Программирование на Python

Урок №7

План урока:

1. Сортировка
2. Реализация сортировки
3. Задачи
4. Контрольные вопросы

Сортировка:

В одной из прошлых задач мы запомнили результаты спортсменов и хранили их в списке. Так же, мы выводили результат победителя. На этом занятии мы попробуем распределить спортсменов по местам. В самом деле, было бы очень удобно, если результаты были расположены в порядке возрастания. Мы рассмотрим один из базовых алгоритмов сортировки — пузырек. Попробуем, для начала, сравнивать соседние элементы списка и менять их местами, если предыдущий больше следующего. Пусть, для примера, у нас есть такой список.

i	0	1	2	3
spisok[i]	2	7	0	4

Попробуем пройтись по всем элементам и менять соседние местами, если предыдущий больше последующего. Начнем с элементом spisok[0] и spisok[1].

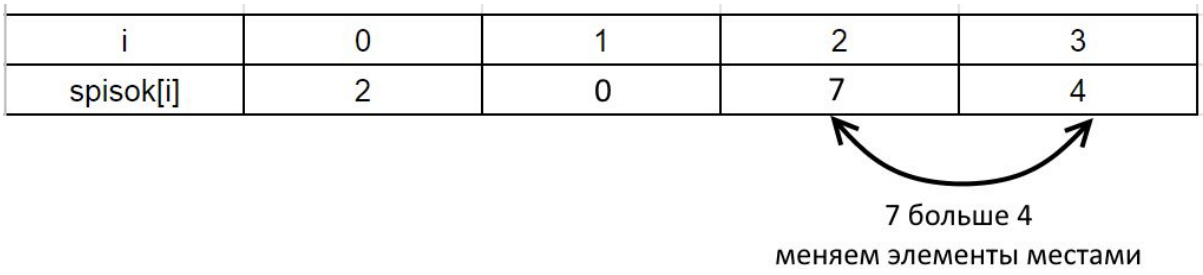
i	0	1	2	3
spisok[i]	2	7	0	4

Теперь применим наш алгоритм к spisok[1] и spisok[2]

i	0	1	2	3
spisok[i]	2	7	0	4

Теперь spisok[2] и spisok[3]

i	0	1	2	3
spisok[i]	2	0	7	4



И вот что у нас получилось.

i	0	1	2	3
spisok[i]	2	0	4	7

Теперь самый большой элемент стоит на последнем месте. Теперь попробуем применить такой алгоритм еще 3 раза для этого списка.

