Студент группы ИС-26 Байрамов Д.Ю.

Практическое занятие № 6

Тема: Составление программ циклической структуры в IDE PyCharm Community. Цель: Закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составления программ циклической структуры в IDE PyCharm Community.

Постановка задач:

1. **Задача 1:**
2. Сформировать и вывести целочисленный список размера 10, содержащий степени двойки от первой до 10-й: 2, 4, 8, 16, ...
3. Тип алгоритма: линейный.

Блок схема:

Текст программы:

```
def generate_powers_of_two():
    """
    Формирует список целых чисел длиной 10, содержащий степени двойки
```

```
    от 2^1 до 2^10.

    :return: list, сформированный список степеней двойки
    """
    # Исходный список степеней двойки от 1 до 10
    powers = [2 ** i for i in range(1, 11)]
    return powers


if __name__ == "__main__":
    try:
        # Формирование списка
        initial_list = list(range(1, 11))  # Для демонстрации "исходного" списка
        result_list = generate_powers_of_two()

        # Вывод исходного и результирующего списков
        print("Исходный список (номера степеней от 1 до 10):", initial_list)
        print("Результирующий список (степени двойки):", result_list)
    except Exception as e:
        print(f"Ошибка: {e}")
```

Протокол работы программы:

Исходный список (номера степеней от 1 до 10): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Результирующий список (степени двойки): [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]

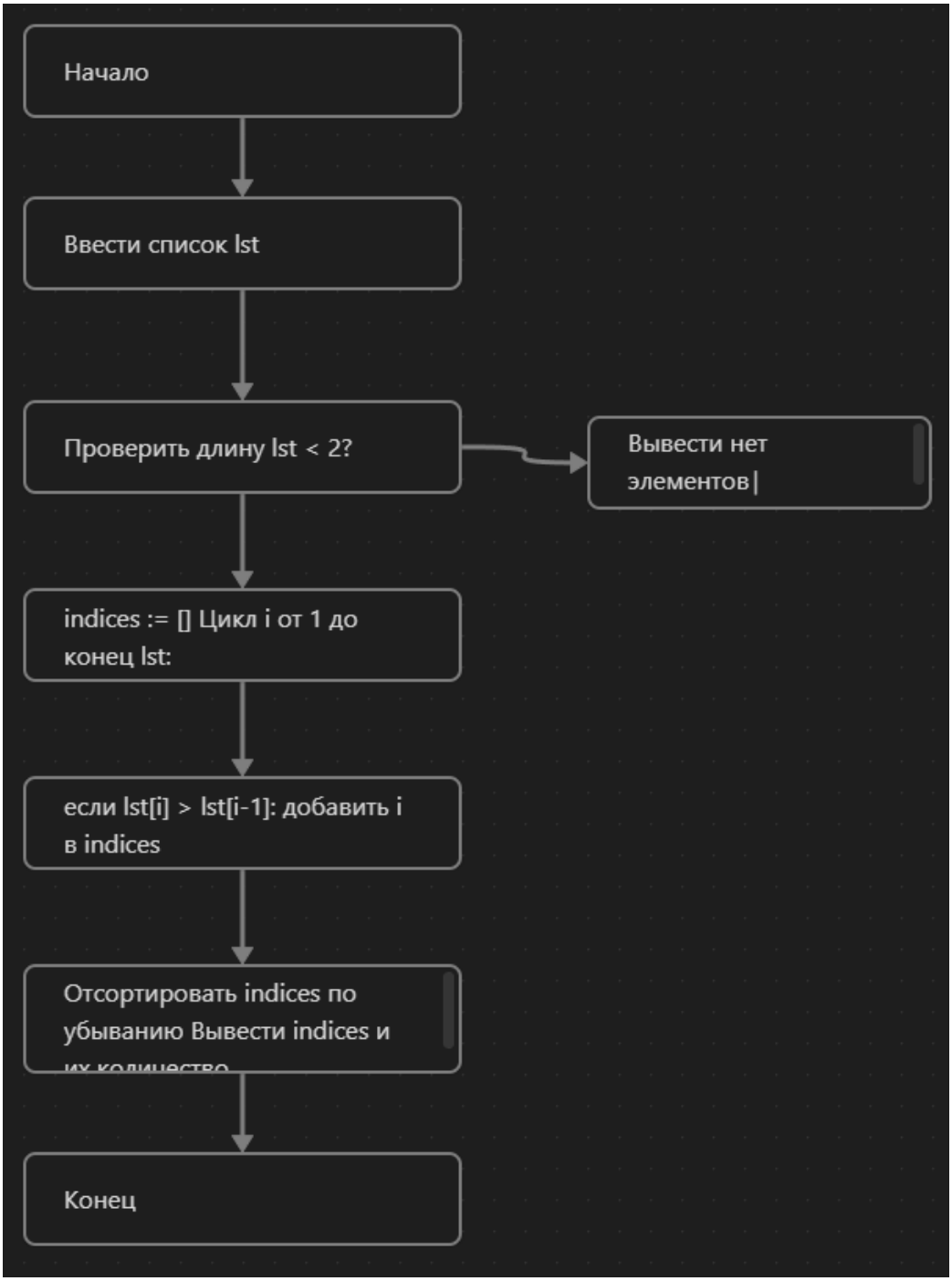
Постановка задач:

Задача 2:

Дан список размера N. Найти номера тех элементов списка, которые больше своего левого соседа, и количество таких элементов. Найденные номера выводить в порядке их убывания.

