|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №1  по дисциплине «Языки программирования»  Вариант 6  Программа «Риэлтерское агентство» | | |
|  | | Работу выполнил  студент группы ПМИ-9-2023 1 курса  Деревянных А.А.  25 января 2024 г. |
| Работу проверил  Рубцова М.Б.  25 января 2024 г. |
| Пермь 2024 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc153130027)

[Алгоритм решения 5](#_Toc153130028)

[Тестирование 9](#_Toc153130029)

[Код программы 13](#_Toc153130030)

# Постановка задачи

**1 Требуется создать файл записей, в котором будет храниться информация о квартирах, продаваемых риэлтерским агентством. Данные, которые необходимо хранить для каждой квартиры, включают:**

- Адрес (улица, дом, квартира)

- Количество комнат

- Общая площадь

- Жилая площадь

- Этаж

- Этажность дома

- Фамилия владельца

- Стоимость

**2 Необходимо разработать и реализовать программу "Риэлтерское агентство", которая будет считывать исходную информацию из файла и предоставлять возможность создавать следующие отчеты:**

2.1 Полный список всех квартир, отсортированный следующим образом:

- Сначала сортировать по количеству комнат в порядке убывания.

- Если количество комнат одинаковое, то сортировать по стоимости в порядке возрастания.

2.2 Список всех квартир с заданным количеством комнат (количество комнат вводится с клавиатуры), отсортированный следующим образом:

- Сначала сортировать по этажу в порядке возрастания.

- Если этаж одинаковый, то сортировать по этажности дома в порядке возрастания.

- Если этажность дома и этаж одинаковые, то сортировать по стоимости в порядке убывания.

2.3 Список всех квартир, стоимость которых находится в диапазоне от N1 до N2 рублей (N1 и N2 вводятся с клавиатуры), отсортированный следующим образом:

- Сначала сортировать по стоимости в порядке убывания.

- Если стоимость одинаковая, то сортировать по общей площади в порядке возрастания.

**3 Требуется создать базу квартир, которая включает не менее 25 записей. На основе этой базы должны быть сформированы все указанные выше списки. Для реализации базы квартир можно использовать вложенные списки или словари.**

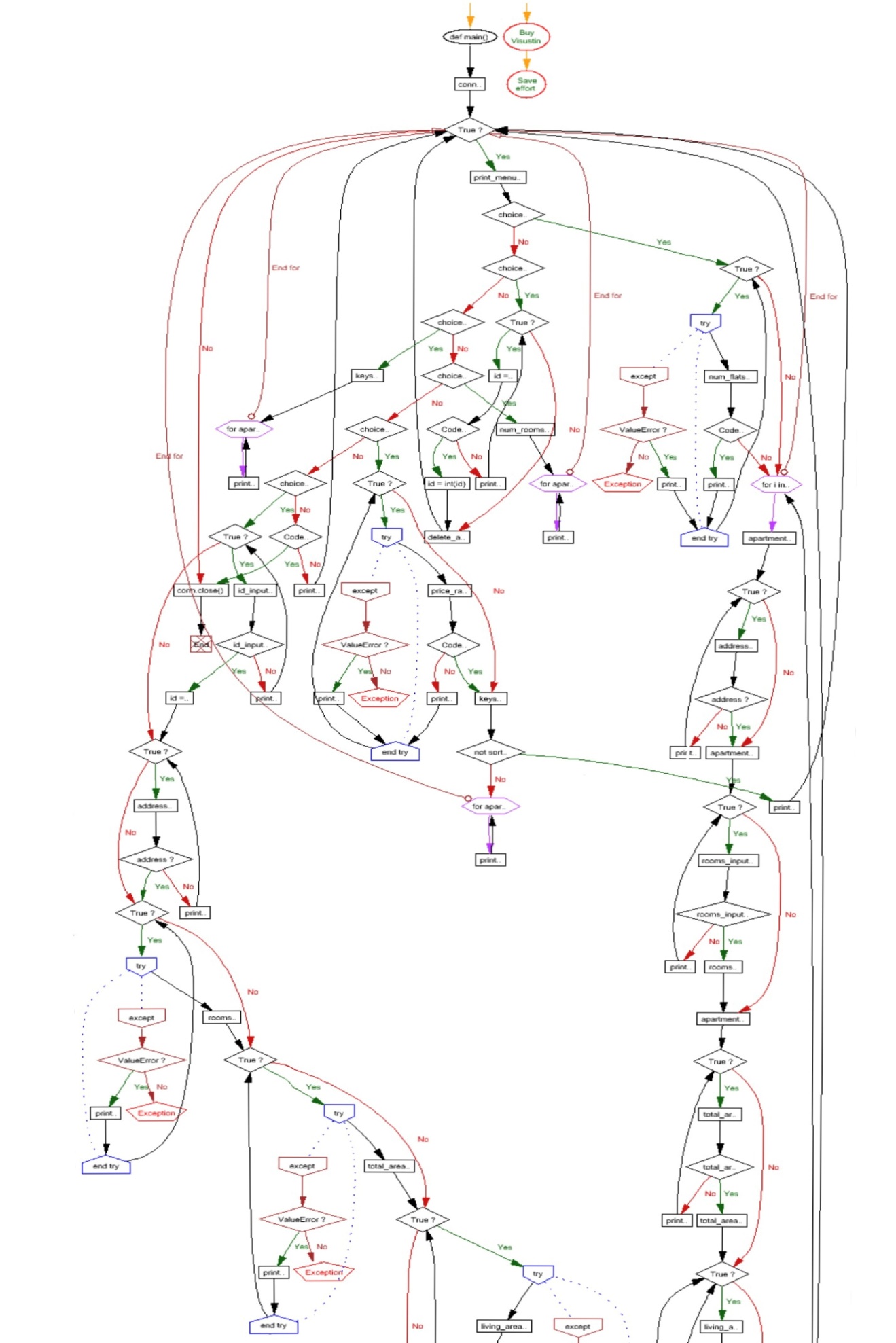
**4 В программе должны быть реализованы отдельные функции для взаимодействия с базой данных, включая добавление, удаление и изменение записей, а также вывод всей базы и отдельных списков на экран.**