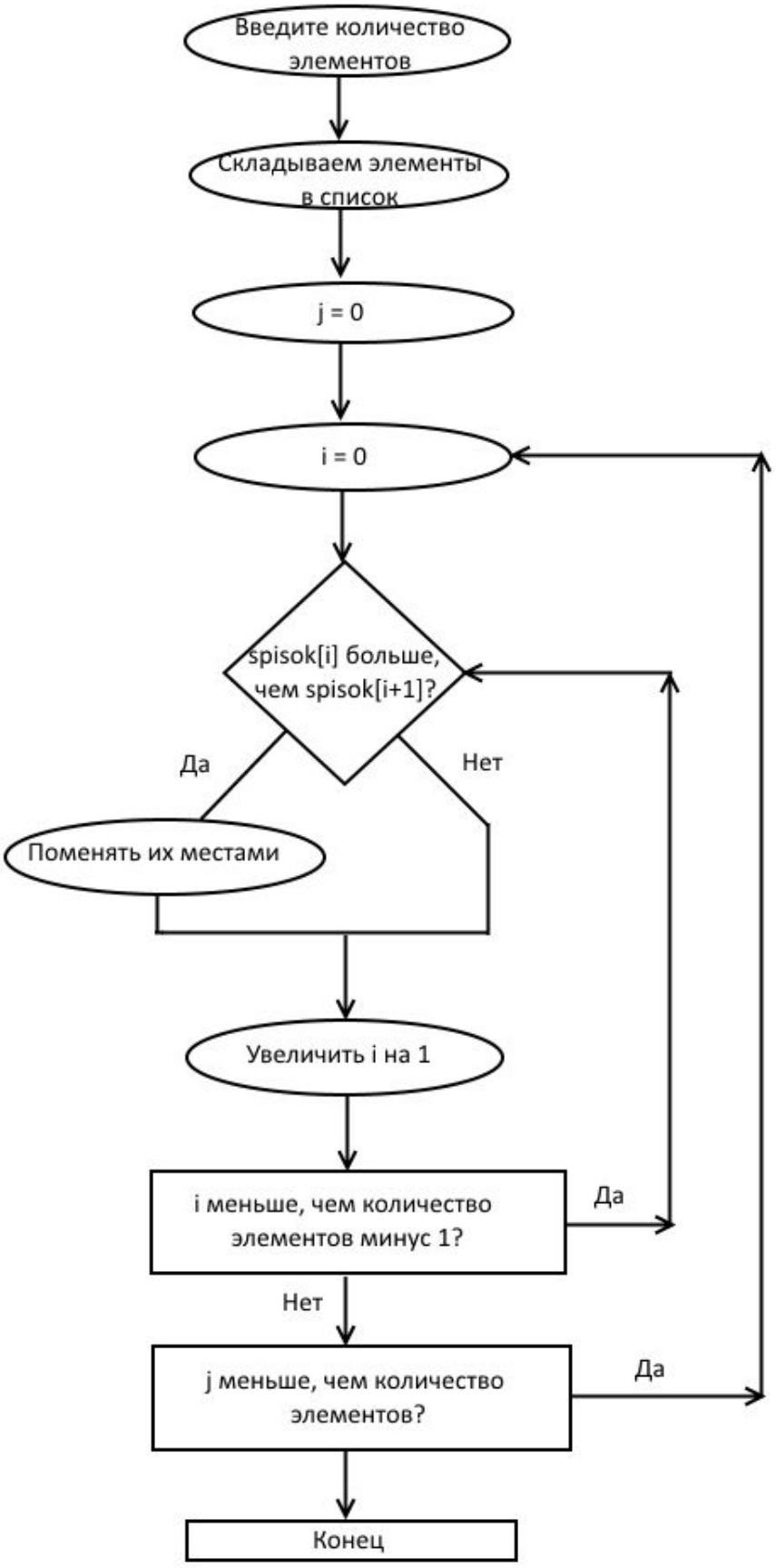
i	0	1	2	3
spisok[i]	0	2	4	7

Элементы в нашем списке теперь стоят в порядке возрастания. Аналогично, мы можем сделать так, чтобы список убывал, если будем менять числа не в случае, если предыдущий элемент больше следующего, а когда он меньше.

Реализация сортировки:

Итак, для того, чтобы сортировать список нам понадобилось сначала повторить сравнения соседних элементов 3 раза. А потом

процедуру сравнения соседних элементов 3 раза, повторить 4 раза. Получается, что мы использовали цикл внутри цикла. Но почему сравниваем мы соседние элементы 3 раза, а повторяем это 4? Все очень просто. Для каждого сравнения нам нужна пара элементов. К примеру, в списке из 2 элементов есть только одна пара. В списке из 3 две пары. В списке из 4 три пары. И так далее. То есть, количество пар всегда на 1 меньше, чем количество элементов. А повторять процедуру сравнения пар нам надо каждый раз ровно столько, сколько элементов в списке. *Задание для детей — составить блок-схему данного алгоритма. Лучше всего разбить задание на две части — сначала нарисовать блок-схему алгоритма сравнения пар в списке, а потом уже к нему добавить блок-схему повторения сравнения пар.*


Введите количество элементов
Складываем элементы в список
j = 0
i = 0
spisok[i] больше, чем spisok[i+1]?
Да
Нет
Поменять их местами
Увеличить i на 1
i меньше, чем количество элементов минус 1?
Да
Нет
j меньше, чем количество элементов?
Да
Конец

Теперь попробуем реализовать нашу задачу вывода результатов спортсменов по местам в порядке возрастания на языке Python. Начнем с того, что напишем программу ввода элементов в список и программу сравнения пар. Необходимо помнить, что пар всегда на 1 меньше, чем количество элементов, а значит повторять сравнения пар надо на 1 раз меньше, чем общее число спортсменов.

```
print("Введите количество спортсменов")
chis_sports = int(input())
spisok = []
i = 0
while i < chis_sports:
    print("Введите результаты спортсмена")
    ves = int(input())
    spisok.append(ves)
    i = i + 1
i = 0
while i < chis_sports - 1:
    if spisok[i] < spisok[i+1]:
        swap = spisok[i]
        spisok[i] = spisok[i+1]
        spisok[i+1] = swap
    i = i + 1
```

Для того, чтобы поменять значения пары элементов, нам необходимо создать буферную переменную. Иначе после первого присваивания, значение элемента spisok[i] пропадет и оба элемента будут равны значению spisok[i+1]. Теперь повторим сравнения пар столько раз, сколько у нас спортсменов.

```
print("Введите количество спортсменов")
chis_sports = int(input())
spisok = []
i = 0
while i < chis_sports:
    print("Введите результаты спортсмена")
    ves = int(input())
    spisok.append(ves)
    i = i + 1
i = 0
j = 0
while j < chis_sports:
    i = 0
    while i < chis_sports - 1:
        if spisok[i] < spisok[i+1]:
            swap = spisok[i]
            spisok[i] = spisok[i+1]
            spisok[i+1] = swap
        i = i + 1
    j = j + 1
print(spisok)
```

Для этого мы создали новую переменную j, так как переменная i будет постоянно меняться внутри циклы сравнения пар. Так же, нам необходимо обнулять переменную i каждый раз, перед тем, как начать сравнивать пары, так как она равна не нулю после выполнения цикла.

Задачи:

1. Сортируйте список из последнего задания в обратном порядке
2. При вводе результатов спортсмена, вводится и его имя. Его необходимо сложить в другой список. Необходимо, чтобы во втором списке результаты и имена у соответствующих одинаковым номерам элементах были правильные.
3. Сортировать программу из задания 2 в обратном порядке

Тайминг:

Тема	Время с начала занятия, мин
Сортировка	25
Реализация сортировки	60
Задачи	85
Программа авторизации	90

Контрольные вопросы:

1. Что такое сортировка?
2. Почему после ее выполнения числа находятся в нужном порядке?
3. Что происходит во внутреннем цикле сортировки?
4. Почему внутренний цикл выполняется на 1 раз меньше?
5. ЗАчем нужен внешний цикл сортировки?