

Повторение примеров

Циклы `while` и `for`

```
In [1]: n = 0
while n < 10
    n += 1
    println(n)
end
```

1
2
3
4
5
6
7
8
9
10

```
In [2]: 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!

```
In [3]: for n in 1:10
    println(n)
end
```

1
2
3
4
5
6
7
8
9
10

```
In [4]: 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!

```
In [5]: # 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
```

```
Out[5]: 5×5 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
```

```
In [6]: # 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

```

```

Out[6]: 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

```

```

In [7]: C = [i + j for i in 1:m, j in 1:n]

```

```

Out[7]: 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

```

Условные выражения

```

In [8]: # используем `&&` для реализации операции "AND"
# операция % вычисляет остаток от деления
function check_(N)
    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
end
check_(3), check_(5), check_(15), check_(2)

```

```

Fizz
Buzz
FizzBuzz
2

```

```

Out[8]: (nothing, nothing, nothing, nothing)

```

```

In [9]: x = 5
y = 10
(x > y) ? x : y

```

```

Out[9]: 10

```

Функции

```

In [10]: function sayhi(name)
           println("Hi $name, it's great to see you!")
       end
# функция возведения в квадрат:
function f(x)
    x^2
end

```

```

Out[10]: f (generic function with 1 method)

```

```

In [11]: sayhi("C-3P0"), f(42)

```

```

Hi C-3P0, it's great to see you!

```

```

Out[11]: (nothing, 1764)

```

```

In [12]: sayhi2(name) = println("Hi $name, it's great to see you!")
f2(x) = x^2

```

```

Out[12]: f2 (generic function with 1 method)

```

```

In [13]: sayhi3 = name -> println("Hi $name, it's great to see you!")
f3 = x -> x^2

```

```

Out[13]: #7 (generic function with 1 method)

```

```

In [14]: v = [3, 5, 2]
sort(v)

```

```
v
```

```
Out[14]: 3-element Vector{Int64}:  
 3  
 5  
 2
```

```
In [15]: sort!(v)  
v
```

```
Out[15]: 3-element Vector{Int64}:  
 2  
 3  
 5
```

```
In [16]: map(f, [1, 2, 3])
```

```
Out[16]: 3-element Vector{Int64}:  
 1  
 4  
 9
```

```
In [17]: broadcast(f, [1, 2, 3])
```

```
Out[17]: 3-element Vector{Int64}:  
 1  
 4  
 9
```

```
In [18]: f.([1, 2, 3])
```

```
Out[18]: 3-element Vector{Int64}:  
 1  
 4  
 9
```

```
In [19]: A = [i + 3*j for j in 0:2, i in 1:3]
```

```
Out[19]: 3×3 Matrix{Int64}:  
 1  2  3  
 4  5  6  
 7  8  9
```

```
In [20]: f(A)
```

```
Out[20]: 3×3 Matrix{Int64}:  
 30  36  42  
 66  81  96  
 102 126 150
```

```
In [21]: f.(A)
```

```
Out[21]: 3×3 Matrix{Int64}:  
 1  4  9  
 16 25 36  
 49 64 81
```

```
In [22]: A .+ 2 .* f.(A) ./ A
```

```
Out[22]: 3×3 Matrix{Float64}:  
 3.0  6.0  9.0  
 12.0 15.0 18.0  
 21.0 24.0 27.0
```

```
In [23]: @. A + 2 * f(A) / A
```

```
Out[23]: 3×3 Matrix{Float64}:  
 3.0  6.0  9.0  
 12.0 15.0 18.0  
 21.0 24.0 27.0
```

