Лабораторная работа № 3

Дисциплина: Компьютерный практику по статистическому анализу данных

Покрас Илья Михайлович

Содержание

Список иллюстраций

1 Цель работы

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

2 Выполнение лабораторной работы

Выполнение пункта 1(2.1):

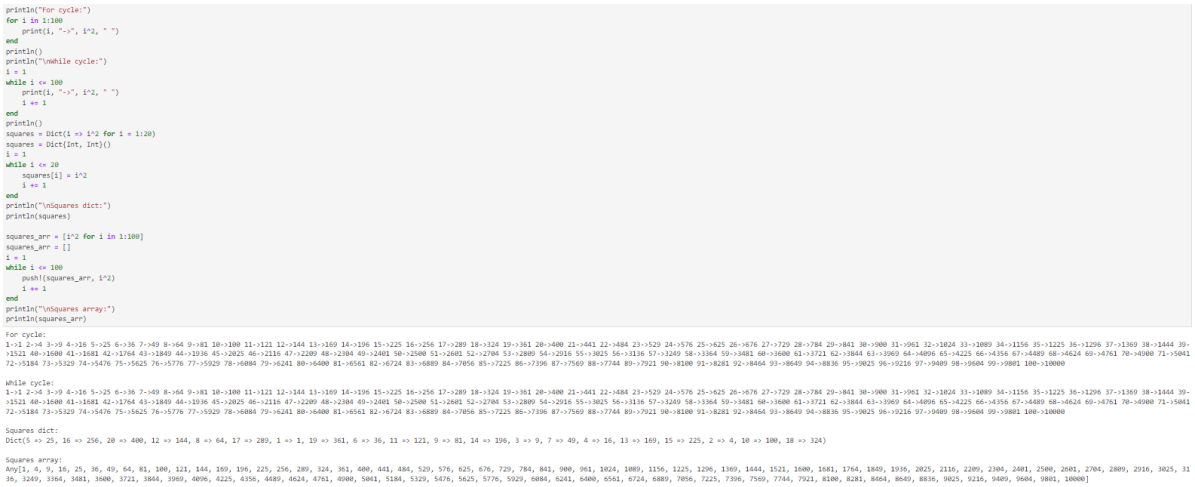

Рис. 2.1: Код выполнения пункта 1

Выполнение пункта 2(2.2):

```
a = parse(Int, readline())
if isodd(a)
    println("Нечётное")
else
    println(a)
end
```

```
stdin>  3
Нечётное
```

Рис. 2.2: Код выполнения пункта 2

Выполнение пункта 3(2.3):

```
function add_one(numb)
    println(numb + 1)
end

for i in 1:3
    a = parse(Int, readline())
    add_one(a)
end
```

```
stdin>  2
3
stdin>  1
2
stdin>  4
5
```

Рис. 2.3: Код выполнения пункта 3

Выполнение пункта 4(2.4):

```
println("Matrix1:")
matrix1 = map(x -> x, reshape(Array(1:20), 4, 5))
display(matrix1)

println("\nMatrix2:")
matrix2 = broadcast(x -> x, reshape(Array(1:20), 4, 5))
display(matrix2)
```

```
Matrix1:
4×5 Matrix{Int64}:
 1  5   9  13  17
 2  6  10  14  18
 3  7  11  15  19
 4  8  12  16  20

Matrix2:
4×5 Matrix{Int64}:
 1  5   9  13  17
 2  6  10  14  18
 3  7  11  15  19
 4  8  12  16  20
```

Рис. 2.4: Код выполнения пункта 4

Выполнение пункта 5(2.5):

```
A = [1 1 3; 5 2 6; -2 -1 -3]
println("Matrix:")
display(A)

println("\nMatrix^3:")
display(A.^3)

for i in 1:3
    A[i, 3] = A[i, 1] + A[i, 2]
end

println("\nMatrix with sum:")
display(A)
```

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

Matrix^3:
3×3 Matrix{Int64}:
   1   1   27
 125   8  216
  -8  -1  -27

Matrix with sum:
3×3 Matrix{Int64}:
  1   1   2
  5   2   7
 -2  -1  -3
```

Рис. 2.5: Код выполнения пункта 5

Выполнение пункта 6(2.6):

```
B = zeros(15, 3)

for i in 1:15
    B[i,1] = 10
    B[i,2] = -10
    B[i,3] = 10
end

Bt = transpose(B)
C = Bt * B

println("\nMuliplication matrix:")
display(C)
```

```
Muliplication matrix:
3×3 Matrix{Float64}:
  1500.0  -1500.0   1500.0
 -1500.0   1500.0  -1500.0
  1500.0  -1500.0   1500.0
```

Рис. 2.6: Код выполнения пункта 6

Выполнение пункта 7(2.7):

