

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»  
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и  
автоматизированных систем

## **Компьютерная практика**

Выполнил: ст. группы ПВ-232  
Чернобровнеко А.Е.  
Проверил: Солонченко Р.Е.

Белгород, 2024г.

# Оглавление

<b>1</b>	<b>Задания к работе</b>	<b>3</b>
<b>2</b>	<b>Основная часть</b>	<b>5</b>
2.1	Тема 1. Линейные алгоритмы. Задание 23. . . . .	5
2.2	Тема 2. Разветвляющиеся алгоритмы. Задание 23. . . . .	7
2.3	Тема 3. Циклические и итерационные алгоритмы. Задание 23. . . . .	10
2.4	Тема 4. Простейшие операции над массивами. Задание 23. . . . .	12
2.5	Тема 5. Векторы и матрицы. Задание 23. . . . .	14
2.6	Тема 6. Линейный поиск. Задание 23. . . . .	17
2.7	Тема 7. Разветвляющиеся алгоритмы. Задание 23. . . . .	22
2.8	Тема 8. Геометрия и теория множеств. Задание 23. . . . .	27
2.9	Тема 9. Линейная алгебра и сжатие информации. Задание 23. . . . .	32
2.10	Тема 10. Алгоритмы обработки символьной информации. Задание 23. . . . .	37

# 1 Задания к работе

## Вариант 23

1. Текущее время (часы, минуты, секунды) задано тремя переменными:  $h$ ,  $m$ ,  $s$ . Округлить его до целых значений минут и часов. Например, 14 ч 21 мин 45 с преобразуется в 14 ч 22 мин или 14 ч, а 9 ч 59 мин 23 с — соответственно в 9 ч 59 мин или 10 ч.
2. Для заданного  $0 < n \leq 200$ , рассматриваемого как возраст человека, вывести фразу вида: «Мне 21 год», «Мне 32 года», «Мне 12 лет».
3. Леспромхоз ведёт заготовку деловой древесины. Первоначальный объём её на территории леспромхоза составлял  $p$  кубометров. Ежегодный прирост составляет  $k\%$ . Годовой план заготовки —  $t$  кубометров. Через сколько лет в бывшем лесу будут расти одни опять?
4. Каждый из элементов  $x_i$  массива  $X(n)$  заменить средним значением первых  $i$  элементов этого массива.
5. Многочлены  $P_n(x)$  и  $Q_m(x)$  заданы своими коэффициентами. Определить коэффициенты их композиции — многочлена  $P_n(Q_m(x))$ .
6. В массиве  $P(n)$  найти самую длинную последовательность, которая является арифметической или геометрической прогрессией.
7. Найти все натуральные числа, не превосходящие заданного  $n$ , десятичная запись которых есть строго возрастающая или строго убывающая последовательность цифр.
8. Медианой множества точек на плоскости назовём прямую, которая делит множество на два подмножества одинаковой мощности. Найти горизонтальную и вертикальную медианы заданного множества, у которого никакие две точки не лежат на одной горизонтальной или вертикальной прямой.
9. Заданный неупакованный двоичный массив сжать, используя полубайтовое представление длин цепочек.
10. По правилам пунктуации пробел может стоять после, а не перед каждым из следующих знаков:  $. , ; : ! ? ) ] \} \dots$ ; перед, а не после знаков:  $( [ \{$ . Заданный текст проверить на соблюдение этих правил и при необходимости исправить. Вместо пробела может быть перевод строки или знак табуляции.
11. Построить прямую  $3x + 2y - 4 = 0$  в диапазоне  $x \in [-1; 3]$  с шагом  $\Delta = 0,25$

12. Построить верхнюю часть эллипса  $0,1 \leq x \leq 5,1$  с шагом  $\Delta = 0,25$ , заданного уравнением  $\frac{x^2}{4} + y^2 = 1$
13. Найдите точку равновесия в заданном диапазоне с заданным шагом.

$$\begin{cases} y = \frac{2}{x} & \text{в диапазоне } 0,1 \leq x \leq 4, \text{ с шагом } \Delta = 0,1 \\ y^2 = 2x \end{cases}$$

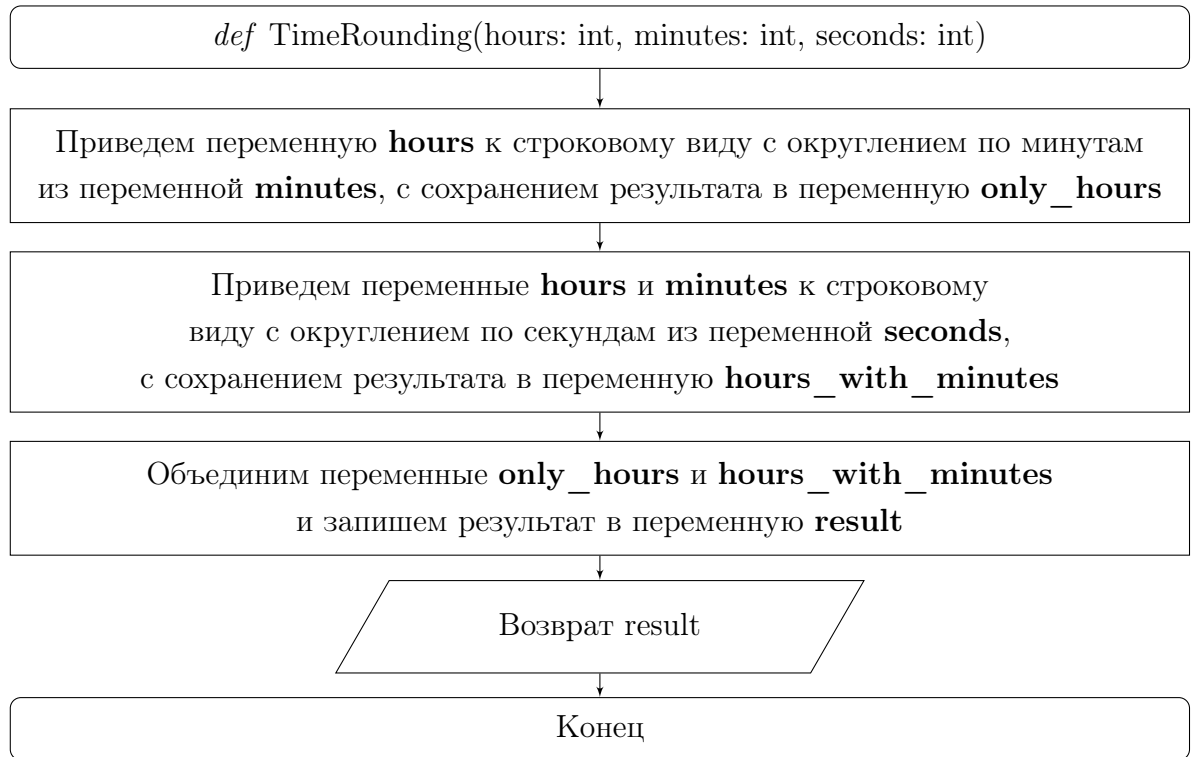
14. Построить плоскость, проходящую через точки  $M_1(3,3,1)$ ,  $M_2(2,3,2)$ ,  $M_3(1,1,3)$ , при  $-1 \leq x \leq 4$  с шагом  $\Delta = 0,5$  и  $-1 \leq y \leq 3$  с шагом  $\Delta = 1$ .
15. Построить часть параболоида, заданного уравнением  $\frac{x^2}{9} + \frac{y^2}{4} = 2z$ , лежащую в диапазоне  $-3 \leq x \leq 3$ ,  $-2 \leq y \leq 2$  с шагом  $\Delta = 0,5$  для обеих переменных.

## 2 Основная часть

### 2.1 Тема 1. Линейные алгоритмы. Задание 23.

1. Текущее время (часы, минуты, секунды) задано тремя переменными:  $h$ ,  $m$ ,  $s$ . Округлить его до целых значений минут и часов. Например, 14 ч 21 мин 45 с преобразуется в 14 ч 22 мин или 14 ч, а 9 ч 59 мин 23 с — соответственно в 9 ч 59 мин или 10 ч.
2. Словесное описание *алгоритма*:
  - (a) Знаем, что в одном часе 60 минут, а в одной минуте 60 секунд. Будем придерживаться принципа округления - если количество секунд  $< 50\%$ , тогда округляем количество минут в меньшую сторону, иначе - в большую. С минутами в часах будем действовать по тому же принципу.
  - (b) Исходя из вышесказанного, если количество секунд  $< 30$ , тогда округляем минуты в меньшую сторону, иначе - в большую. С минутами в часах действуем по тому же принципу.
  - (c) Для решения сначала запишем результаты округлений в соответствующие переменные, а потом объединим их в общую.
3. Спецификация функции *TimeRounding*:
  - (a) Заголовок: `def TimeRounding(hours: int, minutes: int, seconds: int) -> str:`
  - (b) Назначение: используется для нахождения целого числа минут и часов или только часов по введенному времени.

Блок-схема:



4. Код алгоритма на языке *Python*:

```

1 def TimeRounding(hours: int, minutes: int, seconds: int) -> str:
2     only_hours: str = f'{hours + 1 if minutes >= 30 else hours} ч'
3     hours_with_minutes: str = \
4         f'{hours} ч {minutes + 1 if seconds >= 30 else minutes} м'
5     result: str = f'{hours_with_minutes} или {only_hours}'
6
7     return result
  
```

5.