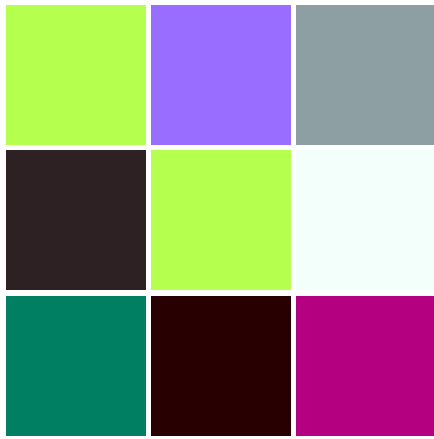
```
In [24]: broadcast(x -> x + 2 * f(x) / x, A)
```

```
Out[24]: 3×3 Matrix{Float64}:  
 3.0  6.0  9.0  
 12.0 15.0 18.0  
 21.0 24.0 27.0
```

Сторонние библиотеки (пакеты) в Julia

```
In [25]: using Colors  
palette = distinguishable_colors(100)  
rand(palette, 3, 3)
```

Out[25]:



Самостоятельная работа

1. Используя циклы while и for:

- выведите на экран целые числа от 1 до 100 и напечатайте их квадраты;

```
In [26]: for i in 1:100
          print(i, " ")
        end
        println()
        for i in 1:100
          print(i^2, " ")
        end
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 4
1 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78
79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
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 1369 1444 1521 1600 1681 1764 1849 1936 2025 2116 2209 2304 2401 2500 2601 2704 2809 29
16 3025 3136 3249 3364 3481 3600 3721 3844 3969 4096 4225 4356 4489 4624 4761 4900 5041 5184 5329 5476 5625 5776
5929 6084 6241 6400 6561 6724 6889 7056 7225 7396 7569 7744 7921 8100 8281 8464 8649 8836 9025 9216 9409 9604 98
01 10000
```

```
In [50]: i = 1
        while i <= 100
          print(i, " ")
          i += 1
        end
        println()
        i = 0
        while i < 100
          i += 1
          print(i^2, " ")
        end
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 4
1 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78
79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
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 1369 1444 1521 1600 1681 1764 1849 1936 2025 2116 2209 2304 2401 2500 2601 2704 2809 29
16 3025 3136 3249 3364 3481 3600 3721 3844 3969 4096 4225 4356 4489 4624 4761 4900 5041 5184 5329 5476 5625 5776
5929 6084 6241 6400 6561 6724 6889 7056 7225 7396 7569 7744 7921 8100 8281 8464 8649 8836 9025 9216 9409 9604 98
01 10000
```

- создайте словарь squares, который будет содержать целые числа в качестве ключей и квадраты в качестве их пар-значений;

```
In [55]: squares = Dict{i => i^2 for i in 2:100}
        squares2 = Dict{2 => 4}
        i = 3
        while i <= 100
          setindex!(squares2, i, i^2)
          i += 1
        end
        squares, squares2
```

```
Out[55]: (Dict{5 => 25, 56 => 3136, 35 => 1225, 55 => 3025, 60 => 3600, 30 => 900, 32 => 1024, 6 => 36, 67 => 4489, 45 =
> 2025...}, Dict{3364 => 58, 9604 => 98, 3025 => 55, 2809 => 53, 1024 => 32, 1296 => 36, 6889 => 83, 784 => 28, 4
900 => 70, 196 => 14...})
```

- создайте массив `squares_arr`, содержащий квадраты всех чисел от 1 до 100.

```
In [56]: squares_arr = [i^2 for i in 1:100]
squares_arr2 = zeros(100)
i = 1
while i <= 100
    squares_arr2[i] = i^2
    i += 1
end
hcat(squares_arr, squares_arr2)
```

```
Out[56]: 100x2 Matrix{Float64}:
 1.0  1.0
 4.0  4.0
 9.0  9.0
16.0 16.0
25.0 25.0
36.0 36.0
49.0 49.0
64.0 64.0
81.0 81.0
100.0 100.0
121.0 121.0
144.0 144.0
169.0 169.0
⋮
7921.0 7921.0
8100.0 8100.0
8281.0 8281.0
8464.0 8464.0
8649.0 8649.0
8836.0 8836.0
9025.0 9025.0
9216.0 9216.0
9409.0 9409.0
9604.0 9604.0
9801.0 9801.0
10000.0 10000.0
```

