Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Программирование на языках высокого уровня

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проектированию
на тему
ИНФОРМАЦИОННАЯ СИСТЕМА ПО ПРОДАЖЕ НЕДВИЖИМОСТИ

БГУИР КП 1-40 02 01 123ПЗ

Студент: гр. 250501 Снитко Д. А.

Руководитель: старший преподаватель
каф. ЭВМ Ковальчук А. М.

Минск 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

C++ - это компилируемый язык программирования, который обладает структурой и ориентирован на объекты. Несмотря на свой долгий возраст, C++ по-прежнему остается одним из самых востребованных и популярных языков программирования в мире. Этот язык поддерживает различные стили программирования, такие как процедурное, абстрактное, объектно-ориентированное и обобщенное программирование, предоставляя разработчику свободу выбора стиля для написания программы.

C++ также есть библиотеки, они представляют собой наборы предопределенных кодов, функций и классов, которые разработчики могут использовать для упрощения различных задач в программировании. Эти библиотеки предоставляют готовые решения для широкого спектра задач, таких как обработка данных, работа с графическим интерфейсом, работа с сетью, многопоточность и многое другое. Вместо того чтобы писать код с нуля, разработчики могут воспользоваться библиотеками для экономии времени и улучшения производительности своих программ. Библиотеки в C++ могут быть частью стандартной библиотеки (STL) или сторонними библиотеками, разработанными сообществами или сторонними организациями.

Одной из ключевых характеристик C++ является его высокая производительность. Важной особенностью этого языка является его объектно-ориентированный аспект, включающий в себя множество функций, таких как конструкторы, деструкторы, перегрузка операторов и создание классов.

Для поддержки принципов объектно-ориентированного программирования, включая инкапсуляцию, полиморфизм, наследование и абстракцию, C++ и другие объектно-ориентированные языки предоставляют соответствующие средства. Инкапсуляция защищает данные и код от внешних воздействий и ошибок. Полиморфизм позволяет обращаться к различным действиям через единый интерфейс. Наследование позволяет объектам наследовать свойства других объектов. Абстракция позволяет использовать только необходимые характеристики объекта, представляя его минимальным набором полей и методов, подходящим для конкретной задачи.

Информационная система (ИС) - это система, предназначенная для хранения, поиска и обработки информации. Результатом работы информационных систем является информационная продукция, такая как документы, базы данных и информационные услуги.

Сегодня C++ активно используется при разработке операционных систем, драйверов устройств и разнообразных прикладных систем в современном мире: разработке игр, научных и инженерных вычислениях, финансовых приложениях, системах управления базами данных, встраиваемых системах (микроконтроллеры, интернет вещей).

3 Постановка задачи

Программа должна иметь удобный пользовательский интерфейс с необходимыми пунктами меню. Информация должна храниться в нескольких файлах, связанных определенным образом. Необходимо хранить следующие сведения: данные о квартирах, частном секторе, нежилых помещениях, коммерческой недвижимости, а также о информации по заданным признакам. Реализовать добавление, удаление, редактирование данных, поиск информации по заданным признакам. При реализации функций добавление, удаление, редактирование данных необходимо предусмотреть операцию отмены последних действий (более одного).

Разработать иерархию классов с использованием наследования (не меньше 3-х уровней наследования). Разработать и использовать классы контейнеров, итераторов и алгоритмов(свои и STL). Производить обработку исключительных ситуаций. При реализации операции редактирования, добавления, удаления информации необходимо предусмотреть операцию отмены последних действий.

4 Обзор литературы

4.1 Обзор методов и алгоритмов решения поставленной задачи

Для реализации данной информационной системы было выбран язык программирования C++.

Каждая команда языка высокого уровня эквивалентна нескольким командам в машинных кодах, поэтому программы, написанные на языках высокого уровня, более компактны, чем аналогичные программы в машинных кодах.

Язык C++ основан на объектно-ориентированной парадигме (ООП). ООП — это парадигма разработки программных систем, в которой приложения состоят из объектов. *Объекты* — это сущности, у которых есть свойства и поведение. Обычно объекты являются экземплярами какого-нибудь класса. *Свойства* — это данные, которые связаны с конкретным объектом. *Методы* — это функция или процедура, принадлежащая какому-то классу или объекту и описывающая их поведение.