Таблица 1

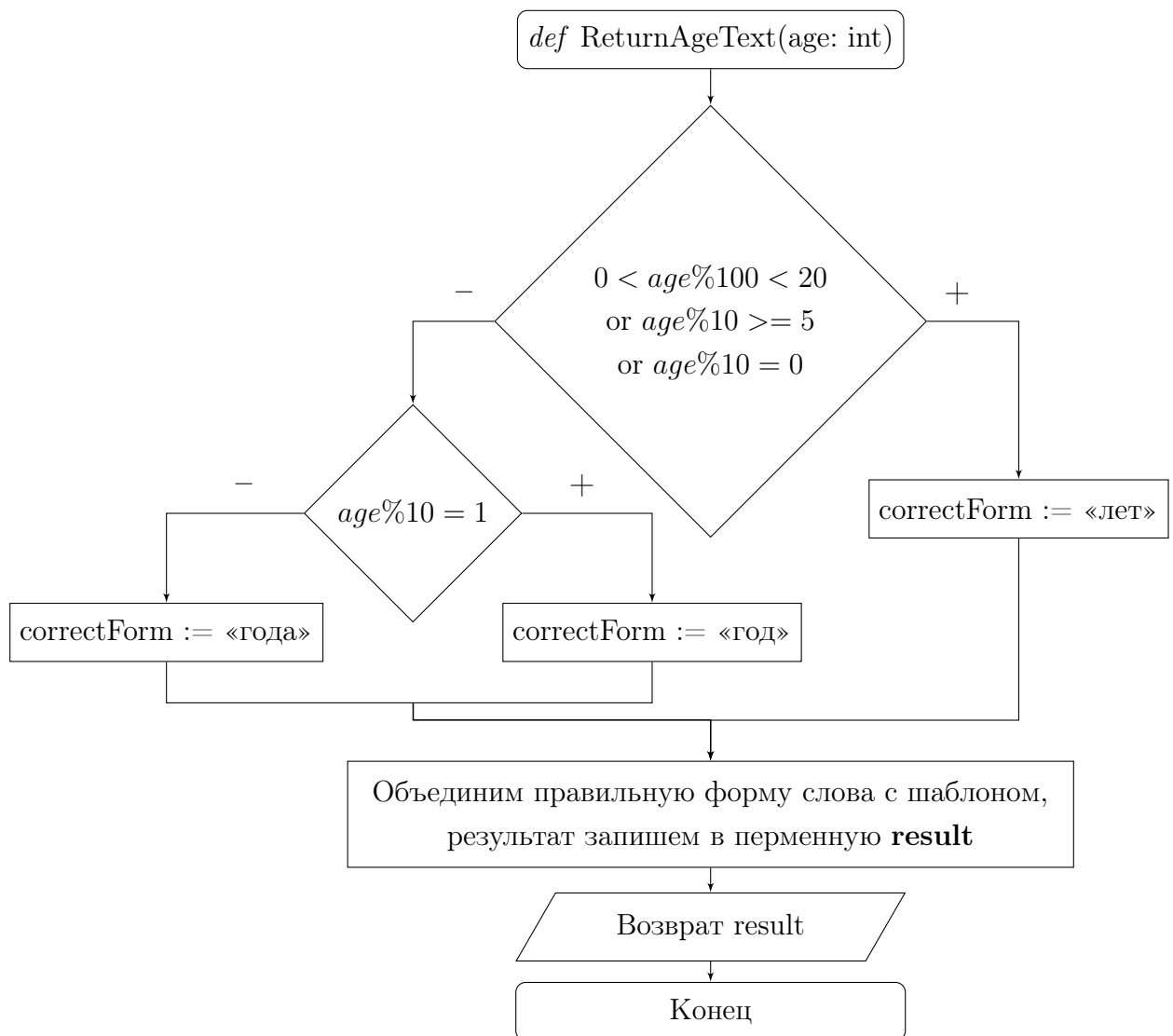
### Тестовые данные

Входные данные	Выходные данные
14 21 45	14 ч 22 м или 14 ч
9 59 23	9 ч 59 м или 10 ч

## 2.2 Тема 2. Разветвляющиеся алгоритмы. Задание 23.

1. Для заданного  $0 < n \leq 200$ , рассматриваемого как возраст человека, вывести фразу вида: «Мне 21 год», «Мне 32 года», «Мне 12 лет».
2. Словесное описание *алгоритма*:
  - (a) Если  $n \% 10 = 0$ ,  $n \% 10 \geq 5$  или  $10 \leq n \leq 20$  - мы пишем «лет». Если  $n$  оканчивается на 1, при этом  $n \neq 11$ , тогда мы пишем «год». В остальных случаях мы пишем «года».
  - (b) Исходя из вышесказанного, запишем условия для получения правильной формы слова.
  - (c) Получив правильную форму слова, объединим ее с шаблоном предложения.
3. Спецификация функции *ReturnAgeText*:
  - (a) Заголовок: `def ReturnAgeText(age: int) -> str`
  - (b) Назначение: используется для нахождения правильной формы слова «лет», обозначающего возраст, в шаблон «Мне  $n$  лет».

Блок-схема:



4. Код алгоритма на языке *Python*:

```
1 def ReturnAgeText(age: int) -> str:
2     if (10 < age % 100 < 20) or (age % 10 >= 5) or (age % 10 == 0):
3         correctForm: str = 'лет'
4     elif age % 10 == 1:
5         correctForm: str = 'год'
6     else:
7         correctForm: str = 'года'
8     answer: str = f'Мне {age} {correctForm}'
9
10    return answer
```



5.

Таблица 2

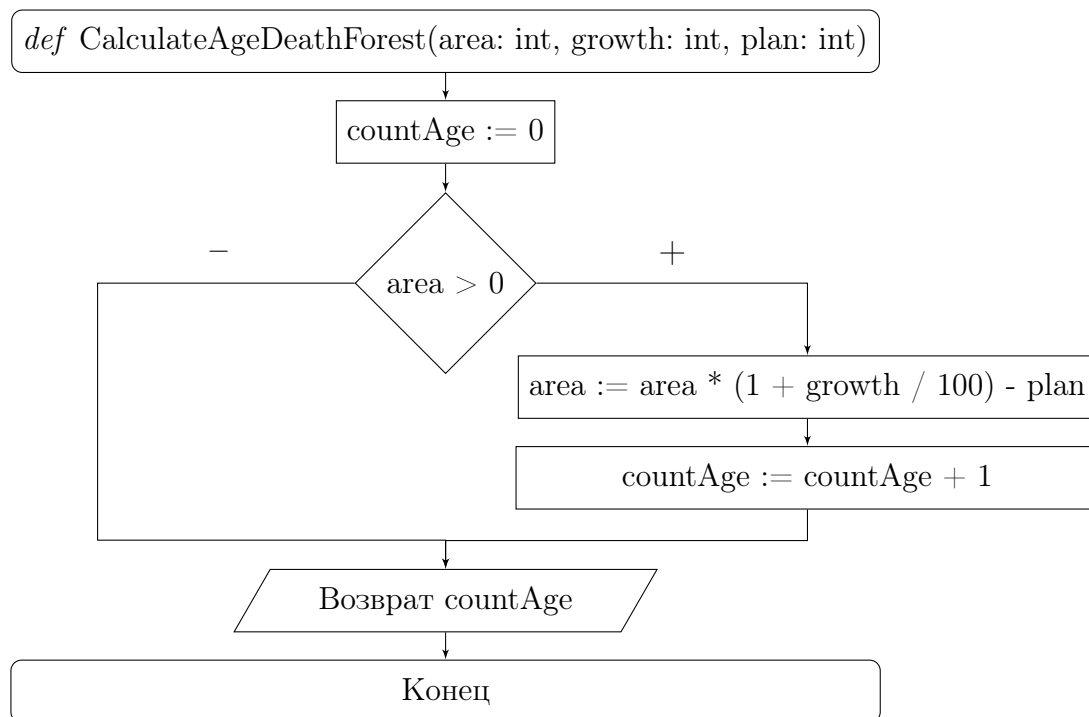
**Тестовые данные**

Входные данные	Выходные данные
11	Мне 11 лет
1	Мне 1 год
21	Мне 21 год
30	Мне 30 лет
42	Мне 42 года

## 2.3 Тема 3. Циклические и итерационные алгоритмы. Задание 23.

1. Леспромхоз ведёт заготовку деловой древесины. Первоначальный объём её на территории леспромхоза составлял  $p$  кубометров. Ежегодный прирост составляет  $k\%$ . Годовой план заготовки —  $t$  кубометров. Через сколько лет в бывшем лесу будут расти одни опята?
2. Словесное описание *алгоритма*:
  - (a) Известно, что ежемесячно объём древесины  $p$  кубометров увеличивается на  $k\%$ , при этом уменьшался на  $t$  кубометров ежегодно.
  - (b) Исходя из условия, самый простой способ узнать срок жизни леса - простой перебор, который будет считать, сколько древесины остается спустя каждый год.
  - (c) Организуем цикл, который будет отслеживать количество древесины спустя каждый год. Если количество ее кубометров будет  $= 0$ , значит срок жизни леса подошел к концу.
3. Спецификация функции *CalculateAgeDeathForest*:
  - (a) Заголовок: `def CalculateAgeDeathForest(area: int, growth: int, plan: int) -> int`
  - (b) Назначение: используется для нахождения срока жизни леса в годах.

