РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Компьютерный практикум по статистическому анализу данных

Студент: Евдокимов Максим Михайлович (1032203019)

Группа: НФИбд-01-20

МОСКВА

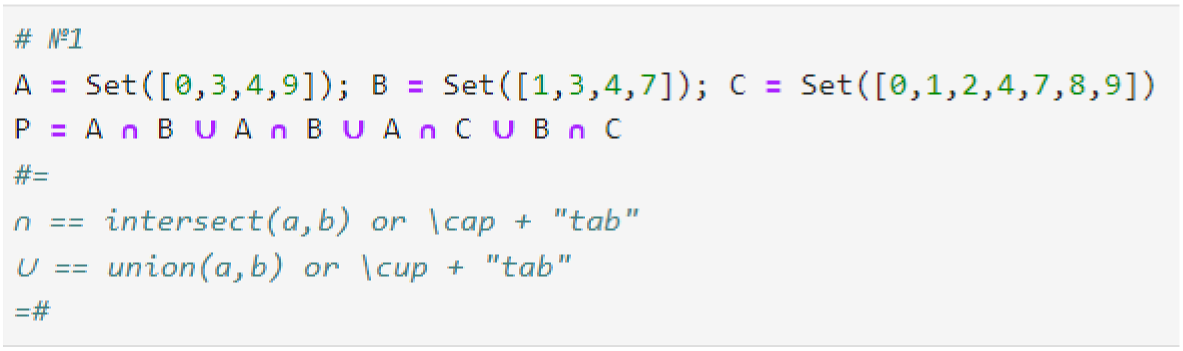
2023 г.

Постановка задачи

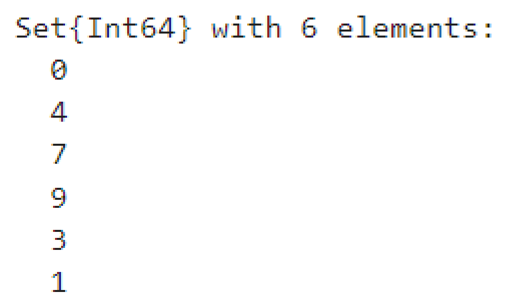
1. Используя Jupyter Lab, повторите примеры из раздела 2.2.
2. Выполните задания для самостоятельной работы (раздел 2.4).

Выполнение работы

1. Даны множества: A = {0, 3, 4, 9}, B = {1, 3, 4, 7}, C = {0, 1, 2, 4, 7, 8, 9}. Найти P = A ∩ B ∪ A ∩ B ∪ A ∩ C ∪ B ∩ C.

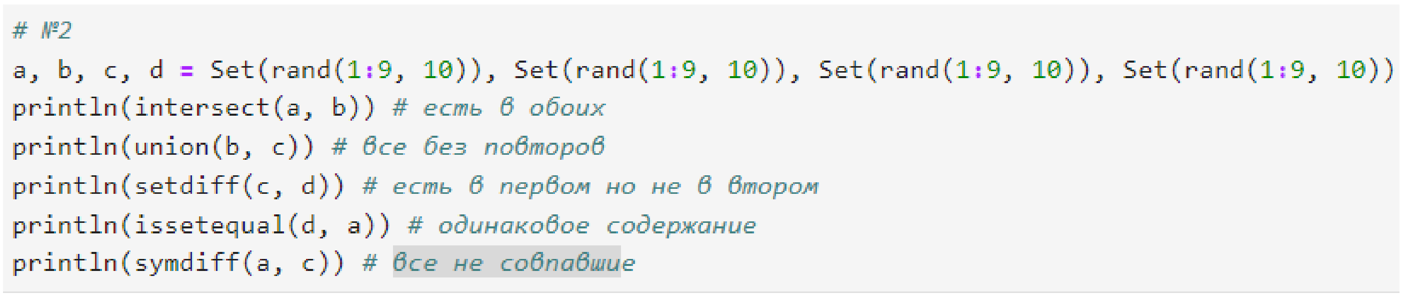


```
# №1
A = Set([0,3,4,9]); B = Set([1,3,4,7]); C = Set([0,1,2,4,7,8,9])
P = A ∩ B ∪ A ∩ B ∪ A ∩ C ∪ B ∩ C
#=
∩ == intersect(a,b) or \cap + "tab"
∪ == union(a,b) or \cup + "tab"
=#
```

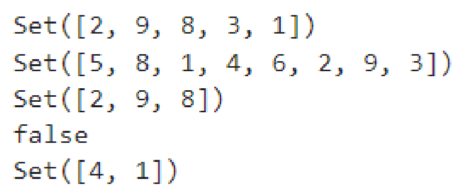


```
Set{Int64} with 6 elements:
  0
  4
  7
  9
  3
  1
```

2. Приведите свои примеры с выполнением операций над множествами элементов разных типов.

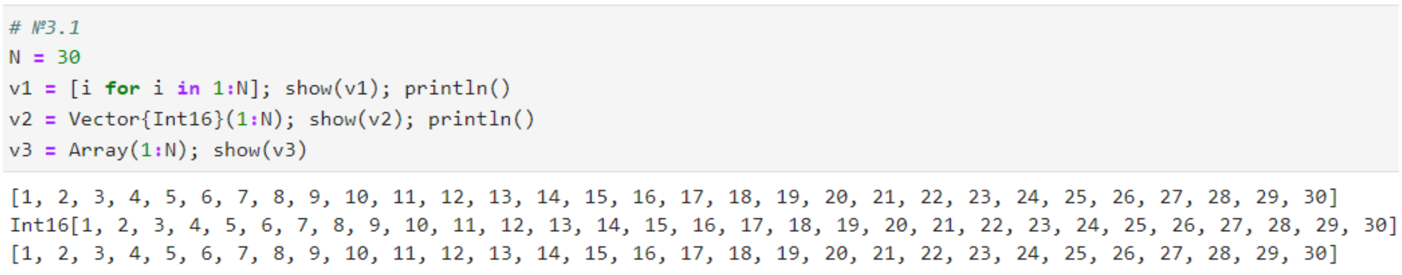


```
# №2
a, b, c, d = Set(rand(1:9, 10)), Set(rand(1:9, 10)), Set(rand(1:9, 10)), Set(rand(1:9, 10))
println(intersect(a, b)) # есть в обоих
println(union(b, c)) # все без повторов
println(setdiff(c, d)) # есть в первом но не в втором
println(issetequal(d, a)) # одинаковое содержание
println(symdiff(a, c)) # все не совпавшие
```



```
Set([2, 9, 8, 3, 1])
Set([5, 8, 1, 4, 6, 2, 9, 3])
Set([2, 9, 8])
false
Set([4, 1])
```

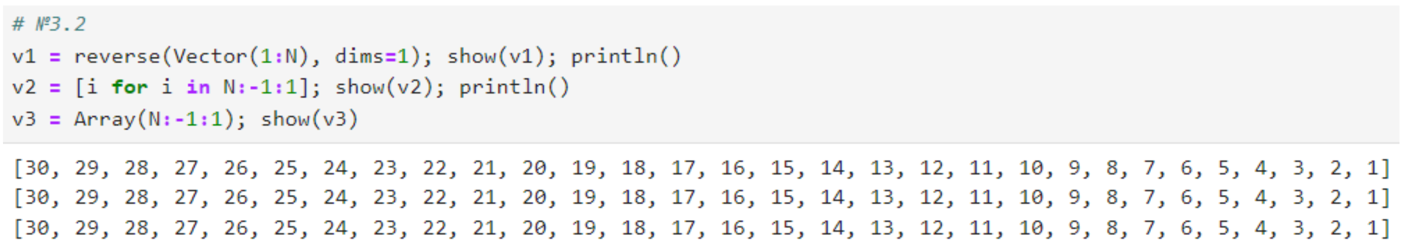
3. Создайте разными способами:
   1) массив (1, 2, 3, … N − 1, N), N выберите больше 20;



```
# №3.1
N = 30
v1 = [i for i in 1:N]; show(v1); println()
v2 = Vector{Int16}(1:N); show(v2); println()
v3 = Array(1:N); show(v3)
```

```
[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]
Int16[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]
[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]
```

