Студент группы ИС-26 Землянский М.Н.

Практическое задание №6

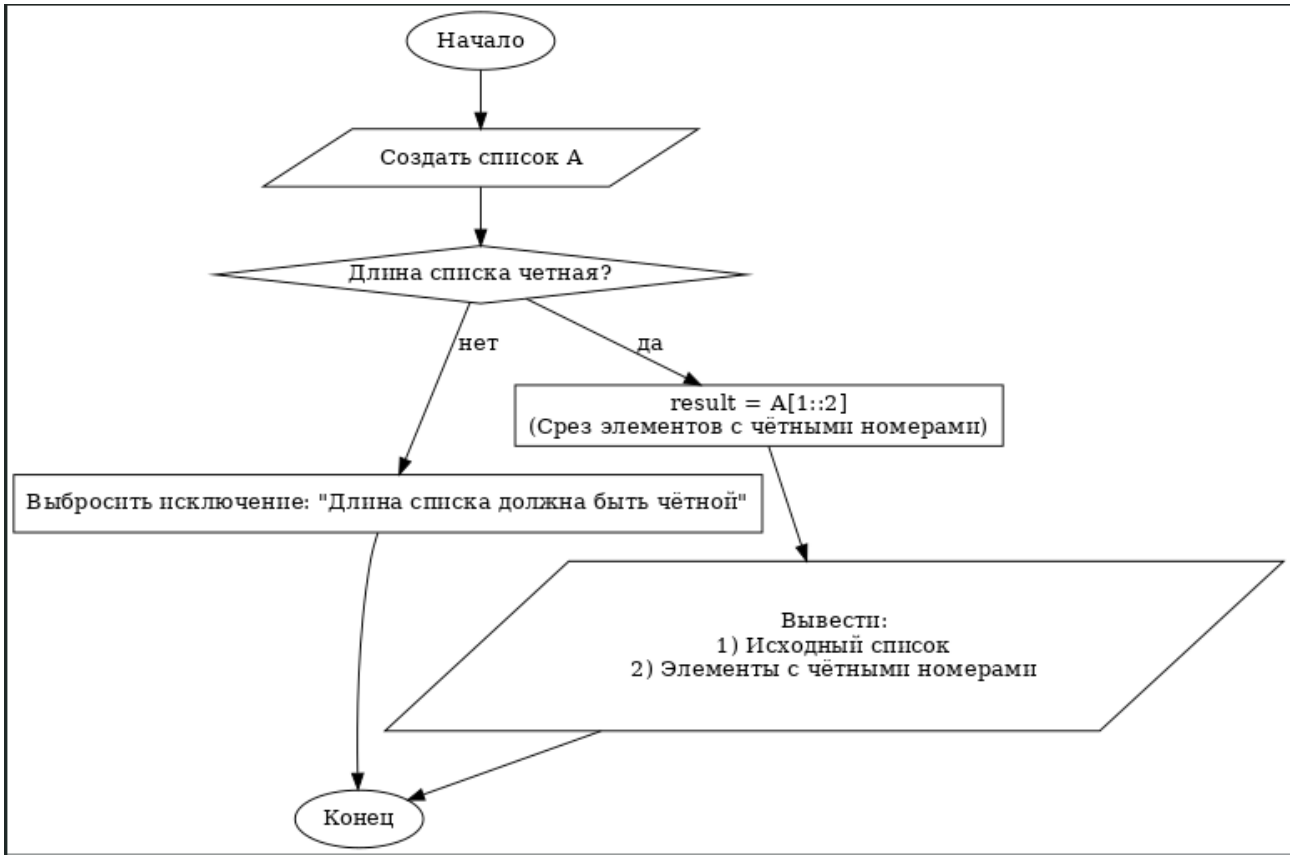
Тема: составление программ со списками в IDE PyCharm Community.

Цель: закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ со списками в IDE PyCharm Community.

Постановка задачи №1:

Дан список A длины N, где N — чётное число. Необходимо вывести элементы списка, имеющие чётные номера, в порядке их следования. Условный оператор использовать нельзя.