Блок-схема:



4. Код алгоритма на языке *Python*:

```

1 def CalculateAgeDeathForest(area: int, growth: int, plan: int) -> int:
2     countAge: int = 0
3     while area > 0:
4         area = area * (1 + growth / 100) - plan
5         countAge += 1
6
7     return countAge

```

5.

Таблица 3

Тестовые данные

Входные данные	Выходные данные
10 9 4	3
10 10 4	4
1 100 2	1
0 1000 1	0

## 2.4 Тема 4. Простейшие операции над массивами. Задание 23.

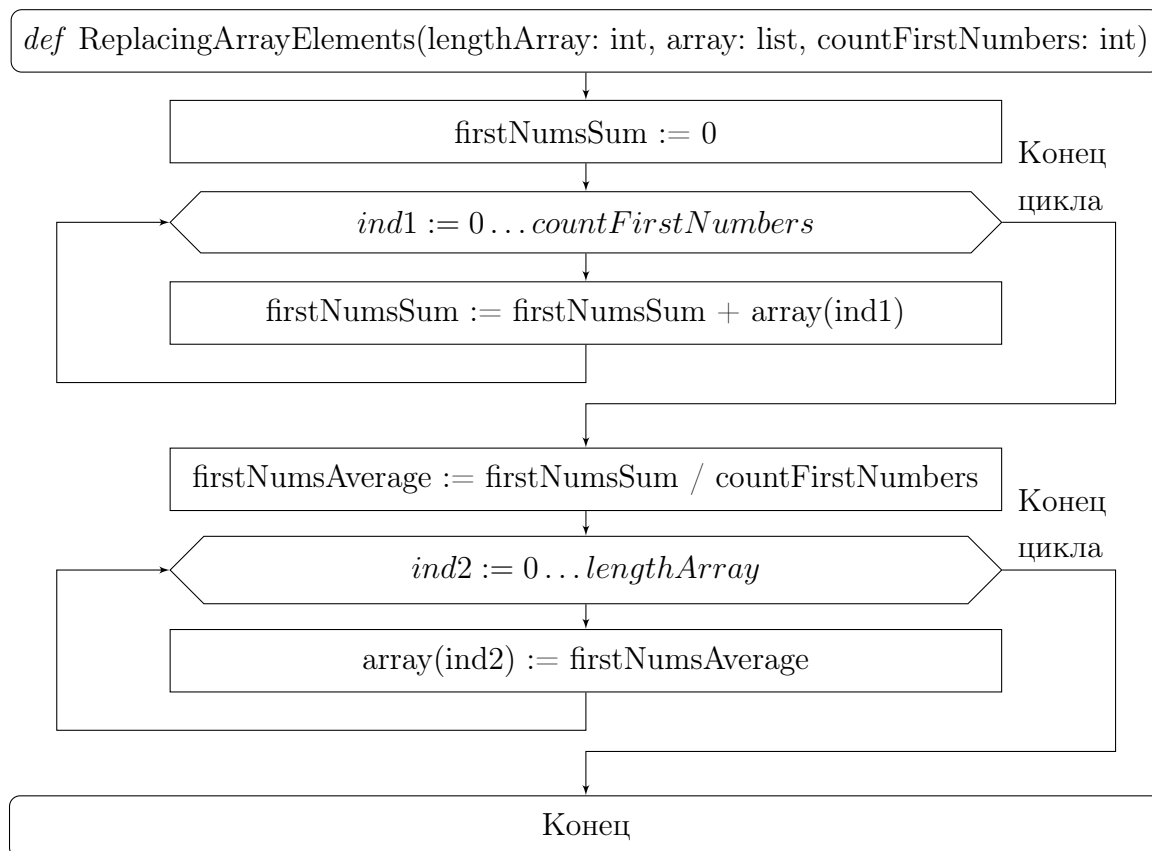
1. Каждый из элементов  $x_i$  массива  $X(n)$  заменить средним значением первых  $i$  элементов этого массива.
2. Словесное описание *алгоритма*:
  - (a) Для того, чтобы заменить каждый элемент массива средним значением первых  $i$  элементов массива, необходимо найти это среднее значение.
  - (b) Просуммируем первые  $i$  элементов массива и разделим сумму на  $i$ , чтобы получить их среднее значение.
  - (c) Имея среднее значение первых  $i$  элементов массива, заменим каждый его член на это значение.

### 3. Спецификация функции *ReplacingArrayElements*:

- (a) Заголовок: 

```
def ReplacingArrayElements(lengthArray: int, array: list, countFirstNumbers: int) -> None:
```
- (b) Назначение: используется для замены элементов массива на среднее значение его первых  $i$  элементов.

Блок-схема:



4. Код алгоритма на языке *Python*:

```

1  def ReplacingArrayElements(lengthArray: int, array: list,
2                                countFirstNumbers: int) -> None:
3      if (countFirstNumbers > lengthArray) | (countFirstNumbers == 0):
4          return
5
6      firstNumsSum: int = 0
7      for ind in range(0, countFirstNumbers):
8          firstNumsSum += array[ind]
9
10     firstNumsAverage: float = firstNumsSum / countFirstNumbers
11
12     for ind in range(0, lengthArray):
13         array[ind] = firstNumsAverage

```

5.

Таблица 4

#### Тестовые данные

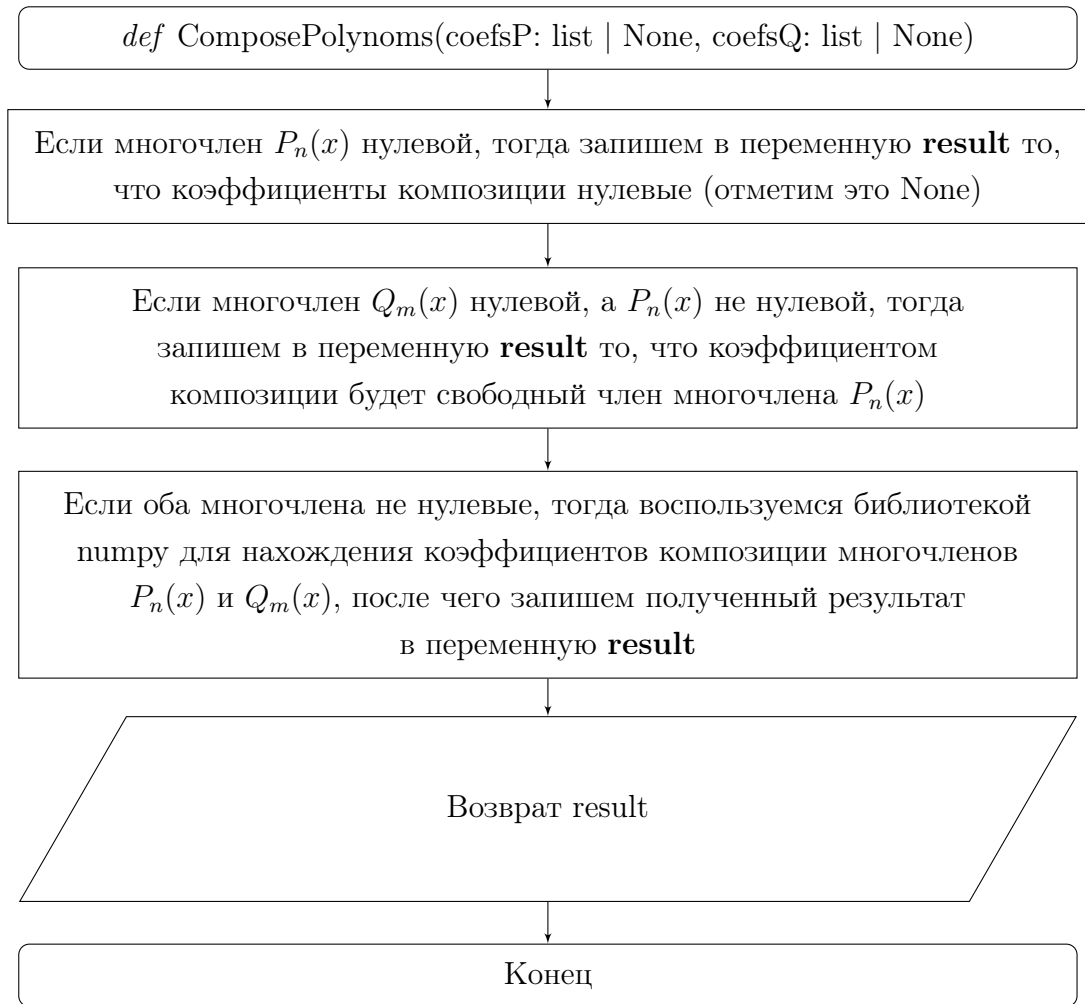
Входные данные	Выходные данные
5 1 2 3 4 5 6	1 2 3 4 5
5 1 2 3 4 5 3	2 2 2 2 2
5 1 2 3 4 5 0	1 2 3 4 5

## 2.5 Тема 5. Векторы и матрицы. Задание 23.

1. Многочлены  $P_n(x)$  и  $Q_m(x)$  заданы своими коэффициентами. Определить коэффициенты их композиции — многочлена  $P_n(Q_m(x))$ .
2. Словесное описание *алгоритма*:
  - (a) Для начала, чтобы получить коэффициенты композиции  $P_n(Q_m(x))$ , нам необходимо подставить на место  $x$  в многочлен  $P_n(x)$  многочлен  $Q_m(x)$ .
  - (b) После замены переменной  $x$  нам необходимо лишь выполнить вычисления для упрощения многочлена.
  - (c) Вышесказанного достаточно, чтобы получить композицию многочленов и выразить ее коэффициенты. Существуют два частных случая:
    1. Если многочлен  $P_n(x)$  нулевой - тогда и композиция будет нулевой.
    2. Если многочлен  $Q_m(x)$  нулевой - тогда у композиции будет один коэффициент, равный свободному члену многочлена  $P_n(x)$ .
3. Спецификация функции *ComposePolynomss*:
  - (a) Заголовок: 

```
def ComposePolynoms(coefsP: list | None,  
coefsQ: list | None) -> list | None
```
  - (b) Назначение: используется для нахождения коэффициентов композиции двух многочленов.

