6 ЦИКЛИЧЕСКИЙ АЛГОРИТМ WHILE

6.1 Оператор цикла while

Для выполнения оператора **for** необходимо задать параметры, которые будут определять, сколько раз должен выполниться оператор (операторы) цикла. Альтернативой циклу с оператором **for** является цикл с неизвестным количеством повторений, в котором оператор (операторы) выполняется, пока логическое выражение не примет определенное значение.

Циклическая структура, в которой число повторений цикла заранее неизвестно, а определяется только в процессе выполнения алгоритма, называется **итеративной**.

Такие циклы нужно применять в тех задачах, где мы не можем знать точно, сколько раз будет повторен цикл. Например, пользователь должен вводить пароль, для того чтобы начать работу с какой-нибудь программой. Какое количество попыток он использует? Неизвестно.

Следовательно, для реализации подобных задач необходимо владеть соответствующими методами их решения. В языке **Python** для реализации конструкции цикла с неизвестным количеством повторений служит оператор **while** или **цикл с предпроверкой условия** (рисунок 57).

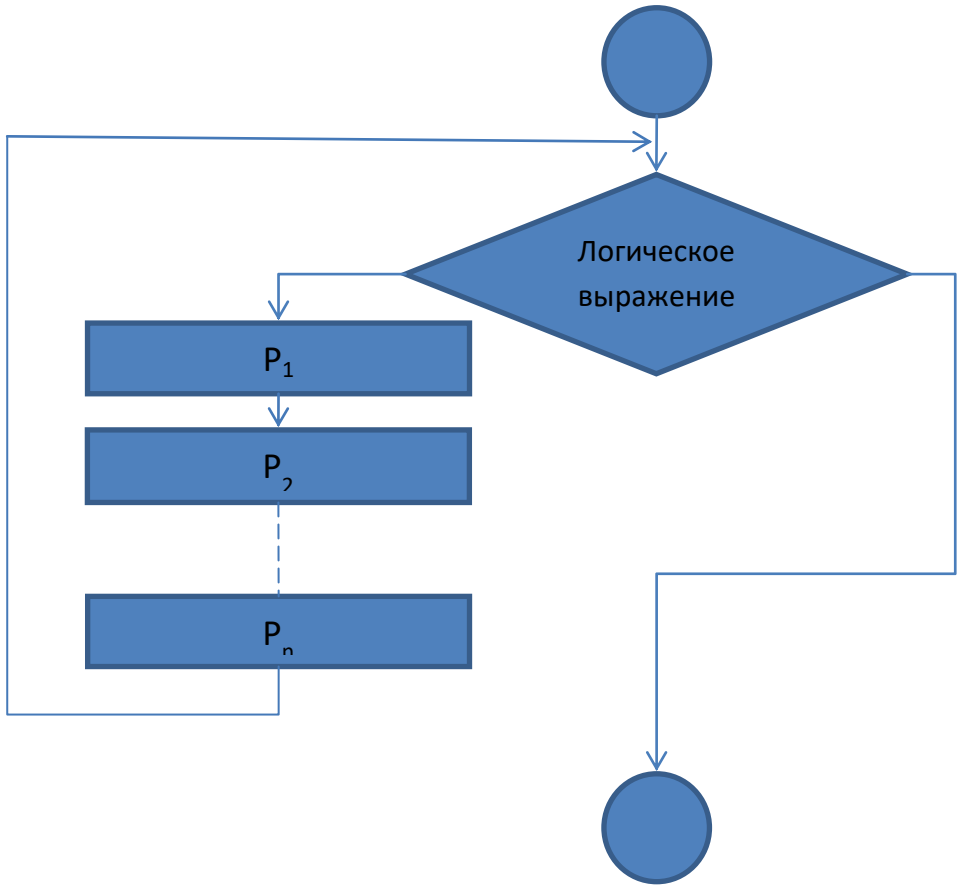

Рисунок 57 – Блок-схема алгоритма оператора **while**

Синтаксис оператора while следующий:

Инициализация начального значения
while логическое выражение:
P_1
P_2

P_n

где P_1, P_2, ... P_n - операторы; **while** (пока, до тех пор) - служебное слово языка **Python**.

Если логическое выражение после служебного слова **while** имеет значение **True** (Истина), то выполняются операторы P_1, P_2, ...P_n, после чего проверка логического выражения повторяется. Если логическое выражение имеет значение **False** (Ложь), то происходит выход из цикла. Если условие в заголовке цикла не является истинным с самого начала, цикл **while** не выполняется. Поскольку параметр цикла, который по умолчанию был в конструкции цикла с оператором **for**, в цикле **while** отсутствует, то возникает необходимость в создании переменной, которая бы играла соответствующую роль. Кроме того, при программировании необходимо еще и предусмотреть ее инициализацию до выполнения цикла, а в самом цикле увеличение этой переменной на определенный шаг.

Например, в листинге приведенном ниже цикл с оператором **while** обрабатывает данные до тех пор, пока не будет введено слово «Студент».

```
name=""
while name !='Студент':
    name=input("Введите любое слово для печати или слово Студент для выхода: ")
    if name!='Студент':
        print("Ответ = ", name)
```

Задача 6.1. Найдите сумму целых чисел от 1 до 50, используя оператор цикла **while**.

Решение. Ранее мы решали такую же задачу, но с оператором **for**. Прежде всего в данном случае ннужно, чтобы какая-нибудь переменная менялась в цикле от 1 до 50. А так как такой величины, как параметр цикла, нет в конструкции **while**, то в этом примере такую роль будет играть переменная **k**. Далее задаем в качестве условия выхода из цикла **k!= 50** и применяем в цикле оператор **sum=sum+k.** Таким образом, мы просуммируем все пятьдесят слагаемых и получим в ответе число 1275. Блок-схема алгоритма решения задачи представлена на рисунке 58.

В листинге приведен код программы, отвечающий за решение задачи:

```
k=0
sum=0
while k!=50:
```

```
    k=k+1
    sum=sum+k
print("Сумма чисел от 1 до 50 sum = ", sum)
```

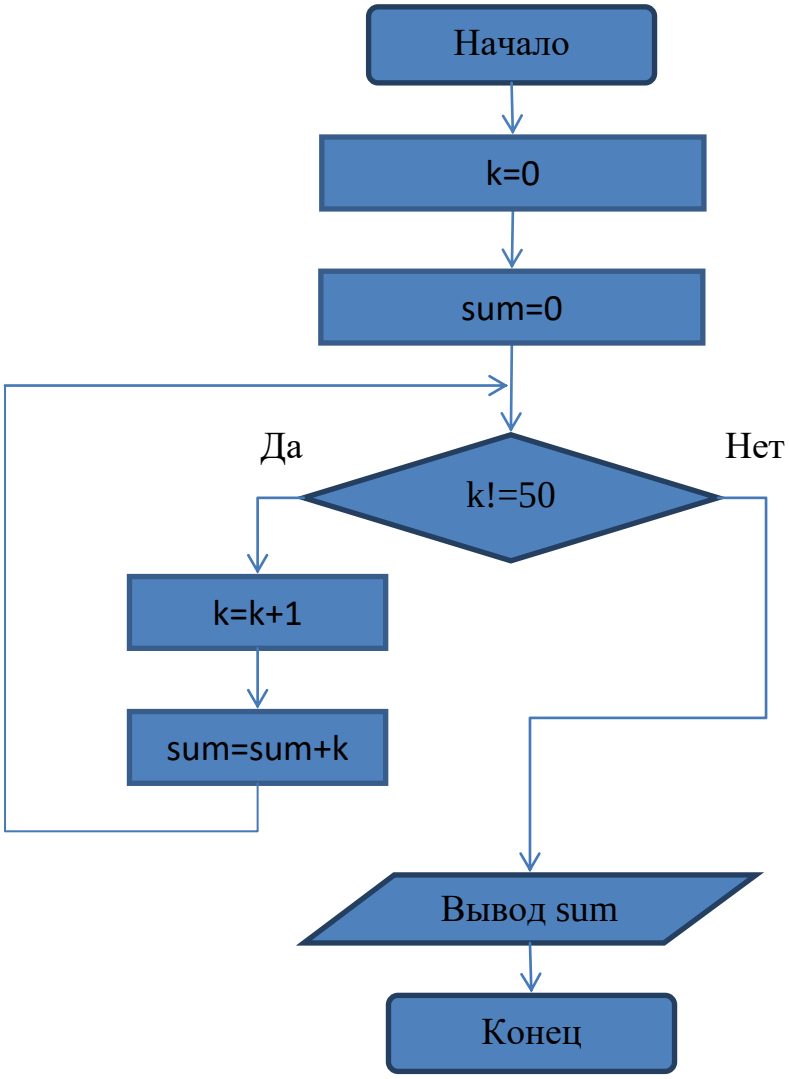

Рисунок 58 – Блок-схема алгоритма решения задачи 6.1

При вычислении значений функций или каких-либо числовых последовательностей (например, арифметической прогрессии), их часто записывают в виде специальных сумм, называемых **рядами**. Многие числа, функции, алгоритмы численных методов могут быть записаны с помощью рядов или итерационных алгоритмов, которые позволяют вычислять их приближенные значения с заданной точностью. При программировании таких задач используются итерационные методы для организации цикла, в котором производится вычисление некоторой **рекуррентной формулы**.

