5 ЦИКЛИЧЕСКИЙ АЛГОРИТМ

5.1 Простой циклический процесс

Очень часто в программах надо выполнить определенные операторы несколько раз. Нелогично записывать эту последовательность действий двадцать или пятьдесят раз подряд. В этих случаях организуют циклические вычисления. Алгоритм называется **циклическим,** если определенная последовательность шагов выполняется несколько раз в зависимости от заданной величины, которая называется **параметром цикла.** Цикл заканчивается, когда параметр принимает определенное значение. Для организации циклов **с известным количеством** повторений в языке Python используется оператор **for.** Его общий вид в алгоритме представлен на рисунке 42.

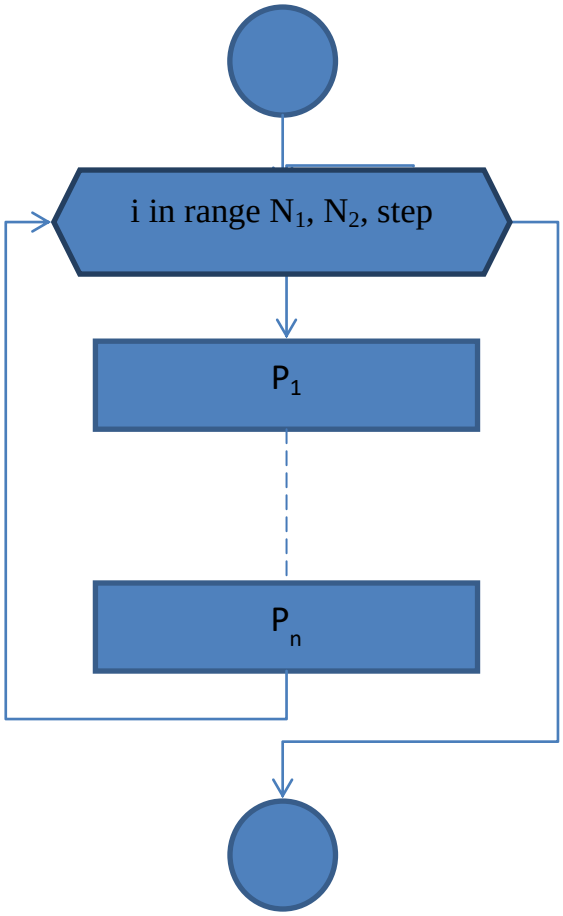

Рисунок 42 – Общий вид оператора цикла **for**

Отметим, что оператор цикла **for** в языке **Python** может иметь разные формы записи. Рассмотрим синтаксис первой из них. Назовем ее **«цикл по возрастающим значениям параметра»**. Обратите внимание на отступы, которые необходимо делать, если мы хотим, чтобы операторы P_1,...,P_n выполнялись внутри цикла.

for i **in range** (N_1, N_2, **step**):

P_1
.
.
P_n

Тело цикла

где **for** (для) - служебное слово; **i** - имя переменной, в которой сохраняются значения элементов; **P_1,...,P_n** - операторы; **in** - в; **range** - встроенная функция языка **Python**; **step** - шаг, является необязательным параметром.

Отметим, что аргументами функции **range** могут быть только целые числа. Работа оператора цикла **for** в такой конструкции заключается в следующем. **При** первом **вхождении в цикл** параметр цикла **i** принимает значение, **равное** величине нижней границы **N_1**, и выполняется оператор или операторы в теле цикла. Затем значение параметра увеличивается на величину **step**, и вновь выполняется тело цикла. Подобные действия будут повторяться до тех пор, пока значение параметра цикла не станет равным величине N_2-1, после чего осуществляется выход из цикла. Если аргумент **step** пропущен в функции **range**, то шаг изменения параметра цикла будет равен единице.

Задача 5.1. Найдите сумму целых чисел от 1 до 50. Блок-схема алгоритма решения задачи представлена на рисунке 43.

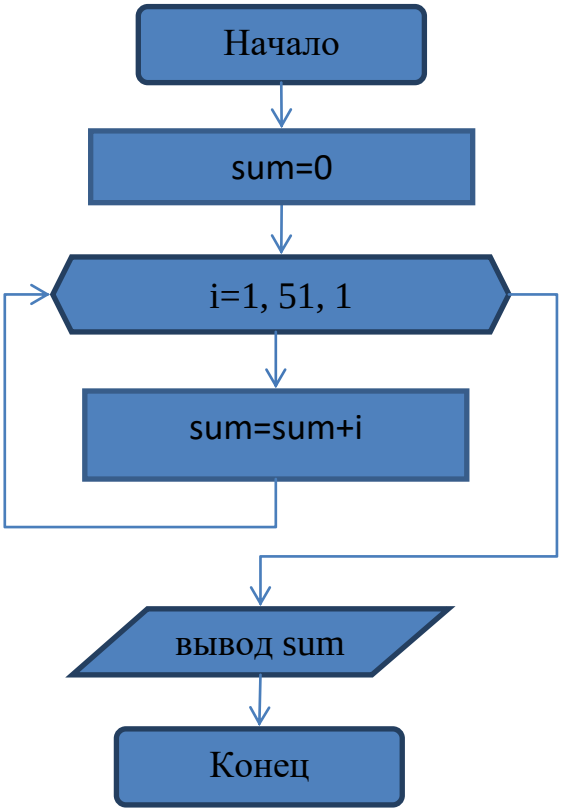

Рисунок 43 – Блок-схема алгоритма решения задачи 5.1

При первом вхождении цикл параметр цикла примет значение, равное единице, и выполнится оператор **sum=sum+i**. Затем параметр **i** будет последовательно увеличиваться на величину шага, который равен единице, и всякий раз в цикле будет выполняться оператор **sum=sum+i**. Следует отметить, что конечное значение верхней границы 51 достигнуто не будет. Счетчик, как иногда называют параметр цикла, остановится на предыдущем значении, т. е. значении, равном 50. В листинге приведен код программы, отвечающий за решение задачи:

```
sum=0
for i in range(1, 51, 1):
   sum=sum+i
print("Сумма=", sum)
```

Если учесть, что параметр **step** является необязательным, то заголовок цикла можно переписать иначе:

```
sum=0
for i in range(1, 51):
   sum=sum+i
print("Сумма=", sum)
```

Попробуем написать ту же программу, но с **циклом по убывающему значению параметра**. Будем считать ее второй формой записи оператора цикла **for**. Последний аргумент в функции **range** теперь равен минус единице. Соответственно, параметр цикла **i** будет изменяться от 50 до 1, значение 0 в перебор включено не будет:

```
sum=0
for i in range(50, 0, -1):
   sum=sum+i
print("Сумма=", sum)
```

Третья форма записи оператора цикла **for** может быть организована с использованием функции **range** и одним параметром, указывающим значение верхней границы. Таким образом, параметр цикла будет меняться от **0** до значения N_2-1. Синтаксис оператора в общем виде можно записать следующим образом:

for i **in range** (N_2):

P_1
.
.
P_n

} Тело цикла

Программу «Найти сумму целых чисел от 1 до 50» теперь можно написать, как следующую последовательность операторов:

```
sum=0
for i in range(51):
   sum=sum+i
print("Сумма=", sum)
```

Отметим, что результатом выполнения всех рассмотренных программ является число 1275.

Более широкие возможности применения оператора цикла **for** мы рассмотрим ниже, а сейчас приведем еще один пример, связанный с обработкой последовательности символов:

```
slovo=input("Введите слово  ")
for i in slovo:
   print(i, end=" ")
```

Как видно из листинга приведенного выше, в цикле происходит перебор символов того слова (строки), которое вводит пользователь с клавиатуры. Результат работы программы представлен на рисунке 44. В операторе **print**, в отличие от выше рассмотренных примеров, используется небольшое дополнение, а именно, оператор **end=" "**, что позволило расположить результаты работы программы в строке, а не в столбик, как было сделано ранее.

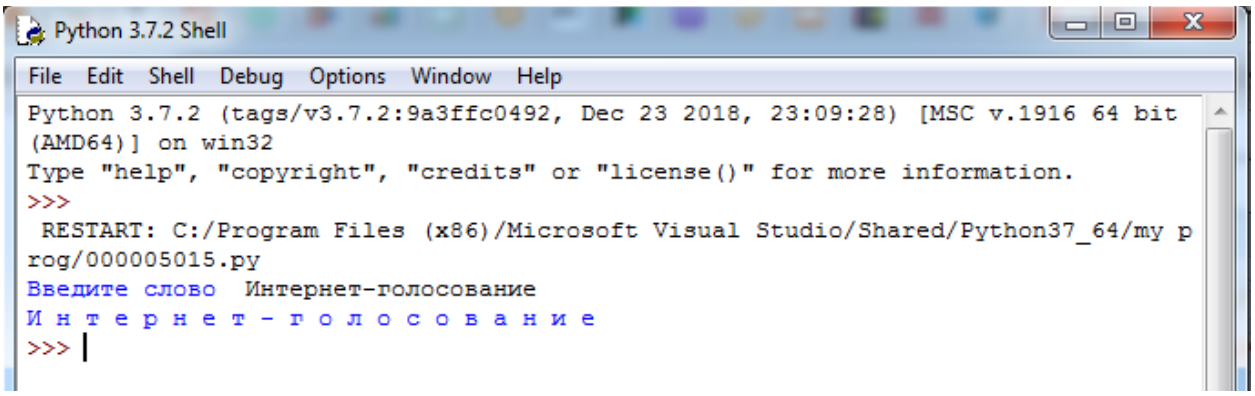

Рисунок 44 – Результат работы программы с циклом **for** для обработки последовательности символов

5.2 Сложный циклический процесс. Вложенные циклы

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

внутренним. Общий вид фрагмента блок-схемы алгоритма сложного циклического процесса представлен на рисунке 45.