Блок-схема:



4. Код алгоритма на языке *Python*:

```
1 def ComposePolynoms(coefsP: list | None, coefsQ: list | None) -> list | None:
2     if coefsP is None:
3         result: None = None
4     elif coefsQ is None:
5         result: list = [coefsP[-1]]
6     else:
7         PolynomP = np.poly1d(coefsP)
8         PolynomQ = np.poly1d(coefsQ)
9         composedPolynom = PolynomP(PolynomQ)
10        result: list = composedPolynom.coefficients.tolist()
11
12    return result
```

5.

Таблица 5

### Тестовые данные

Входные данные	Выходные данные
2 0 -5 1 -1 2	2 -4 10 -8 3
1 2 2 -5	2 -3
1 2 0 11	1
None 2 0 11	None
2 0 11 None	11

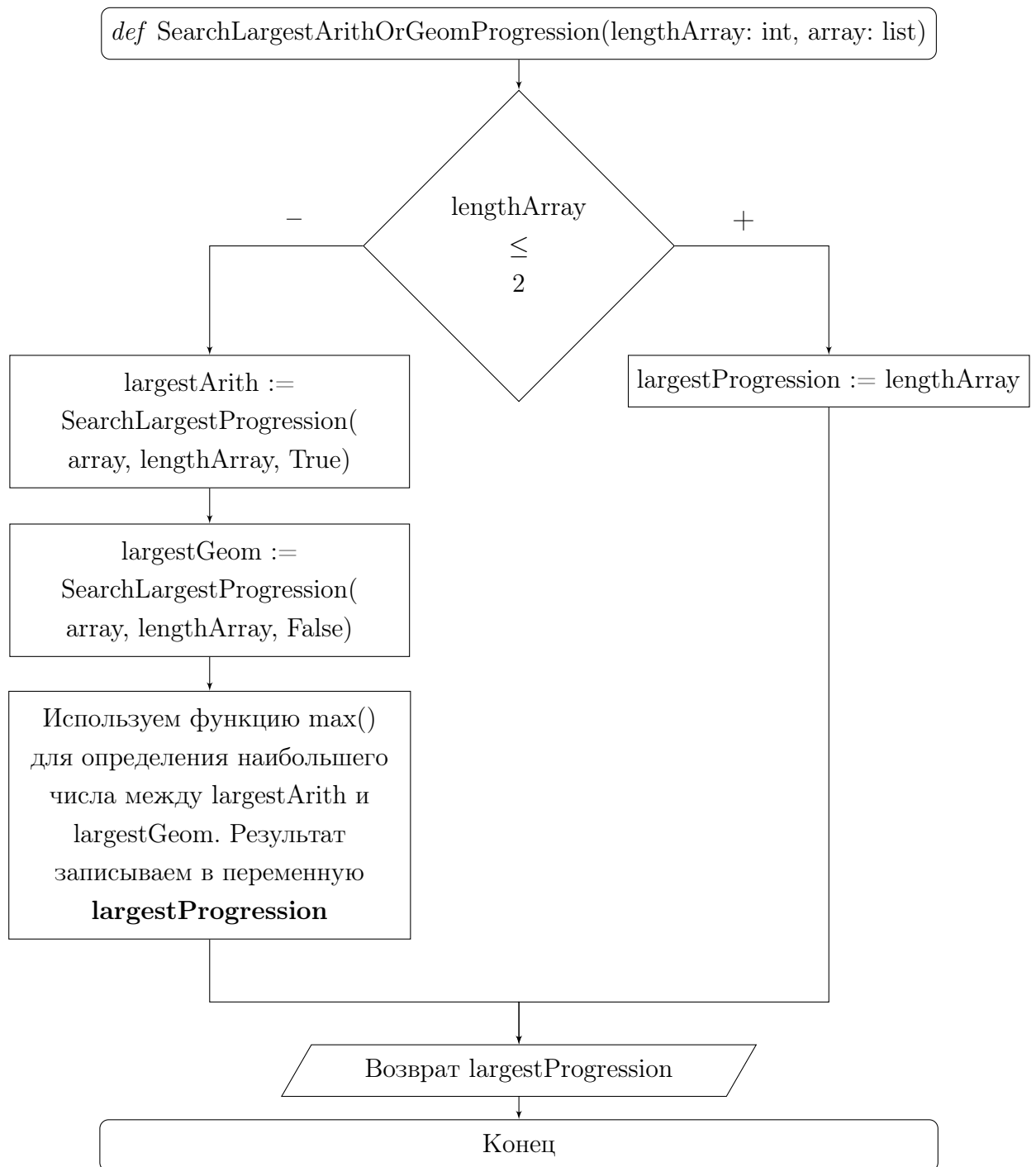


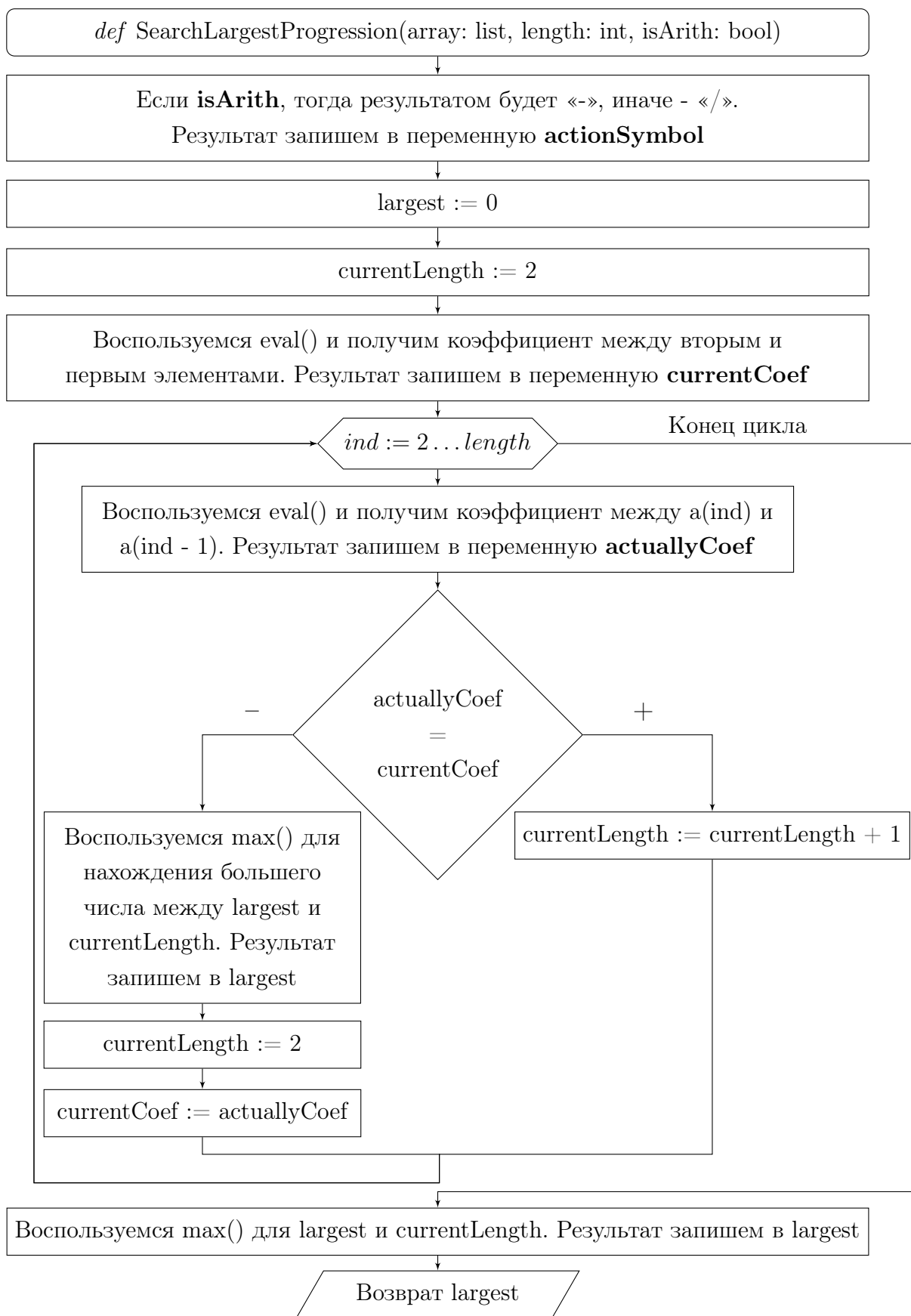
## 2.6 Тема 6. Линейный поиск. Задание 23.

