## Примеры решения задач с циклом в языке Python.

### Пример 4.1.

Вводится натуральное число (n > 0). Необходимо найти сумму и произведение цифр, из которых состоит это число. При этом если в числе встречается цифра 0, то ее не надо учитывать при нахождении произведения.

**Решение:**

Алгоритм решения задачи сводится к извлечению последней цифры числа путем нахождения остатка от деления на 10 и добавлению ее к произведению и сумме. Далее следует само число разделить нацело на 10, чтобы избавится от последней цифры. Эти действия следует выполнять до тех пор, пока число больше 0. На последней итерации цикла однозначное число делится нацело на 10, в результате чего получается 0 и цикл завершается.

Код на языке Python (первый способ):

```
n = int(input("Введите целое число: "))
mult = 1
summa = 0
while n > 0:
  if n % 10 != 0:
          mult = mult * (n%10)
  summa = summa + n % 10
  n = n // 10
print("Сумма цифр:", summa)
print("Произведение значащих цифр:", mult)
```

Выражение добавления цифры к сумме также можно поместить в ветку if.

В языке программирования Python решить подобную задачу может быть легче другим способом - оставить число в строковом представлении и далее в цикле for перебрать элементы последовательности.

Код на языке Python (второй способ):

```
n = input("Введите целое число: ")
mult = 1
summa = 0
for i in n:
  summa += int(i)
  if int(i)!= 0:
          mult *= int(i)
print("Сумма цифр:", summa)
print("Произведение значащих цифр:", mult)
```

Примечание. Выражение `summa += int(i)` - это то же самое, что `summa = summa + int(i)`. Та же сокращенная форма используется и для умножения.

Через строковое представление легко решать и более сложные задачи.

### Пример 4.2.

Например, надо посчитать сумму и произведение цифр дробного числа или всех цифр, встречающихся в строке.

В таком случае решение может быть таким:

```
n = input("Введите число или строку: ")
mult = 1
summa = 0
for i in n:
  if i in '1' <= i <= '9'
          summa += int(i)
          if int(i) != 0:
                  mult *= int(i)
print("Сумма цифр:", summa)
print("Произведение значащих цифр:", mult)
```

### Пример 4.3.

Вводится целое число. Вывести число, обратное по порядку составляющих его цифр введенному. Например, введено 3425, надо вывести 5243.

**Алгоритм:**

1. Найдем остаток от деления на 10 исходного числа. Получим последнюю его цифру.
2. Добавим эту цифру к новому числу.
3. Разделим нацело на 10 исходное число. Тем самым избавимся от последней цифры.
4. Снова найдем остаток от деления на 10 того, что осталось от первого числа. Запомним эту цифру.
5. Умножим на 10 второе число. Тем самым увеличим его разрядность до двух и сдвинем первую цифру в разряд десятков.
6. Добавим к полученному второму числу запомненную ранее цифру из первого числа.
7. Будем повторять действия п.3-6 пока исходное число не уменьшится до нуля, т.е. пока не избавимся от всех его разрядов.

Код на языке Python:

```
n1 = int(input("Введите целое число: "))
n2 = 0
while n1 > 0:
    digit = n1 % 10; # находим остаток - последнюю цифру числа
    n1 = n1 // 10;   # делим нацело - убираем последнюю цифру
    n2 = n2 * 10     # увеличиваем разрядность второго числа
    n2 = n2 + digit  # добавляем очередную цифру
print(' Число "наоборот"   :',n2)
```

Примеры выполнения кода:

```
Введите целое число: 32809
Число "наоборот"   : 90823
Введите целое число: 78290
Число "наоборот"   : 9287
```

### Пример 4.4.

Определить, сколько во введенном пользователем числе четных цифр, а сколько нечетных.

**Решение:**

Если число делится без остатка на 2 (т. е. остаток равен нулю), значит оно четное. Увеличиваем на 1 счетчик четных цифр (`even`). Иначе число нечетное, увеличиваем счетчик нечетных цифр (`odd`). В Python операцию нахождения остатка от деления выполняет знак %.

Чтобы избавиться от младшего (уже учтенного) разряда число следует разделить нацело на 10. Деление нацело обозначается так: `//`.

```
a = input("Введите целое число: ")
a = int(a)
even = 0
odd = 0
while a > 0:
    if a % 2 == 0:
        even += 1
    else:
        odd += 1
    a = a // 10
print("Even: %d, odd: %d" % (even, odd))
```

Примерный результат:

```
65439
Even: 2, odd: 3
```

### Пример 4.5.

Дан ряд чисел 1, $-\frac{1}{2}$, $\frac{1}{4}$, $-\frac{1}{8}$, …Требуется найти сумму стольких его элементов, сколько указал пользователь. Например, если n = 3, то надо сложить 1, $-\frac{1}{2}$, $\frac{1}{4}$, $-\frac{1}{8}$, что в результате даст 0.75.

При решении таких задач сначала необходимо определить, в чем заключается особенность предложенного ряда. В данном случае видим, что каждый последующий элемент в 2 раза меньше предыдущего по модулю, но взят с противоположным знаком. Далее следует найти арифметическое действие, которое из предыдущего элемента дает последующий. Здесь, например, надо предыдущий элемент делить на -2.

**Алгоритм:**

1. Присвоим переменной $a$ первое число ряда.
2. Создадим счетчик суммированных элементов ряда ($i$).
3. Создадим переменную для накопления суммы элементов ряда.
4. Пока счетчик не отсчитает заданного количества элементов ряда (пока $i < n$) будем выполнять нижеследующие действия.
5. К сумме будем добавлять значение текущего элемента.
6. Изменим значение текущего элемента на то, которое должно быть следующим (разделим на -2).
7. Увеличим значение счетчика на 1.

Код на языке Python:

```
n = input("Количество элементов ряда: ")
n = int(n)
a = 1
i = 0
summa = 0
while i < n:
    summa += a
    a = a/-2
    i += 1
print(summa)
```

Примеры выполнения кода:

```
Количество элементов ряда: 4
0.625
Количество элементов ряда: 10
0.666015625
```