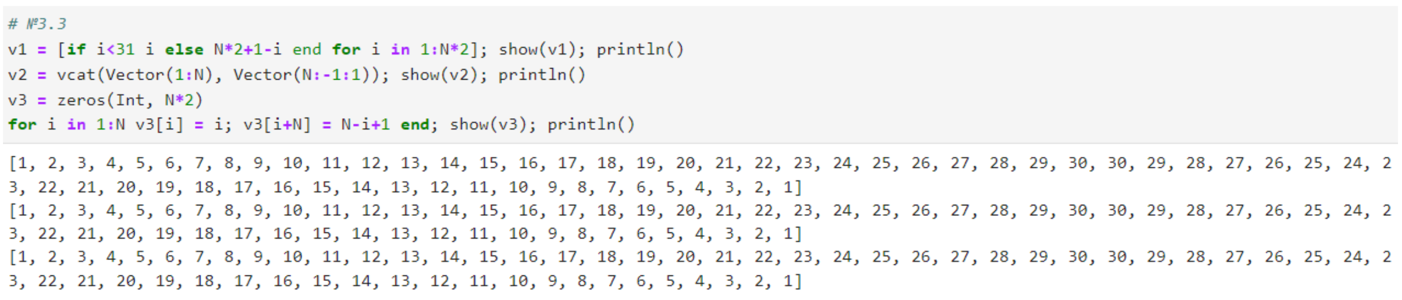
2) массив (N, N − 1 …, 2, 1), N выберите больше 20;



```
# №3.2
v1 = reverse(Vector(1:N), dims=1); show(v1); println()
v2 = [i for i in N:-1:1]; show(v2); println()
v3 = Array(N:-1:1); show(v3)
```

```
[30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
[30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
[30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

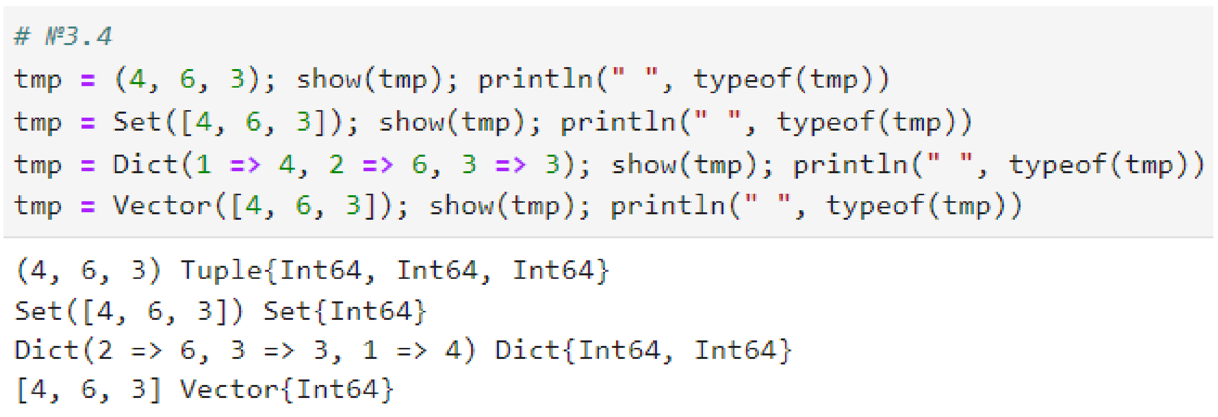
3) массив (1, 2, 3, …, N − 1, N, N − 1, …, 2, 1), *N* выберите больше 20;



```
# №3.3
v1 = [if i<31 i else N*2+1-i end for i in 1:N*2]; show(v1); println()
v2 = vcat(Vector(1:N), Vector(N:-1:1)); show(v2); println()
v3 = zeros(Int, N*2)
for i in 1:N v3[i] = i; v3[i+N] = N-i+1 end; show(v3); println()
```

```
[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, 30, 29, 28, 27, 26, 25, 24, 2
3, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
[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, 30, 29, 28, 27, 26, 25, 24, 2
3, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
[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, 30, 29, 28, 27, 26, 25, 24, 2
3, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

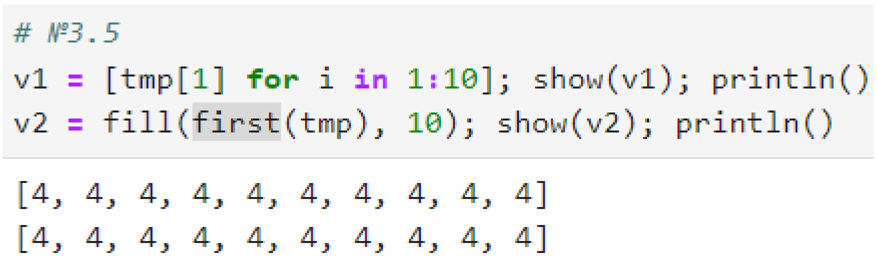
4) массив с именем tmp вида (4, 6, 3);



```
# №3.4
tmp = (4, 6, 3); show(tmp); println(" ", typeof(tmp))
tmp = Set([4, 6, 3]); show(tmp); println(" ", typeof(tmp))
tmp = Dict(1 => 4, 2 => 6, 3 => 3); show(tmp); println(" ", typeof(tmp))
tmp = Vector([4, 6, 3]); show(tmp); println(" ", typeof(tmp))
```

```
(4, 6, 3) Tuple{Int64, Int64, Int64}
Set([4, 6, 3]) Set{Int64}
Dict(2 => 6, 3 => 3, 1 => 4) Dict{Int64, Int64}
[4, 6, 3] Vector{Int64}
```

5) массив, в котором первый элемент массива tmp повторяется 10 раз;



```
# №3.5
v1 = [tmp[1] for i in 1:10]; show(v1); println()
v2 = fill(first(tmp), 10); show(v2); println()
```

```
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
```

6) массив, в котором все элементы массива tmp повторяются 10 раз;

```
# №3.6
v1 = stack([fill(i, 10) for i in tmp], dims=1); show(v1); println()
v2 = [fill(tmp[1], 10); fill(tmp[2], 10); fill(tmp[3], 10)]; show(v2); println()
v3 = vcat([fill(i, 10) for i in tmp]...); show(v1); println()
```

```
[4 4 4 4 4 4 4 4 4 4; 6 6 6 6 6 6 6 6 6 6; 3 3 3 3 3 3 3 3 3 3]
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
[4 4 4 4 4 4 4 4 4 4; 6 6 6 6 6 6 6 6 6 6; 3 3 3 3 3 3 3 3 3 3]
```

7) массив, в котором первый элемент массива tmp встречается 11 раз, второй элемент — 10 раз, третий элемент — 10 раз;

```
# №3.7
v1 = [if i < 12 tmp[1] elseif i < 22 tmp[2] else tmp[3] end for i in 1:31]; show(v1); println()
v2 = [fill(tmp[1], 11); fill(tmp[2], 10); fill(tmp[3], 10)]; show(v2); println()
```

```
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

8) массив, в котором первый элемент массива tmp встречается 10 раз подряд, второй элемент — 20 раз подряд, третий элемент — 30 раз подряд;

```
# №3.8
v1 = [if i < 11 tmp[1] elseif i < 33 tmp[2] else tmp[3] end for i in 1:60]; show(v1); println()
v2 = [fill(tmp[1], 10); fill(tmp[2], 20); fill(tmp[3], 30)]; show(v2); println()
v3 = vcat([fill(tmp[i], i*10) for i in 1:3]...); show(v3); println()
```

```
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

9) массив из элементов вида $2^{tmp[i]}$, i = 1, 2, 3, где элемент $2^{tmp[3]}$ встречается 4 раза; посчитайте в полученном векторе, сколько раз встречается цифра 6, и выведите это значение на экран;

```
# №3.9
v1 = [2^tmp[i] for i in [1,2,3,3,3,3]]; print("вариант 1: "); show(v1); println()
v2 = 2 .^ vcat(tmp, fill(tmp[3], 3)); print("вариант 2: "); show(v2); println()
v3 = fill(2, 6) .^ vcat(tmp, fill(tmp[3], 3)); print("вариант 3: "); show(v3); println()
rez1 = count(x -> occursin("6", string(x)), rand([v1, v2, v3])); print(rez1, " ")
rez2 = count(x -> contains(string(x), "6"), rand([v1, v2, v3])); print(rez1, " ")
rez3 = count(x -> 6 in digits(x), rand([v1, v2, v3])); print(rez1, " ")
rez4 = mapreduce(x -> occursin("6", string(x)), +, rand([v1, v2, v3])); print(rez1, " ")
rez5 = mapreduce(x -> contains(string(x), "6"), +, rand([v1, v2, v3])); print(rez1, " ")
rez6 = mapreduce(x -> 6 in digits(x), +, rand([v1, v2, v3])); print(rez1, " ")
rez7 = sum([if 6 in digits(i) 1 else 0 end for i in rand([v1, v2, v3])]); print(rez1, " ")
```

```
вариант 1: [16, 64, 8, 8, 8, 8]
вариант 2: [16, 64, 8, 8, 8, 8]
вариант 3: [16, 64, 8, 8, 8, 8]
2 2 2 2 2 2 2
```

10) вектор значений $y = e^x \cos(x)$ в точках x = 3, 3.1, 3.2, …, 6, найдите среднее значение y;

```
# №3.10
y1 = [exp(x)*cos(x) for x in 3:0.1:6]
y2 = [exp(x) for x in 3:0.1:6] .* [cos(x) for x in 3:0.1:6]
println(Statistics.mean(y1))
println(sum(y2)/size(y2)[1])
println(sum([exp(x)*cos(x)/31 for x in 3:0.1:6]))
```

```
53.11374594642971
53.11374594642971
53.11374594642972
```