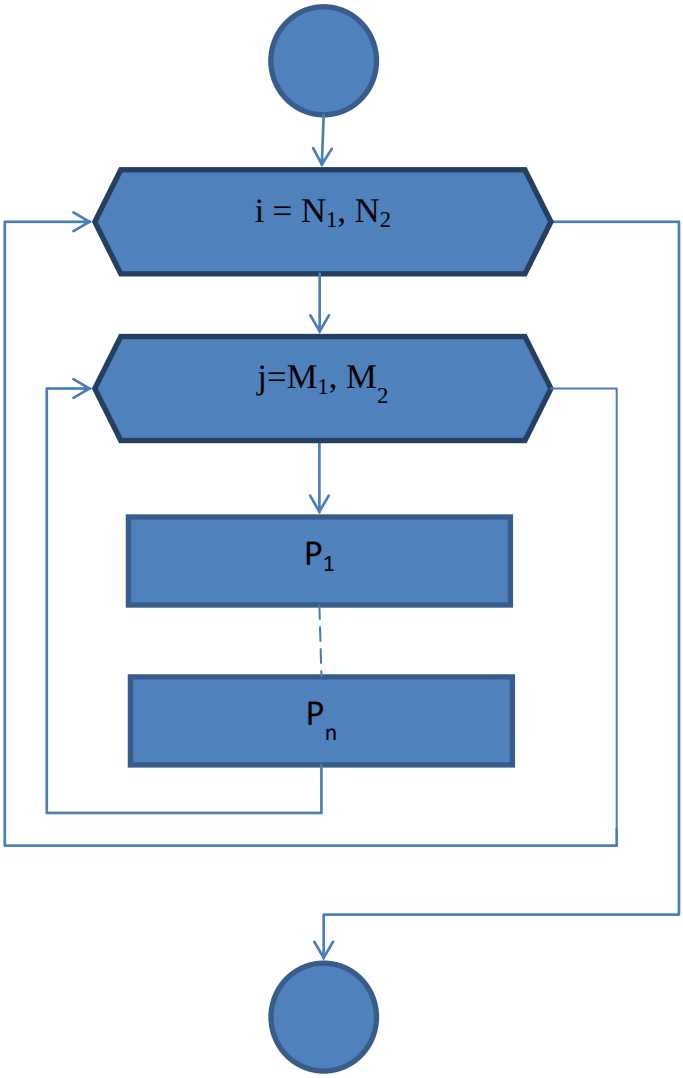

Рисунок 45 – Фрагмент блок-схемы сложного циклического процесса

Синтаксис операторов сложного цикла представлен ниже. Обратите внимание на отступы, выполненные для внутреннего цикла и операторов **P_1 … P_n**, которые в нем будут выполняться:

for i **in range** (N_1, N_2): # внешний цикл
  for j **in range** (M_1, M_2): # внутренний цикл
    P_1
    .
    .
    P_n

Тело цикла

Рассмотрим работу сложного циклического процесса. При первом вхождении в цикл параметр внешнего цикла **i** принимает значение равное **N_1**. Управление передается во внутренний цикл, в котором параметр цикла **j**

принимает значение, равное $\mathbf{M_1}$, и выполняется оператор (операторы), который записан во внутреннем цикле. Затем параметр внутреннего цикла **j** увеличивается на единицу (если опущен аргумент **step** в функции **range**), и вновь выполняется тело цикла. Затем параметр внешнего цикла **i** увеличивается на единицу, и вновь начинает свою работу внутренний цикл, в котором параметр цикла **j** будет изменяться от $\mathbf{M_1}$ до $\mathbf{M_2}$, и при каждом прохождении цикла будут выполняться операторы $\mathbf{P_1, \ldots, P_n}$.

Задача 5.2. Разработайте приложение, которое осуществляет вывод на экран таблицы умножения для значений от 1 до 5.

Решение. Листинг решения представлен ниже:

```
for i in range(1, 6):
    for j in range(1, 6):
        print(i, '*', j, '=', i*j, end='\t')
    print()
```

Итак, согласно алгоритму работы сложного циклического процесса установим границы изменения параметра во внутреннем и внешнем циклах от 1 до 6. Во внутреннем цикле выполнится оператор **print(i, '*', j, '=', i*j, end='\t')** в котором сочетается вывод самих значений параметров, строковых выражений для вывода знаков умножения и равенства на экран и, собственно, самого действия i*j, которое обеспечит перемножение параметров. «Пустой» оператор **print()** (обратите внимание на отступ) не выполняется во внутреннем цикле, а служит для пропуска строки между выводом столбцов ответа. Результат работы программы представлен на рисунке 46.

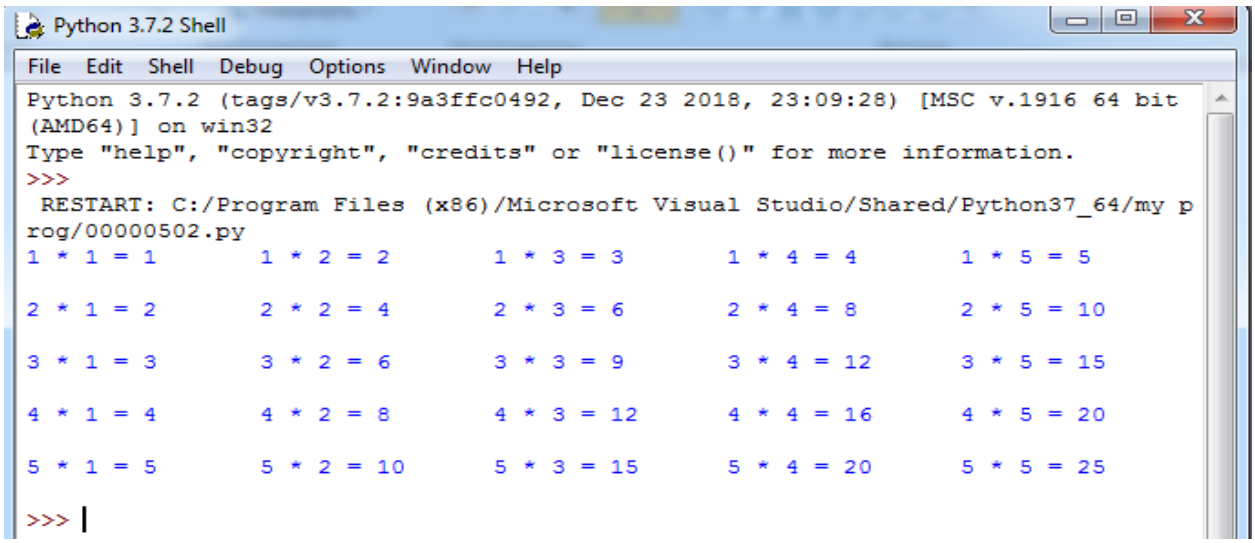

Рисунок 46 – Вывод таблицы умножения

5.3 Пояснение примеров заданий на применение циклов for

Задание 5.1. Сколько раз будет выполнен оператор **d=5** в теле цикла?

```
d=4
r=15
for i in range(d+1, r,1):
   d=5
```

Решение. Подставив значения переменных **d** и **r**, получаем, что параметр цикла **i** меняется от **5** до **14**. При первом вхождении в цикл параметр цикла примет значение **5**, а далее будет автоматически увеличиваться на **+1**, пока не достигнет величины верхней границы **14**. Следовательно, оператор выполнится **10** раз.

Задание 5.2. Определите, какое значение будет в ячейке **r** после выполнения группы операторов?

```
r=50
s=0
for i in range(5, 0, -1):
   s=1
   r=r-s
print("r = ", r)
```

Решение. При первом вхождении в цикл параметр цикла примет значение **5**. В программе используется цикл по убывающим значениям, следовательно, параметр цикла при каждом выполнении тела цикла будет уменьшаться на **-1**. При первом вхождении в цикл переменная **s** принимает значение, равное **1**. Выполняется оператор **r=r-s**. После его выполнения в ячейке **r** будет значение **49**. Затем параметр цикла уменьшается на единицу и будет равен четырем. Вновь выполняются операторы **s=l** и **r= r-s**. Выполняя их каждый раз и подсчитывая полученные значения, получим в ячейке **r** значение **45**.

Задание 5.3. Определите, какое значение находится в ячейке **y** после выполнения группы операторов?

```
a=7
d=5
y=0
for i in range(1,4,1):
   y=d
   y=a+2
print("y = ", y)
```

Решение. В цикле с оператором **for** параметр цикла будет меняться от 1 до 3. В цикле выполняются два оператора: **y=d** и **y=a+2**. Операторы выполняются три раза. Каждый раз при прохождении цикла в ячейке **y** будет находиться значение **5** (**y=d**), а после выполнения оператора **y=a+2** значение ячейки **y** будет равно **9**. Следовательно, после выполнения группы операторов в ячейке **y** будет находиться значение **9**.

