**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**по лабораторной работе № 1**

*дисциплина: Параллельное программирование*

Студент: Ким Реачна

Группа: НПИбд-01-20

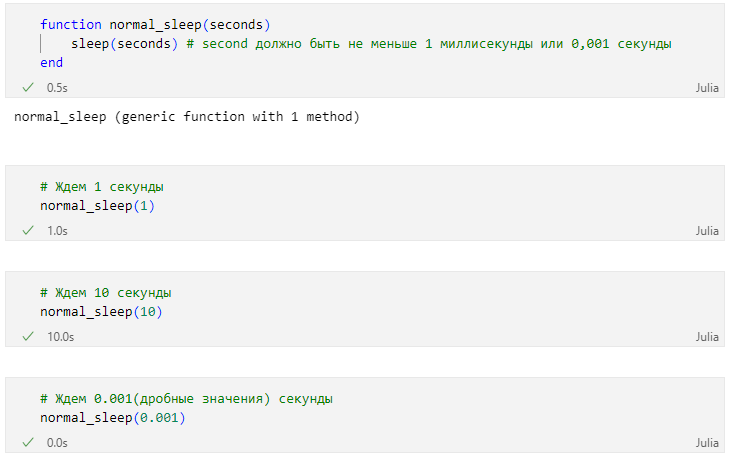
**МОСКВА**

2023 г.

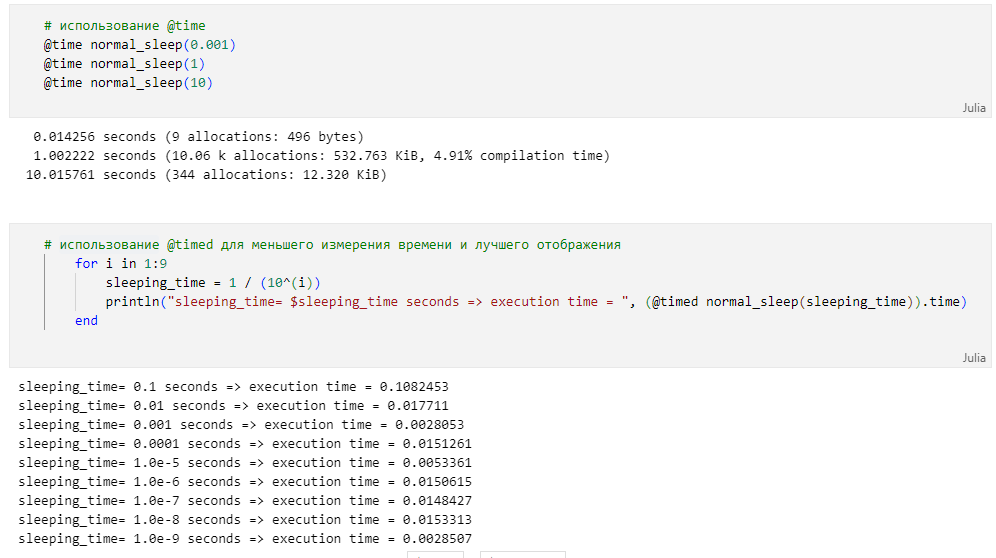
**Задание №1**

Любая программа, использующая параллельные вычисления:

* Напишите подпрограмму ***normal\_sleep***, которая ожидает некоторое время, используя встроенную функцию ***sleep*** и протестируйте ожидание в течение 1, 10 и 0,001 секунды. Поскольку минимальное время ожидания функции ***sleep*** составляет 1 миллисекунду или ввод 0,001 секунды, ввод должен составлять не менее 1 миллисекунды, и да, вы можете передавать ей дробные значения.

