

# Лабораторная работа №3

## Управляющие структуры

---

Чемоданова Ангелина Александровна

13 сентября 2025

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

- Чемоданова Ангелина Александровна
- Студентка НФИбд-02-22
- Российский университет дружбы народов имени Патриса Лумумбы
- 1132226443@pfur.ru
- <https://github.com/aachemodanova>



## 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.

## Циклы while и for

---

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

Синтаксис while

```
while <условие>  
    <тело цикла>  
end
```

## Циклы while и for

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

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
```

Julia

```
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!  
Hi Marshall, it's great to see you!
```

## Циклы while и for

---

Такие же результаты можно получить при использовании цикла for.

Синтаксис for

```
for <переменная> in <диапазон>
    <тело цикла>
end
```

## Циклы while и for

```
for n in 1:2:10
|   println(n)
end
```

Julia

```
1
3
5
7
9
```

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

Julia

```
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!
Hi Marshall, it's great to see you!
```

Рис. 2: Примеры использования цикла for

## 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
```

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
```

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
```

Рис. 3: Пример использования цикла for для создания двумерного массива

## Циклы while и for

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

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
```

Рис. 4: Пример использования цикла for для создания двумерного массива

Довольно часто при решении задач требуется проверить выполнение тех или иных условий.

Для этого используют условные выражения.

Синтаксис условных выражений с ключевым словом:

```
if <условие 1>
    <действие 1>
elseif <условие 2>
    <действие 2>
else
    <действие 3>
end
```

## Условные выражения

```
N = 1500
# используем "&&" для реализации операции "AND"
# операция % вычисляет остаток от деления
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
```

Julia

FizzBuzz

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

Julia

10

Рис. 5: Примеры использования условного выражения

# ФУНКЦИИ

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

```
sayhi (generic function with 1 method)
```

Julia

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

```
f (generic function with 1 method)
```

Julia

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

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

```
1764
```

Julia

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

```
f2 (generic function with 1 method)
```

Julia

Рис. 6: Примеры способов написания функции

# Функции

```
sayhi2("C-3PO")
f2(42)
```

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
```

Julia

```
#13 (generic function with 1 method)
```

```
sayhi3("C-3PO")
f3(42)
```

Julia

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

```
1764
```

Рис. 7: Примеры способов написания функции

## 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]  
sort(v)  
v
```

Julia

```
3-element Vector{Int64}:  
3  
5  
2
```

```
sort!(v)  
v
```

Julia

```
3-element Vector{Int64}:  
2  
3  
5
```

Рис. 8: Сравнение результатов вывода

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.

# Функции

```
map(f, [1, 2, 3])  
julia  
  
3-element Vector{Int64}:  
1  
4  
9  
  
[f(1), f(2), f(3)]  
julia  
  
3-element Vector{Int64}:  
1  
4  
9  
  
x -> x^3  
map(x -> x^3, [1, 2, 3])  
julia  
  
3-element Vector{Int64}:  
1  
8  
27  
  
f(x) = x^2  
broadcast(f, [1, 2, 3])  
julia  
  
3-element Vector{Int64}:  
1  
4  
9
```

Рис. 9: Примеры использования функций map() и broadcast()

# Функции

```
f.([1, 2, 3])
```

Julia

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

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

Julia

```
3x3 Matrix{Int64}:
 1  2  3
 4  5  6
 7  8  9
```

```
# Вызывает функцию f возведения в квадрат
f(A)
```

Julia

```
3x3 Matrix{Int64}:
 30   36   42
 66   81   96
102  126  150
```

```
B = f.(A)
```

Julia

```
3x3 Matrix{Int64}:
 1   4   9
16  25  36
49  64  81
```

Рис. 10: Примеры использования функций map() и broadcast()

# Функции

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

Julia

```
3x3 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
```

Julia

```
3x3 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)
```

Julia

```
3x3 Matrix{Float64}:
 3.0   6.0   9.0
12.0  15.0  18.0
21.0  24.0  27.0
```

Рис. 11: Примеры использования функций map() и broadcast()

## Сторонние библиотеки (пакеты) в Julia

The screenshot shows a Julia REPL interface with three code blocks and their corresponding outputs:

- Code Block 1:** `using Colors` (Julia)
- Code Block 2:** `palette = distinguishable_colors(100)` (Julia)  
Output: A horizontal bar consisting of 100 vertical color swatches, each a different shade of various colors.
- Code Block 3:** `rand(palette, 3, 3)` (Julia)  
Output: A 3x3 grid of colored squares, where each square is a randomly selected color from the palette.

Рис. 12: Пример использования сторонних библиотек

# Самостоятельная работа

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
```

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
...
98^2 = 9408
99^2 = 9604
100^2 = 10000
```

Рис. 13: Выполнение подпунктов задания №1

# Самостоятельная работа

