РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

дисциплина: Компьютерный практикум по статистическому анализу данных

Студент: Евдокимов Максим Михайлович (1032203019)

Группа: НФИбд-01-20

МОСКВА

2023 г.

Постановка задачи

1. Используя Jupyter Lab, повторите примеры из раздела 3.2.
2. Выполните задания для самостоятельной работы (раздел 3.4).

Выполнение работы

1. Используя циклы while и for:

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

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

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

```
# 1.1
for i in 1:100
    print("$i=>", i^2, " ")
end
j = 1; println("\n")
while j <= 100
    print("$j=>", j^2, " "); j += 1
end
```

```
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=>289 18=>324 19=>361 20=>400 21=>44
1 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=>1936 45=>2025 46=>2116 47=>2209 48=>2304 49=>2401 50=>2500 51=>2601 52=>2704 53=>2809 54=>2916 5
5=>3025 56=>3136 57=>3249 58=>3364 59=>3481 60=>3600 61=>3721 62=>3844 63=>3969 64=>4096 65=>4225 66=>4356 67=>4489 68=>4624 69=>4761 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=>8836 95=>9025 96=>9216 97=>9409 98=>9604 99=>9801 100=>10000

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=>289 18=>324 19=>361 20=>400 21=>44
1 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=>1936 45=>2025 46=>2116 47=>2209 48=>2304 49=>2401 50=>2500 51=>2601 52=>2704 53=>2809 54=>2916 5
5=>3025 56=>3136 57=>3249 58=>3364 59=>3481 60=>3600 61=>3721 62=>3844 63=>3969 64=>4096 65=>4225 66=>4356 67=>4489 68=>4624 69=>4761 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=>8836 95=>9025 96=>9216 97=>9409 98=>9604 99=>9801 100=>10000
```

```
# 1.2
squares = Dict(i => i^2 for i = 1:20); show(squares); println()
squares = empty!(squares); i = 1
while i <= 20
    squares[i] = i^2; global i += 1
end; show(squares); println()
```

```
Dict(5 => 25, 16 => 256, 7 => 49, 20 => 400, 12 => 144, 8 => 64, 17 => 289, 1 => 1, 19 => 361, 4 => 16, 6 => 36, 13 => 169, 2 => 4, 10 => 100, 11
=> 121, 9 => 81, 15 => 225, 18 => 324, 14 => 196, 3 => 9)
Dict(5 => 25, 16 => 256, 7 => 49, 20 => 400, 12 => 144, 8 => 64, 17 => 289, 1 => 1, 19 => 361, 4 => 16, 6 => 36, 13 => 169, 2 => 4, 10 => 100, 11
=> 121, 9 => 81, 15 => 225, 18 => 324, 14 => 196, 3 => 9)
```

```
# 1.3
squares_arr = [i^2 for i in 1:100]; show(squares_arr); println()
squares_arr = []; i = 1
while i <= 100
    append!(squares_arr, i^2); global i += 1
end; show(squares_arr); println()
```

```
[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, 1
024, 1089, 1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025, 2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 31
36, 3249, 3364, 3481, 3600, 3721, 3844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5041, 5184, 5329, 5476, 5625, 5776, 5929, 6084, 6241, 640
0, 6561, 6724, 6889, 7056, 7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409, 9604, 9801, 10000]
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, 625, 676, 729, 784, 841, 900, 96
1, 1024, 1089, 1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025, 2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 302
5, 3136, 3249, 3364, 3481, 3600, 3721, 3844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5041, 5184, 5329, 5476, 5625, 5776, 5929, 6084, 624
1, 6400, 6561, 6724, 6889, 7056, 7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409, 9604, 9801, 10000]
```

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

```
# 2
numb = parse(Int, readline())
if isodd(numb) println("Нечётное") else println(numb) end
iseven(numb) ? println(numb) : println("Нечётное")
```

```
stdin>  6
6
6
```