Тип алгоритма: циклический

Блок схема:



Текст программы:

```
def find_increasing_elements_indices(lst):
    """
```

```
Находит индексы элементов списка, которые больше своего левого соседа,
а также их количество. Индексы выводятся в порядке убывания.

:param lst: list, входной список чисел
:return: tuple(list, int), кортеж из списка индексов (в убывающем порядке)
       и количества таких элементов
"""
try:
   if not isinstance(lst, list):
      raise ValueError("Входные данные должны быть списком.")
   if len(lst) < 2:
      # Если длина меньше 2, нельзя сравнить с левым соседом
      return [], 0

   indices = []
   for i in range(1, len(lst)):
      if lst[i] > lst[i - 1]:
         indices.append(i)
   # Сортируем индексы в порядке убывания
   indices.sort(reverse=True)

   return indices, len(indices)
except Exception as e:
   print(f"Ошибка: {e}")
   return [], 0
```

```
if __name__ == "__main__":
    try:
        # Пример: пользовательский ввод
        # Для теста можно ввести: 5 3 5 6 2 10
        input_str = input("Введите список чисел через пробел: ")
        # Преобразуем введенную строку в список чисел
        initial_list = list(map(int, input_str.split()))

        indices_list, count = find_increasing_elements_indices(initial_list)

        print("Исходный список:", initial_list)
        print("Индексы элементов, которые больше левого соседа (по убыванию):", indices_list)
        print("Количество таких элементов:", count)
    except ValueError:
        print("Ошибка: введите корректный список целых чисел.")
    except Exception as e:
        print(f"Ошибка: {e}")
```

Протоколы работы программ:

Вход:

Введите список чисел через пробел: 5 3 5 6 2 10

Выход:

Исходный список: [5, 3, 5, 6, 2, 10]

Индексы элементов, которые больше левого соседа (по убыванию): [5, 3, 2]

Количество таких элементов: 3

Постановка задач:

1. Дан список размера N. Обнулить элементы списка, расположенные между его минимальным и максимальным элементами (не включая минимальный и максимальный элементы).
2. Тип алгоритма: линейно-циклический.

Блок схема:

Текст программы:

```
def zero_between_min_and_max(lst):
    """
    Обнуляет элементы списка, расположенные между минимальным и максимальным
    элементами (не включая сами минимальный и максимальный элементы).

    :param lst: list, входной список чисел
    :return: list, результирующий список
    """
    try:
        if not isinstance(lst, list):
            raise ValueError("Входные данные должны быть списком.")
        if len(lst) < 2:
            # Если список слишком короткий, нет смысла что-то обнулять
            return lst

        min_val = min(lst)
        max_val = max(lst)

        min_index = lst.index(min_val)
        max_index = lst.index(max_val)

        # Определяем границы
        start = min(min_index, max_index)
```

```
        end = max(min_index, max_index)

        # Обнулить все элементы между min и max (не включая сами min и max)
        for i in range(start + 1, end):
            lst[i] = 0

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

if __name__ == "__main__":
    try:
        # Пример: ввод
        # Например: 4 7 1 9 3 8
        input_str = input("Введите список чисел через пробел: ")
        initial_list = list(map(int, input_str.split()))

        # Копия для отображения исходного списка
        original_list = initial_list[:]
        result_list = zero_between_min_and_max(initial_list)

        print("Исходный список:", original_list)
        print("Результирующий список:", result_list)
    except ValueError:
        print("Ошибка: введите корректный список целых чисел.")
```

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

Протакол работы

Вход:

Введите список чисел через пробел: 10 5 1 2 3 9 4

Выход:

Исходный список: [10, 5, 1, 2, 3, 9, 4]

Результирующий список: [10, 0, 1, 2, 3, 9, 4]

Вывод:

В ходе выполнения практического занятия были успешно решены три задачи, продемонстрированы навыки:

- Генерация последовательностей и работа со списками.
- Поиск элементов, удовлетворяющих определённым условиям (сравнение с соседями).
- Обнуление части списка по определённым критериям (между минимальным и максимальным элементами).
- Обработка исключений, документирование функций и соответствие коду стилю PEP 8.

Код и отчёт подготовлены для размещения на GitHub.