РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖ|БЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра теории вероятностей и кибербезопасности

Лабораторная работа № 2 | Структуры данных

Студент: Абд эль хай Мохамад
Номер: 1032215163
Група: НПИбд-01-21

Москва 2025

2.1. Цель работы

Основная цель работы — изучить несколько структур данных, реализованных в Julia,научиться применять их и операции над ними для решения задач.

Примеры

2.2.1. Кортежи

In [12]:
```
favlang = ("python", "c++", "Julia")

x1 = (1,2,2)

x2 = (1,2.0,"Var")

x3 = (a=2, b=1+2)

# Опираций над картежами

length(x2)
println()
# обратиться к элементам кортежа x2:
println("x2[1] = ", x2[1])

# произвести какую-либо операцию (сложение)
# с вторым и третьим элементами кортежа x1

c= x1[2] + x1[3]
println("c= x1[2] + x1[3] = ", c)
```

```
# обращение к элементам именованного кортежа x3:
println("x.a = ",x3.a)

# проверка вхождения элементов tmp и 0 в кортеж x2
res = 0 ∈ x2
println("Is 0 in tuple x2? ",res)
```

```
x2[1] = 1
c= x1[2] + x1[3] = 4
x.a = 2
Is 0 in tuple x2? false
```

2.2.2. Словари

In [26]:
```
d1 = Dict("A"=>4,"B"=>2)
println(typeof(d1))

d2 = Dict([(1322, 20),(1321,14),("A",3)])

push!(d1, "G"=> 350)
println(d1)

pop!(d1, "A")
println(d1)

res = haskey(d1, "A")
println("Does d1 has key \"A\"? ", res)
res ="G" in keys(d1)
println("Does d1 has key \"G\"? ", res)



#If the same key is present in another collection, the value for that key wi
merge(d1,d2)
```

```
Dict{String, Int64}
Dict("B" => 2, "A" => 4, "G" => 350)
Dict("B" => 2, "G" => 350)
Does d1 has key "A"? false
Does d1 has key "G"? true
```

Out[26]:
```
Dict{Any, Int64} with 5 entries:
  "B"  => 2
  "A"  => 3
  1322 => 20
  "G"  => 350
  1321 => 14
```

2.2.3. Множества

In [40]:
```
A = Set([1,2,3,4,5])

s1 = Set([1,2,4]);
s2 = Set([3,4]);
```

```
println("are these twos sets equal? " , issetequal(s1,s2))

# объединение множеств:
u = union(s1,s2)
println(u)
# пересечение множеств:
i = intersect(s1,s2)
println(i)

push!(s2,1)
i = intersect(s1,s2)
println(i)
```

```
are these twos sets equal? false
Set([4, 2, 3, 1])
Set([4])
Set([4, 1])
```

2.2.4. Массивы

In [44]:

```
# arrayes
#column vecotre
cv = [1,2,3]
println("Массив cv ",cv)
#raw vectore
rv = [ 1 2 3]
println("Массив rev ",rv)
#2X2 array raw by raw
m = [1 3 5;2 4 6]
println("Массив m ",m)
#3D arrays with 3 raws 2 columns and 3 layers generate random arrays using r
a1 = rand(3,2,3)
println("Массив a1 ",a1)

#every elemnt using this symbol in plase of raw or column :
println("массив м ",m[:,3])
# also
m[1:2,2:3]
#julia uses column-major order
#m[:]
#arrays can be muted
m[1,2] = 9
#like typles i can use the broadcast function with arrays
println(m .+1)

#rawXcolumn size of an array
println("размер массив m: ",size(m))
#the amoun of elemnts in the array
println("длина массив m: ",length(m))

#mini mac and min/max extrema
println("minimum m: ",minimum(m))
println("maximum m: ",maximum(m))
println("extrema m: ",extrema(m))
```

```
#transpose an array
println("transpose m")
println(transpose(m))


#function for vecotre arrays only
println(cv)
println("sort cv",sort(cv, rev=true))

#function to create array quickly
#using fill create a 5X5 matrix of pi
println(fill(π,5,5))
#a 5X5 matrix of 0's
println(zeros(Int64, 5,5))
#matrix of trues or flase
println(falses(4,4))

#check if the element is or is not in array
A = [1,2,"Igor"]
B = [4,3,2,"Margo"]
println(A)
println(B)
#is 1 in A?
println("is 1 in A ",1 ∈ A)
println("is 1 not in A ",1 ∉ A)
```

```
Массив cv [1, 2, 3]
Массив rev [1 2 3]
Массив m [1 3 5; 2 4 6]
Массив a1 [0.5124940941481768 0.9610543127599882; 0.08433036944583039 0.4663
656978907994; 0.30865124283838574 0.9618258347486327;;; 0.3871897763342327
0.7877254254833421; 0.22618194238210187 0.9403041658968204; 0.78170829730055
52 0.9107845504140774;;; 0.601671080337214 0.9292769086994496; 0.04519127847
609217 0.38606047254254494; 0.3989637781807511 0.7568811253257747]
массив м [5, 6]
[2 10 6; 3 5 7]
размер массив m: (2, 3)
длина массив m: 6
minimum m: 1
maximum m: 9
extrema m: (1, 9)
transpose m
[1 2; 9 4; 5 6]
[1, 2, 3]
sort cv[3, 2, 1]
[π π π π π; π π π π π; π π π π π; π π π π π; π π π π π]
[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]
Bool[0 0 0 0; 0 0 0 0; 0 0 0 0; 0 0 0 0]
Any[1, 2, "Igor"]
Any[4, 3, 2, "Margo"]
is 1 in A true
is 1 not in A false
```

