

Лабораторная работа №3

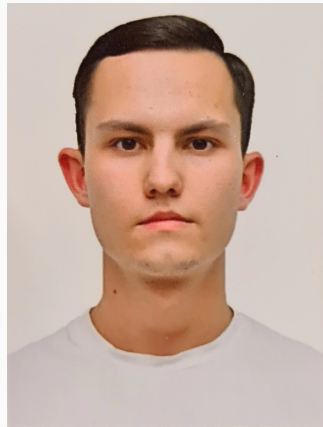
Управляющие структуры

Дурдалыев Максат

2025-10-11

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

- Дурдалыев Максат
- Студент НКНбд-01-22
- Российский университет дружбы народов имени Патриса Лумумбы
- 1132205337@pfur.ru
- <https://github.com/mdurdalyyev>



2 Цели и задачи

Цель работы

Основная цель работы — освоить применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.

Задание

1. Используя Jupyter Lab, повторите примеры.
2. Выполните задания для самостоятельной работы.

3 Циклы while и for

Для различных операций, связанных с перебором индексируемых элементов структур данных, традиционно используются циклы while и for.

Синтаксис while

```
while <condition>  
    <loop body>  
end
```

4 Циклы while и for

```
# пока n<10 прибавить к n единицу и распечатать значение:  
n = 0  
while n < 10  
    n += 1  
    println(n)  
end
```

✓ 0.0s

Julia

1
2
3
4
5
6
7
8
9
10

```
myfriends = ["Ted", "Robyn", "Barney", "Lily", "Marshall"]  
i = 1  
while i <= length(myfriends)  
    friend = myfriends[i]  
    println("Hi $friend, it's great to see you!")  
    i += 1  
end
```

✓ 0.0s

Julia

Hi Ted, it's great to see you!
Hi Robyn, it's great to see you!

5 Циклы while и for

Такие же результаты можно получить при использовании цикла for.

Синтаксис for

```
for <variable> in <range>  
    <loop body>  
end
```

6 Циклы while и for

```
for n in 1:2:10
println(n)
end
myfriends = ["Robyn", "Barney", "Lily", "Marshall"]
for friend in myfriends
println("Hi $friend, it's great to see you!")
end
```

✓ 0.0s

Julia

1
3
5
7
9

Hi Ted, it's great to see you!

Hi Robyn, it's great to see you!

Hi Barney, it's great to see you!

Hi Lily, it's great to see you!

7 Циклы while и for

```
# инициализация массива m x n из нулей:
m, n = 5, 5
A = fill{0, (m, n)}
# формирование массива, в котором значение каждой записи
# является суммой индексов строки и столбца:
for i in 1:m
    for j in 1:n
        A[i, j] = i + j
    end
end
A
```

✓ 0.0s

Julia

5x5 Matrix{Int64}:

```
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
6 7 8 9 10
```

```
# инициализация массива m x n из нулей:
B = fill{0, (m, n)}
for i in 1:m, j in 1:n
    B[i, j] = i + j
end
B
```

✓ 0.0s

Julia

5x5 Matrix{Int64}:

```
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
6 7 8 9 10
```

8 Циклы while и for

```
C = [i + j for i in 1:m, j in 1:n]
```

```
C
```

✓ 0.0s

Julia

5x5 Matrix{Int64}:

2 3 4 5 6

3 4 5 6 7

4 5 6 7 8

5 6 7 8 9

6 7 8 9 10

9 Условные выражения

Довольно часто при решении задач требуется проверить выполнение тех или иных условий. Для этого используют условные выражения.

Синтаксис условных выражений с ключевым словом:

```
if <condition 1>  
    <action 1>  
elseif <condition 2>  
    <action 2>  
else  
    <action 3>  
end
```

10 Условные выражения

```
# используем `&&` для реализации операции "AND"
# операция % вычисляет остаток от деления
N = 100
if (N % 3 == 0) && (N % 5 == 0)
println("FizzBuzz")
elseif N % 3 == 0
println("Fizz")
elseif N % 5 == 0
println("Buzz")
else
println(N)
end
```

✓ 0.0s

Julia

Buzz

```
x = 5
y = 10
(x > y) ? x : y
```

✓ 0.0s

Julia

11 Функции

```
function sayhi(name)
println("Hi $name, it's great to see you!")
end
```

✓ 0.0s

Julia

sayhi (generic function with 1 method)

```
# функция возведения в квадрат:
function f(x)
x^2
end
```

✓ 0.0s

Julia

f (generic function with 1 method)

```
sayhi("C-3PO")
f(42)
```

✓ 0.0s

Julia

Hi C-3PO, it's great to see you!