2. Напишите условный оператор, который печатает число, если число чётное, и строку «нечётное», если число нечётное.

```
In [29]: function chet(x)
    if x%2 == 0
        println(x)
    else
        println("нечётное")
    end
end
chet(2), chet(3)
```

```
2
нечётное
```

```
Out[29]: (nothing, nothing)
```

Перепишите код, используя тернарный оператор.

```
In [30]: function chet2(x)
    x%2 == 0 ? println(x) : println("нечётное")
end
chet2(2), chet2(3)
```

```
2
нечётное
```

```
Out[30]: (nothing, nothing)
```

3. Напишите функцию `add_one`, которая добавляет 1 к своему входу.

```
In [31]: function add_one(x)
    return x+1
end
add_one(5)
```

```
Out[31]: 6
```

4. Используйте `map()` или `broadcast()` для задания матрицы A ,

каждый элемент которой увеличивается на единицу по сравнению с предыдущим.

```
In [32]: A = zeros(Int64, 3, 5)
t = map!(x -> x+1, A, 0:14); display(A)
t = broadcast(add_one, reshape(0:14, 3, 5))
```

```
3x5 Matrix{Int64}:
 1  4  7 10 13
 2  5  8 11 14
 3  6  9 12 15
```

```
Out[32]: 3x5 Matrix{Int64}:
 1  4  7 10 13
 2  5  8 11 14
 3  6  9 12 15
```

5. Задайте матрицу A следующего вида:

$$A = \begin{pmatrix} 1 & 1 & 3 \\ 5 & 2 & 6 \\ -2 & -1 & -3 \end{pmatrix}$$

```
In [33]: A = [1 1 3; 5 2 6; -2 -1 -3]
```

```
Out[33]: 3x3 Matrix{Int64}:
 1  1  3
 5  2  6
-2 -1 -3
```

- Найдите A^3

```
In [34]: display(A^2)
display(A^3)
```

```
3x3 Matrix{Int64}:
 0  0  0
 3  3  9
-1 -1 -3
3x3 Matrix{Int64}:
 0  0  0
 0  0  0
 0  0  0
```

- Замените третий столбец матрицы A на сумму 2-го и 3-го столбцов

```
In [35]: A[:, 3] = A[:, 2] + A[:, 3]
A
```

```
Out[35]: 3x3 Matrix{Int64}:
 1  1  4
 5  2  8
-2 -1 -4
```

6. Создайте матрицу B с элементами

$$B_{i1} = 10, B_{i2} = -10, B_{i3} = 10, \quad i = 1, 2, \dots, 15.$$

```
In [36]: B = fill{10, (15,3)}
B[:, 2] = -B[:,2]
B
```

```
Out[36]: 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
```

Вычислите матрицу $C = B^T B$.

```
In [37]: C = B' * B
```

```
Out[37]: 3x3 Matrix{Int64}:
 1500 -1500 1500
-1500 1500 -1500
 1500 -1500 1500
```

7. Создайте матрицу Z размерности 6×6 , все элементы которой равны нулю, и матрицу E , все элементы которой равны 1.

```
In [38]: Z = zeros{Int64, 6,6}; display(Z)
E = ones{Int64, 6,6}
```

```
6x6 Matrix{Int64}:
 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
```

```
Out[38]: 6x6 Matrix{Int64}:
 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
```