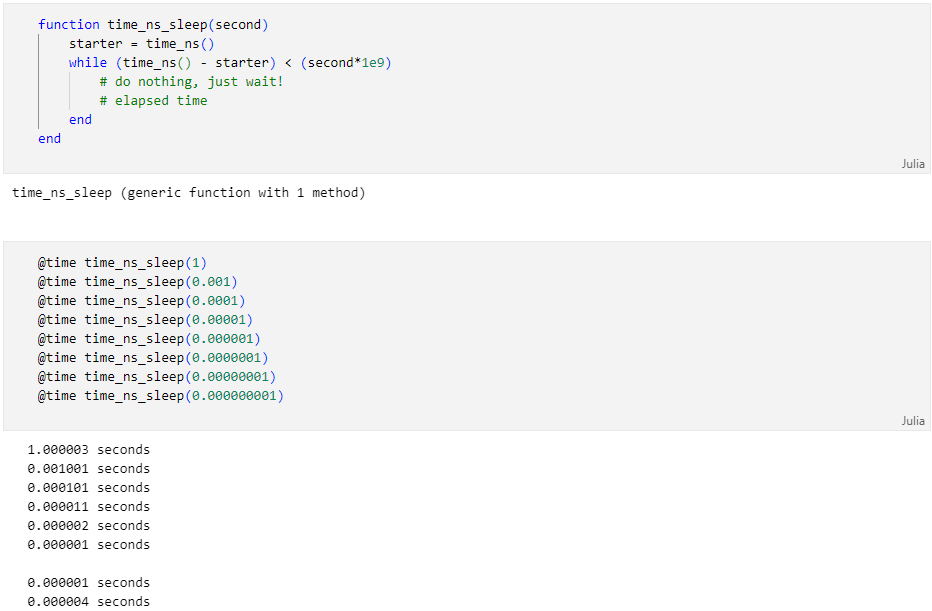


Тестирование функции ***normal\_sleep*** с использованием ***@time*** и ***@timed*** :

