# Лабораторная работа № 4 АЛГОРИТМ ЦИКЛА ПЕРЕБОРА ПОСЛЕДОВАТЕЛЬНОСТИ.

В случаях, когда число повторений заранее известно, для организации циклической обработки информации применяется оператор повтора for. В Python оператор for немного отличается от того, что вы могли использовать в C или Pascal. Вместо того, чтобы всегда выполнять итерацию по арифметической прогрессии чисел (как в Pascal), или давая пользователю возможность определить как шаг итерации, так и условие остановки (как в C), в Python оператор for перебирает элементы любой последовательности (список или строку) в том порядке, в котором они появляются в последовательности.

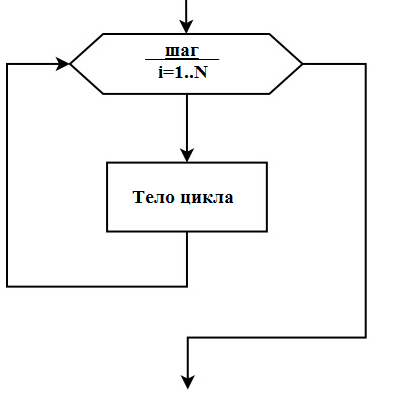


Схема выполнения оператора повтора следующая:

1) задаются последовательность по которой будет происходить итерация;

2) тело цикла будет выполняться до тех пор, пока не будут перебраны все элементы последовательности;

3) после этого управление передается на оператор, следующий за циклом.

Последовательностями для перебора обычно являются списки (будут рассмотрены дальше). Если же необходимо перебрать последовательность чисел, пригодится встроенная функция [range()](http://pythoner.name/documentation/library/functions" \l "range). Она генерирует арифметические прогрессии:

>>> **for** i **in** range(5):

... print(i)

...

Заданная конечная точка никогда не входит в генерируемую последовательность; range(10) генерирует 10 значений, индексируемые как обычная последовательность длинной 10. Возможно установить другое число в качестве начала диапазона или указать другое приращение (даже отрицательное; иногда приращение называют "шагом"):

range(5, 10)

от 5 по 9

range(0, 10, 3)

0, 3, 6, 9

range(-10, -100, -30)

-10, -40, -70

***Задача****.* *Найти сумму всех четных чисел до указанного (без использования условий)*.

***Вербальный алгоритм.***

1. Ввести число в переменную ***n***.
2. Инициализировать переменную ***sum*** отвечающую за сумму

*3.* Запустить цикл по последовательности от 0 до ***n*** с шагом 2.

*4.* Прибавить текущий параметр цикла к переменной ***sum***

*5.*  Вывести сумму на экран.

Начало

Конец

Вывод sum

sum=sum+i

i=0,N (шаг 2)

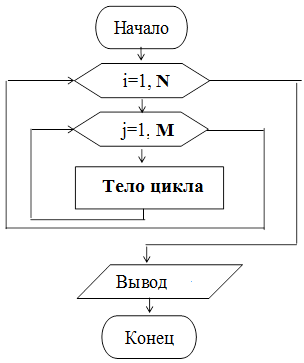
N

***Программная реализация алгоритма.***

n = int(input("Пожалуйста, введите число: "))  
sum = 0  
for i in range(0, n, 2):  
 sum = sum + i  
print(sum)

**ВЛОЖЕННЫЕ ОПЕРАТОРЫ ЦИКЛА. УПРАВЛЕНИЕ ПОТОКОМ ИСПОЛНЕНИЯ ЦИКЛА*.***

Если телом цикла является циклическая структура, то такие циклы называют ***вложенными***. Цикл, содержащий в себе другой цикл, называют ***внешним***, а цикл, содержащийся в теле другого цикла, - ***внутренним***.



Оператор break как и в C, прерывает выполнение самого внутреннего цикла for или while. Оператор continue также заимствован из C, он продолжает выполнение со следующей итерации цикла:

Операторы цикла могут иметь условие else; оно выполняется, когда цикл завершается при исчерпании списка (с for) или когда условие становится ложным (с while), но не в том случае, когда цикл прерван оператором break.

***Задача****.* *Вывести все числа от 2 до определенного числа и пояснения к каждому по следующему признаку:*

*- если число простое, вывести что оно простое*

*-если число не простое, вывести его множители*.

***Вербальный алгоритм.***

1. Ввести число в переменную ***n***.

2. Запустить внешний цикл по последовательности от 2 до ***n*** с параметром ***a***.

3*.* Запустить внутренний цикл по последовательности от 2 до ***a***

С параметром ***b***

4. В теле внутреннего цикла проверить, делится ли число ***a*** на ***b*** без остатка, тем самым определив простое ли оно.

5. Если число составное, то вывести на экран само число ***a*** и его множители, а так же, прекратить выполнение внутреннего цикла.

6. Если ни один множитель не был найден, (условие ***else*** у внутреннего цикла) вывести число ***a*** и сообщение о том, что оно простое.

Начало

Конец

Вывод a простое

a=2,N

N

b=2,a

a % b == 0

Вывод a составное

***Программная реализация алгоритма.***

n = int(input("Пожалуйста, введите число: "))  
for a in range(2, n):  
 for b in range(2, a):  
 if a % b == 0:  
 print(a, 'равен', b, '\*', a//b)  
 break  
 else:  
 print(a, 'простое число')

**ЗАДАЧИ НА САМОСТОЯТЕЛЬНУЮ РАБОТУ.**

1. Даны два целых числа A и В. Выведите все числа от A до B включительно, в порядке возрастания, если A < B, или в порядке убывания в противном случае**.**
2. По данному натуральному n ≤ 9 выведите лесенку из n ступенек, i-я ступенька состоит из чисел от 1 до i без пробелов.
3. Введите число n. Вычислите сумму 1! + 2! + 3! +…+ n!. Пользоваться математической библиотекой math в этой задаче запрещено.
4. Даны два целых числа A и В, A>B. Выведите все нечётные числа от A до B включительно, в порядке убывания. В этой задаче можно обойтись без инструкции if.
5. Вводятся 2 числа. В введенном промежутке натуральных чисел найти те, количество делителей у которых не меньше 3. Для найденных чисел вывести на экран количество делителей и все делители.
6. Ввести 2 числа – высоту елки и символ. Нарисовать елочку из введённого символа заданной высоты. По середине елки должен быть ствол из двух вертикальных слэшей. К примеру для высоты 3 и символа @ елочка должна выглядить так:  
    ||  
    @||@  
   @@||@@  
    ||
7. Вывести на экран таблицу умножения (от 1 до 9). На каждое число должно отводится 2 позиции.
8. Вводится число ***n***. Вывести ***n***-ое число Фибоначчи используя цикл ***for***.