Задание 5.4. Определите, какое значение находится в ячейке **y** после выполнения группы операторов, и какие значения будет принимать ячейка **i** в теле цикла?

```
a=17.0
d=a
for i in range(3):
    print("i = ", i)
    if a!=d:
        a=a+1
    else:
        y=a
print("y = ", y)
```

Решение. В цикле с оператором f**or** происходит проверка логического выражения **a!=d**. Поскольку значения **a** и **d** равны после выполнения операторов **a=17.0** и **d=a**, логическое выражение все время будет оставаться ложным, а следовательно, выполнение оператора **y=a** в ветви **else** гарантировано при каждом прохождении цикла. Таким образом, с помощью оператора **print** на экран будет выведено значение **17.0**. Параметр цикла **i**, согласно теории, рассмотренной выше, будет принимать последовательные значения от **0** до **2**.

Задание 5.5. Определите, какое значение находится в ячейке **y** после выполнения группы операторов?

```
a=1
y=1
for i in range(2,6,1):
    a=a+10
    y=a+10
print("y = ", y)
```

Решение. При первом вхождении в цикл параметр цикла примет значение нижней границы, равное двум. В цикле будут выполняться два оператора, таким образом, в ячейке **a** оказывается значение **11**, а после выполнения оператора **y=a+10** значение, находящееся в ячейке **y**, станет равным **21**. Параметр увеличивается на единицу и будет равен трем,

выполняется оператор **a=a+10**, и в ячейке **a** оказывается число **21**, а в ячейке **y** находится число **31**. Выполнив подобные действия еще два раза, получим в ячейке **y** значение, равное **51**.

Задание 5.6. Определите, какое значение находится в ячейке **y** после выполнения группы операторов?

```
a=5
d=5
y=0
for i in range(7,1,-1):
    a=d
    y=a+10
print("y = ", y)
```

Решение. При первом вхождении в цикл параметр цикла примет значение, равное **7**. В ячейке **a** оказывается значение, равное **5**, а в ячейке **y** после выполнения оператора **y=a+10** будет значение, равное **15**. В цикле по убывающим значениям параметра значение ячейки **i** уменьшится на единицу и будет равно **6**. Выполняется оператор **a=d**, и в ячейке **a** снова оказывается число **5**. В ячейке **y** после выполнения оператора **y=a+10** будет значение **15**. Эти действия будут повторяться до тех пор, пока параметр цикла не достигнет значения границы, равного **2**. Происходит выход из цикла. В ячейке **y** находится значение, равное **15**.

Задание 5.7. Определите, какое значение находится в ячейке **y** после выполнения цикла с оператором **for**?

```
a=5
d=5
y=0
for i in range(2,6,1):
    y=d+10
y=a+10
y=a*d
print("y = ", y)
```

Решение. В данном фрагменте программы при выполнении цикла смещен вправо только один оператор **y=d+10**. Следовательно, он и будет выполняться. При первом прохождении цикла в ячейке **y** оказывается значение, равное **15**. Параметр цикла увеличивается на единицу. Вновь выполняется оператор **y=d+10**, и так как значение ячейки d не изменялось, оно вновь оказывается равным **15**. Эти действия будут повторяться до тех пор, пока значение параметра не достигнет значения верхней границы, равной **5**. После чего происходит выход из цикла, и в ячейке **y** будет

находиться значение, равное **15**. Поставив вопрос к данному заданию по-другому: «Какое значение находится в ячейке y после выполнения фрагмента программы?», - следует проанализировать еще два оператора, где изменяется значение **y**, и получить значение, равное **25**.

Задание 5.8. Сколько раз будет выполнен цикл в следующем фрагменте программы?

```
n=2
s=13
y=0
for i in range(s+2,n+3,-1):
   y=y+1
```

Решение. Подставив исходные значения переменных в заголовок цикла, видим, что параметр цикла **i** в цикле по убывающим значениям будет меняться от **15** до **6**, изменяясь при этом с шагом **-1**. При первом вхождении в цикл параметр цикла примет значение, равное **15**, затем он будет равен **14**, **13**, **12** и т. д., и цикл будет выполняться до тех пор, пока параметр не станет **равным шести**. Таким образом, цикл выполнится **10** раз.

5.4 Примеры решения задач

Задача 5.4.1. Последовательно вводятся шесть целых чисел. Определите, каких среди них больше: положительных или отрицательных.

Решение. Блок-схема алгоритма решения задачи представлена на рисунке 47.

Ячейки **pol** и **otr** играют роль счетчиков. Счетчики в цикле будут увеличиваться на единицу операторами **otr=otr+1** и **pol=pol+1**, поэтому для того, чтобы конечный результат не был искажен, ячейки предварительно обнуляются операторами **pol=0** и **otr=0**.

В листинге ниже приведен код программы, отвечающий за решение задачи:

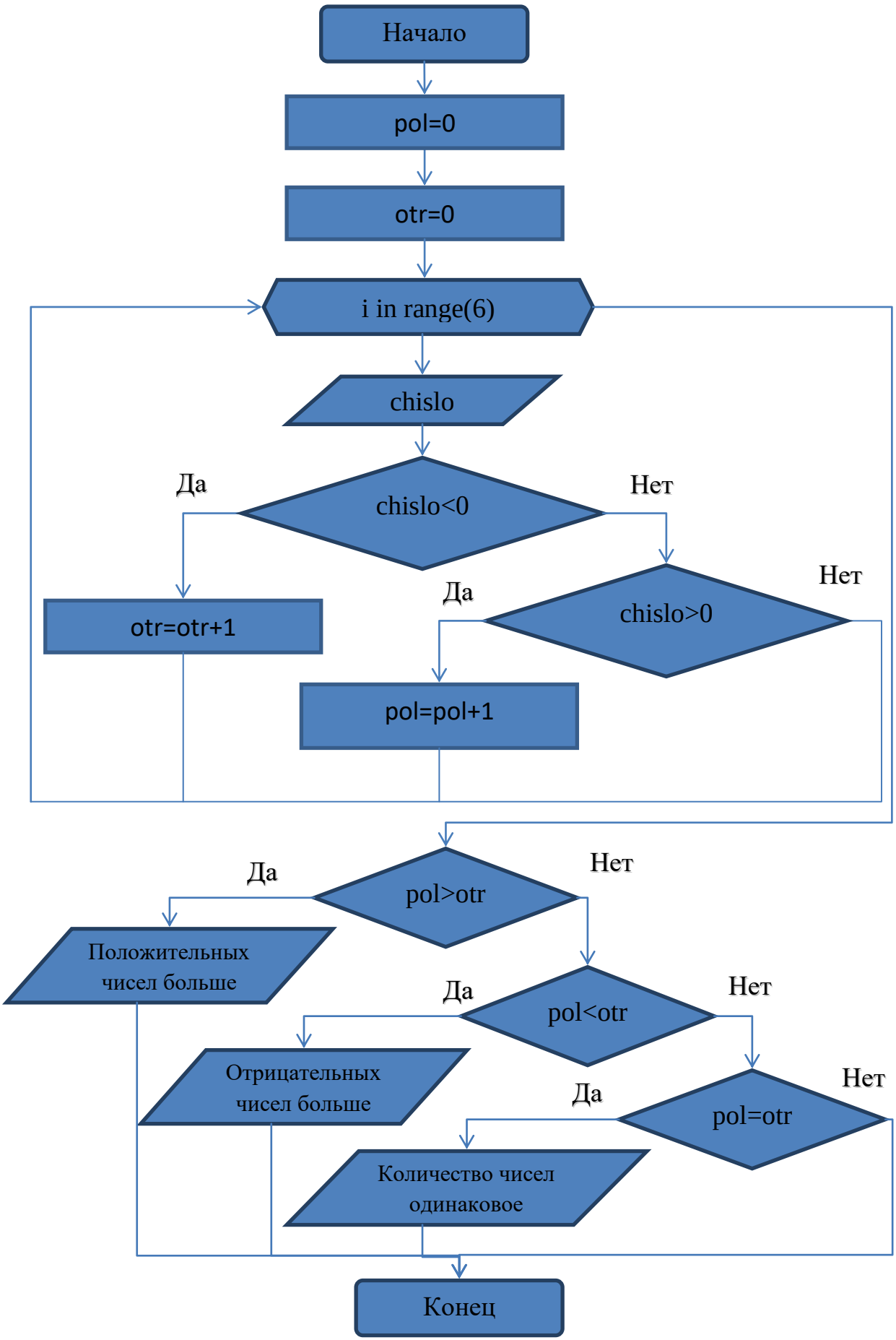

Рисунок 47 – Блок-схема алгоритма решения задачи 5.4.1

```
pol=0 #Счетчик положительных чисел предварительно обнуляется
otr=0 #Счетчик отрицательных чисел предварительно обнуляется
for i in range (6):
   chislo=int(input("Введите число "))
   if chislo<0:
      otr=otr+1 #Счетчик отрицательных чисел увеличивается на
единицу
   elif chislo>0:
      pol=pol+1 #Счетчик положительных чисел увеличивается на
единицу
if pol>otr:
   print("Положительных чисел больше")
if pol<otr:
   print("Отрицательных чисел больше")
if pol==otr:
   print("Количество чисел одинаковое")
```

Задача 5.4.2. Последовательно вводится пять вещественных чисел. Найдите минимальное из положительных чисел.