Блок-схема



Код программы

```
#Дан список A размера N (N — четное число). Вывести его элементы с четными
#номерами в порядке возрастания номеров: A2, A4, A6, ..., AN. Условный оператор не
#использовать.

def main():
    try:
        # Исходный список
        A = [10, 20, 30, 40, 50, 60, 70, 80]
        N = len(A)

        # Проверка, что длина списка чётная
        if N % 2 != 0:
```

```
            raise ValueError("Длина списка должна быть чётной.")


        result = A[1::2]  # Берем элементы с индексами 1, 3, 5 и так далее

        # Вывод результатов
        print("Исходный список:", A)
        print("Элементы с чётными номерами:", result)

    except Exception as e:
        print(f"Ошибка: {e}")

# Запуск программы
if __name__ == "__main__":
    main()
```

Протокол работы:

Шаги выполнения программы:

1. **Инициализация списка:** Программа задаёт исходный список A.
2. **Проверка длины списка:**
   - Проверяется, является ли длина списка N чётным числом.
   - Если длина нечётная, выбрасывается исключение с сообщением: `"Длина списка должна быть чётной."`
3. **Выбор элементов с чётными номерами:**
   - С помощью среза `A[1::2]` выбираются элементы с чётными номерами (индексация начинается с 0).
4. **Вывод результатов:**
   - Программа выводит:
     - Исходный список.
     - Список элементов с чётными номерами.

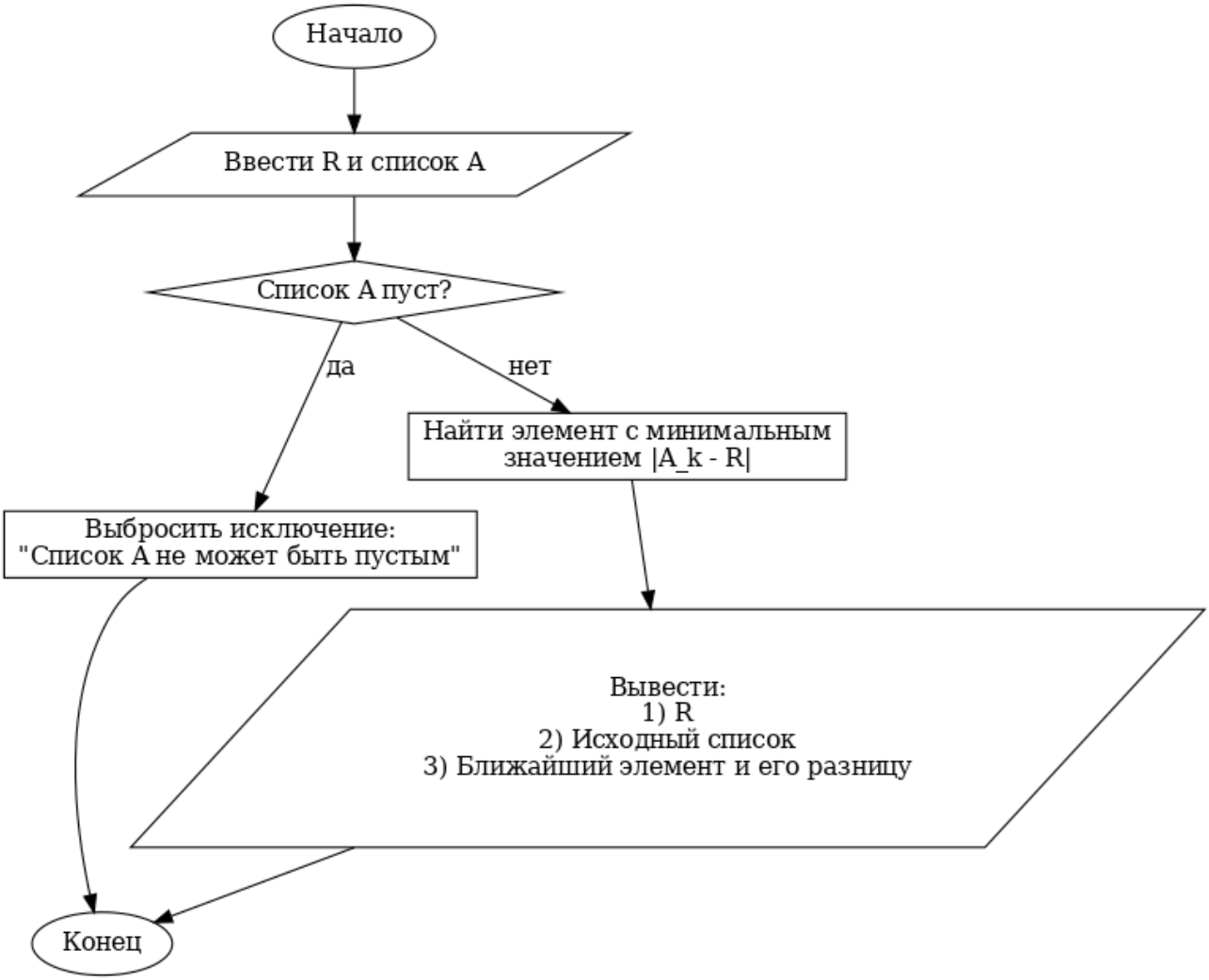
Вывод:

В ходе выполнения практической работы была написана и протестирована программа, которая извлекает элементы с чётными номерами из заданного списка. Программа соответствует требованиям PEP 8, реализована обработка исключений, и успешно решает поставленную задачу.

Постановка задачи №2

Дано число R и список A размера N. Найти элемент списка, который наиболее близок к числу R (то есть такой элемент AK, для которого величина |AK - R| является минимальной).

Блок-схема



Код программы

```
def find_closest_element(R, A):
    try:
        if not A:
            raise ValueError("Список A не может быть пустым.")

        # Ищем элемент, наиболее близкий к числу R
        closest = min(A, key=lambda x: abs(x - R))
        difference = abs(closest - R)

        # Вывод результата
        print("Число R:", R)
        print("Исходный список:", A)
        print(f"Элемент, ближайший к R: {closest} (разница: {difference})")

    except Exception as e:
        print(f"Ошибка: {e}")


# Пример использования
if __name__ == "__main__":
    # Исходные данные
```

```
R = 15
A = [10, 20, 30, 40, 50]

# Вызов функции
find_closest_element(R, A)
```

Протокол работы программы

1. **Исходные данные:**
    - ➓ Число R задаётся пользователем или в коде программы.
    - ➓ Список A содержит произвольные числа, переданные пользователем.
2. **Этапы выполнения:**
    - ➓ Проверяется, что список A не пустой. Если список пуст, выбрасывается исключение.
    - ➓ Для нахождения ближайшего элемента используется встроенная функция `min()` с ключом | Ak−R| |A_k - R|| Ak−R| , которая находит минимальное абсолютное отклонение.
    - ➓ Выводятся R, исходный список A, ближайший элемент и величина его отклонения от R.
3. **Результаты:**
    - ➓ При корректных данных программа возвращает ближайший элемент.
    - ➓ При некорректных входных данных (например, пустом списке) программа сообщает об ошибке.