Рекуррентная формула - это такая формула, которая сводит вычисления **n**-го члена последовательности к вычислению нескольких предыдущих членов (или, часто, одного предыдущего члена этой последовательности – **n-1**). В общем случае такая формула имеет вид:

$$S_n = S_{n-1} + U_n,$$

где S_n сумма первых **n** слагаемых ряда, которая образуется из суммы, полученной на предыдущем шаге S_{n-1}; U_n- слагаемое, полученное на текущем шаге.

6.2 Пояснение примеров заданий на применение циклов while

Рассмотрим ряд заданий, в которых будут использоваться итеративные циклы и рекуррентные соотношения.

Задание 6.1. Среди чисел $1, 1+\frac{1}{2}, 1+\frac{1}{2}+\frac{1}{3}, \cdots$ найдите первое число, большее вводимого значения переменной **a**.

Решение. Алгоритм решения данного задания относится к алгоритмам вычисления членов бесконечных последовательностей. Очередной член бесконечной последовательности обозначим как **b**. Его номер, который совпадает со значением знаменателя дроби, добавляемой к предыдущему члену для получения значения очередного члена последовательности, обозначим как **n**. Тогда итерационная формула вычисления очередного члена последовательности примет вид:

$$b_n = b_{n-1} + \frac{1}{n}.$$

Блок-схема алгоритма решения задания представлена на рисунке 59.

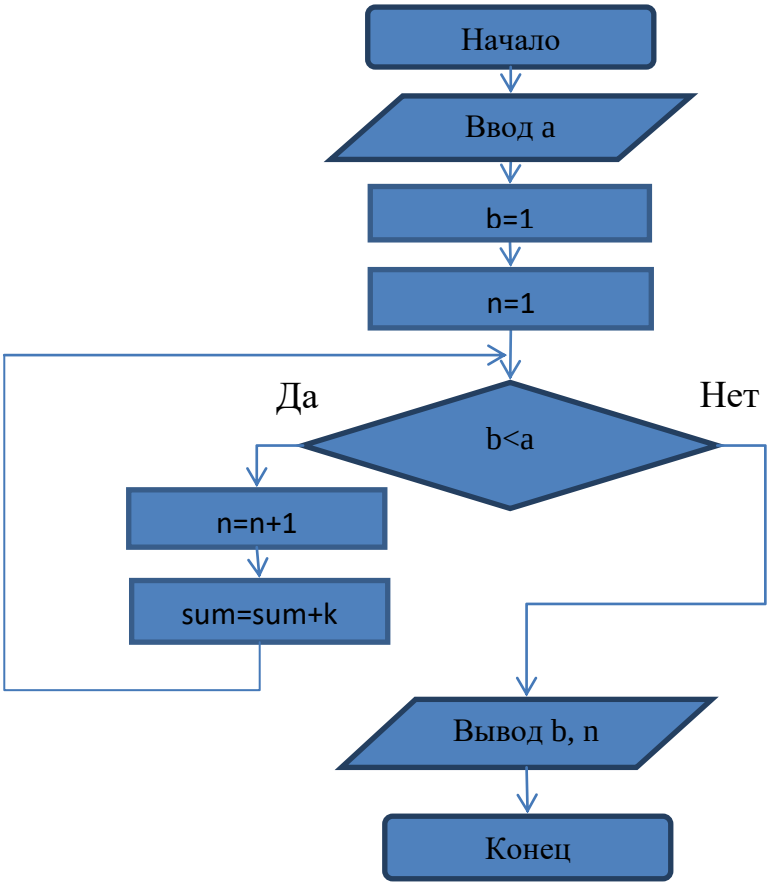

Рисунок 59 – Блок-схема алгоритма решения задания 6.1

В листинге приведен код программы, отвечающий за решение задания:

```
a=float(input("Введите число a = "))
b=1
n=1
print("\n ", n, " ", b)
while b<a:
   n+=1
   b=b+1/n
   print(" ", n, " ", b)
print("Первое число, превышающее значение a:", '{0:4f}'.format(b), "Количество итераций:", n)
```

На рисунке 60 показан результат работы программы при введенном значении **a**, равном **3.95**.

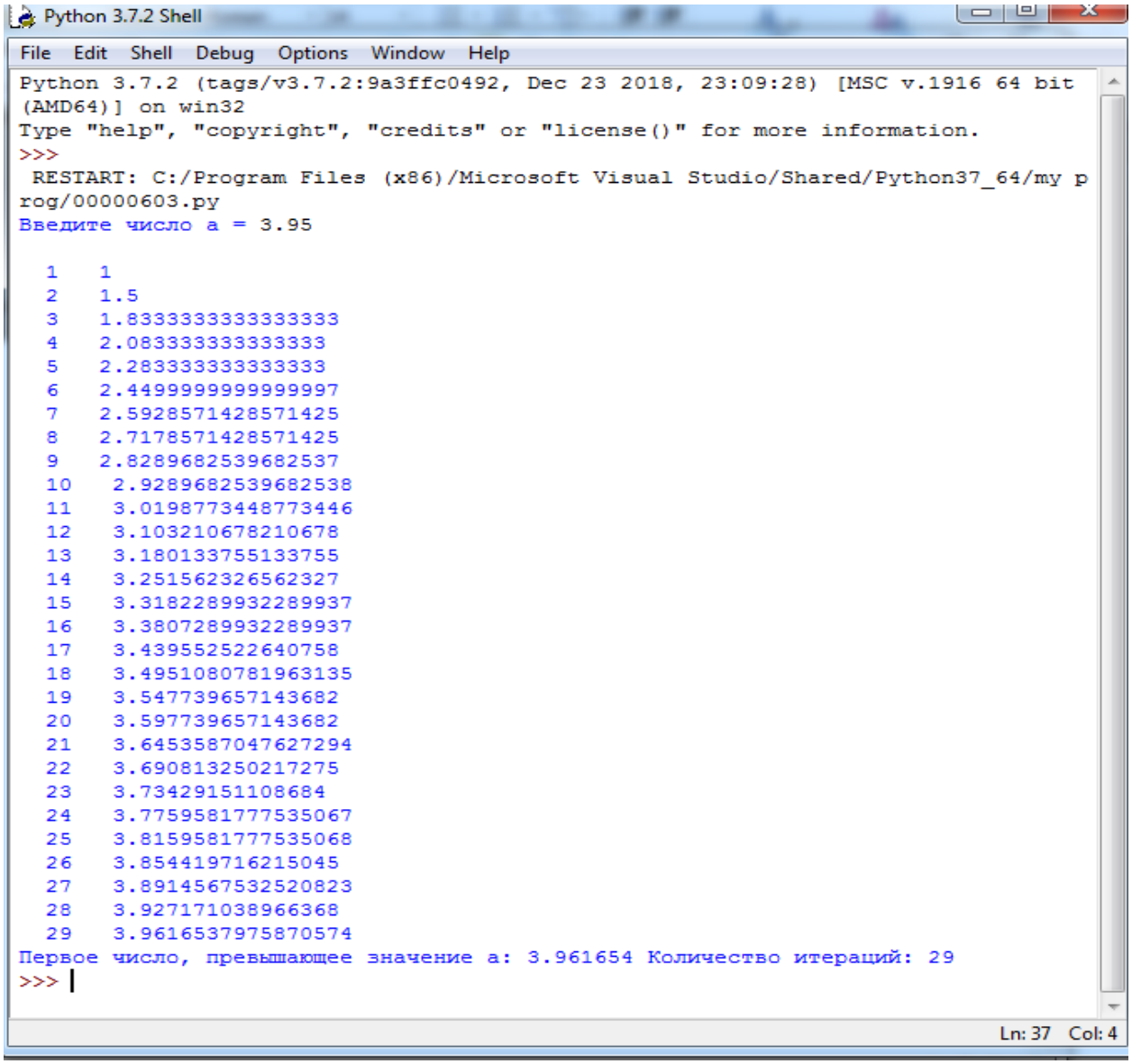

Рисунок 60 – Результат работы программы для **a=3.95**

Задание 6.2. Вычислите с точностью $\varepsilon = 10^{-4}$ корень уравнения $\boldsymbol{e^x + x = 0}$, воспользовавшись итерационной формулой $x_{i+1} = -e^{x_i}$, где $i = 0, 1, 2, \cdots$; $x_0 = 0$. Закончите итеративный процесс, как только $|x_{i+1} - x_i|$ станет меньше ε.

Решение. Для решения данного задания необходимо из очередного значения вычисленного корня $\boldsymbol{x_{i+1}}$вычитать предыдущее значение корня $\boldsymbol{x_i}$. Для этого при каждом повторении цикла перед вычислением очередного корня $\boldsymbol{x}$ сохраняем в переменной **a** текущее значение $\boldsymbol{x}$ (оно становится предыдущим). Цикл заканчивается, когда разность между $\boldsymbol{x}$ – ми (т. е. $\boldsymbol{x_{i+1}}$ и **a** (т. е. $\boldsymbol{x_i}$)) станет меньше ($\varepsilon = 10^{-4}$). В программе обозначим ε как **e1**.

Блок-схема алгоритма решения задания представлена на рисунке 61.

