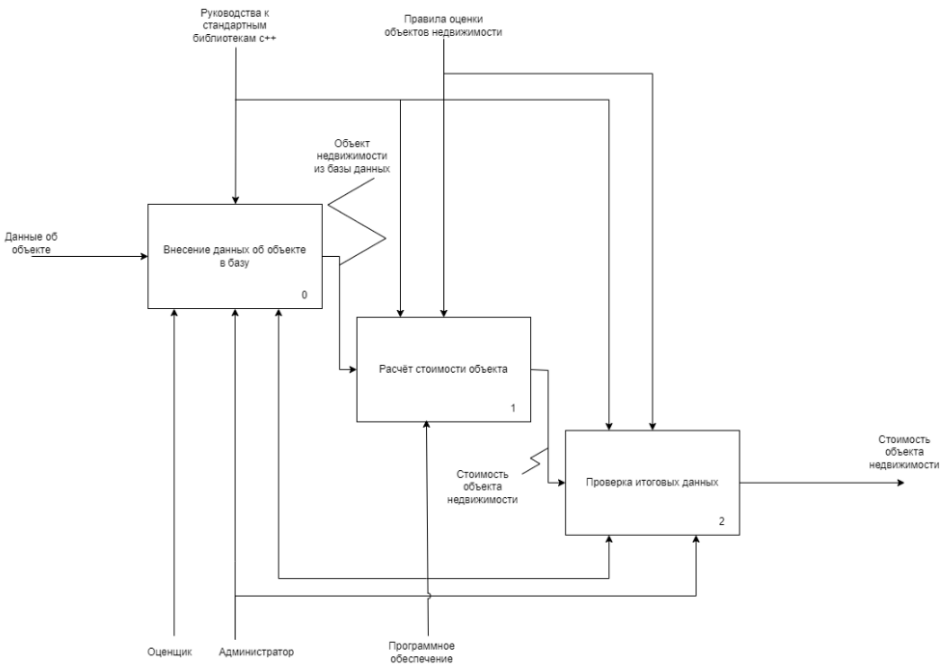

Рис. 1.3 – Декомпозиция блока «Добавление недвижимости в базу данных»

На рисунке 1.4 предоставлена декомпозиция процесса «Внесение данных об объекте в базу». Этот блок включает в себя создание объекта и занесение в него значений, включая оценку.

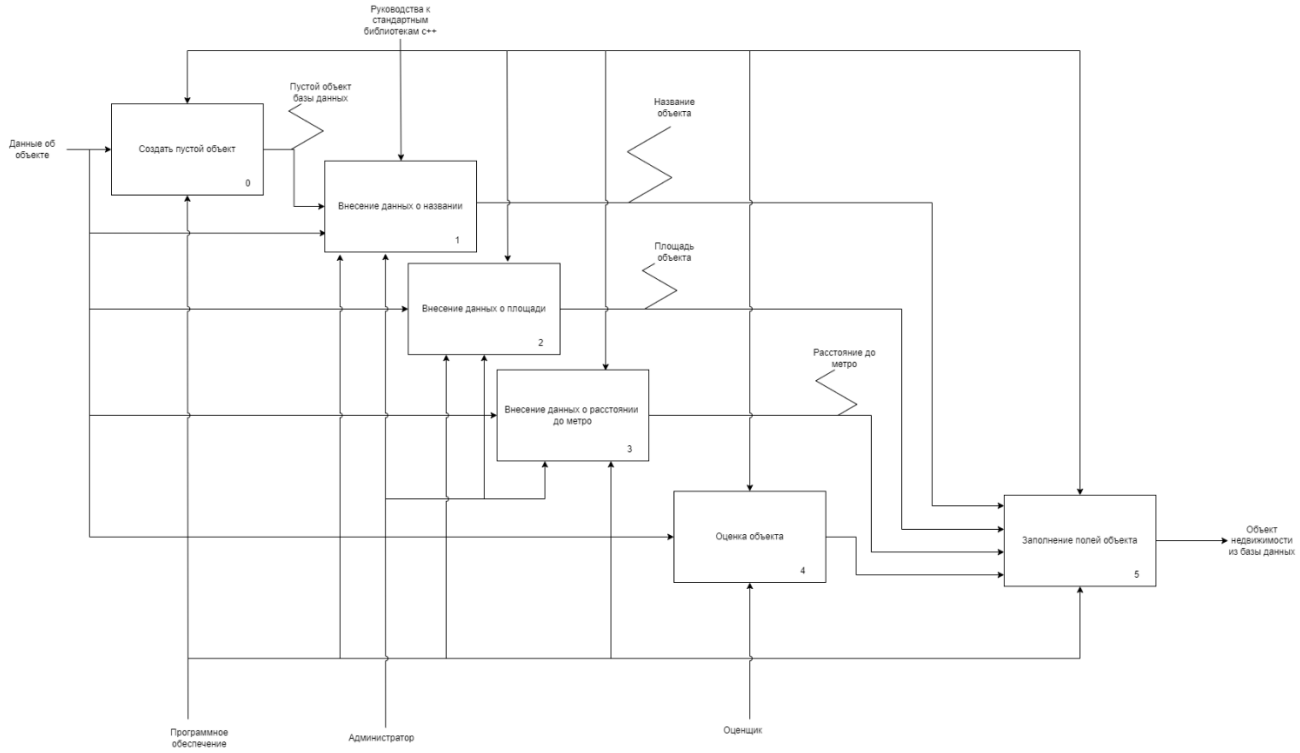

Рис. 1.4 – Декомпозиция блока «Внесение данных об объекте в базу»

Таким образом, мы визуализировали организацию и задачи нашей системы, используя язык IDEF0.

Анализ и моделирование бизнес-процессов, связанных с оценкой объектов недвижимости, с использованием диаграмм IDEF0, позволяет лучше понять структуру системы, визуализировать модели объектов и процессов, определить потоки информации и выявить возможные узкие места и проблемы в процессе. Использование такого подхода помогает разработчикам создать эффективную и удобную систему тестирования студентов по языку C++, которая удовлетворит потребности пользователей и бизнес-процессов.

2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ОЦЕНКИ ОБЪЕКТОВ НЕДВИЖИМОСТИ

В рамках данной курсовой работы разрабатывается автоматизированная система оценки объектов недвижимости. Данная система представляет собой программное обеспечение, в основе которого лежат принципы объектно-ориентированного программирования, что позволяет обеспечить структурированный и удобный подход к разработке.

Основу системы составляют данные о пользователях и тестах, которые будут представлены в виде объектов с четко определенными атрибутами и методами. Это обеспечивает гибкость в обработке информации и упрощает взаимодействие между компонентами программы. Реализация системы предполагает создание набора классов, моделирующих ключевые элементы системы: учетные записи пользователей, тесты, вопросы и ответы.

Использование объектно-ориентированного подхода позволяет не только систематизировать процесс разработки, но и заложить возможности для дальнейшего масштабирования и улучшения программы. Такой подход также способствует повышению читаемости кода и упрощению его сопровождения.

Проект ориентирован на применение C++ как основного языка разработки, что обусловлено его широкими возможностями для реализации объектно-ориентированных концепций, таких как наследование, полиморфизм и инкапсуляция.

2.1 Информационная модель системы и ее описание

Разработка автоматизированной системы оценки недвижимости требует тщательной проработки информационной модели, которая задает структуру и взаимодействие между элементами системы. В данной части описывается структура хранимых данных, формат файлов, используемых в системе, и их взаимосвязь.

Для реализации автоматизированной системы тестирования студентов используется файловая система, состоящая из двух текстовых файлов: `accounts.txt` и `objects.txt`. Каждый из них имеет свое назначение, формат данных и правила обработки.

objects.txt. Этот файл хранит данные об объектах недвижимости. Каждая строка файла соответствует одному объекту недвижимости и его оценкам, и содержит следующую информацию:

– тип объекта недвижимости;

– название объекта;

– количество квадратных метров;

– расстояние до метро;

– рассчётная цена;
– цена квадратного метра ГОСТ;
– имена оценщиков;
– оценки.
Формат записи:

```
Тип Объекта(h/o);название_объекта;кол-
во_кв.м.;рассч.цена;расст.до_метро;цена_кв.м._ГОСТ
r;имя_оценщика;оценка
```

Пример:

```
h House1 75 8986 900
r Appraiser1 75
r АнтонЗакревский 52
h House2 42 17109 1200
r Appraiser1 60
h House3 110 14519 2000
r Appraiser1 80
o Office1 20 0 310
r Appraiser1 50
o Office2 60 0 100
h Saray 12 0 3200
```

accounts.txt Данный файл содержит информацию о зарегистрированных учётных записях в системе. Запись включает:
– логин;
– пароль в зашифрованном виде;
– тип учётной записи(2 – оценщик,1 – администратор, 0 – обычный пользователь);
– наличие доступа к базе данных.

Формат записи:

```
логин пароль_в_зашифрованном_виде тип_уч._записи доступ
```

Пример:

```
АлексейЕднач феыщсш 0 0
АнтонЗакревский Лйщыйднмщ 1 1
АлексейЦевелюк 4;8768 2 1
```

Файлы тесно взаимосвязаны через логику программы:
– accounts.txt используется для аутентификации пользователей и проверки их роли (администратор, оценщик или гость).

–objects.txt хранит результаты оценки, которые используются для дальнейшего отслеживания цен.

IDEF1X-схема структуры файлов и их взаимодействия представлена ниже:

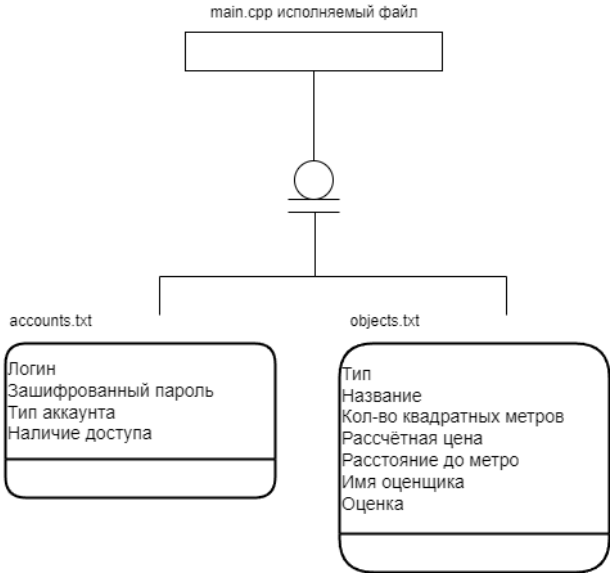

Рис. 2.1 – IDEF1X-схема структуры файлов и их взаимодействия

Взаимодействие файлов можно описать следующими процессами:

1 Аутентификация. При входе в систему программа обращается к accounts.txt, чтобы проверить логин и пароль пользователя. Тип учётной записи отвечает за набор функций, предоставляемых пользователю Параметр доступа определяет доступ к функциям базы данных.

2 Управление объектами недвижимости. Администраторы могут добавлять или удалять объекты в базе данных, а оценщики могут вносить оценки, которые записываются в файл objects.txt. Оценка влияет на расчётную стоимость объекта.

3 Оценка объектов. Оценщики выбирают объекты для оценки, вводят свои данные, и система сохраняет результаты в соответствующей строке файла objects.txt. Эти данные могут быть использованы для анализа и расчёта стоимости объектов.

Преимущества разработанной файловой структуры:

1 Простота реализации. Текстовые файлы не требуют сложной настройки и легко обрабатываются программой.

2 Удобство переноса данных. Файлы легко переносятся и могут быть прочитаны вручную при необходимости.

3 Минимальные системные требования. Текстовые файлы подходят для работы даже на компьютерах с ограниченными ресурсами.

Таким образом, информационная модель системы автоматизированной оценки недвижимости обеспечивает простоту и эффективность в работе с

данными, позволяя пользователям легко управлять объектами недвижимости, а также автоматизировать процессы оценки и расчёта стоимости.