```
matrix1 = [[if i-j in [1, -1] 1 else 0 end for j in 1:6] for i in 1:6]
hcat(matrix1...)
println("Z1:")
display(matrix1)

matrix2 = [[if i-j in [2, 0, -2] 1 else 0 end for j in 1:6] for i in 1:6]
hcat(matrix2...)
println("\nZ2:")
display(matrix2)

matrix3 = [[if i+j in [5, 7, 9] 1 else 0 end for j in 1:6] for i in 1:6]
hcat(matrix3...)
println("\nZ3:")
display(matrix3)

matrix4 = [[if i-j in [4, 2, 0, -2, -4] 1 else 0 end for j in 1:6] for i in 1:6]
hcat(matrix4...)
println("\nZ4:")
display(matrix4)
```

```
Z1:
6-element Vector{Vector{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:
6-element Vector{Vector{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:
6-element Vector{Vector{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]

Z4:
6-element Vector{Vector{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]
```

Рис. 2.7: Код выполнения пункта 7

Выполнение пункта 8(2.8):

```
outer(x, y, operation) = transpose(hcat([[sum(operation(x[i, k], y[k, j]) for k in 1:size(x)[2]) for j in 1:size(y)[2]] for i in 1:size(x)[1]]...))

B = reshape(rand(1:5, 12), 2, 6)
A = reshape(rand(1:5, 12), 6, 2)
println("Source matrices:")
display(A)
display(B)
println("\nSum matrix:")
display(outer(A, B, +))

println("\nSub matrix:")
display(outer(A, B, -))

println("\nMul matrix:")
display(outer(A, B, *))

println("\nDiv matrix:")
display(outer(A, B, /))

println("\nPow matrix:")
display(outer(A, B, ^))

println("\nA1:")
display(outer(reshape(0:4, 5, 1), reshape(0:4, 1, 5), +))

println("\nA2:")
println(outer(reshape(0:4, 5, 1), reshape(1:5, 1, 5), ^))

println("\nA3:")
display(outer(hcat([[if i==j 1 else 0 end for j in 0:4] for i in 0:4]...), hcat([Vector(i:i+4).%5 for i in 0:4]...), *))

println("\nA4:")
display(outer(hcat([[if i==j 1 else 0 end for j in 0:9] for i in 0:9]...), hcat([Vector(i:i+9).%10 for i in 0:9]...), *))

println("\nA5:")
display(outer(hcat([[if i==j 1 else 0 end for j in 0:8] for i in 0:8]...), hcat([Vector(i+9:-1:i+1).%9 for i in 0:8]...), *))
```

Рис. 2.8: Код выполнения пункта 8

Выполнение пункта 9(2.9):

```
M = [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]
N = [7, -1, -3, 5, 17]
println("Solution Vector: ")
display(M \ N)
```

```
Solution Vector:
5-element Vector{Float64}:
 -2.0000000000000036
  3.0000000000000058
  4.999999999999998
  1.9999999999999991
 -3.999999999999999
```

Рис. 2.9: Код выполнения пункта 9

Выполнение пункта 10(2.10):

```
matrix = rand(1:10, 6, 10)
println("Matrix: ")
display(matrix)
num = 5

ans_vec1 = []
for i in 1:size(matrix)[1]
    counter = 0
    for j in 1:size(matrix)[2]
            if matrix[i, j] > num
                counter = counter +1
            end
    end
    push!(ans_vec1, counter)
end
println("The amount of numbers, greater than N: ")
println(ans_vec1)


ans_vec2 = []
M = 7
for i in 1:size(matrix)[1]
    counter = 0
    for j in 1:size(matrix)[2]
            if matrix[i, j] == M
                counter = counter + 1
        end
    end
    push!(ans_vec2, counter)
end
println("\nThe amount of numbers, that equal M: ")
println(ans_vec2)

K = 75
ans_vec3 = [(i, j) for i in 1:size(matrix)[2]-1 for j in i+1:size(matrix)[2] if sum(matrix[:, i] + matrix[:, j]) > K]
println("\nThe amount of column pairs, sum of which is greater than K: ")
println(ans_vec3)
```

```
Matrix:
6×10 Matrix{Int64}:
 9   2  5  2  10   6  6  5   8  6
 6   5  3  4   7   9  9  5   3  4
 6  10  4  4  10   6  5  4   1  8
 4   5  3  7   6  10  9  9  10  3
 7   3  2  3   2  10  5  1   4  9
 5   5  6  2   2   2  2  2   1  6
The amount of numbers, greater than N:
Any[6, 4, 5, 6, 3, 2]