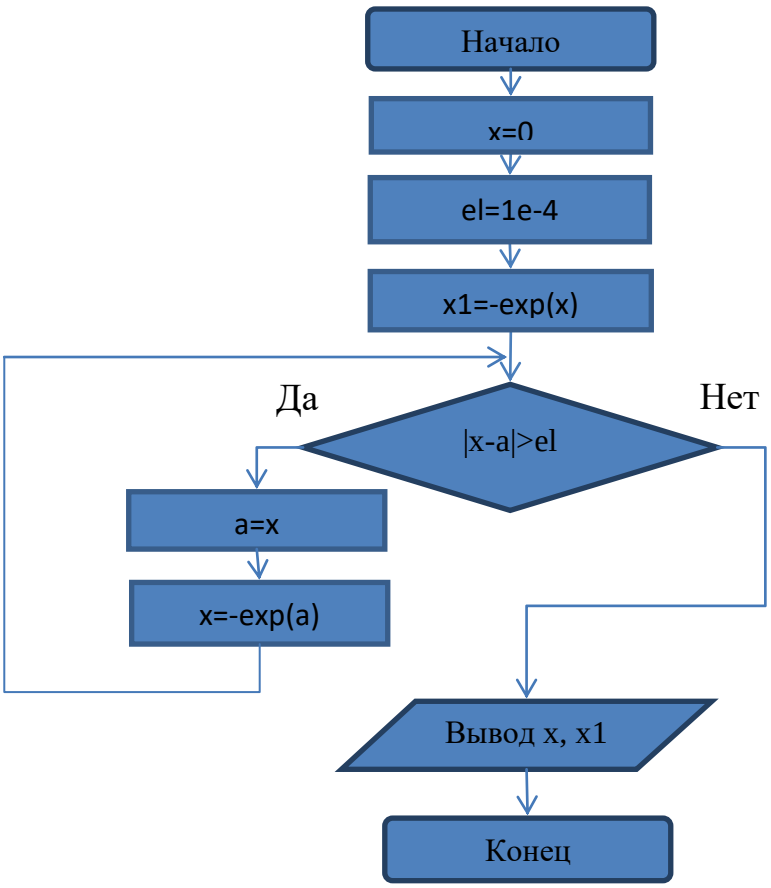

Рисунок 61 – Блок-схема алгоритма решения задания 6.2

Программный код решаемого задания представлен в листинге:

```
from math import *
a=1 #Инициализация значения a
x=0
x1=-exp(x)
el=0.0001
while abs(x-a)>el:
    a=x
```

```
    x=-exp(a)
print("Корень уравнения, вычисленный с заданной точностью = ", '{0:.14f}'.format(x))
print("Корень уравнения, вычисленный через математическую функцию модуля Math = ", '{0:.14f}'.format(x1))
```

Результат работы программы представлен на рисунке 62. Из него видно, что значения корня уравнения, вычисленное через математическую функцию модуля Math, имеет более грубое значение.

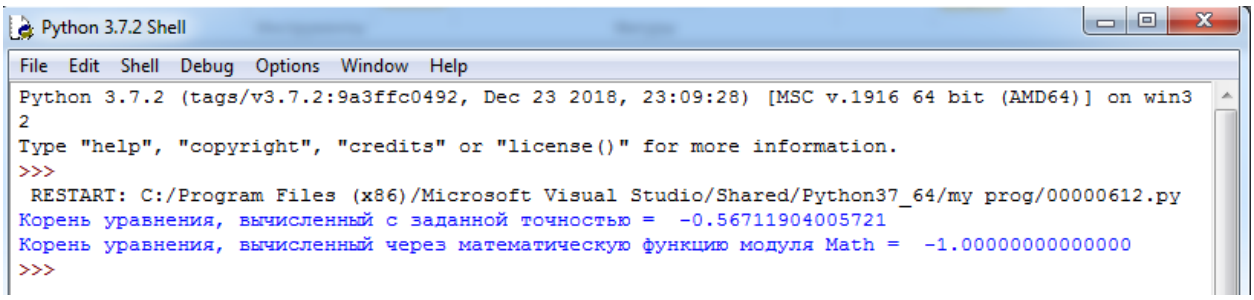

Рисунок 62 – Результат работы программы задания 6.2

Задание 6.3. Напишите программу, вычисляющую приближенное значение функции e^x с заданной точностью ε.

Решение. Функцию e^x можно представить бесконечным степенным рядом:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \cdots$$

Для вычисления значения функции **exp(x)** с точностью ε необходимо просуммировать все члены ряда, которые по модулю превышают заданную точность, то есть ε. Следовательно, необходимо организовать циклический процесс для вычисления суммы и повторять его пока выполняется условие

$$\left|\frac{x^n}{n!}\right| > \varepsilon.$$

Рассматривая приведенную выше формулу можно заметить, что соседние члены ряда (обозначим их как C_nи C_{n+1}) связаны между собой соотношением

$$C_{n+1} = C_n \cdot \frac{x}{n+1}, \qquad \text{где } n = 0, 1, 2, \cdots \text{ причем } C_0 = 1.$$

Используя это соотношение, можно последовательно вычислять один член ряда за другим. При этом не возникает необходимости вводить операции вычисления факториала и возведения в степень. Рассмотренное свойство степенных рядов позволяет организовывать чрезвычайно простые и очень эффективные (в смысле оптимизации скорости вычислений) алгоритмы вычисления их частичных сумм.

Еще раз отметим, что формулы, позволяющие вычислять последующие значения чего-либо на основе предыдущего значения, называются **рекуррентными соотношениями**. Блок-схема алгоритма решения задания представлена на рисунке 63.

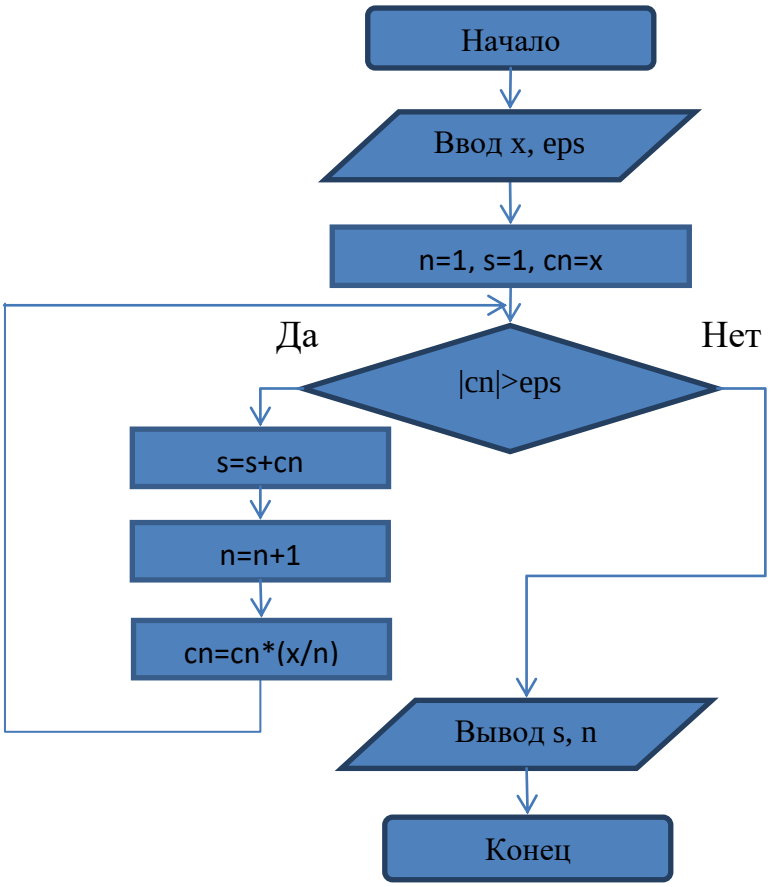

Рисунок 63 – Блок-схема алгоритма решения задания 6.3

В листинге приведен код программы, отвечающий за решение задания:

```
from math import *
x=float(input("Введите значение x = "))
eps=float(input("Введите точность вычислений eps = "))
s=1
n=1
cn=x
while abs(cn)>eps:
    s=s+cn
    n=n+1
    cn=cn*(x/n)
y=exp(x)
print("Значение функции, вычисленное с заданной точностью = ", s)
print("Значение функции, вычисленное через математическую функцию модуля Math = ", y)
```

Из результатов, представленных на рисунке 64, видно, что значение функции вычисленное через сумму ряда дает достаточную точность. А при значительно меньших значениях ε даст результат более точный, чем значения вычисленные посредством оператора **exp(x)**.