1. В массиве  $P(n)$  найти самую длинную последовательность, которая является арифметической или геометрической прогрессией.
2. Словесное описание *алгоритма*:
  - (a) Так как любые два числа образуют прогрессию, получаем коэффициент для поиска прогрессии от них.
  - (b) Идем по элементам до тех пор, пока соблюдаются условия прогрессии.
  - (c) Когда прогрессия прерывается, берем ее последние два числа, получаем новый коэффициент и ищем новую прогрессию.
3. Спецификация функции *SearchLargestArithOrGeomProgression*:
  - (a) Заголовок: 

```
def SearchLargestArithOrGeomProgression(lengthArray: int,  
array: list) -> int
```
  - (b) Назначение: используется для нахождения наибольшей длины алгоритмической или геометрической прогрессии в массиве  $P(n)$ .

Блок-схемы:





4. Код алгоритма на языке *Python*:

```
1 def SearchLargestArithOrGeomProgression(lengthArray: int, array: list) -> int:
2     if lengthArray <= 2:
3         largestProgression: int = lengthArray
4     else:
5         largestArith: int = SearchLargestProgression(
6             array, lengthArray, True)
7         largestGeom: int = SearchLargestProgression(
8             array, lengthArray, False)
9         largestProgression: int = max(largestArith, largestGeom)
10
11     return largestProgression
12
13
14 def SearchLargestProgression(array: list, length: int, isArith: bool) -> int:
15     actionSymbol: str = '-' if isArith else '/'
16     largest: int = 0
17     currentLength: int = 2
18     currentCoef: float = eval(f'{array[1]} {actionSymbol} {array[0]}')
19     for ind in range(2, length):
20         actuallyCoef = eval(f'{array[ind]} {actionSymbol} {array[ind - 1]}')
21
22         if actuallyCoef == currentCoef:
23             currentLength += 1
24         else:
25             largest = max(largest, currentLength)
26             currentLength = 2
27             currentCoef = actuallyCoef
28
29     largest = max(largest, currentLength)
30
31     return largest
```

5.

Таблица 6

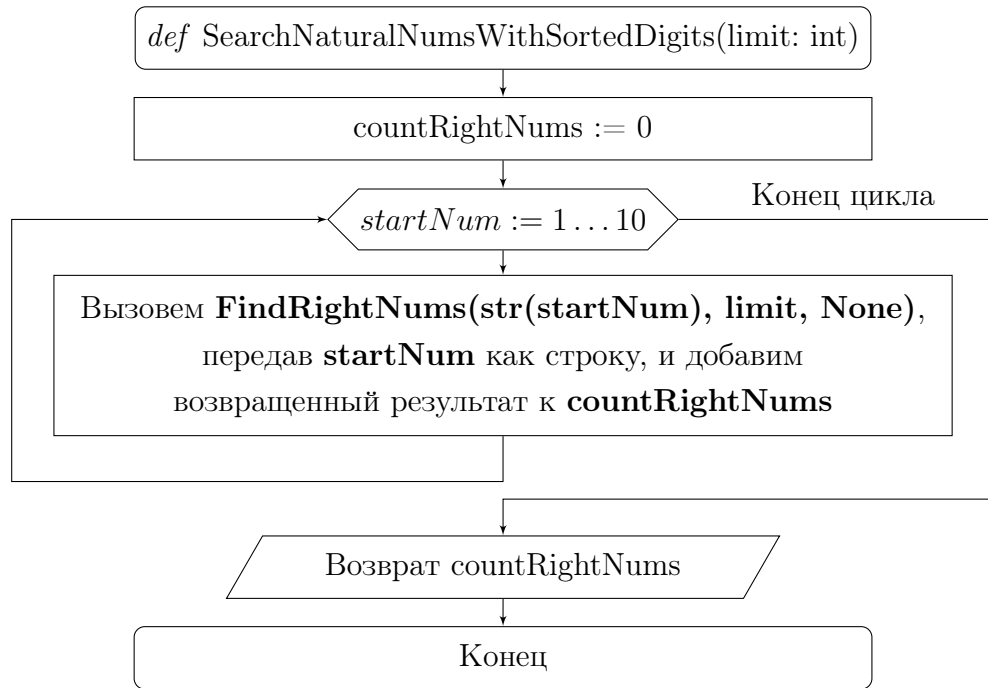
## Тестовые данные

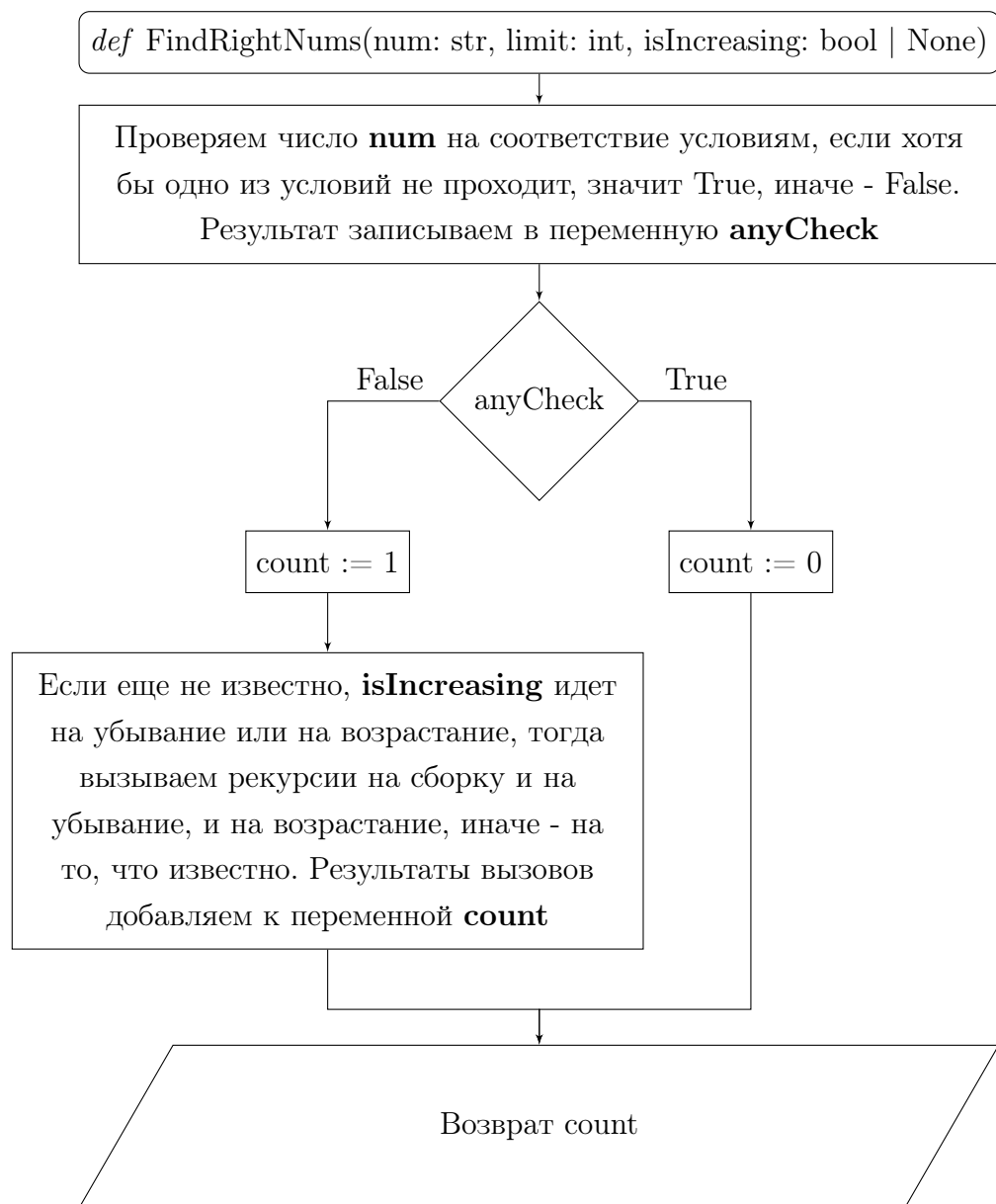
Входные данные	Выходные данные
5 1 3 5 7 9	5
5 2 1 3 5 9	3
7 2 4 1 3 9 27 7	4
4 3 4 11 2	2
5 1 2 4 8 16	5
2 1 12	2
6 8 9 10 12 14 16	4

## 2.7 Тема 7. Разветвляющиеся алгоритмы. Задание 23.

1. Найти все натуральные числа, не превосходящие заданного  $n$ , десятичная запись которых есть строго возрастающая или строго убывающая последовательность цифр.
2. Словесное описание *алгоритма*:
  - (a) Идея решения заключается в том, чтобы рекурсивно составлять числа, добавляя к существующему цифры, и проверять их на соответствие условиям задачи.
  - (b) Для реализации озвученной выше идеи создадим цикл для старта составления чисел с определенной цифры.
  - (c) Определенную цифру подаем в рекурсивную функцию, которая проверяет, не превышает ли оно заданного и является ли его десятичная запись строго возрастающей или строго убывающей последовательностью цифр.
  - (d) Соответственно, если число будет подходить - тогда добавляем его к общей сумме таких чисел и продолжаем его составление дальше, в противном случае - заканчиваем эту рекурсивную ветвь.
  - (e) В конце нужно будет лишь просуммировать количество подходящих чисел от каждой начальной цифры и вернуть результат пользователю.
3. Спецификация функции *SearchNaturalNumsWithSortedDigits*:
  - (a) Заголовок: `def SearchNaturalNumsWithSortedDigits(limit: int) -> int`
  - (b) Назначение: используется для нахождения всех натуральных чисел, не превосходящих  $n$ , десятичная запись которых - строго возрастающая или строго убывающая последовательность цифр.