The amount of numbers, that equal M:
Any[0, 1, 0, 1, 1, 0]

The amount of column pairs, sum of which is greater than K:
[(1, 6), (5, 6), (6, 7), (6, 10)]
```

Рис. 2.10: Код выполнения пункта 10

Выполнение пункта 11(2.11):

```
A = [[i^4/(3+j) for j in 1:5] for i in 1:20]
B = [[i^4/(3+i*j) for j in 1:5] for i in 1:20]
display(sum(sum(A)))
display(sum(sum(B)))
```

```
639215.2833333333
89912.02146097136
```

Рис. 2.11: Код выполнения пункта 11

In [27]:
```
println("For cycle:")
for i in 1:100
    print(i, "->", i^2, " ")
end
println()
println("\nWhile cycle:")
i = 1
while i <= 100
    print(i, "->", i^2, " ")
    i += 1
end
println()
squares = Dict(i => i^2 for i = 1:20)
squares = Dict{Int, Int}()
i = 1
while i <= 20
    squares[i] = i^2
    i += 1
end
println("\nSquares dict:")
println(squares)

squares_arr = [i^2 for i in 1:100]
squares_arr = []
i = 1
while i <= 100
    push!(squares_arr, i^2)
    i += 1
end
println("\nSquares array:")
println(squares_arr)
```

```
For cycle:
1->1 2->4 3->9 4->16 5->25 6->36 7->49 8->64 9->81 10->100 11->121 12->144 13->169 14->196 15->225 16->256 17->28
9 18->324 19->361 20->400 21->441 22->484 23->529 24->576 25->625 26->676 27->729 28->784 29->841 30->900 31->961
32->1024 33->1089 34->1156 35->1225 36->1296 37->1369 38->1444 39->1521 40->1600 41->1681 42->1764 43->1849 44->1
936 45->2025 46->2116 47->2209 48->2304 49->2401 50->2500 51->2601 52->2704 53->2809 54->2916 55->3025 56->3136 5
7->3249 58->3364 59->3481 60->3600 61->3721 62->3844 63->3969 64->4096 65->4225 66->4356 67->4489 68->4624 69->47
61 70->4900 71->5041 72->5184 73->5329 74->5476 75->5625 76->5776 77->5929 78->6084 79->6241 80->6400 81->6561 82
->6724 83->6889 84->7056 85->7225 86->7396 87->7569 88->7744 89->7921 90->8100 91->8281 92->8464 93->8649 94->883
6 95->9025 96->9216 97->9409 98->9604 99->9801 100->10000

While cycle:
1->1 2->4 3->9 4->16 5->25 6->36 7->49 8->64 9->81 10->100 11->121 12->144 13->169 14->196 15->225 16->256 17->28
9 18->324 19->361 20->400 21->441 22->484 23->529 24->576 25->625 26->676 27->729 28->784 29->841 30->900 31->961
32->1024 33->1089 34->1156 35->1225 36->1296 37->1369 38->1444 39->1521 40->1600 41->1681 42->1764 43->1849 44->1
936 45->2025 46->2116 47->2209 48->2304 49->2401 50->2500 51->2601 52->2704 53->2809 54->2916 55->3025 56->3136 5
7->3249 58->3364 59->3481 60->3600 61->3721 62->3844 63->3969 64->4096 65->4225 66->4356 67->4489 68->4624 69->47
61 70->4900 71->5041 72->5184 73->5329 74->5476 75->5625 76->5776 77->5929 78->6084 79->6241 80->6400 81->6561 82
->6724 83->6889 84->7056 85->7225 86->7396 87->7569 88->7744 89->7921 90->8100 91->8281 92->8464 93->8649 94->883
6 95->9025 96->9216 97->9409 98->9604 99->9801 100->10000

Squares dict:
Dict(5 => 25, 16 => 256, 20 => 400, 12 => 144, 8 => 64, 17 => 289, 1 => 1, 19 => 361, 6 => 36, 11 => 121, 9 => 8
1, 14 => 196, 3 => 9, 7 => 49, 4 => 16, 13 => 169, 15 => 225, 2 => 4, 10 => 100, 18 => 324)

Squares array:
Any[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441, 484, 529, 576, 6
25, 676, 729, 784, 841, 900, 961, 1024, 1089, 1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2
025, 2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 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, 9801, 10000]
```

In [2]:
```
a = parse(Int, readline())
if isodd(a)
    println("Нечётное")
else
    println(a)
end
```