**5 Интерфейс пользователя предпочтительно организовать в виде меню, где пользователь может выбрать соответствующее действие.**

**6 Программа должна иметь дружественный интерфейс для удобного взаимодействия с пользователем.**

# Алгоритм решения

Вот блок-схема, соответствующая данному коду:



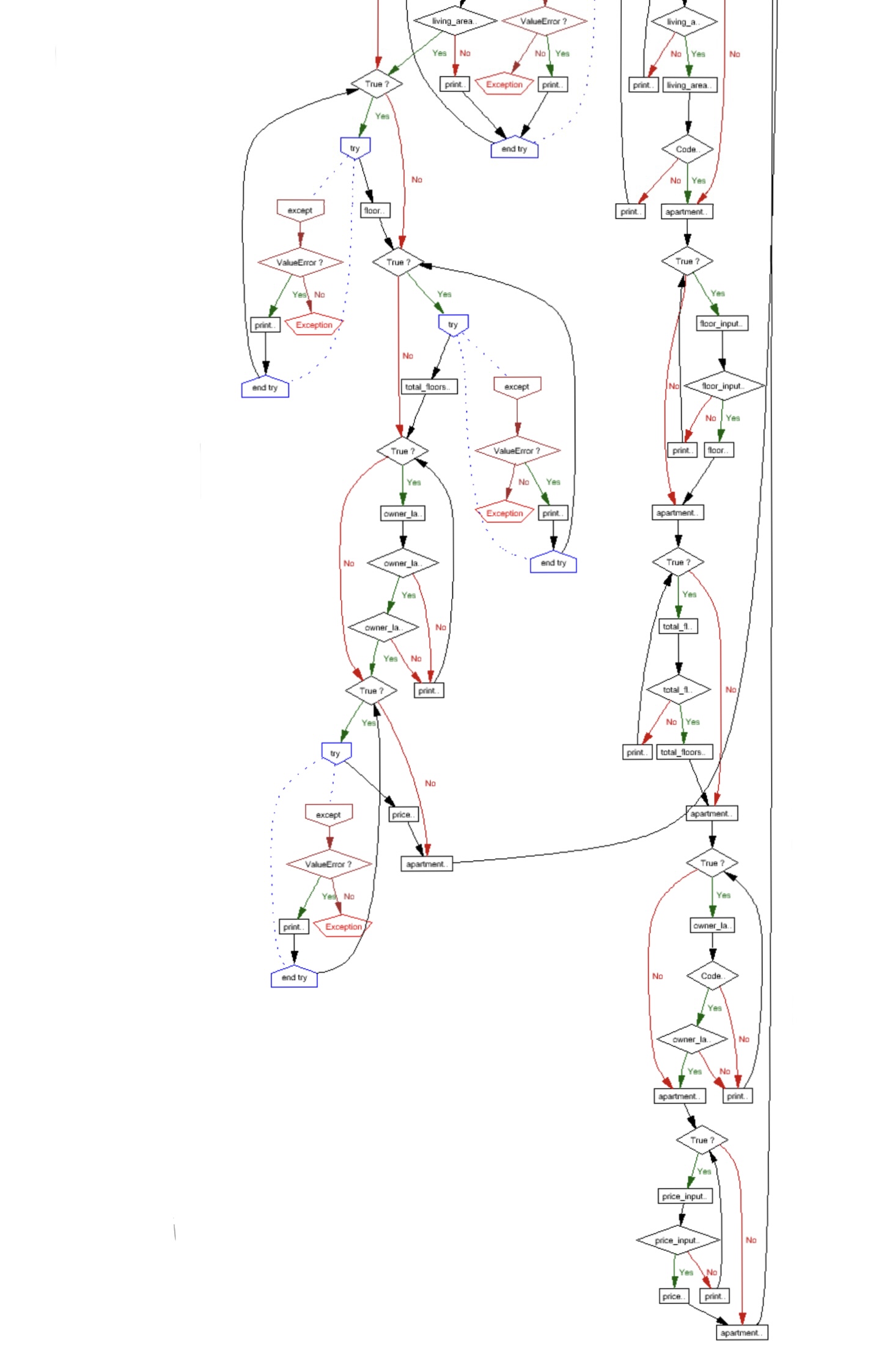


Диаграмма программы представляет собой визуализацию функций в коде, где блоки обозначают различные функции, а стрелки указывают на поток управления и данных.

1. **Основные функции программы:**
2. **create\_connection():**
   * Создает соединение с базой данных SQLite с именем "apartments.db".
   * Если база данных не существует, она будет создана.
3. **create\_table(conn):**
   * Создает таблицу "apartments" в базе данных с колонками для адреса, количества комнат, общей и жилой площади, этажа, общего количества этажей, фамилии владельца и цены.
4. **insert\_apartment(conn, apartment):**
   * Вставляет новую запись о квартире в таблицу "apartments".
5. **shell\_sort(apartments, keys):**
   * Сортирует список словарей (записей квартир) по указанным ключам с использованием алгоритма сортировки Шелла.
6. **get\_sorted\_apartments(conn, keys):**
   * Извлекает все записи из таблицы "apartments" и сортирует их по указанным ключам с помощью алгоритма сортировки Шелла.
7. **get\_sorted\_apartments\_by\_rooms(conn, num\_rooms, keys):**
   * Извлекает записи с указанным количеством комнат и сортирует их по указанным ключам.
8. **get\_sorted\_apartments\_by\_price\_range(conn, price\_range, keys):**
   * Извлекает записи в указанном ценовом диапазоне и сортирует их по указанным ключам.
9. **delete\_apartment(conn, id):**
   * Удаляет запись из таблицы "apartments" по идентификатору.
10. **update\_apartment(conn, id, apartment):**
    * Обновляет запись в таблице "apartments" по идентификатору новой информацией о квартире.
11. **print\_menu():**
    * Отображает меню опций для управления базой данных квартир.
12. **main():**
    * Основная функция, которая запускает систему управления базой данных квартир.
    * Отображает меню опций, обрабатывает ввод пользователя и вызывает соответствующие функции.
13. **Алгоритм Shell Sort:**

Алгоритм Shell Sort используется для сортировки списка записей квартир по указанным ключам. Он разделяет список на подсписки, сортирует их и объединяет обратно в отсортированном порядке. Размер подсписков определяется последовательностью пробелов.