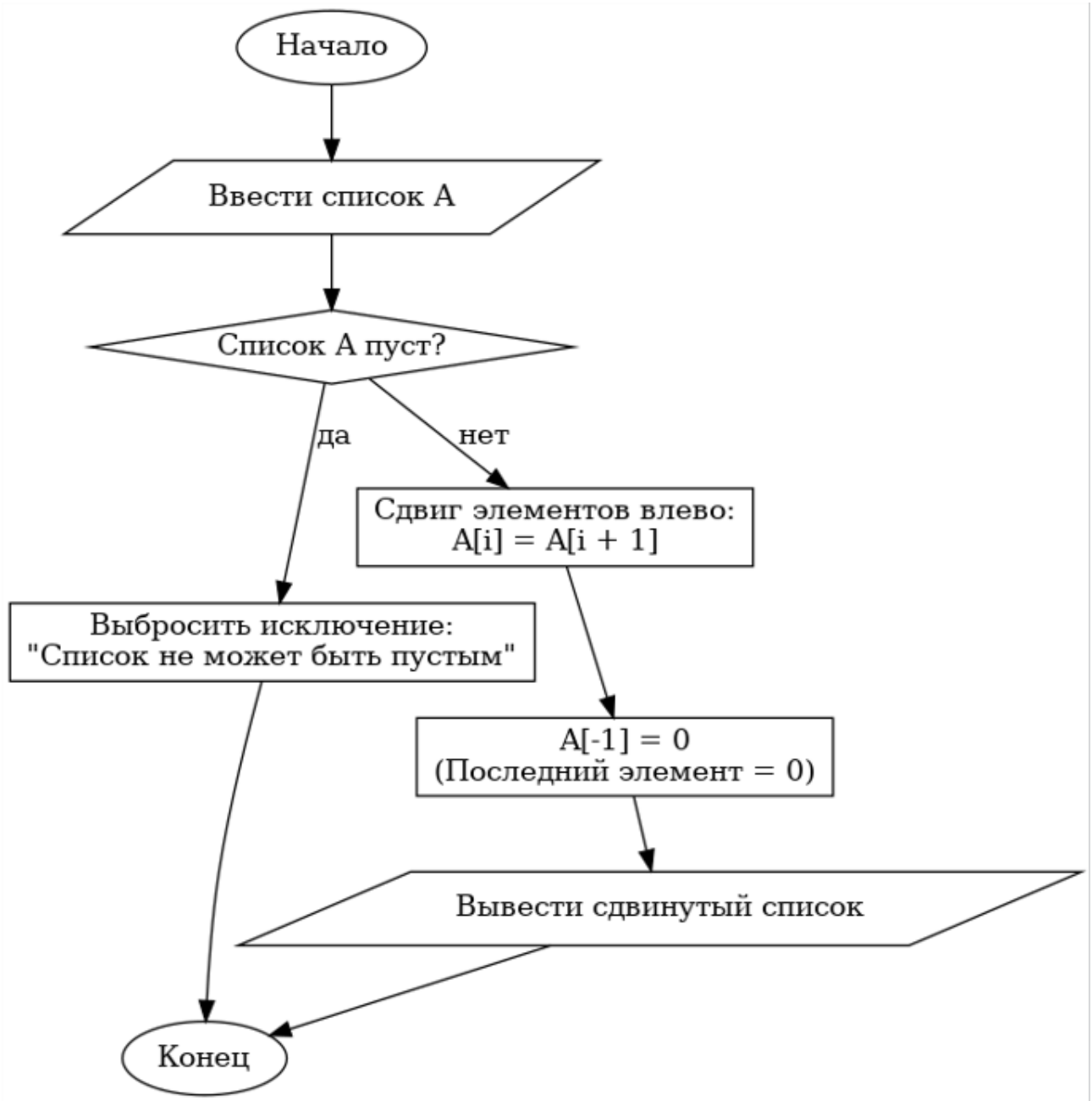
Вывод

В ходе выполнения практической работы была написана и протестирована программа для поиска ближайшего элемента списка к заданному числу RRR. Программа соответствует требованиям PEP 8, содержит обработку исключений и успешно решает поставленную задачу.

Постановка задачи №3

Дан список размера N. Осуществить сдвиг элементов списка влево на одну позицию (при этом AN перейдет в AN-1, AN-1 — в AN-2, .., A2 — в A1, а исходное значение первого элемента будет потеряно). Последний элемент полученного списка положить равным 0.

Блок-схема



Код программы

```
#Дан список размера N. Осуществить сдвиг элементов списка влево на одну позицию
#(при этом AN перейдет в AN-1, AN-1 — в AN-2, .., A2 — в A1, а исходное значение
#первого элемента будет потеряно). Последний элемент полученного списка
#положить равным 0.

def shift_left(A):
    try:
        # Проверяем, что список не пустой
        if not A:
            raise ValueError("Список не может быть пустым.")

        # Сдвиг элементов влево
        N = len(A)
        for i in range(N - 1):
            A[i] = A[i + 1]  # Перемещаем каждый элемент влево

        # Последний элемент становится равным 0
```

```
        A[-1] = 0

        # Вывод результата
        print("Результат после сдвига:", A)

    except Exception as e:
        print(f"Ошибка: {e}")


# Пример использования
if __name__ == "__main__":
    # Исходные данные
    A = [10, 20, 30, 40, 50]

    print("Исходный список:", A)
    shift_left(A)
```

Протокол работы

- Пользователь задаёт список A длины N.
- Проверяется, что список не пустой.
- Каждый элемент списка (кроме последнего) сдвигается влево: A[i]=A[i+1]A[i] = A[i + 1]A[i]=A[i+1].
- Последний элемент списка заменяется на 0.
- **Результаты:**
- Программа возвращает сдвинутый список, где последний элемент равен 0.
- При некорректных данных (например, пустом списке) выводится сообщение об ошибке.

Вывод:
В ходе выполнения практической работы была написана и протестирована программа, реализующая сдвиг элементов списка влево на одну позицию. Последний элемент списка заменяется на 0. Программа соответствует требованиям PEP 8, включает обработку исключений и корректно решает поставленную задачу.