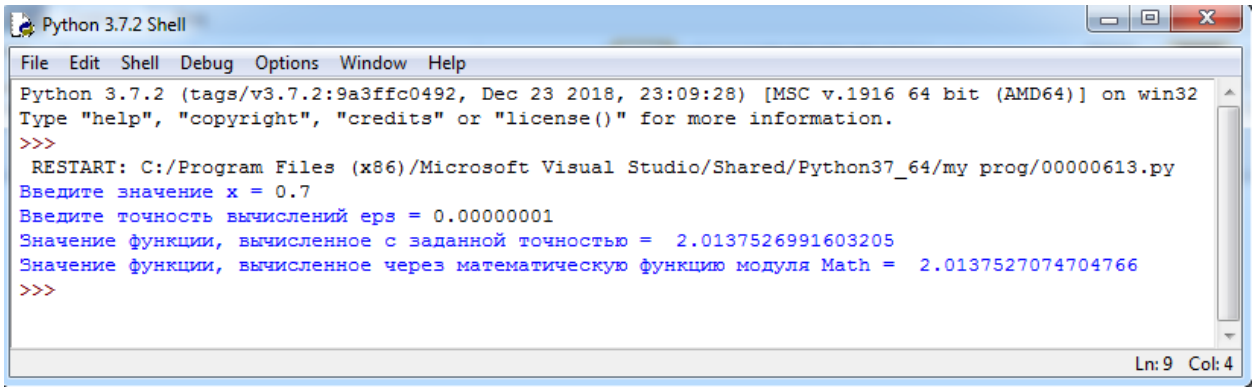

Рисунок 64 – Результат работы программы задания 6.3

Задание 6.4. Вычислите сумму членов знакопеременной убывающей последовательности с заданной точностью ε.

$$\frac{(x-1)}{1!}-\frac{(x-1)^2}{2!}+\frac{(x-1)^3}{3!}-\cdots+(-1)^n\frac{(x-1)^{n+1}}{(n+1)!}+\cdots$$

Решение. Вычисление с заданной точностью ε означает, что суммирование членов ряда надо продолжать до тех пор, пока очередной вычисленный член ряда не станет меньше по абсолютной величине числа ε.

Отметим, что во многих задачах непосредственный подсчет очередного члена связан с вычислительными трудностями. В этом случае целесообразно использовать **рекуррентную формулу**, которая позволяет вычислить значение переменной на следующем шаге, используя ее значение на текущем шаге $a_{n+1}=a_n\cdot q$- Выражение для $\boldsymbol{q}$ можно получить, разделив a_{n+1} член на a_n член.

Приведем вывод рекуррентной формулы для заданного в задании ряда. Формула a_nчлена

$$a_n=(-1)^n\frac{(x-1)^{n+1}}{(n+1)!},$$

тогда формула a_{n+1} члена

$$a_{n+1}=(-1)^{n+1}\frac{(x-1)^{n+1+1}}{(n+1+1)!}=(-1)^{n+1}\frac{(x-1)^{n+2}}{(n+2)!}.$$

Разделив a_{n+1} член на a_n получим выражение для **q**

$$q=\frac{a_{n+1}}{a_n}=\frac{(-1)^{n+1}\frac{(x-1)^{n+2}}{(n+2)!}}{(-1)^n\frac{(x-1)^{n+1}}{(n+1)!}}=\frac{(x-1)^{n+2}\cdot(-1)^{n+1}\cdot(n+1)!}{(-1)^n\cdot(x-1)^{n+1}\cdot(n+2)!}=-\frac{(x-1)}{n+2}.$$

Таким образом, рекуррентная формула для данного ряда примет вид

$$a_{n+1}=-a_n\frac{(x-1)}{n+2}.$$

Выбор начального значения номера члена ряда **n** для нашего случая будет **n=0**, так как при подстановке этого значения в формулу **n**-го члена ряда

$$a_n = (-1)^n \frac{(x-1)^{n+1}}{(n+1)!}$$

мы получим значение первого члена, равного **x-1** или **a=x-1**.

Результат работы программы представлен на рисунке 65. В листинге приведен код программы, отвечающий за решение задания:

```
from math import *
x=float(input("Введите значение x = "))
el=float(input("Введите точность вычислений el = "))
n=0
a=x-1
s=0
print("\n ", n, " ", a)
while abs(a)>el:
    print("\n ", n, " ", a)
    s=s+a
    a=-a*(x-1)/(n+2)
    n=n+1
print("\n Значение итерации не учтенное в сумме ряда ", n, " ", a)
print("\n Сумма ряда, вычисленное с заданной точностью = ", s)
```

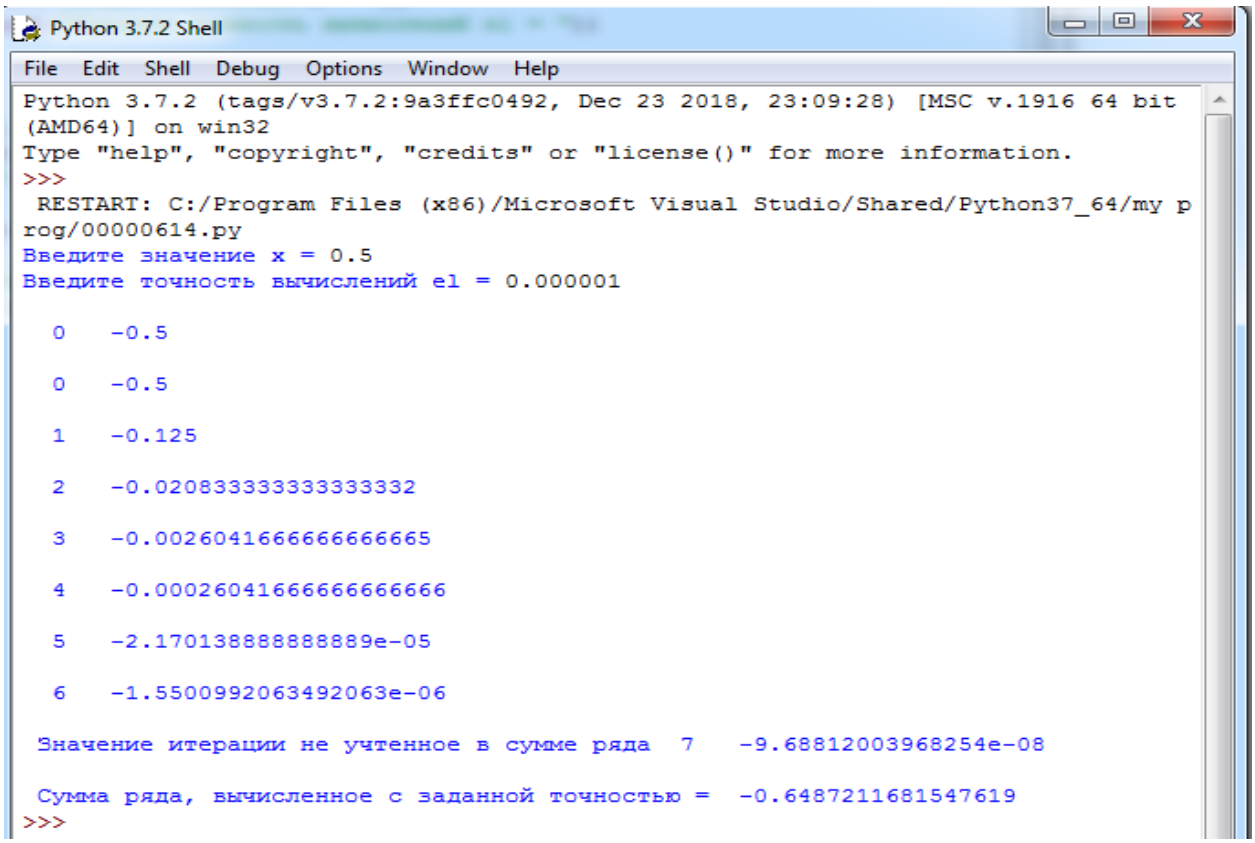

Рисунок 65 – Результат работы программы задания 6.4

Задание 6.5. Вычислите с точностью $\varepsilon = 10^{-5}$ корень уравнения $\boldsymbol{f(x) = x^3 - 2x^2 + x - 3 = 0}$, воспользовавшись итерационной формулой

$$x_{i+1} = x_i - \frac{f(x)}{f'(x)}, \text{ где } i = 0, 1, 2, \cdots;\ x_0 = 2{,}2.$$

Закончите итеративный процесс, как только $|x_{i+1} - x_i|$ станет меньше ε.

Решение. Решив задание, проверим правильность решения подстановкой найденного корня в уравнение.

Для начала вычислим производную $f'(x)$:

$$f'(x) = F(x) = 3x^2 - 4x + 1.$$

Обозначим за **x** - текущее приближение к корню, **a** - предыдущее приближение, **f** - значение функции $f(x)$ для предыдущего значения, **p** - значение производной $f'(x)$ для предыдущего значения, **i** - номер итерации, совпадающий с номером текущего приближения к корню уравнения, **y** - значение функции $f(x)$ для найденного с заданной точностью корня уравнения.

Будем считать, что заданная точность ε обеспечена, если модуль разности между текущим и предыдущим значениями корня меньше точности ε, т. е. для нашего случая $|x - a| < \varepsilon$. В листинге приведен код программы, отвечающий за решение задания.

```
from math import *
x=2.2
el=0.00001
i=0
a=1
print("\n ", i, " ", x)
while abs(x-a)>el:
    a=x
    f=pow(x, 3)-2*a*a+a-3
    p=3*a*a-4*a+1
    x=a-f/p
    i=i+1
    print("\n ", i, " ", x)
y=pow(x, 3)-2*x*x+x-3
print("\n Искомый корень x =  ", '{0:.14f}'.format(x))
print("\n Значение выражения при подстановке найденного корня в уравнение = ", '{0:.14f}'.format(y))
```

Таким образом, из результатов, представленных на рисунке 66, видно, что значение **y** найденное как подстановка в уравнение найденного корня, оказывается достаточно точным.