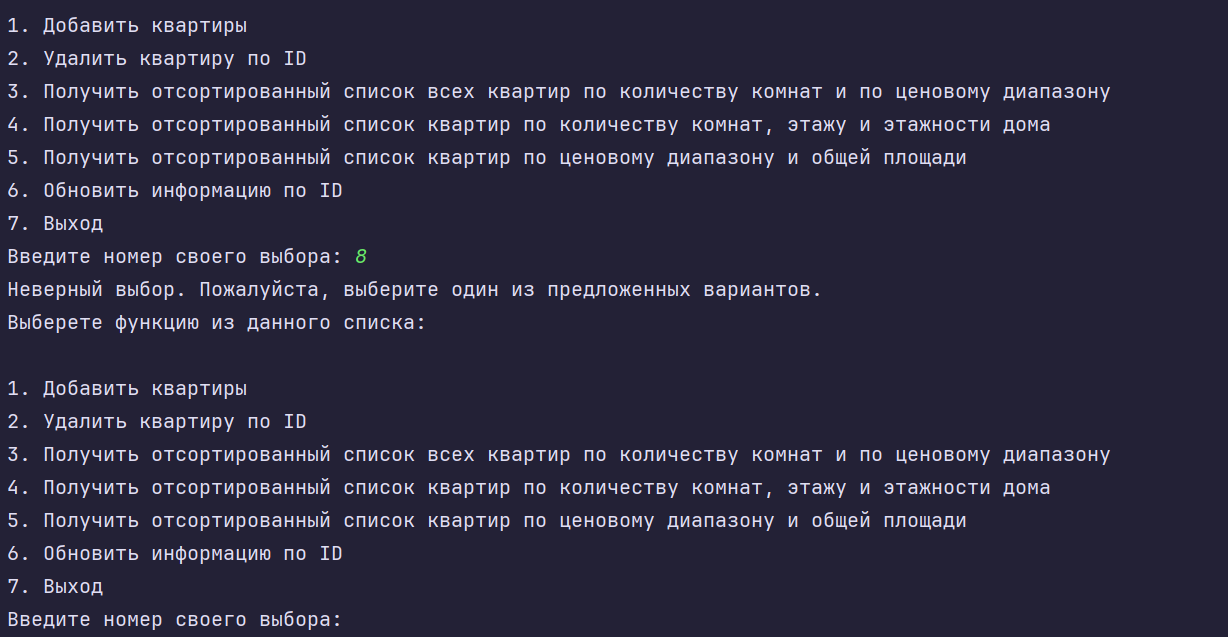
1. **Использование программы:**

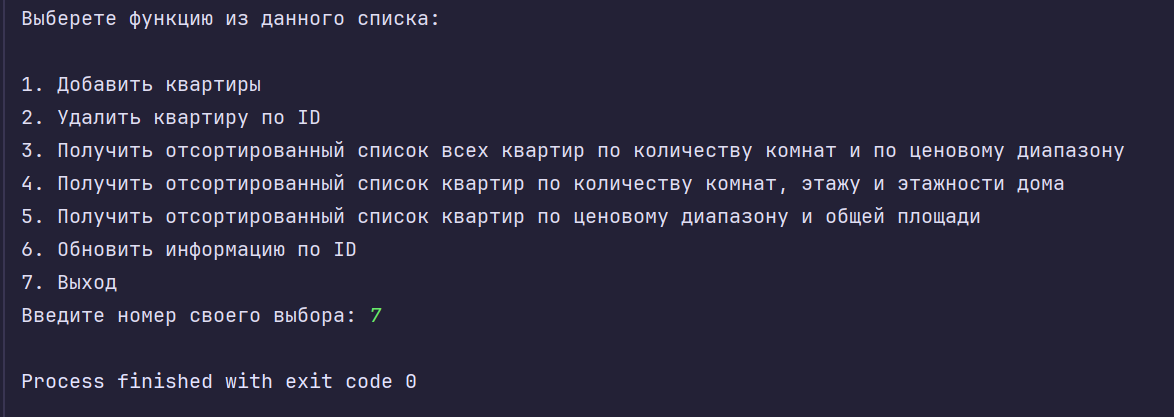
Пользователь может выбирать опции из меню, включая добавление, удаление и обновление записей о квартирах. Результаты отображаются в виде отсортированного списка