11) вектор вида (x^i, y^j), x = 0.1, i = 3, 6, 9, …, 36, y = 0.2, j = 1, 4, 7, …, 34;

```
# №3.11
v1 = [[0.1^i for i in 3:3:36], [0.2^i for i in 1:3:34]]; display(v1); println()
v2 = [(0.1^(i+2), 0.2^i) for i in 1:3:34]; display(v2); println()
```

```
2-element Vector{Vector{Float64}}:
 [0.0010000000000000002, 1.0000000000000004e-6, 1.0000000000000005e-9, 1.0000000000000008e-12, 1.0000000000000009e-15, 1.0000000000000008e-18,
1.0000000000000012e-21, 1.0000000000000012e-24, 1.0000000000000015e-27, 1.0000000000000017e-30, 1.0000000000000018e-33, 1.000000000000002e-36]
 [0.2, 0.0016000000000000003, 1.2800000000000006e-5, 1.0240000000000006e-7, 8.192000000000005e-10, 6.553600000000055e-12, 5.242880000000056e
-14, 4.194304000000005e-16, 3.3554432000000044e-18, 2.684354560000004e-20, 2.147483648000004e-22, 1.7179869184000035e-24]

12-element Vector{Tuple{Float64, Float64}}:
 (0.0010000000000000002, 0.2)
 (1.0000000000000004e-6, 0.0016000000000000003)
 (1.0000000000000005e-9, 1.2800000000000006e-5)
 (1.0000000000000008e-12, 1.0240000000000006e-7)
 (1.0000000000000009e-15, 8.192000000000005e-10)
 (1.0000000000000008e-18, 6.553600000000055e-12)
 (1.0000000000000012e-21, 5.242880000000056e-14)
 (1.0000000000000012e-24, 4.194304000000005e-16)
 (1.0000000000000015e-27, 3.3554432000000044e-18)
 (1.0000000000000017e-30, 2.684354560000004e-20)
 (1.0000000000000018e-33, 2.147483648000004e-22)
 (1.000000000000002e-36, 1.7179869184000035e-24)
```

12) вектор с элементами $2^i/i$, i = 1, 2, …, M, M = 25;

```
# №3.12
v1 = [2^i/i for i in 1:25]; show(v1); println()
v2 = 2 .^ Vector(1:25) ./ Vector(1:25); show(v2); println()
v3 = map(z -> 2^z/z, Vector(1:25)); show(v3); println()
```

```
[2.0, 2.0, 2.6666666666666665, 4.0, 6.4, 10.666666666666666, 18.285714285714285, 32.0, 56.888888888888886, 102.4, 186.1818181818182, 341.33333
33333333, 630.1538461538462, 1170.2857142857142, 2184.5333333333333, 4096.0, 7710.117647058823, 14563.555555555555, 27594.105263157893, 52428.
8, 99864.38095238095, 190650.18181818182, 364722.0869565217, 699050.6666666666, 1.34217728e6]
[2.0, 2.0, 2.6666666666666665, 4.0, 6.4, 10.666666666666666, 18.285714285714285, 32.0, 56.888888888888886, 102.4, 186.1818181818182, 341.33333
33333333, 630.1538461538462, 1170.2857142857142, 2184.5333333333333, 4096.0, 7710.117647058823, 14563.555555555555, 27594.105263157893, 52428.
8, 99864.38095238095, 190650.18181818182, 364722.0869565217, 699050.6666666666, 1.34217728e6]
[2.0, 2.0, 2.6666666666666665, 4.0, 6.4, 10.666666666666666, 18.285714285714285, 32.0, 56.888888888888886, 102.4, 186.1818181818182, 341.33333
33333333, 630.1538461538462, 1170.2857142857142, 2184.5333333333333, 4096.0, 7710.117647058823, 14563.555555555555, 27594.105263157893, 52428.
8, 99864.38095238095, 190650.18181818182, 364722.0869565217, 699050.6666666666, 1.34217728e6]
```

13) вектор вида (”fn1”, ”fn2”, …, ”fnN”), N = 30;

```
# №3.13
v1 = ["fn"*string(i) for i in 1:N]; show(v1); println()
v2 = ["fn$i" for i in 1:N]; show(v2); println()
v3 = "fn".*[string(i) for i in 1:N]; show(v3); println()
```

```
["fn1", "fn2", "fn3", "fn4", "fn5", "fn6", "fn7", "fn8", "fn9", "fn10", "fn11", "fn12", "fn13", "fn14", "fn15", "fn16", "fn17", "fn18", "fn1
9", "fn20", "fn21", "fn22", "fn23", "fn24", "fn25", "fn26", "fn27", "fn28", "fn29", "fn30"]
["fn1", "fn2", "fn3", "fn4", "fn5", "fn6", "fn7", "fn8", "fn9", "fn10", "fn11", "fn12", "fn13", "fn14", "fn15", "fn16", "fn17", "fn18", "fn1
9", "fn20", "fn21", "fn22", "fn23", "fn24", "fn25", "fn26", "fn27", "fn28", "fn29", "fn30"]
["fn1", "fn2", "fn3", "fn4", "fn5", "fn6", "fn7", "fn8", "fn9", "fn10", "fn11", "fn12", "fn13", "fn14", "fn15", "fn16", "fn17", "fn18", "fn1
9", "fn20", "fn21", "fn22", "fn23", "fn24", "fn25", "fn26", "fn27", "fn28", "fn29", "fn30"]
```

14) векторы x = (x_1, x_2, …, x_n) и y = (y_1, y_2, …, y_n) целочисленного типа длины N = 250 как случайные выборки из совокупности 0, 1, …, 999; на его основе:

```
# №3.14
N = 250
x, y = rand(0:999, N), rand(0:999, N)
show(x); println(); show(y)
```

```
[654, 146, 605, 707, 699, 65, 740, 342, 950, 404, 861, 490, 448, 212, 12, 61, 978, 332, 995, 71, 561, 908, 878, 48, 527, 384, 40, 411, 298, 62
5, 750, 872, 588, 353, 308, 690, 203, 176, 391, 502, 886, 831, 114, 539, 420, 982, 736, 143, 369, 624, 520, 728, 453, 505, 561, 677, 985, 706,
809, 63, 102, 582, 782, 468, 333, 451, 696, 209, 25, 193, 784, 81, 534, 990, 635, 910, 952, 595, 508, 594, 148, 164, 893, 433, 925, 918, 804,
652, 981, 24, 354, 290, 795, 904, 86, 735, 484, 548, 41, 508, 870, 999, 899, 983, 451, 874, 489, 548, 310, 677, 468, 676, 8, 550, 227, 381, 84
8, 54, 86, 584, 54, 370, 52, 279, 408, 194, 46, 758, 394, 871, 971, 9, 819, 20, 902, 85, 674, 27, 267, 983, 211, 653, 664, 410, 59, 573, 641,
229, 25, 754, 114, 982, 26, 667, 658, 65, 615, 74, 901, 68, 609, 701, 722, 578, 433, 475, 0, 506, 40, 357, 322, 180, 581, 911, 689, 241, 521,
614, 345, 778, 360, 357, 865, 108, 550, 900, 77, 372, 835, 445, 277, 359, 2, 84, 500, 90, 146, 103, 158, 462, 598, 76, 559, 423, 91, 658, 957,
354, 323, 825, 430, 576, 618, 687, 268, 815, 720, 388, 576, 627, 633, 566, 502, 62, 957, 991, 745, 930, 852, 89, 672, 333, 129, 34, 153, 708,
982, 359, 547, 286, 534, 774, 801, 145, 412, 49, 661, 941, 193, 416]
[495, 310, 5, 976, 821, 910, 198, 865, 381, 762, 835, 634, 424, 862, 772, 156, 35, 984, 545, 885, 165, 829, 480, 730, 576, 976, 543, 681, 958,
483, 844, 829, 504, 539, 595, 375, 829, 590, 646, 838, 866, 298, 791, 256, 581, 97, 651, 903, 700, 840, 784, 829, 426, 30, 168, 993, 991, 932,
737, 224, 563, 982, 272, 988, 866, 514, 221, 580, 876, 310, 892, 898, 814, 327, 867, 824, 331, 365, 566, 256, 221, 26, 871, 105, 880, 598, 18
3, 999, 666, 564, 178, 700, 143, 268, 745, 959, 728, 741, 403, 434, 60, 484, 718, 442, 571, 217, 965, 593, 611, 12, 895, 231, 859, 84, 600, 15
5, 893, 491, 694, 389, 949, 278, 996, 303, 163, 315, 512, 255, 502, 112, 209, 882, 753, 439, 286, 581, 94, 478, 107, 960, 886, 765, 180, 0, 41
1, 465, 789, 502, 582, 262, 948, 512, 887, 114, 874, 339, 654, 113, 724, 191, 19, 448, 103, 346, 726, 495, 124, 819, 966, 177, 760, 751, 953,
869, 19, 317, 115, 882, 404, 374, 273, 443, 922, 85, 49, 225, 57, 656, 517, 234, 185, 866, 730, 878, 468, 244, 261, 462, 868, 57, 105, 203, 15
2, 397, 949, 165, 203, 995, 781, 444, 308, 51, 972, 279, 901, 732, 89, 102, 412, 390, 196, 137, 695, 247, 206, 359, 75, 887, 560, 618, 969, 40
3, 137, 941, 134, 652, 778, 331, 947, 821, 971, 716, 548, 50, 855, 104, 368, 317, 994, 634]
```

