**Московский государственный технический  
университет им. Н.Э. Баумана**

Факультет информатика и системы управления  
Кафедра системы обработки информации и управления

Курс «Парадигмы и конструкции языков программирования»  
Отчет по рубежному контролю №1  
Вариант Б6

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-32Б: |  | преподаватель каф. ИУ5 |
| Кунев В. |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

## Постановка задачи

В рамках рубежного контроля №1 по курсу «Парадигмы и конструкции языков программирования» требовалось разработать программу на языке Python.

**Цель работы:**

1. Создать два класса данных, соответствующие предметной области **«Дом» (Класс 1) и «Улица» (Класс 2)** (Вариант №6).
2. Реализовать связь «один-ко-многим» между классами.
3. Создать дополнительный класс для реализации связи «многие-ко-многим».
4. Создать списки объектов классов, содержащие тестовые данные (3-5 записей).
5. Реализовать три запроса в соответствии с **Вариантом Б**, адаптировав их под выбранную предметную область.
6. При реализации запросов использовать функциональные возможности Python (list comprehensions).

**Задачи (Вариант Б):**

1. «Улица» и «Дом» связаны соотношением «один-ко-многим». Вывести список всех связанных домов и улиц, отсортированный по домам (номеру дома).
2. «Улица» и «Дом» связаны соотношением «один-ко-многим». Вывести список улиц с количеством домов в каждой, отсортированный по количеству домов.
3. «Улица» и «Дом» связаны соотношением «многие-ко-многим». Вывести список всех домов, у которых номер заканчивается на «а», и названия их улиц.

## Текст программы

"""Решение Рубежного контроля №1 по курсу ПиК ЯП.

Вариант предметной области: 6 (Дом, Улица)

Вариант запросов: Б

Скрипт создает модели данных для 'Дома' и 'Улицы', реализует связи "один-ко-многим" и "многие-ко-многим",

а затем выполняет три запроса из варианта 'Б'.

"""

from operator import itemgetter

class House:

    """Представление дома.

    Attributes:

        identifier (int): Уникальный идентификатор дома.

        number (str): Номер дома (например, '10а', '22').

        population (int): Количество жильцов в доме.

        street\_id (int): Внешний ключ, ссылающийся на ID улицы (для связи 1:M).

    """

    def \_\_init\_\_(self, identifier: int, number: str, population: int, street\_id: int):

        """Инициализирует экземпляр дома.

        Args:

            identifier (int): Уникальный идентификатор дома.

            number (str): Номер дома.

            population (int): Количество жильцов.

            street\_id (int): ID связанной улицы.

        """

        self.identifier = identifier

        self.number = number

        self.population = population

        self.street\_id = street\_id

class Street:

    """Представление улицы.

    Attributes:

        identifier (int): Уникальный идентификатор улицы.

        name (str): Название улицы.

    """

    def \_\_init\_\_(self, identifier: int, name: str):

        """Инициализирует экземпляр улицы.

        Args:

            identifier (int): Уникальный идентификатор улицы.

            name (str): Название улицы.

        """

        self.identifier = identifier

        self.name = name

class HouseStreet:

    """

    Ассоциативная сущность для связи "многие-ко-многим" между Домом и Улицей.

    Attributes:

        street\_id (int): Внешний ключ, ссылающийся на ID улицы.

        house\_id (int): Внешний ключ, ссылающийся на ID дома.

    """

    def \_\_init\_\_(self, street\_id: int, house\_id: int):

        """Инициализирует связь M:M.

        Args:

            street\_id (int): ID связанной улицы.

            house\_id (int): ID связанного дома.

        """

        self.street\_id = street\_id

        self.house\_id = house\_id

# Тестовые данные

streets: list[Street] = [

    Street(1, "ул. Ленина"),

    Street(2, "ул. Пушкина"),

    Street(3, "просп. Мира"),

]

houses: list[House] = [

    House(1, "10а", 50, 1),

    House(2, "12", 120, 1),

    House(3, "5а", 80, 2),

    House(4, "7б", 200, 3),

    House(5, "22", 75, 2),

]

houses\_streets: list[HouseStreet] = [

    HouseStreet(1, 1),

    HouseStreet(2, 1),

    HouseStreet(1, 2),

    HouseStreet(2, 3),

    HouseStreet(3, 4),

    HouseStreet(2, 5),

]

def main():

    """

    Основная функция программы.

    Выполняет подготовку данных (join) и решает три задачи варианта 'Б'.

    """

    # Подготовка данных

    # Соединение "один-ко-многим" (Дом -> Улица)

    one\_to\_many: list[tuple[str, int, str]] = [

        (h.number, h.population, s.name)

        for s in streets

        for h in houses

        if h.street\_id == s.identifier

    ]

    # Соединение "многие-ко-многим"

    many\_to\_many\_temp: list[tuple[str, int, int]] = [

        (s.name, hs.street\_id, hs.house\_id)

        for s in streets

        for hs in houses\_streets

        if s.identifier == hs.street\_id

    ]

    many\_to\_many: list[tuple[str, int, str]] = [

        (h.number, h.population, s\_name)

        for s\_name, s\_id, h\_id in many\_to\_many\_temp

        for h in houses

        if h.identifier == h\_id

    ]

    # Выполнение запросов

    print("Задание Б1")

    # Вывести список всех связанных домов и улиц, отсортированный по домам (номеру дома).

    res\_1: list[tuple[str, int, str]] = sorted(one\_to\_many, key=itemgetter(0))

    print(res\_1)

    print()

    print("Задание Б2")

    # Вывести список улиц с количеством домов в каждой, отсортированный по количеству домов.

    res\_2\_unsorted: list[tuple[str, int]] = []

    for s in streets:

        s\_houses: list[tuple[str, int, str]] = [

            item for item in one\_to\_many if item[2] == s.name

        ]

        if len(s\_houses) > 0:

            s\_count: int = len(s\_houses)

            res\_2\_unsorted.append((s.name, s\_count))

    res\_2: list[tuple[str, int]] = sorted(res\_2\_unsorted, key=itemgetter(1))

    print(res\_2)

    print()

    print("Задание Б3")

    # Вывести список всех домов, у которых номер заканчивается на «а», и названия их улиц.

    res\_3: list[tuple[str, str]] = [

        (h\_num, s\_name) for h\_num, \_, s\_name in many\_to\_many if h\_num.endswith("а")

    ]

    print(res\_3)

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

    main()

## Скриншот работы приложения

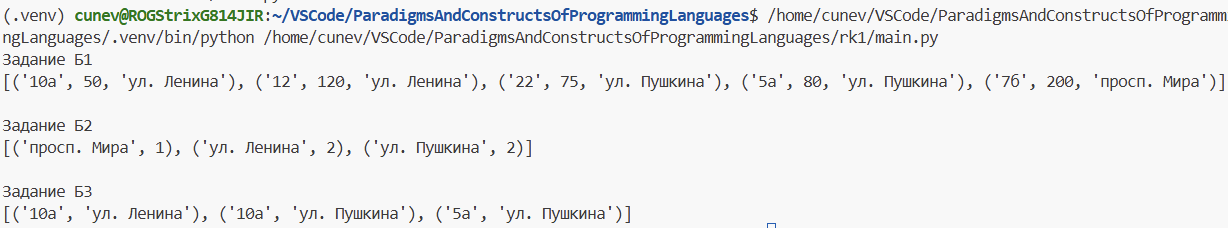


Рисунок 1. Вывод результатов программы