Циклы: две части

Python 3 • от основ до уровня задания №16 ОГЭ

120 минут: 10 минут на памятку, 57 минут на часть 1, 43 минуты на часть 2 и 10 минут на итоговую проверку. Время каждого задания включает разбор примера, написание программы и её запуск.

Часть 1. Цикл for
Задания 1–8, страницы 2–9. От простого вывода чисел до отбора элементов последовательности. Число повторений задаёт range.

Часть 2. Цикл while
Задания 9–14, страницы 10–15. От счётчика до обработки ввода с признаком конца. Цикл продолжается, пока условие истинно.

Как работать с каждой страницей
1. Прочитайте похожую задачу и код её решения.
2. Проследите, как меняются переменные.
3. Решите своё задание: измените границы, условие или действие.
4. Проверьте пример и крайние случаи. Решения своих заданий смотрите после попытки.

Нужные операции
`int(input())` - прочитать целое число;
`x % d` - остаток от деления;
`k += 1` - увеличить количество на один;
`s += x` - увеличить сумму на x;
and - оба условия верны одновременно.

Для задач с системами счисления
Последняя цифра в основании b: x % b.
Две последние цифры: остаток от деления на b^2.
Трёхзначное число в основании b: $b^2 \le x < b^3$.

Ввод и вывод
Точка с запятой в примерах разделяет строки. Каждое число вводите с новой строки. Не добавляйте в программу приглашения ко вводу. Отступ внутри цикла и if - 4 пробела.

Решения: страницы 16–20. Краткие ответы: страница 21.

Часть 1 • Цикл for

Шаг 1. Первые повторения • 3 минут

Сначала разберите аналогичный пример

Вывести числа от 2 до 4 включительно.

```
for i in range(2, 5):
    print(i)
```

Правая граница 5 не включается. Переменная i по очереди принимает значения 2, 3, 4.

Ввод примера: нет
Вывод примера: 2; 3; 4

Теперь решите задание 1

Используя for, выведите числа от 1 до 5 включительно, каждое на отдельной строке. Ввод не требуется.

Ввод: нет
Вывод: 1; 2; 3; 4; 5

Что нужно изменить по сравнению с примером?

_____

_____

Проверено: ☐ пример ☐ граничные значения
Мой дополнительный тест: _____

Часть 1 • Цикл for

Шаг 2. Чётные числа до n • 5 минут

Сначала разберите аналогичный пример

Дано n. Вывести положительные нечётные числа, не превосходящие n.

```
n = int(input())
for i in range(1, n + 1, 2):
    print(i)
```

Начинаем с 1 и прибавляем 2. Верхняя граница n + 1 позволяет включить n, если оно нечётное.

Ввод примера: 7
Вывод примера: 1; 3; 5; 7

Теперь решите задание 2

Дано натуральное n (1 ≤ n ≤ 100). Через for выведите все чётные числа от 2 до n включительно, каждое на новой строке. Если их нет, ничего не выводите.

Ввод: 9
Вывод: 2; 4; 6; 8

Что нужно изменить по сравнению с примером?

_____

_____

Проверено: ☐ пример ☐ граничные значения
Мой дополнительный тест: _____

Часть 1 • Цикл for

Шаг 3. Читаем ровно N чисел • 6 минут

Сначала разберите аналогичный пример

Дано N, затем N натуральных чисел. Найти сумму их квадратов.

```
n = int(input())
s = 0
for i in range(n):
    x = int(input())
    s += x * x
print(s)
```

Первое число задаёт количество повторений. Каждое следующее число возводим в квадрат и прибавляем к s.

Ввод примера: 3; 2; 3; 4
Вывод примера: 29

Теперь решите задание 3

Сначала вводится N (1 ≤ N ≤ 100), затем N натуральных чисел. Найдите их сумму с помощью for. Число N в сумму не входит.

Ввод: 4; 3; 8; 2; 7
Вывод: 20

Что нужно изменить по сравнению с примером?

_____

_____

Проверено: ☐ пример ☐ граничные значения
Мой дополнительный тест: _____

Часть 1 • Цикл for

Шаг 4. Сколько чисел кратно 3? • 7 минут

Сначала разберите аналогичный пример

Дано N, затем N натуральных чисел. Посчитать чётные числа.

```
n = int(input())
k = 0
for i in range(n):
    x = int(input())
    if x % 2 == 0:
        k += 1
print(k)
```

У чётного числа остаток при делении на 2 равен 0. За каждое подходящее число увеличиваем k на 1.