2.2 Модели представления системы и их описание

В данном подразделе будет продемонстрировано моделирование автоматизированной системы оценки стоимости недвижимости с использованием стандарта UML (Unified Modeling Language). Этот стандарт применяет графические обозначения для создания абстрактной модели системы и предназначен для определения, визуализации, проектирования и документирования программных систем.

UML предоставляет возможность описывать систему с разных точек зрения, таких как:

– статическая структура системы (классы, объекты, их атрибуты и отношения);

– динамическое поведение, включающее взаимодействие между компонентами системы, последовательность выполнения операций и управление потоками данных.

Использование UML позволяет не только детализировать архитектуру программного обеспечения, но и сделать проект более понятным для разработчиков, тестировщиков и других участников процесса. Это способствует формализации требований, улучшает согласованность между всеми этапами разработки и обеспечивает полноценную документацию проекта.

2.2.1 Диаграмма вариантов использования

Диаграмма вариантов использования – это один из ключевых инструментов UML, который позволяет визуализировать взаимодействие между пользователями (акторами) и системой. Она предоставляет наглядное представление о том, какие функции доступны различным ролям, а также помогает разработчикам понять, как пользователи будут работать с системой оценки объектов недвижимости.

В рамках данной автоматизированной системы оценки недвижимости диаграмма вариантов использования помогает проанализировать задачи, которые пользователи должны выполнять. Это важно для проектирования удобного интерфейса и обеспечения необходимой функциональности.

Преимущества диаграммы вариантов использования:

1 Простота и ясность. Диаграмма позволяет легко понять, какие задачи решает система, и как она взаимодействует с различными пользователями. Это помогает в дальнейшем проектировании интерфейсов и логики.

2 Гибкость в дополнении. Если в будущем потребуется добавить новые функции (например, добавление методов оценки или расширение списка критериев недвижимости), их легко интегрировать в текущую модель.

3 Понимание для всех участников разработки. Диаграмма доступна для понимания не только программистам, но и другим участникам проекта, включая оценщиков, администраторов или даже клиентов, использующих систему.

Обоснование выбора вариантов использования:

1 Роли в системе. В системе выделяются три основные роли: Администратор, Оценщик и Пользователь. Администратор отвечает за управление базой объектов недвижимости (добавление, удаление и редактирование объектов), а также управляет учетными записями пользователей. Оценщик занимается выставлением оценок для объектов, влияя на их стоимость. Пользователь имеет возможность просматривать объекты недвижимости, искать по определённым критериям и сортировать их.

2 Уникальные варианты использования. Каждый вариант использования был выбран на основе ключевых сценариев работы системы. Например, добавление объектов недвижимости и управление аккаунтами — это основные процессы для администратора. Оценка объектов и анализ стоимости недвижимости — ключевые задачи оценщика. Пользователь может воспользоваться функциями поиска, сортировки и фильтрации объектов, что делает систему удобной для работы с большим количеством данных.

3 Взаимодействие с файлами. Использование текстовых файлов для хранения данных об объектах и пользователях также учитывается в диаграмме. Например, при добавлении или оценке объектов недвижимости система записывает данные в файл objects.txt, а для аутентификации пользователей используется файл accounts.txt. Это показывает связь между функциональностью системы и её файловой структурой.

Таким образом, диаграмма вариантов использования позволяет наглядно представить взаимодействие всех ролей с системой, обеспечивая ясность и упрощение разработки системы оценки недвижимости.

В результате была разработана диаграмма вариантов использования, показанная ниже:

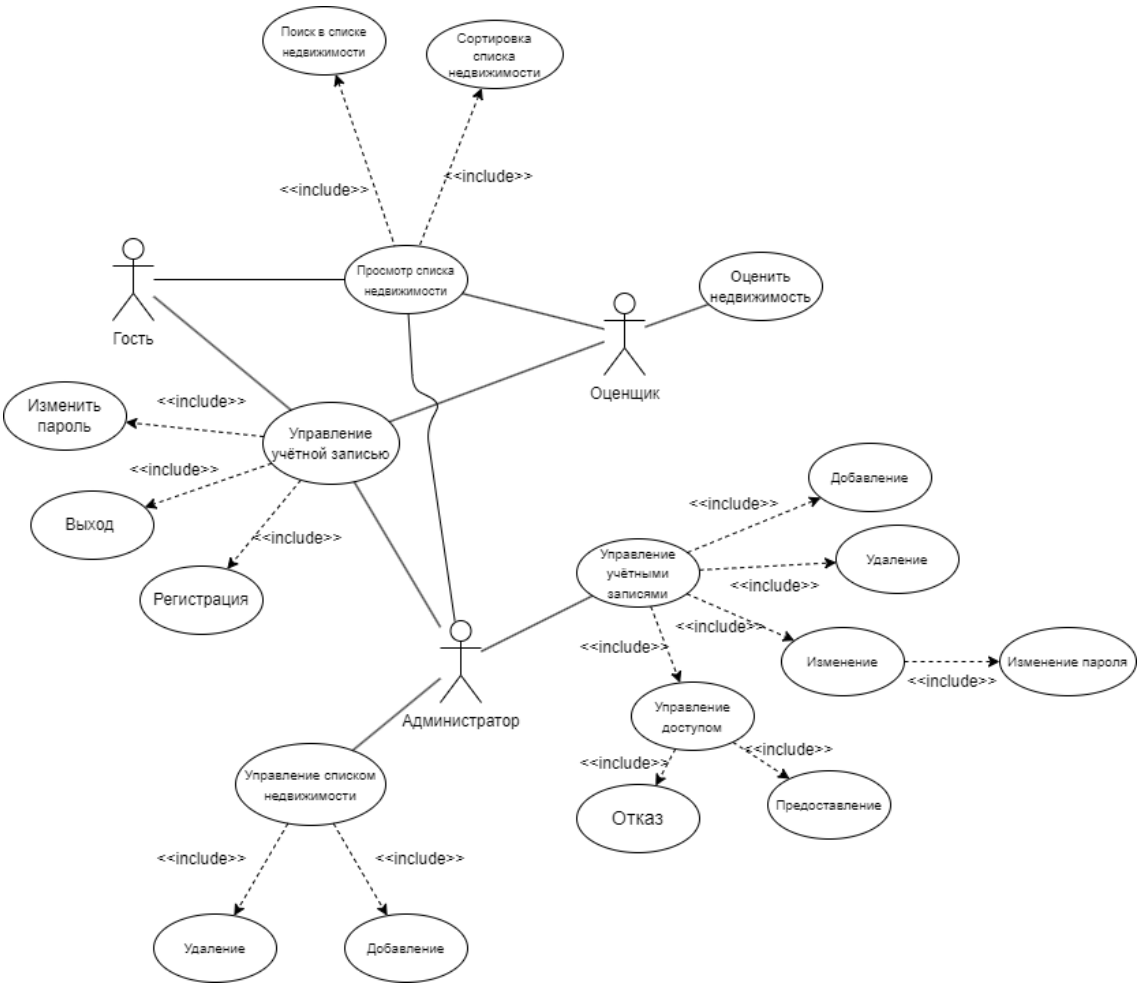

Рис. 2.2 – UML-диаграмма вариантов использования

Описание вариантов использования исключительно для оценщика:

1 Регистрация. Гости должны иметь возможность зарегистрироваться в системе, используя логин и пароль.

2 Оценка объекта недвижимости. Самая главная функция системы, которая формирует стоимость недвижимости.

Описание вариантов использования исключительно для администратора:

1 Управление списком объектов недвижимости. Включая удаление и добавление.

2 Управление учётными записями. Включая добавление, удаление, изменение, отказ и предоставление доступа.

Описание вариантов использования для всех пользователей:

1 Авторизация. Вне зависимости от роли, система потребует перед использованием ввести свой логин и пароль, чтобы авторизироваться в ней.

2 Просмотр списка объектов недвижимости. Также предусматривается возможность поиска и сортировки объекта по определённому критерию.

3 Изменение пароля. Любой пользователь может изменить свой пароль.

4 Выход из аккаунта. После завершения работы, пользователь может выйти из своей учетной записи, предоставив возможность войти другому.

5 Выход из программы. После того, как все пользователи закончили свои сессии в программе, ее можно закрыть безопасным способом, сохранив все изменения в файлах системы.

Таким образом, диаграмма вариантов использования является важным этапом моделирования системы. Она демонстрирует ключевые аспекты ее поведения и взаимодействия с пользователями, создавая основу для детальной проработки остальных моделей. В данном проекте диаграмма способствует структурному подходу к проектированию, что в итоге улучшает ее функциональность, производительность и удобство для пользователей.

2.2.2 Диаграмма последовательностей

Диаграмма последовательностей — это диаграмма поведения, которая показывает взаимодействие объектов системы и подчеркивает временную последовательность событий [9]. В рамках данной работы диаграмма последовательностей используется для точного определения логики сценария выполнения ключевых вариантов использования.

Диаграммы последовательностей помогают разработчикам понять, какие объекты участвуют в выполнении сценария, какие сообщения передаются между ними и в каком порядке эти сообщения отправляются. Также отображаются возможные возвращаемые значения, что важно для отладки и проверки работы системы.

Рассмотрим сценарий «Оценка объекта недвижимости».

На диаграмме последовательностей рассматривается пример сценария, где оценщик авторизуется в системе, выбирает объект недвижимости, оценивает его и программа сохраняет изменения. Основными участниками взаимодействия в данном сценарии являются:

– оценщик (актор, представляющий оценщика);

– система (обеспечивает интерфейс и выполняет основные функции);

– база данных об объектах недвижимости (обеспечивает загрузку и сохранение данных).

В результате была разработана диаграмма последовательностей, показанная ниже:

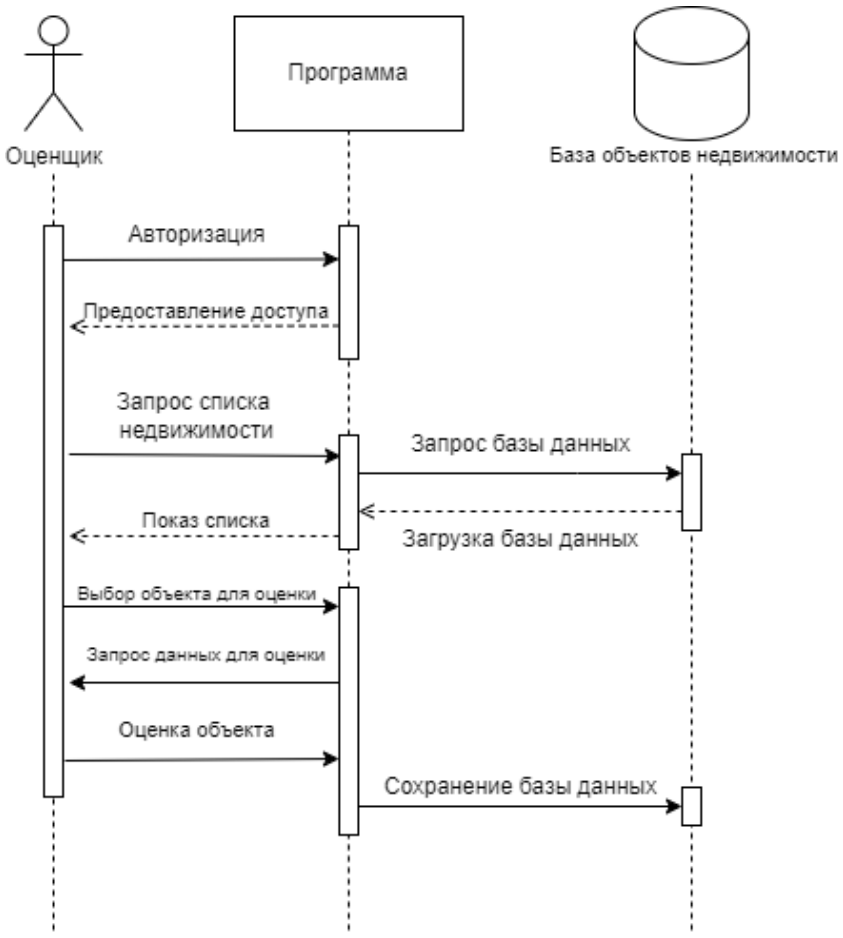

Рис. 2.3 – UML-диаграмма последовательностей

Расшифровка диаграммы:

1 Авторизация. Оценщик вводит данные своей учётной записи.

