Анализ цифр числа

Информатика. Язык программирования Python в курсе информатики. Все классы.

При помощи операции взятия остатка можно легко узнать последнюю цифру числа: это остаток при делении на основание системы счисления.  
Например, последняя цифра числа 15328 в 10-ной системе счисления равна 15328 mod 10.  
Если же необходимо достать n-ю цифру по старшинству (т. е. справа), то нужно вначале поделить число на на единицу меньшую степень основания системы счисления:  
15328  mod 10(3−1)=3, третья цифра справа.

Решим задачу нахождения суммы цифр данного числа n (при этом нам заранее неизвестно, сколько в этом числе разрядов). Будем брать остаток при делении на 10 до тех пор, пока это возможно, прибавлять его к некоторой переменной (искомой сумме), а затем целочисленно делить число n на 10 (обрезая последнюю цифру).

РЕШЕНИЕ НА PYTHON 3

n = int(input())  
sum = 0  
while n > 0:  
   sum += n % 10  
   n = n // 10

print(sum)

РЕШЕНИЕ НА PASCAL

var  
  n, sum: longint;  
begin  
  readln(n);  
  sum := 0;  
  while (n > 0) do   
  begin  
    sum += n mod 10;  
    n := n div 10;  
  end;  
  writeln(sum);  
end.

Еще одна похожая задача — перевернуть число n. То есть, если n=12345, то результатом работы программы будет 54321. Будем действовать схожим образом — отрезать цифру от данного числа, пока это можно сделать и прибавлять ее к некоторой переменной ans, но прибавлять не просто так, а предварительно умножая ans на 10, чтобы "подвинуть" разряды.

РЕШЕНИЕ НА PYTHON 3

n = int(input())  
ans = 0  
while n > 0:  
   ans = ans \* 10 + n % 10  
   n = n // 10  
print(sum)

РЕШЕНИЕ НА PASCAL

var   
 n, ans: longint;   
begin   
 readln(n);   
 ans := 0;   
 while (n > 0) do    
  begin   
   ans := ans \* 10;   
   ans += n mod 10;   
   n := n div 10;   
 end;   
 writeln(ans);  
  
end.



Сумма и произведение последовательности чисел

Информатика. Язык программирования Python в курсе информатики. Все классы.

СУММА ПОСЛЕДОВАТЕЛЬНОСТИ ЧИСЕЛ

Задача накопления суммы или произведения встречается в программировании довольно часто и звучит в частности так:

Дана последовательность чисел, завершающаяся числом 0. Найдите сумму всех этих чисел.

|  |  |
| --- | --- |
| Ввод | Вывод |
| 1  7  9  0 | 17 |

**РЕШЕНИЕ НА ЯЗЫКЕ PYTHON**

sum\_int = 0  
x = int(input())  
while x != 0:  
   sum\_int += x  
   x = int(input())  
print(sum\_int)

**РЕШЕНИЕ НА ЯЗЫКЕ PASCALABC.NET**

var summa, x: longint;  
begin  
   summa:= 0;  
   readln(x);  
   while x <> 0 do begin  
       summa += x;  
       readln(x);  
   end;  
writeln(summa);  
end.

При поиске суммы чисел начальное значение переменной, в которой происходит накопление результата, должно равняться нулю.

ПРОИЗВЕДЕНИЕ ПОСЛЕДОВАТЕЛЬНОСТИ ЧИСЕЛ

Дана последовательность чисел, завершающаяся числом 0. Найдите произведение всех этих чисел.

|  |  |
| --- | --- |
| Ввод | Вывод |
| 1  7  9  0 | 63 |

**Решение на языке Python**

proizvedenie = 1  
x = int(input())  
while x != 0:  
   proizvedenie \*= x  
   x = int(input())  
print(proizvedenie)

**Решение на языке PascalABC.NET**

var proizvedenie, x: longint;  
begin  
   proizvedenie:= 1;  
   readln(x);  
   while x <> 0 do begin  
       proizvedenie \*= x;  
       readln(x);  
   end;  
writeln(proizvedenie);  
end.

Очевидно, при поиске произведения начальное значение переменной, в которой происходит накопление результата, должно равняться 1.

Отметим также, что в этой задаче нельзя поменять местами строки proizvedenie \*= x; и readln(x); внутри цикла, поскольку иначе итоговый результат всегда будет равен 0!