Используя цикл `while` или `for` и закономерности расположения элементов, создайте следующие матрицы размерности 6×6 :

$$Z_1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad Z_2 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix},$$

$$Z_3 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \quad Z_4 = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

```
In [39]: Z1, Z2, Z3, Z4 = zeros{Int64, 6, 6}, zeros{Int64, 6, 6}, zeros{Int64, 6, 6}, zeros{Int64, 6, 6}
for i in 1:6, j in 1:6
    if i == j+1 || i == j-1
        Z1[i,j] = 1
    end
    if i == j || i == j+2 || i == j-2
        Z2[i,j] = 1
    end
end
```

```

    if i == 7-j || i == 5-j || i == 9-j
        Z3[i,j] = 1
    end
    if (i+j)%2 == 0
        Z4[i,j] = 1
    end
end
display(Z1); display(Z2); display(Z3); display(Z4)

```

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

```

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

```

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

```

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).

- Напишите свою функцию, аналогичную функции outer() языка R. Функция должна иметь следующий интерфейс:

`outer(x,y,operation)`. Таким образом, функция вида `outer(A,B,*)` должна быть эквивалентна произведению матриц A и B размерностями $L \times M$ и $M \times N$ соответственно, где элементы результирующей матрицы C имеют вид $C_{ij} = \sum_{k=1}^M A_{ik} B_{kj}$ (или в тензорном виде $C_i^j = \sum_{k=1}^M A_k^i B_j^k$)

```

In [40]: function outer(A, B, operation)
    if size(A)[2] != size(B)[1]
        println("Size incompatibility! A's amount of columns DOES NOT equal B's amount of rows")
        return
    end
    answ = zeros(size(A)[1], size(B)[2])
    for i in 1:size(A)[1], j in 1:size(B)[2]
        answ[i,j] = sum(operation(A[i,k], B[k,j]) for k in 1:size(A)[2])
    end
    return answ
end
mtrx1, mtrx2 = rand(1:10, 4, 6), rand(1:10, 6, 3); display(mtrx1); display(mtrx2)
display(outer(mtrx1, mtrx1', *))
display(outer(mtrx2', mtrx2, +))
display(outer(mtrx1, mtrx2, %))
display(outer(mtrx1, mtrx2, /))
display(outer(mtrx1, mtrx2, -))
display(outer(mtrx1, mtrx2, div))

```

4x6 Matrix{Int64}:

```

10  4  7  10  4  1
 2  2  8  10  9  1
 6  7  1  1  2 10
 4  2  7  7  4  2

```

6x3 Matrix{Int64}:

```

1  3  1
5  3  1
9  7  5
5  1  2
2  6  8
8 10  1

```



```

4x4 Matrix{Float64}:
282.0 221.0 123.0 185.0

221.0 254.0 72.0 176.0
123.0 72.0 191.0 80.0
185.0 176.0 80.0 138.0
3x3 Matrix{Float64}:
60.0 60.0 48.0
60.0 60.0 48.0
48.0 48.0 36.0
4x3 Matrix{Float64}:
12.0 7.0 6.0
12.0 9.0 4.0
6.0 4.0 4.0
13.0 9.0 7.0
4x3 Matrix{Float64}:
15.7028 16.4333 21.9
9.91389 14.0762 12.725
9.96111 6.80952 23.95
8.82778 10.8667 13.4
4x3 Matrix{Float64}:
6.0 6.0 18.0
2.0 2.0 14.0
-3.0 -3.0 9.0
-4.0 -4.0 8.0
4x3 Matrix{Float64}:
14.0 15.0 21.0
8.0 12.0 12.0
9.0 6.0 23.0
7.0 9.0 12.0

```

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

$$A_1 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 4 & 8 & 16 & 32 \\ 3 & 9 & 27 & 81 & 243 \\ 4 & 16 & 64 & 256 & 1024 \end{pmatrix},$$

$$A_3 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 0 \\ 2 & 3 & 4 & 0 & 1 \\ 3 & 4 & 0 & 1 & 2 \\ 4 & 0 & 1 & 2 & 3 \end{pmatrix}, \quad A_4 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix},$$

$$A_5 = \begin{pmatrix} 0 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 1 & 0 & 8 & 7 & 6 & 5 & 4 & 3 & 2 \\ 2 & 1 & 0 & 8 & 7 & 6 & 5 & 4 & 3 \\ 3 & 2 & 1 & 0 & 8 & 7 & 6 & 5 & 4 \\ 4 & 3 & 2 & 1 & 0 & 8 & 7 & 6 & 5 \\ 5 & 4 & 3 & 2 & 1 & 0 & 8 & 7 & 6 \\ 6 & 5 & 4 & 3 & 2 & 1 & 0 & 8 & 7 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & 8 \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{pmatrix}.$$

В каждом случае ваше решение должно быть легко обобщаемым на случай создания матриц большей размерности, но той же структуры.

