

# **Лабораторная работа №3**

**Управляющие структуры**

Чемоданова Ангелина Александровна

# Содержание

<b>1 Введение</b>	<b>4</b>
1.1 Цели и задачи . . . . .	4
<b>2 Выполнение лабораторной работы</b>	<b>5</b>
2.1 Циклы while и for . . . . .	5
2.2 Условные выражения . . . . .	7
2.3 Функции . . . . .	8
2.4 Сторонние библиотеки (пакеты) в Julia . . . . .	11
2.5 Самостоятельная работа . . . . .	12
<b>3 Выводы</b>	<b>21</b>
<b>Список литературы</b>	<b>22</b>

# Список иллюстраций

2.1	Примеры использования цикла while . . . . .	5
2.2	Примеры использования цикла for . . . . .	6
2.3	Пример использования цикла for для создания двумерного массива	7
2.4	Пример использования цикла for для создания двумерного массива	7
2.5	Примеры использования условного выражения . . . . .	8
2.6	Примеры способов написания функции . . . . .	8
2.7	Примеры способов написания функции . . . . .	9
2.8	Сравнение результатов вывода . . . . .	9
2.9	Примеры использования функций map() и broadcast() . . . . .	10
2.10	Примеры использования функций map() и broadcast() . . . . .	10
2.11	Примеры использования функций map() и broadcast() . . . . .	11
2.12	Пример использования сторонних библиотек . . . . .	12
2.13	Выполнение подпунктов задания №1 . . . . .	13
2.14	Выполнение подпунктов задания №1 . . . . .	13
2.15	Выполнение подпунктов задания №1 . . . . .	14
2.16	Выполнение задания №2 и №3 . . . . .	14
2.17	Выполнение задания №4 . . . . .	15
2.18	Выполнение задания №5 . . . . .	15
2.19	Выполнение задания №6 . . . . .	16
2.20	Выполнение задания №7 . . . . .	16
2.21	Выполнение задания №7 . . . . .	17
2.22	Выполнение задания №7 . . . . .	17
2.23	Выполнение задания №7 и №8 . . . . .	18
2.24	Выполнение подпунктов задания №8 . . . . .	18
2.25	Выполнение подпунктов задания №8 . . . . .	19
2.26	Выполнение задания №9 и №10 . . . . .	19
2.27	Выполнение подпунктов задания №10 . . . . .	20
2.28	Выполнение задания №11 . . . . .	20