Блок-схемы:







#### 4. Код алгоритма на языке Python:

```
1 def SearchNaturalNumsWithSortedDigits(limit: int) -> int:
2     countRightNums: int = 0
3     for startNum in range(1, 10):
4         countRightNums += FindRightNums(str(startNum), limit, None)
5
6     return countRightNums
7
8
9 def FindRightNums(num: str, limit: int, isIncreasing: bool | None) -> int:
10     numGreaterLimit: bool = (int(num) > limit)
11     numIsStrictlyIncreasing: bool = (
12         len(num) > 1 and int(num[-1]) <= int(num[-2])) if isIncreasing \
13         else len(num) > 1 and int(num[-1]) >= int(num[-2])
14     numIsBig: bool = len(num) > 10
15     anyCheck: bool = numGreaterLimit or numIsStrictlyIncreasing or numIsBig
16
17     if anyCheck:
18         count: int = 0
19
20     else:
21         count: int = 1
22         if isIncreasing or isIncreasing is None:
23             for nextDigit in range(int(num[-1]), 10):
24                 count += FindRightNums(num + str(nextDigit), limit, True)
25         if not isIncreasing or isIncreasing is None:
26             for nextDigit in range(0, int(num[-1])):
27                 count += FindRightNums(num + str(nextDigit), limit, False)
28
29     return count
```

5.

Таблица 7

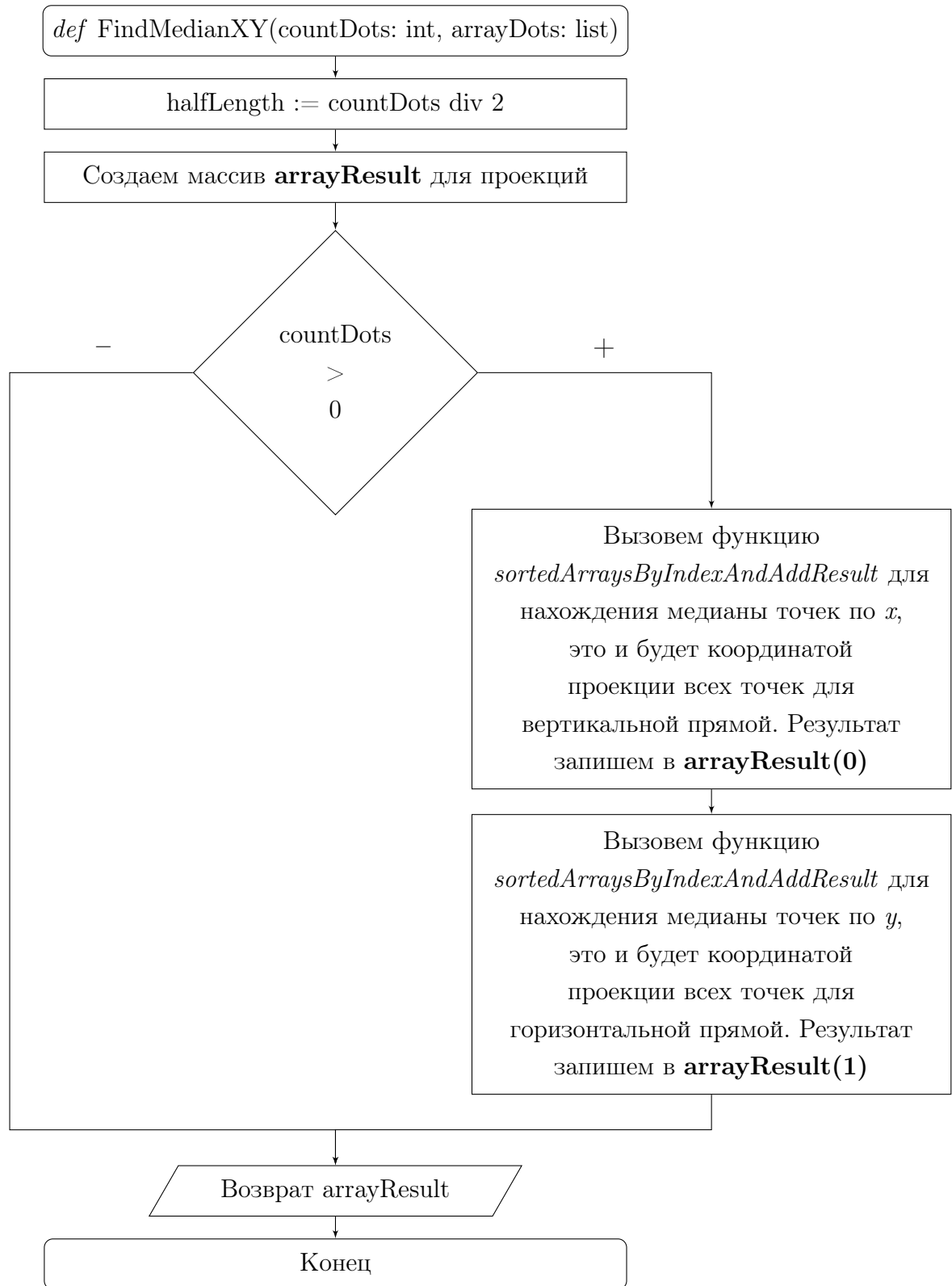
**Тестовые данные**

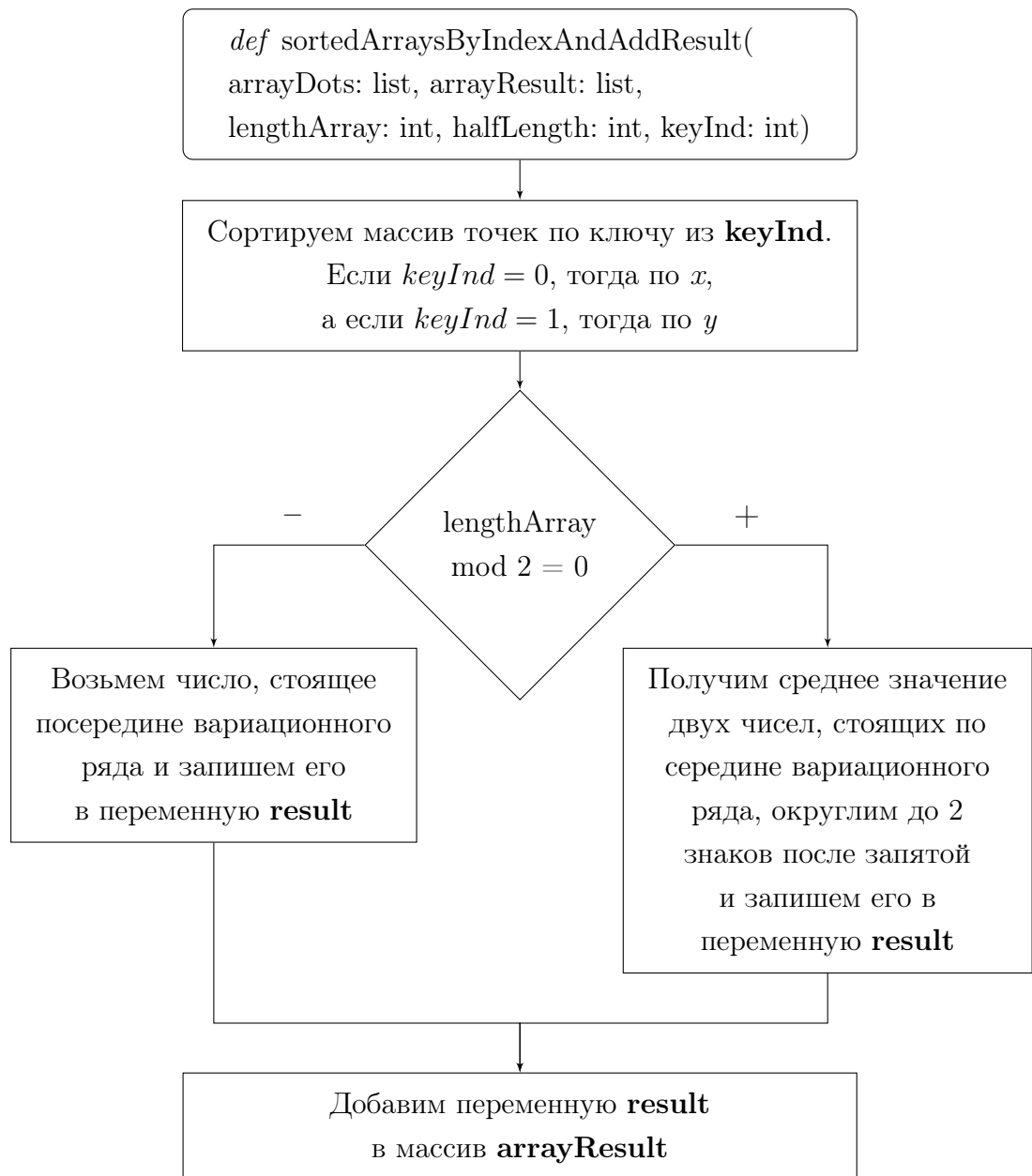
Входные данные	Выходные данные
50	46
100	90
500	174
1000	294
10000000	1458