```

In [59]: n = 5
A1 = Int64.(outer(hcat([i//n for i in 0:n-1] for j in 1:n)...), hcat([i//n for i in 0:n-1] for j in 1:n)...)'
display(A1)
t2 = hcat([j-1 for i in 1:n] for j in 1:n)...)'
A2 = Int64.(outer(t2, hcat([i==j ? i : 1 for i in 1:n] for j in 1:n)...), ^)-hcat([(n-1)*i for i in 0:n-1] for j in 1:n)...)'
display(A2)
#=
Матрицы A3 и A4 аналогичны по своей структуре первой матрице, единственное, чем они отличаются - каждый элемент
его предыдущего значения на сторону матрицы
=#
t3 = outer(hcat([i//n for i in 0:n-1] for j in 1:n)...), hcat([i//n for i in 0:n-1] for j in 1:n)...)'
A3 = Int64.(outer(t3, hcat([i==j ? n/1 : 1.0 for i in 1:n] for j in 1:n)...), mod))

```

```

display(A3)
n = 10
t4 = outer(hcat([[i//n for i in 0:n-1] for j in 1:n]...), hcat([[i//n for i in 0:n-1] for j in 1:n]...)', +)
A4 = Int64.(outer(t4, hcat([[i==j ? n/1 : 1.0 for i in 1:n] for j in 1:n]...), mod)) # Остаток от деления на n
display(A4)
n = 9
t5 = outer(hcat([[i==j-1 ? j-1 : 0 for i in 0:n-1] for j in 1:n]...), hcat([[i==j-1 ? (n-i) : 0 for i in 0:n-1]
A5 = Int64.(outer(round.(t5), hcat([[i==j ? n/1 : 1.0 for i in 1:n] for j in 1:n]...), mod)) # Остаток от деления
display(A5)

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
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
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
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
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

```

9. Решите следующую систему линейных уравнений с 5 неизвестными

$$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 = 7, \\ 2x_1 + x_2 + 2x_3 + 3x_4 + 4x_5 = -1, \\ 3x_1 + 2x_2 + x_3 + 2x_4 + 3x_5 = -3, \\ 4x_1 + 3x_2 + 2x_3 + x_4 + 2x_5 = 5, \\ 5x_1 + 4x_2 + 3x_3 + 2x_4 + x_5 = 17, \end{cases}$$

рассмотрев соответствующее матричное уравнение $Ax = y$. Обратите внимание на особый вид матрицы A . Метод, используемый для решения данной системы уравнений, должен быть легко обобщаем на случай большего числа уравнений, где матрица A будет иметь такую же структуру.

Решение будет осуществлено методом Гаусса.

```

In [42]: function gauss_method(mtrx, vec)
    if size(mtrx)[1] != size(mtrx)[2]
        println("Size incompatibility! ", size(mtrx)[1] >= size(mtrx)[2] ? "More" : "Less", " equations than va
        return
    end
    if size(mtrx)[1] != size(vec)[1]
        println("Size incompatibility! ", size(mtrx)[1] >= size(vec)[1] ? "Less" : "More", " answers than equat
        return
    end
    n = size(mtrx)[1]
    max_el, max_row = 0, 0
    for i in 1:n
        max_el = abs(mtrx[i, i])
    end

```