# 1 Введение

## 1.1 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[1].

### Задание

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

## 2 Выполнение лабораторной работы

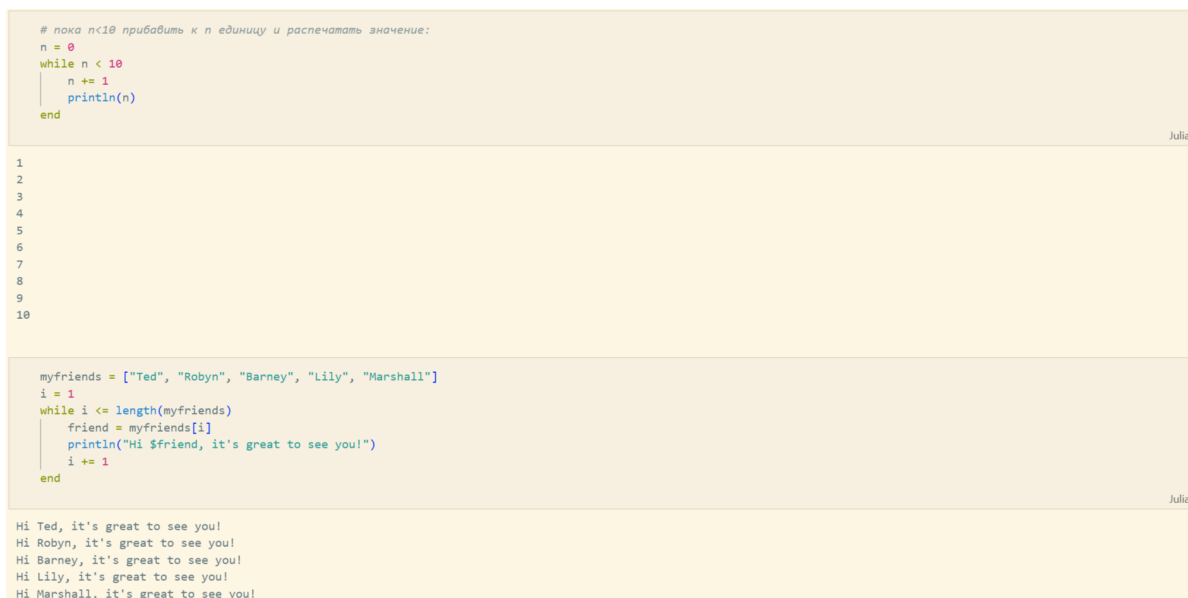
### 2.1 Циклы while и for

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

Синтаксис while

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

Примеры использования цикла while (рис. 2.1):



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

```
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
```

```
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. 2.1: Примеры использования цикла while

Такие же результаты можно получить при использовании цикла `for`.

Синтаксис `for`

**for** <переменная> **in** <диапазон>

<тело цикла>

**end**

Примеры использования цикла `for` (рис. 2.2):

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

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
```

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. 2.2: Примеры использования цикла `for`

Пример использования цикла `for` для создания двумерного массива, в котором значение каждой записи является суммой индексов строки и столбца (рис. 2.3 - рис. 2.4):

```
# 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
```

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
```

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. 2.3: Пример использования цикла for для создания двумерного массива

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

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. 2.4: Пример использования цикла for для создания двумерного массива

## 2.2 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 <условие 1>
    <действие 1>
elseif <условие 2>
    <действие 2>
else
```

<действие 3>

end

Примеры использования условного выражения (рис. 2.5):

<pre>N = 1500 # используем `&amp;&amp;` для реализации операции "AND" # операция % вычисляет остаток от деления if (N % 3 == 0) &amp;&amp; (N % 5 == 0)     println("FizzBuzz")   elseif N % 3 == 0     println("Fizz")   elseif N % 5 == 0     println("Buzz")   else     println(N) end</pre>	Julia
FizzBuzz	
<pre>x = 5 y = 10 (x &gt; y) ? x : y</pre>	Julia
10	

Рис. 2.5: Примеры использования условного выражения

## 2.3 Функции

Julia дает нам несколько разных способов написать функцию.

Примеры способов написания функции (рис. 2.6 - рис. 2.7):

<pre>function sayhi(name)     println("Hi \$name, it's great to see you!") end</pre>	Julia
sayhi (generic function with 1 method)	
<pre># функция возведения в квадрат: function f(x)     x^2 end</pre>	Julia
f (generic function with 1 method)	
<pre>sayhi("C-3PO") f(42)</pre>	Julia
Hi C-3PO, it's great to see you!  1764	
<pre>sayhi2(name) = println("Hi \$name, it's great to see you!") f2(x) = x^2</pre>	Julia
f2 (generic function with 1 method)	

Рис. 2.6: Примеры способов написания функции



<pre>sayhi2("C-3PO") f2(42)</pre>	Julia
<pre>Hi C-3PO, it's great to see you!  1764</pre>	
<pre>sayhi3 = name -&gt; println("Hi \$name, it's great to see you!") f3 = x -&gt; x^2</pre>	Julia
#13 (generic function with 1 method)	
<pre>sayhi3("C-3PO") f3(42)</pre>	Julia
<pre>Hi C-3PO, it's great to see you!  1764</pre>	

Рис. 2.7: Примеры способов написания функции

По соглашению в Julia функции, сопровождаемые восклицательным знаком, изменяют свое содержимое, а функции без восклицательного знака не делают этого (рис. 2.8):

<pre># задаём массив v: v = [3, 5, 2] sort(v) v</pre>	Julia
<pre>3-element Vector{Int64}:  3  5  2</pre>	
<pre>sort!(v) v</pre>	Julia
<pre>3-element Vector{Int64}:  2  3  5</pre>	

Рис. 2.8: 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.