- сформируйте вектор ($y_2 - x_1, \ldots, y_n - x_{n-1}$);

```
# №3.14.1
z1 = [t[2]-t[1] for t in zip(x[1:N-1], y[2:N])]; show(z1); println()
z2 = y[2:end].-x[1:end-1]; show(z2); println()
```

```
[-344, -141, 371, 114, 211, 133, 125, 39, -188, 431, -227, -66, 414, 560, 144, -26, 6, 213, -110, 94, 268, -428, -148, 528, 449, 159, 641, 54
7, 185, 219, 79, -368, -49, 242, 67, 139, 387, 470, 447, 364, -588, -40, 142, 42, -323, -331, 167, 557, 471, 160, 309, -302, -423, -337, 432,
314, -53, 31, -585, 500, 880, -310, 206, 398, 181, -230, -116, 667, 285, 699, 114, 733, -207, -123, 189, -579, -587, -29, -252, -373, -122, 70
7, -788, 447, -327, -735, 195, 14, -417, 154, 346, -147, -527, -159, 873, -7, 257, -145, 393, -448, -386, -281, -457, -412, -234, 91, 104, 63,
-298, 218, -237, 183, 76, 50, -72, 512, -357, 640, 303, 365, 224, 626, 251, -116, -93, 318, 209, -256, -282, -662, -89, 744, -380, 266, -321,
9, -196, 80, 693, -97, 554, -473, -664, 1, 406, 216, -139, 353, 237, 194, 398, -95, 88, 207, -319, 589, -502, 650, -710, -49, -161, -598, -37
6, 148, 62, -351, 819, 460, 137, 403, 429, 773, 288, -892, -372, -126, 361, -210, 29, -505, 83, 565, -780, -59, -325, -843, 579, 145, -601, -2
60, 589, 371, 876, 384, -256, 171, 316, 765, -101, -357, -395, 76, -162, 526, 74, -455, 38, 427, 121, -517, -379, 396, -339, 214, 464, -726, -
618, 24, -186, -431, -496, 129, -255, 144, -598, -916, 142, -370, -234, 880, -269, -196, 812, 100, 499, 70, -651, 588, 274, 685, 182, -226, -7
51, 710, -308, 319, -344, 53, 441]
[-344, -141, 371, 114, 211, 133, 125, 39, -188, 431, -227, -66, 414, 560, 144, -26, 6, 213, -110, 94, 268, -428, -148, 528, 449, 159, 641, 54
7, 185, 219, 79, -368, -49, 242, 67, 139, 387, 470, 447, 364, -588, -40, 142, 42, -323, -331, 167, 557, 471, 160, 309, -302, -423, -337, 432,
314, -53, 31, -585, 500, 880, -310, 206, 398, 181, -230, -116, 667, 285, 699, 114, 733, -207, -123, 189, -579, -587, -29, -252, -373, -122, 70
7, -788, 447, -327, -735, 195, 14, -417, 154, 346, -147, -527, -159, 873, -7, 257, -145, 393, -448, -386, -281, -457, -412, -234, 91, 104, 63,
-298, 218, -237, 183, 76, 50, -72, 512, -357, 640, 303, 365, 224, 626, 251, -116, -93, 318, 209, -256, -282, -662, -89, 744, -380, 266, -321,
9, -196, 80, 693, -97, 554, -473, -664, 1, 406, 216, -139, 353, 237, 194, 398, -95, 88, 207, -319, 589, -502, 650, -710, -49, -161, -598, -37
6, 148, 62, -351, 819, 460, 137, 403, 429, 773, 288, -892, -372, -126, 361, -210, 29, -505, 83, 565, -780, -59, -325, -843, 579, 145, -601, -2
60, 589, 371, 876, 384, -256, 171, 316, 765, -101, -357, -395, 76, -162, 526, 74, -455, 38, 427, 121, -517, -379, 396, -339, 214, 464, -726, -
618, 24, -186, -431, -496, 129, -255, 144, -598, -916, 142, -370, -234, 880, -269, -196, 812, 100, 499, 70, -651, 588, 274, 685, 182, -226, -7
51, 710, -308, 319, -344, 53, 441]
```

- сформируйте вектор ($x_1 + 2x_2 - x_3, x_2 + 2x_3 - x_4, \ldots, x_{n-2} + 2x_{n-1} - x_n$);

```
# №3.14.2
z1 = [x[i-1]+2*x[i]-x[i+1] for i in 2:N-1]; show(z1); println()
z2 = x[1:N-2]+2*x[2:N-1]-x[3:N]; show(z2); println()
```

```
[341, 649, 1320, 2040, 89, 1203, 474, 1838, 897, 1636, 1393, 1174, 860, 175, -844, 1685, 647, 2251, 576, 285, 1499, 2616, 447, 718, 1255, 53,
564, 382, 798, 1253, 1906, 1695, 986, 279, 1485, 920, 164, 456, 509, 1443, 2434, 520, 772, 397, 1648, 2311, 653, 257, 1097, 936, 1523, 1129, 9
02, 950, 930, 1941, 1588, 2261, 833, -315, 484, 1678, 1385, 683, 539, 1634, 1089, 66, -373, 1680, 412, 159, 1879, 1350, 1503, 2219, 1634, 101
7, 1548, 726, -417, 1517, 834, 1365, 1957, 1874, 1127, 2590, 675, 442, 139, 976, 2517, 341, 1072, 1155, 1539, 122, 187, 1249, 1969, 1814, 241
4, 1011, 1710, 1304, 1275, 491, 1196, 937, 1812, 142, 881, 623, 141, 2023, 870, -358, 1200, 322, 742, 195, 202, 901, 750, -472, 1168, 675, 116
5, 2804, 170, 1627, -43, 1739, 398, 1406, 461, -422, 2022, 752, 853, 1571, 1425, -45, 564, 1626, 1074, -475, 1419, 0, 2052, 367, 702, 1918, 17
3, 1221, -138, 1808, 428, 585, 1289, 1567, 1445, 969, 1383, -31, 972, 229, 432, 821, 101, 431, 1714, 2048, 650, 669, 1404, 526, 1541, 1141, 20
9, 1979, 531, 308, 2273, 682, -14, 1597, 1448, 640, 993, 279, -330, 994, 534, 279, 194, -43, 484, 1582, 191, 771, 1314, -53, 450, 2218, 1342,
175, 1543, 1109, 964, 1125, 1724, 408, 1178, 1867, 920, 913, 1197, 1327, 1263, 1508, -331, 985, 2194, 1551, 1753, 2545, 358, 1100, 1209, 557,
44, -368, 587, 2313, 1153, 1167, 585, 580, 1281, 2231, 679, 920, -151, 430, 2350, 911]
[341, 649, 1320, 2040, 89, 1203, 474, 1838, 897, 1636, 1393, 1174, 860, 175, -844, 1685, 647, 2251, 576, 285, 1499, 2616, 447, 718, 1255, 53,
564, 382, 798, 1253, 1906, 1695, 986, 279, 1485, 920, 164, 456, 509, 1443, 2434, 520, 772, 397, 1648, 2311, 653, 257, 1097, 936, 1523, 1129, 9
02, 950, 930, 1941, 1588, 2261, 833, -315, 484, 1678, 1385, 683, 539, 1634, 1089, 66, -373, 1680, 412, 159, 1879, 1350, 1503, 2219, 1634, 101
7, 1548, 726, -417, 1517, 834, 1365, 1957, 1874, 1127, 2590, 675, 442, 139, 976, 2517, 341, 1072, 1155, 1539, 122, 187, 1249, 1969, 1814, 241
4, 1011, 1710, 1304, 1275, 491, 1196, 937, 1812, 142, 881, 623, 141, 2023, 870, -358, 1200, 322, 742, 195, 202, 901, 750, -472, 1168, 675, 116
5, 2804, 170, 1627, -43, 1739, 398, 1406, 461, -422, 2022, 752, 853, 1571, 1425, -45, 564, 1626, 1074, -475, 1419, 0, 2052, 367, 702, 1918, 17
3, 1221, -138, 1808, 428, 585, 1289, 1567, 1445, 969, 1383, -31, 972, 229, 432, 821, 101, 431, 1714, 2048, 650, 669, 1404, 526, 1541, 1141, 20
9, 1979, 531, 308, 2273, 682, -14, 1597, 1448, 640, 993, 279, -330, 994, 534, 279, 194, -43, 484, 1582, 191, 771, 1314, -53, 450, 2218, 1342,
175, 1543, 1109, 964, 1125, 1724, 408, 1178, 1867, 920, 913, 1197, 1327, 1263, 1508, -331, 985, 2194, 1551, 1753, 2545, 358, 1100, 1209, 557,
44, -368, 587, 2313, 1153, 1167, 585, 580, 1281, 2231, 679, 920, -151, 430, 2350, 911]
```