Решение. Блок-схема алгоритма решения задачи представлена на рисунке 48.

Для реализации алгоритма поиска минимального числа мы отводим ячейку **min**, в которую предварительно заносим любое большое значение, например, в данной программе **+32767**, оператором **min=32767**. В цикле введенное пользователем число сравнивается с хранящимся в ячейке **min**, и если оно меньше, то введенное число заносится в ячейку **min** оператором **min=chislo**. Таким образом, к моменту завершения цикла в ячейке **min** будет находиться минимальный элемент.

При решении подобных задач следует учесть, что для их более корректной работы можно дополнить код программ методами обработки исключений, изложенными в предыдущих темах, либо сделать дополнительные проверки условий (в данной задаче можно было бы предусмотреть тот случай, когда пользователь вводит все отрицательные числа).

В листинге ниже приведен код программы, отвечающий за решение задачи:

```
min=32767
for i in range (5):
   chislo=float(input("Введите число "))
   if chislo>0: #Проверка на положительность очередного введенного
числа
      if chislo<min: #Поиск минимального положительного элемента
         min=chislo
```

```
print("Минимальное положительное число = ", min)
```

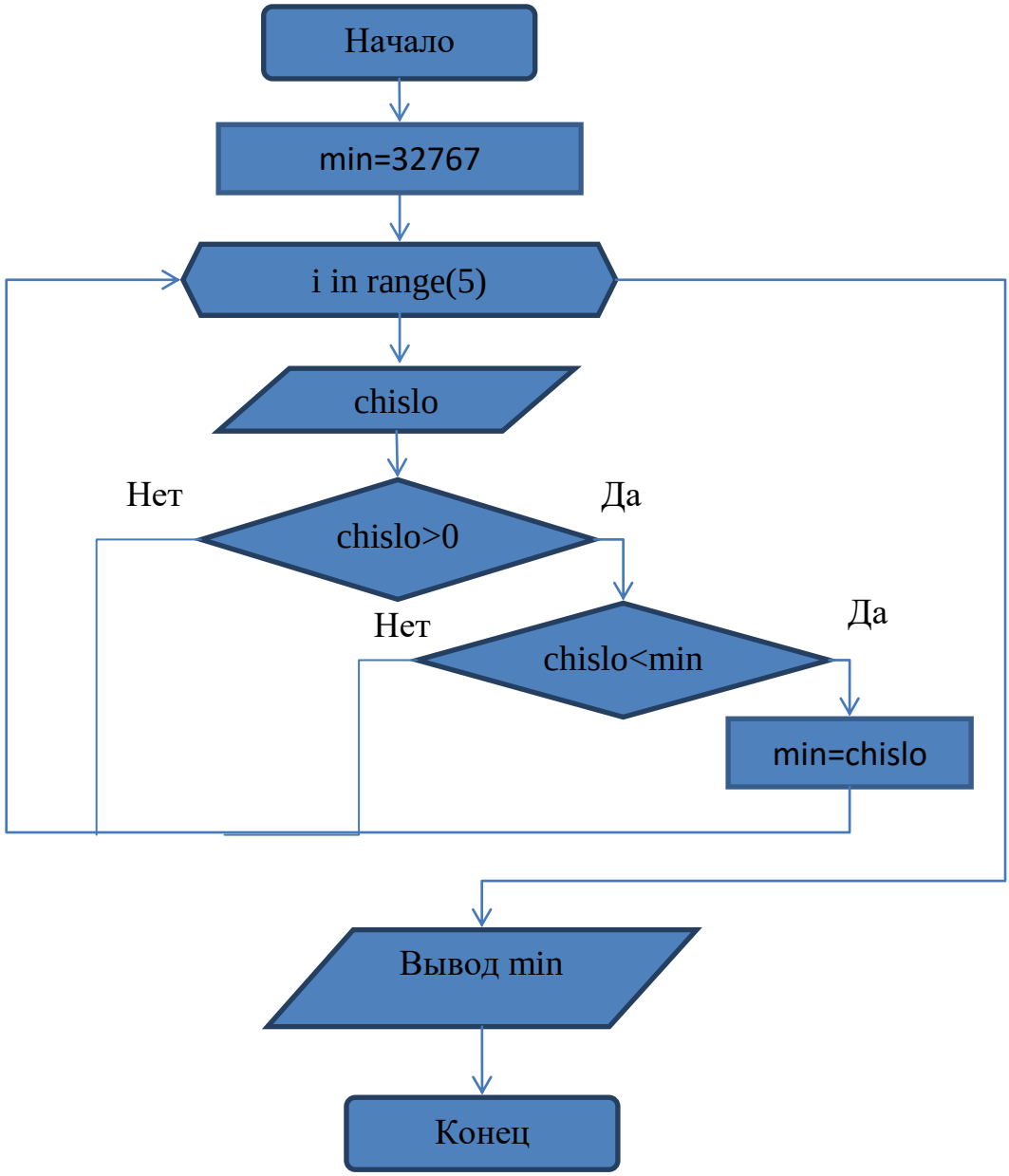

Рисунок 48 – Блок-схема алгоритма решения задачи 5.4.2

Задача 5.4.3. Последовательно вводятся **N** целых чисел. Найдите максимальное из отрицательных значений.

Решение. Блок-схема алгоритма решения задачи представлена на рисунке 49.

Алгоритм поиска максимального элемента обратный алгоритму поиска минимального числа. Для реализации алгоритма поиска максимального числа отводим ячейку **max**, в которую предварительно заносим любое малое значение, например, в данной программе отрицательное число - **32768**, оператором **max= - 32768**. В цикле введенное пользователем число сравнивается с хранящимся в ячейке **max**, и если оно больше, то введенное число заносится в ячейку **max** оператором **max=chislo**. Таким образом, к

моменту завершения цикла в ячейке **max** будет находиться максимальный элемент из всех введенных отрицательных чисел.

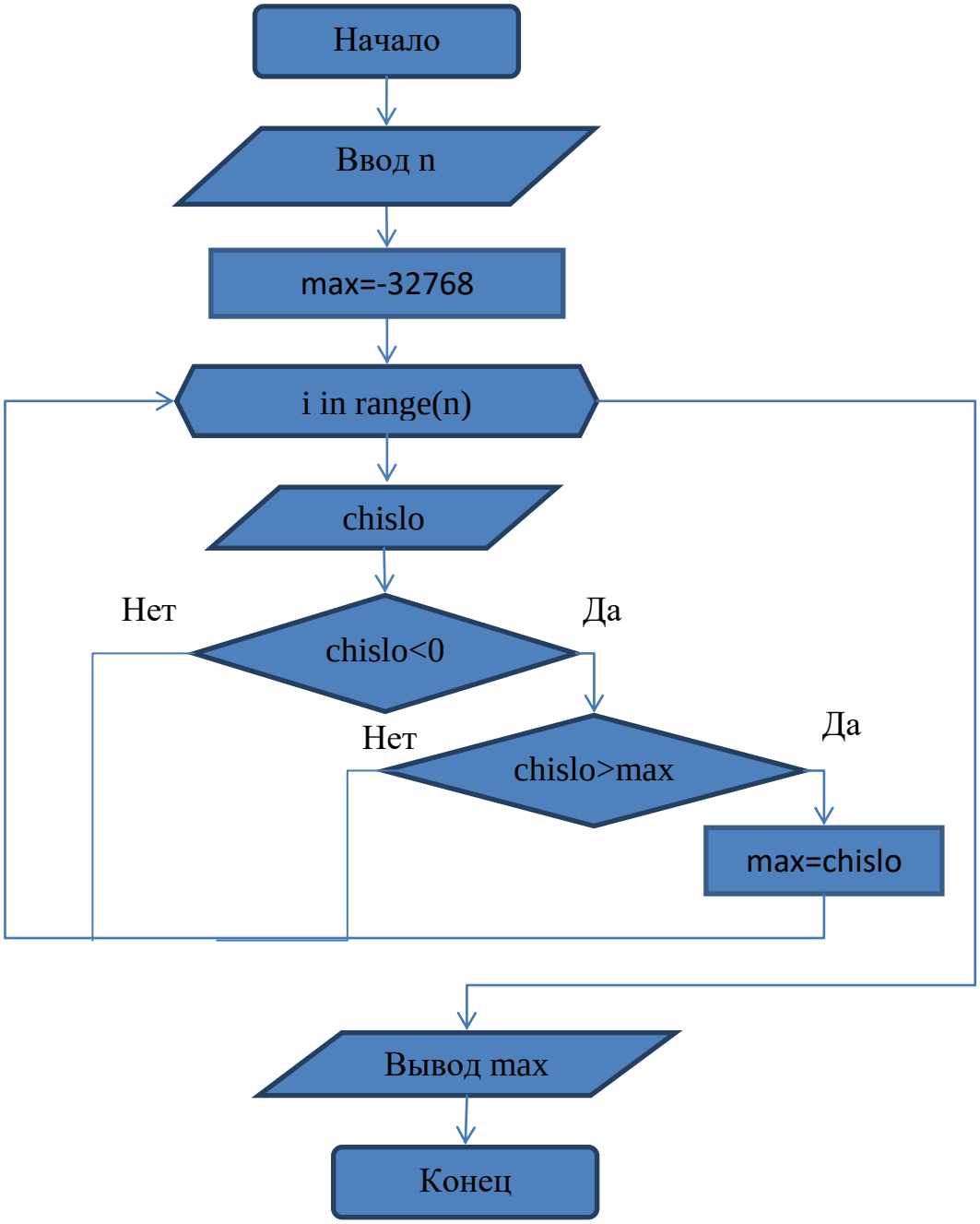

Рисунок 49 – Блок-схема алгоритма решения задачи 5.4.3

В листинге ниже приведен код программы, отвечающий за решение задачи:

```
#Ввод количества чисел для цикла
n=int(input("Введите количество чисел N = "))
max=-32768
for i in range(n):
    chislo=int(input("Введите число "))
```

```
        if chislo<0: #Проверка на отрицательность очередного введенного
числа
            if chislo>max: #Поиск максимального из отрицательных элементов
                max=chislo
    print("Максимальное отрицательное число = ", max)
```