2.4. Задания для самостоятельного выполнения

1. Даны множества: A = {0, 3, 4, 9}, B = {1, 3, 4, 7}, C = {0, 1, 2, 4, 7, 8, 9}. Найти $P = A \cap B \cup A \cap B \cup A \cap C \cup B \cap C$.

In [48]:
```
A = ([0, 3, 4, 9])
B = ([1, 3, 4, 7])
C = ([0, 1, 2, 4, 7, 8, 9])

P =  union(union(union(intersect(A,B),intersect(A,B)),intersect(A,C)),inters
println(P)
```

```
[3, 4, 0, 9, 1, 7]
```

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

In [56]:
```
s1= Set([]);
println(typeof(s1))
s2= Set([false,false,true,true]);
println(typeof(s2))

union(s1,s2)
```

```
Set{Any}
Set{Bool}
```

Out[56]:
```
Set{Any} with 2 elements:
  false
  true
```

3. Создайте разными способами:

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

In [62]:
```
N = 22
A = collect(1:N)
println(A)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22]
```

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

In [59]:
```
B = [i for i in N:-1:0]
println(B)
```

```
[22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2,
1, 0]
```

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

In [97]:
```
tmp = [4,6,3]
```

Out[97]:
```
3-element Vector{Int64}:
 4
 6
 3
```

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

In [98]:
```
for i in 1:9
    pushfirst!(tmp, tmp[1])
end
println(tmp)
```

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

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

In [128…
```
tmp = [4,6,3]
base= []

for j in 1:3
    for i in 1:10
        # println(j)
        push!(base, tmp[j])
    end
end
println(base)
```

```
Any[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]
```

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

In [134…
```
tmp = [4, 6, 3]
arr1 = [fill(tmp[1], 11);repeat(tmp[2:end], 10)]
print(arr1)
```

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

3.8) массив, в котором первый элемент массива tmp встречается 10 раз подряд, второй элемент – 20 раз подряд, третий элемент

– 30 раз подряд;

In [135…
```
tmp = [4, 6, 3]
arr2 = [fill(tmp[1], 10); fill(tmp[2], 20); fill(tmp[3], 30);]
print(arr2)
```

```
[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, 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 раза; посчитайте в полученном векторе, сколько раз встречается цифра 6, и выведите это значение на экран;

In [136…
```
tmp = [4, 6, 3]
tmp_square = [[2^tmp[i] for i in 1:length(tmp)]; fill(2^tmp[3], 3);]

println(tmp_square)

# разделим все числа в массиве, чтобы найти количество цифр 6
count = 0
numbers = []  # Empty array of Ints
for i in 1:1:length(tmp_square)
    while tmp_square[i] != 0
        rem = tmp_square[i] % 10  # Modulo division - get last digit
        push!(numbers, rem)
        tmp_square[i] = div(tmp_square[i],10)
    end
end

# подсчитаем количество цифр 6
for i in 1:1:length(numbers)
    if numbers[i] == 6
        count += 1
    end
end
print(count)
```