```
Нечётное
```

In [3]:
```
function add_one(numb)
    println(numb + 1)
end

for i in 1:3
    a = parse(Int, readline())
```

```
    add_one(a)
end
```

```
3
2
5
```

In [4]:
```
println("Matrix1:")
matrix1 = map(x -> x, reshape(Array(1:20), 4, 5))
display(matrix1)

println("\nMatrix2:")
matrix2 = broadcast(x -> x, reshape(Array(1:20), 4, 5))
display(matrix2)
```

```
Matrix1:
4×5 Matrix{Int64}:
 1  5   9  13  17
 2  6  10  14  18
 3  7  11  15  19
 4  8  12  16  20
Matrix2:
4×5 Matrix{Int64}:
 1  5   9  13  17
 2  6  10  14  18
 3  7  11  15  19
 4  8  12  16  20
```

In [5]:
```
A = [1 1 3; 5 2 6; -2 -1 -3]
println("Matrix:")
display(A)

println("\nMatrix^3:")
display(A.^3)

for i in 1:3
    A[i, 3] = A[i, 1] + A[i, 2]
end

println("\nMatrix with sum:")
display(A)
```

```
Matrix:
3×3 Matrix{Int64}:
  1   1   3
  5   2   6
 -2  -1  -3
Matrix^3:
3×3 Matrix{Int64}:
   1   1   27
 125   8  216
  -8  -1  -27
Matrix with sum:
3×3 Matrix{Int64}:
  1   1   2
  5   2   7
 -2  -1  -3
```

In [6]:
```
B = zeros(15, 3)

for i in 1:15
    B[i,1] = 10
    B[i,2] = -10
    B[i,3] = 10
end

Bt = transpose(B)
C = Bt * B

println("\nMuliplication matrix:")
display(C)
```

```
Muliplication matrix:
3×3 Matrix{Float64}:
  1500.0  -1500.0   1500.0
 -1500.0   1500.0  -1500.0
  1500.0  -1500.0   1500.0
```

In [8]:
```
matrix1 = [[if i-j in [1, -1] 1 else 0 end for j in 1:6] for i in 1:6]
hcat(matrix1...)
println("Z1:")
display(matrix1)
```

```
matrix2 = [[if i-j in [2, 0, -2] 1 else 0 end for j in 1:6] for i in 1:6]
hcat(matrix2...)
println("\nZ2:")
display(matrix2)

matrix3 = [[if i+j in [5, 7, 9] 1 else 0 end for j in 1:6] for i in 1:6]
hcat(matrix3...)
println("\nZ3:")
display(matrix3)

matrix4 = [[if i-j in [4, 2, 0, -2, -4] 1 else 0 end for j in 1:6] for i in 1:6]
hcat(matrix4...)
println("\nZ4:")
display(matrix4)
```

```
Z1:
6-element Vector{Vector{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:
6-element Vector{Vector{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:
6-element Vector{Vector{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]
Z4:
6-element Vector{Vector{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]
```

In [9]:
```
outer(x, y, operation) = transpose(hcat([[sum(operation(x[i, k], y[k, j]) for k in 1:size(x)[2]) for j in 1:size

B = reshape(rand(1:5, 12), 2, 6)
A = reshape(rand(1:5, 12), 6, 2)
println("Source matrices:")
display(A)
display(B)
println("\nSum matrix:")
display(outer(A, B, +))

println("\nSub matrix:")
display(outer(A, B, -))

println("\nMul matrix:")
display(outer(A, B, *))

println("\nDiv matrix:")
display(outer(A, B, /))

println("\nPow matrix:")
display(outer(A, B, ^))

println("\nA1:")
display(outer(reshape(0:4, 5, 1), reshape(0:4, 1, 5), +))

println("\nA2:")
println(outer(reshape(0:4, 5, 1), reshape(1:5, 1, 5), ^))

println("\nA3:")
display(outer(hcat([[if i==j 1 else 0 end for j in 0:4] for i in 0:4]...), hcat([Vector(i:i+4).%5 for i in 0:4].
```

```
println("\nA4:")
display(outer(hcat([[if i==j 1 else 0 end for j in 0:9] for i in 0:9]...), hcat([Vector(i:i+9).%10 for i in 0:9]

println("\nA5:")
display(outer(hcat([[if i==j 1 else 0 end for j in 0:8] for i in 0:8]...), hcat([Vector(i+9:-1:i+1).%9 for i in
```