Ввод примера: 4; 2; 5; 8; 9
Вывод примера: 2

Теперь решите задание 4

Сначала вводится N (1 ≤ N ≤ 100), затем N натуральных чисел. Выведите количество чисел, делящихся на 3 без остатка. Используйте for.

Ввод: 5; 3; 4; 9; 10; 12
Вывод: 3

Что нужно изменить по сравнению с примером?

_____

_____

Проверено: ☐ пример ☐ граничные значения
Мой дополнительный тест: _____

Часть 1 • Цикл for

Шаг 5. Два условия одновременно • 8 минут

Сначала разберите аналогичный пример

Дано N, затем N натуральных чисел. Посчитать трёхзначные числа, кратные 5.

```
n = int(input())
k = 0
for i in range(n):
    x = int(input())
    if 100 <= x <= 999 and x % 5 == 0:
        k += 1
print(k)
```

Оба требования должны выполниться одновременно: число от 100 до 999 и остаток при делении на 5 равен 0.

Ввод примера: 4; 95; 105; 230; 1000
Вывод примера: 2

Теперь решите задание 5

Сначала вводится N (1 ≤ N ≤ 1000), затем N натуральных чисел не больше 30 000. Посчитайте двузначные числа, кратные 4. Используйте for.

Ввод: 6; 8; 12; 24; 99; 100; 96
Вывод: 3

Что нужно изменить по сравнению с примером?

_____

_____

Проверено: ☐ пример ☐ граничные значения
Мой дополнительный тест: _____

Часть 1 • Цикл for

Шаг 6. Максимум среди подходящих • 8 минут

Сначала разберите аналогичный пример

Дано N, затем N натуральных чисел. Найти максимум среди чётных, либо 0, если их нет.

```
n = int(input())
best = 0
for i in range(n):
    x = int(input())
    if x % 2 == 0 and x > best:
        best = x
print(best)
```

best хранит лучший найденный результат. Меняем его только для чётного числа, которое больше текущего best.

Ввод примера: 4; 3; 14; 8; 5
Вывод примера: 14

Теперь решите задание 6

Сначала вводится N (1 ≤ N ≤ 1000), затем N натуральных чисел не больше 30 000. Найдите наибольшее число, кратное 5. Если таких нет, выведите 0. Используйте for.

Ввод: 5; 7; 15; 40; 9; 25
Вывод: 40

Что нужно изменить по сравнению с примером?

_____

_____

Проверено: ☐ пример ☐ граничные значения
Мой дополнительный тест: _____

Часть 1 • Цикл for

Шаг 7. Последняя цифра в основании 7 • 9 минут

Сначала разберите аналогичный пример

Дано N, затем N натуральных десятичных чисел. Найти сумму чисел, запись которых в основании 5 оканчивается цифрой 2.

```
n = int(input())
s = 0
for i in range(n):
    x = int(input())
    if x % 5 == 2:
        s += x
print(s)
```

Последняя цифра в основании b равна x % b. В примере подходят 2, 7 и 12; складываем исходные десятичные числа.

Ввод примера: 4; 2; 7; 12; 10
Вывод примера: 21

Теперь решите задание 7

Сначала вводится N (1 ≤ N ≤ 1000), затем N натуральных десятичных чисел не больше 30 000. Найдите сумму чисел, запись которых в семеричной системе оканчивается цифрой 1. Выведите сумму в десятичной системе. Используйте for.

Ввод: 5; 1; 8; 15; 16; 50
Вывод: 74

Что нужно изменить по сравнению с примером?

_____

_____

Проверено: ☐ пример ☐ граничные значения
Мой дополнительный тест: _____

Часть 1 • Цикл for

Шаг 8. Самостоятельно: две последние цифры • 11 минут

Сначала разберите аналогичный пример

Дано N, затем N натуральных десятичных чисел. Посчитать числа с трёхзначной записью в основании 5, оканчивающейся цифрами 23.

```
n = int(input())
k = 0
for i in range(n):
    x = int(input())
    if 25 <= x <= 124 and x % 25 == 13:
        k += 1
print(k)
```

Трёхзначная запись: $5^2 \le x < 5^3$, то есть 25…124. Конец 23 в основании 5 равен $2 \cdot 5 + 3 = 13$; проверяем остаток от деления на 25. Подходят 38 и 63.