```
[16, 64, 8, 8, 8, 8]
2
```

3.10) вектор значений $y = ex\cos(x)$ в точках x = 3, 3.1, 3.2, … , 6, найдите средне значение y;

In [137…
```
using Statistics
y = [exp(i)*cos(i) for i in 3:0.1:6]

mean(y)
```

Out[137… 53.11374594642971

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

In [111…
```
tmp = [4,6,3]
#repeat first element 10 times
for i in 1:9
    pushfirst!(tmp, tmp[1])
end
println(tmp)
#repeat every element 10 times
for j in 11:10:21
    for i in 1:9
        # println(j)
        pushfirst!(tmp, tmp[j])
    end
end
println(tmp)
```

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

3.12 вектор с элементами $\frac{2^i}{i}$, i = 1, 2, … , M, M = 25;

In [75]:
```
M = 25
vector_1 = [(2^i)/i for i=1:1:M]
print(vector_1)
```

```
[2.0, 2.0, 2.6666666666666665, 4.0, 6.4, 10.666666666666666, 18.285714285714
285, 32.0, 56.888888888888886, 102.4, 186.1818181818182, 341.3333333333333,
630.1538461538462, 1170.2857142857142, 2184.5333333333333, 4096.0, 7710.1176
47058823, 14563.555555555555, 27594.105263157893, 52428.8, 99864.3809523809
5, 190650.18181818182, 364722.0869565217, 699050.6666666666, 1.34217728e6]
```

3.13 вектор вида (”fn1”, ”fn2”, ..., ”fnN”), N = 30;

In [76]:
```
N = 30
vector_2 = [string("fn", i) for i=1:1:N]
print(vector_2)
```

```
["fn1", "fn2", "fn3", "fn4", "fn5", "fn6", "fn7", "fn8", "fn9", "fn10", "fn1
1", "fn12", "fn13", "fn14", "fn15", "fn16", "fn17", "fn18", "fn19", "fn20",
"fn21", "fn22", "fn23", "fn24", "fn25", "fn26", "fn27", "fn28", "fn29", "fn3
0"]
```

3.14 векторы x = (x1, x2, … , xn) и y = (y1, y2, … , yn) целочисленного типа длины n = 250 как случайные выборки из совокупности 0, 1, … , 999; на его основе:

In [77]:
```
using StatsBase
n = 250
x = sample(0:999, n, replace=false)
y = sample(0:999, n, replace=false)