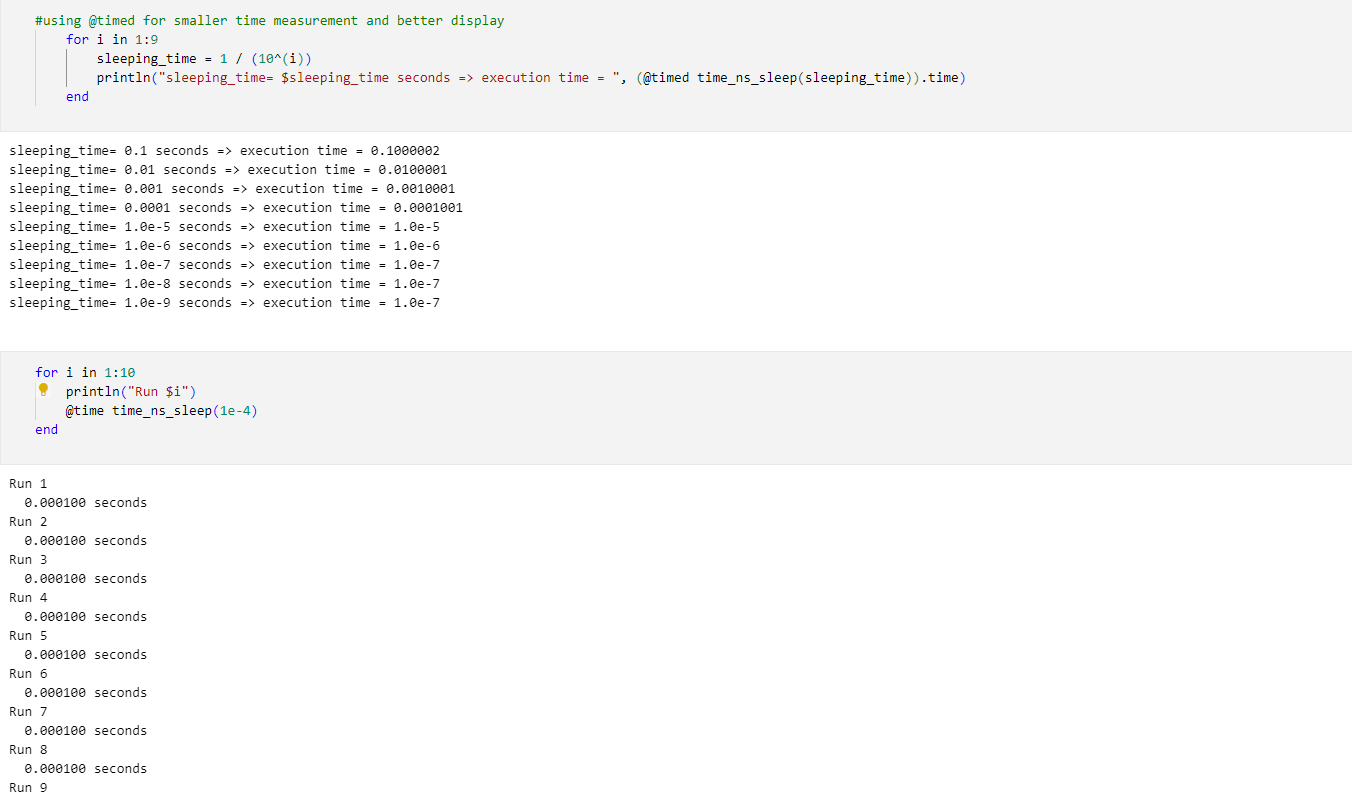


Выходные данные при запуске ***@time*** содержат информацию о прошедшем времени, выделении памяти и использовании памяти, на пример:

* normal\_sleep(0.001):
* Elapsed time /Затраченное время: 0,014256 секунды
* Memory allocations/Объем выделяемой памяти: 9
* Memory usage/Использование памяти: 496 байт
* Создайте вызов функции ***time\_ns\_sleep*** с помощью ***time\_ns()***

****

Тестирование функции ***time\_ns\_sleep*** с использованием ***@time*** и ***@timed*** :

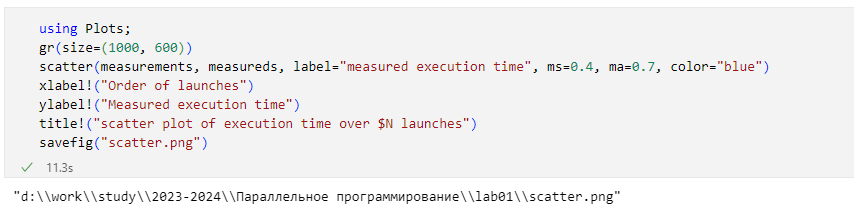
****

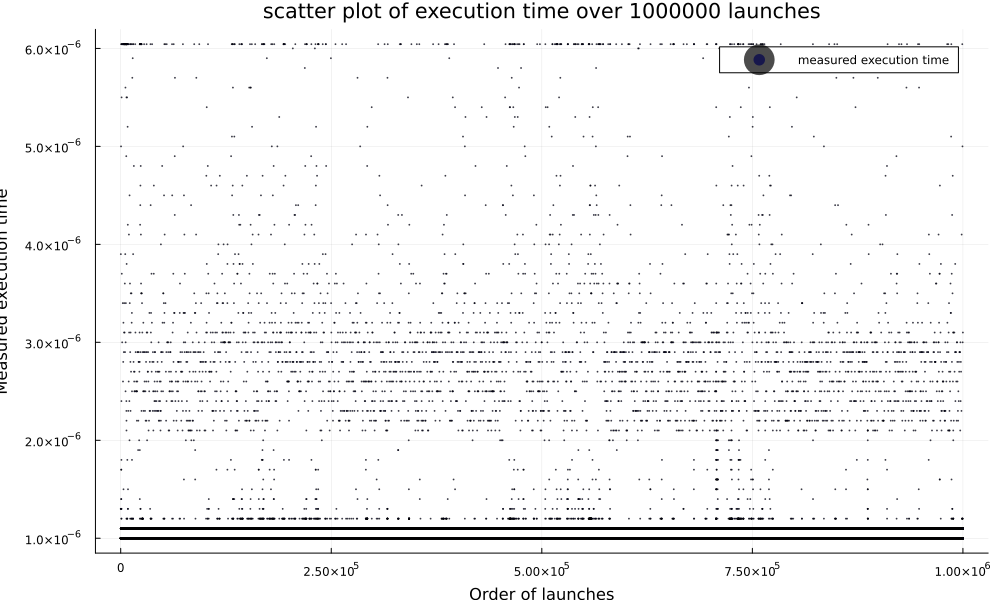
* Сделайте замеры времени для запусков. Замерьте время работы программы, распечатайте в виде облака точек (scatter) и гистограммы. Для замеров времени можно использовать BenchmarkTools:

Измерение времени выполнения функции ***time\_ns\_sleep*** для построения scatter и гистограммы:

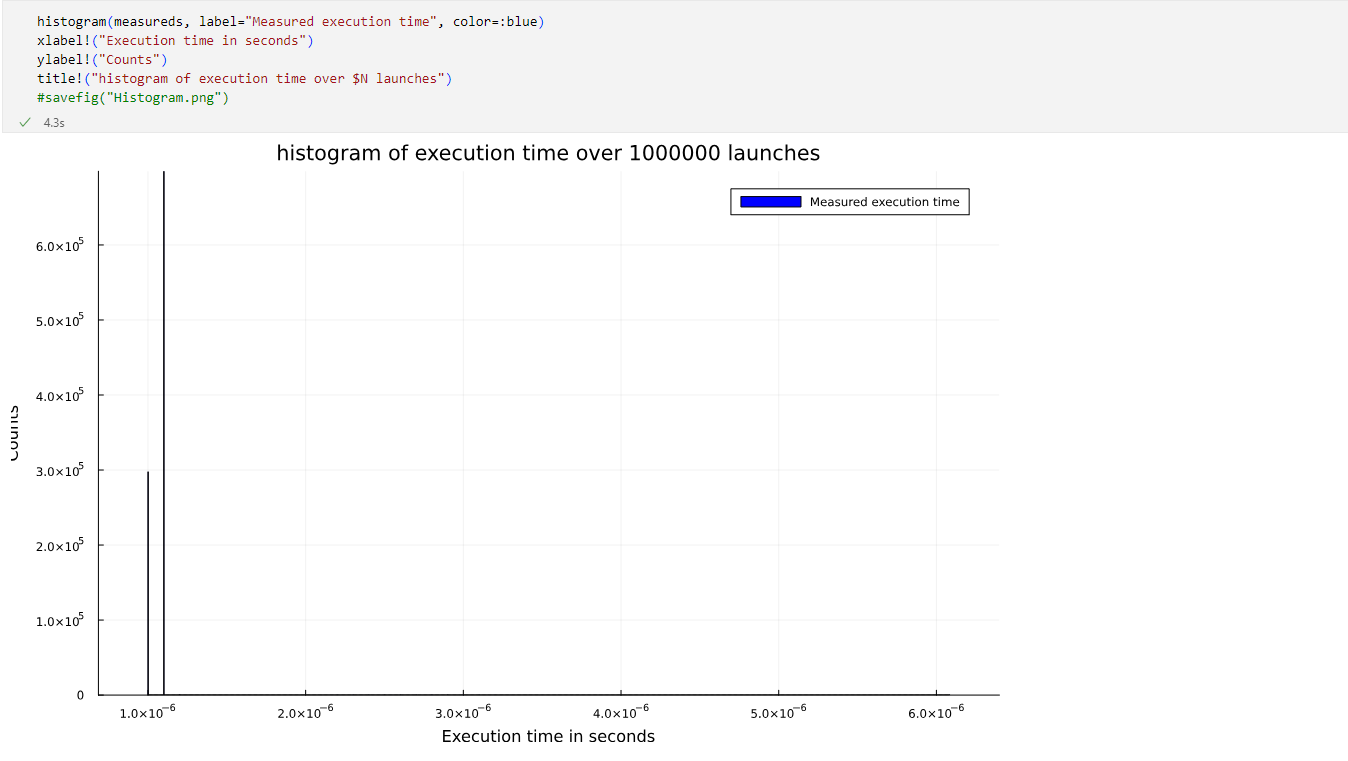


Построения scatter:

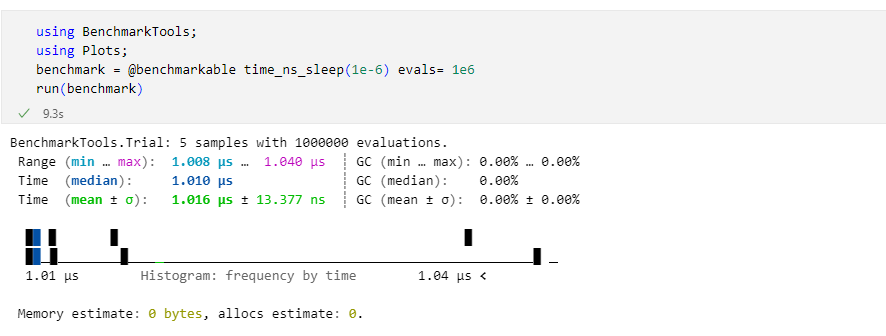




Построения гистограммы:



Измерение производительности с помощью BenchmarkTool:



**Задание №2**

* Создайте программу которая порождает потоки. Распечатайте количество созданных потоков

using Base.Threads

function spawn\_threads()

    # spawrn 4 Threads

    for i in 1:4

        sleep(0.01)

        Threads.@spawn println("Threads", i)

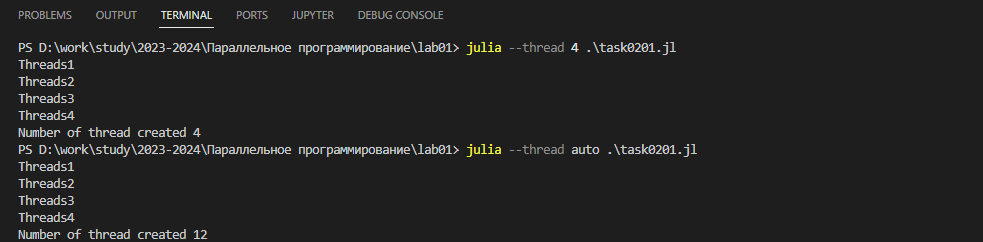
    end

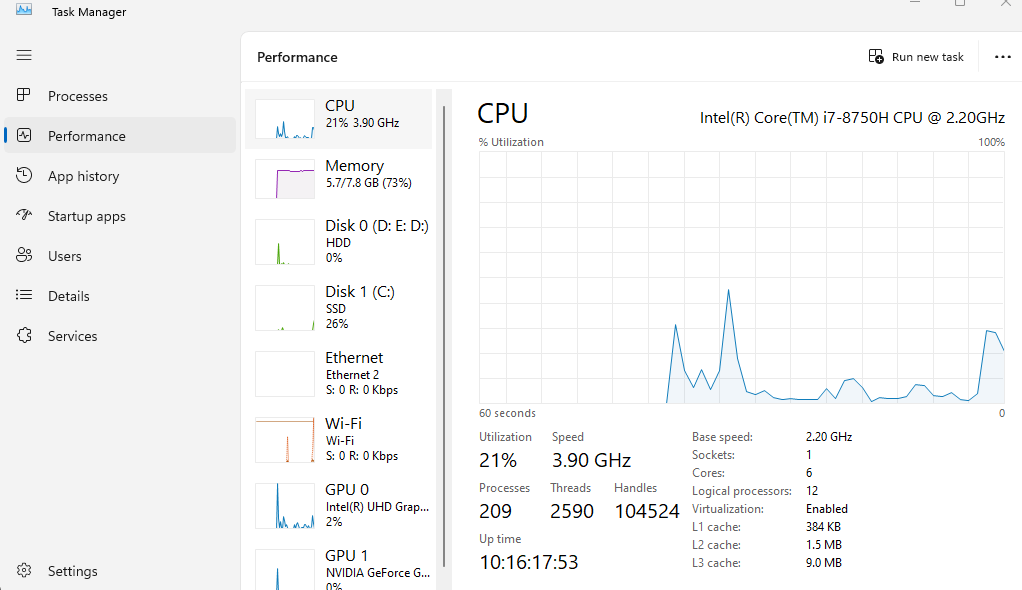
end

spawn\_threads()

println("Number of thread created ", Threads.nthreads())

Используя команду ***julia --thread 4 task 0201.jl***, выведите количество созданных потоков используя команду ***julia --thread auto task 0201.jl*** чтобы вывести, сколько потоков поддерживает производительность процессора или просмотра логического процессора (Logical processors) в Task Manager.