Задача 5.4.4. Последовательно вводятся пять вещественных чисел. Найдите количество отрицательных чисел и минимальное из отрицательных.

Решение. Блок-схема алгоритма решения задачи представлена на рисунке 50.

Алгоритм поиска минимального элемента из отрицательных подобен алгоритму поиска минимального числа в задаче 5.4.3. Для реализации алгоритма поиска минимального числа отводим ячейку **min**, в которую предварительно заносим любое большое значение, например, в данной программе число **32767**, оператором **min= 32767**. Для подсчета количества отрицательных чисел вводим и обнуляем переменную **kolotr=0**. В цикле введенное пользователем число проверяется на отрицательность **chislo<0** и в случае **True** счетчик отрицательных чисел увеличивается на единицу **kolotr=kolotr+1**. Далее в цикле для отрицательных чисел проверяется является ли введенное число наименьшим из всех введеных ранее отрицательных чисел **chislo<min** и при выполнении этого условия введенное число заносится в ячейку **min** оператором **min=chislo**. Таким образом, к моменту завершения цикла в ячейке **min** будет находиться минимальный элемент из всех введенных отрицательных чисел или исходное введенное число 32767 при отсутствии отрицательных чисел. По завершению цикла проверяем наличие введенных отрицательных чисел условием **kolotr==0.** В случае отсутствия отрицательных чисел выводим на печать сообщение "Нет отрицательных чисел ", а при наличии отрицательных чисел выводим на печать количество отрицательных чисел **kolotr** и наименьшее из них **min**.

В листинге ниже приведен код программы, отвечающий за решение задачи:

```
min=32767
kolotr=0
for i in range(1,6):
    chislo=float(input("Введите число = "))
    if chislo<0:
        kolotr=kolotr+1
        if chislo<min:
            min=chislo
if kolotr==0:
    print("Нет отрицательных чисел ")
else:
    print("Количество отрицательных чисел = ", kolotr)
```

print("Минимальное из отрицательных чисел = ", min)

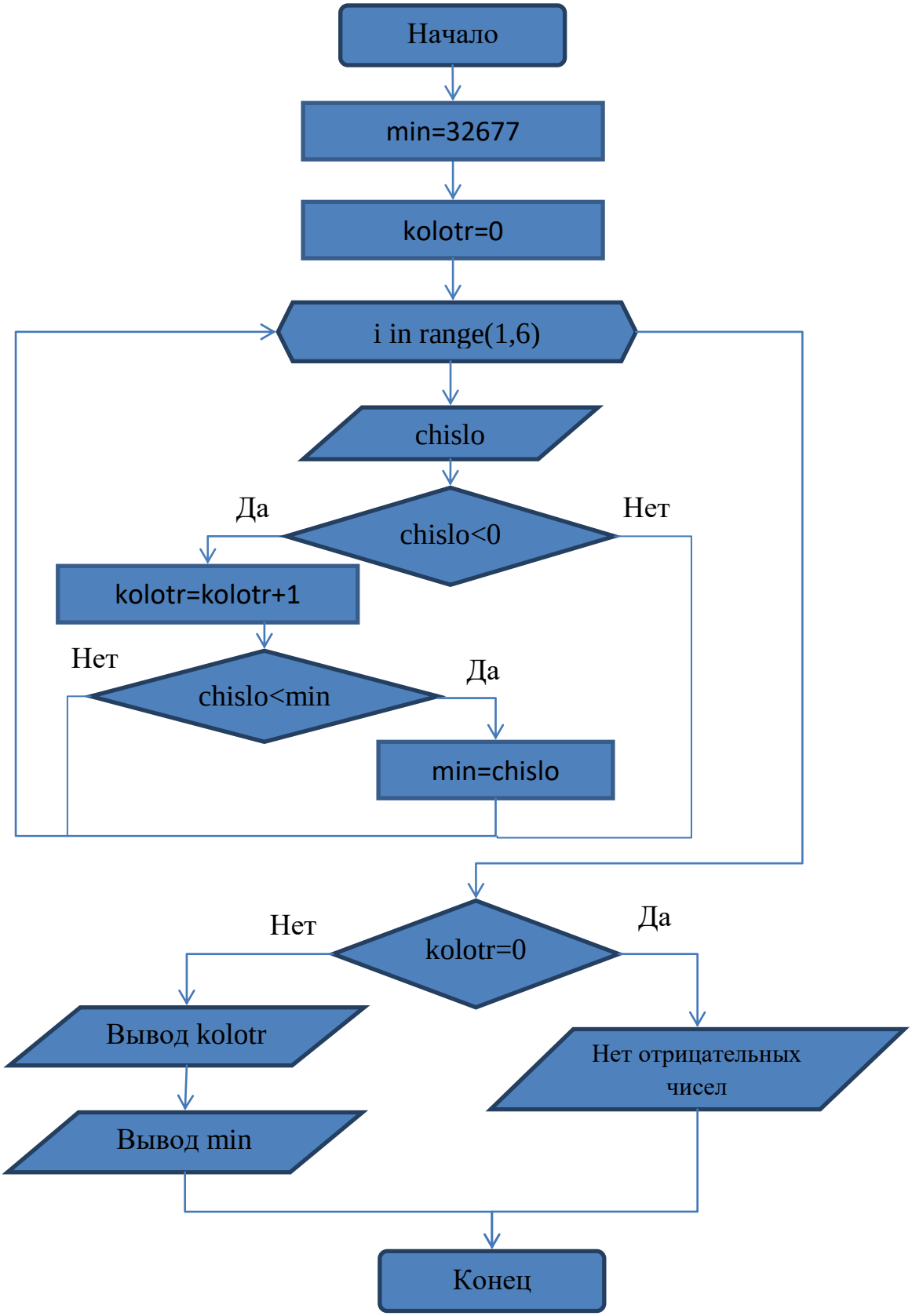

Рисунок 50 – Блок-схема алгоритма решения задачи 5.4.4

Задача 5.4.5. Последовательно вводятся **N** вещественных чисел. Найдите количество положительных чисел и максимальное из них.

Решение. Алгоритм задачи 5.4.5 подобен алгоритму 5.4.4 с точностью до наоборот.

Блок-схема алгоритма алгоритма решения задачи представлена на рисунке 51.