- сформируйте вектор ($\sin(y_1)/\cos(x_2), \sin(y_2)/\cos(x_3), \ldots, \sin(y_{n-1})/\cos(x_n)$);

```
# №3.14.3
z1 = [sin(i[2]) / cos(i[1]) for i in zip(x[2:N], y[1:N-1])]; show(z1[1:50]); println()
z2 = [sin(y[i-1]) / cos(x[i]) for i in 2:N]; show(z2[1:50]); println()
```

```
[-11.675086614106473, -3.530397057867017, 0.9686260458677545, 197.00859535910598, 1.537160325035007, -5.659141790846066, 0.08769382692916872,
-2.681228687467314, 2.5366731055407357, 1.0074288179992754, -0.6184106451212102, 1.7830995276083714, -1.9966941875935436, 1.1060470342170388,
2.863755929880605, 1.547870367662984, -0.8035578794500278, 0.994583178735657, 3.228892848516964, 3.5795332878478665, -1.0009765858187363, 4.94
6297892540829, -0.9623090852837307, 1.2939421845987025, -1.1843694325942733, -1.289509948335625, -0.5572869006126295, -0.7374192141352354, -0.
18762216468291448, 1.0810123231905624, 4.28920124406658, 0.4284438240202893, 2.342249197872688, -0.9841244160348341, -2.316513181005651, 2.542
544339545724, -0.3722810905014775 6, -4.537988622531194, -1.1599279704619232, 0.7228104404527134, 18.097022488557943, 0.7039921195590009, -2.92
72589328377605, -1.7764368686701553, -0.7748108504709241, 0.5868347188833204, -11.074026567444491, 7.157398413685215, -1.4174975520182251, -1
4.014915427447852]
[-11.675086614106473, -3.530397057867017, 0.9686260458677545, 197.00859535910598, 1.537160325035007, -5.659141790846066, 0.08769382692916872,
-2.681228687467314, 2.5366731055407357, 1.0074288179992754, -0.6184106451212102, 1.7830995276083714, -1.9966941875935436, 1.1060470342170388,
2.863755929880605, 1.547870367662984, -0.8035578794500278, 0.994583178735657, 3.228892848516964, 3.5795332878478665, -1.0009765858187363, 4.94
6297892540829, -0.9623090852837307, 1.2939421845987025, -1.1843694325942733, -1.289509948335625, -0.5572869006126295, -0.7374192141352354, -0.
18762216468291448, 1.0810123231905624, 4.28920124406658, 0.4284438240202893, 2.342249197872688, -0.9841244160348341, -2.316513181005651, 2.542
544339545724, -0.37228109050147756, -4.537988622531194, -1.1599279704619232, 0.7228104404527134, 18.097022488557943, 0.7039921195590009, -2.92
72589328377605, -1.7764368686701553, -0.7748108504709241, 0.5868347188833204, -11.074026567444491, 7.157398413685215, -1.4174975520182251, -1
4.014915427447852]
```

- вычислите $\sum_{i=1}^{n-1} \frac{e^{-x_{i+1}}}{x_i + 10}$

```
# №3.14.4
z1 = [exp(-x[i+1])/(x[i]+10) for i in 1:N-1]; show(sum(z1)); println()
z2 = sum(map(z -> exp(-z), x[2:N]) ./ (x[1:N-1] .+ 10)); show(z2); println()
```

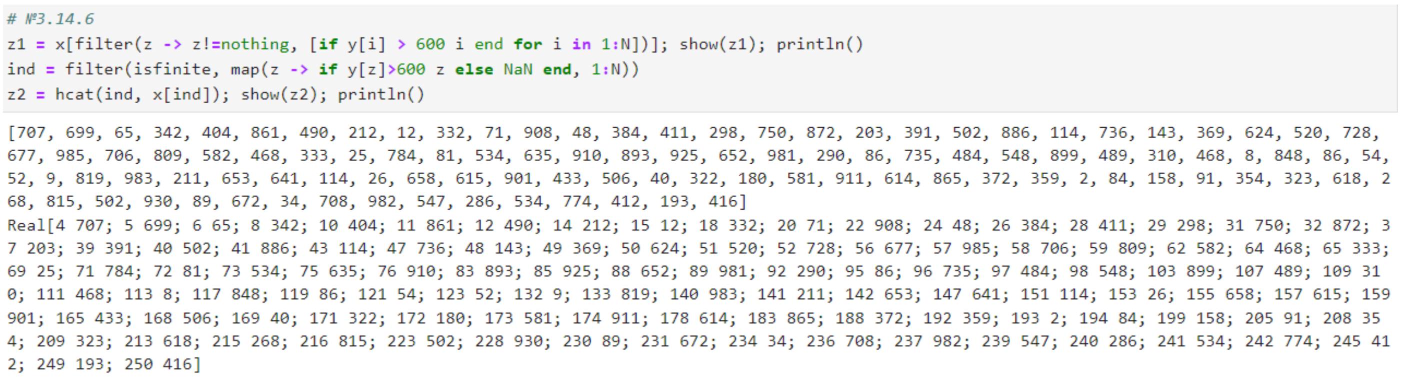
```
0.002429260447164512
0.002429260447164512
```

- выберите элементы вектора y, значения которых больше 600, и выведите на экран; определите индексы этих элементов;

```
# №3.14.5
z1 = filter(z -> z[2] > 600, [[i,y[i]] for i in 1:N]); show(z1); println()
ind = filter(isfinite, map(z -> if y[z]>600 z else NaN end, 1:N))
z2 = hcat(ind, y[ind]); show(z2); println()
```

```
[[4, 976], [5, 821], [6, 910], [8, 865], [10, 762], [11, 835], [12, 634], [14, 862], [15, 772], [18, 984], [20, 885], [22, 829], [24, 730], [2
6, 976], [28, 681], [29, 958], [31, 844], [32, 829], [37, 829], [39, 646], [40, 838], [41, 866], [43, 791], [47, 651], [48, 903], [49, 700],
[50, 840], [51, 784], [52, 829], [56, 993], [57, 991], [58, 932], [59, 737], [62, 982], [64, 988], [65, 866], [69, 876], [71, 892], [72, 898],
[73, 814], [75, 867], [76, 824], [83, 871], [85, 880], [88, 999], [89, 666], [92, 700], [95, 745], [96, 959], [97, 728], [98, 741], [103, 71
8], [107, 965], [109, 611], [111, 895], [113, 859], [117, 893], [119, 694], [121, 949], [123, 996], [132, 882], [133, 753], [140, 960], [141,
886], [142, 765], [147, 789], [151, 948], [153, 887], [155, 874], [157, 654], [159, 724], [165, 726], [168, 819], [169, 966], [171, 760], [17
2, 751], [173, 953], [174, 869], [178, 882], [183, 922], [188, 656], [192, 866], [193, 730], [194, 878], [199, 868], [205, 949], [208, 995],
[209, 781], [213, 972], [215, 901], [216, 732], [223, 695], [228, 887], [230, 618], [231, 969], [234, 941], [236, 652], [237, 778], [239, 94
7], [240, 821], [241, 971], [242, 716], [245, 855], [249, 994], [250, 634]]
Real[4 976; 5 821; 6 910; 8 865; 10 762; 11 835; 12 634; 14 862; 15 772; 18 984; 20 885; 22 829; 24 730; 26 976; 28 681; 29 958; 31 844; 32 82
9; 37 829; 39 646; 40 838; 41 866; 43 791; 47 651; 48 903; 49 700; 50 840; 51 784; 52 829; 56 993; 57 991; 58 932; 59 737; 62 982; 64 988; 65
866; 69 876; 71 892; 72 898; 73 814; 75 867; 76 824; 83 871; 85 880; 88 999; 89 666; 92 700; 95 745; 96 959; 97 728; 98 741; 103 718; 107 965;
109 611; 111 895; 113 859; 117 893; 119 694; 121 949; 123 996; 132 882; 133 753; 140 960; 141 886; 142 765; 147 789; 151 948; 153 887; 155 87
4; 157 654; 159 724; 165 726; 168 819; 169 966; 171 760; 172 751; 173 953; 174 869; 178 882; 183 922; 188 656; 192 866; 193 730; 194 878; 199
868; 205 949; 208 995; 209 781; 213 972; 215 901; 216 732; 223 695; 228 887; 230 618; 231 969; 234 941; 236 652; 237 778; 239 947; 240 821; 24
1 971; 242 716; 245 855; 249 994; 250 634]
```