* Создайте многопоточную программу с четырьмя потоками, которая принимает на вход массив целых чисел. Нужно вручную распределить работу между потоками. Первый поток должен просуммировать 1, 5, 9 и т.д. числа; второй поток — 2, 6, 10 и т.д.; третий — 3, 7, 11; четвертый — 4, 8, 12 и т.д. Результаты суммирования распечатываются с указанием, какой поток какой результат получил.
* Напишите автоматические тесты для данной программы, которые проверяют ее работоспособность для разных последовательностей чисел.

using Base.Threads

using Test

function sum\_subset(input::Vector{Int}, thread\_num::Int)

    subset = input[thread\_num:4:end]

    result = sum(subset)

    println("Thread $thread\_num: Sum = $result")

    return result

end

function multithreaded\_sum(arr)

    # Create an array to store the summation results

    results = Vector{Int}(undef, 4)

    threads = Vector{Task}(undef, 4)

    # Generate 4 threads

    for i in 1:4

        threads[i] = Threads.@spawn sum\_subset(arr, i)

    end

    for i in 1:4

        results[i] = fetch(threads[i])

    end

    total\_sum = sum(results)

    return total\_sum

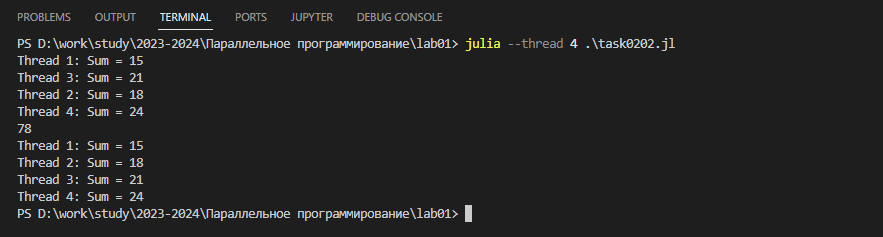
end

# Test the program with an example input array

input\_array = [x for x in 1:12]

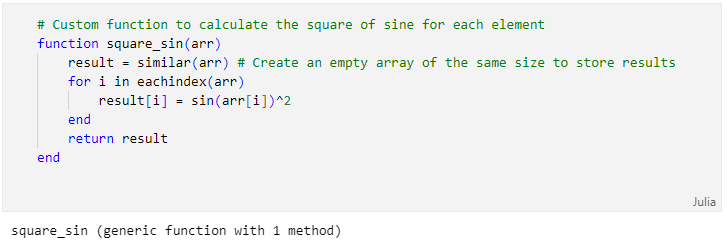
println(multithreaded\_sum(input\_array))

@test sum(input\_array) == multithreaded\_sum(input\_array)



**Задание №3**

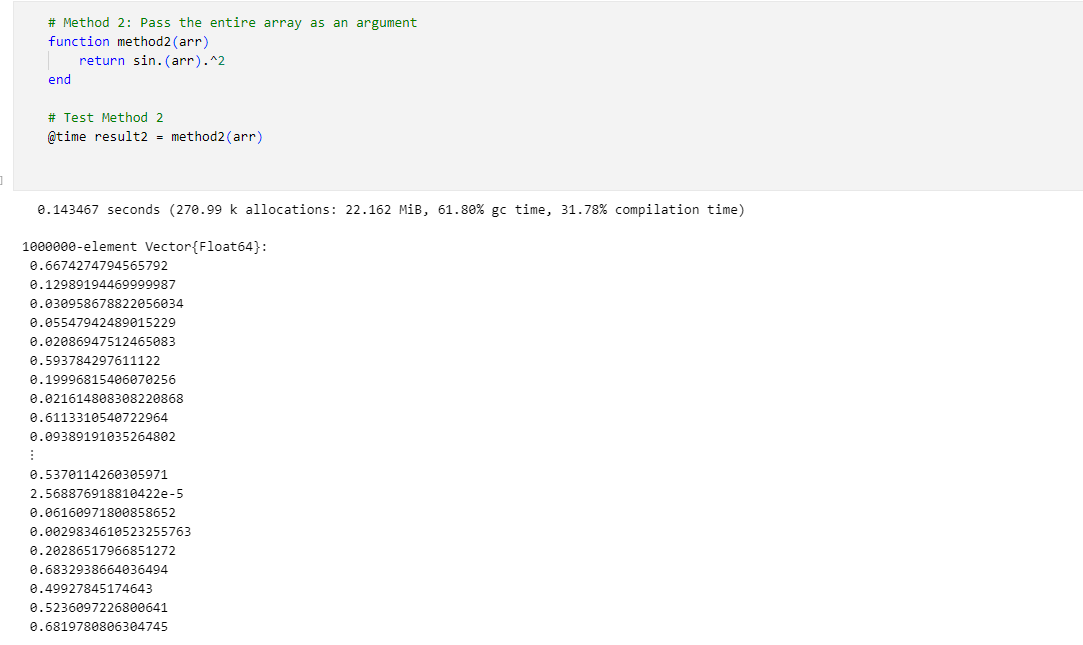
* Создайте функцию ***square\_sin***, которая вычисляет квадрат синуса каждого элемента в массиве:



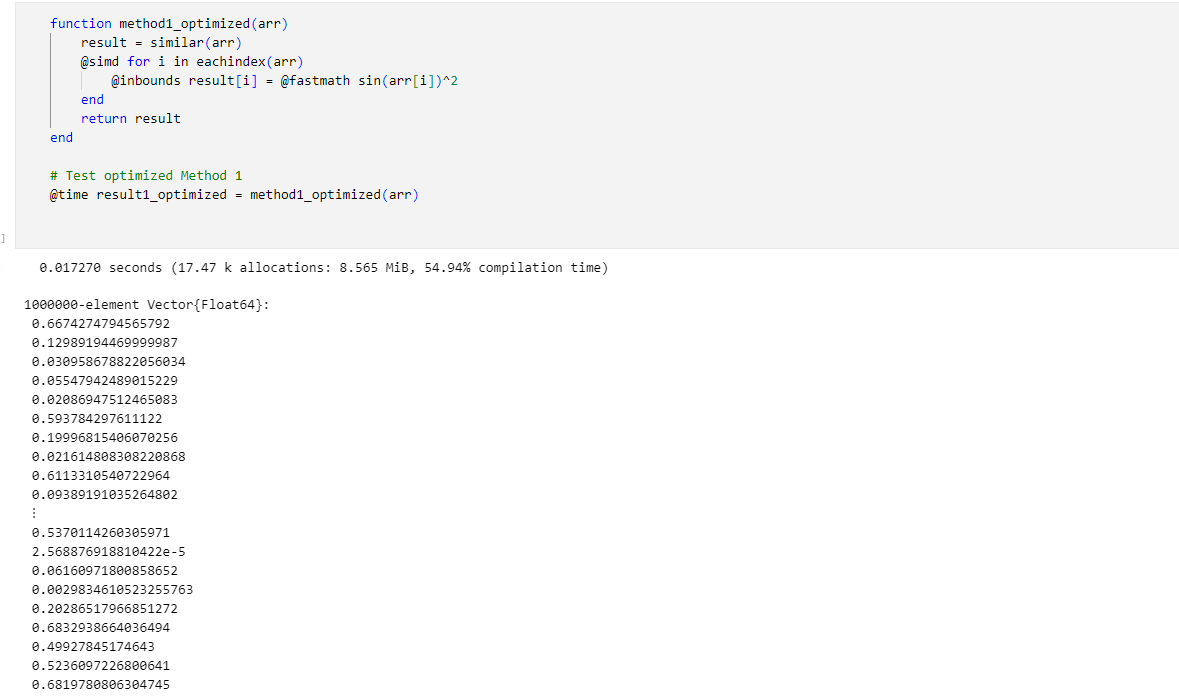
* Первый способ заключается в вычислении значений функций от элементов массива в цикле, передавая каждый элемент массива в функцию по отдельности.



