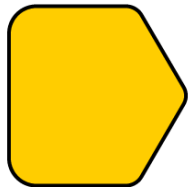

Яндекс

Академия Яндекса

# Вложенные циклы

# Вложенные циклы. Принцип работы

# Вложенные циклы



Циклы называются **вложенными** (т. е. один цикл находится внутри другого), если внутри одного цикла во время каждой итерации необходимо выполнить другой цикл. Так для каждого витка внешнего цикла выполняются все витки внутреннего цикла. Основное требование для таких циклов — чтобы **все** действия вложенного цикла располагались внутри внешнего.

Давайте рассмотрим следующую задачу: необходимо вывести в строку таблицу умножения для заданного числа.

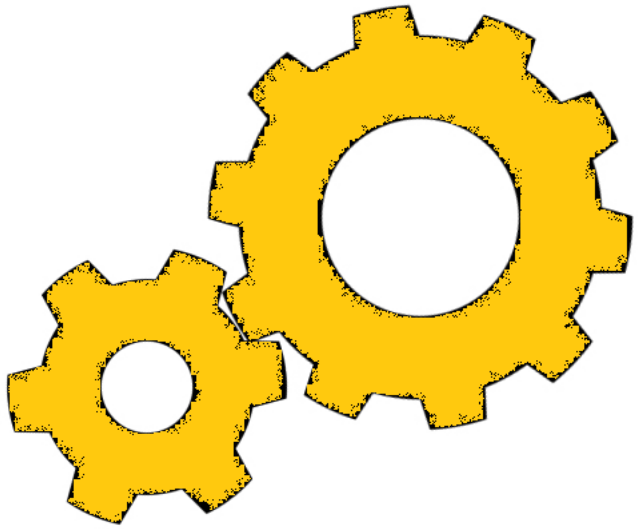
```
k = int(input())
for i in range(1, 10):
    print(i, '*', k, '=', k * i, sep='', end='\t')
```

А если нам нужно вывести таблицу умножения для всех чисел от 1 до k?

```
k = int(input())
for j in range(1, k + 1):
    for i in range(1, 10):
        print(i, '*', j, '=', j * i, sep='', end='\t')
    print()
```

# Вложенные циклы

```
k = int(input())
for j in range(1, k + 1):
    for i in range(1, 10):
        print(i, '*', j, '=', j * i, sep='', end='\t')
    print()
```



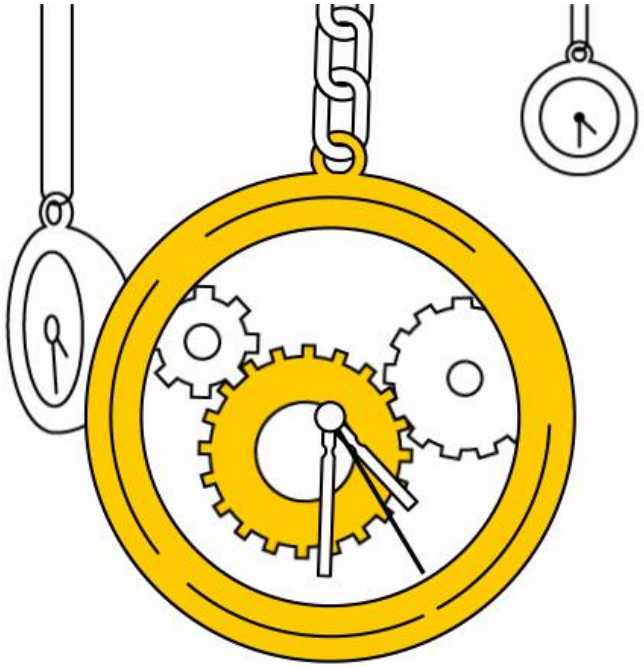
```
3
1*1=1   2*1=2   3*1=3   4*1=4   5*1=5   6*1=6   7*1=7   8*1=8   9*1=9
1*2=2   2*2=4   3*2=6   4*2=8   5*2=10  6*2=12  7*2=14  8*2=16  9*2=18
1*3=3   2*3=6   3*3=9   4*3=12  5*3=15  6*3=18  7*3=21  8*3=24  9*3=27
```

При вложении циклов **внутренний цикл** выполняется полностью от начального до конечного значения параметра, **при неизменном значении параметра внешнего цикла**. Затем *значение параметра внешнего цикла* изменяется на единицу, и опять от начала и до конца выполняется вложенный цикл. И так до тех пор, *пока значение параметра внешнего цикла* не станет больше конечного значения, определенного в операторе **`for`** внешнего цикла.

# Графическое представление вложенных циклов

Внешний цикл — это как бы большая шестерёнка, за один свой оборот (виток цикла), внешний цикл заставляет вращаться вложенный цикл (меньшую шестерёнку) несколько раз.

Обратите внимание, что такая иллюстрация точна в случае, если число повторов вложенного цикла не зависит от того какой именно (1-ый, n-ый или иной) виток делает внешний цикл, а так бывает не всегда.

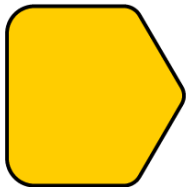


# Операторы **break** и **continue** во вложенных циклах

# Оператор **break** и **continue** во вложенных циклах

Необходимо распечатать все строки таблицы умножения для чисел от **1** до **10**, кроме строки для числа **k**.
Тогда нам нужно будет пропустить выполнение внутреннего цикла, когда придет **k**-ая строка.

```
k = int(input())
for j in range(1, 10):
    if j == k:
        continue
    for i in range(1, 10):
        print(i, '*', j, '=', j * i, sep='', end='\t')
    print()
```



Обратите внимание, если оператор **break** или **continue** расположен внутри вложенного цикла, то он действует именно на вложенный цикл, а не на внешний. Нельзя выскочить из вложенного цикла сразу на самый верхний уровень.

# Пример

Программа учит пользователя вводить числа палиндромы — программа выполняется до тех пор, пока не будет введено число палиндром

```
print('Тренажер по вводу палиндрома:')
while True:
    print('Введите число палиндром:')
    number = n = int(input())
    reverse = 0
    while n > 0:
        reverse = reverse * 10 + n % 10
        n //= 10
    if number == reverse:
        print('Вы ввели палиндром! Программа остановлена.')
        break
    print('Введенное число не палиндром, попробуйте еще раз.')
```


Яндекс