```
for n in 1:100
|   println(n, "² = ", n²)
end
```

Julia

```
1² = 1
2² = 4
3² = 9
4² = 16
5² = 25
6² = 36
7² = 49
8² = 64
9² = 81
10² = 100
11² = 121
12² = 144
13² = 169
14² = 196
15² = 225
16² = 256
17² = 289
18² = 324
19² = 361
20² = 400
21² = 441
22² = 484
23² = 529
24² = 576
25² = 625
...
97² = 9409
98² = 9604
99² = 9801
100² = 10000
```

Рис. 14: Выполнение подпунктов задания №1

# 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)
```

Julia

```
Dict{Any, Any}(5 => 25, 56 => 3136, 35 => 1225, 55 => 3025, 60 => 3600, 30 => 900, 32 => 1024, 6 => 36, 67 => 4489, 45 => 2025, 73 => 5329, 64 => 4096, 90 => 8100, 4 => 16,
```



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)
```

Julia

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024, 1089, 1156, 1225, 1296
```



Рис. 15: Выполнение подпунктов задания №1

# Самостоятельная работа

№2. Напишите условный оператор, который печатает число, если число чётное, и строку «нечётное», если число нечётное.  
Перепишите код, используя тернарный оператор:

```
n = 33
if n % 2 == 0
|   println(n)
else
|   println("Нечётное")
end
```

Julia

Нечётное

```
n = 52
println(n % 2 == 0 ? n : "Нечётное")
```

Julia

52



№3. Напишите функцию add\_one, которая добавляет 1 к своему входу:

```
function add_one(x)
|   return x + 1
end
println(add_one(15))
```

Julia

16

Рис. 16: Выполнение задания №2 и №3

## Самостоятельная работа

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

```
A = [1 2 3; 1 2 3; -1 -2 -3]
B = map(x -> x + 1, A)
println(B)
```

Julia

```
[2 3 4; 2 3 4; 0 -1 -2]
```

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

Julia

```
[2 3 4; 2 3 4; 0 -1 -2]
```

Рис. 17: Выполнение задания №4

## 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))
```

Julia

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

A

Julia

```
3x3 Matrix{Int64}:
 1   1   3
 5   2   6
 -2  -1  -3
```

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

Julia

```
3x3 Matrix{Int64}:
 1   1   4
 5   2   8
 -2  -1  -4
```

Рис. 18: Выполнение задания №5

## Самостоятельная работа

№6. Создайте матрицу  $B$  с элементами  $Bi1 = 10$ ,  $Bi2 = -10$ ,  $Bi3 = 10$ ,  $i = 1, 2, \dots, 15$ . Вычислите матрицу  $C = B^T B$ :

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

Julia

```
15x3 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)
```

Julia

```
[1500 -1500 1500; -1500 1500 -1500; 1500 -1500 1500]
```

Рис. 19: Выполнение задания №6

# Самостоятельная работа

№7. Создайте матрицу  $Z$  размерности  $6 \times 6$ , 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  $6 \times 6$ :

```
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
            Zi[i, j] = 1
        end
    end
end

println(Z1)

Z2 = copy(Z)
for i in 1:n
    if i<=2
        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)
```

Рис. 20: Выполнение задания №7

# 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)
```

Julia

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

Рис. 21: Выполнение задания №7

# Самостоятельная работа

```
Z1
Julia
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

Z2
Julia
6x6 Matrix{Int64}:
1 0 1 0 0 0
0 1 0 1 0 0
1 0 1 0 1 0
0 1 0 1 0 1
0 0 1 0 1 0
0 0 0 1 0 1

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

Рис. 22: Выполнение задания №7

# Самостоятельная работа

24

Julia

```
6x6 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
```

№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
```

Julia

```
outer (generic function with 1 method)
```

Рис. 23: Выполнение задания №7 и №8

# Самостоятельная работа

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

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

```
A1
```

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
```

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
```

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

Рис. 24: Выполнение подпунктов задания №8

# Самостоятельная работа

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

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
```

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

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
```

Рис. 25: Выполнение подпунктов задания №8

# 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)
```

Julia

```
[-2.000000000000036, 3.000000000000058, 4.99999999999998, 1.999999999999991, -3.999999999999999]
```

#10. Создайте матрицу  $M$  размерности  $6 \times 10$ , элементами которой являются целые числа, выбранные случайным образом с повторениями из совокупности  $1, 2, \dots, 10$ :

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

Julia

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

Рис. 26: Выполнение задания №9 и №10

# 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)
```

Julia

32

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)
```

Julia

[2, 4]

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)
```

Julia

Any[(2, 3), (3, 4), (3, 7), (3, 9)]

Рис. 27: Выполнение подпунктов задания №10

# Самостоятельная работа

№11. Вычислите:

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



Julia

639215.2833333334

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

Julia

89912.02146097136

Рис. 28: Выполнение задания №11

## Выводы

---

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