записей квартир в соответствии с выбранными ключами.

Этот Python-скрипт обеспечивает эффективное управление базой данных квартир с использованием SQLite.

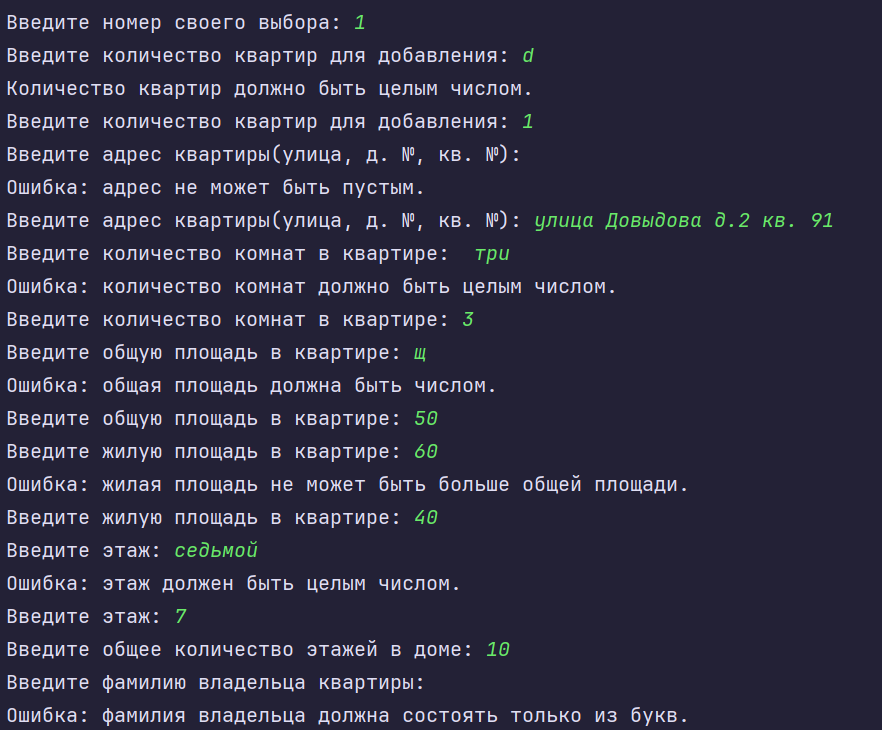
# Тестирование

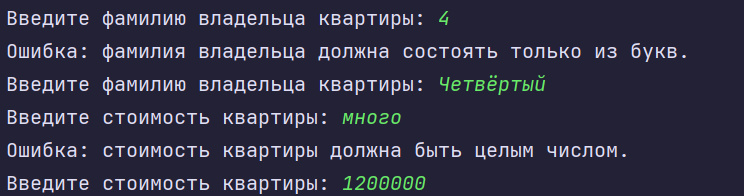
Тестирование работы меню:



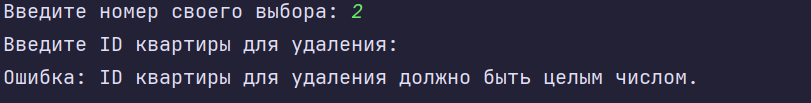


Проверка добавления квартиры:



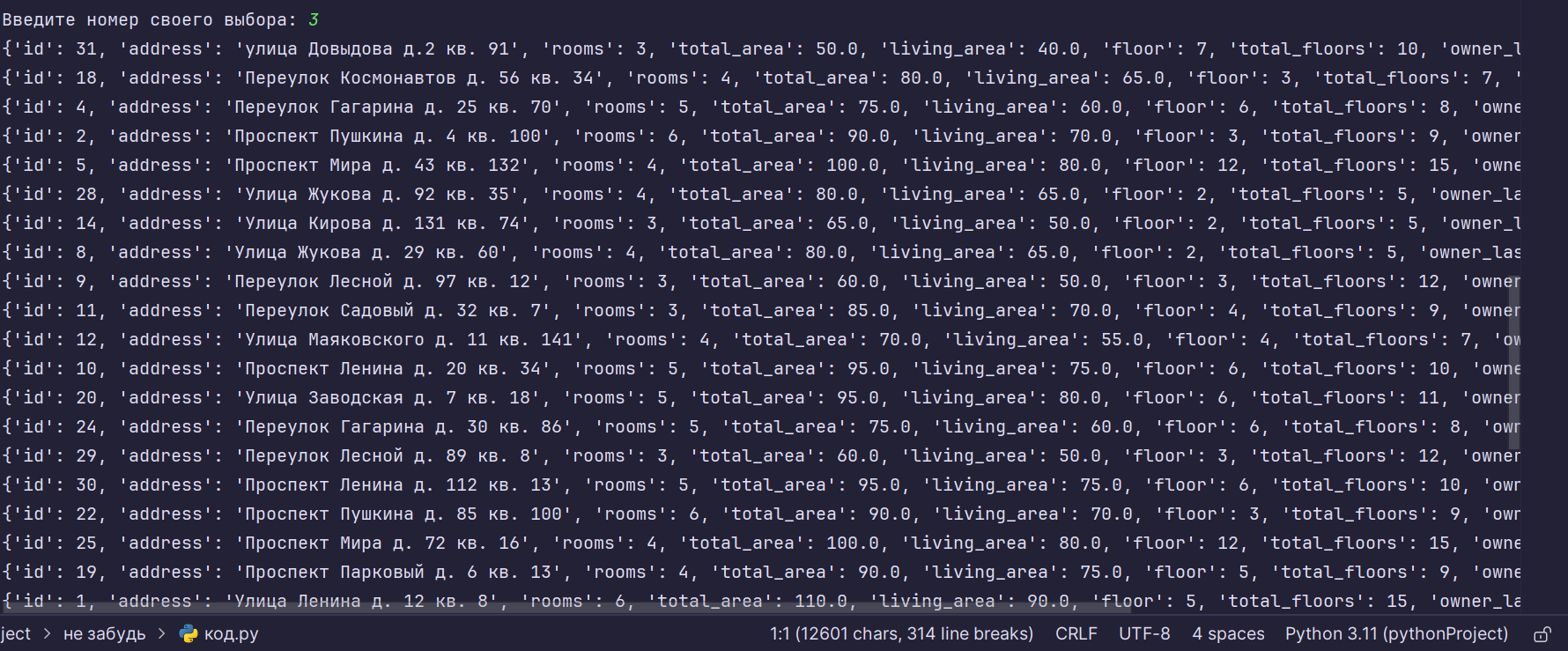


Тестирование удаления квартиры по ID:

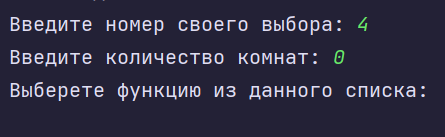


Если ввести существующий ID, то данные этого ID удаляться из базы данных.