Ввод примера: 4; 13; 38; 63; 138
Вывод примера: 2

Теперь решите задание 8

Сначала вводится N ($1 \le N \le 1000$), затем N натуральных десятичных чисел не больше 30 000. Посчитайте числа, запись которых в восьмеричной системе трёхзначна и оканчивается цифрами 16. Выведите одно число - найденное количество. Используйте for.

Ввод: 6; 14; 78; 142; 462; 526; 511
Вывод: 3

Что нужно изменить по сравнению с примером?

_____

_____

Проверено: ☐ пример ☐ граничные значения
Мой дополнительный тест: _____

Часть 2 • Цикл while

Шаг 9. Первые повторения через while • 4 минут

Сначала разберите аналогичный пример

Вывести числа от 3 до 6 включительно через while.

```
i = 3
while i <= 6:
    print(i)
    i += 1
```

Пока i не больше 6, печатаем i и увеличиваем его на 1. После вывода 6 получаем i = 7 и завершаем цикл.

Ввод примера: нет
Вывод примера: 3; 4; 5; 6

Теперь решите задание 9

Используя while, выведите числа от 1 до 5 включительно, каждое на отдельной строке. Ввод не требуется. Перед циклом создайте переменную i.

Ввод: нет
Вывод: 1; 2; 3; 4; 5

Что нужно изменить по сравнению с примером?

_____

_____

Проверено: ☐ пример ☐ граничные значения
Мой дополнительный тест: _____

Часть 2 • Цикл while

Шаг 10. Сумма от 1 до n • 6 минут

Сначала разберите аналогичный пример

Дано натуральное n. Найти сумму чётных чисел от 2 до n.

```
n = int(input())
s = 0
i = 2
while i <= n:
    s += i
    i += 2
print(s)
```

s накапливает сумму. Переменная i принимает только чётные значения. При n = 1 цикл не выполнится, сумма останется нулевой.

Ввод примера: 6
Вывод примера: 12

Теперь решите задание 10

Дано натуральное n ≤ 1000. Вычислите сумму всех целых чисел от 1 до n включительно. Используйте while, не используйте готовую формулу суммы.

Ввод: 5
Вывод: 15

Что нужно изменить по сравнению с примером?

_____

_____

Проверено: ☐ пример ☐ граничные значения
Мой дополнительный тест: _____

Часть 2 • Цикл while

Шаг 11. Ввод до нуля: количество • 7 минут

Сначала разберите аналогичный пример

Вводятся неотрицательные целые числа, затем -1. Посчитать числа до -1, не учитывая признак конца.

```
k = 0
x = int(input())
while x != -1:
    k += 1
    x = int(input())
print(k)
```

Первое чтение стоит до цикла; повторное - внутри. Здесь признак конца равен -1, поэтому введённый 0 тоже считается.

Ввод примера: 4; 0; 7; -1
Вывод примера: 3

Теперь решите задание 11

Вводятся натуральные числа, затем 0. Количество чисел заранее неизвестно, их не более 1000. Через while посчитайте числа до нуля. Сам 0 не учитывайте; он может быть первым.

Ввод: 7; 4; 9; 0
Вывод: 3

Что нужно изменить по сравнению с примером?

_____

_____

Проверено: ☐ пример ☐ граничные значения
Мой дополнительный тест: _____

Часть 2 • Цикл while

Шаг 12. Ввод до нуля: сумма чётных • 8 минут

Сначала разберите аналогичный пример

Вводятся натуральные числа, затем 0. Найти сумму нечётных чисел; при их отсутствии вывести 0.

```
s = 0
x = int(input())
while x != 0:
    if x % 2 != 0:
        s += x
    x = int(input())
print(s)
```

Проверяем нечётность и прибавляем x к сумме. Новое число читаем вне if, чтобы продолжать ввод при любом x.

Ввод примера: 3; 8; 5; 0
Вывод примера: 8

Теперь решите задание 12

Вводятся натуральные числа, затем 0. Чисел не более 1000. Найдите сумму чётных чисел до нуля через while. Если чётных чисел нет, выведите 0.

Ввод: 5; 8; 2; 7; 0
Вывод: 10

Что нужно изменить по сравнению с примером?