# проверим длину выборки x
println(length(x))
```

```
println(x)
println(y)
```

```
250
[46, 451, 313, 626, 171, 42, 503, 432, 621, 323, 179, 433, 396, 124, 270, 46
3, 56, 341, 12, 223, 205, 346, 787, 668, 155, 255, 805, 203, 794, 535, 769,
17, 73, 176, 641, 991, 365, 320, 974, 619, 399, 252, 400, 57, 999, 373, 561,
291, 960, 670, 525, 379, 439, 380, 935, 456, 408, 184, 352, 928, 867, 271, 6
03, 358, 627, 766, 20, 663, 247, 530, 764, 117, 90, 256, 474, 771, 540, 962,
231, 554, 628, 629, 569, 141, 749, 845, 295, 102, 610, 531, 871, 788, 378, 9
67, 566, 258, 87, 319, 193, 524, 946, 713, 3, 215, 942, 578, 225, 403, 71, 2
11, 251, 80, 698, 458, 941, 85, 642, 242, 617, 562, 166, 66, 527, 796, 479,
640, 338, 276, 72, 107, 945, 327, 382, 752, 79, 424, 303, 865, 915, 449, 85
5, 507, 457, 786, 77, 748, 859, 537, 192, 398, 280, 70, 92, 883, 62, 207, 19
5, 613, 757, 732, 930, 735, 803, 647, 105, 840, 126, 598, 418, 651, 149, 92
1, 510, 746, 174, 669, 906, 483, 894, 309, 394, 827, 98, 492, 475, 898, 615,
969, 708, 884, 852, 287, 237, 304, 776, 548, 493, 325, 572, 811, 817, 35, 8,
693, 577, 905, 51, 353, 300, 553, 875, 401, 878, 550, 274, 879, 687, 622, 94
7, 461, 15, 329, 208, 664, 560, 709, 916, 97, 582, 984, 173, 576, 266, 240,
450, 63, 209, 979, 743, 792, 638, 161, 412, 799, 272, 93, 779, 911, 791, 92
5]
[503, 485, 830, 62, 307, 800, 502, 427, 261, 471, 323, 734, 921, 276, 91, 21
8, 888, 227, 678, 423, 741, 383, 298, 278, 549, 368, 488, 438, 98, 47, 233,
697, 998, 390, 392, 812, 132, 499, 635, 643, 768, 886, 197, 773, 166, 748, 7
22, 649, 917, 764, 955, 858, 717, 26, 67, 663, 844, 910, 478, 448, 508, 599,
521, 231, 382, 135, 452, 222, 270, 493, 517, 838, 763, 511, 103, 257, 88, 20
2, 758, 811, 424, 505, 211, 95, 295, 751, 584, 157, 616, 250, 407, 918, 214,
589, 828, 630, 17, 729, 675, 31, 946, 238, 825, 644, 702, 514, 879, 247, 50
4, 991, 671, 762, 815, 360, 338, 4, 892, 662, 386, 989, 449, 724, 174, 980,
561, 251, 993, 332, 665, 453, 277, 464, 796, 732, 183, 655, 426, 182, 365, 7
4, 753, 823, 846, 672, 661, 894, 265, 632, 591, 331, 126, 793, 136, 928, 35
9, 981, 791, 658, 109, 824, 937, 840, 554, 693, 224, 401, 940, 538, 667, 80
2, 496, 585, 393, 771, 19, 472, 526, 333, 719, 952, 27, 451, 530, 268, 113,
83, 29, 480, 57, 850, 745, 73, 608, 509, 306, 852, 402, 739, 788, 956, 376,
787, 164, 482, 920, 890, 168, 560, 775, 700, 28, 151, 450, 500, 400, 834, 12
1, 142, 573, 144, 578, 18, 228, 99, 601, 760, 148, 209, 992, 133, 219, 115,
939, 568, 772, 718, 799, 582, 794, 767, 637, 607, 507, 866, 207, 12, 316, 28
7, 974, 433]
```

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

In [78]:
```
j = 1
vector1 = Vector{Int}()
for i = 2:1:n+1
    if j != n
        append!(vector1, y[i] - x[j])
        j += 1
    end
end
print(vector1)
```

```
[439, 379, -251, -319, 629, 460, -76, -171, -150, 0, 555, 488, -120, -33, -5
2, 425, 171, 337, 411, 518, 178, -48, -509, -119, 213, 233, -367, -105, -74
7, -302, -72, 981, 317, 216, 171, -859, 134, 315, -331, 149, 487, -55, 373,
109, -251, 349, 88, 626, -196, 285, 333, 338, -413, -313, -272, 388, 502, 29
4, 96, -420, -268, 250, -372, 24, -492, -314, 202, -393, 246, -13, 74, 646,
421, -153, -217, -683, -338, -204, 580, -130, -123, -418, -474, 154, 2, -26
1, -138, 514, -360, -124, 47, -574, 211, -139, 64, -241, 642, 356, -162, 42
2, -708, 112, 641, 487, -428, 301, 22, 101, 920, 460, 511, 735, -338, -120,
-937, 807, 20, 144, 372, -113, 558, 108, 453, -235, -228, 353, -6, 389, 381,
170, -481, 469, 350, -569, 576, 2, -121, -500, -841, 304, -32, 339, 215, -12
5, 817, -483, -227, 54, 139, -272, 513, 66, 836, -524, 919, 584, 463, -504,
67, 205, -90, -181, -110, -423, 296, 100, 412, 69, 384, -155, 436, -528, 26
1, -727, 298, -143, -573, 236, 58, -282, 57, -297, 170, -379, -392, -869, -1
35, -912, 142, -139, -779, 321, 272, 2, 76, -146, 246, 463, 384, -435, -30,
129, 474, 227, 313, -737, 509, 422, 400, -525, -724, 49, -378, -150, 560, -7
58, -545, -49, -803, 117, 3, -101, -109, -63, 200, -561, -707, 895, -449, -7
65, -58, 363, 302, 532, 268, 736, 373, -185, 24, -155, -31, 346, 454, -592,
-260, 223, -492, 63, -358]
```

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

In [79]:
```
vector2 = Vector{Int}()
for i in 2:1:n-1
    append!(vector2, x[i-1] + 2*x[i] - x[i+1])
end
print(vector2)
```