2 Предоставление доступа. Программа проверяет их и предоставляет доступ.

3 Запрос списка недвижимости. Оценщик запрашивает список объектов недвижимости.

4 Запрос базы данных. Запрос данных для просмотра и изменения.

5 Загрузка базы данных. Загрузка базы данных для просмотра и изменения.

6 Показ списка. Программа показывает список объектов недвижимости оценщику.

7 Выбор объекта для оценки. Оценщик выбирает объект недвижимости, который будет оценивать.

8 Оценка объекта. Добавление объекту недвижимости оценки.

9 Сохранение базы данных. Сохранение изменений и запись новых данных в файл.

Диаграмма последовательностей позволяет детализировать процесс выполнения сценария "Оценка объекта недвижимости". Она наглядно демонстрирует взаимодействие между пользователем, системой и ее

компонентами, а также уточняет порядок обмена данными. Это помогает не только разработчикам понять логику системы, но и выделить возможные узкие места в сценарии, требующие оптимизации.

Использование данной диаграммы на этапе проектирования системы способствует улучшению структуры кода, упрощает тестирование и дальнейшую доработку функционала.

2.2.3 Диаграмма классов

Диаграмма классов является одним из ключевых инструментов объектно-ориентированного проектирования и моделирования систем [8]. Она описывает структуру системы, определяя классы, их атрибуты, методы, а также связи между классами, такие как ассоциации, наследование и зависимости.

Диаграмма классов позволяет разработчику:

– увидеть общую архитектуру системы;

– определить основные сущности и их взаимодействие;

– наглядно представить реализацию бизнес-логики;

– выявить потенциальные места улучшений структуры кода.

На представленной диаграмме классов системы автоматизированного тестирования студентов по языку C++ выделены основные классы: `UI`, `Database`, `Account`, `Object`, `Rating`, а также класс для обработки ошибок связанных с учётными записями `AccountException`.

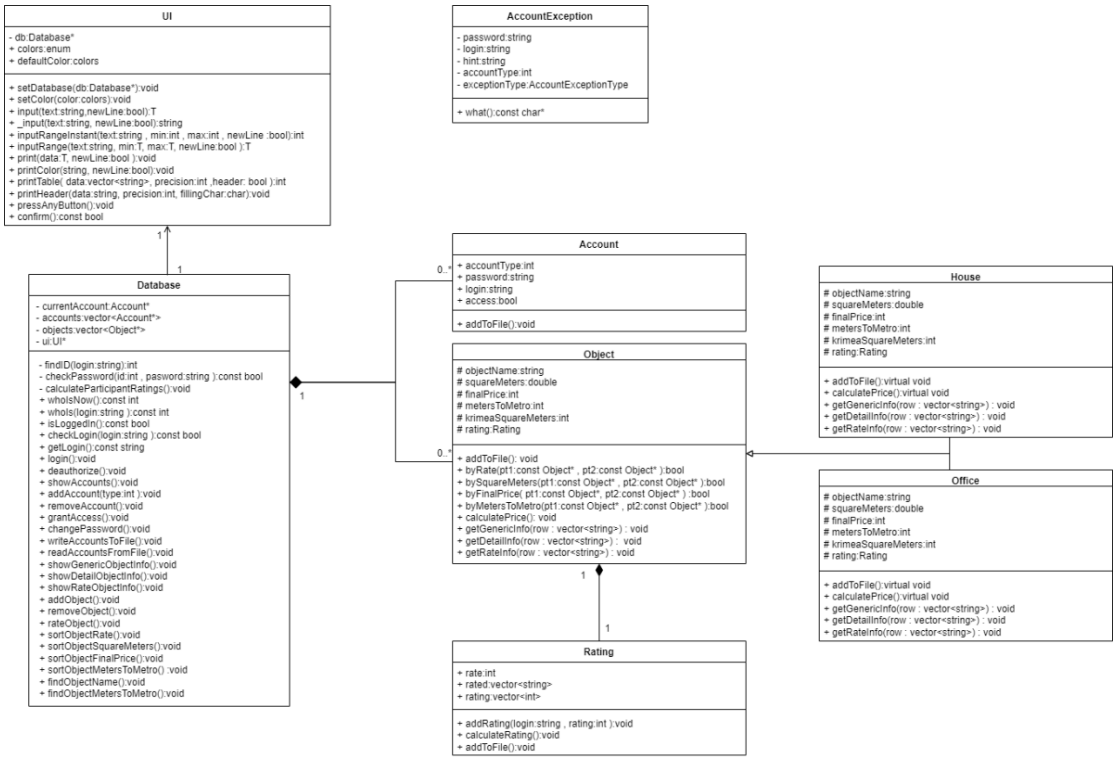

Рис. 2.4 – UML-диаграмма классов

Основные элементы диаграммы:

1 Класс UI. Назначение: отвечает вывод данных в консоль и ввод пользователя, получаемый из консоли. Ключевые методы: содержит в себе все методы ввода-вывода данных, а также меню с которыми взаимодействует пользователь. Причина включения: обеспечивает взаимодействие программы с пользователем.

2 Класс Database. Назначение: отвечает за хранение информации об объектах недвижимости и учётных записях пользователей, их изменение и сохранение. Ключевые методы: предоставляют доступ к базе данных, обеспечивают изменение, удаление и сохранение данных и учётных записей пользователей. Причина включения: объединяет в себе все элементы программы и предоставляет основной функционал программы.

3 Класс Object Назначение: хранит данные об объекте недвижимости. Ключевые методы: позволяют рассчитать стоимость недвижимости и записать её в файл. Причина включения: данные об объектах недвижимости необходимо объединить вместе, чтобы ими было удобно управлять.

4 Класс Account Назначение: хранит данные о пользователях системы (логин, пароль, тип и право доступа). Ключевые методы: отвечают за запись в файл. Причина включения: система должна различать гостей, оценщиков и администраторов, что обеспечивает гибкость и безопасность.

5 Класс AccountException Назначение: используется для обработки ошибок на этапах работ с учётными записями пользователей. Ключевые методы: позволяют получить данные об ошибке. Причина включения: улучшение тестирования и отладки программы.

Связи между классами:

– наследование: класс Object является родительским по отношению к классам House и Office;

– ассоциация: класс Database пользуется функционалом класса UI, для ввода-вывода информации пользователю;

– композиция: класс Database включает в себя коллекцию объектов Object и Account, так как объекты недвижимости являются неотъемлемой частью базы данных, а также доступ к ним не может осуществляться без учётных записей. Объекты и аккаунты существовать обособленно не могут.

Также класс Object включает в себя объект класса Rating, который содержит в себе оценки объекта недвижимости.

Диаграмма классов разработанной системы позволяет структурировать и упрощать разработку, так как она наглядно демонстрирует взаимодействие между ключевыми элементами. Выбранные классы и их связи полностью отвечают функциональным требованиям к системе: возможность оценки недвижимости, авторизации пользователей, управления данными об объектах недвижимости.

2.2.4 Диаграмма состояний

Диаграмма состояний – это диаграмма поведения, которая моделирует жизненный цикл объекта. Она показывает, в каких состояниях может находиться объект, а также переходы между этими состояниями, вызванные внешними или внутренними событиями.

Объекты системы изменяют свое состояние в зависимости от поступающих событий или выполненных действий. Каждое состояние характеризуется особыми параметрами, а переходы между состояниями указывают на реакцию системы на определенные события. Диаграмма состояний помогает проанализировать и понять динамическую природу объекта в системе, что делает ее незаменимым инструментом при проектировании сложных приложений.

Диаграмма состояний особенно полезна в системах, где важно отслеживать последовательность действий или событий, влияющих на поведение компонентов.

В контексте автоматизированной системы оценки объектов недвижимости, для которого логично построить диаграмму состояний, является процесс оценки недвижимости. На диаграмме отражены ключевые этапы, включая начальное и конечное состояния, а также промежуточные действия.

Рис. 2.5 – UML-диаграмма состояний

Описание диаграммы состояний:

1 Начальное состояние. Система переходит в это состояние после выбора оценщиком пункта меню «Оценить объект».

2 Состояние «Выбор объекта недвижимости». В этом состоянии программа выводит оценщику список недвижимости и ожидает, пока оценщик введёт номер недвижимости к оценке.

3 Состояние «Ввод оценки объекта». Выбрав объект недвижимости программа ждёт ввода оценки. Это состояние завершается, как только оценка введена.

4 Состояние «Применение и сохранение изменений». На этом этапе программа рассчитывает итоговую стоимость объекта недвижимости и сохраняет данные в файл.

5 Конечное состояние. После того, как оценка введена, стоимость объекта рассчитана и записана в файл, программа переходит в это состояние.

Использование диаграммы состояний способствует разработке более стабильной системы, так как помогает понять, как объект ведет себя в зависимости от различных событий. В системе оценки объектов недвижимости это особенно важно для создания понятного и интуитивного интерфейса взаимодействия с пользователями.

2.3 Описание алгоритмов, реализующих бизнес-логику системы

Алгоритмы являются ключевым элементом бизнес-логики любой системы. Они определяют, как система обрабатывает данные, реагирует на пользовательские действия и обеспечивает выполнение поставленных задач.

В данной части представим основные алгоритмы автоматизированной системы оценки объектов недвижимости. Были разработаны блок-схемы, описывающие логику работы программы в целом, а также основные методы: проверка ответов на вопросы теста и формирование итогового результата теста.