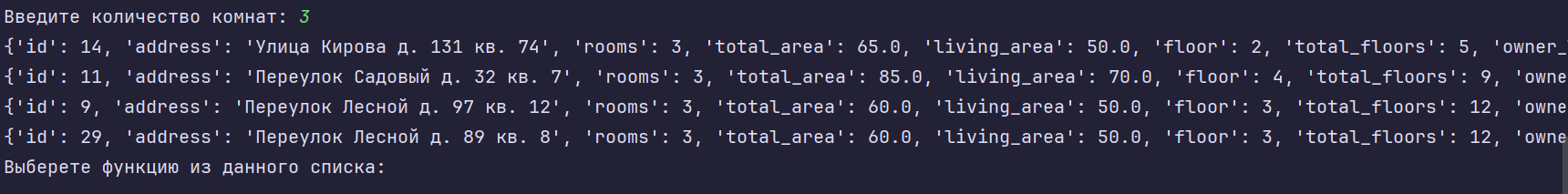
Проверка сортировки списка всех квартир по количеству комнат и по ценовому диапазону:



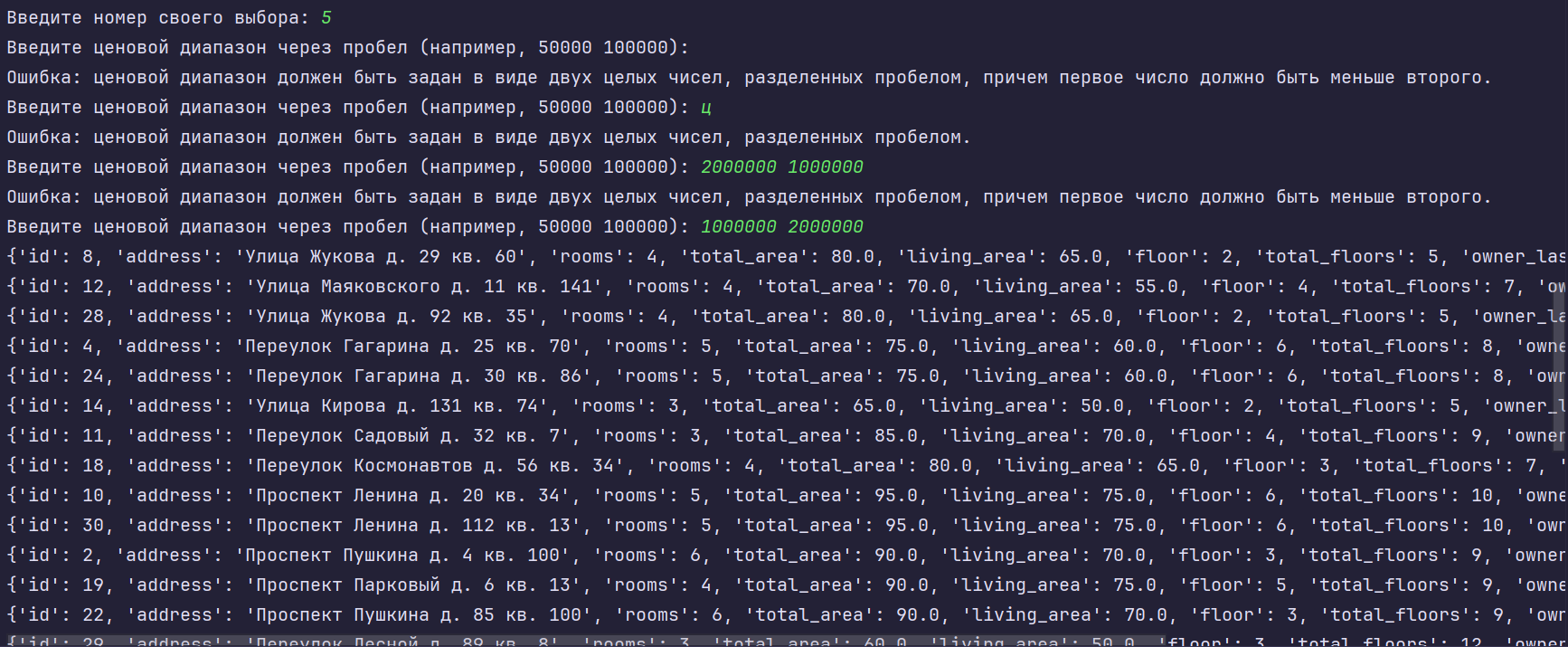
Проверка сортировки списка квартир по количеству комнат, этажу и этажности дома:



Когда нет искомого количества комнат, список не формируется, но возвращается в главное меню.

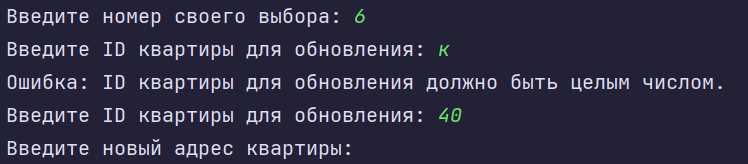


Тестирование сортировки списка квартир по ценовому диапазону и общей площади:





Тестирование по обновлению информации по ID:



Дальше код работает как в функции добавление квартиры.