ВАРИАНТ ПОИСКА СУММЫ С ИСПОЛЬЗОВАНИЕМ СПИСКА

На языке Python возможно решение при помощи создания списка и использования встроенной функции sum:

a = []  
x = int(input())  
while x != 0:  
   a.append(x)  
   x = int(input())  
print(sum(a))

Однако оно не лишено недостатка: сумма вычисляется не в один проход по данным, а в два. Следовательно потребуется память на сохранение значений всех чисел потока, а само вычисление будет происходить медленно из-за двойной работы.  
Однако, когда список уже есть, использование встроенной функции является оптимальным решением.

Фильтрация потока чисел

Информатика. Язык программирования Python в курсе информатики. Все классы.

Пусть на вход программы подаются числа, подлежащие обработке, причем из них нужно использовать только те, которые удовлетворяют некоторому условию.

Например, если требуется просуммировать только четные числа, это можно сделать так (0 — признак конца ввода):

sum\_even = 0  
x = int(input())  
while x != 0:  
   if x % 2 == 0:  
       sum\_even += x  
   x = int(input())  
print(sum\_even)

Или же, если требуется найти произведение только отрицательных чисел:

product = 1  
x = int(input())  
while x != 0:  
   if x < 0:  
       product \*= x  
       x = int(input())  
print(product)

Может потребоваться подсчитать количество определенных (например, четных) чисел:

counter = 0  
x = int(input())  
while x != 0:  
   if x % 2 == 0:  
   counter += 1  
   x = int(input())  
print(counter)

Например, это может понадобиться для того, чтобы найти среднее арифметическое только четных чисел:

sum\_even = 0  
counter = 0  
x = int(input())  
while x != 0:  
   if x % 2 == 0:  
       sum\_even += x  
       counter += 1  
       x = int(input())  
print(sum\_even / counter)



Поиск максимального и минимального числа в потоке

Информатика. Язык программирования Python в курсе информатики. Все классы.

Поиск максимального и минимального числа в последовательности происходит схожим образом с вычислением суммы, но переменная, в которой хранится временный результат — это максимум или минимум среди уже просмотренных значений.

Чему следует положить значение временного максимума/минимума в самом начале? Первому элементу последовательности, который нужно считать до начала циклической обработки.

maximum = int(input())  
x = int(input())  
while x != 0:  
   if x > maximum:  
       maximum = x  
   x = int(input())  
print(maximum)

Если же числа последовательности дополнительно фильтруются (например, нужно найти минимум только среди четных элементов), то в качестве минимума следует положить самое большое из возможных чисел в потоке, (а в качестве максимума — самое маленькое из возможных)

Если об этом информации нет, то поиск первого подходящего по критериям фильтрации числа должен быть произведен (с помощью цикла while) до цикла отбора минимального/максимального.

КОД НА ЯЗЫКЕ PASCAL

Readln(maximum);  
Readln(x);  
while x <> 0 do begin  
    if x > maximum then  
        maximum := x;  
    Readln(x);    
end;  
Writeln(maximum);



Среднее арифметическое и отклонение от среднего

Информатика. Язык программирования Python в курсе информатики. Все классы.

В физическом эксперименте после набора эмпирических данных требуется их обработка — усреднение и нахождение ошибки.

Программа поиска среднего арифметического — это однопроходный алгоритм, где числа обрабатываются «на ходу», а значит их не требуется запоминать.

Формула среднего арифметического:

M=a1+a2+…+ann=1n∑i=1nan

ПРОГРАММА:

(Число 0 — признак конца ввода)

sum\_int = 0  
counter = 0  
x = int(input())  
while x != 0:  
    sum\_int += x  
    counter += 1  
    x = int(input())  
  
print(sum\_int / counter)

Но для поиска среднеквадратического отклонения от среднего арифметического согласно формуле потребуется еще один проход:

D=(a1−M)2+(a2−M)2+…+(an−M)2n

Отдельный проход нужен не потому, что в конце понадобится извлечь корень, или потому, что в процессе суммирования члены последовательности нужно возводить в квадрат, а потому, что при расчете каждого из членов последовательности уже требуется знание среднего арифметического.

Согласиться на двухпроходный алгоритм — значит сохранять все значения в памяти, потребуется значительный объем ОЗУ, зависящий от длины последовательности усредняемых чисел…

Нам поможет смекалка и математика.