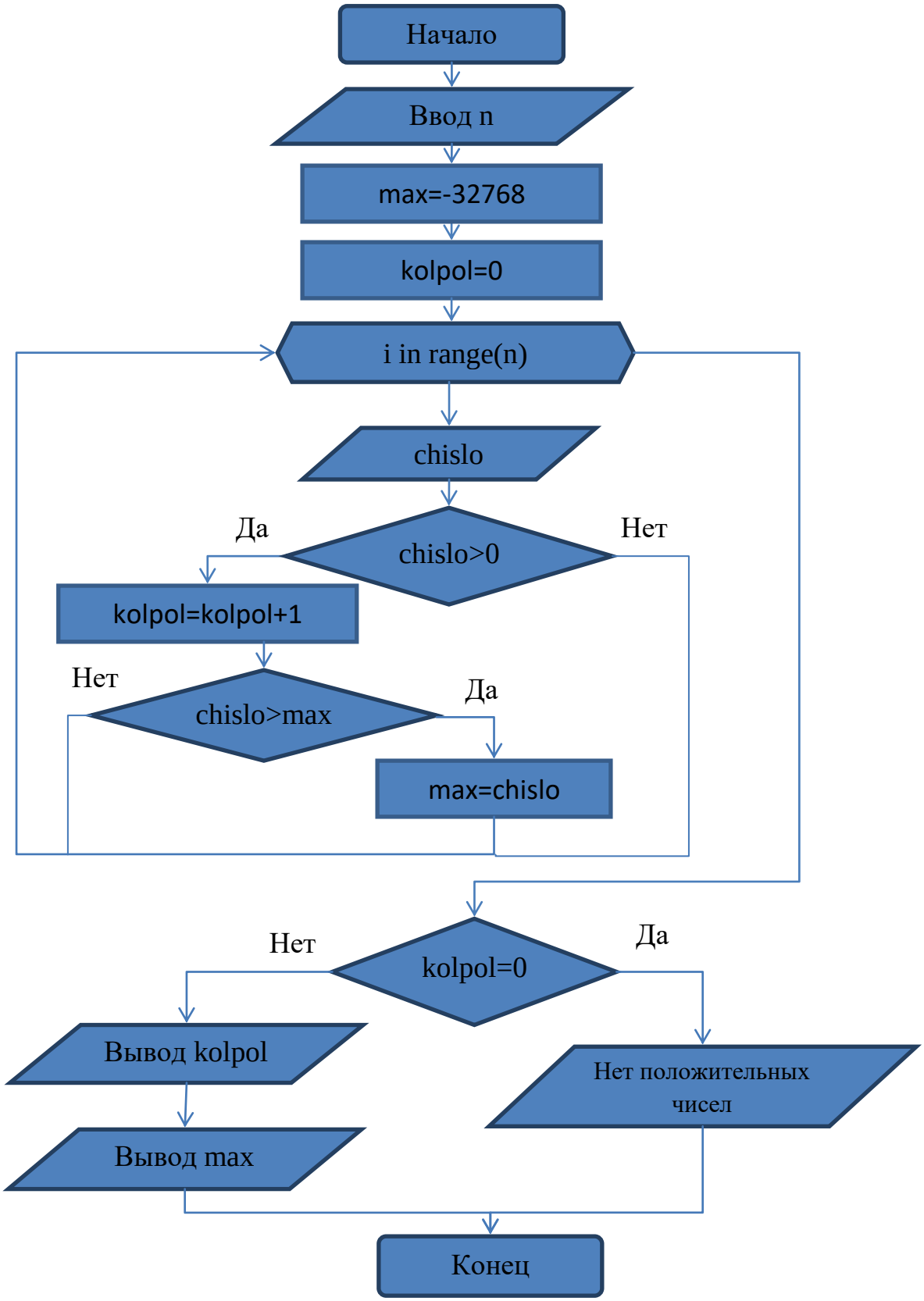

Рисунок 51 – Блок-схема алгоритма решения задачи 5.4.5

В листинге ниже приведен код программы, отвечающий за решение задачи:

```
n=int(input("Ведите количество чисел N = "))
max=-32768
kolpol=0
for i in range(n):
   chislo=float(input("Введите число = "))
   if chislo>0:
      kolpol=kolpol+1
      if chislo>max:
         max=chislo
if kolpol==0:
   print("Нет положительных чисел ")
else:
   print("Количество положительных чисел = ", kolpol)
   print("Максимальное из положительных чисел = ", max)
```

Задача 5.4.6. Последовательно вводятся пять целых чисел. Найдите количество положительных чисел и их среднее арифметическое.

Решение. Поиск среднего арифметического положительных значений заключается в нахождении суммы чисел (ячейка **sum**), нахождении их количества (ячейка **kolpol**) и вычислении среднего арифметического по формуле **srarifm=sum/kolpol**.

В листинге ниже приведен код программы, отвечающий за решение задачи:

```
kolpol=0
sum=0
for i in range(1,6,1):
   chislo=int(input("Введите число = "))
   if chislo>0:
      kolpol=kolpol+1
      sum=sum+chislo
if kolpol==0:
   print("Нет положительных чисел ")
else:
   srarifm=sum/kolpol
   print("Количество положительных чисел = ", kolpol)
   print("среднее арифметическое положительных значений = ",srarifm)
```

Блок-схема алгоритма решения задачи представлена на рисунке 52.

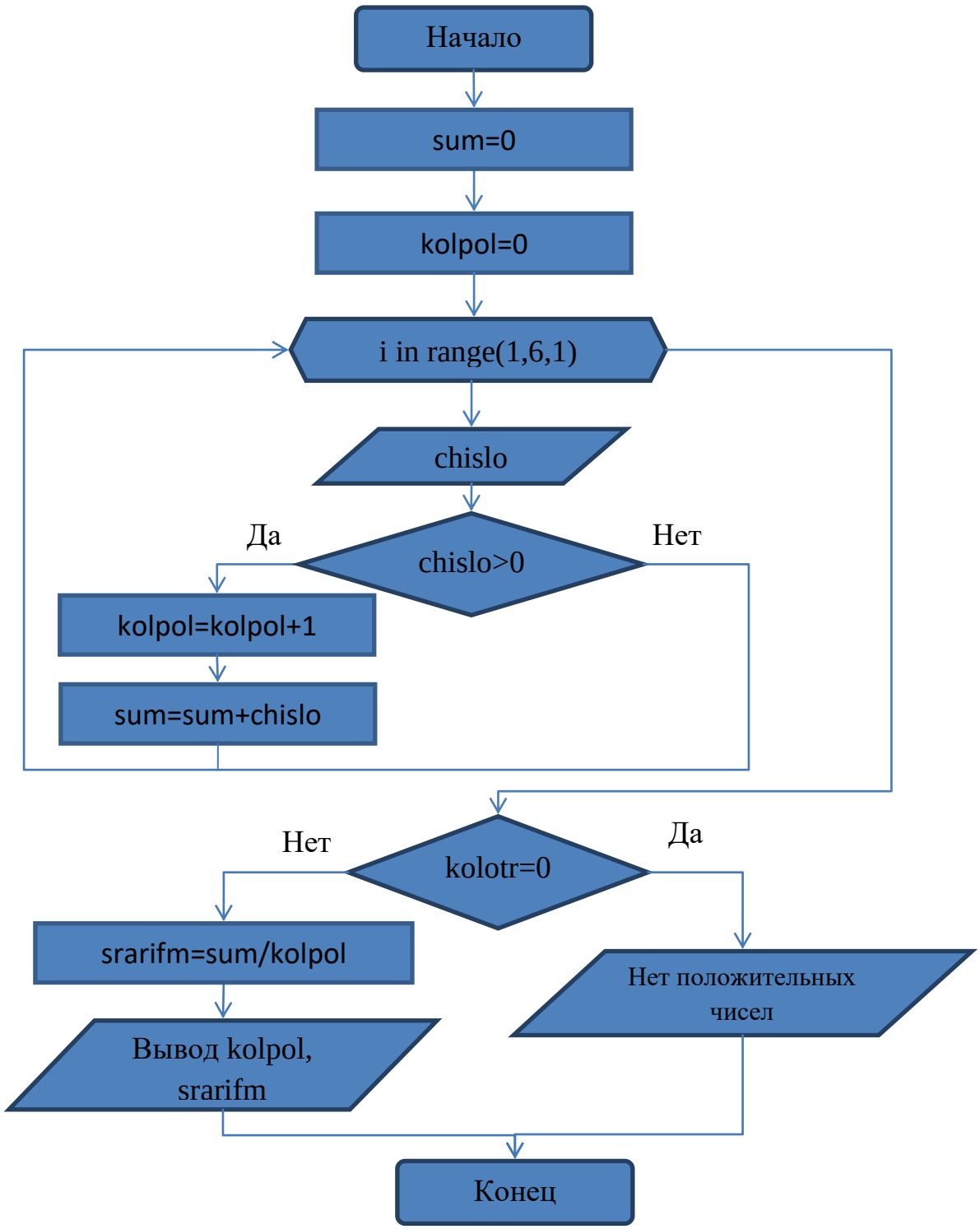

Рисунок 52 – Блок-схема алгоритма решения задачи 5.4.6

Задача 5.4.7. Вводится последовательность из **N** целых чисел. Найдите, сколько в ней чисел, равных числу **100**, а также количество отрицательных чисел.

Решение. Блок-схема алгоритма решения задачи представлена на рисунке 53.

