Дмитриев Дмитрий Игоревич, БПИ227

**Вариант 21. Условие задачи:** Разработать программу вычисления числа π с точностью не хуже 0,05% посредством ряда Нилаканта.

**Претендую на 10 баллов за выполнение задания**

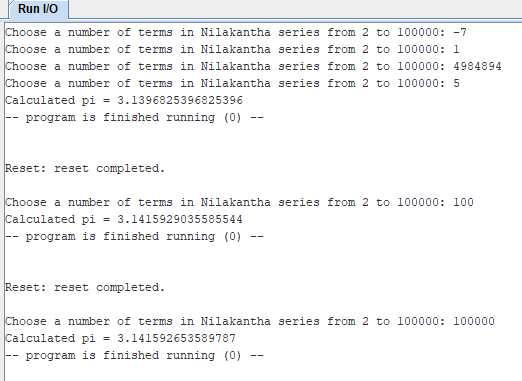
[**Репозиторий с кодом на Ассемблере и автоматическими тестами**](https://github.com/TIN-slayer/AVS_IDZ-1)

(Для запуска основной программы нужно использовать **main.asm**, а для тестов **tests.asm**. Также в репозитории есть программа на python **test.py**, которая реализует автоматические тесты с подсчётом ряда Нилаканта)

**Прогрессирующий отчёт на баллы от 4 до 10:**

**4-5 баллов:**

1. Решил задачу 21 на Ассемблере. Оформил ввод с клавиатуры и вывод в консоль.
2. Комментарии добавил.
3. Реализовал подпрограммы ввода, вычисления числа Пи по ряду Нилаканта и вывода:
   1. Подпрограмма ввода (**input**) запрашивает у пользователя число слагаемых для вычислении ряда Нилаканта (минимальное число слагаемых – это 2, потому что иначе точность Пи будет меньше 0.05, что противоречит условию задачи). Возвращает число слагаемых в регистре a0.
   2. Подпрограмма вычисления Пи (**get\_pi**) принимает на вход число слагаемых в регистрах a0. Считает ряд Никаланта с заданным числом слагаемых в а0 (для вычисления Пи использую double, чтобы повысить точность). Возвращает посчитанное число Пи в регистре fa0.
   3. Подпрограмма вывода (**output**) принимает на вход посчитанное число Пи в регистре fa0. Выводит в консоль посчитанное число Пи.
4. Прикладываю результаты **ручных** тестовых прогонов:

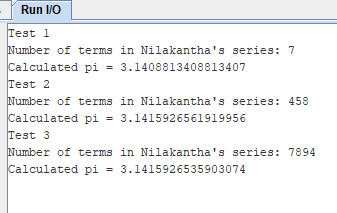


**6-7 баллов:**

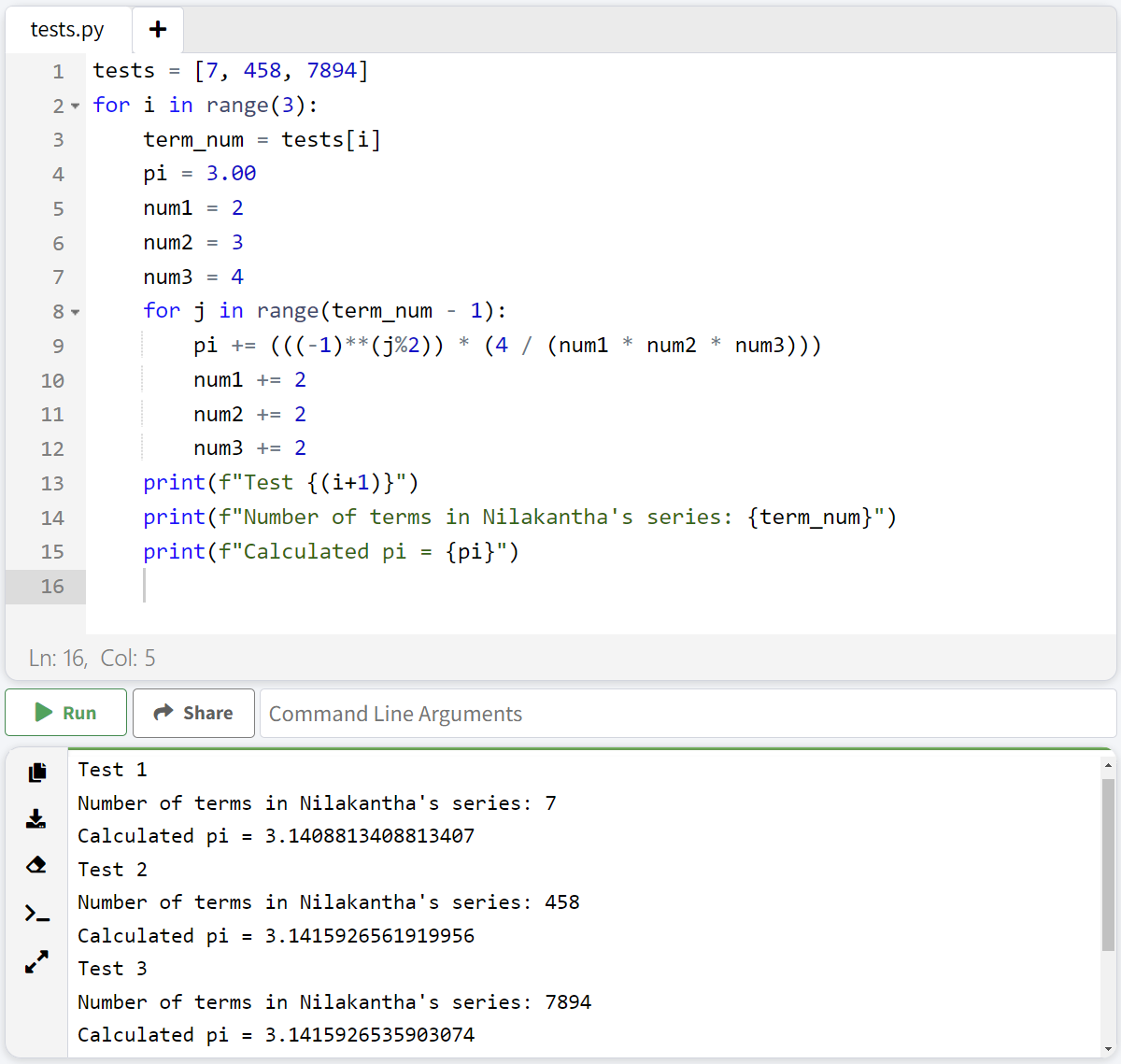
1. Аргументы в подпрограммы передавал через регистры a\_. По конвенции сохранял регистр адреса возврата в подпрограммах.
2. Использовал временные регистры, т.к. их хватило.
3. Добавил комментарии для принимаемых и возвращаемых параметров подпрограмм(функций).

**8 баллов:**

1. Подпрограммы **input, get\_pi и output** независимы от main и могут принимать аргументы от другой программы (Например, у меня реализована программа **tests.asm** для автоматического тестирования).
2. Реализовал программу **tests.asm** для автоматического тестирования на различном числе слагаемых ряда Нилаканта. Результат **автоматических** тестовых прогонов:



1. Реализовал программу на python **test.py** для дополнительной проверки моего решения на Ассемблере. **tests.py** считает число Пи, используя ряд Нилаканта (не нашёл библиотеки с рядом Нилаканта, поэтому написал подсчёт ряда самостоятельно по формуле из [википедии](https://en.wikipedia.org/wiki/Pi)), для тех же чисел слагаемых ряда, что и в **tests.asm**. Вывод обоих программ совпал, что подтверждает верность моего решения на Ассемблере. Результат работы **tests.py**:



**9 баллов:**

1. Реализовал макросы ввода и вывода (**print\_int, print\_double, read\_int, print\_str, print\_char,** ***input****,* ***output***), поддерживающие различные аргументы.
2. Обернул подпрограммы **input, get\_pi, output** одноимёнными макросами.

**10 баллов:**

1. Разбил программу на несколько файлов, которые собираются вместе во время компиляции. Подпрограммы **get\_pi** и **output** используются и в **main.asm**, и в **tests.asm**, также и там, и там используются макросы из общей библиотеки **macroslib.asm**.
2. Выделил макросы в автономную библиотеку **macroslib.asm**.