Основные принципы структурирования в случае ООП связаны с различными аспектами базового понимания предметной задачи, которое требуется для оптимального управления соответствующей моделью:

- *Абстракция* для выделения в моделируемом предмете важного для решения конкретной задачи по предмету, в конечном счёте — контекстное понимание предмета, формализуемое в виде класса;
- *Инкапсуляция* для быстрой и безопасной организации собственно иерархической управляемости: чтобы было достаточно простой команды «что делать», без одновременного уточнения как именно делать, так как это уже другой уровень управления;
- *Наследование* для быстрой и безопасной организации родственных понятий: чтобы было достаточно на каждом иерархическом шаге учитывать только изменения, не дублируя всё остальное, учтённое на предыдущих шагах;
- *Полиморфизм* для определения точки, в которой единое управление лучше распараллелить или наоборот — собрать воедино.

Выбор языка C++ обусловлен высокой производительностью, описанными выше свойствами и методами, что позволит максимально раскрыть потенциал языка, а так же значительно ускорить разработку, сделать код более читаемым.

5 Функциональное проектирование

В данном разделе пояснительной записки описываются входные и выходные данные программы, используемые классы и методы.

Основные части программы:

- файловая система;
- недвижимость;
- интерфейс;
- ввод данных;
- исключительные ситуации.

5.1 Структура входных и выходных данных

Таблица 5.1 - файл дом *House.txt*

Название	ID	Актуально	Стоимость	Площадь	Адрес	Водоемы	Деревья	Этажи	Комнаты	Количество парковочных мест
Домик	1	Да	85000	180	Минск, Приозерная ул., д. 19	Нет	Да	2	7	2

Таблица 5.2 - файл квартира *Flat.txt*

Название	ID	Актуально	Стоимость	Площадь	Адрес	Этаж	Комнаты	Балкон	Парковка
Квартирка	2	Да	28000	40	Минск, Приозерная ул., д. 45, кв. 55	5	2	Да	Да

Таблица 5.3 - файл коммерческое помещение *Commercial.txt*

Назва-ние	ID	Акту-ально	Стои-мость	Пло-щадь	Адрес	Этаж	Комнаты	Балкон
	3	Да	28000	40	Минск, Приозерная ул., д. 45, кв. 55	5	2	Да

Тип	Парковочные места							
Офис	5							

Таблица 5.4 - файл пустой участок *Empty.txt*

ID	Пригодность почвы	Постройки						
38	Да	Нет						

Таблица 5.5 - файл участок *Piece.txt*

ID	Водоем	Газон	Коммуникации					
38	Нет	Да	Электричество, газ, во-да, канализация					

Таблица 5.6 - файл контактные данные *ContactDetails.txt*

ID	Телефон	Email	Telegram					
38	+375688888	bam@gmail.com	@mishisha					

5.2 Разработка диаграммы классов

Диаграмма классов представлена в приложении А

5.3 Описание классов

Класс `Storage`.

Поля класса:

— `filepath = "id.txt"`. Данная константа содержит имя файла, который хранит в себе свободное значение ID;

— поле `freeId`. Содержит свободное значение уникального номера (ID);

— `vector<House>`. Список, который хранит в себе объекты класса `House`;

— `vector <Empty>`. Список, который хранит в себе объекты класса `Empty`;

— `vector <Flat>`. Список, который хранит в себе объекты класса `Flat`;

— `vector <Parking>`. Список, который хранит в себе объекты класса `Parking`;

— `vector <Commercial>`. Список, который хранит в себе объекты класса `Commercial`;

Методы класса:

— `Storage()`: конструктор по умолчанию, используя конструкцию `try{…}catch…` вызывает метод `uploadAppFile()`, а после вызывает метод `loadAll()`;

— `uploadAppFile()`: этот метод открывает файл `filepath` и заносит значение из файла в `freeId`. В случае отсутствия файла, он его создаёт и заносит туда 0;