Примеры использования функций `map()` и `broadcast()` (рис. 2.9 - рис. 2.11):

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

Рис. 2.9: Примеры использования функций `map()` и `broadcast()`

```
f.([1, 2, 3])  
  
3-element Vector{Int64}:  
 1  
 4  
 9  
  
# Задаём матрицу A:  
A = [i + 3*j for j in 0:2, i in 1:3]  
  
3×3 Matrix{Int64}:  
 1  2  3  
 4  5  6  
 7  8  9  
  
# Вызываем функцию f возведения в квадрат  
f(A)  
  
3×3 Matrix{Int64}:  
 30  36  42  
 66  81  96  
 102 126 150  
  
B = f.(A)  
  
3×3 Matrix{Int64}:  
 1  4  9  
 16 25 36  
 49 64 81
```

Рис. 2.10: Примеры использования функций `map()` и `broadcast()`

```
A .* 2 .* f.(A) ./ A
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
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)
3x3 Matrix{Float64}:
 3.0  6.0  9.0
12.0 15.0 18.0
21.0 24.0 27.0
```

Рис. 2.11: Примеры использования функций `map()` и `broadcast()`

## 2.4 Сторонние библиотеки (пакеты) в Julia

Julia имеет более 2000 зарегистрированных пакетов, что делает их огромной частью экосистемы Julia. Есть вызовы функций первого класса для других языков, обеспечивающие интерфейсы сторонних функций. Можно вызвать функции из Python или R, например, с помощью `PyCall` или `Rcall`.

С перечнем доступных в Julia пакетов можно ознакомиться на страницах следующих ресурсов: - <https://julialang.org/packages/> - <https://juliahub.com/ui/Home> - <https://juliaobserver.com/> - <https://github.com/svaksha/Julia.jl>

При первом использовании пакета в вашей текущей установке Julia вам необходимо использовать менеджер пакетов, чтобы явно его добавить:

```
import Pkg
Pkg.add("Example")
```

При каждом новом использовании Julia (например, в начале нового сеанса в REPL или открытии блокнота в первый раз) нужно загрузить пакет, используя ключевое слово `using`:

Например, добавим и загрузим пакет `Colors`:

```
Pkg.add("Colors")  
using Colors
```

Затем создадим палитру из 100 разных цветов:

```
palette = distinguishable_colors(100)
```

А затем определим матрицу  $3 \times 3$  с элементами в форме случайного цвета из палитры, используя функцию `rand`:

```
rand(palette, 3, 3)
```

Пример использования сторонних библиотек (рис. 2.12):