```
[635, 451, 1394, 926, -248, 616, 746, 1351, 1088, 248, 649, 1101, 374, 201,
1140, 234, 726, 142, 253, 287, 110, 1252, 1968, 723, -140, 1662, 417, 1256,
1095, 2056, 730, -13, -216, 467, 2258, 1401, 31, 1649, 1813, 1165, 503, 995,
-485, 1682, 1184, 1204, 183, 1541, 1775, 1341, 844, 877, 264, 1794, 1439, 10
88, 424, -40, 1341, 2391, 806, 1119, 692, 846, 2139, 143, 1099, 627, 543, 19
41, 908, 41, 128, 433, 1476, 889, 2233, 870, 711, 1181, 1317, 1626, 102, 79
4, 2144, 1333, -111, 791, 801, 1485, 2069, 577, 1746, 1841, 995, 113, 532, 1
81, 295, 1703, 2369, 504, -509, 1521, 1873, 625, 960, 334, 242, 633, -287, 1
018, 673, 2255, 469, 1127, 509, 914, 1575, 828, -229, 324, 1640, 1114, 1421,
1040, 818, 313, -659, 1670, 1217, 339, 1807, 486, 624, 165, 1118, 2246, 958,
1652, 1412, 635, 1952, 192, 714, 1929, 1741, 523, 708, 888, 328, -629, 1796,
800, 281, -16, 664, 1395, 1291, 1857, 1597, 1694, 1992, 17, 1659, 494, 904,
783, 1571, 28, 1481, 1195, 1828, 425, 606, 1998, 978, 1962, 1118, 270, 1950,
531, 607, 544, 1656, 1159, 1845, 1501, 1624, 2301, 1189, 457, 69, 1308, 137
9, 1209, 571, 658, 1377, 2410, 879, -642, 817, 942, 2336, 654, 457, 400, 53
1, 1902, 799, 1607, 1704, 219, 1345, 1631, 984, 2055, 1854, 162, 465, 81, 97
6, 1075, 1062, 2444, 528, 277, 2377, 754, 1059, 868, 296, 1077, 367, -498, 1
424, 1673, 1689, 1907, 548, 186, 1738, 1250, -321, 740, 1810, 1568]
```

– сформируйте вектор $(\frac{\sin(y1)}{\cos(x2)}, \frac{\sin(y2)}{\cos(x3)}, \dots, \frac{\sin(y_{n-1})}{\cos(x_n)})$;

In [80]:
```
vector3 = Vector{Float32}()
j=1
for i = 2:1:n+1
    if j != n
        append!(vector3, sin(y[j])/cos(x[i]))
        j += 1
    end
```