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

```
# 3
function add_one(numb)
    return numb + 1
end

n = parse(Int, readline())
while true
    global n = add_one(n); print(n, " Если хотите завершить введите y: ")
    if readline() == "y"
        break
    println()
    end
end
```

```
stdin>  5
6 Если хотите завершить введите y:
stdin>  n
7 Если хотите завершить введите y:
stdin>  n
8 Если хотите завершить введите y:
stdin>  n
9 Если хотите завершить введите y:
stdin>  n
10 Если хотите завершить введите y:
stdin>  y
```

4. Используйте map() или broadcast() для задания матрицы *A*, каждый элемент которой увеличивается на единицу по сравнению с предыдущим.

```
# 4
array1 = map(x -> x, reshape(Array(1:20), 4, 5)); display(array1); println()
array2 = broadcast(x -> x, reshape(Array(1:20), 4, 5)); display(array2); println()
# не совсем понимаю зачем тут map и broadcast без них это решается проще
```

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

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

5. Задайте матрицу A следующего вида:

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

– Найдите $A3$.

– Замените третий столбец матрицы A на сумму второго и третьего столбцов.

```
# 5.1
A = [1 1 3; 5 2 6; -2 -1 -3]
display(A)
display(A^3)
```

```
3×3 Matrix{Int64}:
  1   1   3
  5   2   6
 -2  -1  -3
3×3 Matrix{Int64}:
 0  0  0
 0  0  0
 0  0  0
```

```
# 5.2
for i in 1:3 A[i, 3] = A[i, 1] + A[i, 2] end
display(A)
```

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

6. Создайте матрицу B с элементами $B_{i1} = 10$, $B_{i2} = -10$, $B_{i3} = 10$, i = 1, 2, …, 15. Вычислите матрицу $C = B^T * B$

```
# 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
```

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

7. Создайте матрицу Z размерности 6 × 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 × 6:

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

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

```
# 7
Z1 = hcat([[if i-j in [1, -1] 1 else 0 end for j in 1:6] for i in 1:6]...); display(Z1)
Z2 = hcat([[if i-j in [2, 0, -2] 1 else 0 end for j in 1:6] for i in 1:6]...); display(Z2)
Z3 = hcat([[if i+j in [5, 7, 9] 1 else 0 end for j in 1:6] for i in 1:6]...); display(Z3)
Z4 = hcat([[if i-j in [4, 2, 0, -2, -4] 1 else 0 end for j in 1:6] for i in 1:6]...); display(Z4)
```

```
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
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
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
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
```

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 $C_{ij} = M \sum k{=}1\ A_{ik}B_kj$ (или в тензорном виде $C_{ji} = M \sum k{=}1\ A_{ik}B_{kj}$).

