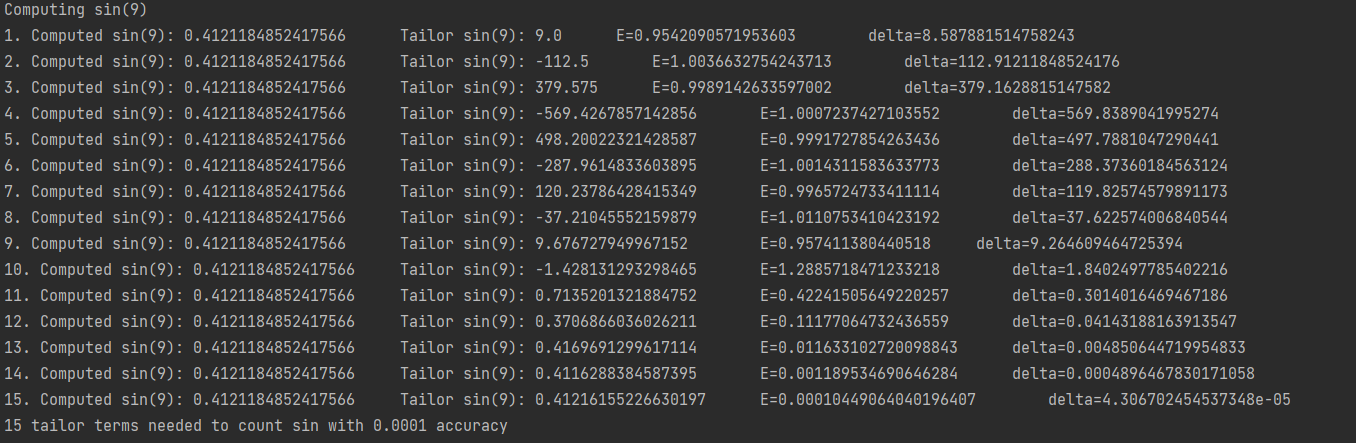
**Отчёт ИПР1  
Выполнил: Заломов Роман Андреевич, 121702**

Листинг программы для вычисления рядов Тейлора для синуса. Язык программирования - Python:

import math  
  
  
DELTA = 1E-4  
PI = math.pi  
  
  
def tailor\_term(n: int, x: float) -> float:  
 return (-1)\*\*n\*((x\*\*(2\*n+1))/math.factorial(2\*n+1))  
  
  
def sin\_tailor(term\_num: int, x: float):  
 # while x >= 2\*PI:  
 # x -= 2\*PI  
 tailor\_list = [tailor\_term(i, x) for i in range(term\_num)]  
 # print("Tailor terms:", ', '.join(list(map(str, tailor\_list))))  
 sin = sum(tailor\_list)  
 return sin  
  
  
def scores() -> str:  
 return "-"\*200  
  
  
print(scores())  
print("Computed sin\t\tTailor sin")  
for x in range(30):  
 print(scores())  
 print(f"Computing sin({x})")  
 n = 1  
 while abs(math.sin(x) - sin\_tailor(n, x)) >= DELTA:  
 print(f"{n}. Computed sin({x}): {math.sin(x)}\t\tTailor sin({x}): {sin\_tailor(n, x)}\t\t"  
 f"E={abs((math.sin(x)-sin\_tailor(n, x))/sin\_tailor(n, x))}"  
 f"\t\tdelta={abs(math.sin(x) - sin\_tailor(n, x))}")  
 n += 1  
 try:  
 print(f"{n}. Computed sin({x}): {math.sin(x)}\t\tTailor sin({x}): {sin\_tailor(n, x)}\t\t"  
 f"E={abs((math.sin(x) - sin\_tailor(n, x)) / sin\_tailor(n, x))}"  
 f"\t\tdelta={abs(math.sin(x) - sin\_tailor(n, x))}")  
 except ZeroDivisionError:  
 print(f"{n}. Computed sin({x}): {math.sin(x)}\t\tTailor sin({x}): {sin\_tailor(n, x)}\t\t"  
 f"E=0\t\tdelta={abs(math.sin(x) - sin\_tailor(n, x))}")  
 print(f"{n} tailor terms needed to count sin with {DELTA} accuracy")

Для получения заранее просчитанного значения синуса использовался модуль math. Для того, чтобы синус, просчитанный с помощью ряда Тейлора считался правильным, разница между им и заранее просчитанным синусом должна составлять не более 10^-4(константа DELTA в листинге).

Пример вывода:



Computed sin – заранее просчитанный синус  
Tailor sin – синус, подсчитанный с помощью рядов Тейлора при очередной итерации  
E – относительная погрешность  
delta – разница между заранее просчитанным синусом и синусом, подсчитанным при помощи ряда Тейлора.

Как видно, для обеспечения требуемой точности потребовалось просчитать и просуммировать 15 слагаемых ряда (последняя строка).

В данной конфигурации можно просчитать sin(t) до t = 30 (не включая t = 30). Т.е., начиная с t = 30, увеличение количества слагаемых ряда Тейлора не помогает.

Т.к. Python поддерживает только один вещественный тип(float), то для увеличения «радиуса сходимости» требуется использовать формулы приведения, т.е. привести аргумент к виду 0 < t < 2π.



(PI = 3.141592653589793)

Это приведёт к потере точности(ибо проводятся дополнительные вычислительные процедуры). Но синус можно будет вычислять для больших чисел. Также в этом случае можно увеличить точность с 10^-4, до, например, 10^-8. И даже при такой хорошей точности, синус вычисляется для достаточно больших чисел(проверено на числах <= 10000). Так, что, если использовать данный метод подсчёта, то «радиус сходимости» будет «неограниченным», что проверить достаточно сложно. Единственным способом «сломать» метод будет введение достаточно маленького числа погрешности, например, 10^-15, что позволит сократить «радиус сходимости» (при точности 10^-15 максимальное число, для которого вычисляется синус, составляет 4). Но такая точность является «запредельной» и ненужной.