_____

_____

Проверено: ☐ пример ☐ граничные значения
Мой дополнительный тест: _____

Часть 2 • Цикл while

Шаг 13. Минимум с признаком конца • 8 минут

Сначала разберите аналогичный пример

Вводятся натуральные числа, затем 0. Найти наименьшее однозначное число; если его нет, вывести 0.

```
best = 10
x = int(input())
while x != 0:
    if 1 <= x <= 9 and x < best:
        best = x
    x = int(input())
if best == 10:
    print(0)
else:
    print(best)
```

10 больше любого однозначного натурального числа. Если best осталось равным 10, подходящих чисел не было.

Ввод примера: 12; 8; 3; 0
Вывод примера: 3

Теперь решите задание 13

Вводятся натуральные числа не больше 30 000, затем 0. Чисел не более 1000. Найдите наименьшее двузначное число. Если его нет, выведите 0. Используйте while.

Ввод: 7; 42; 18; 105; 0
Вывод: 18

Что нужно изменить по сравнению с примером?

_____

_____

Проверено: ☐ пример ☐ граничные значения
Мой дополнительный тест: _____

Часть 2 • Цикл while

Шаг 14. Самостоятельно: отбор и сумма • 10 минут

Сначала разберите аналогичный пример

Вводятся натуральные числа, затем 0. Найти сумму двузначных десятичных чисел, оканчивающихся цифрой 7.

```
s = 0
x = int(input())
while x != 0:
    if 10 <= x <= 99 and x % 10 == 7:
        s += x
    x = int(input())
print(s)
```

Проверяем диапазон и последнюю цифру. Подходят 17 и 27. Число 7 слишком мало, а 107 уже трёхзначное.

Ввод примера: 7; 17; 27; 107; 0
Вывод примера: 44

Теперь решите задание 14

Вводятся натуральные десятичные числа не больше 30 000, затем 0 (признак конца, не входит в последовательность). Чисел не более 1000. Найдите сумму трёхзначных десятичных чисел, оканчивающихся цифрой 4. Если таких нет, выведите 0. Используйте while.

Ввод: 24; 104; 314; 999; 1004; 0
Вывод: 418

Что нужно изменить по сравнению с примером?

_____

_____

Проверено: ☐ пример ☐ граничные значения
Мой дополнительный тест: _____

Решения 1–3

1. Первые повторения

```
for i in range(1, 6):
    print(i)
```

Правая граница range не включается: range(1, 6) даёт 1, 2, 3, 4, 5.

Проверка. Выведите от 1 до 1: должна получиться одна строка.

2. Чётные числа до n

```
n = int(input())
for i in range(2, n + 1, 2):
    print(i)
```

Шаг 2 перебирает только чётные числа. Граница n + 1 позволяет включить n.

Проверка. При n = 1 вывод пустой; при n = 2 выводится 2.

3. Читаем ровно N чисел

```
n = int(input())
s = 0
for i in range(n):
    x = int(input())
    s += x
print(s)
```

range(n) выполняет тело ровно n раз. Каждый вызов input читает очередное число.

Проверка. Ввод 1; 9 даёт 9, а не 10.

Решения 4–6

4. Сколько чисел кратно 3?

```
n = int(input())
k = 0
for i in range(n):
    x = int(input())
    if x % 3 == 0:
        k += 1
print(k)
```

Остаток x % 3 равен нулю для кратных 3. Счётчик увеличивается на 1, а не на x.

Проверка. Ввод 2; 1; 2 даёт 0.

5. Два условия одновременно

```
n = int(input())
k = 0
for i in range(n):
    x = int(input())
    if 10 <= x <= 99 and x % 4 == 0:
        k += 1
print(k)
```

Двузначность означает 10 ≤ x ≤ 99. Оператор and требует выполнения обоих условий.

Проверка. Ввод 4; 8; 12; 96; 100 даёт 2.

6. Максимум среди подходящих

```
n = int(input())
best = 0
for i in range(n):
    x = int(input())
    if x % 5 == 0 and x > best:
        best = x
print(best)
```

Натуральные числа больше нуля, поэтому best = 0 подходит для старта и для случая без подходящих чисел.

Проверка. Ввод 3; 1; 2; 3 даёт 0.

Решения 7–8

7. Последняя цифра в основании 7

```
n = int(input())
s = 0
for i in range(n):
    x = int(input())
    if x % 7 == 1:
        s += x
print(s)
```

Последняя цифра числа в основании b равна x % b. Подходят 1, 8, 15 и 50: их сумма 74. Переводить всё число не нужно.