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

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

$$A_3 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 0 \\ 2 & 3 & 4 & 0 & 1 \\ 3 & 4 & 0 & 1 & 2 \\ 4 & 0 & 1 & 2 & 3 \end{pmatrix}, \quad A_4 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix} \quad A_5 = \begin{pmatrix} 0 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 1 & 0 & 8 & 7 & 6 & 5 & 4 & 3 & 2 \\ 2 & 1 & 0 & 8 & 7 & 6 & 5 & 4 & 3 \\ 3 & 2 & 1 & 0 & 8 & 7 & 6 & 5 & 4 \\ 4 & 3 & 2 & 1 & 0 & 8 & 7 & 6 & 5 \\ 5 & 4 & 3 & 2 & 1 & 0 & 8 & 7 & 6 \\ 6 & 5 & 4 & 3 & 2 & 1 & 0 & 8 & 7 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & 8 \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{pmatrix}$$

```
# 8.1
function outer(x, y, operation)
    res = [[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]]
    return transpose(hcat(res...))
end

M1, M2 = reshape(rand(1:5, 15), 5, 3), reshape(rand(1:5, 15), 3, 5); display(M1); display(M2)
display(outer(M1, M2, +))
display(outer(M1, M2, -))
display(outer(M1, M2, *))
display(outer(M1, M2, /))
display(outer(M1, M2, ^))
```

```
5×3 Matrix{Int64}:
 5  4  4
 5  2  3
 3  4  5
 5  5  4
 3  3  1
3×5 Matrix{Int64}:
 4  1  3  3  5
 4  1  1  1  3
 2  3  4  2  5
5×5 transpose(::Matrix{Int64}) with eltype Int64:
 23  18  21  19  26
 20  15  18  16  23
 22  17  20  18  25
 24  19  22  20  27
 17  12  15  13  20
5×5 transpose(::Matrix{Int64}) with eltype Int64:
  3  8   5  7   0
  0  5   2  4  -3
  2  7   4  6  -1
  4  9   6  8   1
 -3  2  -1  1  -6
5×5 transpose(::Matrix{Int64}) with eltype Int64:
 44  21  35  27  57
 34  16  29  23  46
 38  22  33  23  52
 48  22  36  28  60
 26   9  16  14  29
5×5 transpose(::Matrix{Float64}) with eltype Float64:
 4.25  10.3333   6.66667  7.66667  3.13333
 3.25   8.0      4.41667  5.16667  2.26667
 4.25   8.66667  6.25     7.5      2.93333
 4.5   11.3333   7.66667  8.66667  3.46667
 2.0    6.33333  4.25     4.5      1.8
5×5 transpose(::Matrix{Int64}) with eltype Int64:
  897   73  385  145  4213
  650   34  208  136  3376
  362  132  656   56  3432
 1266   74  386  146  4274
  163    7   31   31   271
```

```julia
# 8.2
A1 = outer(reshape(0:4, 5, 1), reshape(0:4, 1, 5), +); display(A1)
A2 = outer(reshape(0:4, 5, 1), reshape(1:5, 1, 5), ^); display(A2)
A3 = 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]...), *); display(A3)
A4 = 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]...), *); display(A4)
A5 = 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]...), *); display(A5)
```

```
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
5×5 transpose(::Matrix{Int64}) with eltype 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
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
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
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
```

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

```julia
# 9
using LinearSolve
M, N = Matrix{Float64}([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]), Vector{Float64}([7, -1, -3, 5, 17])
display(M); display(N)
prob = LinearProblem(M, N)
sol = solve(prob)
sol.u
```

```
5×5 Matrix{Float64}:
 1.0  2.0  3.0  4.0  5.0
 2.0  1.0  2.0  3.0  4.0
 3.0  2.0  1.0  2.0  3.0
 4.0  3.0  2.0  1.0  2.0
 5.0  4.0  3.0  2.0  1.0
5-element Vector{Float64}:
  7.0
 -1.0
 -3.0
  5.0
 17.0
5-element Vector{Float64}:
 -1.9999999999999971
  2.9999999999999982
  4.999999999999999
  1.9999999999999982
 -3.9999999999999987
```

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

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

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

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

```julia
# 10.1
M = rand(1:10, 6, 10); display(M)
N = 5
for i in 1:6
    println(length(filter(z -> z > N, M[i, 1:10])))
end
```

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

```
# 10.2
m, n = 5, 2
for i in 1:6
    if length(filter(z -> z > m, M[i, 1:10])) == n
        println("in line $(i) there are $(n) numbers greater than $(m)")
    end
end
```

```
in line 6 there are 2 numbers greater than 5
```

```
# 10.3
K = 75
for i in 1:6
    for j in i+1:6
        if sum(filter(z -> z > m, M[i, 1:10])) + sum(filter(z -> z > m, M[j, 1:10])) > K
            println("the sum of lines $(i) and $(j) is greater than $(K)")
        end
    end
end
```

```
the sum of lines 1 and 4 is greater than 75
the sum of lines 2 and 4 is greater than 75
the sum of lines 3 and 4 is greater than 75
the sum of lines 4 and 5 is greater than 75
```

11. Вычислите:

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

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

```
# 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
```

Выводы

В ходе выполнения работы были изучены принципы работы условных выражений if-elseif-else, циклов while-for и функций function, а также принципы работы их с векторами и матрицами.