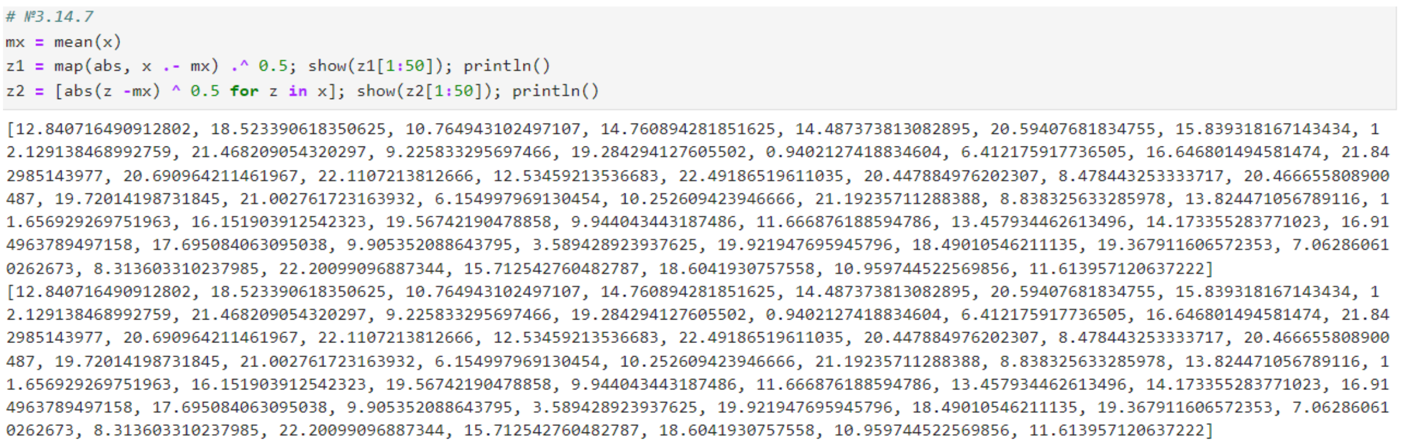
- определите значения вектора x, соответствующие значениям вектора y, значения которых больше 600 (под соответствием понимается расположение на аналогичных индексных позициях);



```
# №3.14.6
z1 = x[filter(z -> z!=nothing, [if y[i] > 600 i end for i in 1:N])]; show(z1); println()
ind = filter(isfinite, map(z -> if y[z]>600 z else NaN end, 1:N))
z2 = hcat(ind, x[ind]); show(z2); println()
```

```
[707, 699, 65, 342, 404, 861, 490, 212, 12, 332, 71, 908, 48, 384, 411, 298, 750, 872, 203, 391, 502, 886, 114, 736, 143, 369, 624, 520, 728,
677, 985, 706, 809, 582, 468, 333, 25, 784, 81, 534, 635, 910, 893, 925, 652, 981, 290, 86, 735, 484, 548, 899, 489, 310, 468, 8, 848, 86, 54,
52, 9, 819, 983, 211, 653, 641, 114, 26, 658, 615, 901, 433, 506, 40, 322, 180, 581, 911, 614, 865, 372, 359, 2, 84, 158, 91, 354, 323, 618, 2
68, 815, 502, 930, 89, 672, 34, 708, 982, 547, 286, 534, 774, 412, 193, 416]
Real[4 707; 5 699; 6 65; 8 342; 10 404; 11 861; 12 490; 14 212; 15 12; 18 332; 20 71; 22 908; 24 48; 26 384; 28 411; 29 298; 31 750; 32 872; 3
7 203; 39 391; 40 502; 41 886; 43 114; 47 736; 48 143; 49 369; 50 624; 51 520; 52 728; 56 677; 57 985; 58 706; 59 809; 62 582; 64 468; 65 333;
69 25; 71 784; 72 81; 73 534; 75 635; 76 910; 83 893; 85 925; 88 652; 89 981; 92 290; 95 86; 96 735; 97 484; 98 548; 103 899; 107 489; 109 31
0; 111 468; 113 8; 117 848; 119 86; 121 54; 123 52; 132 9; 133 819; 140 983; 141 211; 142 653; 147 641; 151 114; 153 26; 155 658; 157 615; 159
901; 165 433; 168 506; 169 40; 171 322; 172 180; 173 581; 174 911; 178 614; 183 865; 188 372; 192 359; 193 2; 194 84; 199 158; 205 91; 208 35
4; 209 323; 213 618; 215 268; 216 815; 223 502; 228 930; 230 89; 231 672; 234 34; 236 708; 237 982; 239 547; 240 286; 241 534; 242 774; 245 41
2; 249 193; 250 416]
```

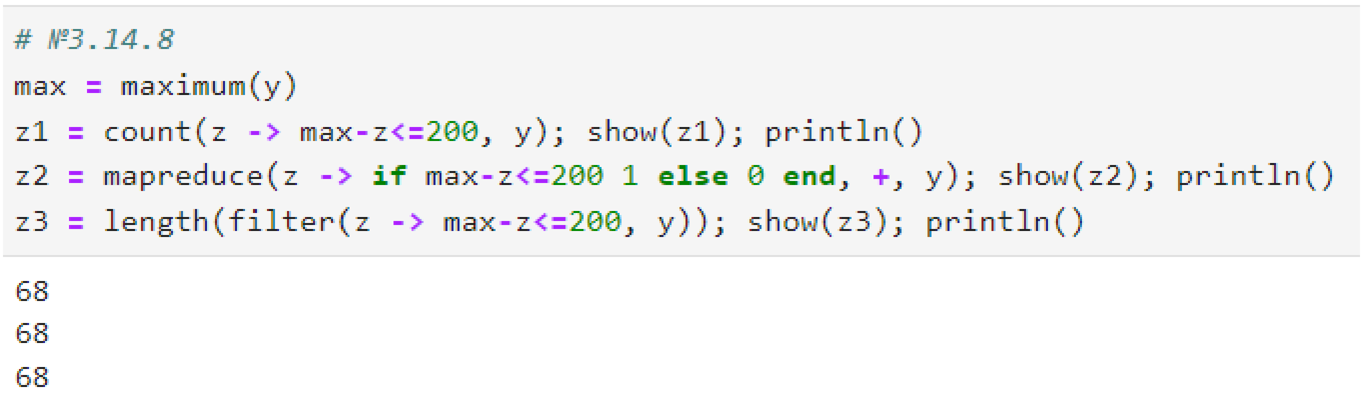
- сформируйте вектор ($|x_1 - x|^{1/2}, |x_2 - x|^{1/2}, \dots, |x_n - x|^{1/2}$), где x обозначает среднее значение вектора $x = (x_1, x_2, \dots, x_n)$;



```
# №3.14.7
mx = mean(x)
z1 = map(abs, x .- mx) .^ 0.5; show(z1[1:50]); println()
z2 = [abs(z -mx) ^ 0.5 for z in x]; show(z2[1:50]); println()
```

```
[12.840716490912802, 18.523390618350625, 10.764943102497107, 14.760894281851625, 14.487373813082895, 20.59407681834755, 15.839318167143434, 1
2.129138468992759, 21.468209054320297, 9.225833295697466, 19.284294127605502, 0.9402127418834604, 6.412175917736505, 16.646801494581474, 21.84
2985143977, 20.690964211461967, 22.1107213812666, 12.53459213536683, 22.49186519611035, 20.447884976202307, 8.478443253333717, 20.466655808900
487, 19.72014198731845, 21.002761723163932, 6.154997969130454, 10.252609423946666, 21.19235711288388, 8.838325633285978, 13.824471056789116, 1
1.656929269751963, 16.151903912542323, 19.56742190478858, 9.944043443187486, 11.666876188594786, 13.457934462613496, 14.173355283771023, 16.91
4963789497158, 17.695084063095038, 9.905352088643795, 3.589428923937625, 19.921947695945796, 18.49010546211135, 19.367911606572353, 7.06286061
0262673, 8.313603310237985, 22.20099096887344, 15.712542760482787, 18.6041930757558, 10.959744522569856, 11.613957120637222]
[12.840716490912802, 18.523390618350625, 10.764943102497107, 14.760894281851625, 14.487373813082895, 20.59407681834755, 15.839318167143434, 1
2.129138468992759, 21.468209054320297, 9.225833295697466, 19.284294127605502, 0.9402127418834604, 6.412175917736505, 16.646801494581474, 21.84
2985143977, 20.690964211461967, 22.1107213812666, 12.53459213536683, 22.49186519611035, 20.447884976202307, 8.478443253333717, 20.466655808900
487, 19.72014198731845, 21.002761723163932, 6.154997969130454, 10.252609423946666, 21.19235711288388, 8.838325633285978, 13.824471056789116, 1
1.656929269751963, 16.151903912542323, 19.56742190478858, 9.944043443187486, 11.666876188594786, 13.457934462613496, 14.173355283771023, 16.91
4963789497158, 17.695084063095038, 9.905352088643795, 3.589428923937625, 19.921947695945796, 18.49010546211135, 19.367911606572353, 7.06286061
0262673, 8.313603310237985, 22.20099096887344, 15.712542760482787, 18.6041930757558, 10.959744522569856, 11.613957120637222]
```

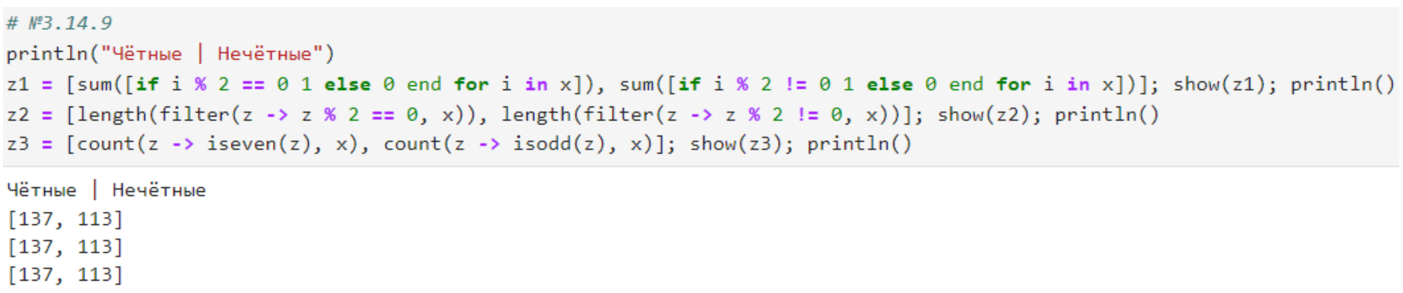
- определите, сколько элементов вектора y отстоят от максимального значения не более, чем на 200;



```
# №3.14.8
max = maximum(y)
z1 = count(z -> max-z<=200, y); show(z1); println()
z2 = mapreduce(z -> if max-z<=200 1 else 0 end, +, y); show(z2); println()
z3 = length(filter(z -> max-z<=200, y)); show(z3); println()
```

```
68
68
68
```