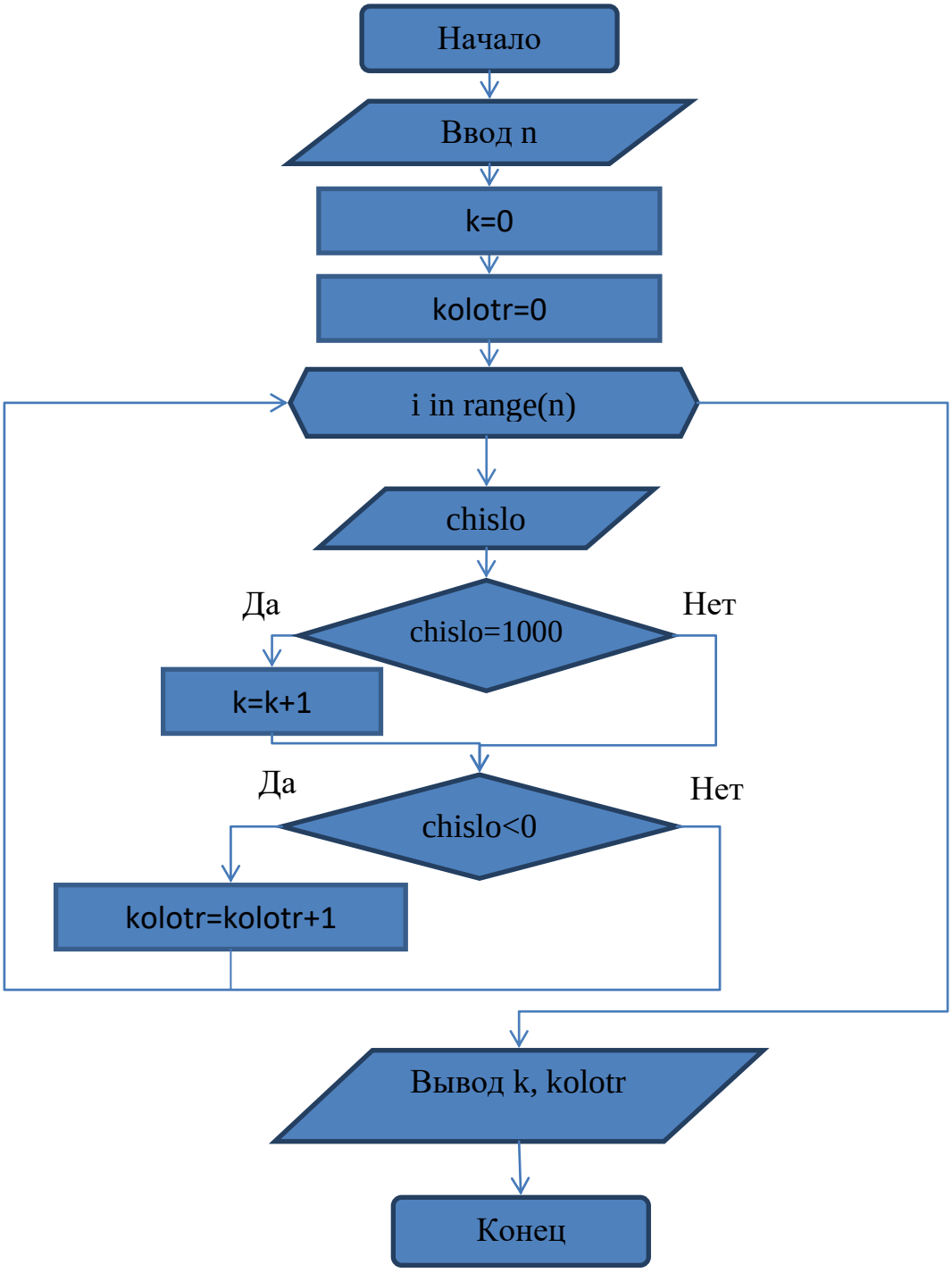

Рисунок 53 – Блок-схема алгоритма решения задачи 5.4.7

В листинге ниже приведен код программы, отвечающий за решение задачи:

```
n=int(input("Ведите количество чисел N = "))
k=0
kolotr=0
for i in range(n):
    chislo=int(input("Введите число = "))
    if chislo==100:
```

```
        k=k+1
    if chislo<0:
        kolotr=kolotr+1
print("Количество чисел равных сотне = ", k)
print("Количество отрицательных чисел = ", kolotr)
```

Задача 5.4.8. Последовательно вводится десять вещественных чисел. Определите, сколько из них совпадет с первым введенным числом.

Решение. Блок-схема алгоритма решения задачи представлена на рисунке 54.

В ячейку **chislo1** заносим первое число. Все последующие введенные числа, начиная со второго, будем сравнивать в цикле с тем числом, которое находится в ячейке **chislo1**. В случае совпадения будем увеличивать на единицу значение ячейки **kol**, играющую роль счетчика.

В листинге ниже приведен код программы, отвечающий за решение задачи:

```
chislo1=float(input("Ведите первое число = "))
kol=0
for i in range(2,11):
    chislo=float(input("Введите число = "))
    if chislo==chislo1:
        kol=kol+1
if kol==0:
    print("Нет совпадений с первым числом")
else:
    print("С первым числом совпало = ", kol)
```

Задача 5.4.9. Последовательно вводится десять целых чисел. Найдите разность между максимальным и минимальным из них.

Решение. Блок-схема алгоритма решения задачи представлена на рисунке 55.

В задаче реализуются алгоритмы поиска минимального и максимального элемента, описанные выше. По окончании цикла в ячейке **maxim** будет находиться максимальный элемент, а в ячейке **minim** - минимальный элемент. Оператором **raznost=maxim-minim** находится разность чисел.

В листинге ниже приведен код программы, отвечающий за решение задачи:

```
maxim=-32768
minim=32767
for i in range(10):
```

```
        chislo=int(input("Введите число = "))
        if chislo>maxim:
            maxim=chislo
        if chislo<minim:
            minim=chislo
    raznost=maxim-minim
    print("Разность между максимальным и минимальным числами = ", raznost)
```