— `saveAppFile()`: этот метод открывает файл `filepath` (в случае отсутствия он его создаёт) и записывает в него значение поля `freeId`;

— `upload(T className)`: этот шаблонный метод создаёт файл по типу `*id*.txt`, где `id` - уникальный номер класса который был передан(`className`). После он записывает значение полей класса в только что созданный файл;

— `load(unsigned int id)`: этот метод используя полученное значение id ищет среди файлов необходимый, после считывает файл, анализирует и идентифицирует. Потом на основе полученной информации создаёт копию этого объекта в динамической памяти;

— `loadAll()`: этот метод циклически вызывает `load(unsigned int id)` пока не будут загружены все существующие объекты;

— `addClass(T className)`: этот метод полученный объект загружает и в локальную, и в динамическую память;

— `addHouse()`: этот метод используя запросы из класса `ObjectManager` создаёт объект типа `House`, затем вызывает метод `addClass(T className)`, где в качестве параметра передаётся только что созданный объект;

— `addEmpty()`: этот метод используя запросы из класса `ObjectManager` создаёт объект типа `Empty`, затем вызывает метод `addClass(T className)`, где в качестве параметра передаётся только что созданный объект;

— `addFlat()`: этот метод используя запросы из класса `ObjectManager` создаёт объект типа `Flat`, затем вызывает метод `addClass(T className)`, где в качестве параметра передаётся только что созданный объект;
— `addParking()`: этот метод используя запросы из класса `ObjectManager` создаёт объект типа `Parking`, затем вызывает метод `addClass(T className)`, где в качестве параметра передаётся только что созданный объект;
— `addCommercial()`: этот метод используя запросы из класса `ObjectManager` создаёт объект типа `Commercial`, затем вызывает метод `addClass(T className)`, где в качестве параметра передаётся только что созданный объект;
— `getFreeId()`: этот метод возвращает значение поля `freeId`;
— `requestId()`: этот метод возвращает значение поля `freeId` и увеличивает его на один;
— `identifyObject(unsigned int id)`: этот метод используя переданный параметр `id`, открывает необходимый файл и определяет его тип.

Класс `Interfaces`.

Поля класса:

— поле interfaceCode: содержит код интерфейса. По умолчанию код = 0 (Коды: 0 Главное меню, 100 Поиск по ID, 110 Просмотреть объект, 111 Редактировать объект, 112 Удаление объекта, 200 Просмотреть все объекты, 300 Добавить новый объект, 999 Выход);
— поле resentId: содержит id недавно просмотренного объекта.

Методы класса:

— run(): данный метод является бесконечным циклом, внутри которого мы можем переключаться среди интерфейсов, и выйти из цикла можно только выбрав интерфейс под кодом 999;
— printMainMenu(): данный метод выводит на экран интерфейс “Главное меню”;
— selectorMainMenu(): данный метод используя класс Input обеспечивает выбор действия;
— printFindByID(): данный метод выводит на экран интерфейс “Поиск по ID”;
— selectorFindByID(): данный метод используя класс Input обеспечивает выбор действия; — printViewAll(): данный метод выводит на экран интерфейс “Просмотреть всё”;
— selectorViewAll(): данный метод используя класс Input обеспечивает выбор действия;
— printAddNew(): данный метод выводит на экран интерфейс “Добавить новый”;

— selectorAddNew(): данный метод используя класс Input предоставляет выбор типа недвижимости, после переадресовывает на более конкретный метод из класса Storage (Например метод addHouse());
— printListItem(): данный метод находит объект в массиве и выводит на экран информацию о объекте;
— printViewItem(): данный метод выводит на экран интерфейс “Просмотреть объект”;
— selectorViewItem(): данный метод используя класс Input обеспечивает выбор действия; 13
— actionOnObject(unsigned int id, bool justHide): данный метод находит по id необходимый объект и предоставляет выбор редактировать или удалить/скрыть;
— hideObject(T &object): данный шаблонный метод позволяет удалить/скрыть полученный объект;
— editObject(T &object): данный шаблонный метод позволяет отредактировать основные параметры объекта, так же именно этот метод позволяет восстановить объект после удаления;
— editImmovable(T &object): данный шаблонный метод позволяет внести изменения в общие параметры, присущие всем классам;
— editHouse(House &object): данный метод позволяет внести изменения параметров доступных, только для класса House;
— editEmpty(Empty &object): данный метод позволяет внести изменения параметров доступных, только для класса Empty;
— editFlat(Flat &object): данный метод позволяет внести изменения параметров доступных, только для класса Flat;
— editParking(Parking &object): данный метод позволяет внести изменения параметров доступных, только для класса Parking;
— editCommercial(Commercial &object(): данный метод позволяет внести изменения параметров доступных, только для класса Commercial.

