Практическое занятие № 6

Тема: составление программ со списками в IDE PyCharm Community.

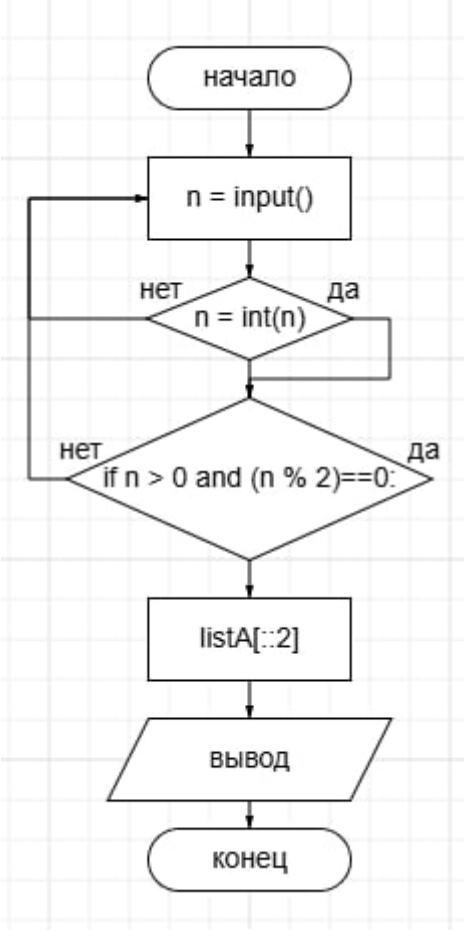
Цель: Закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ со списками в IDE PyCharm Community.

Постановка задачи №1.

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.

Тип алгоритма: линенйный

Блок-схема алгоритма №1:



Текст программы №1:

```
n = input("Введите чётное положительное число >> ")


while True:
    try:
        n = int(n)
        if n > 0 and (n % 2)==0:
            break  # Если все условия выполнены, выходим из цикла
        else:
            print("Число должно быть положительным и чётным")
            n = input("Введите чётное положительное число >>")
    except ValueError:
        print("Неправильно ввели число!")
        n = input("Введите чётное положительное число >>")


listA = []


listA = list(range(n))


even = listA[::2]


print(listA)
print(even)
```

Протокол работы программы:

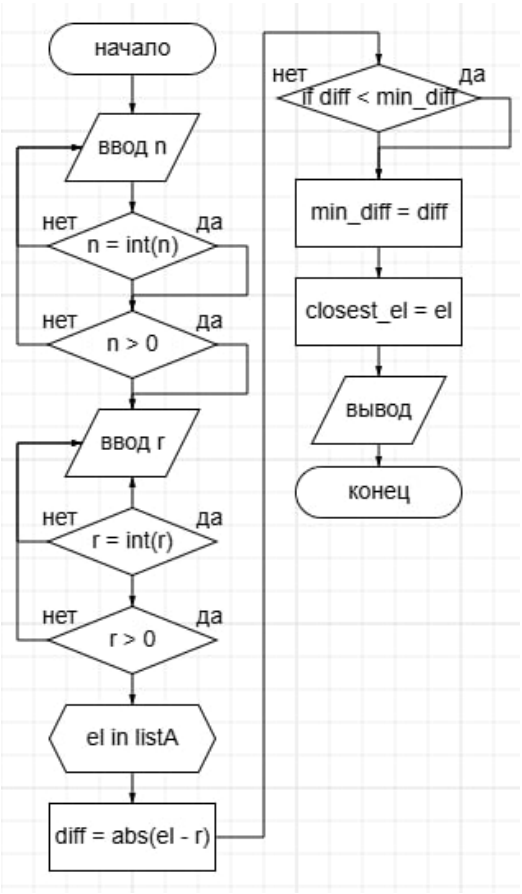
Введите чётное положительное число >> 6
[0, 1, 2, 3, 4, 5]
[0, 2, 4]

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).

Тип алгоритма: циклический

Блок-схема алгоритма №2:



Текст программы №2:

```python
n = input("Введите кол-во элементов в списке >> ")

# Обработка исключений
while True:
    try:
        n = int(n)
        if n > 0:
            break
        else:
            print("Число должно быть положительным")
            n = input("Введите кол-во элементов в списке >> ")
    except ValueError:
        print("Неправильно ввели число!")
        n = input("Введите кол-во элементов в списке >> ")

# Запрашиваем у пользователя число R для поиска ближайшего элемента
r = input("Введите число чтобы найти наиболее близкое к нему >> ")

# Обработка исключений
while type(r) != int:
    try:
        r = int(r)
    except ValueError:
        print("Неправильно ввели число!")
        r = input("Введите число чтобы найти наиболее близкое к нему >> ")

# Создаем список A из элементов от 1 до N
listA = list(range(1, n + 1))

# Инициализируем минимальную разницу как бесконечность
min_diff = float('inf')

# Перебираем все элементы списка A
for el in listA:
    diff = abs(el - r)  # Вычисляем абсолютную разницу между элементом и R
    if diff < min_diff:
        min_diff = diff  # Обновляем минимальную разницу
        closest_el = el  # Запоминаем текущий элемент как ближайший

# Выводим список A и ближайший элемент к числу R
print(f'Список A: {listA}')
print(f'Ближайшее число к R: {closest_el}')
```

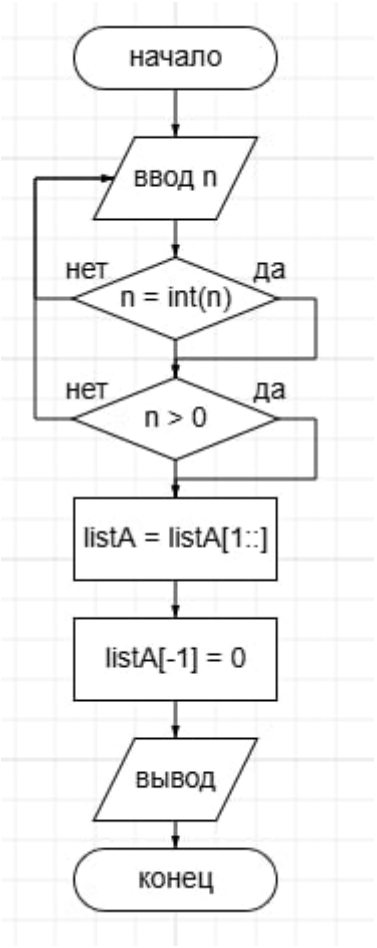
Протокол работы программы:
Введите кол-во элементов в списке >> 6
Введите число чтобы найти наиболее близкое к нему >> 4
Список A: [1, 2, 3, 4, 5, 6]
Ближайшее число к R: 4

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.

Тип алгоритма: линейный

Блок-схема алгоритма №3:



Текст программы №3:

```
n = input("Введите кол-во элементов в списке >> ")


# Обработка исключений
while True:
    try:
        n = int(n)
        if n > 0:
            break
        else:
            print("Число должно быть положительным")
    except ValueError:
        print("Неправильно ввели число!")
    n = input("Введите кол-во элементов в списке >> ")


listA = list(range(1, n + 1))
print("Исходный список:", listA)


# Сдвиг элементов влево на одну позицию
listA = listA[1::]


# Установка последнего элемента равным 0
listA[-1] = 0
print("Изменённый список:", listA)
```

Протокол работы программы:

Вывод: в процессе выполнения практического занятия я закрепил усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрел навыки составления функций в IDE PyCharm Community. Были использованы языковые конструкции while, if, for. Выполнены разработка кода, отладка, тестирование, оптимизация программного кода. Готовый программный код выложен на GitHub.