```
Source matrices:
6×2 Matrix{Int64}:
 1  2
 1  2
 1  3
 1  1
 3  1
 2  5
2×6 Matrix{Int64}:
 2  3  4  2  5  2
 3  4  5  2  3  4
Sum matrix:
6×6 transpose(::Matrix{Int64}) with eltype Int64:
  8  10  12   7  11   9
  8  10  12   7  11   9
  9  11  13   8  12  10
  7   9  11   6  10   8
  9  11  13   8  12  10
 12  14  16  11  15  13
Sub matrix:
6×6 transpose(::Matrix{Int64}) with eltype Int64:
 -2  -4  -6  -1  -5  -3
 -2  -4  -6  -1  -5  -3
 -1  -3  -5   0  -4  -2
 -3  -5  -7  -2  -6  -4
 -1  -3  -5   0  -4  -2
  2   0  -2   3  -1   1
Mul matrix:
6×6 transpose(::Matrix{Int64}) with eltype Int64:
  8  11  14   6  11  10
  8  11  14   6  11  10
 11  15  19   8  14  14
  5   7   9   4   8   6
  9  13  17   8  18  10
 19  26  33  14  25  24
Div matrix:
6×6 transpose(::Matrix{Float64}) with eltype Float64:
 1.16667   0.833333  0.65  1.5  0.866667  1.0
 1.16667   0.833333  0.65  1.5  0.866667  1.0
 1.5       1.08333   0.85  2.0  1.2       1.25
 0.833333  0.583333  0.45  1.0  0.533333  0.75
 1.83333   1.25      0.95  2.0  0.933333  1.75
 2.66667   1.91667   1.5   3.5  2.06667   2.25
Pow matrix:
6×6 transpose(::Matrix{Int64}) with eltype Int64:
   9   17    33   5    9   17
   9   17    33   5    9   17
  28   82   244  10   28   82
   2    2     2   2    2    2
  10   28    82  10  244   10
 129  633  3141  29  157  629
A1:
5×5 transpose(::Matrix{Int64}) with eltype 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:
[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:
5×5 transpose(::Matrix{Int64}) with eltype 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
A4:
```

```
10×10 transpose(::Matrix{Int64}) with eltype 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:
9×9 transpose(::Matrix{Int64}) with eltype Int64:
 0  1  2  3  4  5  6  7  8
 8  0  1  2  3  4  5  6  7
 7  8  0  1  2  3  4  5  6
 6  7  8  0  1  2  3  4  5
 5  6  7  8  0  1  2  3  4
 4  5  6  7  8  0  1  2  3
 3  4  5  6  7  8  0  1  2
 2  3  4  5  6  7  8  0  1
 1  2  3  4  5  6  7  8  0
```

In [10]:

```
M = [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]
N = [7, -1, -3, 5, 17]
println("Solution Vector: ")
display(M \ N)
```

```
Solution Vector:
5-element Vector{Float64}:
 -2.0000000000000036
  3.0000000000000058
  4.999999999999998
  1.9999999999999991
 -3.999999999999999
```

In [18]:

```
matrix = rand(1:10, 6, 10)
println("Matrix: ")
display(matrix)
num = 5

ans_vec1 = []
for i in 1:size(matrix)[1]
    counter = 0
    for j in 1:size(matrix)[2]
            if matrix[i, j] > num
                counter = counter +1
            end
    end
    push!(ans_vec1, counter)
end
println("The amount of numbers, greater than N: ")
println(ans_vec1)


ans_vec2 = []
M = 7
for i in 1:size(matrix)[1]
    counter = 0
    for j in 1:size(matrix)[2]
            if matrix[i, j] == M
                counter = counter + 1
        end
    end
    push!(ans_vec2, counter)
end
println("\nThe amount of numbers, that equal M: ")
println(ans_vec2)

K = 75
ans_vec3 = [(i, j) for i in 1:size(matrix)[2]-1 for j in i+1:size(matrix)[2] if sum(matrix[:, i] + matrix[:, j])
println("\nThe amount of column pairs, sum of which is greater than K: ")
println(ans_vec3)
```

```
Matrix:
```

```
6×10 Matrix{Int64}:
 9   2  5  2  10   6  6  5   8  6
 6   5  3  4   7   9  9  5   3  4
 6  10  4  4  10   6  5  4   1  8
 4   5  3  7   6  10  9  9  10  3
 7   3  2  3   2  10  5  1   4  9
 5   5  6  2   2   2  2  2   1  6
The amount of numbers, greater than N:
Any[6, 4, 5, 6, 3, 2]

The amount of numbers, that equal M:
Any[0, 1, 0, 1, 1, 0]

The amount of column pairs, sum of which is greater than K:
[(1, 6), (5, 6), (6, 7), (6, 10)]
```

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

```
639215.2833333333
89912.02146097136
```

In [ ]:
```

```

3 Выводы

Я освоил применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.