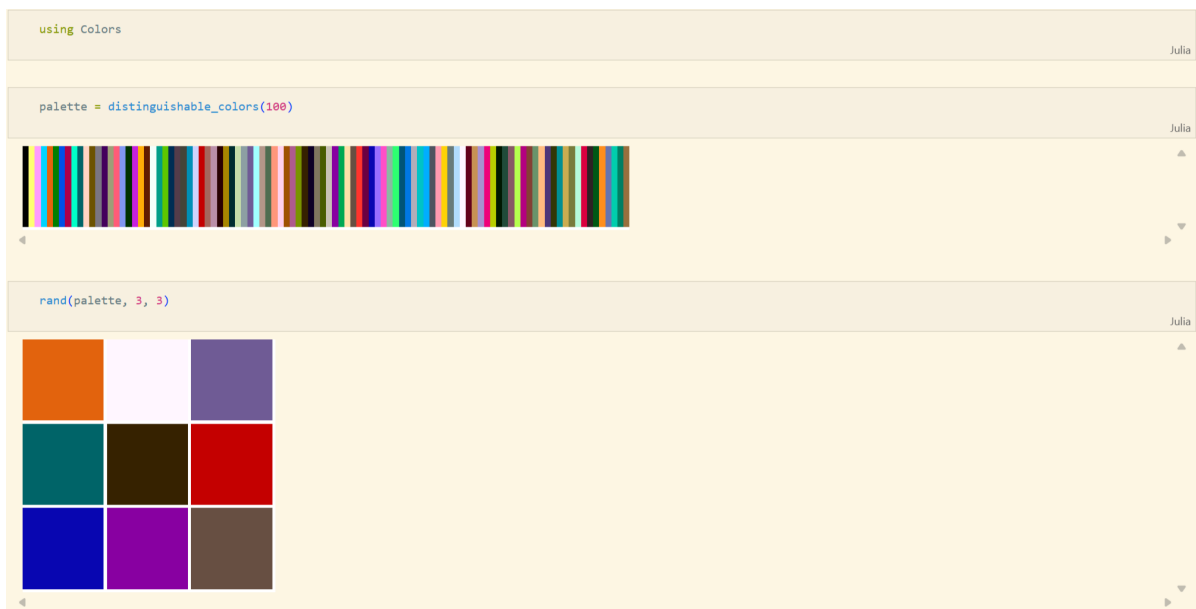


Рис. 2.12: Пример использования сторонних библиотек

## 2.5 Самостоятельная работа

Выполнение задания №1 (рис. 2.13 - рис. 2.15):

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
```

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

Julia

Рис. 2.13: Выполнение подпунктов задания №1

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

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

Julia

Рис. 2.14: Выполнение подпунктов задания №1

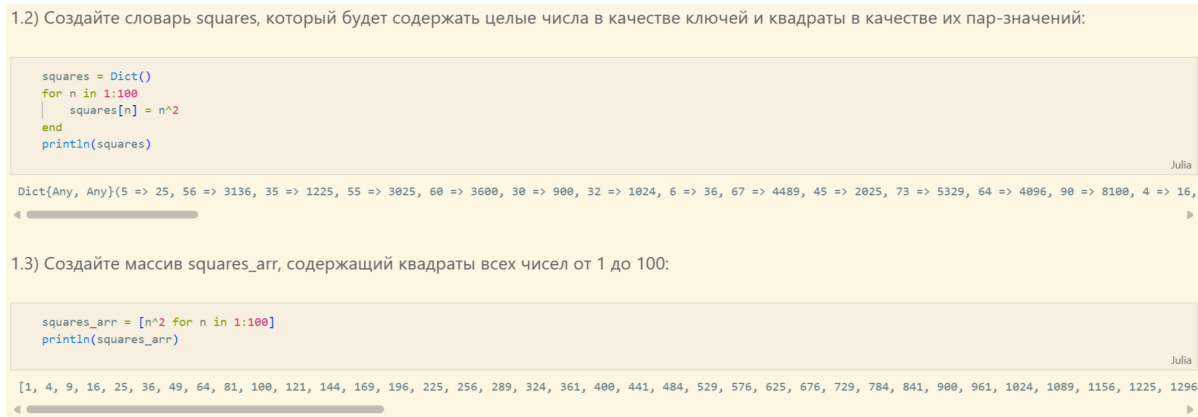


Рис. 2.15: Выполнение подпунктов задания №1

Выполнение задания №2 и №3 (рис. 2.16):

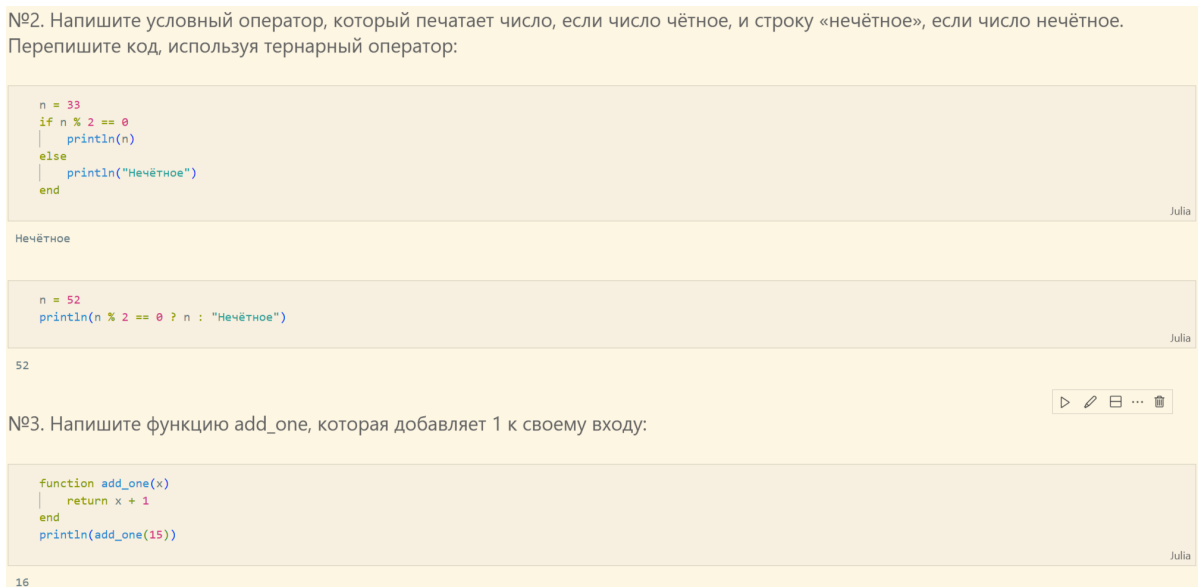


Рис. 2.16: Выполнение задания №2 и №3

Выполнение задания №4 (рис. 2.17):

№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. 2.17: Выполнение задания №4

Выполнение задания №5 (рис. 2.18):

№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. 2.18: Выполнение задания №5

Выполнение задания №6 (рис. 2.19):

№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)
```

