Студента группы ИС-23 Оганнисяна Г.А.

Практическое занятие №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.

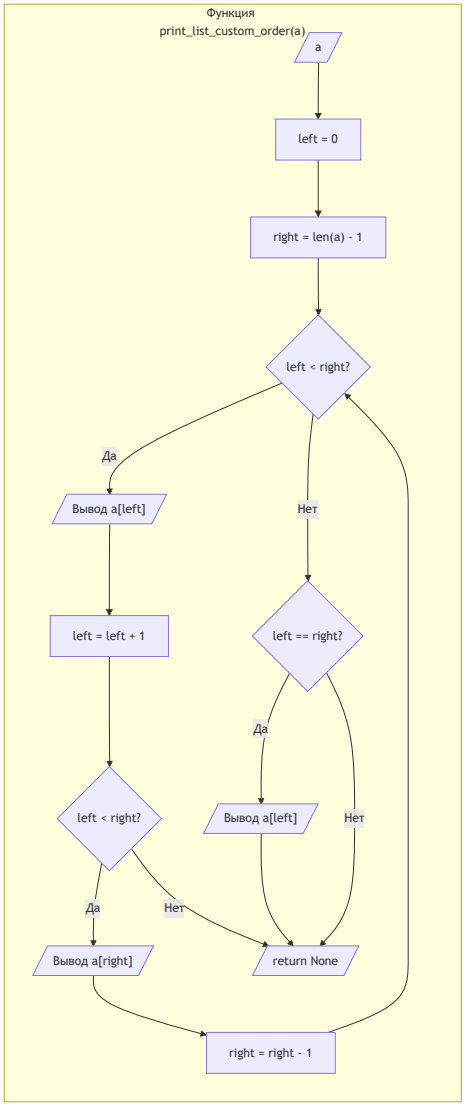
Задание №1

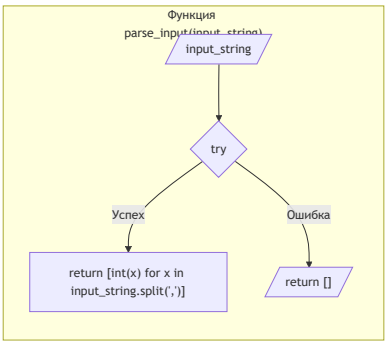
Постановка задачи:

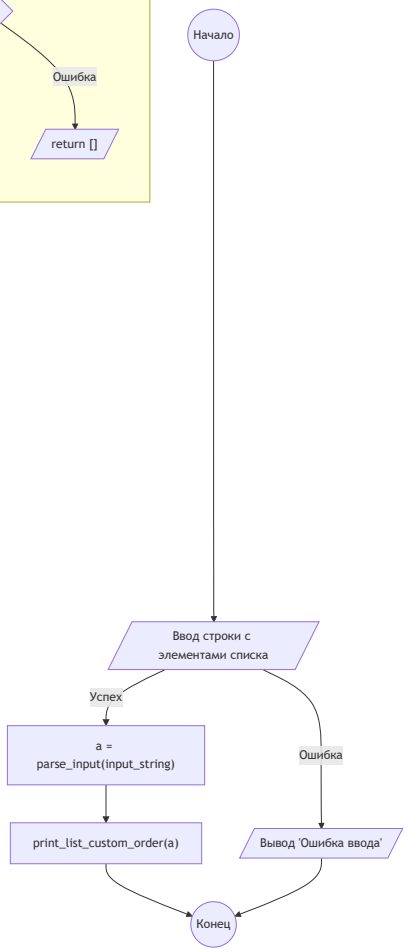
Дан список A. Элементы списка вводятся в формате "1, 2, 3, 4". Вывести элементы списка в следующем порядке: A[1], A[2], A[N], A[N-1], A[3], A[4], A[N-2], A[N-3], … .

Тип алгоритма: циклический.

Блок-схема алгоритма:







Текст программы:

```
def parse_input(input_string):
    """Парсит строку с элементами списка, разделенными запятыми, в список целых чисел."""
    try:
        return [int(x) for x in input_string.split(',')]
    except ValueError:
        return []
```

```python
def print_list_custom_order(a):
    """Выводит элементы списка в заданном порядке."""
    left = 0
    right = len(a) - 1

    while left < right:
        print(a[left], end=" ")
        left += 1
        if left < right:
            print(a[right], end=" ")
            right -= 1
    if left == right:
        print(a[left], end=" ")
    print()


try:
    input_string = input("Введите элементы списка через запятую: ")
    a = parse_input(input_string)
    if not a:
        print("Ошибка ввода")
    else:
        print_list_custom_order(a)

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

Протокол работы программы (примеры):