1764

```
sayhi2(name) = println("Hi $name, it's great to see you!")
f2(x) = x^2
```

✓ 0.0s

Julia

12 Функции

```
sayhi("C-3PO")
```

```
f(42)
```

✓ 0.0s

Julia

Hi C-3PO, it's great to see you!

1764

```
sayhi3 = name -> println("Hi $name, it's great to see you!")
```

```
f3 = x -> x^2
```

✓ 0.0s

Julia

#83 (generic function with 1 method)

```
sayhi("C-3PO")
```

```
f(42)
```

✓ 0.0s

Julia

Hi C-3PO, it's great to see you!

13 Функции

По соглашению в Julia функции, сопровождаемые восклицательным знаком, изменяют свое содержимое, а функции без восклицательного знака не делают этого:

```
# задаём массив v:
```

```
v = [3, 5, 2]
```

✓ 0.0s

Julia

```
3-element Vector{Int64}:
```

```
3
```

```
5
```

```
2
```

```
sort(v)
```

```
v
```

✓ 0.0s

Julia

```
3-element Vector{Int64}:
```

```
3
```

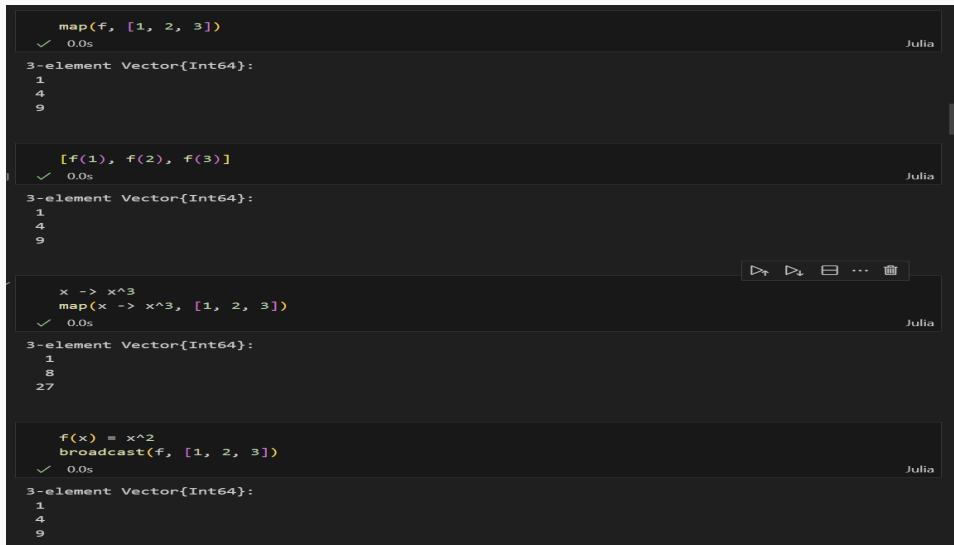
```
5
```

```
2
```

В Julia функция `map` является функцией высшего порядка, которая принимает функцию в качестве одного из своих входных аргументов и применяет эту функцию к каждому элементу структуры данных, которая ей передаётся также в качестве аргумента.

Функция `broadcast` — ещё одна функция высшего порядка в Julia, представляющая собой обобщение функции `map`. Функция `broadcast()` будет пытаться привести все объекты к общему измерению, `map()` будет напрямую применять данную функцию поэлементно.