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
10 -10 10
```

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

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

Рис. 2.19: Выполнение задания №6

Выполнение задания №7 (рис. 2.20 - рис. 2.23):

№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
            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)
```

Рис. 2.20: Выполнение задания №7



```

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 1; 0 0 0 0 1 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 1]
[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. 2.21: Выполнение задания №7

Z1	Julia
<pre> 6×6 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 </pre>	
Z2	Julia
<pre> 6×6 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 </pre>	
Z3	Julia
<pre> 6×6 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 </pre>	

Рис. 2.22: Выполнение задания №7

Z4

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. 2.23: Выполнение задания №7 и №8

Выполнение задания №8 (рис. 2.24 - рис. 2.25):

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

Julia

```

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

```

```

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

```

```

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

```

```

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. 2.24: Выполнение подпунктов задания №8

```

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

10x10 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

9x9 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. 2.25: Выполнение подпунктов задания №8

Выполнение задания №9 и №10(рис. 2.26):

№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)

[-2.0000000000000036, 3.0000000000000058, 4.999999999999998, 1.9999999999999991, -3.999999999999999]

```

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

```

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

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. 2.26: Выполнение задания №9 и №10

Выполнение задания №10 (рис. 2.27):

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. 2.27: Выполнение подпунктов задания №10

Выполнение задания №11 (рис. 2.28):

№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. 2.28: Выполнение задания №11

## 3 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.

## Список литературы

1. JuliaLang [Электронный ресурс]. 2025 JuliaLang.org contributors. URL: <https://julialang.org/> (дата обращения: 09.13.2025).
2. Julia 1.11 Documentation [Электронный ресурс]. 2025 JuliaLang.org contributors. URL: <https://docs.julialang.org/en/v1/> (дата обращения: 09.13.2025).