

Министерство образования и науки Российской Федерации
РГУ нефти и газа (НИУ) имени И.М. Губкина

**Лабораторная работа №6 по предмету
«Языки программирования»**

Выполнил студент
группы КФ-22-02
Муртазин К.Э.
Проверил профессор
Кафедры безопасности
информационных технологий
Корнеев Н.В.

Москва, 2023

Цель работы: Выполнение циклический операций с использованием оператора while.

Задача: разрабатывается программа, которая вычисляет значение числа π с точностью, задаваемой пользователем во время работы программы. В основе алгоритма вычисления лежит тот факт, что сумма ряда приближается к значению $\pi/4$ при достаточно большом количестве членов ряда.

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots + (-1)^{n+1} * \frac{1}{(2n-1)} + \dots$$

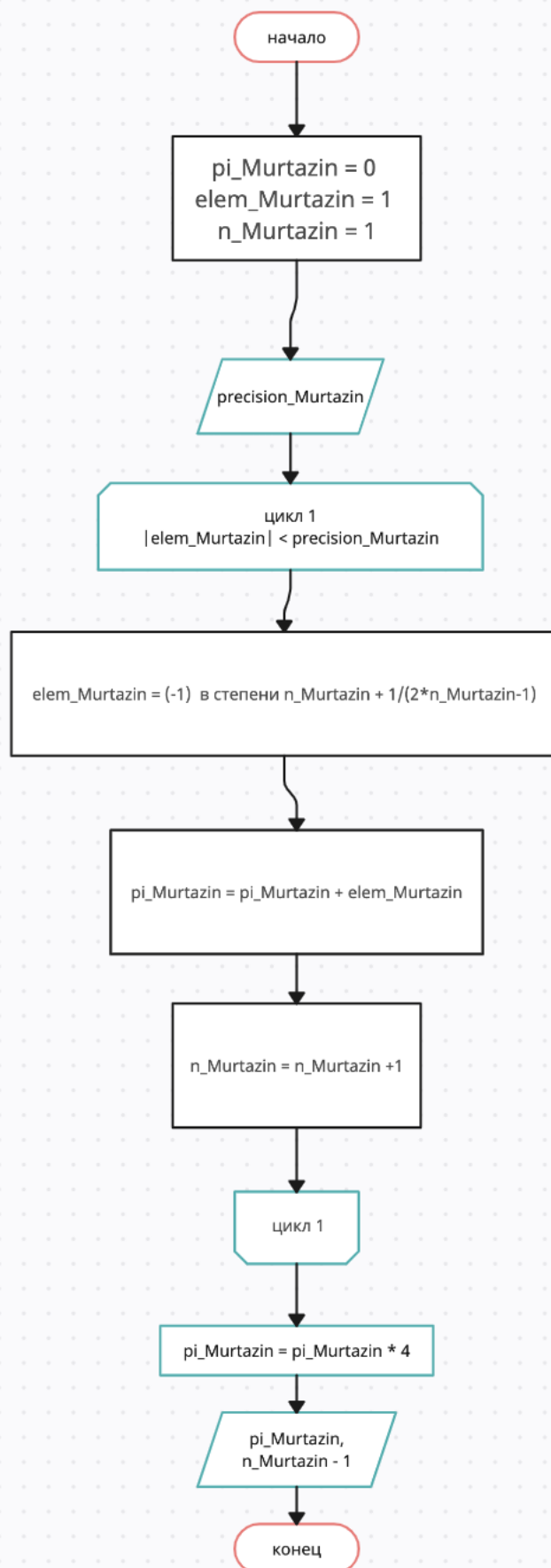
Описание алгоритма:

- 1) Создание переменных $\pi_Murtazin = 0$, $elem_Murtazin = 1$, $n_Murtazin = 1$
- 2) Пользователь вводит с клавиатуры переменную $precision_Murtazin$
- 3) Создание цикла while при условии $|elem_Murtazin| < precision_Murtazin$ для определения приближенного значения числа π и количества элементов ряда.

Действия в цикле

- 3.1) $elem_Murtazin = (-1)^{n_Murtazin} * 1/(2*n_Murtazin-1)$
- 3.2) Прибавляем к $\pi_Murtazin$ $elem_Murtazin$. И прибавляем количеству элементов ряда $n_Murtazin + 1$
- 4) Домножаем переменную $\pi_Murtazin$ на 4
- 5) Выводим $\pi_Murtazin$ и $n_Murtazin - 1$

Блок-схема:



Код с комментарием:

```
pi_Murtazin= 0 # задаем начальное значение для переменной pi_Murtazin равным 0
elem_Murtazin = 1 # задаем начальное значение для переменной elem_Murtazin равным 1
n_Murtazin = 1 # задаем начальное значение для переменной n_Murtazin равным 1
precision_Murtazin = float(input("Введите точность: ")) # присваиваем переменной precision_Murtazin значение, введенное пользователем с клавиатуры

while elem_Murtazin >= precision_Murtazin: # начало цикла while, который будет выполняться, пока elem_Murtazin больше или равен точности
    elem_Murtazin = 1/(2*n_Murtazin - 1) # вычисляем текущее значение элемента ряда, используя значение n_Murtazin
    if n_Murtazin % 2 == 0: # проверяем, является ли n_Murtazin четным числом
        pi_Murtazin -= elem_Murtazin # если n_Murtazin четное число, то вычитаем текущий элемент из значения pi_Murtazin
    else:
        pi_Murtazin += elem_Murtazin # если n_Murtazin нечетное число, то прибавляем текущий элемент к значению pi_Murtazin
    n_Murtazin += 1 # увеличиваем значение n_Murtazin на 1
pi_Murtazin = pi_Murtazin * 4 # вычисляем значение pi, умножая pi_Murtazin на 4
print('ПИ приближенно равно: ', pi_Murtazin) # выводим на экран приближенное значение числа ПИ
print('Количество членов ряда: ', n_Murtazin - 1) # выводим на экран количество членов ряда, которое было необходимо вычислить для достижения заданной точности
```

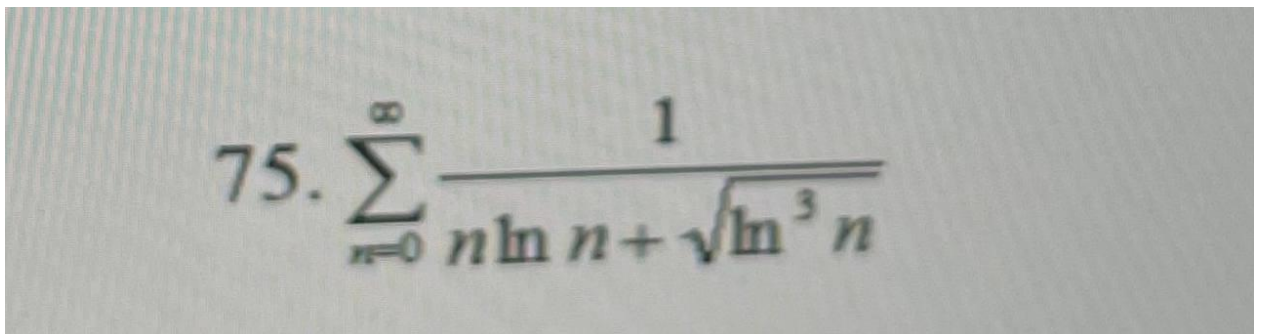
Результат:

```
Введите точность: 0.00001
ПИ приближенно равно: 3.141612653189785
Количество членов ряда: 50001
```

Вывод: я разработал программу посредством языка программирования Python для определения приближенного значения числа ПИ и количества элементов ряда с использованием цикла while и операторов ветвления if и else.

Индивидуальное задание. Вариант 75

Задача: составить программу вычисления суммы ряда, значение точности элемента ряда вводить в формате вещественного числа, с помощью оператора условия if проверить область сходимости ряда и исключить недопустимые значения, вывести на экран значение суммы ряда и количество членов ряда.