# Код программы

import sqlite3

#создаёт файл под бд

def create\_connection():

conn = sqlite3.connect('apartments.db')

return conn

def create\_table(conn):

conn.execute('''CREATE TABLE IF NOT EXISTS apartments (

id INTEGER PRIMARY KEY AUTOINCREMENT,

address TEXT NOT NULL,

rooms INTEGER NOT NULL,

total\_area REAL NOT NULL,

living\_area REAL NOT NULL,

floor INTEGER NOT NULL,

total\_floors INTEGER NOT NULL,

owner\_lastname TEXT NOT NULL,

price INTEGER NOT NULL)''')

def insert\_apartment(conn, apartment):

cursor = conn.cursor()

cursor.execute("""

INSERT INTO apartments (address, rooms, total\_area, living\_area, floor, total\_floors, owner\_lastname, price)

VALUES (?, ?, ?, ?, ?, ?, ?, ?)

""", (apartment['address'], apartment['rooms'], apartment['total\_area'],

apartment['living\_area'], apartment['floor'], apartment['total\_floors'],

apartment['owner\_lastname'], apartment['price']))

conn.commit()

#сортировка Шелла

def shell\_sort(apartments, keys):

gap = len(apartments) // 2

while gap > 0:

for i in range(gap, len(apartments)):

temp = apartments[i]

j = i

while j >= gap and all(temp[keys[k]] < apartments[j - gap][keys[k]] for k in range(len(keys))):

apartments[j] = apartments[j - gap]

j -= gap

apartments[j] = temp

gap //= 2

return apartments

# ключи для сортировки

#вывод всей бд

def get\_sorted\_apartments(conn, keys):

cursor = conn.execute("SELECT \* FROM apartments")

apartments = [dict(zip([column[0] for column in cursor.description], row)) for row in cursor.fetchall()]

return shell\_sort(apartments, keys)

#сортировка по комнатам

def get\_sorted\_apartments\_by\_rooms(conn, num\_rooms, keys):

cursor = conn.execute("SELECT \* FROM apartments WHERE rooms = ?", (num\_rooms,))

apartments = [dict(zip([column[0] for column in cursor.description], row)) for row in cursor.fetchall()]

return shell\_sort(apartments, keys)

#сортировка по цене

def get\_sorted\_apartments\_by\_price\_range(conn, price\_range, keys):

cursor = conn.execute("SELECT \* FROM apartments WHERE price >= ? AND price <= ?", price\_range)

apartments = [dict(zip([column[0] for column in cursor.description], row)) for row in cursor.fetchall()]

return shell\_sort(apartments, keys)

#айди

def delete\_apartment(conn, id):

query = "DELETE FROM apartments WHERE id = ?"

conn.execute(query, (id,))

conn.commit()

#айди по квартирам

def update\_apartment(conn, id, apartment):

query = "UPDATE apartments SET address = ?, rooms = ?, total\_area = ?, living\_area = ?, floor = ?, total\_floors = ?, owner\_lastname = ?, price = ? WHERE id = ?"

conn.execute(query, apartment + (id,))

conn.commit()

#Меню управления

def print\_menu():

print("Выберете функцию из данного списка:")

print("\n1. Добавить квартиры")

print("2. Удалить квартиру по ID")

print("3. Получить отсортированный список всех квартир по количеству комнат и по ценовому диапазону")

print("4. Получить отсортированный список квартир по количеству комнат, этажу и этажности дома")

print("5. Получить отсортированный список квартир по ценовому диапазону и общей площади")

print("6. Обновить информацию по ID")

print("7. Выход")

# работа с меню

#вход в меню

def main():

conn = create\_connection()

create\_table(conn)

#выбор варианта меню

while True:

print\_menu()

choice = int(input("Введите номер своего выбора: "))

#добавление квартир

if choice == 1:

while True:

try:

num\_flats = int(input("Введите количество квартир для добавления: "))

if num\_flats < 0:

print("Количество квартир должно быть положительным числом.")

else:

break

except ValueError:

print("Количество квартир должно быть целым числом.")

for i in range(num\_flats):

apartment = {}

while True:

address = input("Введите адрес квартиры(улица, д. №, кв. №): ")

if address:

break

print("Ошибка: адрес не может быть пустым.")

apartment['address'] = address

while True:

rooms\_input = input("Введите количество комнат в квартире: ")

if rooms\_input.isdigit():

rooms = int(rooms\_input)

break

print("Ошибка: количество комнат должно быть целым числом.")

apartment['rooms'] = rooms

while True:

total\_area\_input = input("Введите общую площадь в квартире: ")

if total\_area\_input.replace('.', '', 1).isdigit():

total\_area = float(total\_area\_input)

break

print("Ошибка: общая площадь должна быть числом.")

while True:

living\_area\_input = input("Введите жилую площадь в квартире: ")

if living\_area\_input.replace('.', '', 1).isdigit():

living\_area = float(living\_area\_input)

if living\_area <= total\_area:

break

print("Ошибка: жилая площадь не может быть больше общей площади.")

else:

print("Ошибка: жилая площадь должна быть числом.")