Класс `Exception`.

Поля класса:

— поле `msg`: содержит описание исключительной ситуации;
— поле `exceptionTypeCode`: содержит код ошибки

Методы класса:

— `Exception(const string &msg, int type = 0)`: конструктор инициализирует поля класса, используя переданные параметры;
— метод `what()`: Вывод в консоль `msg`.

Класс `Input`.

Методы класса:

— input(T minValue = NULL, T maxValue = NULL): данный шаблонный метод позволяет вводить значения только в приделах от minValue до maxValue;
— cp1251_to_utf8(const char *str): данный метод преобразовывает массив символов с кодировкой cp1251 в строку с кодировкой UTF-8;
— inputInt(int minValue, int maxValue, const string &msg): данный метод выводит на экран запрос на ввод числа и вызывает шаблонный метод input, после возвращает получение значение;
— inputFloat(float minValue, float maxValue, const string &msg): данный метод выводит на экран запрос на ввод числа и вызывает шаблонный метод input, после возвращает получение значение;
— inputDouble(double minValue, double maxValue, const string &msg): данный метод выводит на экран запрос на ввод числа и вызывает шаблонный метод input, после возвращает получение значение;
— inputMobile(): данный метод выводит запрос на ввод номера телефона и проверят результат ввода на действительность;
— inputEmail(): данный метод выводит запрос на ввод email и проверят результат ввода на действительность;
— inputString(const string &question): данный метод выводит на экран сообщение запроса (параметр question) и принимает строку в качестве ответа. Т.к строку приняли в кодировке cp1251 она преобразовывается в UTF-8, чтобы в дальнейшем строка корректно отображалась и записывалась в файл ;
— inputBool(const string &question): данный метод выводит на экран сообщение запроса (параметр question) и занимается преобразованием строкового, понятного для человека ответа в булевой тип данных (например: Да -> 1).

Класс `Immovable`.

Содержит контейнерный класс ContactDetails.

Поля класса:

— поле `id`: содержит уникальный номер.
— поле `isActual`: содержит булевое значение актуальность (по умолчанию оно выставлено как true).
— поле `cost`: содержит значение о стоимости недвижимости.
— поле `square`: содержит значение о общей площади недвижимости.
— поле `address`: содержит строковое значение с адресом недвижимости.

Методы класса:

— метод `printInfo()`: данный метод выводит в консоль ID и акутальность.
— метод `getId()`: данный метод возвращает значение поля `id`.
— метод `printCost()`: данный метод выводит в консоль стоимость недвижимости.
— метод `printSquare()`: данный метод выводит в консоль общую площадь недвижимости.

— гетеры и сетеры для полей.

Класс `ContactDetails`.

Поля класса:

— поле `phoneNumber`: содержит строку с номером телефона.

— поле `email`: содержит пустую строку или строку со значением email.

Методы класса:

— `getContactDetails()`: данный метод возвращает строковое значение, содержащее информацию для связи.

— гетеры и сетеры для полей.

Класс `ObjectManager`, предоставляющий методы для ввода информации о недвижимости.

Поля класса:

— in: объект класса Input, используемый для ввода данных.

Методы класса:

— requestPhone(): запрашивает у пользователя номер телефона и возвращает его в виде строки.

— requestEmail(): запрашивает у пользователя адрес электронной почты и возвращает его в виде строки.