```
Введите элементы списка через запятую: 1,2,3,4,5
1 2 5 4 3

Введите элементы списка через запятую: 10,20,30,40
10 20 40 30

Введите элементы списка через запятую: 1
1

Введите элементы списка через запятую: 1,2,a,4 # обработка некорректного ввода
Ошибка ввода

Введите элементы списка через запятую: 1, 2, 3,  4,  5 # пробелы между запятыми и числами
1 2 5 4 3
```

Вывод:

В ходе выполнения практического задания были закреплены навыки работы со списками, циклами и генераторами списков. Программа принимает ввод в заданном формате, парсит его в список целых чисел и выводит элементы в указанном порядке. Реализована обработка ошибок ввода, а блок-схема отражает структуру программы с использованием subgraph для функций.

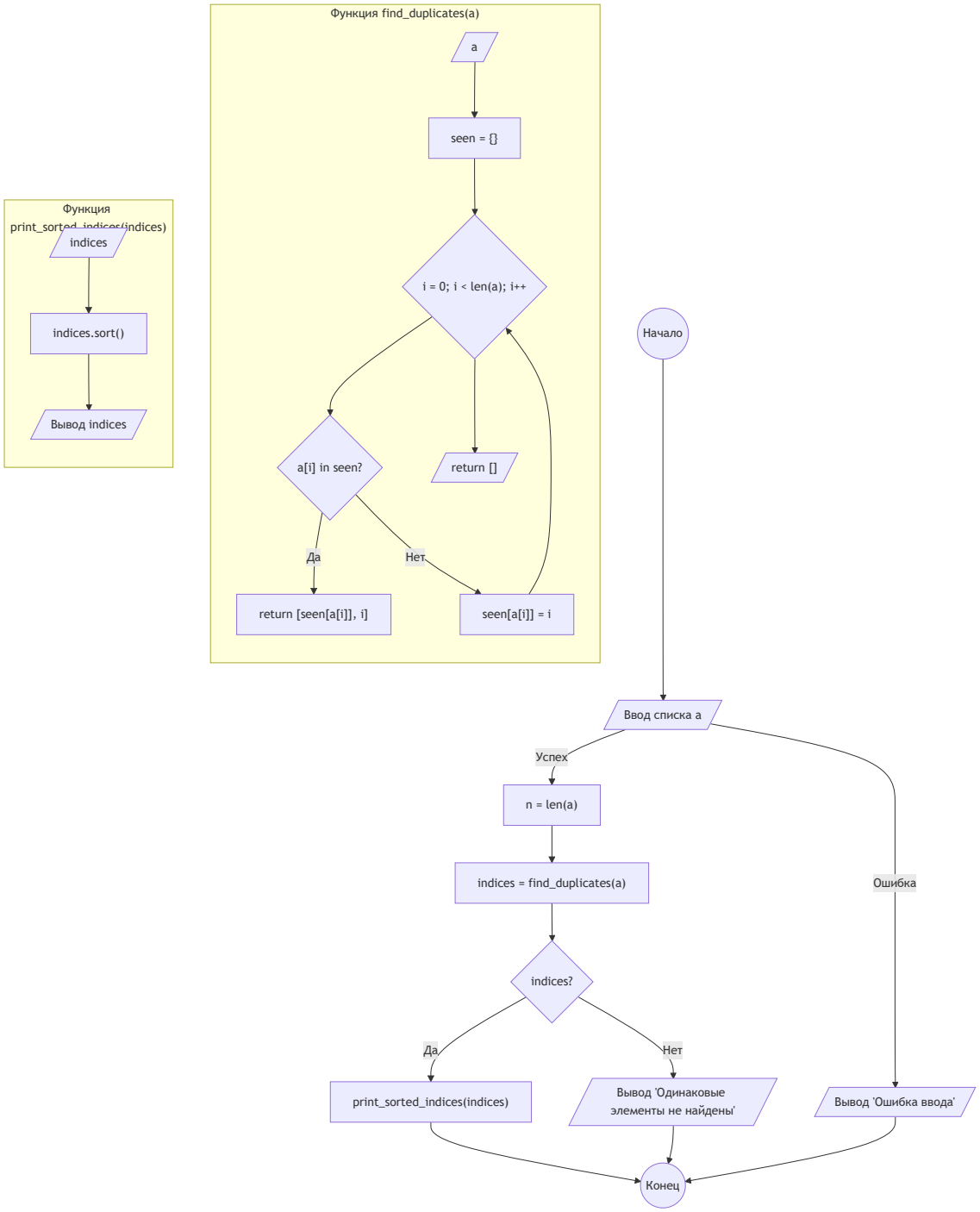
Задание №2

Постановка задачи:

Дан целочисленный список размера N, содержащий ровно два одинаковых элемента. Найти номера одинаковых элементов и вывести эти номера в порядке возрастания.

Тип алгоритма: циклический.

Блок-схема алгоритма:



Текст программы:

```python
def find_duplicates(a):
    """Находит индексы двух одинаковых элементов в списке."""
    seen = {}
    for i, x in enumerate(a):
        if x in seen:
            return [seen[x], i]
        seen[x] = i
    return []


def print_sorted_indices(indices):
    """Выводит индексы в отсортированном порядке."""
    indices.sort()
    print(*indices)


try:
    input_string = input("Введите элементы списка через запятую: ")
    a = [int(x) for x in input_string.split(',')]
    indices = find_duplicates(a)
    if indices:
        print_sorted_indices(indices)
    else:
        print("Одинаковые элементы не найдены")

except ValueError:
    print("Ошибка ввода")
```

Протокол работы программы (примеры):

```
Введите элементы списка через запятую: 1,2,3,2,5
1 3

Введите элементы списка через запятую: 10,20,30,20
1 3

Введите элементы списка через запятую: 1,2,3,4,5 # Нет одинаковых элементов
Одинаковые элементы не найдены

Введите элементы списка через запятую: 1,2,a,4 # Ошибка ввода
Ошибка ввода

Введите элементы списка через запятую: 1,1,1,1 # Больше двух одинаковых элементов, вернет
первые два найденные
0 1
```

Вывод:

В ходе выполнения практического задания были закреплены навыки работы со списками, словарями и циклами. Программа успешно находит индексы первых двух одинаковых элементов в списке и выводит их в

отсортированном порядке. Реализована обработка ошибок ввода и случая, когда одинаковых элементов нет. Блок-схема детализирует логику программы, включая subgraph для функций.

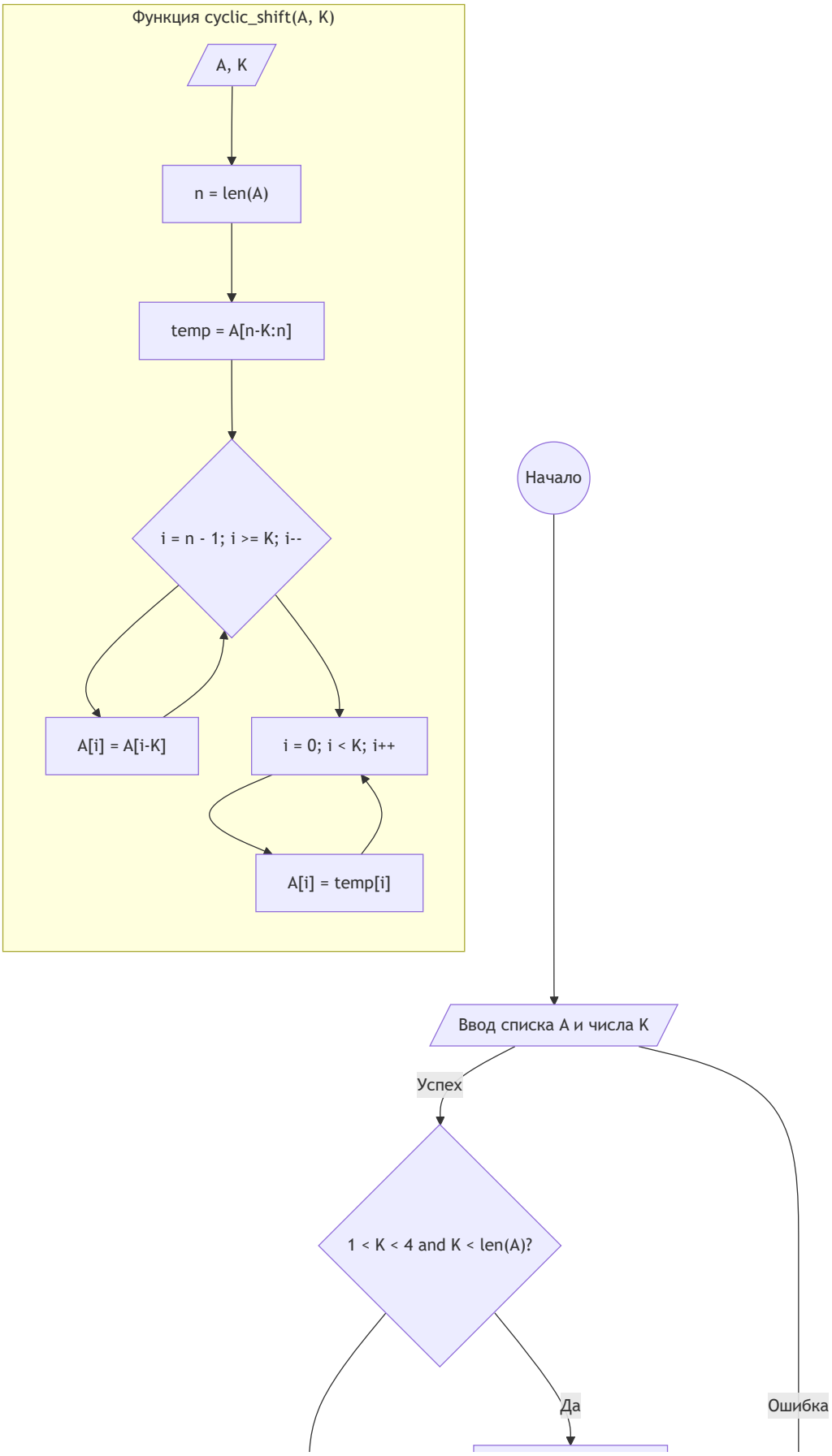
Задание №3

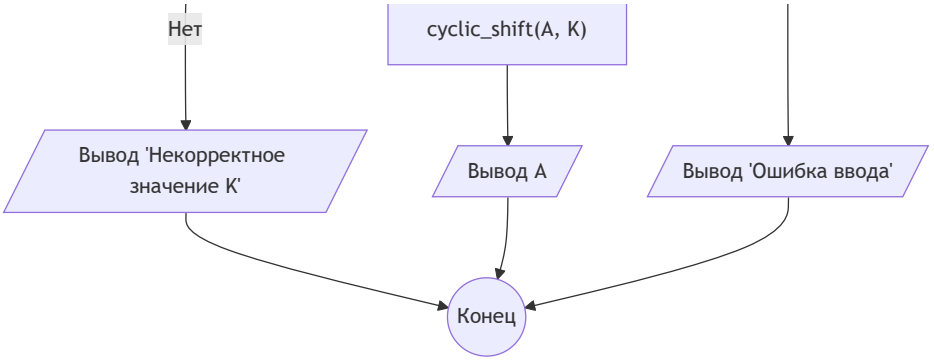
Постановка задачи:

Дан список A размера N и целое число K (1 < K < 4, K < N). Осуществить циклический сдвиг элементов списка вправо на K позиций (при этом A[1] перейдет в A[K+1], A[2] — в A[K+2], ..., A[N] — в A[K]). Допускается использовать вспомогательный список из 4 элементов.

Тип алгоритма: циклический.

Блок-схема алгоритма:


Функция cyclic_shift(A, K)
A, K
n = len(A)
temp = A[n-K:n]
i = n - 1; i >= K; i--
A[i] = A[i-K]
i = 0; i < K; i++
A[i] = temp[i]
Начало
Ввод списка A и числа K
Успех
1 < K < 4 and K < len(A)?
Да
Ошибка



Текст программы:

```
def cyclic_shift(a, k):
    """Осуществляет циклический сдвиг списка a вправо на k позиций."""
    n = len(a)
    temp = a[n - k:]  # Используем срезы для создания копии последних k элементов
    for i in range(n - 1, k - 1, -1):
        a[i] = a[i - k]
    for i in range(k):
        a[i] = temp[i]


try:
    input_string = input("Введите элементы списка через запятую: ")
    a = [int(x) for x in input_string.split(',')]
    k = int(input("Введите число K (1 < K < 4 и K < N): "))

    if 1 < k < 4 and k < len(a):
        cyclic_shift(a, k)
        print(*a)  # Вывод элементов списка через пробел
    else:
        print("Некорректное значение K")

except ValueError:
    print("Ошибка ввода")
```

Протокол работы программы (примеры):

```
Введите элементы списка через запятую: 1,2,3,4,5
Введите число K (1 < K < 4 и K < N): 2
4,5,1,2,3

Введите элементы списка через запятую: 1,2,3,4,5,6,7
Введите число K (1 < K < 4 и K < N): 3
5,6,7,1,2,3,4

Введите элементы списка через запятую: 1,2,3
Введите число K (1 < K < 4 и K < N): 1
Некорректное значение K
```

```
Введите элементы списка через запятую: 1,2,3,4
Введите число K (1 < K < 4 и K < N): 4
Некорректное значение K

Введите элементы списка через запятую: 1,2,3
Введите число K (1 < K < 4 и K < N): 2
2,3,1
```

Вывод:

В ходе выполнения практического задания были закреплены навыки работы со списками, циклами и срезами. Программа осуществляет циклический сдвиг элементов списка вправо на заданное количество позиций, используя вспомогательный список (реализованный через срезы для эффективности). Блок-схема корректно отражает алгоритм, а функция `cyclic_shift` вынесена в `subgraph`. Также реализована проверка корректности вводимых данных.