15 Функции



The screenshot displays four separate Julia REPL sessions, each showing a function call, its execution time, and the resulting output. The sessions are separated by horizontal lines. The first session uses `map(f, [1, 2, 3])` with `f(x) = x^4` (implied), resulting in `[1, 4, 9]`. The second session uses `[f(1), f(2), f(3)]` with `f(x) = x^4` (implied), also resulting in `[1, 4, 9]`. The third session defines `x -> x^3` and uses `map(x -> x^3, [1, 2, 3])`, resulting in `[1, 8, 27]`. The fourth session defines `f(x) = x^2` and uses `broadcast(f, [1, 2, 3])`, resulting in `[1, 4, 9]`. Each session includes a green checkmark icon and a 'Julia' label in the top right corner. A toolbar with icons for navigation and editing is visible between the second and third sessions.

```
map(f, [1, 2, 3])
✓ 0.0s
3-element Vector{Int64}:
 1
 4
 9
```

```
[f(1), f(2), f(3)]
✓ 0.0s
3-element Vector{Int64}:
 1
 4
 9
```

```
x -> x^3
map(x -> x^3, [1, 2, 3])
✓ 0.0s
3-element Vector{Int64}:
 1
 8
27
```

```
f(x) = x^2
broadcast(f, [1, 2, 3])
✓ 0.0s
3-element Vector{Int64}:
 1
 4
 9
```

Рисунок 0: Примеры использования функций `map()` и `broadcast()`

16 Функции

```
f.([1, 2, 3])
✓ 0.0s Julia

3-element Vector{Int64}:
 1
 4
 9
```

```
# Задаём матрицу A:
A = [i + 3*j for j in 0:2, i in 1:3]
✓ 0.0s Julia

3×3 Matrix{Int64}:
 1  2  3
 4  5  6
 7  8  9
```

```
f(A)
✓ 0.0s Julia

3×3 Matrix{Int64}:
30  36  42
66  81  96
102 126 150
```

```
B = f.(A)
✓ 0.0s Julia

3×3 Matrix{Int64}:
 1  4  9
16 25 36
49 64 81
```

Рисунок 10: Примеры использования функций `map()` и `broadcast()`

17 Функции

```
A .+ 2 .* f.(A) ./ A
```

✓ 0.6s

Julia

```
3×3 Matrix{Float64}:
```

```
 3.0  6.0  9.0
12.0 15.0 18.0
21.0 24.0 27.0
```

```
@. A + 2 * f(A) / A
```

✓ 0.0s

Julia

```
3×3 Matrix{Float64}:
```

```
 3.0  6.0  9.0
12.0 15.0 18.0
21.0 24.0 27.0
```

```
broadcast(x -> x + 2 * f(x) / x, A)
```

✓ 0.0s

Julia

```
3×3 Matrix{Float64}:
```

```
 3.0  6.0  9.0
```

18 Сторонние библиотеки (пакеты) в Julia

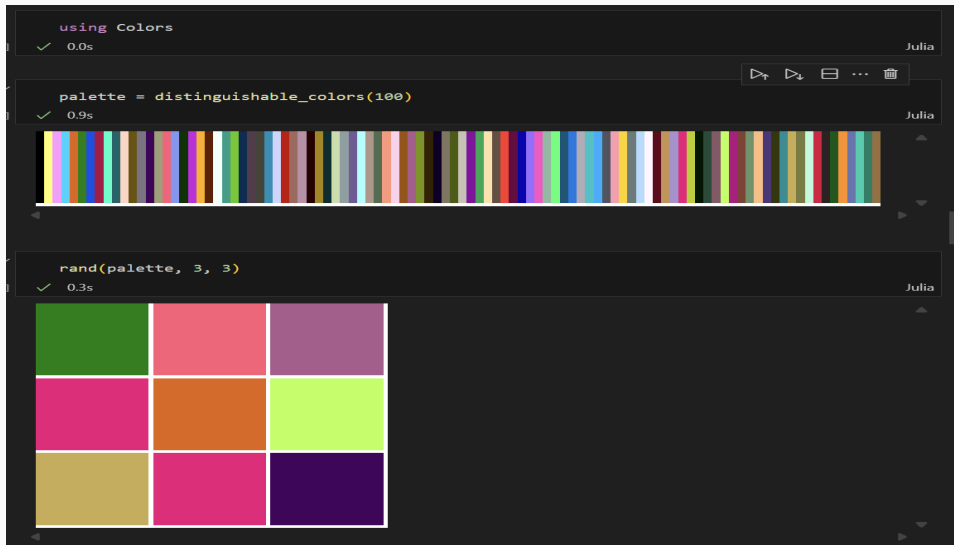


Рисунок 12: Пример использования сторонних библиотек

19 Самостоятельная работа

Задания для самостоятельного выполнения

№1. Используя циклы `while` и `for`:

1.1) выведите на экран целые числа от 1 до 100 и напечатайте их квадраты:

```
n = 1
while n <= 100
    println(n, "^2 = ", (n^2))
    n += 1
end
```

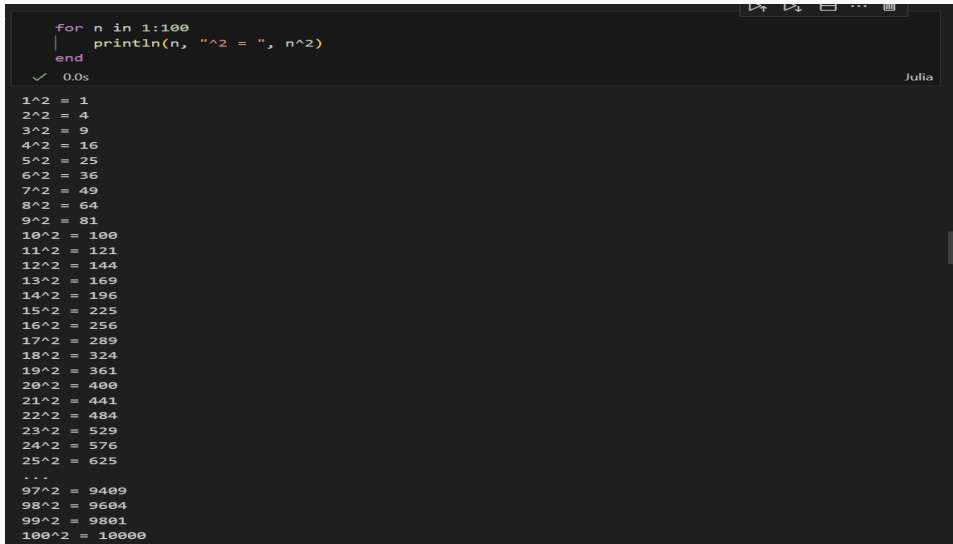
✓ 0.0s

Julia

```
1^2 = 1
2^2 = 4
3^2 = 9
4^2 = 16
5^2 = 25
6^2 = 36
7^2 = 49
8^2 = 64
9^2 = 81
10^2 = 100
11^2 = 121
12^2 = 144
13^2 = 169
14^2 = 196
15^2 = 225
16^2 = 256
17^2 = 289
18^2 = 324
19^2 = 361
20^2 = 400
21^2 = 441
22^2 = 484
23^2 = 529
24^2 = 576
25^2 = 625
...
97^2 = 9409
98^2 = 9604
99^2 = 9801
100^2 = 10000
```

Рисунок 13: Выполнение подпунктов задания №1

20 Самостоятельная работа



```
for n in 1:100
    println(n, "^2 = ", n^2)
end
```

✓ 0.0s Julia

1^2 = 1
2^2 = 4
3^2 = 9
4^2 = 16
5^2 = 25
6^2 = 36
7^2 = 49
8^2 = 64
9^2 = 81
10^2 = 100
11^2 = 121
12^2 = 144
13^2 = 169
14^2 = 196
15^2 = 225
16^2 = 256
17^2 = 289
18^2 = 324
19^2 = 361
20^2 = 400
21^2 = 441
22^2 = 484
23^2 = 529
24^2 = 576
25^2 = 625
...
97^2 = 9409
98^2 = 9604
99^2 = 9801
100^2 = 10000

Рисунок 14: Выполнение подпунктов задания №1

21 Самостоятельная работа

1.2) создайте словарь `squares`, который будет содержать целые числа в качестве ключей и квадраты в качестве их пар-значений:

```
squares = Dict()
for n in 1:100
    squares[n] = n^2
end
println(squares)
```

✓ 0.4s

Julia

Dict{Any, Any}(5 => 25, 56 => 3136, 35 => 1225, 55 => 3025, 60 => 3600, 30 => 900, 32 => 1024, 6

1.3) создайте массив `squares_arr`, содержащий квадраты всех чисел от 1 до 100:

```
squares_arr = [n^2 for n in 1:100]
println(squares_arr)
```

✓ 0.1s

Julia

22 Самостоятельная работа

№2. Напишите условный оператор, который печатает число, если число чётное, и строку «нечётное», если число нечётное.