## 2.8 Тема 8. Геометрия и теория множеств. Задание 23.

1. Медианой множества точек на плоскости назовём прямую, которая делит множество на два подмножества одинаковой мощности. Найти горизонтальную и вертикальную медианы заданного множества, у которого никакие две точки не лежат на одной горизонтальной или вертикальной прямой.
2. Словесное описание *алгоритма*:
  - (a) Для решения задачи необходимо найти горизонтальную и вертикальную медианы, соответствующие условию. Для этого отсортируем точки сначала по  $x$ , затем - по  $y$ , взяв медианы из отсортированных точек.
  - (b) Полученные медианы будут являться координатами проекции всех точек для построения медиан по условию.
3. Спецификация функции *FindMedianXY*:
  - (a) Заголовок: `def FindMedianXY(countDots: int, arrayDots: list) -> list`
  - (b) Назначение: используется для нахождения координат проекции всех точек прямой, являющихся медианами, которые делят вертикально и горизонтально множество точек на два равных по мощности подмножества.

Блок-схемы:





4. Код алгоритма на языке *Python*:

```
1 def FindMedianXY(countDots: int, arrayDots: list) -> list:
2     halfLength: int = countDots // 2
3     arrayResult: list = []
4
5     if countDots > 0:
6         # вертикальная
7         sortedArraysByIndexAndAddResult(arrayDots, arrayResult, countDots,
8                                         halfLength, 0)
9
10        # горизонтальная
11        sortedArraysByIndexAndAddResult(arrayDots, arrayResult, countDots,
12                                        halfLength, 1)
13
14    return arrayResult
15
16
17 def sortedArraysByIndexAndAddResult(arrayDots: list, arrayResult: list,
18                                     lengthArray: int, halfLength: int,
19                                     keyInd: int) -> None:
20     arrayDots.sort(key=lambda x: x[keyInd])
21     if lengthArray % 2 == 0:
22         result = round((arrayDots[halfLength][keyInd] +
23                         arrayDots[halfLength - 1][keyInd]) / 2, 2)
24     else:
25         result = arrayDots[halfLength][keyInd]
26
27     arrayResult.append(result)
```

## Тестовые данные

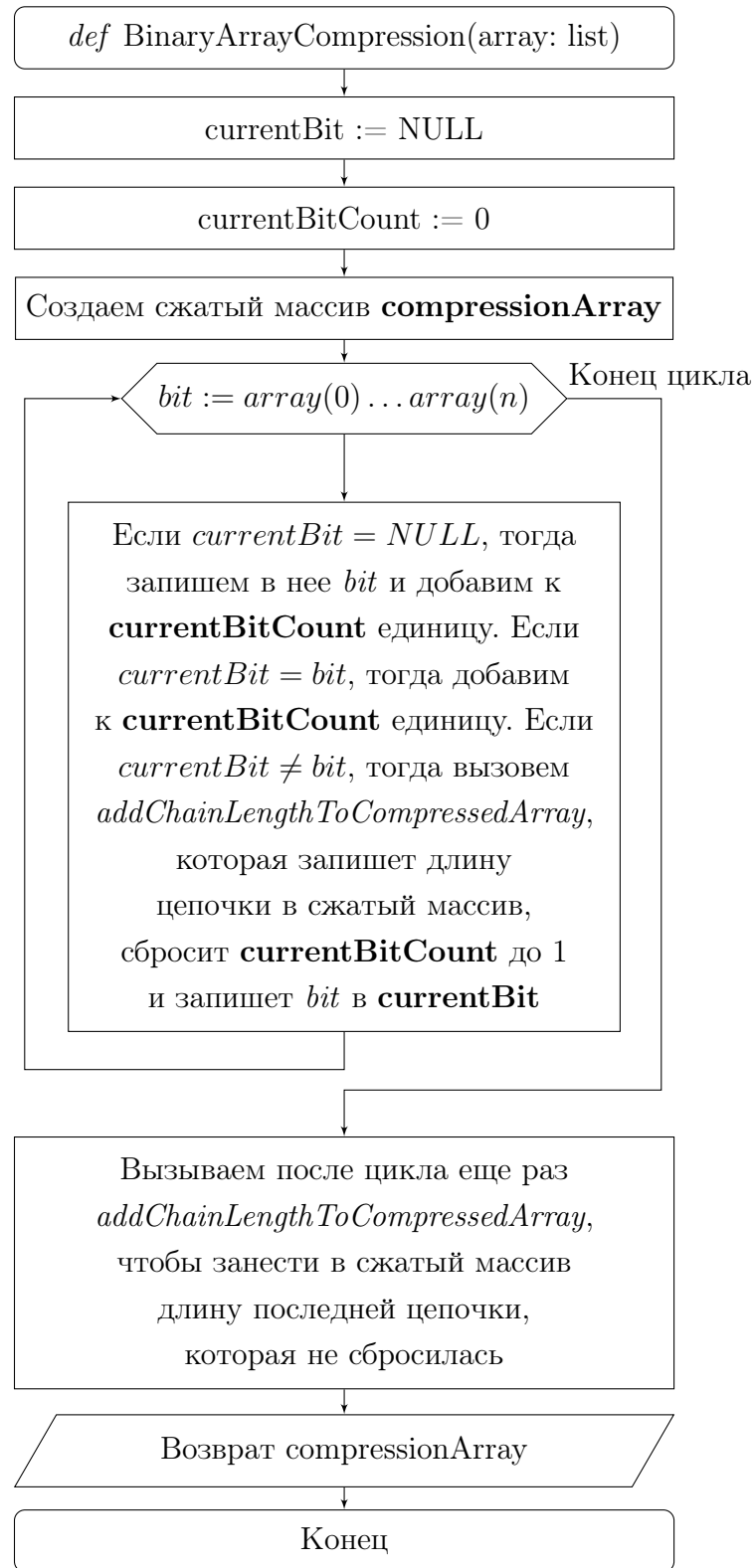
Входные данные	Выходные данные
4 3 4 1 2 7 8 5 6	4.00 5.00
5 4 6 1 7 10 20 11 13 18 26	10 13
1 10 20	10 20
4 4.5 5.4 1.1 7.2 11.3 5.4 12.3 7.7	7.90 6.30