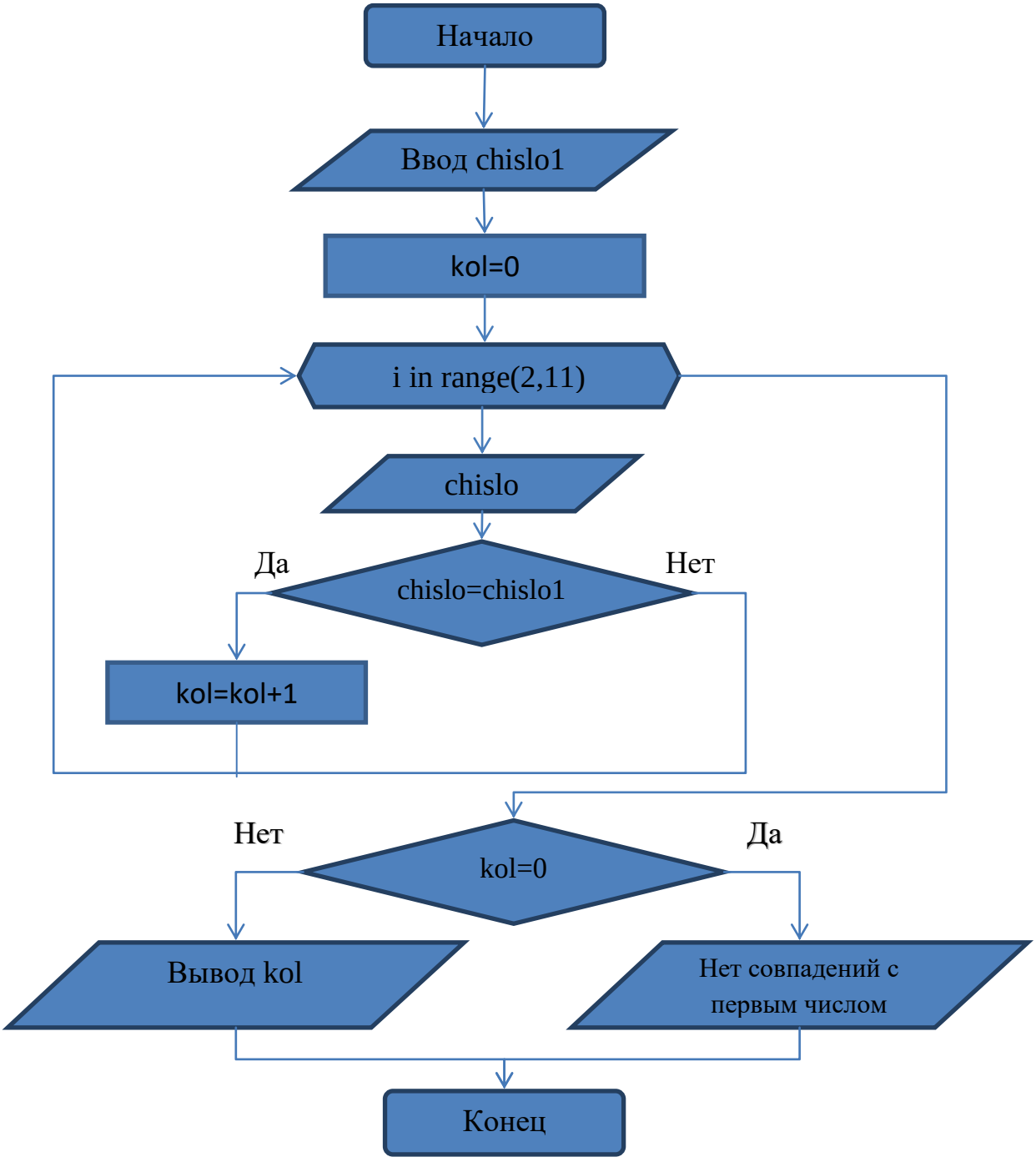

Рисунок 54 – Блок-схема алгоритма решения задачи 5.4.8

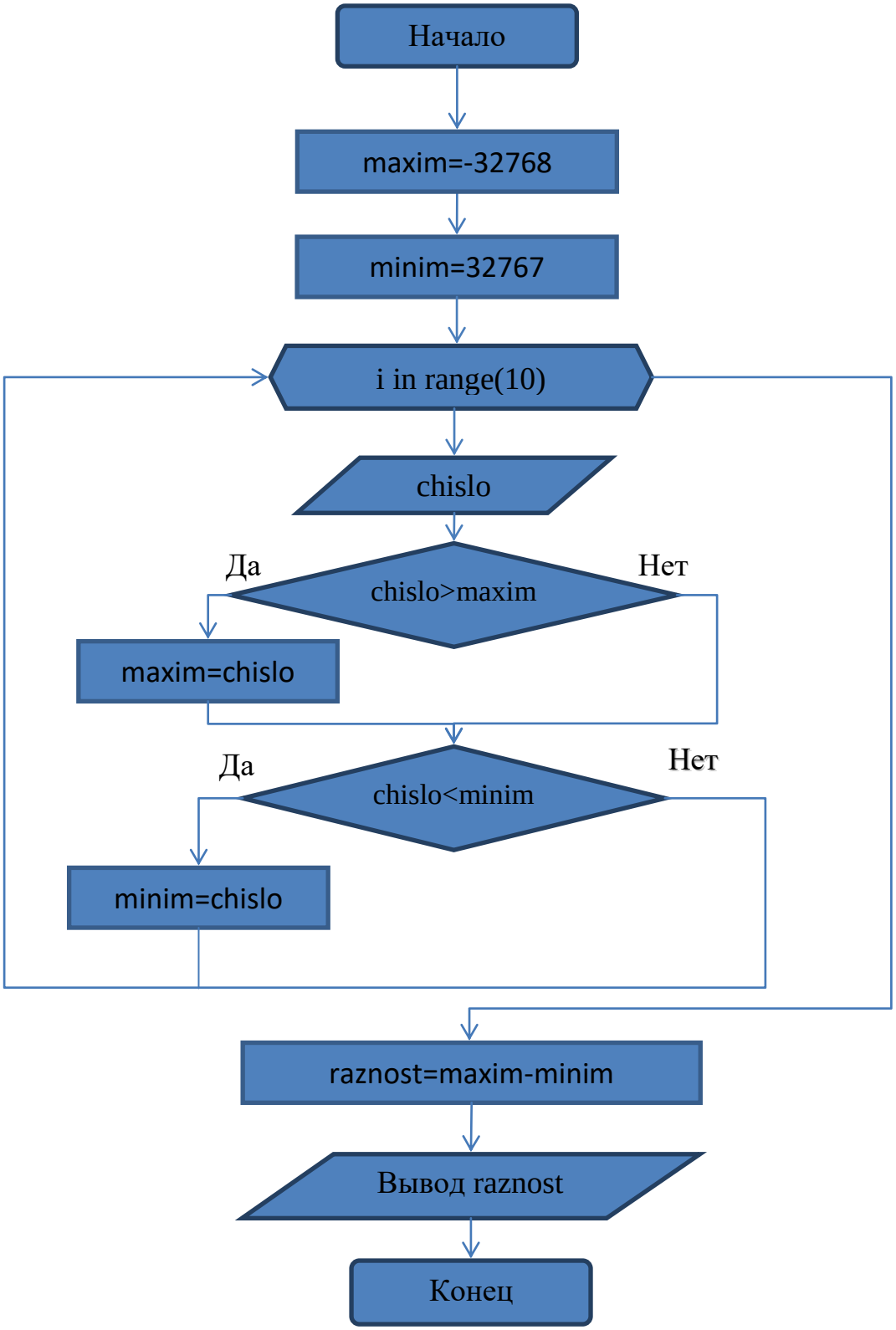

Рисунок 55 – Блок-схема алгоритма решения задачи 5.4.9

Задача 5.4.10. Последовательно вводится десять целых чисел. Найдите произведение сумм всех положительных и всех отрицательных чисел. Для этого предварительно вычислите суммы всех положительных и всех отрицательных чисел.

Решение. Блок-схема алгоритма решения задачи представлена на

рисунке 56.

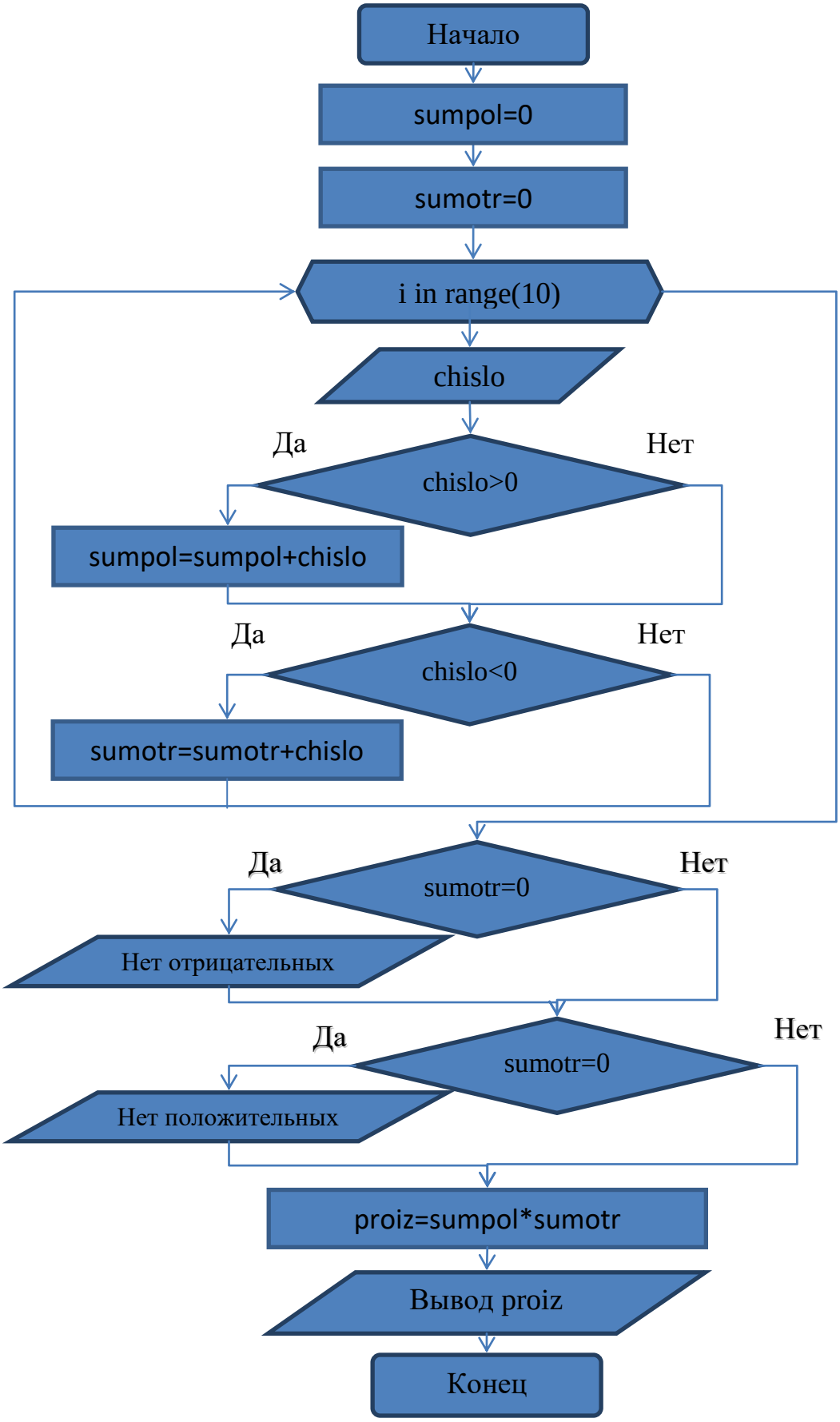

Рисунок 56 – Блок-схема алгоритма решения задачи 5.4.10

В листинге ниже приведен код программы, отвечающий за решение задачи:

```
sumpol=0
sumotr=0
for i in range(10):
   chislo=int(input("Введите число = "))
   if chislo>0:
      sumpol=sumpol+chislo
   if chislo<0:
      sumotr=sumotr+chislo
if sumpol==0:
  print("Нет положительных чисел")
if sumotr==0:
  print("Нет отрицательных чисел")
proiz=sumpol*sumotr
print("Произведение сумм всех положительных и всех отрицательных чисел = ", proiz)
```

5.5 Контрольные вопросы

1. Дайте определение циклического алгоритма.
2. Расскажите о работе оператора цикла **for** по возрастающим значениям параметра, нарисовав общий вид алгоритма и синтаксис этого оператора.
3. Расскажите о работе цикла с оператором **for** по убывающим значениям параметра цикла.
4. Расскажите о работе сложного циклического процесса, нарисовав общий вид алгоритма и синтаксис этого оператора.
5. Какой цикл называется внешним, а какой - внутренним?

5.6 Задачи для самостоятельного решения

1. Разработайте алгоритм и программу решения следующей задачи. Имеется **N** значений температур. Найдите количество отрицательных температур.
2. Разработайте алгоритм и программу решения следующей задачи. Имеется **N** значений температур. Найдите среднюю температуру.
3. Разработайте алгоритм и программу решения следующей задачи. Имеется **N** значений температур. Определите среднее значение отрицательных температур.
4. Разработайте алгоритм и программу решения следующей задачи. Имеется **N** значений температур. Определите максимальную температуру.
5. Разработайте алгоритм и программу решения следующей задачи. Имеется **N** значений температур. Найдите максимальное и минимальное

значения температуры.

6. Разработайте алгоритм и программу решения следующей задачи. Некоторая сумма денег помещена в банк под процент. Определите величину вклада в конце каждого года.