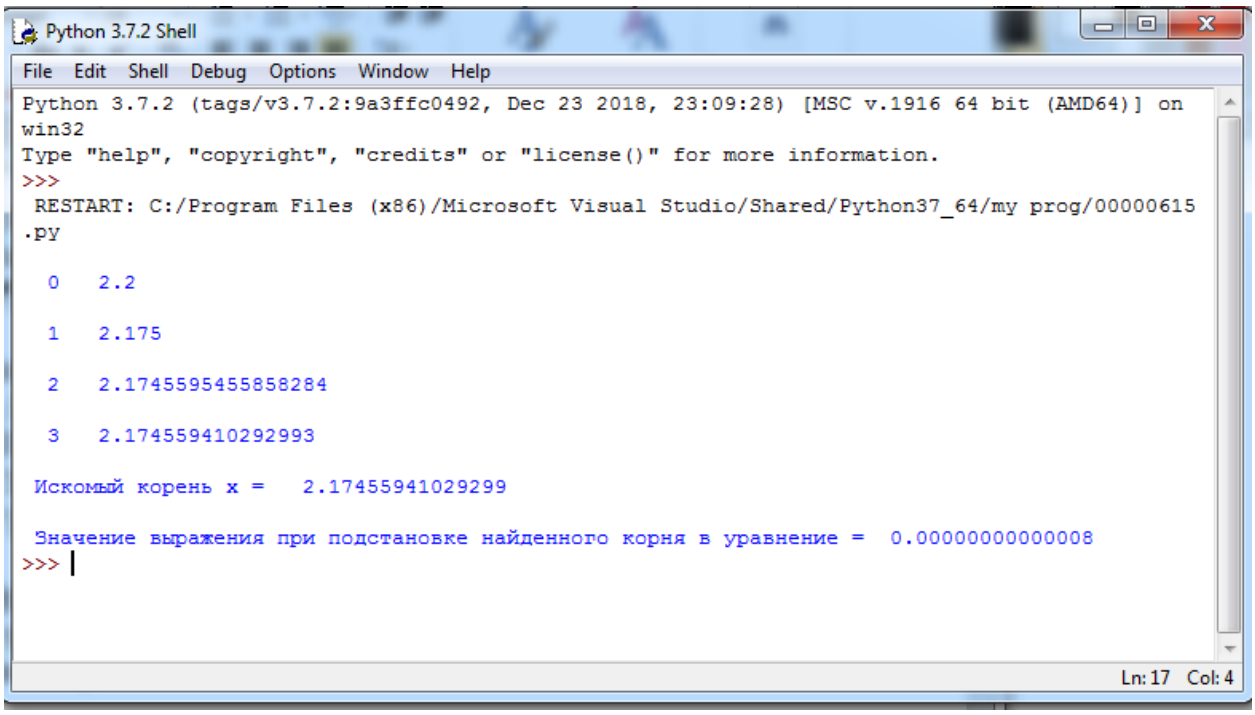

Рисунок 66 – Результат работы программы задания 6.5

6.3 Вопросы в упражнениях с циклом while

Вопрос 1. Определите, какое значение находится в ячейке y после выполнения группы операторов?

```
k=0
y=3
while k<y:
    y=y+1
    k=k+3
print("y = ", y)
```

Ответ. В цикле с оператором **while** выход из цикла происходит тогда, когда логическое выражение имеет значение **False** (Ложь). Подставив исходные значения, получаем, что логическое выражение истинно. При выполнении операторов цикла в ячейку **y** заносится число четыре, а в ячейку **k** заносится число три. Вновь проверяется логическое выражение (**3<4**). Его результат: **Истина**. При втором прохождении цикла в ячейку y заносится число **5** (**y=y+1**), а в ячейку **k** -число **6** (**k = k +3**). Проверив логическое выражение, убеждаемся в том, что оно **ложно** (**6<5**), следовательно, происходит выход из цикла. В ячейке y хранится число **5**.

Вопрос 2. Определите, какое значение находится в ячейке **s** после выполнения группы операторов?

```
s=7
```

```
i=1
while i*i==2:
   s=s+1/i
   i+=i
print("s = ", s)
```

Ответ. В данном примере надо быть внимательным и заметить, что логическое выражение всегда будет иметь значение **False** (Ложь), следовательно, тело цикла выполняться не будет. Значение ячейки **s** останется неизменным и будет равно **7**.

Вопрос 3. Каким будет значение переменной z после выполнения группы операторов?

```
a=1
z=4
while a<=3:
   a+=a
   z=a+1
print("z = ", z)
```

Ответ. Для того чтобы логическое выражение стало ложным и произошел выход из цикла, необходимо, чтобы в ячейке a находилось число четыре или большее. При этом, при первом прохождении цикла в ячейке **a** будет значение, равное **2**, а в ячейке **z** будет находиться число **3**. Таким образом, цикл выполнится еще два раза. Оператор **z=a+1** увеличит значение ячейки **z** на **два**. Окончательный ответ: **z = 5**.

Вопрос 4. Определите, какое значение находится в ячейке **a** после выполнения группы операторов?

```
a=2
d=1
while (a+d)<=7:
   a=a+1
   d=d+1
a=a*d
print("a = ", a)
```

Ответ. При первом вхождении в цикл в ячейке **(a+d)** будет находиться значение **три.** После выполнения оператора **a=a+1** значение **a** будет равное **3**, а в ячейке **d** - значение **2** (после выполнения оператора **d=d+l**). **Сумма** значений этих **двух ячеек**, проверяемая в логическом выражении, будет **равна пяти**. Условие **5 < 7** истинно, следовательно, цикл выполнится еще раз. Увеличившись на единицу, значения ячеек **a** и **d** примут, соответственно,

значения **четыре** и **три**. Их **сумма** будет **равна семи**, значит, **цикл выполнится** третий раз. Увеличившись на единицу, значение ячейки **a** будет равно **пяти**, а значение ячейки **d** - равно **четырем**. Логическое выражение: **5+4 <= 7** оказывается **ложным**, происходит **выход из цикла**, и **выполняется** оператор **a=a*d**. Умножив пять на четыре, получим число **20**, которое и будет ответом в данном упражнении.

Вопрос 5. Определите, какое значение будет в ячейке **n** после выполнения группы операторов?

```
n=2
x=1
while x<=4:
   x=x+1
   n=n+x
print("n = ", n)
```

Ответ. Выход из цикла с оператором **while** происходит тогда, когда логическое выражение принимает значение **False** (Ложь). В данном примере выход из цикла произойдет, когда в ячейке **x** будет находиться число, большее по величине, чем число четыре. Видно, что в цикле ячейка **x** увеличивает свое значение на единицу, т. е. выход из цикла произойдет, когда в ячейке **x** будет число **пять**. Таким образом, в цикле нужно подсчитывать значение ячейки **n**, складывая его со значением переменной **x**. На момент выхода из цикла в ячейке **n** находится число **16**.

Вопрос 6. Каким будет значение переменной **a** после выполнения группы операторов?

```
a=1
z=1
while a<=3:
   a=a+1
a=a+z
a=a+10
print("a = ", a)
```

Ответ. В цикле с оператором **while** выполняется один оператор **a=a+1**. Это можно понять по отступу, обозначающему выполнение одного оператора в цикле. **Выход** из цикла произойдет тогда, когда в ячейке **a** будет число **четыре** (так как **a** увеличивается на единицу за каждый шаг цикла). Выполнив после выхода из цикла оператор **a=a+z,** получаем, что в ячейке **a** находится число **пять.** Далее выполняется оператор **a=a+10**. Значение ячейки **a** равно **15**.

Вопрос 7. Определите, какое значение находится в ячейке **s** после выполнения группы операторов?

```
a=20
d=5
while (a-d)>=10:
   a=a-1
   d=d+1
s=a+d
print("s = ", s)
```

Ответ. Проверив логическое выражение, убеждаемся в том, что оно истинно. Далее уменьшается значение ячейки **a** на единицу, значение ячейки **d** увеличивается на единицу. Каждый раз после этой процедуры проверяется разность значений ячеек **(a-d)>=10**. Выход из цикла происходит тогда, когда в ячейке **a** будет находиться число **17**, а в ячейке **d** - число **8**. Выполнив оператор **s=a+d**, получим, что значение ячейки **s = 25**.

Вопрос 8. Каким будет значение переменной **a** после выполнения группы операторов?

```
a=1
z=1
while z<=4:
   z+=z
   z=a+1
   a=z
print("a = ", a)
```

Ответ. В данном примере в теле цикла выполняются три оператора. Можно установить следующую закономерность: значения ячеек **z** и **a** при выполнении операторов будут **равны**, **так как** то значение, которое находится в ячейке **z, переходит** в ячейку **a** после выполнения оператора **a=z. Выход** из цикла произойдет тогда, когда в ячейке **z** будет находиться число **пять**, следовательно, и в ячейке **a** тоже будет храниться число **пять**, что и является ответом в данном упражнении.

