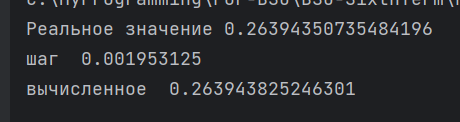


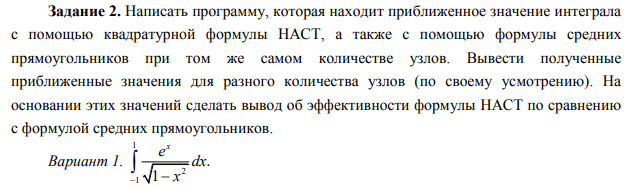
Вот листинг программы:

import math  
  
def function(x):  
 return math.log(math.pow(x,2)+1)  
  
def indefinite\_integral(x):  
 return x\*math.log(math.pow(x,2)+1)-2\*x+2\*math.atan(x)  
  
def integral(a,b):  
 return indefinite\_integral(b)-indefinite\_integral(a)  
  
  
def trapecia(a,b,n):  
 h = (b - a) / n  
 result = 0.5 \* (function(a) + function(b))  
 for i in range(1, n):  
 result += function(a + i \* h)  
 result \*= h  
 return result  
  
def process(a, b, epsilon):  
 n = 1  
 prev = trapecia(a, b, n)  
 while True:  
 n \*= 2  
 result = trapecia(a, b, n)  
 if abs(result - prev) < epsilon:  
 return result, (b - a) / n  
 prev = result  
  
  
a = 0  
b = 1  
epsilon = math.pow(10,-6)  
  
  
print('Реальное значение',integral(a,b))  
  
calculated, step = process(a,b,epsilon)  
  
print('шаг ', step)  
print('вычисленное ', calculated)

Вот результаты программы:

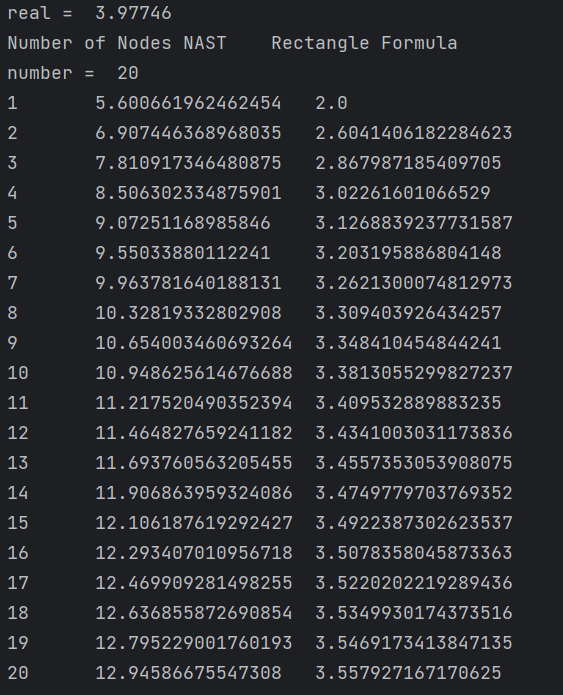


Вывод: В задании 1 точное значение интеграла совпадает с приближенным.



Листинг программы:

import math  
  
import numpy as np  
  
def f(x):  
 return np.exp(x) / np.sqrt(1 - x\*\*2)  
  
def NAST(func, a, b, n):  
 sum\_result = 0  
 for i in range(n + 1):  
 sum\_result += func(np.cos(np.pi \* (2 \* i + 1) / (2 \* (n + 1))))  
 return sum\_result \* np.pi / (n + 1)  
  
def RECTANGLE(func, a, b, n):  
 h = (b - a) / n  
 sum\_result = 0  
 for i in range(n):  
 sum\_result += func(a + i \* h + h / 2)  
 return h \* sum\_result  
  
  
a = -1  
b = 1  
num\_nodes = [20,40,200,1000,10000,100000]  
  
print("real = ", 3.97746)  
print("Number of Nodes\tNAST\tRectangle Formula")  
for num in num\_nodes:  
 print("number = ", num)  
 for n in range(1, num + 1):  
 nastRes = NAST(f, a, b, n)  
 rectRes = RECTANGLE(f, a, b, n)  
 print(f"{n}\t\t{nastRes}\t{rectRes}")  
 print()



Вывод: в нашем случае, НАСТ намного менее точная формула.