Перепишите код, используя тернарный оператор:

```
n = 17
if n % 2 == 0
|   println(n)
else
|   println("Нечётное")
end
```

✓ 0.0s

Julia

Нечётное

```
n = 12
println(n % 2 == 0 ? n : "Нечётное")
```

✓ 0.0s

Julia

12

№3. Напишите функцию `add_one`, которая добавляет 1 к своему входу:

```
function add_one(x)
|   return x + 1
end
println(add_one(12))
```

✓ 0.0s

Julia

23 Самостоятельная работа

№4. Используйте `map()` или `broadcast()` для задания матрицы A, каждый элемент которой увеличивается на единицу по сравнению с предыдущим:

```
A = [12 11 10; 12 11 10; -12 -11 -10]
B = map(x -> x + 1, A)
println(B)
```

✓ 0.0s

Julia

```
[13 12 11; 13 12 11; -11 -10 -9]
```

```
B = broadcast(x -> x + 1, A)
println(B)
```

✓ 0.0s

Julia

```
[13 12 11; 13 12 11; -11 -10 -9]
```

24 Самостоятельная работа

№5. Задайте матрицу A следующего вида. Найдите A^3 . Замените третий столбец матрицы A на сумму второго и третьего столбцов:

```
A = [1 1 3; 5 2 6; -2 -1 -3]
println(map(x -> x^3, A))
```

✓ 0.0s

Julia

```
[1 1 27; 125 8 216; -8 -1 -27]
```

```
A
```

✓ 0.0s

Julia

```
3×3 Matrix{Int64}:
```

```
 1   1   3
 5   2   6
-2  -1  -3
```

```
A[:, 3] = A[:, 2] + A[:, 3]
```

```
A
```

✓ 0.0s

Julia

```
3×3 Matrix{Int64}:
```

25 Самостоятельная работа

№6. Создайте матрицу B с элементами $B_{i1} = 10$, $B_{i2} = -10$, $B_{i3} = 10$, $i = 1, 2, \dots, 15$. Вычислите матрицу $C = B^T B$:

```
B = repeat([10 -10 10], 15, 1)
```

✓ 0.1s

Julia

15×3 Matrix{Int64}:

```
10  -10  10
10  -10  10
10  -10  10
10  -10  10
10  -10  10
10  -10  10
10  -10  10
10  -10  10
10  -10  10
10  -10  10
10  -10  10
10  -10  10
10  -10  10
10  -10  10
10  -10  10
```

```
C = B' * B
println(C)
```

26 Самостоятельная работа

№7. Создайте матрицу Z размерности 6 x 6, все элементы которой равны нулю и матрицу E, все элементы которой равны 1. Используя цикл while или for и закономерности расположения элементов, создайте следующие матрицы размерности 6 x 6:

```
Z = zeros(Int, 6, 6)
println(Z)
E = ones(Int, 6, 6)
println(E)
n = 6

Z1 = copy(Z)
for i in 1:n
    for j in 1:n
        if abs(i - j) == 1
            Z1[i, j] = 1
        end
    end
end
println(Z1)

Z2 = copy(Z)
for i in 1:n
    if i + 2 < n
        Z2[i, i] = 1
        Z2[i, i+2] = 1
    elseif (i == 3) || (i == 4)
        Z2[i, i] = 1
        Z2[i, i+2] = 1
        Z2[i, i-2] = 1
    else
        Z2[i, i] = 1
        Z2[i, i-2] = 1
    end
end
println(Z2)
```

Рисунок 20: Выполнение задания №7

27 Самостоятельная работа

```
Z3 = copy(Z)
for i in 1:n
    for j in 1:n
        if (i + j) % 2 != 0
            Z3[i, j] = 1
        end
    end
end
println(Z3)

Z4 = copy(Z)
for i in 1:n
    for j in 1:n
        if (i + j) % 2 == 0
            Z4[i, j] = 1
        end
    end
end
println(Z4)
```