— requestCost(): запрашивает у пользователя стоимость недвижимости и возвращает её в виде числа с плавающей точкой.

— requestSqr(): запрашивает у пользователя общую площадь недвижимости и возвращает её в виде числа с плавающей точкой.

— requestAddr(): запрашивает у пользователя адрес недвижимости и возвращает его в виде строки.

— requestPond(): запрашивает у пользователя информацию о наличии водоемов и возвращает соответствующий ответ.

— requestPlats(): запрашивает у пользователя информацию о наличии деревьев/кустарников и возвращает соответствующий ответ.

— requestCommun(): запрашивает у пользователя информацию о наличии коммуникаций и возвращает соответствующий ответ.

— requestFloor(isFlat): запрашивает у пользователя количество этажей (или этаж) и возвращает введенное значение в виде целого числа.

— requestRooms(): запрашивает у пользователя количество комнат и возвращает введенное значение в виде целого числа.

— requestParking(): запрашивает у пользователя количество парковочных мест и возвращает введенное значение в виде целого числа.

— requestSuiFCons(): запрашивает у пользователя информацию о пригодности для строительства и возвращает соответствующий ответ.

— requestSuiFFarm(): запрашивает у пользователя информацию о пригодности для фермерства и возвращает соответствующий ответ.

— requestHaveBalcony(): запрашивает у пользователя информацию о наличии балкона и возвращает соответствующий ответ.
— requestType(isParking): запрашивает у пользователя тип недвижимости (или парковки) и возвращает введенное значение в виде целого числа.

Класс `Piece(Plot-участок)`, участок является наследником класса `Immovable` и дополнительно содержит следующие поля:
— поле `availablePond`: содержит значение о наличии водоёмов на участке;
— поле `availablePlants`: содержит значение о наличии деревьев/кустарниках на участке;
— поле `availabilityOfCommunications`: содержит значение о проведённых коммуникациях на участке;
Методы класса:
— гетеры и сетеры для полей.

Класс `Empty(пустой участок)`, почва является наследником класса `Piece(Plot)`.
Поля класса:
— поле `suitableForConstruction`: содержит значение о пригодности почвы для строительства;
— поле `suitableForFarming`: содержит значение о пригодности почвы для фермерства.
Методы класса:
— гетеры и сетеры для полей.

Класс `House`, является наследником класса `Piece`.
Поля класса:
— поле `floors`: содержит значение о количестве этажей в здании;
— поле `rooms`: содержит значение о количестве жилых комнат в здании;
— поле `parkingSpaces`: содержит значение о количестве парковочных мест.
Методы класса:
— `printInfo() override`: данный метод является переписанным и вводит на экран информацию о классе.
— гетеры и сетеры для полей.

Класс `Flat`, является наследником класса `Immovable`.
Методы класса:
— поле `floors`: содержит номер этажа на котором расположена квартира;
— поле `rooms`: содержит значение о количестве жилых комнат в квартире;
— поле `haveBalcony`: содержит значение о наличии балкона.
Методы класса:
— `printInfo() override`: данный метод является переписанным и вводит на экран информацию о недвижимости.

— гетеры и сетеры для полей.

Класс `Parking` является наследником класса `Immovable` и дополнительно содержит поле type. Это поле содержит значение о типе парковки.

Методы класса:

— printInfo() override: данный метод является перегруженным и выводит на экран информацию о недвижимости;

— printType(): данный метод вводит на экран информацию о типе недвижимости;

— Гетеры и сетеры для поля type.

6 Разработка программных модулей

Все классы можно разделить на три основных группы:

— Services (Сервисные), к этой группе мы можем отнести все классы, которые помогают программе с обработкой различных процессов. Например для реализации отлова исключительных ситуаций был создан класс Exception. К этой группе так же относится класс Input этот класс предназначен для обработки всех вводимых пользователем параметров. Класс ObjectManager содержит перечень вопросов, которые используются при редактировании/создании нового объекта недвижимости. А также класс Storage, который является ядром программы.

— Immovable что переводится как недвижимость, в этой группе собраны все классы которые описывают различные типы недвижимости и на основе этих классов создаются объекты.