Вопрос 9. Определите, какое значение находится в ячейке **p** после выполнения группы операторов?

```
p=1
i=2
while p<9:
   p=p*i
i=i+1
```

```
p=p*i
print("p = ", p)
```

Ответ. В цикле выполняется оператор **p=p*i**. Остальные операторы выполняются после завершения цикла. Значение ячейки **i** остается неизменным до выхода из цикла и будет равно **двум**, **меняется** только значение ячейки **p** после выполнения оператора **p=p*i**. Таким образом, необходимо умножать на два значение, которое находится в ячейке **p**, и проверять условие выхода из цикла. В ячейку **p** последовательно будут заноситься значения **2, 4, 8, 16**. Из этого перечисления видно, что, **когда** в ячейке **p** находится число **16**, происходит **выход из цикла**. Выполнив оператор **i=i+l**, получаем, что **i** равно **трем**. Выполнив оператор **p=p*i**, получим, что в ячейке **p** хранится число **48**.

6.4 Примеры решения задач

Задача 6.4.1. Вводится последовательность вещественных чисел. Известно, что последний элемент последовательности равен пяти. Найдите количество положительных чисел и минимальное из них.

Решение. Блок-схема алгоритма решения задачи представлена на рисунке 67. Выход из цикла с оператором **while** осуществляется в соответствии с алгоритмом, представленным на рисунке 67 тогда, когда логическое выражение принимает значение False. Следовательно, до тех пор, пока пользователь вводит значения, не равные числу **5**, в цикле будут выполняться соответствующие операторы, необходимые для поиска положительных чисел и минимального из них. В листинге приведен код программы, отвечающий за решение задачи:

```
kolpol=0
min=32767
chislo=0
while chislo!=5: # Пока число, введенное в цикле, не равно числу 5, выполняются операторы цикла
    chislo=float(input("Введите очредное число = "))
    if chislo>0: # Проверка на положительность очередного введенного числа
        kolpol=kolpol+1
        if chislo<min: # Реализация алгоритма поиска минимального положительного элемента
            min=chislo
if kolpol==1:
    print("\n Вы ввели только одно положительное число: 5, предназначенное для выхода из программы")
else:
```

```
print("\n Количество положительных чисел Kolpol = ", kolpol)
print("\n Минимальное из них min = ", min)
```

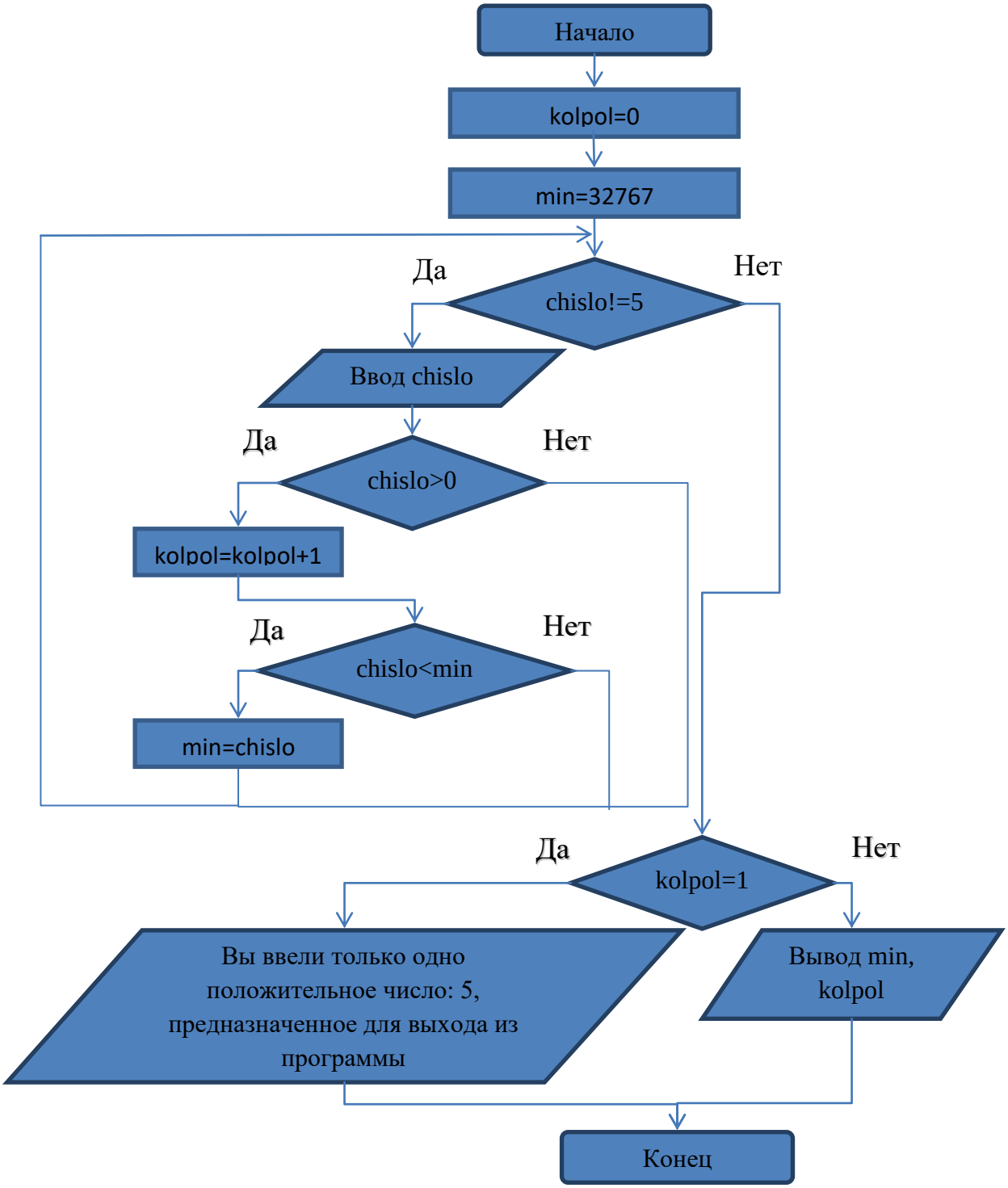

Рисунок 67 – Блок-схема алгоритма решения задачи 6.4.1

Задача 6.4.2. Вводится последовательность вещественных чисел. Известно, что последний элемент последовательности равен пяти. Определите, каких среди них больше отрицательных или положительных.

Решение. Блок-схема алгоритма решения задачи представлена на рисунке 68. Алгоритм решения очень похож на алгоритм решения задачи 6.4.1.

