Студент группы ИС-28 Холкин Е.А.

Практическое занятие № 5

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:
1. Дан список ненулевых целых чисел размера N. Проверить, образуют ли его элементы геометрическую прогрессию. Если образуют, то вывести знаменатель прогрессии, если нет — вывести 0.

Тип алгоритма: линейный

Текст программы #1:

```
#V 24
#1. Дан список ненулевых целых чисел размера N. Проверить,
# образуют ли его элементы геометрическую прогрессию. Если образуют,
# то вывести знаменатель прогрессии, если нет – вывести 0.

def check_geometric_progression(arr):
    n = len(arr)
    if n < 2:
        return 0  # Нельзя проверить прогрессию, если меньше двух элемен-
тов
    # Вычисляем знаменатель прогрессии
    ratio = arr[1] / arr[0]

    for i in range(2, n):
        if arr[i] / arr[i - 1] != ratio:
            return False  # Не образуют геометрическую прогрессию

    return True

sp = []
for i in range(int(input("Введите длину списка: "))):
    sp.append(int(input(f"Введите {i+1}-й элемент списка: ")))

print(check_geometric_progression(sp))
```

Протокол работы программы #1:
Введите длину списка: 4
Введите 1-й элемент списка: 2
Введите 2-й элемент списка: 4
Введите 3-й элемент списка: 8
Введите 4-й элемент списка: 16
True

Постановка задачи #2:
2. Дан целочисленный список A размера N. Переписать в новый целочисленный список B того же размера вначале все элементы исходного списка с четными номерами, а затем — с нечетными: A2, A4, A6, …, A1, A3, A5, … . Условный оператор не использовать.

Тип алгоритма: линейный

Текст программы #2:

```
#V 24
#2. Дан целочисленный список A размера N. Переписать в новый целочислен-
ный
# список B того же размера вначале все элементы исходного списка с чет-
ными
# номерами, а затем — с нечетными: A2, A4, A6, …, A1, A3, A5, … . Услов-
ный
# оператор не использовать.

A = []
for i in range(int(input("Введите количество элементов в списке: "))):
    A.append(int(input(f"Введите {i+1}-й элемент списка A: ")))

print("Первый список:", A)
# Выводим исходный список A

B = A[1::2] + A[0::2]
print("Второй список:", B)
# Выводим список B
```

Протокол работы программы #2:
Введите количество элементов в списке: 5
Введите 1-й элемент списка A: 2
Введите 2-й элемент списка A: 3
Введите 3-й элемент списка A: 4
Введите 4-й элемент списка A: 5
Введите 5-й элемент списка A: 6
Первый список: [2, 3, 4, 5, 6]
Второй список: [3, 5, 2, 4, 6]

Постановка задачи #3:
3. Дано множество A из N точек (точки заданы своими координатами x, y). Найти пару различных точек этого множества с максимальным расстоянием между ними и само это расстояние (точки выводятся в том же порядке, в котором они перечислены при задании множества A).
Расстояние R между точками с координатами (x1, y1) и (x2, y2) вычисляется по формуле:
$R = \sqrt{(x2 - x1)2 + (y2 - y1)2}$.
Для хранения данных о каждом наборе точек следует использовать по два списка: первый список для хранения абсцисс, второй — для хранения ординат.

Текст программы #3:

```
#V 24
#3. Дано множество A из N точек (точки заданы своими координатами x, y).
# Найти пару различных точек этого множества с максимальным расстоянием
# между ними и само это расстояние (точки выводятся в том же порядке, в
# котором они перечислены при задании множества A).
# Расстояние R между точками с координатами (x1, y1) и (x2, y2) вычисляется по формуле:
# R = √(x2 – x1)2 + (y2 – y1)2 .
# Для хранения данных о каждом наборе точек следует использовать по два списка: первый
# список для хранения абсцисс, второй — для хранения ординат.

abciss = []
ordinate = []
max_distance = 0
max_pair = []

for i in range(int(input("Введите количество точек: "))):
    abciss.append(int(input(f"Введите {i+1}-ю абсциссу: ")))
    ordinate.append(int(input(f"Введите {i+1}-ю ординату: ")))

max_pair = [None, None]
max_distance = 0
```

```
for i in range(len(abciss)):
    for j in range(i + 1, len(abciss)):
        # Вычисление расстояния между точками i и j
        distance = ((abciss[j] - abciss[i]) ** 2 + (ordinate[j] - ordinate[i]) ** 2) **
0.5

        # Если текущее расстояние больше максимального, обновляем максимальное расстоя-
ние и пару
        if distance > max_distance:
            max_distance = distance
            max_pair = (i, j)

# Вывод результата
if max_pair[0] is not None and max_pair[1] is not None:
    print(f"Максимальное расстояние: {max_distance}")
    print(f"Пара точек с максимальным расстоянием: ({abciss[max_pair[0]]},
{ordinate[max_pair[0]]}), ({abciss[max_pair[1]]}, {ordinate[max_pair[1]]})")
else:
    print("Нет доступных точек для вычисления расстояния.")
```

Протокол работы программы #3:
Введите количество точек: 3
Введите 1-ю абсциссу: 0
Введите 1-ю ординату: 0
Введите 2-ю абсциссу: 3
Введите 2-ю ординату: 4
Введите 3-ю абсциссу: 1
Введите 3-ю ординату: 1
Максимальное расстояние: 5.0
Пара точек с максимальным расстоянием: (0, 0), (3, 4)

Вывод: в процессе выполнения практического занятия я закрепил усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ ветвящейся структуры в IDE PyCharm Community.

Выполнены разработка кода, отладка, тестирование, оптимизация программного кода. Готовые программные коды выложены на GitHub