— Interfaces (Интерфейсы) к этой группе относиться класс Interfaces, который ответственен за интерфейс(меню) программы.

6.1 Разработка схем алгоритмов

Для конвертации строки из многобайтовой кодировки cp1251 в UTF-8 используется функция cp1251_to_utf8, принимающая строку. Для преобразования строки из кодировки CP1251 в UTF-16 (широкий формат символов) используется функция WindowsAPI MultiByteToWideChar, а для для преобразования строки из UTF-16 в UTF-8 используется WideCharToMultiByte.

Для ввода целочисленного используется функция inputInt, принадлежащая классу Input. Функция принимает два целочисленных значения, которые описывают минимальную и максимальную границу допустимых значений при вводе. Так же эта функция может принимать строку, которая будет выводиться перед вводом значения, например эта строка может содержать вопрос или пояснения.

Схемы алгоритмов приведены в приложении Б и В.

6.2 Разработка алгоритмов

Функция inputMobile() предназначена для ввода и валидации мобильного телефонного номера.

Шаг 1: Инициализация переменных:

Создается константная строка msg, содержащая приглашение для ввода номера телефона. Инициализируются переменные maxValue и value. maxValue устанавливается в 999999999 (максимальное значение для номера телефона), и value будет использоваться для временного хранения введенного значения.

Шаг 2: Бесконечный цикл:

Запускается бесконечный цикл (while (true)), который будет выполняться до тех пор, пока не будет успешно введено корректное значение.

Шаг 3: Вывод приглашения для ввода:

Выводится приглашение для ввода номера телефона с использованием cout.

Шаг 4: Попытка ввода:

Используется функция input<unsigned long>(100000000, maxValue) для ввода значения. Диапазон ввода установлен от 100000000 до maxValue (999999999). Эта функция может сгенерировать исключение типа Exception, которое будет перехвачено в блоке catch.

Шаг 5: Обработка исключения:

Если произошло исключение типа Exception, блок catch перехватывает его и вызывает метод what() для вывода соответствующего сообщения об ошибке.

Шаг 6: Выход из цикла:

Если ввод прошел успешно (без исключений), цикл прерывается с помощью break.

Шаг 7: Возврат результата:

Формируется строка, содержащая номер телефона в формате "+375" + введенное значение (to_string(value)). Строка возвращается в качестве результата функции.

Функция identifyObject(unsigned int id), выполняет чтение информации из файла, соответствующего переданному id, и возвращает значение атрибута "class", которое определяет тип объекта в файле.

Шаг 1: Формирование пути к файлу :
Создается строка path, содержащая путь к файлу, основанная на переданном id.
Шаг 2: Открытие файла для чтения :
Функция открывает файл по указанному пути для чтения.
Шаг 3: Чтение строк из файла :
Запускается цикл while, который читает строки из файла.
Каждая строка разбивается на имя и значение. Имя ищется до первого пробела, а значение — после пробела и до конца строки.
Если имя равно "class", цикл завершается, и значение этой строки сохраняется в переменной value.
Шаг 4: Закрытие файла :
Файл закрывается.
Шаг 5: Возврат значения :
Функция возвращает значение атрибута "class" (тип объекта), прочитанное из файла.