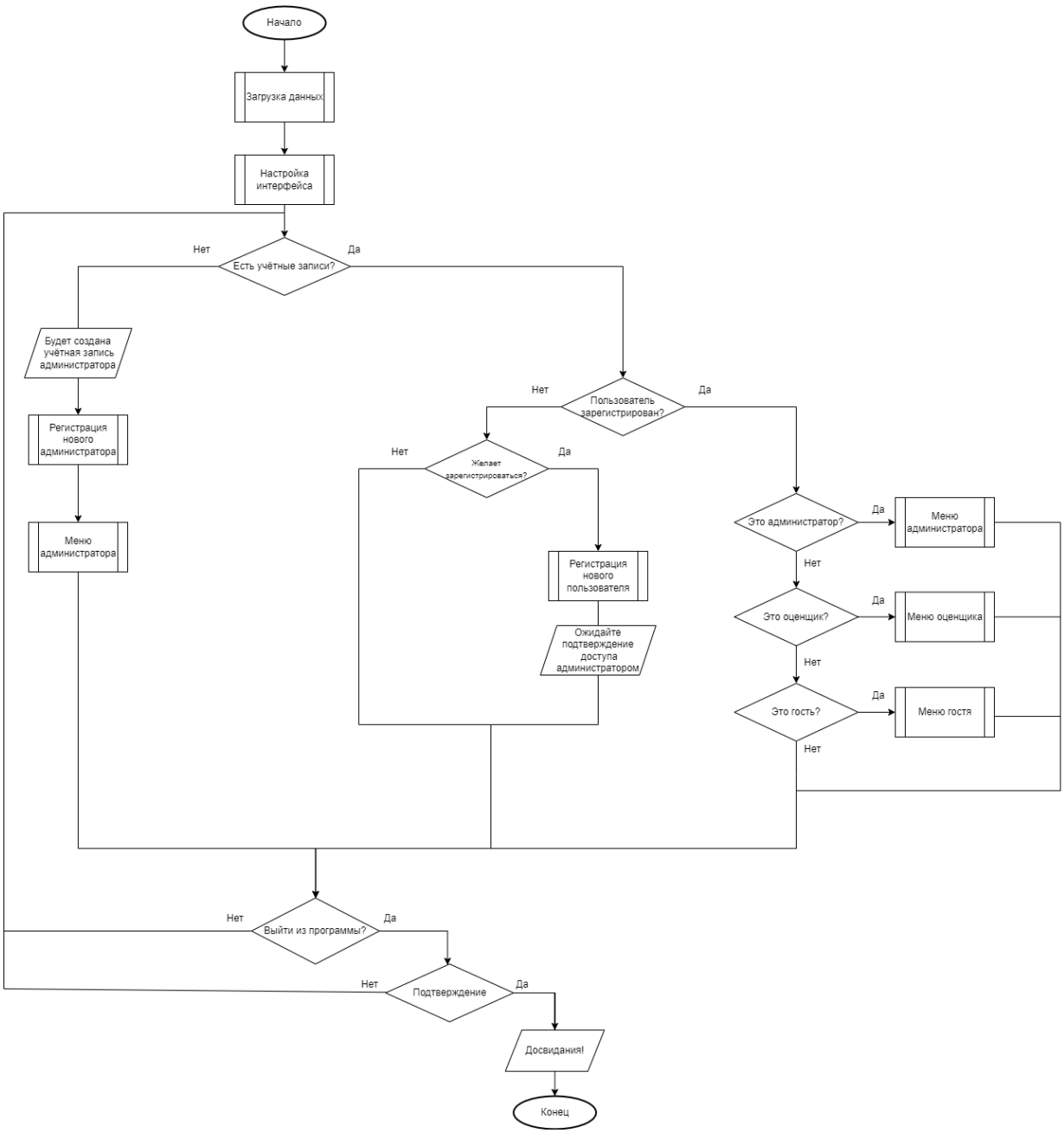

Рис. 2.6 – Схема алгоритма работы программы

Данный алгоритм демонстрирует, как система функционирует в зависимости от роли пользователя (администратор, гость или оценщик). Включены этапы авторизации, выбора функционала, предоставляемого пользователю и завершения программы, а также предложение зарегистрироваться и проверка наличия учётных записей.

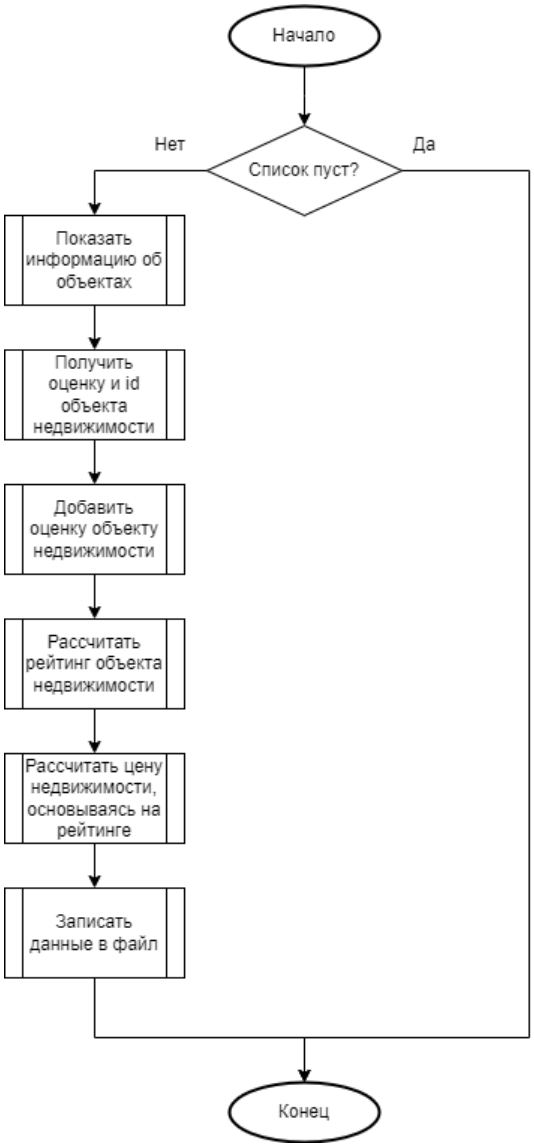

Рис. 2.7 – Схема алгоритма оценки недвижимости

Алгоритм показывает логику оценки объекта недвижимости. Вначале программа смотрит: есть ли данные в списке. Затем показывает информацию об объектах. Получает номер и оценку выбранного объекта недвижимости, добавляет оценку, рассчитывает рейтинг объекта, рассчитывает цену и записывает объекты в файл.

Алгоритмы, реализующие бизнес-логику системы, обеспечивают ее функциональную целостность и корректное выполнение задач. Разработанные схемы наглядно иллюстрируют процессы работы системы в целом, а также ключевых методов, таких как оценка недвижимости.

2.4 Описание созданных программных конструкций

Для реализации автоматизированной системы тестирования студентов по языку C++ использовались разнообразные возможности языка программирования C++, включая основные принципы объектно-ориентированного программирования, работу с потоками, обработку ошибок и использование стандартной библиотеки. Это позволило создать надежную, расширяемую и удобную в сопровождении программу. Рассмотрим примененные программные конструкции и их роль в реализации системы:

1 Реализация базовых принципов объектно-ориентированного программирования:

– инкапсуляция. В программе класс Database демонстрирует инкапсуляцию. Поля currentAccount, accounts, objects сделаны закрытыми (private), а доступ к ним осуществляется через методы. Это защищает данные от прямого изменения извне и позволяет добавлять логику проверки:

```
const int whoIsNow() const {
     if (isLoggedIn()) return currentAccount->accountType;
          else return -1;
     }
```

– наследование и полиморфизм. В файле видно, что абстрактный класс Object наследуется классами Office и House. Это позволяет реализовать общий интерфейс управления данными:

```
class Office : public Object {
public:
     void calculatePrice() override;
     void save(std::ostream& out) override;
     void getGenericInfo(vector<string>& row) override;
     void getDetailInfo(vector<string>& row) override;
     void getRateInfo(vector<string>& row) override;
     void addToFile() const override;
};
```

2 Использование абстрактных классов. Класс Object является абстрактным, так как содержит чисто виртуальные функции. Он служит родителем для классов Office и House. Это позволяет унифицировать работу с разными типами недвижимости:

```
class Object {
public:
     virtual void addRating(string login, int rating) = 0;
     virtual void calculatePrice() = 0;
     virtual void addToFile() const = 0;
     virtual void getGenericInfo(vector<string>& row) = 0;
```

```
    virtual void getDetailInfo(vector<string>& row) = 0;
    virtual void getRateInfo(vector<string>& row) = 0;
};
```

3 Передача параметров:

– по значению: Пример функции addRating в классе Object. Здесь объекты login и rating передаются по значению:

```
void addRating(string login, int rating) {
    for (int i = 0; i < rated.size(); i++)
        if (rated.at(i) == login) {
            ratings.at(i) = rating;
            return;
        }
    rated.emplace_back(login);
    ratings.emplace_back(rating);
}
```

– по ссылке. Методы вроде setDatabase в классе UI используют передачу параметров по ссылке. Это экономит память и позволяет работать с оригинальным объектом:

```
void setDatabase(Database& db) { this->db = &db;}
```

4 Использование функций:

– пользовательские функции. Пример поиска объекта недвижимости по названию. Эта функция реализует специфичную логику, необходимую для поиска объектов недвижимости:

```
void findObjectName();
```

– дружественные функции. В классе Account перегружен оператор <<, объявленный дружественным. Это позволяет упрощать работу с потоками:

```
friend ostream& operator <<(ostream& out, Account& acc);
```

– виртуальные функции. В классе Object используются виртуальные функции addToFile, calculatePrice, что позволяет полиморфно работать с разными типами объектов.

5 Пространства имен:

– встроенные. Например, пространство имен std активно используется для работы с потоками и контейнерами:

```
#include <iostream>
using namespace std;
```

– собственные. В программе существует собственное пространство имен app, что упрощает использование программы:

```
using namespace defaultLabStructures;
using namespace realtyPriceRate;

int main(){/*...*/}
```

6 Обработка ошибок. Блоки обработки исключений (try-catch) применяются для надежной работы с учётными записями пользователей. Это гарантирует устойчивость программы при ошибках вызванных неправильным вводом данных. Они используются в функции login() отвечающей за авторизацию пользователя.

7 Перегрузка и переопределение:

– перегрузка операторов. Класс Account перегружает оператор << для удобного вывода в файл:

```
friend ostream& operator <<(ostream& out, Account& acc) {
     out << acc.login << ' ' << enDecrypt(acc.password)
     << ' ' << acc.accountType << ' ' << acc.access << '\n';
     return out;
}
```

– переопределение методов. В классе Office переопределяется метод calculatePrice базового класса Object:

```
void calculatePrice() override {
     // Рассчёт стоимости объекта недвижимости
}
```

8 Шаблоны и шаблонные методы. В программе активно используются шаблоны, такие как std::vector, и шаблонные методы как input в классе UI. Они позволяют работать с универсальными контейнерами и указателями без написания дополнительного кода.

9 Динамическое выделение памяти и умные указатели. Используются умные указатели std::shared_ptr для управления памятью. Это предотвращает утечки памяти:

```
std::vector<std::shared_ptr<Account>> accounts;
```

В данной главе были рассмотрены основные программные конструкции, использованные в разработке проекта. Программа демонстрирует грамотное применение принципов ООП, обработку ошибок, работу с памятью через умные указатели, использование шаблонов и стандартной библиотеки C++.

Это делает систему гибкой, надежной и легко расширяемой для новых требований и задач.

Для описания системы были разработаны и использованы UML-диаграммы различных типов, таких как диаграмма классов, диаграмма последовательностей и диаграмма состояний. Диаграмма классов позволила наглядно отразить архитектуру системы, включая взаимосвязи между компонентами и их ключевые атрибуты и методы. Диаграмма последовательностей показала взаимодействие объектов во времени, что позволило уточнить логику ключевых сценариев работы. Диаграмма состояний отразила переходы между состояниями объекта при изменении его контекста. Эти модели значительно упростили процесс проектирования системы и позволили избежать потенциальных ошибок на этапе разработки.

Помимо UML-моделирования, для описания информационной структуры системы была разработана модель IDEF1X, которая показала структуру данных, используемых в программе, и их взаимосвязь. Это помогло детально спроектировать механизмы работы с данными, включая чтение, запись и манипуляцию информацией, представленной в файлах.

В программной реализации системы активно использовались базовые принципы объектно-ориентированного программирования, такие как инкапсуляция, наследование и полиморфизм, что обеспечило гибкость и модульность кода. Использование абстрактных классов, шаблонов и умных указателей улучшило безопасность и удобство работы с памятью. Реализация дружественных функций, перегрузка операторов и методов, а также применение пространств имен способствовали упрощению структуры программы и улучшению ее читаемости.

Кроме того, в системе реализована обработка ошибок средствами языка C++, что делает ее устойчивой к некорректным данным или действиям пользователя [11]. Использование потоков ввода-вывода позволило организовать удобное взаимодействие с файлами, содержащими тесты, пользователей и результаты. Программа демонстрирует эффективное управление ресурсами, динамическое выделение памяти и применение современных подходов к программированию.

Таким образом, созданная система сочетает в себе эффективные методы проектирования, современные подходы к реализации и интуитивно понятный интерфейс. В результате, разработанная автоматизированная система тестирования является функциональной, надежной и расширяемой, что соответствует требованиям, поставленным в начале работы.