✓ 0.0s

Julia

n::Int64 = 6

[0 0 0 0 0 0; 0 0 0 0 0 0; 0 0 0 0 0 0; 0 0 0 0 0 0; 0 0 0 0 0 0; 0 0 0 0 0 0]
[1 1 1 1 1 1; 1 1 1 1 1 1; 1 1 1 1 1 1; 1 1 1 1 1 1; 1 1 1 1 1 1; 1 1 1 1 1 1]
[0 1 0 0 0 0; 1 0 1 0 0 0; 0 1 0 1 0 0; 0 0 1 0 1 0; 0 0 0 1 0 1; 0 0 0 0 1 0]
[1 0 1 0 0 0; 0 1 0 1 0 0; 0 0 1 0 1 0; 0 1 0 1 0 1; 0 0 1 0 1 0; 0 0 0 1 0 1]
[0 1 0 1 0 1; 1 0 1 0 1 0; 0 1 0 1 0 1; 1 0 1 0 1 0; 0 1 0 1 0 1; 1 0 1 0 1 0]
[1 0 1 0 1 0; 0 1 0 1 0 1; 1 0 1 0 1 0; 0 1 0 1 0 1; 1 0 1 0 1 0; 0 1 0 1 0 1]

Рисунок 21: Выполнение задания №7

28 Самостоятельная работа

```

Z1
✓ 0.0s
6x6 Matrix{Int64}:
0 1 0 0 0 0
1 0 1 0 0 0
0 1 0 1 0 0
0 0 1 0 1 0
0 0 0 1 0 1
0 0 0 0 1 0
Julia

Z2
✓ 0.0s
6x6 Matrix{Int64}:
1 0 1 0 0 0
0 1 0 1 0 0
0 0 1 0 1 0
0 1 0 1 0 1
0 0 1 0 1 0
0 0 0 1 0 1
Julia

Z3
✓ 0.0s
6x6 Matrix{Int64}:
0 1 0 1 0 1
1 0 1 0 1 0
0 1 0 1 0 1
1 0 1 0 1 0
0 1 0 1 0 1
1 0 1 0 1 0
Julia
```

Рисунок 22: Выполнение задания №7

29 Самостоятельная работа

Z4

✓ 0.0s

Julia

```
6×6 Matrix{Int64}:
 1  0  1  0  1  0
 0  1  0  1  0  1
 1  0  1  0  1  0
 0  1  0  1  0  1
 1  0  1  0  1  0
 0  1  0  1  0  1
```

Создать Code Markdown

№8. В языке R есть функция `outer()`. Фактически, это матричное умножение с возможностью изменить применяемую операцию (например, заменить произведение на сложение или возведение в степень):

8.1) Напишите свою функцию, аналогичную функции `outer()` языка R. Функция должна иметь следующий интерфейс: `outer(x,y,operation)`:

```
function outer(x, y, operation)
|   return [operation(xi, yj) for xi in x, yj in y]
end
```

✓ 0.0s

Julia

`outer` (generic function with 1 method)

Рисунок 23: Выполнение задания №7 и №8

30 Самостоятельная работа

8.2) Используя написанную вами функцию `outer()`, создайте матрицы следующей структуры:

```
A1 = outer(0:4, 0:4, +)
```

```
A1
```

```
✓ 0.0s
```

Julia

```
5x5 Matrix{Int64}:
```

```
0 1 2 3 4
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
```

```
A2 = outer(0:4, 1:5, ^)
```

```
A2
```

```
✓ 0.0s
```

Julia

```
5x5 Matrix{Int64}:
```

```
0 0 0 0 0
1 1 1 1 1
2 4 8 16 32
3 9 27 81 243
4 16 64 256 1024
```

```
A3 = outer(0:4, 0:4, (x, y) -> mod(x+y,5))
```

```
A3
```

```
✓ 0.0s
```

Julia

```
5x5 Matrix{Int64}:
```

```
0 1 2 3 4
1 2 3 4 0
2 3 4 0 1
3 4 0 1 2
4 0 1 2 3
```