Проверка. Ввод 3; 7; 14; 21 даёт 0.

8. Самостоятельно: две последние цифры

```
n = int(input())
k = 0
for i in range(n):
    x = int(input())
    if 64 <= x <= 511 and x % 64 == 14:
        k += 1
print(k)
```

Три восьмеричные цифры: $8^2 \le x < 8^3$, то есть 64…511. Концовка 16_8 имеет значение $1 \cdot 8 + 6 = 14$. Последние две цифры задаёт остаток от деления на $8^2 = 64$. Подходят 78, 142, 462.

Проверка. Ввод 4; 14; 78; 462; 526 даёт 2.

Решения 9–11

9. Первые повторения через while

```
i = 1
while i <= 5:
    print(i)
    i += 1
```

Сначала проверяем условие, затем выводим число и увеличиваем i. Без увеличения цикл станет бесконечным.

Проверка. После последнего повторения i равно 6.

10. Сумма от 1 до n

```
n = int(input())
s = 0
i = 1
while i <= n:
    s += i
    i += 1
print(s)
```

s хранит накопленную сумму. Ноль - начальное значение суммы, i - очередное слагаемое.

Проверка. При n = 1 ответ 1; при n = 3 ответ 6.

11. Ввод до нуля: количество

```
k = 0
x = int(input())
while x != 0:
    k += 1
    x = int(input())
print(k)
```

Первое число читаем до цикла. Следующее читаем в конце каждого повторения, иначе x не изменится.

Проверка. При вводе только 0 ответ 0.

Решения 12–14

12. Ввод до нуля: сумма чётных

```
s = 0
x = int(input())
while x != 0:
    if x % 2 == 0:
        s += x
    x = int(input())
print(s)
```

Читаем новое число независимо от результата if. При подходящем числе прибавляем его значение к сумме.

Проверка. Ввод 1; 3; 0 даёт 0; ввод 0 тоже даёт 0.

13. Минимум с признаком конца

```
best = 100
x = int(input())
while x != 0:
    if 10 <= x <= 99 and x < best:
        best = x
    x = int(input())
if best == 100:
    print(0)
else:
    print(best)
```

100 больше любого двузначного числа. Если best не изменилось, подходящих чисел не было. Старт с 0 здесь ошибочен.

Проверка. Ввод 9; 100; 0 даёт 0; ввод 99; 0 даёт 99.

14. Самостоятельно: отбор и сумма

```
s = 0
x = int(input())
while x != 0:
    if 100 <= x <= 999 and x % 10 == 4:
        s += x
    x = int(input())
print(s)
```

Нужно проверить и диапазон, и последнюю цифру. Из примера подходят только 104 и 314.

Проверка. Ввод 4; 94; 1004; 0 даёт 0; ввод 994; 0 даёт 994.

Ответы и самопроверка

Ответы относятся к примерам из самостоятельных заданий. Программа должна работать на любых допустимых данных.

Часть 1. for

1. Первые повторения: 1; 2; 3; 4; 5

2. Чётные числа до n: 2; 4; 6; 8

3. Читаем ровно N чисел: 20

4. Сколько чисел кратно 3?: 3

5. Два условия одновременно: 3

6. Максимум среди подходящих: 40

7. Последняя цифра в основании 7: 74

8. Самостоятельно: две последние цифры: 3

Часть 2. while

9. Первые повторения через while: 1; 2; 3; 4; 5

10. Сумма от 1 до n: 15

11. Ввод до нуля: количество: 3

12. Ввод до нуля: сумма чётных: 10

13. Минимум с признаком конца: 18

14. Самостоятельно: отбор и сумма: 418

Последние 10 минут
Проверьте задания, в которых были ошибки. Для for: выполнено ровно N чтений, число N не обрабатывается как элемент. Для while: есть первое чтение до цикла и повторное внутри; признак конца не обрабатывается. Для отбора: проверьте отсутствие подходящих чисел и границы диапазона. Вывод итогового результата должен находиться после цикла.

Авторский учебный практикум, не официальный вариант экзамена. Ориентир по обработке последовательностей: ФИПИ, методические материалы для предметных комиссий по информатике, 2025, линия 16.
https://doc.fipi.ru/oge/dlya-predmetnyh-komissiy-subektov-rf/2025/mr_oge_informatika_2025.pdf