3 ОПИСАНИЕ АЛГОРИТМА ЗАПУСКА ПРИЛОЖЕНИЯ, ЕГО ИСПОЛЬЗОВАНИЯ, РЕЗУЛЬТАТЫ РАБОТЫ, ТЕСТИРОВАНИЯ ОБРАБОТКИ ОШИБОК

3.1 Алгоритм запуска приложения

Для того, чтобы воспользоваться приложением на другом компьютере, изначально необходимо скомпилировать проект в исполняемый файл платформы Windows (.exe). Это автоматически делает встроенный в IDE Visual Studio 2022 компилятор clang. Затем, в подпапке x64 проекта можно найти исполняемый файл. Его использование является безопасным для другого компьютера.

Затем, вместе с исполняемым файлом необходимо передать все текстовые файлы, в которых хранятся данные о пользователях, тестах и результатах. Очень важно, чтобы исполняемый файл и текстовые данные лежали в одной директории.

Для запуска программы на другом компьютере, на нем должны быть установлены VC++ Runtime библиотеки (программы на C++, созданные посредством Visual Studio, требуют наличия этих библиотек). Они не нужны на нашем компьютере, так как устанавливаются во время установки Visual Studio, однако не устанавливаются по умолчанию на компьютере конечного пользователя [10].

Затем, достаточно просто запустить исполняемый файл и начать пользоваться приложением.

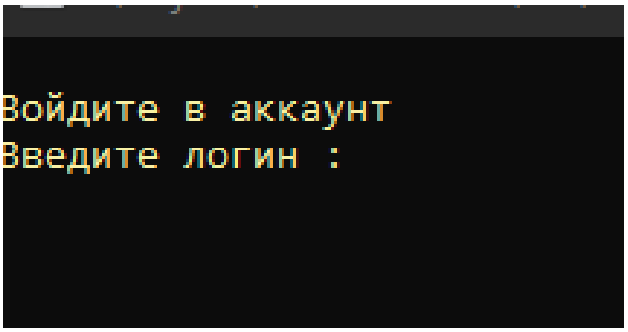

Рис. 3.1 – Скриншот окна авторизации

3.2 Руководство пользователя

В данной части представлено руководство пользователя программы автоматизированной системы тестирования студентов по языку C++. Здесь представлены инструкции по осуществлению входа или регистрации в систему. Особое внимание уделяется процедуре аутентификации, необходимой для доступа к функционалу программы.

Если учётных записей не будет обнаружено, то программа выдаст предупреждение и начнёт процедуру регистрации администратора (рис 3.2)

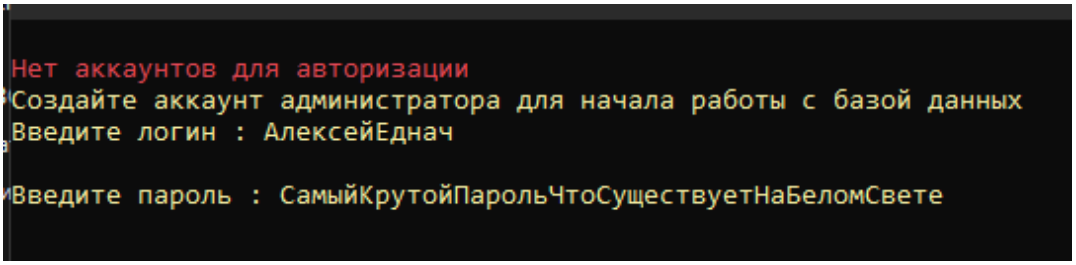

Рис. 3.2 – Скриншот процесса аварийной регистрации нового администратора

Также программа предложит зарегистрироваться и в случае, если у пользователя нету учётной записи. (рис 3.3)

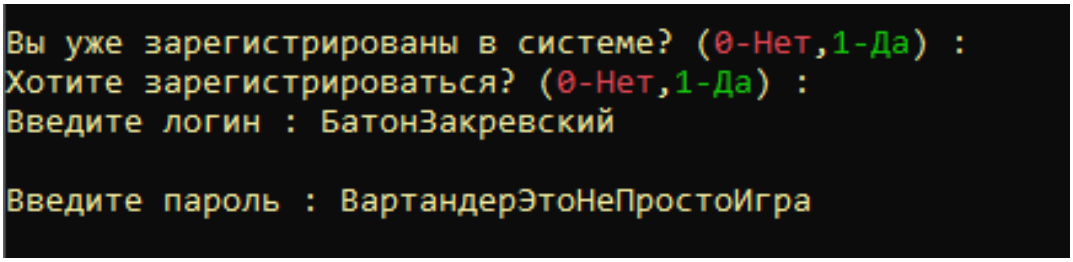

Рис. 3.3 – Скриншот процесса регистрации если нет учётной записи

Рассмотрим функционал администратора. Он имеет доступ ко всем функциям, кроме оценки объекта, поэтому ему доступно сразу два меню управления: объектами и учётными записями. (рисунки 3.4, 3.5 и 3.6)

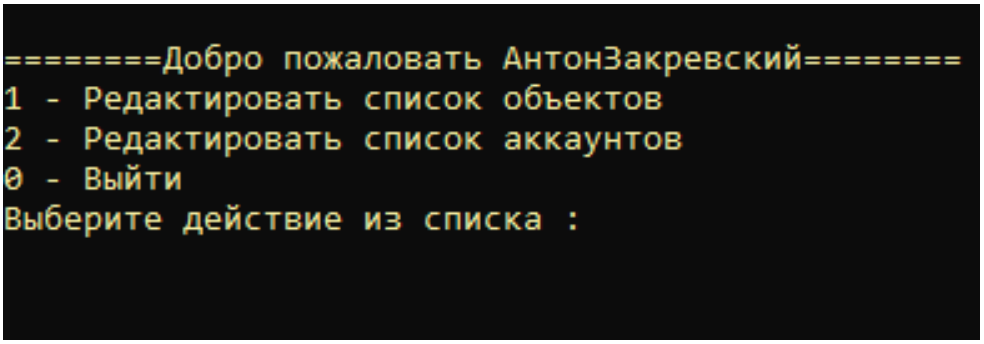

Рис. 3.4 – Скриншот меню администратора

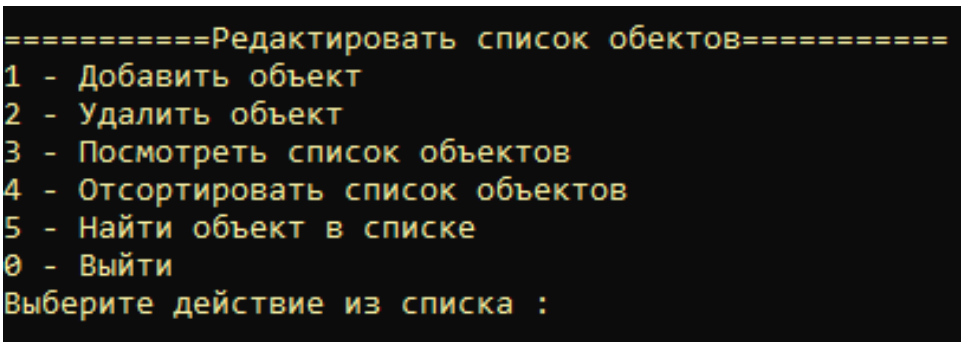

Рис. 3.5 – Скриншот меню редактирования списка объектов

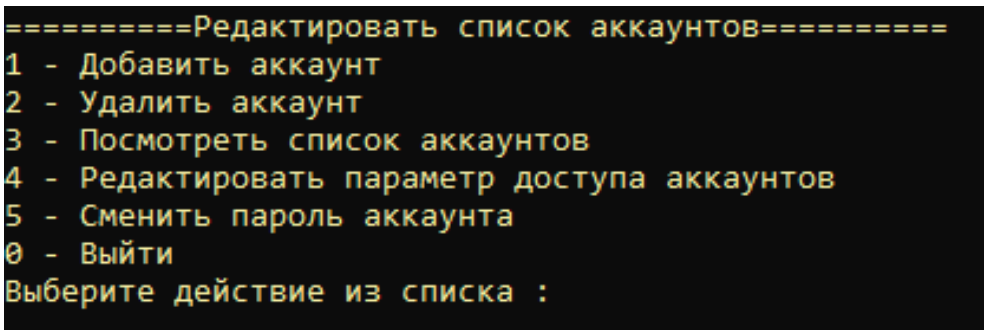

Рис. 3.6 – Скриншот меню редактирования учётных записей

Начнем с пункта «Редактировать список объектов». В этом пункте предоставлены возможности по добавлению, удалению, просмотру, сортировке и поиску объектов недвижимости. Нажав 0 можно вернуться назад в меню администратора.

Далее «Добавить объект». При выборе этого пункта меню администратор начинает процесс добавления нового объекта.

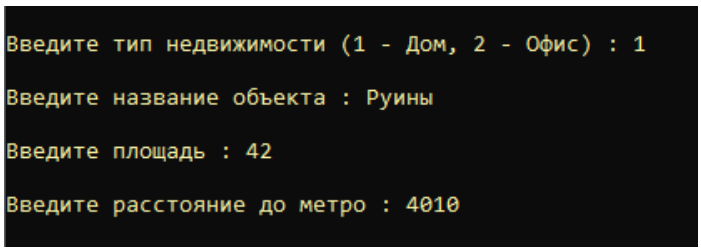

Рис. 3.7 – Скриншот пункта «Добавить объект»

Далее «Удалить объект». При выборе этого пункта меню администратор начинает процесс удаления нового объекта.

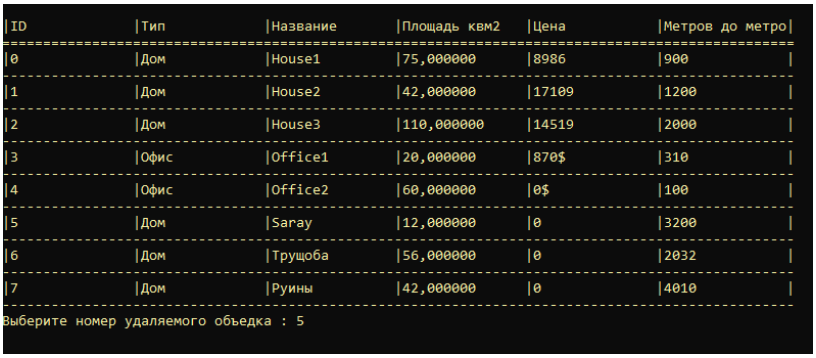

Рис. 3.7 – Скриншот пункта «Удалить объект»

Разберем пункт «Отсортировать список объектов». При выборе этого пункта меню пользователь может отсортировать список объектов по заданному параметру.

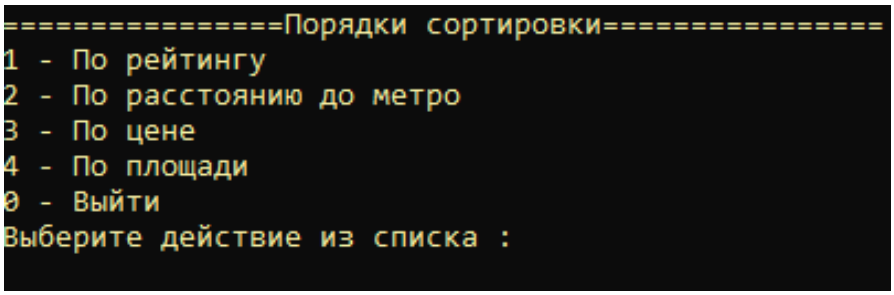

Рис. 3.8 – Скриншот пункта «Поиск и сортировка тестов»

Следующий пункт – «Найти объект в списку». При выборе этого пункта меню пользователь может отфильтровать объекты по заданному параметру.