- определите, сколько чётных и нечётных элементов вектора x;



```
# №3.14.9
println("Чётные | Нечётные")
z1 = [sum([if i % 2 == 0 1 else 0 end for i in x]), sum([if i % 2 != 0 1 else 0 end for i in x])]; show(z1); println()
z2 = [length(filter(z -> z % 2 == 0, x)), length(filter(z -> z % 2 != 0, x))]; show(z2); println()
z3 = [count(z -> iseven(z), x), count(z -> isodd(z), x)]; show(z3); println()
```

```
Чётные | Нечётные
[137, 113]
[137, 113]
[137, 113]
```

- определите, сколько элементов вектора x кратны 7;

```
# №3.14.10
z1 = sum([if i % 7 == 0 1 else 0 end for i in x]); show(z1); println()
z2 = length(filter(z -> z % 7 == 0, x)); show(z2); println()
z3 = mapreduce(z -> z % 7 == 0, +, x); show(z3); println()
```

```
31
31
31
```

- отсортируйте элементы вектора x в порядке возрастания элементов вектора y;

```
# №3.14.11 (ОЧЕНЬ НЕ ТОЧНОЕ УСЛОВИЕ!)
ys = [z[2] for z in sort([[y[i],i] for i in 1:N])]; print("Индексы после сортировки: "); show(ys); println()
z1 = [x[ys[i]] for i in 1:N]; show(z1); println()
z2 = x[ys]; show(z2); println()
```

```
Индексы после сортировки: [144, 3, 110, 161, 175, 82, 54, 17, 185, 244, 212, 187, 200, 101, 227, 114, 184, 217, 137, 46, 218, 163, 246, 84, 20
1, 139, 130, 158, 154, 177, 167, 235, 222, 233, 93, 203, 116, 16, 125, 21, 206, 55, 170, 91, 143, 87, 191, 160, 221, 7, 202, 207, 225, 131, 10
6, 67, 81, 60, 186, 112, 190, 196, 224, 128, 44, 80, 197, 150, 94, 63, 181, 122, 214, 135, 42, 124, 211, 2, 70, 126, 176, 248, 74, 77, 238, 15
6, 164, 226, 78, 247, 180, 36, 9, 120, 220, 204, 99, 232, 179, 145, 219, 13, 53, 100, 134, 104, 182, 210, 162, 198, 146, 195, 138, 23, 30, 10
2, 118, 1, 166, 129, 148, 33, 127, 152, 66, 189, 34, 27, 19, 243, 229, 61, 90, 79, 105, 25, 68, 45, 136, 149, 38, 108, 35, 86, 115, 109, 230,
12, 250, 39, 47, 236, 157, 188, 89, 28, 119, 223, 49, 92, 242, 103, 159, 165, 97, 24, 193, 216, 59, 98, 95, 172, 133, 171, 10, 142, 15, 237, 2
09, 51, 147, 43, 73, 168, 5, 240, 76, 22, 32, 37, 52, 11, 40, 50, 31, 245, 113, 14, 8, 41, 65, 192, 75, 199, 174, 83, 155, 69, 194, 85, 132, 1
78, 20, 141, 153, 228, 71, 117, 111, 72, 215, 48, 6, 183, 58, 234, 239, 151, 121, 205, 173, 29, 96, 140, 107, 169, 231, 241, 213, 4, 26, 62, 1
8, 64, 57, 56, 249, 208, 123, 88]
[410, 605, 677, 609, 689, 164, 505, 978, 550, 145, 576, 77, 462, 870, 745, 550, 108, 720, 674, 982, 388, 722, 49, 433, 598, 267, 871, 74, 667,
521, 0, 153, 566, 129, 795, 559, 381, 61, 408, 561, 658, 561, 357, 354, 664, 804, 277, 68, 633, 740, 76, 957, 957, 971, 874, 696, 148, 63, 90
0, 676, 445, 90, 62, 758, 539, 594, 146, 754, 904, 782, 360, 370, 687, 902, 831, 279, 430, 146, 193, 194, 241, 941, 990, 952, 359, 65, 578, 99
1, 595, 661, 778, 690, 950, 584, 627, 423, 41, 333, 345, 59, 576, 448, 453, 508, 20, 983, 357, 825, 701, 103, 573, 500, 27, 878, 625, 999, 54,
654, 475, 394, 229, 588, 46, 982, 451, 835, 353, 40, 995, 801, 852, 102, 24, 508, 451, 527, 209, 420, 85, 25, 176, 548, 308, 918, 227, 310, 8
9, 490, 416, 391, 736, 708, 615, 372, 981, 411, 86, 502, 369, 290, 774, 899, 901, 433, 484, 48, 2, 815, 809, 548, 86, 180, 819, 322, 404, 653,
12, 982, 323, 520, 641, 114, 534, 506, 699, 286, 910, 908, 872, 203, 728, 861, 502, 624, 750, 412, 8, 212, 342, 886, 333, 359, 635, 158, 911,
893, 658, 25, 84, 925, 9, 614, 71, 211, 26, 930, 784, 848, 468, 81, 268, 143, 65, 865, 706, 34, 547, 114, 54, 91, 581, 298, 735, 983, 489, 40,
672, 534, 618, 707, 384, 582, 332, 468, 985, 677, 193, 354, 52, 652]
[410, 605, 677, 609, 689, 164, 505, 978, 550, 145, 576, 77, 462, 870, 745, 550, 108, 720, 674, 982, 388, 722, 49, 433, 598, 267, 871, 74, 667,
521, 0, 153, 566, 129, 795, 559, 381, 61, 408, 561, 658, 561, 357, 354, 664, 804, 277, 68, 633, 740, 76, 957, 957, 971, 874, 696, 148, 63, 90
0, 676, 445, 90, 62, 758, 539, 594, 146, 754, 904, 782, 360, 370, 687, 902, 831, 279, 430, 146, 193, 194, 241, 941, 990, 952, 359, 65, 578, 99
1, 595, 661, 778, 690, 950, 584, 627, 423, 41, 333, 345, 59, 576, 448, 453, 508, 20, 983, 357, 825, 701, 103, 573, 500, 27, 878, 625, 999, 54,
654, 475, 394, 229, 588, 46, 982, 451, 835, 353, 40, 995, 801, 852, 102, 24, 508, 451, 527, 209, 420, 85, 25, 176, 548, 308, 918, 227, 310, 8
9, 490, 416, 391, 736, 708, 615, 372, 981, 411, 86, 502, 369, 290, 774, 899, 901, 433, 484, 48, 2, 815, 809, 548, 86, 180, 819, 322, 404, 653,
12, 982, 323, 520, 641, 114, 534, 506, 699, 286, 910, 908, 872, 203, 728, 861, 502, 624, 750, 412, 8, 212, 342, 886, 333, 359, 635, 158, 911,
893, 658, 25, 84, 925, 9, 614, 71, 211, 26, 930, 784, 848, 468, 81, 268, 143, 65, 865, 706, 34, 547, 114, 54, 91, 581, 298, 735, 983, 489, 40,
672, 534, 618, 707, 384, 582, 332, 468, 985, 677, 193, 354, 52, 652]
```

- выведите элементы вектора x, которые входят в десятку наибольших (top-10)?

```
# №3.14.12 (От большего к меньшему)
z1 = Vector(1:10)
x1 = copy(x)
for i in 1:10
    ind = argmax(x1); z1[i] = x1[ind]; splice!(x1, ind)
end; show(z1); println()
z2 = sort(x)[end:-1:end-9]; show(z2); println()
z3 = x[sortperm(x)[end:-1:end-9]]; show(z3); println()
```

```
[999, 995, 991, 990, 985, 983, 983, 982, 982, 982]
[999, 995, 991, 990, 985, 983, 983, 982, 982, 982]
[999, 995, 991, 990, 985, 983, 983, 982, 982, 982]
```