```

max_row = i
for k in i:n
    if abs(mtrx[k, k]) > max_el
        max_el = abs(mtrx[k, k])
        max_row = k
    end
end
mtrx[i, :], mtrx[max_row, :] = mtrx[max_row, :], mtrx[i, :]
vec[i], vec[max_row] = vec[max_row], vec[i]

for k in i+1:n
    c = -mtrx[k, i] / mtrx[i, i]
    mtrx[k, :] = mtrx[k, :].+(c.*mtrx[i, :])
    mtrx[k, i] = 0
    vec[k] += c*vec[i]
end
end
answ = zeros(n)
for i in n:-1:1
    answ[i] = vec[i] / mtrx[i, i]
    for k in i-1:-1:1
        vec[k] -= mtrx[k, i] * answ[i]
    end
end
return answ
end

```

Out[42]: gauss_method (generic function with 1 method)

```

In [43]: m = 5
A = hcat([abs(i-j)+1 for j in 1:m for i in 1:m]...); display(A)
y = [7, -1, -3, 5, 17]; display(y)
x = gauss_method(A, y)

```

5x5 Matrix{Int64}:

```

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

```

5-element Vector{Int64}:

```

7
-1
-3
5
17

```

Out[43]: 5-element Vector{Float64}:

```

-2.0
3.0
5.0
2.0
-4.0

```

10. Создайте матрицу M размерности 6×10 , элементами которой являются целые числа, выбранные случайным образом с повторениями из совокупности 1, 2, ..., 10.

```

In [73]: M = rand(1:10, 6, 10)

```

Out[73]: 6x10 Matrix{Int64}:

```

9 8 10 9 8 4 7 9 10 9
5 1 3 5 3 4 6 8 7 9
7 2 8 5 1 5 8 10 1 3
1 9 3 7 3 10 6 5 6 9
7 7 10 10 4 6 10 6 2 8
1 10 10 7 5 6 6 3 8 10

```

- Найдите число элементов в каждой строке матрицы M , которые больше числа N (например, $N = 4$).

```

In [74]: N = 4
[size(findall(x -> x>N, M[i, :]))[1] for i in 1:6]

```

Out[74]: 6-element Vector{Int64}:

```

9
6
6
7
8
8

```

- Определите, в каких строках матрицы M число T (например, $T = 7$) встречается ровно 2 раза?

```
In [75]: T = 7
findall(x -> x==2, [size(findall(x -> x==T, M[i, :]))[1] for i in 1:6])
```

```
Out[75]: 1-element Vector{Int64}:
5
```

- Определите все пары столбцов матрицы M , сумма элементов которых больше K (например, $K = 75$).

```
In [76]: K = 75
findall(x -> x>K, hcat([i==j ? -1 : sum(M[:, i])+sum(M[:, j]) for j in 1:6] for i in 1:6)...))
```

```
Out[76]: 10-element Vector{CartesianIndex{2}}:
 CartesianIndex{3, 2}
 CartesianIndex{4, 2}
 CartesianIndex{2, 3}
 CartesianIndex{4, 3}
 CartesianIndex{6, 3}
 CartesianIndex{2, 4}
 CartesianIndex{3, 4}
 CartesianIndex{6, 4}
 CartesianIndex{3, 6}
 CartesianIndex{4, 6}
```

11. Вычислите:

- $$\sum_{i=1}^{20} \sum_{j=1}^5 \frac{i^4}{(3+j)}$$

```
In [48]: sum([sum([i^4/(3+j)] for j in 1:5) for i in 1:20])
```

```
Out[48]: 1-element Vector{Float64}:
639215.2833333334
```

- $$\sum_{i=1}^{20} \sum_{j=1}^5 \frac{i^4}{(3+ij)}$$

```
In [49]: sum([sum([i^4/(3+i*j)] for j in 1:5) for i in 1:20])
```

```
Out[49]: 1-element Vector{Float64}:
89912.02146097136
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js