apartment['total\_area'] = total\_area

apartment['living\_area'] = living\_area

while True:

floor\_input = input("Введите этаж: ")

if floor\_input.isdigit():

floor = int(floor\_input)

break

print("Ошибка: этаж должен быть целым числом.")

apartment['floor'] = floor

while True:

total\_floors\_input = input("Введите общее количество этажей в доме: ")

if total\_floors\_input.isdigit():

total\_floors = int(total\_floors\_input)

break

print("Ошибка: общее количество этажей должно быть целым числом.")

apartment['total\_floors'] = total\_floors

while True:

owner\_lastname = input("Введите фамилию владельца квартиры: ")

if owner\_lastname:

if owner\_lastname.isalpha():

break

print("Ошибка: фамилия владельца должна состоять только из букв.")

apartment['owner\_lastname'] = owner\_lastname

while True:

price\_input = input("Введите стоимость квартиры: ")

if price\_input.isdigit():

price = int(price\_input)

break

print("Ошибка: стоимость квартиры должна быть целым числом.")

apartment['price'] = price

insert\_apartment(conn, apartment)

#удоление квартиры по айди

elif choice == 2:

while True:

id = input("Введите ID квартиры для удаления: ")

if id.isdigit():

id = int(id)

break

print("Ошибка: ID квартиры для удаления должно быть целым числом.")

delete\_apartment(conn, id)

# отсортированный список всех квартир

elif choice == 3:

keys = ['rooms', 'price']

sorted\_apartments = get\_sorted\_apartments(conn, keys)

for apartment in sorted\_apartments:

print(apartment)

#отсортированный список квартир по количеству комнат

elif choice == 4:

num\_rooms = int(input("Введите количество комнат: "))

keys = ['floor', 'total\_floors', 'price']

sorted\_apartments = get\_sorted\_apartments\_by\_rooms(conn, num\_rooms, keys)

for apartment in sorted\_apartments:

print(apartment)

#отсортированный список квартир по ценовому диапазону

elif choice == 5:

while True:

try:

price\_range\_input = input("Введите ценовой диапазон через пробел (например, 50000 100000): ")

price\_range = tuple(map(int, price\_range\_input.split()))

if len(price\_range) == 2 and price\_range[0] < price\_range[1]:

break

print(

"Ошибка: ценовой диапазон должен быть задан в виде двух целых чисел, разделенных пробелом, причем первое число должно быть меньше второго.")

except ValueError:

print("Ошибка: ценовой диапазон должен быть задан в виде двух целых чисел, разделенных пробелом.")

keys = ['price']

sorted\_apartments = get\_sorted\_apartments\_by\_price\_range(conn, price\_range, keys)

if not sorted\_apartments:

print("Квартир в данном ценовом диапазоне нет.")

else:

for apartment in sorted\_apartments:

print(apartment)

# Обновить информацию по ID

elif choice == 6:

while True:

id\_input = input("Введите ID квартиры для обновления: ")

if id\_input.isdigit():

id = int(id\_input)

break

print("Ошибка: ID квартиры для обновления должно быть целым числом.")

while True:

address = input("Введите новый адрес квартиры: ")

if address:

break

print("Ошибка: адрес квартиры не может быть пустым.")

while True:

try:

rooms = int(input("Введите новое количество комнат в квартире: "))

break

except ValueError:

print("Ошибка: количество комнат должно быть целым числом.")

while True:

try:

total\_area = float(input("Введите новую общую площадь в квартире: "))

break

except ValueError:

print("Ошибка: общая площадь должна быть числом.")

while True:

try:

living\_area = float(input("Введите новую жилую площадь в квартире: "))

if living\_area <= total\_area:

break

print("Ошибка: жилая площадь не может быть больше общей площади.")

except ValueError:

print("Ошибка: жилая площадь должна быть числом.")

while True:

try:

floor = int(input("Введите новый этаж: "))

break

except ValueError:

print("Ошибка: этаж должен быть целым числом.")

while True:

try:

total\_floors = int(input("Введите новое общее количество этажей в доме: "))

break

except ValueError:

print("Ошибка: общее количество этажей должно быть целым числом.")

while True:

owner\_lastname = input("Введите новую фамилию владельца квартиры: ")

if owner\_lastname:

if owner\_lastname.isalpha():

break

print("Ошибка: фамилия владельца должна состоять только из букв.")

while True:

try:

price = int(input("Введите новую стоимость квартиры: "))

break

except ValueError:

print("Ошибка: стоимость квартиры должна быть целым числом.")

apartment = (id, address, rooms, total\_area, living\_area, floor, total\_floors, owner\_lastname, price)

update\_apartment(conn, id, apartment)

# выход

elif choice == 7:

break

else:

print("Неверный выбор. Пожалуйста, выберите один из предложенных вариантов.")

# звкрытие бд

conn.close()

if \_\_name\_\_ == "\_\_main\_\_":

main()