- сформируйте вектор, содержащий только уникальные (неповторяющиеся) элементы вектора x.

```
# №3.14.13 (для проверки отсортировал)
z1 = Vector{Int16}([])
for i in x
    if i ∉ z1
        append!(z1, i)
    end
end; show(sort(z1)); println()
z2 = sort(collect(Set(x))); show(z2); println()
z3 = sort(unique(x)); show(z3); println()
```

```
Int16[0, 2, 8, 9, 12, 20, 24, 25, 26, 27, 34, 40, 41, 46, 48, 49, 52, 54, 59, 61, 62, 63, 65, 68, 71, 74, 76, 77, 81, 84, 85, 86, 89, 90, 91,
102, 103, 108, 114, 129, 143, 145, 146, 148, 153, 158, 164, 176, 180, 193, 194, 203, 209, 211, 212, 227, 229, 241, 267, 268, 277, 279, 286, 29
0, 298, 308, 310, 322, 323, 332, 333, 342, 345, 353, 354, 357, 359, 360, 369, 370, 372, 381, 384, 388, 391, 394, 404, 408, 410, 411, 412, 416,
420, 423, 430, 433, 445, 448, 451, 453, 462, 468, 475, 484, 489, 490, 500, 502, 505, 506, 508, 520, 521, 527, 534, 539, 547, 548, 550, 559, 56
1, 566, 573, 576, 578, 581, 582, 584, 588, 594, 595, 598, 605, 609, 614, 615, 618, 624, 625, 627, 633, 635, 641, 652, 653, 654, 658, 661, 664,
667, 672, 674, 676, 677, 687, 689, 690, 696, 699, 701, 706, 707, 708, 720, 722, 728, 735, 736, 740, 745, 750, 754, 758, 774, 778, 782, 784, 79
5, 801, 804, 809, 815, 819, 825, 831, 835, 848, 852, 861, 865, 870, 871, 872, 874, 878, 886, 893, 899, 900, 901, 902, 904, 908, 910, 911, 918,
925, 930, 941, 950, 952, 957, 971, 978, 981, 982, 983, 985, 990, 991, 995, 999]
[0, 2, 8, 9, 12, 20, 24, 25, 26, 27, 34, 40, 41, 46, 48, 49, 52, 54, 59, 61, 62, 63, 65, 68, 71, 74, 76, 77, 81, 84, 85, 86, 89, 90, 91, 102,
103, 108, 114, 129, 143, 145, 146, 148, 153, 158, 164, 176, 180, 193, 194, 203, 209, 211, 212, 227, 229, 241, 267, 268, 277, 279, 286, 290, 29
8, 308, 310, 322, 323, 332, 333, 342, 345, 353, 354, 357, 359, 360, 369, 370, 372, 381, 384, 388, 391, 394, 404, 408, 410, 411, 412, 416, 420,
423, 430, 433, 445, 448, 451, 453, 462, 468, 475, 484, 489, 490, 500, 502, 505, 506, 508, 520, 521, 527, 534, 539, 547, 548, 550, 559, 561, 56
6, 573, 576, 578, 581, 582, 584, 588, 594, 595, 598, 605, 609, 614, 615, 618, 624, 625, 627, 633, 635, 641, 652, 653, 654, 658, 661, 664, 667,
672, 674, 676, 677, 687, 689, 690, 696, 699, 701, 706, 707, 708, 720, 722, 728, 735, 736, 740, 745, 750, 754, 758, 774, 778, 782, 784, 795, 80
1, 804, 809, 815, 819, 825, 831, 835, 848, 852, 861, 865, 870, 871, 872, 874, 878, 886, 893, 899, 900, 901, 902, 904, 908, 910, 911, 918, 925,
930, 941, 950, 952, 957, 971, 978, 981, 982, 983, 985, 990, 991, 995, 999]
[0, 2, 8, 9, 12, 20, 24, 25, 26, 27, 34, 40, 41, 46, 48, 49, 52, 54, 59, 61, 62, 63, 65, 68, 71, 74, 76, 77, 81, 84, 85, 86, 89, 90, 91, 102,
103, 108, 114, 129, 143, 145, 146, 148, 153, 158, 164, 176, 180, 193, 194, 203, 209, 211, 212, 227, 229, 241, 267, 268, 277, 279, 286, 290, 29
8, 308, 310, 322, 323, 332, 333, 342, 345, 353, 354, 357, 359, 360, 369, 370, 372, 381, 384, 388, 391, 394, 404, 408, 410, 411, 412, 416, 420,
423, 430, 433, 445, 448, 451, 453, 462, 468, 475, 484, 489, 490, 500, 502, 505, 506, 508, 520, 521, 527, 534, 539, 547, 548, 550, 559, 561, 56
6, 573, 576, 578, 581, 582, 584, 588, 594, 595, 598, 605, 609, 614, 615, 618, 624, 625, 627, 633, 635, 641, 652, 653, 654, 658, 661, 664, 667,
672, 674, 676, 677, 687, 689, 690, 696, 699, 701, 706, 707, 708, 720, 722, 728, 735, 736, 740, 745, 750, 754, 758, 774, 778, 782, 784, 795, 80
1, 804, 809, 815, 819, 825, 831, 835, 848, 852, 861, 865, 870, 871, 872, 874, 878, 886, 893, 899, 900, 901, 902, 904, 908, 910, 911, 918, 925,
930, 941, 950, 952, 957, 971, 978, 981, 982, 983, 985, 990, 991, 995, 999]
```

4. Создайте массив squares, в котором будут храниться квадраты всех целых чисел от 1 до 100.

```
# №4
squaresV1 = [i^2 for i in 1:100]
show(squaresV1)
squaresV2 = Vector(1:100) .^ 2
println(); show(squaresV2)
```

```
[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, 3
025, 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]
[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, 3
025, 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]
```

5. Подключите пакет Primes (функции для вычисления простых чисел). Сгенерируйте массив myprimes, в котором будут храниться первые 168 простых чисел. Определите 89-е наименьшее простое число. Получите срез массива с 89-го до 99-го элемента включительно, содержащий наименьшие простые числа.

```
# №5
using Primes
myprimes = [Primes.prime(i) for i in 1:168]
show(myprimes)
print("\n89-е простое число: ", myprimes[89])
print("\n", myprimes[89:99])
```

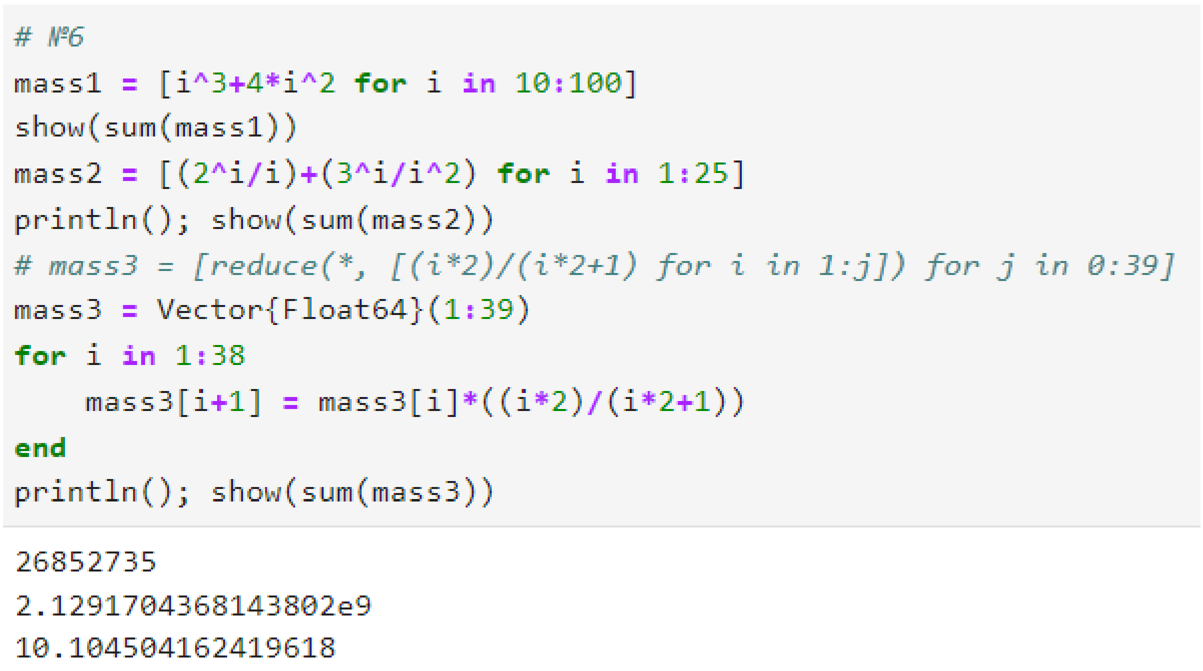
```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139,
149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 30
7, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467,
479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 65
9, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857,
859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]
 89-е простое число: 461
[461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]
```

6. Вычислите следующие выражения:

6.1) $\sum\limits_{i=10}^{100} (i^3 + 4i^2)$;

6.2) $\sum\limits_{i=1}^{M} \left(\frac{2^i}{i} + \frac{3^i}{i^2}\right), M = 25$;

6.3) $1 + \frac{2}{3} + \left(\frac{2}{3}\frac{4}{5}\right) + \left(\frac{2}{3}\frac{4}{5}\frac{6}{7}\right) + \dots + \left(\frac{2}{3}\frac{4}{5}\dots\frac{38}{39}\right)$.



```
# №6
mass1 = [i^3+4*i^2 for i in 10:100]
show(sum(mass1))
mass2 = [(2^i/i)+(3^i/i^2) for i in 1:25]
println(); show(sum(mass2))
# mass3 = [reduce(*, [(i*2)/(i*2+1) for i in 1:j]) for j in 0:39]
mass3 = Vector{Float64}(1:39)
for i in 1:38
    mass3[i+1] = mass3[i]*((i*2)/(i*2+1))
end
println(); show(sum(mass3))
```

```
26852735
2.1291704368143802e9
10.104504162419618
```

Выводы

В ходе работы были изучены и повторены принципы работы с массивами, множествами, картежами и словарями, а так методы и математические операторы для работы с ними.