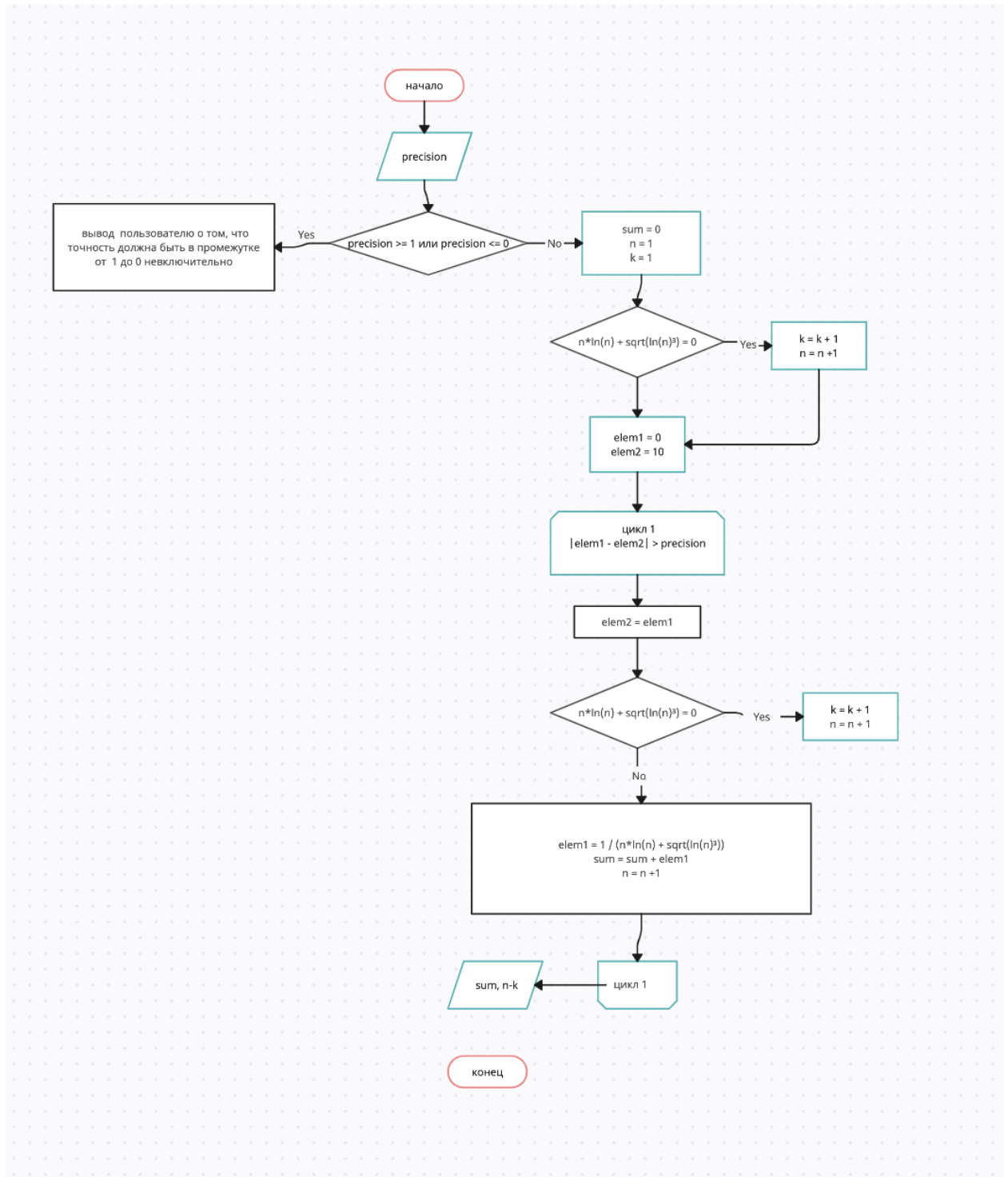

$$75. \sum_{n=0}^{\infty} \frac{1}{n \ln n + \sqrt{\ln^3 n}}$$

Описание алгоритма:

- 1) Пользователь вводит с клавиатуры переменную precision
- 2) Проверка на правильно введенную точность (правильная точность от 0 до 1 не включительно) с использованием оператора if

- 3) Создание переменной $sum = 0$, $n = 2$ (так как $n = 1$ не определён), $elem1 = 0$, $elem2 = 10$ (значение предыдущего элемента ряда в самом начале может быть любым, но таким, чтобы выполнялось условие из цикла `while`)
- 4) Условие цикла `while` - модуль разности следующих друг за другом элементов будет больше точности. Внутри цикла `while` происходит
 - 1) Присваивание $elem2 = elem1$
 - 2) вычисление значения текущего элемента ряда $elem1$ по заданной формуле.
 - 3) Затем текущий элемент ряда добавляется к общей сумме, и номер текущего элемента ряда увеличивается на 1.
- 5) После того, как достигнута необходимая точность, программа выводит sum и $n-2$.

Блок-схема:



Дополнение Стрелка между sum, n-k и концом (должна быть)

Код с комментарием:

```
import math
precision = float(input("Введите точность: ")) # ввод точности
if precision <= 0 or precision >= 1: # проверка на верно введеную точность
    print("Точность должна быть в промежутке от 0 до 1")
    precision = float(input("Введите точность: "))
sum = 0 # сумма ряда
n = 1 # элементы
k = 1 # элементы которые не учитываем
if (n * math.log(n) + math.sqrt(math.log(n)**3)) == 0: # проверка на то, что знаменатель равен 0, если равен 0, то переходим к след элементу ряда
    n += 1 # увеличени номера элемента ряда
    k += 1 # кол-во элементов ряда, которые не учитываем
elem1 = 0 # значение первого элемента ряда
elem2 = 10 # значение предыдущего элемента ряда (в начале может быть любым значением, но таким, чтобы выполнялось условие в цикле while)

while (abs(elem1 - elem2) > precision): # цикл while, который будет выполняться пока модуль разности следующих друг за другом элементов будет больше точности
    elem2 = elem1 # значение предыдущего элемента
    if (n * math.log(n) + math.sqrt(math.log(n) ** 3)) == 0: # проверка на то, что знаменатель равен 0, если равен 0, то переходим к след элементу ряда
        n += 1 # увеличени номера элемента ряда
        k += 1 # кол-во элементов ряда, которые не учитываем
    else:
        elem1 = 1/(n * math.log(n) + math.sqrt(math.log(n)**3)) # текущий элемент ряда
        sum += elem1 # добавление текущего значения элемента ряда к сумме ряда
        n += 1 # увеличение номера текущего элемента ряда
print("Значение суммы ряда: ", sum) # вывод значения суммы ряда
print("Количество членов ряда: ", n - k) # вывод количества просуммированных членов ряда, не включая те, которые не существуют
```

Результат:

```
Введите точность: 0.1
Значение суммы ряда: 0.8735057098842504
Количество членов ряда: 3
```

Вывод:

я разработал программу посредством языка программирования Python для вычисления суммы ряда с использованием цикла while