* Второй способ заключается в передаче всего массива в виде аргумента

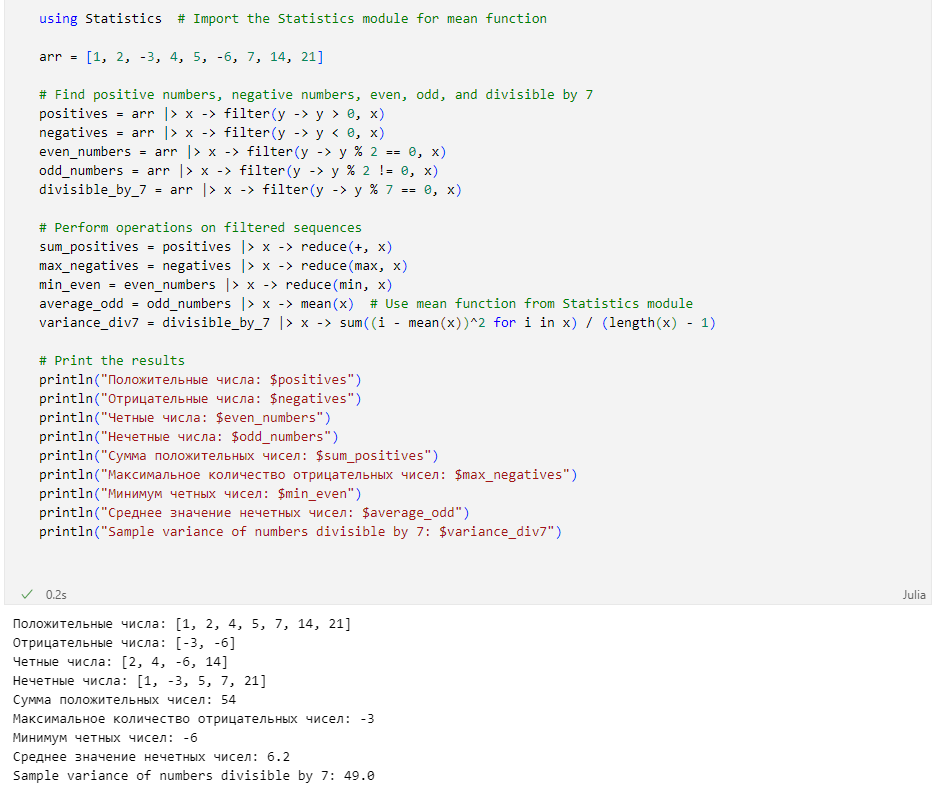


Используя макросы ***@inbounds, @fastmath и @simd***:

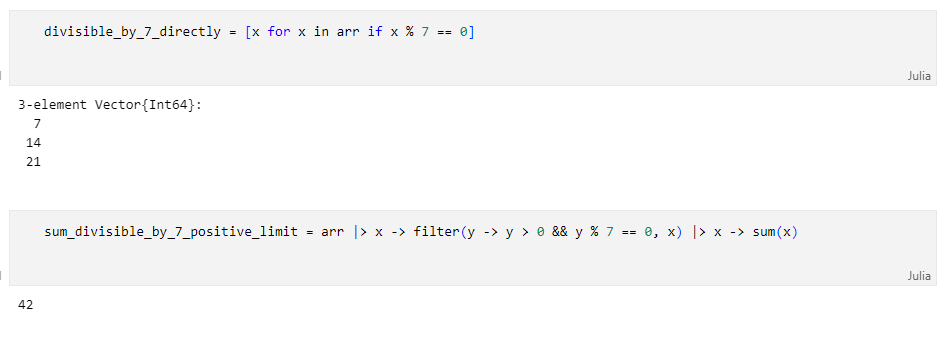


**Задание №4**

* Создайте небольшой массив целых чисел, такой, чтобы можно было проверить корректность вычислений. С помощью reduce сделайте с ним следующие действия.
* Найдите все положительные числа, отрицательные числа, четные, нечетные, делящиеся без остатка на 7.
* Затем с получившимися в результате такой фильтрации последовательностями проделайте следующие операции: просуммируйте, найдите максимум, минимум, среднее, выборочную дисперсию.

****

* Попробуйте сразу создать массив с перечисленными выше условиями, то есть например такой, который состоит из целых чисел, делящихся без остатка на 7.
* Объедините все условия вместе, то есть например найдите сумму всех элементов, которые делятся на 7 без остатка, при этом положительные, не больше какого-то числа.

****