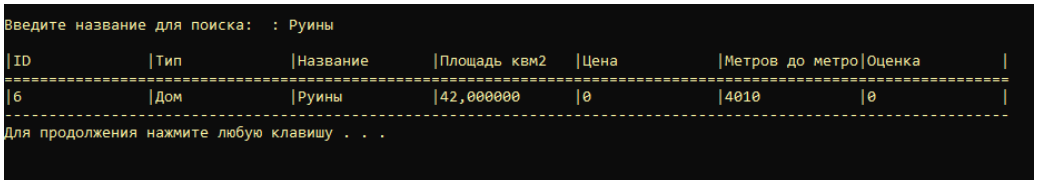

Рис. 3.9 – Скриншот пункта «Найти по имени»

При выборе пункта «Посмотреть список объектов» внезапно, производится отображения списка объектов.

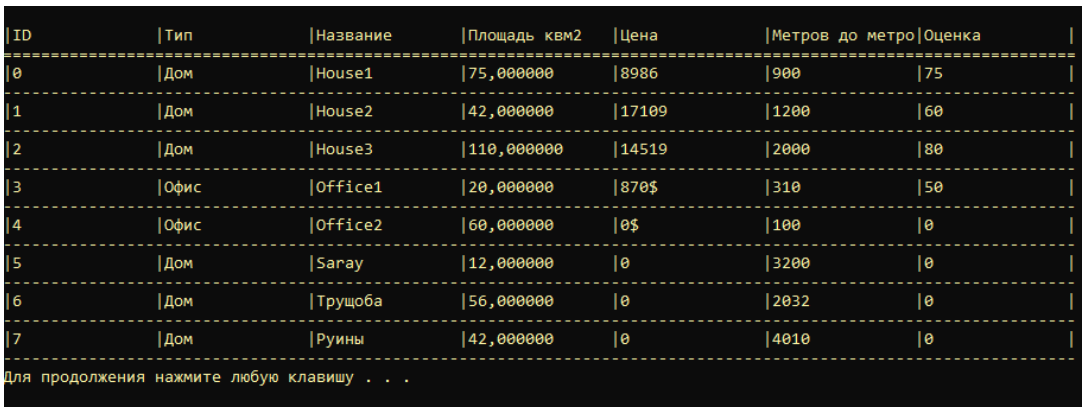

Рис. 3.10 – Скриншот пункта «Посмотреть список объектов»

Рассмотрим меню управления учётными записями. Начнём с пункта «Добавить аккаунт». При выборе этого пункта меню администратор начинает процесс добавления новой учётной записи.

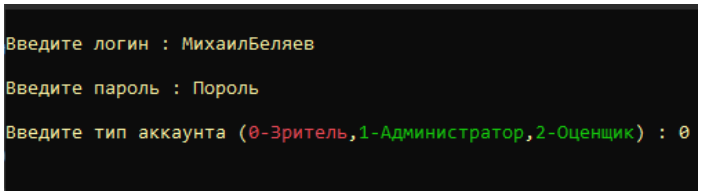

Рис. 3.11 – Скриншот пункта «Добавить аккаунт»

Далее «Удалить аккаунт». При выборе этого пункта меню администратор начинает процесс удаления аккаунта.

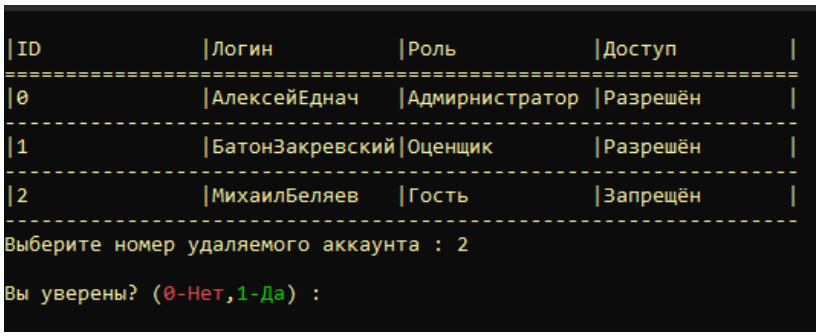

Рис. 3.12 – Скриншот пункта «Удалить аккаунт»

Разберем пункт «Редактировать параметр доступа аккаунтов». При выборе этого пункта меню пользователь может предоставить или отобрать доступ к базе данных у учётной записи.

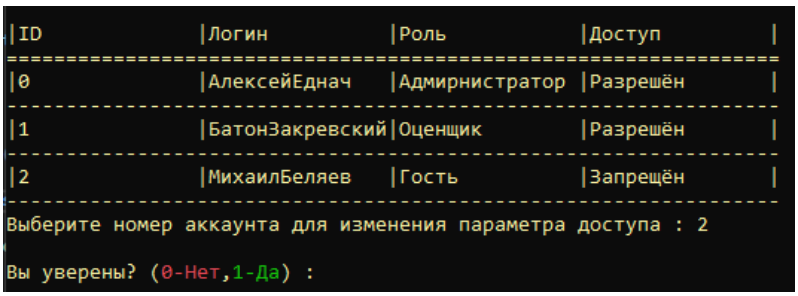

Рис. 3.13 – Скриншот пункта «Редактировать параметр доступа аккаунтов»

Следующий пункт – «Сменить пароль аккаунта». При выборе этого пункта меню администратор может изменить пароль выбранной учётной записи.

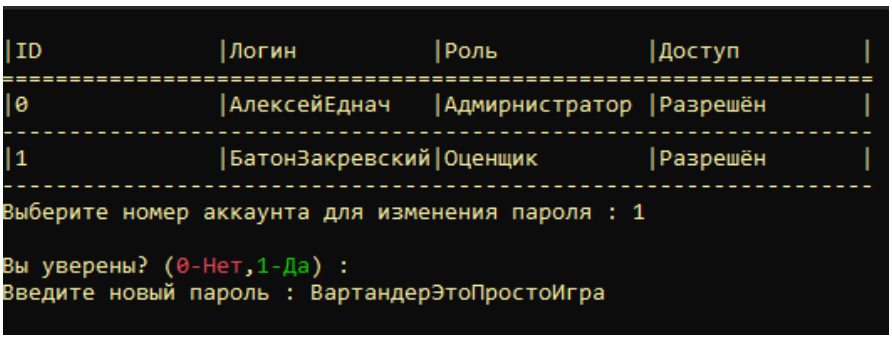

Рис. 3.14 – Скриншот пункта «Найти по имени»

При выборе пункта «Посмотреть список объектов» внезапно, производится отображения списка объектов.

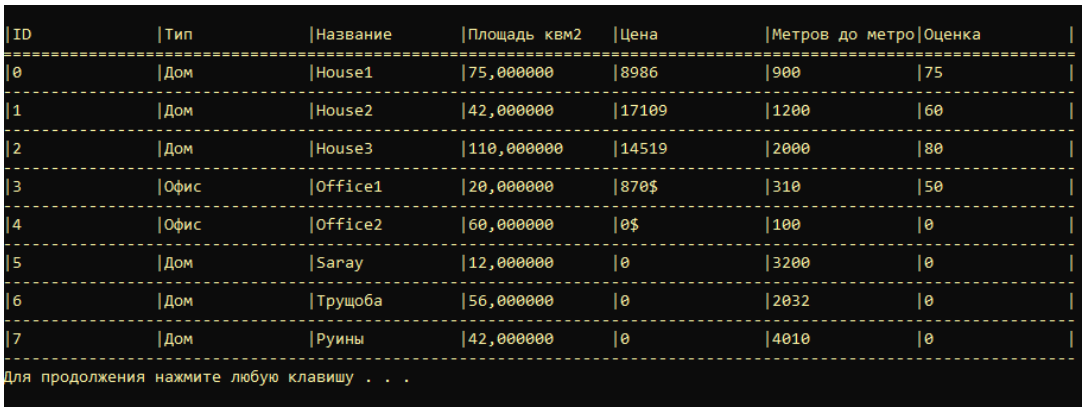

Рис. 3.15 – Скриншот пункта «Посмотреть список объектов»

Рассмотрим функционал гостя. При авторизации от имени студента на экране будет отображено меню возможных действий, включающее в себя такие пункты как «Посмотреть список объектов», «Посмотреть список

объектов по рейтингу», «Поиск в списке объектов», «Отсортировать список объектов» и «Сменить пароль аккаунта».

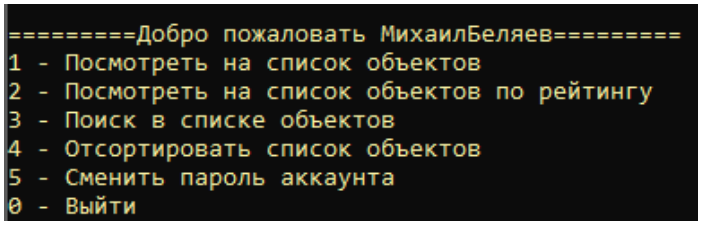

Рис. 3.16 – Скриншот меню гостя

Эти пункты работают по аналогии с вышерассмотренными, поэтому мы можем перейти к меню оценщика.

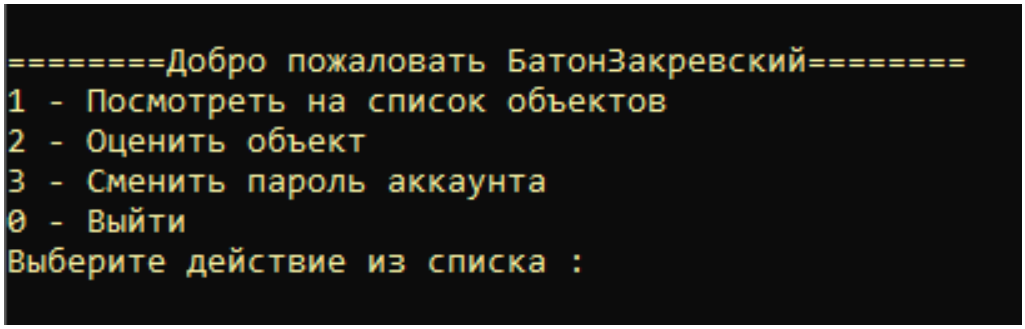

Рис. 3.17 – Скриншот меню оценщика

Далее «Оценить объект». Самый важный пункт программы. Оценщику предложат оценить объект, после чего будет рассчитана его стоимость и рейтинг.

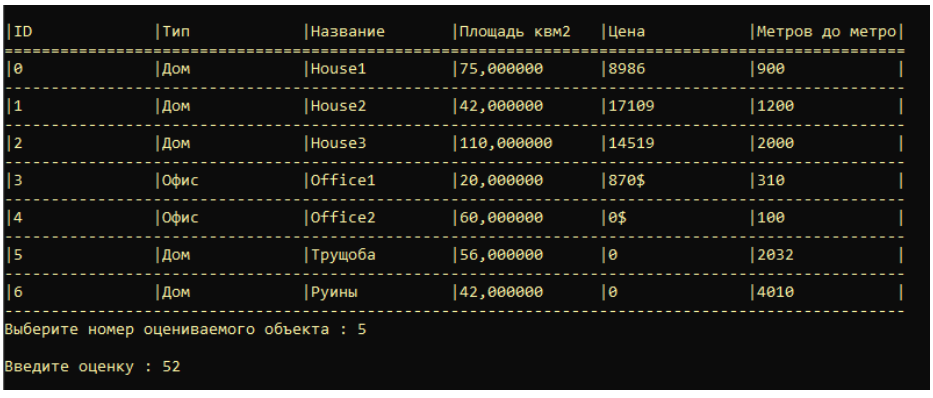

Рис. 3.18 – Скриншот пункта «Оценить объект»

После завершения работы с программой предоставляется возможность выйти из программы, выбрав пункт «Выйти». Выход с помощью этого пункта предотвращает неправильное сохранение данных, их потерю и какие-либо

дальнейшие ошибки при использовании разработанного ПО. После этого необходимо подтвердить своё намерение выйти из программы.