Рисунок 24: Выполнение подпунктов задания №8

31 Самостоятельная работа

```
A4 = outer(0:9, 0:9, (x, y) -> mod(x+y,10))
```

```
A4
```

✓ 0.0s

Julia

```
10×10 Matrix{Int64}:
```

```
0 1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 0
2 3 4 5 6 7 8 9 0 1
3 4 5 6 7 8 9 0 1 2
4 5 6 7 8 9 0 1 2 3
5 6 7 8 9 0 1 2 3 4
6 7 8 9 0 1 2 3 4 5
7 8 9 0 1 2 3 4 5 6
8 9 0 1 2 3 4 5 6 7
9 0 1 2 3 4 5 6 7 8
```

🔗 Создать

+ Code

+ Markdown

```
A5 = outer(0:8, 0:8, (x, y) -> mod(x-y,9))
```

```
A5
```

✓ 0.0s

Julia

```
9×9 Matrix{Int64}:
```

```
0 8 7 6 5 4 3 2 1
1 0 8 7 6 5 4 3 2
2 1 0 8 7 6 5 4 3
3 2 1 0 8 7 6 5 4
4 3 2 1 0 8 7 6 5
5 4 3 2 1 0 8 7 6
6 5 4 3 2 1 0 8 7
7 6 5 4 3 2 1 0 8
8 7 6 5 4 3 2 1 0
```

32 Самостоятельная работа

№9. Решите следующую систему линейных уравнений с 5 неизвестными:

```
A = [1 2 3 4 5;  
     2 1 2 3 4;  
     3 2 1 2 3;  
     4 3 2 1 2;  
     5 4 3 2 1]  
b = [7; -1; -3; 5; 17]
```

```
x = A \ b  
println(x)
```

✓ 0.1s

Julia

[-2.00000000000000036, 3.00000000000000058, 4.999999999999998, 1.9999999999999991, -3.9999999999999999]

№10. Создайте матрицу M размерности 6 x 10, элементами которой являются целые числа, выбранные случайным образом с повторениями из совокупности 1, 2, ..., 10:

```
M = rand(1:10, 6, 10)  
M
```

✓ 0.0s

Julia

```
6x10 Matrix{Int64}:  
 4  1  3  7  6  9  3  7  5  4  
 5  9  1  8  5  7  9  7  2  6  
 7  6  4  3  1  3  9  4 10 10  
 1  1  7  6  2  5  7  7  7  7  
 8  1  6  3  9  1 10  6  1  5  
 6  7  3  5  6  7  9  7 10  7
```

Рисунок 26: Выполнение задания №9 и №10

33 Самостоятельная работа

10.1) Найдите число элементов в каждой строке матрицы M, которые больше числа N (например, N = 4):

```
N = 4
greater_than_N = sum(M .> N)
println(greater_than_N)
```

✓ 0.0s

Julia

40

10.2) Определите, в каких строках матрицы M число M (например, M = 7) встречается ровно 2 раза:

```
rows_with_M_twice = [i for i=1:6 if sum(M[i, :]) == 7] == 2]
println(rows_with_M_twice)
```

✓ 0.0s

Julia

[1, 2]

10.3) Определите все пары столбцов матрицы M, сумма элементов которых больше K (например, K = 75):

```
K = 75
col_pairs = []
for i in 1:size(M, 2)-1
    for j in i+1:size(M, 2)
        if sum(M[:,i]) + M[:,j] > K
            push!(col_pairs, (i, j))
        end
    end
end
println(col_pairs)
```

✓ 0.2s

Julia

Any{Tuple{Int64, Int64}} = [(1, 7), (4, 7), (5, 7), (6, 7), (7, 8), (7, 9), (7, 10), (8, 10)]

Рисунок 27: Выполнение подпунктов задания №10

34 Самостоятельная работа

№11. Вычислите:

```
sum_1 = sum(i^4 / (3 + j) for i in 1:20 for j in 1:5)  
println(sum_1)
```

✓ 0.0s

Julia

639215.2833333334

```
sum_2 = sum(i^4 / (3 + i * j) for i in 1:20 for j in 1:5)  
println(sum_2)
```

✓ 0.0s

Julia

89912.02146097136

Рисунок 28: Выполнение задания №11

В результате выполнения данной лабораторной работы я освоил применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.