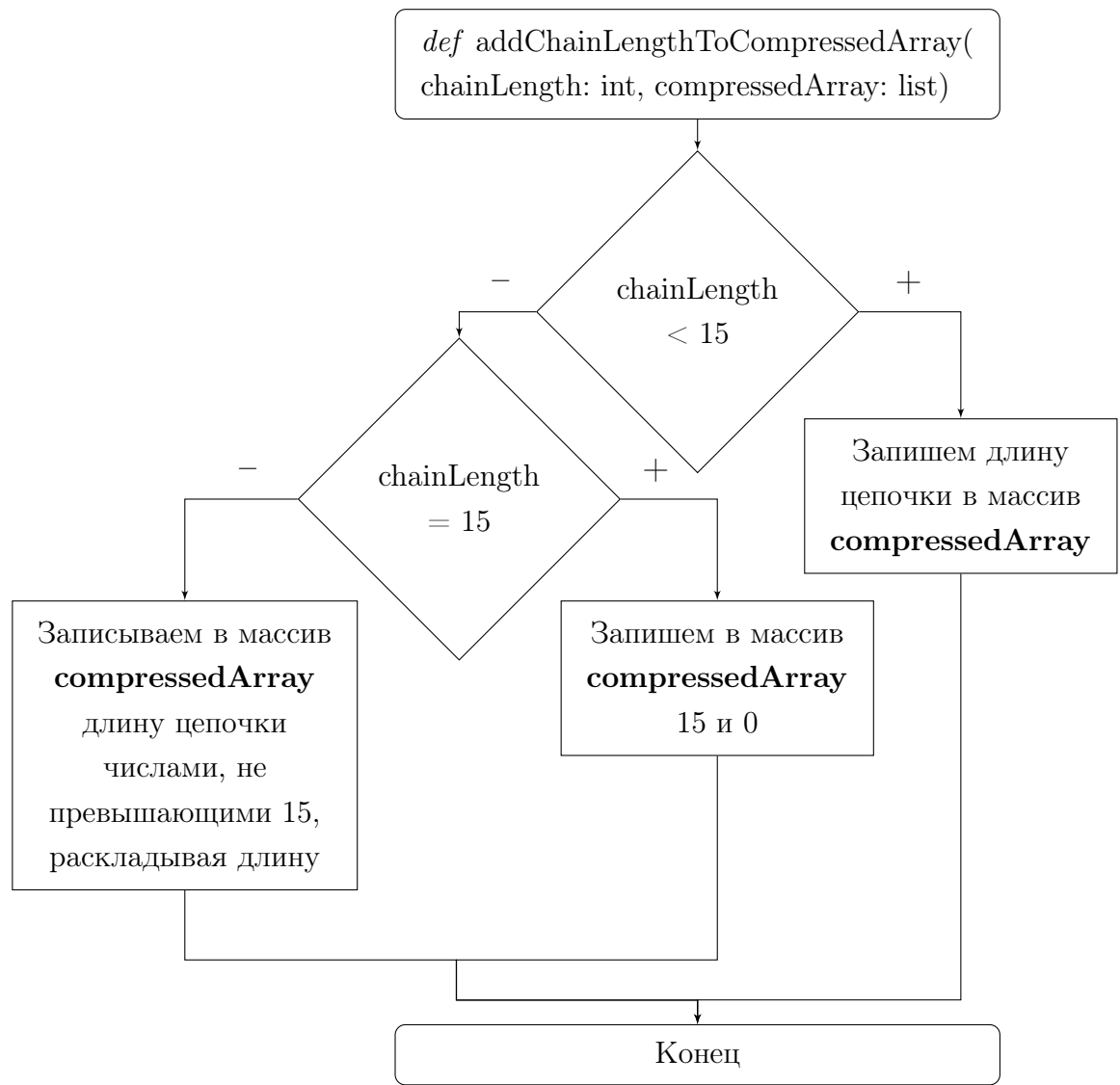
## 2.9 Тема 9. Линейная алгебра и сжатие информации. Задание 23.

1. Заданный неупакованный двоичный массив сжать, используя полубайтовое представление длин цепочек.
2. Словесное описание *алгоритма*:
  - (a) Задача довольно непростая, так как подобные способы сжатия практически не используются. Разработаем для этого свой алгоритм.
  - (b) Будем отталкиваться, что полбайта - это 4 бит. Максимальное число - 15.
  - (c) Считаем, что массив начинается с 1, то есть нет незначащих нулей.
  - (d) Будем хранить в сжатом виде лишь количество подряд идущих бит, то есть цепочки.
  - (e) Придерживаться будем следующих правил:
    1. Если длина цепочки меньше 15, тогда записываем фактическую длину цепочки.
    2. Если длина цепочки равна 15, тогда запишем в сжатый массив два числа - 15 и 0. 0 будет признаком того, что длина цепочки кратна 15. Дальше может идти лишь противоположный бит исходного массива.
    3. Если длина цепочки больше 15, тогда делаем переносы, записывая длину числами, не превышающими 15, до полного разложения исходной длины. Если после какого-то числа 15 идет не 0, значит это число также относится к цепочке. На примере: имеем 31 подряд идущий бит 0 в исходном массиве, а затем 3 подряд идущие 1. Запишем это так: [15, 15, 1, 3]. А если в исходном массиве будет 30 подряд идущих бит 0, а затем 3 подряд идущие 1, тогда запишем это так: [15, 15, 0, 3].
3. Спецификация функции *BinaryArrayCompression*:
  - (a) Заголовок: `def BinaryArrayCompression(array: list) -> list`
  - (b) Назначение: используется для сжатия исходного неупакованного двоичного массива, применяя способ полубайтового представления длин цепочек.



Блок-схемы:





4. Код алгоритма на языке *Python*:

```
1 def BinaryArrayCompression(array: list) -> list:
2     currentBit: int | None = None
3     currentBitCount: int = 0
4     compressionArray: list = []
5
6     for bit in array:
7         if currentBit is None:
8             currentBit = bit
9             currentBitCount += 1
10        elif bit == currentBit:
11            currentBitCount += 1
12        else:
13            addChainLengthToCompressedArray(currentBitCount, compressionArray)
14            currentBitCount = 1
15            currentBit = bit
16
17    if currentBitCount > 0:
18        addChainLengthToCompressedArray(currentBitCount, compressionArray)
19
20    return compressionArray
21
22
23 def addChainLengthToCompressedArray(chainLength: int,
24                                     compressedArray: list) -> None:
25     if chainLength < 15:
26         compressedArray.append(chainLength)
27     elif chainLength == 15:
28         compressedArray.append(chainLength)
29         compressedArray.append(0)
30     else:
31         while chainLength > 0:
32             if chainLength >= 15:
33                 numAdd = 15
34             else:
35                 numAdd = chainLength
36
37             chainLength -= numAdd
38             compressedArray.append(numAdd)
```

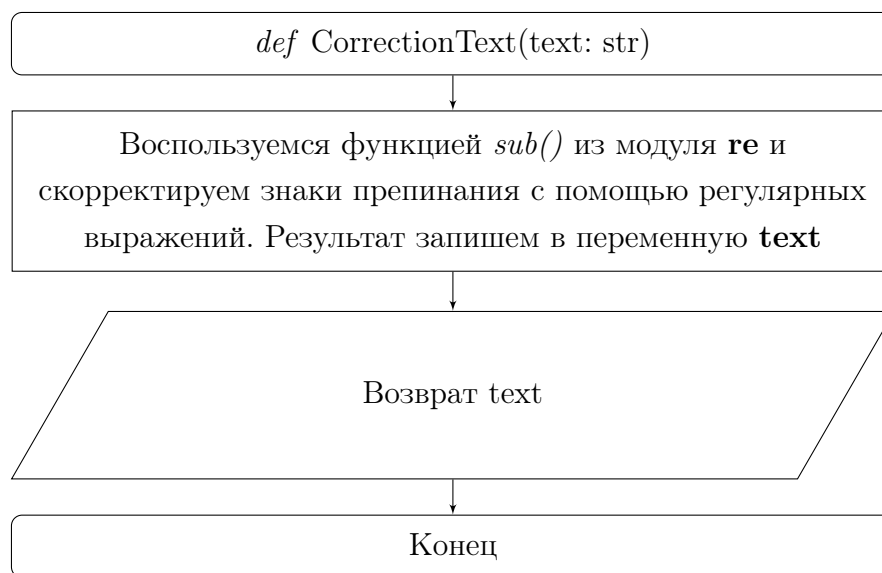
## Тестовые данные

Входные данные	Выходные данные
1 1 1 0 0 1	3 2 1
1 0 1 0 1 0	1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0	15 0 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1	15 2 1 1

## 2.10 Тема 10. Алгоритмы обработки символьной информации. Задание 23.

1. По правилам пунктуации пробел может стоять после, а не перед каждым из следующих знаков: . , ; : ! ? ) ] } ...; перед, а не после знаков: ( [ { . Заданный текст проверить на соблюдение этих правил и при необходимости исправить. Вместо пробела может быть перевод строки или знак табуляции.
2. Словесное описание *алгоритма*:
  - (a) Убираем лишние пробелы после знаков препинания: . , ; : ! ? ) ] } ...;
  - (b) Добавляем недостающие пробелы после знаков препинания: . , ; : ! ? ) ] } ...;
  - (c) Убираем пробелы после открывающих скобок: ( [ { .
  - (d) Убираем лишние пробелы перед открывающими скобками: ( [ { .
3. Спецификация функции *CorrectionText*:
  - (a) Заголовок: `def CorrectionText(text: str) -> str`
  - (b) Назначение: используется для корректировки пробелов перед знаками препинания и после них, согласно правилам пунктуации.

Блок-схема:



#### 4. Код алгоритма на языке Python:

```

1 def CorrectionText(text: str) -> str:
2     # Убираем лишние пробелы после знаков препинания .,;:!?)]}
3     text = re.sub(r'\s+([,;:!?)]\])', r'\1', text)
4
5     # Добавляем недостающие пробелы после знаков препинания .,;:!?)]}
6     text = re.sub(r'([,;:!?)]\])' + r'([^\s.,;:!?)]\])', r'\1 \2', text)
7
8     # Убираем пробелы после открывающих скобок
9     text = re.sub(r'(\(\[{\s+)', '(', text)
10    text = re.sub(r'(\[\{\s+)', '[', text)
11    text = re.sub(r'(\{\{\s+)', '{', text)
12
13    # Убираем лишние пробелы перед открывающими скобками
14    text = re.sub(r'\s+([\(\[\{])', r'\1', text)
15
16    return text

```

#### 5.

Таблица 10

#### Тестовые данные

Входные данные	Выходные данные
( Привет, Андрей ) , - говорил он	(Привет, Андрей), - говорил он
(Да \n? Ты в этом уверен ? )	(Да? Ты в этом уверен?)
( ( Проверю на скобках ) )	((Проверю на скобках))
Да , забавно (получается)	Да, забавно (получается)
( Чек ) , посмотрим	(Чек), посмотрим