Раскроем квадрат в каждом слагаемом по формуле сокращенного умножения:

D2=1n∑i=1n(an−M)2=1n∑i=1n(an2−2anM+M2)

Поскольку сложение ассоциативно и коммутативно, сумму сумм трех слагаемых можно записать как три суммы:

D2=1n(∑i=1nan2−∑i=1n2anM+∑i=1nM2)

Пользуясь тем, что среднее арифметическое не зависит от i, можно вынести его за скобку во второй сумме, а третью сумму просто посчитать, т. к. это n одинаковых слагаемых:

D2=1n(∑i=1nan2−2M∑i=1nan+nM2)

Остается только увидеть во второй из оставшихся сумм формулу среднего арифметического, привести слагаемые, и в итоге получится следующее:

D2=1n∑i=1nan2−M2=1n∑i=1nan2−(1n∑i=1nan)2

Таким образом, для подсчета среднеквадратического отклонения достаточно параллельно считать две суммы: сумму чисел и сумму квадратов чисел, затем обе поделить на n, второе из двух возвести в квадрат и вычесть из первого. От разности достаточно взять корень квадратный, чтобы получить само D.

ОДНОПРОХОДНАЯ ПРОГРАММА:

summa = 0  
summa\_2 = 0  
kolichestvo = 0  
x = int(input())  
while x != 0:  
    summa += x  
    summa\_2 += x\*x  
    kolichestvo += 1  
    x = int(input())  
srednee = summa/kolichestvo  
srednee\_2 = summa\_2/kolichestvo  
otklonenie = (srednee\_2 - srednee\*\*2)\*\*0.5  
print('Среднее значение:', srednee, '+-', otklonenie)



Поиск второго по значению экстремума

Информатика. Язык программирования Python в курсе информатики. Все классы.

Допустим, что на вход программе подается последовательность натуральных чисел, заканчивающаяся нулем (сам 0 не является элементом последовательности).

Решим задачу нахождения второго по величине максимального элемента (не сохраняя в памяти все элементы последовательности).

Будем действовать так: считаем первое и второе число и запишем значение наибольшего из них во временную переменную max\_elem — в этой переменной будет храниться наибольший из уже рассмотренных элементов последовательности. Во временной переменной max\_elem2 будем хранить второй по значению максимум среди всех просмотренных.

Далее будем считывать числа до той поры, пока нам не попадется 0, при этом каждый раз определяя, может ли новый элемент быть максимальным или вторым по значению. Для этого сначала сравним его с max\_elem — если новый больше, то записываем значение max\_elem в max\_elem2, а значение нового элемента — в max\_elem. Если новый меньше или равен max\_elem, но больше max\_elem2, то он будет вторым максимумом и его значение сохраняем в max\_elem2. В ином случае этот элемент нас не интересует.

РЕШЕНИЕ НА ЯЗЫКЕ PASCAL

var

    y, x, max\_elem, max\_elem2: longint;

begin

    readln(y, x);

    if x > y then begin

        max\_elem := x;

        max\_elem2 := y;

    end else begin

        max\_elem2 := x;

        max\_elem := y;

  end;

  while x <> 0 do begin

      readln(x);

      if x > max\_elem then begin

          max\_elem2 := max\_elem;

          max\_elem := x;

      end

      else

      if x > max\_elem2 then

          max\_elem2 := x;

    end;

    writeln(max\_elem2);

end.

РЕШЕНИЕ НА PYTHON

my\_list = [(lambda x:(x\*73+51)%100)(i) for i in range(30)] #генерируем список из 30 элементов  
print(my\_list)  
max1 = my\_list.pop()  
max2 = my\_list.pop()  
if max2 > max1:  
    max1, max2 = max2, max1  
while len(my\_list) > 0:  
    num = my\_list.pop()  
    if num > max1:  
        max2 = max1  
        max1 = num  
    elif num > max2:  
        max2 = num  
print(max1, max2)

Количество равных максимумов

Информатика. Язык программирования Python в курсе информатики. Все классы.

Для подсчета количества равных максимумов можно использовать следующий подход:

my\_list = [(lambda x:(x\*87+12)%10)(i) for i in range(20)] #генерируем список из 20 элементов  
print(my\_list)  
max\_value = my\_list.pop()  
max\_number = 1  
while len(my\_list) > 0:  
    num = my\_list.pop()  
    if num > max\_value:  
        max\_value = num  
        max\_number = 1  
    elif num == max\_value:  
        max\_number += 1  
print('Максимум', max\_value, 'достигается', max\_number, 'раз.')