7 Результаты работы программы

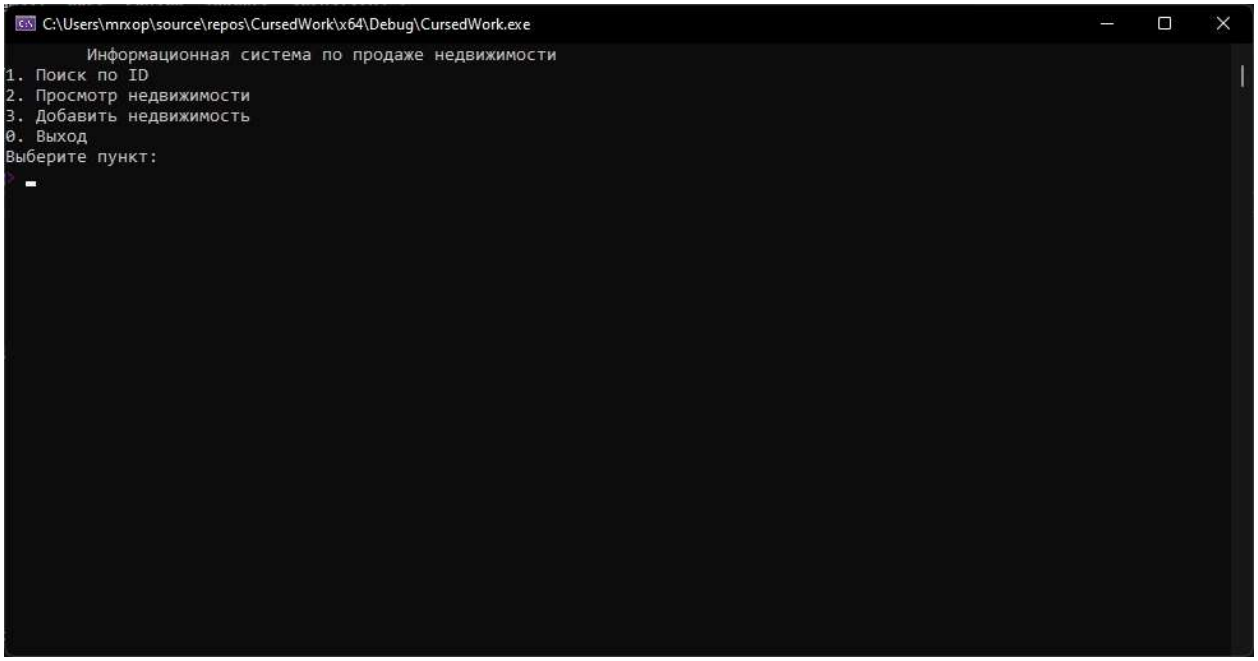

Рисунок 1 – Просмотр, поиск и добавление недвижимости

Рисунок 2 – Просмотр информации о недвижимости

Рисунок 3 – Редактирование и удаление недвижимости

ЗАКЛЮЧЕНИЕ

В ходе курсового проектирования были решены следующие задачи:

- собрана необходимая информация для реализации проекта;
- продумана структура программы;
- реализация проекта;
- тестирование и отладка программы;
- разработка документации.

Была разработана программа «Информационная система по продаже недвижимости».

Разработанная программа запускается на операционных системах семейства Windows начиная с Windows 7 x64 при наличии процессора с т.ч. 1GHz, оперативной памяти объёмом 500Mb и 5Mb свободного места на жёстком диске.

ЛИТЕРАТУРА

[1] Страуструп, Б. Язык программирования C++ / Б. Страуструп; специальное издание. Пер с англ. - СПб.: BHV, 2008

[2] Шилдт, Г. C++: базовый курс / Г. Шилдт - М.: Вильямс; специальное издание. Пер. с англ. - М.: Вильямс, 2007

[3] Луцик Ю. А. Объектно-ориентированное программирование на языке C++ / Ю. А. Луцик, А. М. Ковальчук, И. В. Лукьянова - Мн.: БГУИР, 2003

[4] Прата, С. - Язык программирования C++. Лекции и упражнения / С. Прата; специальное издание. Пер. с англ. - М.: Вильямс, 2018

[5] Конструирование программ и языки программирования: метод. указания по курсовому проектированию для студ. спец. I-40 02 01 “Вычислительные машины, системы и сети” для всех форм обуч. / сост. А. В. Бушкевич, А. М. Ковальчук, И. В. Лукьянова. – Минск : БГУИР, 2009.

ПРИЛОЖЕНИЕ А
(обязательное)

Диаграмма классов

ПРИЛОЖЕНИЕ Б

(обязательное)

Схема алгоритма CP1251 в UTF-8

ПРИЛОЖЕНИЕ В
(обязательное)

Схема алгоритма inputInt

ПРИЛОЖЕНИЕ Г
(обязательное)

Код программы

ПРИЛОЖЕНИЕ Д
(обязательное)

Ведомость документов