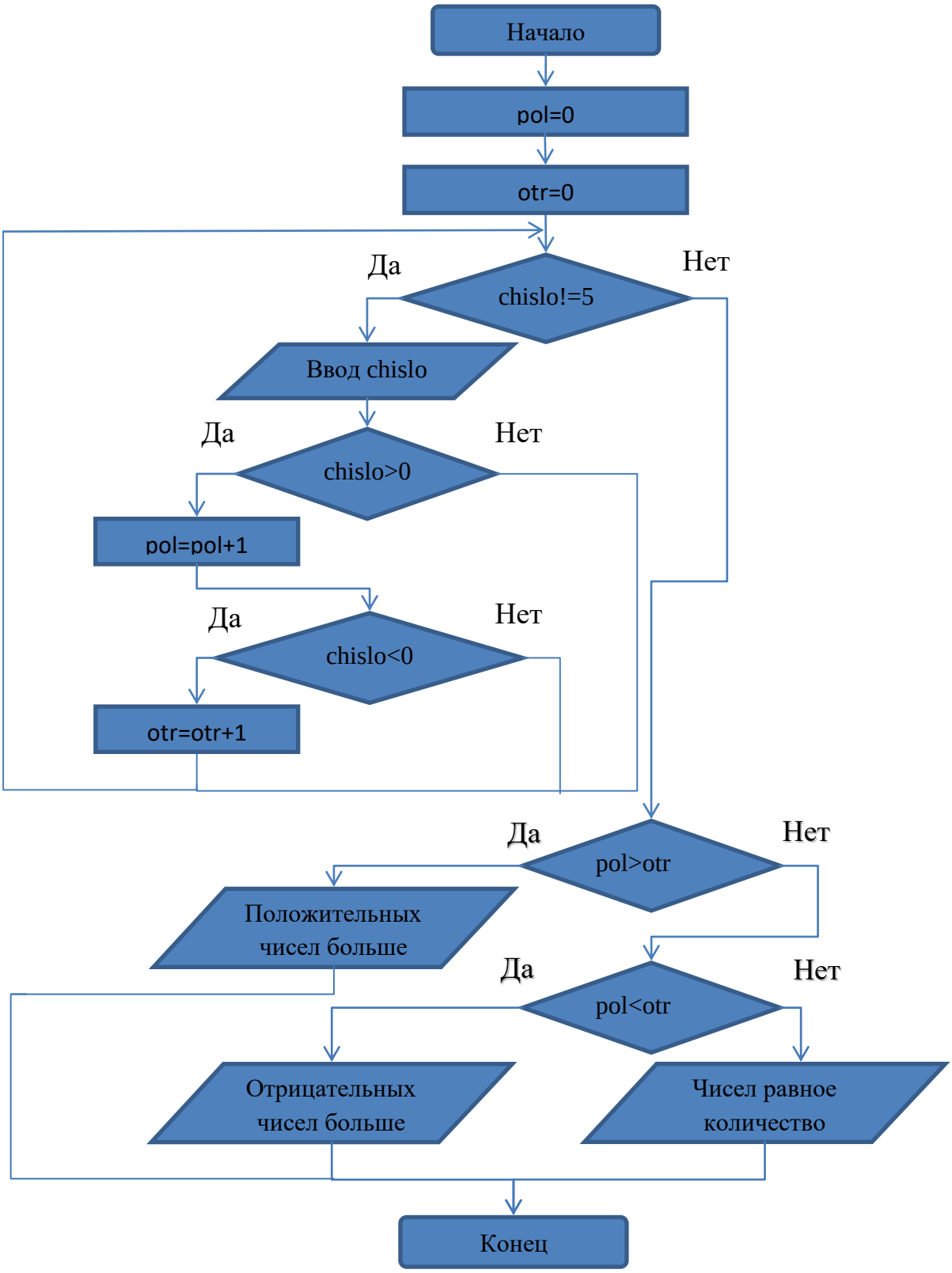

Рисунок 68 – Блок-схема алгоритма решения задачи 6.4.2

В листинге приведен код программы, отвечающий за решение задачи:

```
pol=0
otr=0
chislo=0
while chislo!=5: # Пока число, введенное в цикле, не равно числу 5, выполняются операторы цикла
   chislo=float(input("Введите очредное число = "))
   if chislo>0: # Проверка на положительность очередного введенного числа
      pol=pol+1
   if chislo<0: # Проверка на отрицательность очередного введенного числа
         otr=otr+1
if pol>otr:
   print("\n Положительных чисел больше")
else:
   if pol<otr:
      print("\n Отрицательных чисел больше")
   else:
      print("\n Чисел равное количество")
print("Положительных чисел = ", pol)
print("Отрицательных чисел = ", otr)
```

Задача 6.4.3. Вводится последовательность вещественных чисел не равных нулю. Известно, что последний элемент последовательности равен пяти. Определить отдельно суммы всех отрицательных чисел и положительных чисел, исключив последнее введенное число равное пяти.

Решение. Блок-схема алгоритма решения задачи представлена на рисунке 69. Алгоритм решения очень похож на алгоритмы решения задач 6.4.1 и 6.4.2.

В листинге приведен код программы, отвечающий за решение задачи:

```
pol=0
otr=0
chislo=0
while chislo!=5: # Пока число, введенное в цикле, не равно числу 5, выполняются операторы цикла
   chislo=float(input("Введите очредное число = "))
   if chislo>0: # Проверка на положительность очередного введенного числа
      pol=pol+chislo # Суммирование введеных положительных чисел
   else:
      otr=otr+chislo # Суммирование введеных отрицательных чисел
```

pol=pol-5 # Согласно условию вычитаем из суммы последнее число равное 5

print("\n Сумма положительных чисел равна = ", pol)

print("\n Сумма отрицательных чисел равна = ", otr)

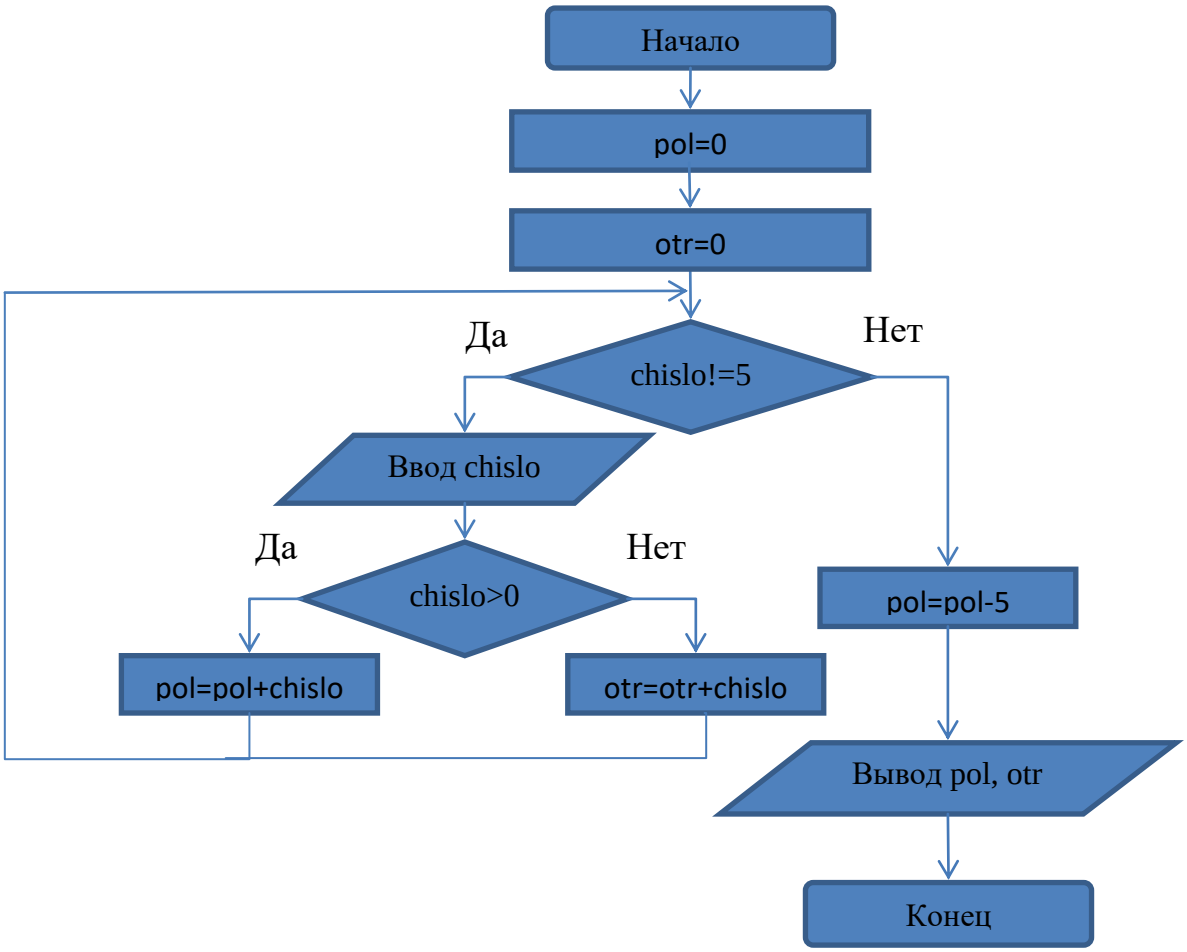

Рисунок 69 – Блок-схема алгоритма решения задачи 6.4.3

Задача 6.4.4. Вводится последовательность целых чисел не равных нулю. Известно, что для завершения последовательности нужно ввести 0. Найти среднее арифметическое этих чисел.

Решение. Блок-схема алгоритма решения задачи представлена на рисунке 70. Нужно учитывать, что последнее число равное нулю вводится для выхода и не принадлежит последовательности целых чисел не равных нулю. В остальном алгоритм решения очень похож на алгоритмы решения задач 6.4.1 - 6.4.3.

В листинге приведен код программы, отвечающий за решение задачи:

```
kol=-1
sum=0
chislo=1
```

```
while chislo!=0: # Пока число, введенное в цикле, не равно числу 0, выполняются операторы цикла
    chislo=int(input("Введите очередное число "))
    kol=kol+1
    sum=sum+chislo
if kol==0:
    print("\n В последовательности нет ни одного числа")
    input("Нажмите ENTER, чтобы завершить программу")
else:
    srarifm=sum/kol
    print("\n Среднее арифметическое чисел равно = ", srarifm)
    print("\n Количество чисел = ", kol, "Сумма чисел чисел = ", sum)
    input("Нажмите ENTER, чтобы завершить программу")
```