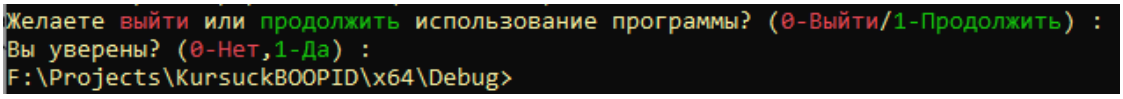

Рис. 3.19 – Скриншот выхода из программы

3.3 Тестирование работы приложения

В процессе тестирования работы приложения была проведена проверка обработки исключительных ситуаций [12]. Были смоделированы различные ошибки ввода данных, отсутствия или повреждения файлов, некорректных действий пользователя и проверена реакция программы [14].

Рассмотрим обработанные ошибки.

Ошибка: Некорректный ввод в меню. Если пользователь вводит нечисловое значение или некорректный пункт меню, программа выводит сообщение об ошибке и предлагает повторить ввод.

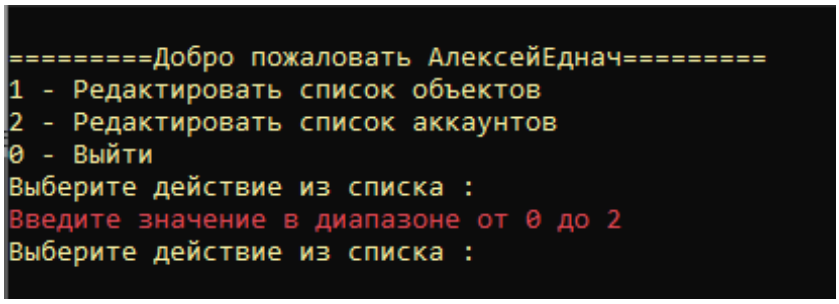

Рис. 3.20 – Обработка ошибки «Некорректный ввод в меню»

Ошибка: Неправильный пароль. Возникает, если несколько раз подряд ввести неверный пароль во время авторизации.

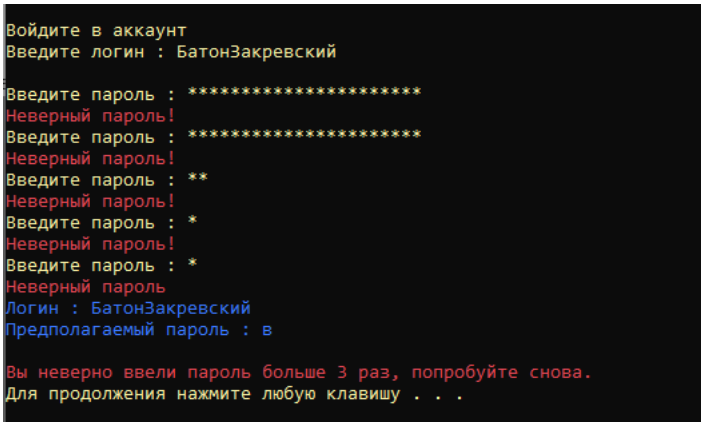

Рис. 3.21 – Обработка ошибки «Неправильный пароль»

Ошибка: Неверный логин. Если учётной записи с таким логином не существует, то программа выдаст ошибку.

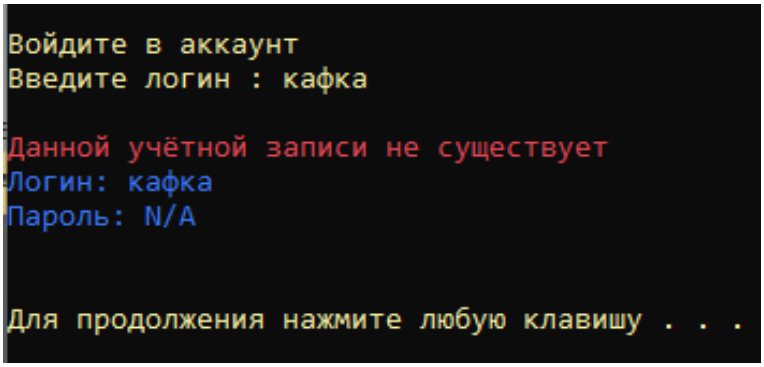

Рис. 3.22 – Обработка ошибки «Неверный логин»

Ошибка: Взаимодействие с пустым списком объектов. Если пользователь попробует посмотреть, отсортировать или найти что-либо в пустой базе данных, то программа выдаст ошибку. Это касается и функций редактирования списка администратором.

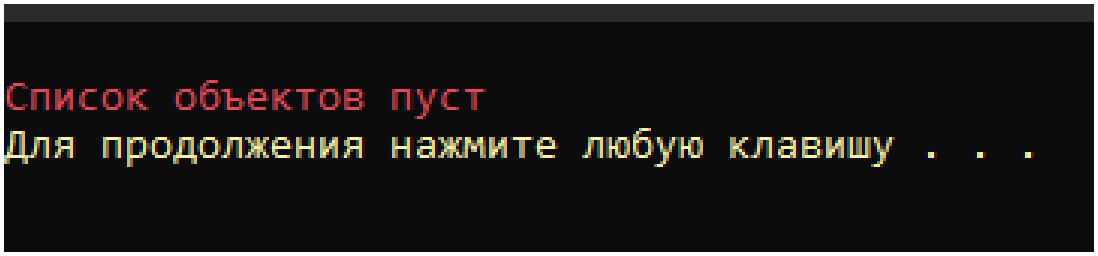

Рис. 3.23 – Обработка ошибки «Взаимодействие с пустым списком объектов»

Ошибка: Неверный формат данных. При создании объектов недвижимости или учётных записей программа проверяет корректность данных. Если ввод некорректен, будет выведена ошибка.

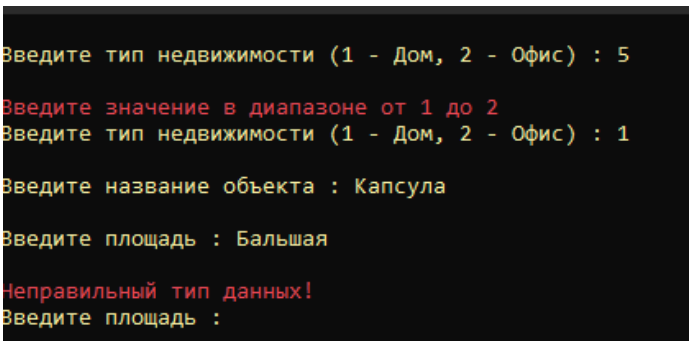

Рис. 3.24 – Обработка ошибки «Неверный формат данных»

Ошибка: Удаление или отказ в доступе своей учётной записи. Если администратор попробует отобрать у своей учётной записи доступ к базе данных или удалить её, то программа не позволит это сделать.

Тестирование показало, что приложение устойчиво к некорректным действиям пользователя или отсутствию данных. Программа обрабатывает ошибки корректно, информирует пользователя о возникших проблемах и предлагает повторить действия [15]. Это обеспечивает удобство и безопасность использования.

Проведённое тестирование подтвердило, что приложение полностью соответствует требованиям:

– реализованы основные алгоритмы и сценарии использования;

– пользователи информируются об ошибках с помощью удобных и понятных сообщений;

– система устойчива к некорректному вводу, отсутствию данных и повреждённым файлам.

Эти результаты свидетельствуют о высокой надёжности и эффективности разработанной системы автоматизированного тестирования.

А это означает, что разработка программного обеспечения прошла успешно и программа готова к эксплуатации в том числе коммерческими организациями.

Тестирование и отладка безусловно важные пункты в разработке программы.

Тестирование проводилось на всех этапах и на протяжении всего времени, что я разрабатывал программу.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы был проведён анализ и разработка программы для автоматизированной оценки объектов недвижимости. Разработанная система позволила решить задачи, связанные с обработкой, хранением и представлением данных о недвижимости, её оценках и пользователях системы. Программа полностью соответствует поставленным требованиям, обеспечивая поддержку множества пользователей, удобное управление данными об объектах и оценщиках, а также надёжность при работе с файлами и устойчивость к ошибкам.

В первом разделе работы была изучена предметная область, связанная с процессом оценки недвижимости, включая ключевые критерии оценки, такие как площадь объекта, расстояние до метро, цена за квадратный метр и другие важные параметры. Были исследованы основные методы оценки, что позволило построить модель IDEF0 для детального описания основных процессов системы. Модель отразила взаимодействие между различными элементами системы, что способствовало более точному пониманию и оптимизации её работы.

Во втором разделе была разработана архитектура системы, включая создание модели IDEF1X, описывающей структуру файлов и их взаимосвязь, а также проектирование структуры программы с использованием UML-диаграмм. Диаграмма классов помогла определить основные компоненты системы и их взаимосвязь, диаграмма последовательностей – проанализировать взаимодействие объектов, а диаграмма состояний – описать жизненный цикл ключевых процессов, таких как оценка недвижимости. Эти модели позволили задать чёткую архитектуру приложения, улучшив её надёжность и возможность расширения в будущем.

В третьем разделе проводилось тестирование приложения, включая проверку обработки исключительных ситуаций. Были рассмотрены ошибки, связанные с некорректным вводом данных, отсутствием или повреждением файлов, а также дублированием записей. Программа продемонстрировала высокую устойчивость и способность информировать пользователя о возникших проблемах, что повышает удобство и безопасность её использования.

В результате выполнения курсовой работы была достигнута поставленная цель – разработка автоматизированной системы оценки недвижимости. Созданная программа упростила процесс расчёта стоимости объектов, сократила трудозатраты, связанные с обработкой данных, и повысила объективность оценок. Разработанное программное средство успешно интегрирует современные технологии программирования и проектирования, что делает его готовым к дальнейшему использованию и возможному расширению.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Иванов, А.С. Основы автоматизации оценки недвижимости / А.С. Иванов. – Москва: Юрайт, 2019. – С. 15–22.

[2] Carter, J. Real Estate Valuation Automation: Methods and Implementation / J. Carter. – London: Springer, 2021. – P. 45–52.

[3] Petrov, D.V. Modern Approaches to Property Valuation / D.V. Petrov, K.A. Smirnov. – New York: Wiley, 2022. – P. 40–46.

[4] Moodle Documentation [Электронный ресурс]. – Режим доступа: https://docs.moodle.org. – Дата доступа: 30.09.2024.

[5] Blackboard Help Center [Электронный ресурс]. – Режим доступа: https://help.blackboard.com. – Дата доступа: 30.09.2024.

[6] Карпов, Д.В. Платформы для дистанционного обучения: сравнительный анализ / Д.В. Карпов, D.R. Krathwohl. – Москва : МГПУ, 2020. – С. 57–62.

[7] IDEF0. Знакомство с нотацией и пример использования [Электронный ресурс]. – Режим доступа: https://trinion.org/blog/idef0. – Дата доступа: 05.10.2024.

[8] Rumbaugh, J. Unified Modeling Language User Guide / J. Rumbaugh, D.R. Krathwohl. – New York : Addison-Wesley, 2005. – P. 145–155.

[9] Fowler, M UML Distilled: A Brief Guide to the Standard Object Modeling Language / M Fowler, D.R. Krathwohl. – New York : Addison-Wesley, 2004. – P. 50–55.

[10] Павловская, Т.А. C/C++. Программирование на языке высокого уровня: Учебник для вузов / Т.А. Павловская, D.R. Krathwohl. – Санкт-Петербург : Питер, 2012. – С. 321–330.

[11] Шилдт, Г. Язык программирования C++ для начинающих / Г. Шилдт, М. Вильямс. – Санкт-Петербург : Питер, 2021. – С. 89–95.

[12] Stroustrup, B. The C++ Programming Language / B. Stroustrup, M. Вильямс. – New York : Addison-Wesley, 2013. – P. 400–410.

[13] Google C++ Style Guide [Электронный ресурс]. – Режим доступа: https://google.github.io/styleguide/cppguide.html. – Дата доступа: 12.11.2024.

[14] Meyers, S. Effective C++: 55 Specific Ways to Improve Your Programs and Designs / S. Meyers, M. Вильямс. – New York : Addison-Wesley, 2005. – P. 210–220.