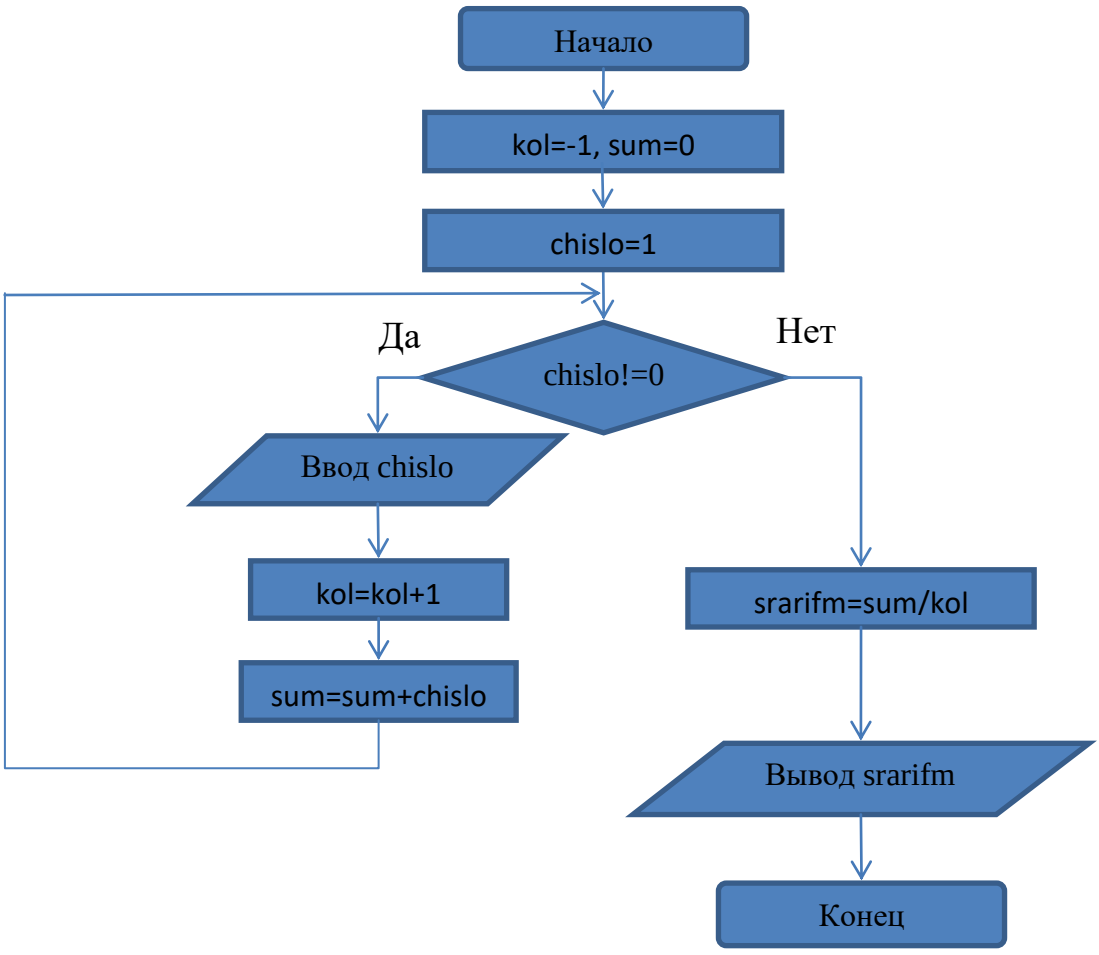

Рисунок 70 – Блок-схема алгоритма решения задачи 6.4.4

Задача 6.4.5. Вводится последовательность вещественных чисел не равных нулю. Известно, что для завершения последовательности нужно ввести 0. Найти количество отрицательных чисел и их среднее арифметическое.

Решение. Блок-схема алгоритма решения задачи представлена на рисунке 71. Алгоритм решения схож с алгоритмами разобранными в решениях задач 6.4.1 - 6.4.4.

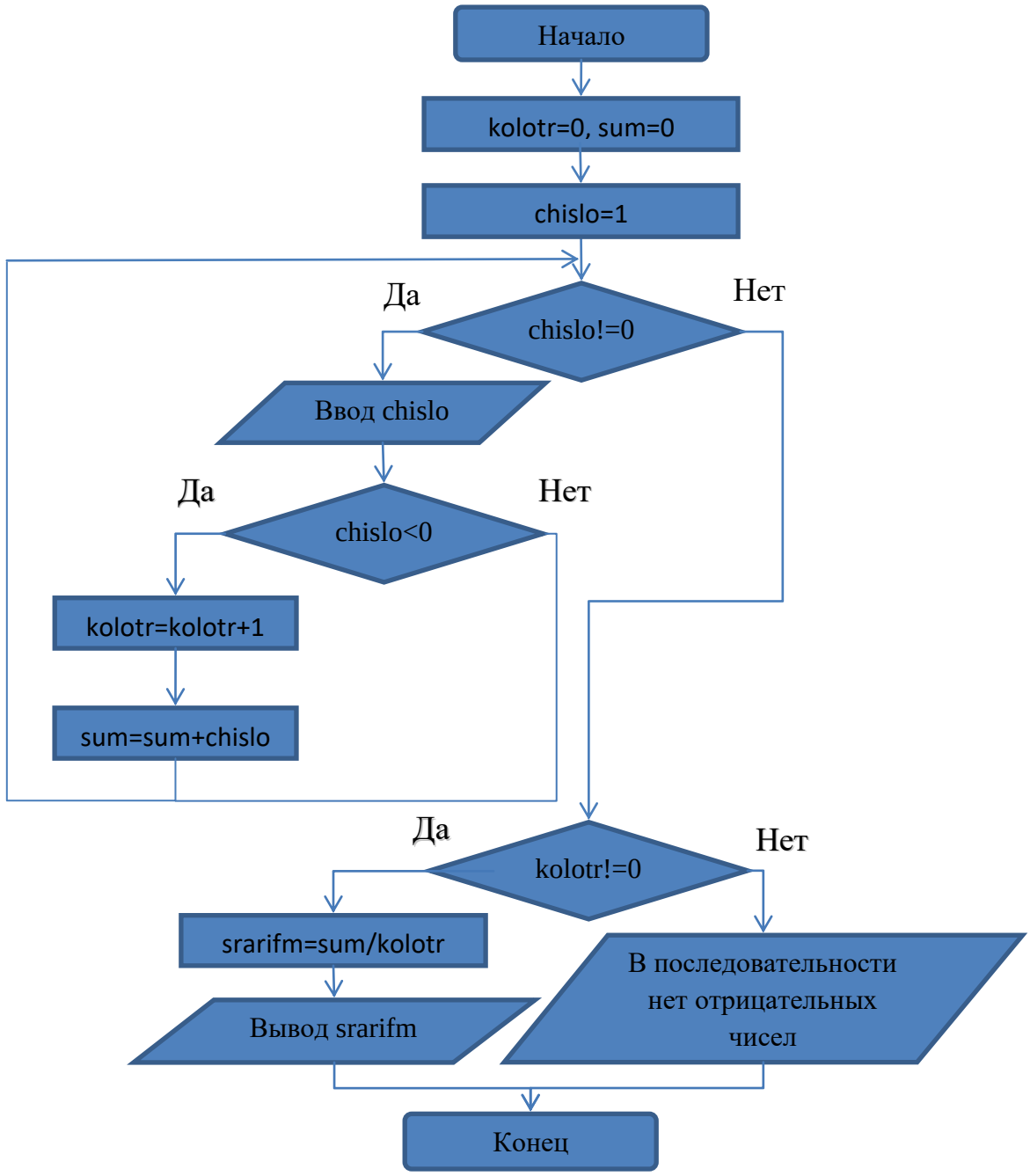

Рисунок 71 – Блок-схема алгоритма решения задачи 6.4.5

В листинге приведен код программы, отвечающий за решение задачи:

```
kolotr=0
sum=0
chislo=1
while chislo!=0: # Пока число, введенное в цикле, не равно числу 0, выполняются операторы цикла
```

```
        chislo=float(input("Введите очередное число "))
        if chislo<0:
            kolotr=kolotr+1
            sum=sum+chislo
    if kolotr!=0:
        srarifm=sum/kolotr
        print("\n Среднее арифметическое отрицательных чисел равно = ", srarifm)
        print("\n Количество отрицательных чисел = ", kolotr, "Сумма отрицательных чисел чисел = ", sum)
    else:
        print("\n В последовательности нет отрицательных чисел")
    input("Нажмите ENTER, чтобы завершить программу")
```

6.5 Контрольные вопросы

1. Расскажите, в каких случаях применяются циклы с неизвестным числом повторений.
2. Какая циклическая структура может считаться итеративной?
3. Нарисуйте общий вид алгоритма оператора цикла while.
4. Напишите синтаксис оператора цикла while.
5. Расскажите о работе оператора цикла while. Приведите примеры.
6. В каких случаях применяются рекуррентные соотношения? Расскажите об алгоритме вывода рекуррентной формулы.

6.6 Задачи для самостоятельного решения

1. Разработайте алгоритм и программу решения следующей задачи. Найдите максимальное значение функции $y = {cosx}/{sinx}$ на отрезке $[a, b]$ с шагом h.
2. Вводится последовательность вещественных чисел. Известно, что последний элемент последовательности равен 5. Найдите количество положительных чисел и минимальное из них. Разработайте алгоритм и программу.
3. Вводится последовательность целых чисел. Известно, что последний элемент последовательности равен 1. Определите, каких среди них больше: положительных или отрицательных. Разработайте алгоритм и программу.
4. Вводится последовательность вещественных чисел, не равных нулю. Известно, что для завершения последовательности нужно ввести 0. В программе должны вычисляться суммы всех положительных и всех отрицательных чисел. Разработайте алгоритм и программу.
5. Вводится последовательность целых чисел. Известно, что для завершения последовательности нужно ввести 0. Определите сумму положительных чисел до первого отрицательного числа. Разработайте алгоритм и программу.

6. Вводится последовательность целых чисел. Известно, что для завершения последовательности нужно ввести 10. Вычислите сумму не равных нулю чисел и выведите эту сумму в качестве ответа. Разработайте алгоритм и программу.

7. Вводится последовательность целых чисел. Известно, что для завершения последовательности нужно ввести 100.В программе определить количество чисел, сумма которых меньше заданного числа. Разработайте алгоритм и программу.

8. Известно, что начальный вклад клиента R в банк и процент годового дохода Pr. Определите срок, через который вклад превысит 1 млн рублей и величину этого вклада. Разработайте алгоритм и программу.

9. Вычислите сумму четных чисел на отрезке от 10 до 30. Разработайте алгоритм и программу.

10. Определите идеальный вес для взрослых людей по формуле: Идеальный вес = Рост - 100. Выход из цикла: значение роста = 250. Разработайте алгоритм и программу.