[15] Технология программирования [Электронный ресурс]. – Режим доступа: https://studfile.net/preview/1195601. – Дата доступа: 29.11.2024.

ПРИЛОЖЕНИЕ А
(обязательное)
Отчет о проверке на заимствования в системе «Антиплагиат»

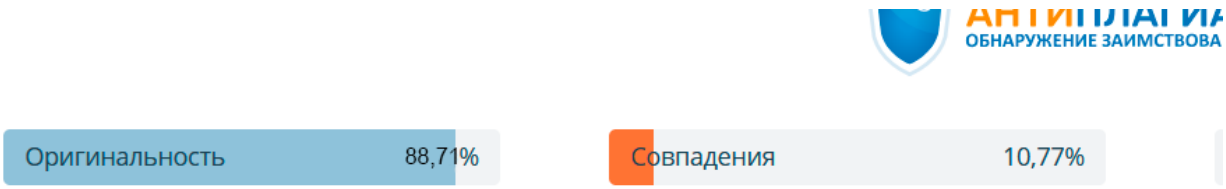

Рисунок А.1 – Проверка на Антиплагиат

ПРИЛОЖЕНИЕ Б
(обязательное)
Листинг кода алгоритмов, реализующих основную бизнес-логику

Функция main(), также является функцией входа и выхода из программы:

```
int main() {


	UI ui(UI::colors::Yellow);
	Database database;

	ui.setDatabase(&database);

	while (true) {
		system("cls");

		database.login();

		switch (database.whoIsNow()) {
		case 0:
			ui.spectatorMenu();
			break;
		case 1:
			ui.adminMenu();
			break;
		case 2:
			ui.judgeMenu();
			break;
		}
		bool choice = ui.inputRangeInstant("Желаете &2выйти&0 или &4продолжить &9использование программы? (&20-Выйти&0/&41-Продолжить&0)",0,1);

		if (!choice) {
			if (!ui.confirm()) continue; /// XD
			return 0;
		}
	}

	return 0;
}
```

Функция считывания объектов из файла:

```
void readObjectsFromFile()
{
	objects.clear();
	fstream file("objects.txt", ios::in);

	if (file.eof() || !file.is_open())
	{
		file.close();
		return;
	}

	string temp;
```

```
            getline(file, temp, '\n');
            int size = stoi(temp);

            for (int i = 0; i < size; )
            {
                  getline(file, temp, ' ');
                  if (temp == "h" || temp == "o") {
                        i++;
                        string objectName,objectType = temp;
                        int squareMeters, finalPrice, rate, metersToMetro;
                        getline(file, objectName, ' ');
                        getline(file, temp, ' ');
                        squareMeters = stoi(temp);
                        getline(file, temp, ' ');
                        finalPrice = stoi(temp);
                        getline(file, temp, '\n');
                        metersToMetro = stoi(temp);
                        if (objectType == "h")
                        objects.emplace_back(new              House(objectName,
squareMeters, finalPrice, metersToMetro, 27000));
                        else if (objectType == "o")
                              objects.emplace_back(new      Office(objectName,
squareMeters, finalPrice, metersToMetro, 27000));
                  }
                  else if (temp == "r") {
                        string login;
                        int rating;
                        getline(file, login, ' ');
                        getline(file, temp, '\n');
                        rating = stoi(temp);
                        objects.at(objects.size()    -    1)->addRating(login,
rating);

                        objects[i - 1]->calculatePrice();
                  }
            }

            file.close();
      }
```

Функция считывания пользователей из файла:

```
      void readAccountsFromFile()
      {
            accounts.clear();
            fstream file("accounts.txt", ios::in);
            if (file.eof() || !file.is_open())
            {
                  file.close();
                  return;
            }
            string temp;
            getline(file, temp, '\n');
            int size = stoi(temp);
            for (int i = 0; i < size; i++)
                  {
                  string login, password;
                  getline(file, login, ' ');
                  getline(file, password, ' ');
```

```
                getline(file, temp, ' ');
                int accountType = stoi(temp);
                getline(file, temp, '\n');
                bool access = stoi(temp);
                accounts.emplace_back(new                    Account(login,
enDecrypt(password), accountType, access));
                }
          file.close();
    }
```

Функция добавления нового пользователя:

```
    void Database::addAccount(int type) {
          string login;
          while (true) {
                login = ui->input<string>("Введите логин");
                if (!checkLogin(login)) break;
                ui->printColor("&2Этот пользователь уже существует");
          }
          string password = ui->input<string>("Введите пароль");

          int accountType = type;
          if (type < 0 || type > 2)
                accountType = ui->input<int>("Введите тип аккаунта (&20-
Зритель&0,&41-Администратор&0,&42-Оценщик&0)");

          if (isLoggedIn()) accounts.emplace_back(new Account(login,
password, accountType, true));
          else accounts.emplace_back(new Account(login, password,
accountType, false));


          writeAccountsToFile();
    }
```

Функция рассчёта стоимости объекта недвижимости для офиса и дома:

```
    void calculatePrice() override {
          rating.calculateRating();
          finalPrice = (((double)rating.rate/100.0) * (GOSTSquareMetersPrice
* squareMeters)) / metersToMetro;
    }
```

Функция отображения детальной информации о недвижимости:

```
    void showDetailObjectInfo() {
          if (objects.empty())
          {
                ui->printColor("&2Список объектов пуст");
                return;
          }

          vector<string> table;

          SetConsoleCP(1251);
```

```
	table.emplace_back("ID");
	table.emplace_back("Тип");
	table.emplace_back("Название");
	table.emplace_back("Площадь квм2");
	table.emplace_back("Цена");
	table.emplace_back("Метров до метро");
	table.emplace_back("Оценка");

	ui->printTable(table, 15, true);

	for (int i = 0; i < objects.size(); i++) {
		table.clear();
		table.emplace_back(to_string(i));
		objects.at(i)->getDetailInfo(table);
		ui->printTable(table, 15);
	}
	SetConsoleCP(866);
}
```

Функция получения общей информации объекта недвижимости для офиса:

```
void getGenericInfo(vector<string>& row) override {
	row.emplace_back("Офис");
	row.emplace_back(objectName);
	row.emplace_back(to_string(squareMeters));
	row.emplace_back(to_string(finalPrice) + "$");
	row.emplace_back(to_string(metersToMetro));
}
```

Функция записи учётных записей в файл:

```
void writeAccountsToFile()
{
	fstream file("accounts.txt", ios::out);
	file << accounts.size() << '\n';
	file.close();

	for (int i = 0; i < accounts.size(); i++)
		accounts.at(i)->addToFile();
}
```

Функция записи объектов недвижимости в файл:

```
void writeObjectsToFile()
{
	fstream file("objects.txt", ios::out);
	file << objects.size() << '\n';
	file.close();

	for (int i = 0; i < objects.size(); i++)
		objects.at(i)->addToFile();

}
```

Функция записи объектов недвижимости в файл:

```
void writeObjectsToFile()
{
	fstream file("objects.txt", ios::out);
	file << objects.size() << '\n';
	file.close();

	for (int i = 0; i < objects.size(); i++)
		objects.at(i)->addToFile();

}
```

Функция записи дома в файл:

```
void addToFile() const override
{
	fstream file("objects.txt", ios::app);

	file << "h " << objectName << ' ' << squareMeters << ' ' <<
finalPrice << ' ' << metersToMetro << '\n';

	file.close();

	rating.addToFile();
}
```

Функция оценки объекта:

```
void Database::rateObject()
{
	if (objects.empty())
	{
		ui->printColor("&2Нечего оценивать");
		return;
	}
	showGenericObjectInfo();

	int id = ui->inputRange<int>("Выберите номер оцениваемого объекта",
0, objects.size() - 1);
	int rating = ui->inputRange<int>("Введите оценку", 0, 100);

	objects.at(id)->addRating(getLogin(), rating);
	objects.at(id)->calculatePrice();

	writeObjectsToFile();
}
```

Функция отображения меню гостя:

```
void spectatorMenu() {
	while (true) {
		if (!db->isLoggedIn()) return;
		system("cls");
		printHeader("&0Добро пожаловать " + db->getLogin(), 50);
```

```
                printColor("1 - Посмотреть на список объектов");
                printColor("2 - Посмотреть на список объектов по рейтингу");
                printColor("3 - Поиск в списке объектов");
                printColor("4 - Отсортировать список объектов");
                printColor("5 - Сменить пароль аккаунта");
                printColor("0 - Выйти");
                int choice = inputRangeInstant("Выберите действие из списка",
0, 5);
                system("cls");
                switch (choice) {
                case 0:
                      db->deauthorize();
                      return;
                      break;
                case 1:
                      db->showGenericObjectInfo();
                      pressAnyButton();
                      break;
                case 2:
                      db->showRateObjectInfo();
                      pressAnyButton();
                      break;
                case 3:
                      searchMenu();
                      break;
                case 4:
                      sortMenu();
                      break;
                case 5:
                      system("cls");
                      db->changePassword();
                      break;
                }
          }
    }
```

Функция отображения меню судьи:

```
    void judgeMenu() {
          while (true) {
                if (!db->isLoggedIn()) return;
                system("cls");
                printHeader("&0Добро пожаловать " + db->getLogin(), 50);
                printColor("1 - Посмотреть на список объектов");
                printColor("2 - Оценить объект");
                printColor("3 - Сменить пароль аккаунта");
                printColor("0 - Выйти");
                int choice = inputRangeInstant("Выберите действие из списка",
0, 3);
                system("cls");
                switch (choice) {
                case 0:
                      db->deauthorize();
                      return;
                      break;
                case 1:
                      db->showDetailObjectInfo();
                      pressAnyButton();
```

```
                        break;
                  case 2:
                        system("cls");
                        db->rateObject();
                        break;
                  case 3:
                        system("cls");
                        db->changePassword();
                        break;
                  }
            }
      }
```

Функция отображения меню судьи:

```
      void adminMenu() {
            while (true) {
                  if (!db->isLoggedIn()) return;
                  system("cls");
                  printHeader("&0Добро пожаловать " + db->getLogin(), 50);
                  printColor("1 - Редактировать список объектов");
                  printColor("2 - Редактировать список аккаунтов");
                  printColor("0 - Выйти");
                  int choice = inputRangeInstant("Выберите действие из списка",
0, 2);
                  system("cls");
                  switch (choice) {
                  case 0:
                        db->deauthorize();
                        return;
                        break;
                  case 1:
                        objectsEditing();
                        break;
                  case 2:
                        accountsEditing();
                        break;
                  }
            }
      }
```

Функция ввода данных любого типа:

```
      template<typename T>
      T input(string text, bool newLine = true) {
            T input{};
            while (true) {
                  printColor(text + " : ", newLine, true);
      #ifdef SOUND
                  cout << (char)7;
      #endif
                  SetConsoleCP(1251);
                  cin >> input;
                  SetConsoleCP(866);
                  if (cin.fail()) {
                        printColor("&2Неправильный тип данных!");
                        cin.clear();
```

```
                cin.ignore(100500, '\n');
            }
            else {
                return input;
            }
        }
}
```

Функция вывода данных любого типа:

```
template <typename T>
void print(T data, bool newLine = true) {
      SetConsoleCP(1251);
      if (newLine)  cout << '\n';
      cout << data;
      SetConsoleCP(866);
}
```

Функция вывода текста с цветным форматированием:

```
void printColor(string str, bool newLine = true, bool animation = false)
{
      SetConsoleCP(1251);
      if (newLine)  cout << '\n';
      bool flag = false;
      for (char ch : str) {
            if (animation) Sleep(ANIMATIONSPEED);
#ifdef SOUND
            cout << (char)7;
#endif
            if (!flag)
                  if (ch != '&')
                        cout << ch;
                  else flag = true;
            else {
                  setColor(static_cast<UI::colors>((ch - 48)));
                  flag = false;
            }
      }
      SetConsoleCP(866);
      setColor(defaultColor);
}
```