```
end
print(vector3)
```

```
Float32[1.8749949, 2.3252172, -0.85393816, -3.4363673, 1.920659, 0.95000464,
-19.64254, -0.49741244, 0.29401755, 0.23722611, 0.642795, -0.9171692, 5.2933
874, -0.45113146, -0.28227082, -1.1046675, -6.416357, 0.8544827, 0.552154, -
1.2841488, -0.44339216, 8.7280245, -1.0902334, -2.051725, -0.81452155, -0.57
55044, 2.420071, 1.4239347, -0.9581341, -0.16025177, -1.8125209, 0.57065535,
-0.85762244, 0.43106204, -3.7494705, 1.1856586, 0.05873994, 0.4937813, -0.39
003658, -0.8556233, 1.269323, -0.13480462, 0.8843596, 0.16747378, -0.7318836
5, -1.3245153, 1.3690804, 4.007944, 0.5072001, 0.5957872, 0.10408222, -0.497
59918, -0.6628607, 2.0756924, 0.9591125, -0.1346332, -4.1178703, -0.8820993,
-1.3767064, 0.9512826, -1.1858546, 0.8796595, -0.48814824, -3.9891434, -1.12
08501, 0.21654636, 0.3825145, -2.3138855, 0.29414117, -0.2749662, -1.351260
8, -1.6091015, -9.948787, -0.9496898, -2.4142785, -0.6112448, 0.045256272,
8.689271, -1.6291236, 0.47602525, 0.1477566, -0.76725876, 0.5270324, 2.56301
8, 0.3059773, -0.1666091, -3.247838, -0.092252836, -0.24592727, 1.3636136,
1.1501323, 1.1432191, 0.4432829, -1.1465937, -1.0614014, 1.7444167, -7.51129
44, -0.7262525, -0.53615594, 0.43518013, 0.3752401, 0.6966478, 4.7800407, 0.
029831726, -0.99076074, -2.5577455, -0.9403715, -2.9991555, -1.118837, -1.04
04319, 8.730894, 1.1692256, -1.2407162, 10.325772, 0.97664773, -1.7196583,
0.21172492, 2.4205575, -0.4293199, -0.6465555, -0.24530925, 1.4039886, 2.440
9738, -1.8975075, 1.5388186, -1.168615, 0.28338757, 0.87485534, -0.86578095,
-0.7042287, 0.53397626, -2.7953992, 2.3091078, 0.0099463, -0.7133932, 6.1349
42, 1.9524294, 0.30146632, -0.56118107, -1.1146266, 2.3167238, 0.9563411, -
0.95896256, 9.566124, 0.99865454, -4.365888, -0.9145513, 0.5498297, -0.66913
06, 0.98248047, 0.5210498, -1.5458347, 0.80861557, -1.3994576, 0.8042432, 0.
75157315, 0.6808176, 0.9942505, -0.816775, 0.7881072, 0.7274945, -2.9296677,
0.89441836, -3.9898102, 2.2100847, -0.9552865, -1.3513727, 0.71884346, -1.07
88107, 3.483957, 0.41687754, 1.264717, 2.3173444, 2.7543285, -0.15179762, 2.
0143108, -1.409399, 0.040991772, 0.95351803, 0.36484188, -1.3208598, 1.20053
27, -2.3971617, 1.0094227, -0.11046768, 1.3261366, -3.6798735, -1.480482, -
1.2436527, -1.2116919, 0.9730098, 3.5789137, 1.3404136, -0.061972372, -4.620
418, 0.6037491, 0.8010756, -0.6816506, 0.5761655, 0.8311162, 0.925801, -6.86
95397, -2.1616523, -1.9652227, 0.479357, -1.0795448, -2.3963633, -32.354206,
0.8294272, -8.192686, 0.6267991, -2.6926043, 0.70036125, 0.60228103, -1.0656
04, 1.8684976, 0.9994395, 3.1093583, -1.3729395, 0.64634645, 0.081959985, -
0.9468795, -2.2507985, -1.4290558, -1.5115625, -1.1719508, 0.3657075, -1.438
2092, 0.8713367, -0.888943, 1.7043048, -1.8530573, 1.0126593, -0.8049632, -
0.7497136, -11.779446, 2.2411168, 54.065968, 0.77201337, -0.45099574, -0.950
76084, 0.6923745, -1.8296124, 3.5331113, -1.0657669, -0.5401393, 0.9656349,
-1.1560619, 0.53648996]
```

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

In [81]:
```
sum_vector = 0
for i in 1:1:n-1
    sum_vector += exp(-x[i+1])/(x[i]+10)
end

print(sum_vector)
```

```
7.633471719244134e-5
```

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

In [82]:
```
# возврат индексов элементов массива, удовлетворяющих условию:
indexes_y = findall(y .> 600)
# элементы вектора y, значения которых больше 600
print(y[indexes_y])
```

```
[830, 800, 734, 921, 888, 678, 741, 697, 998, 812, 635, 643, 768, 886, 773,
748, 722, 649, 917, 764, 955, 858, 717, 663, 844, 910, 838, 763, 758, 811, 7
51, 616, 918, 828, 630, 729, 675, 946, 825, 644, 702, 879, 991, 671, 762, 81
5, 892, 662, 989, 724, 980, 993, 665, 796, 732, 655, 753, 823, 846, 672, 66
1, 894, 632, 793, 928, 981, 791, 658, 824, 937, 840, 693, 940, 667, 802, 77
1, 719, 952, 850, 745, 608, 852, 739, 788, 956, 787, 920, 890, 775, 700, 83
4, 601, 760, 992, 939, 772, 718, 799, 794, 767, 637, 607, 866, 974]
```

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

In [83]:
```
indices_y = findall(y .> 600)
print(x[indices_y])
```

```
[313, 42, 433, 396, 56, 12, 205, 17, 73, 991, 974, 619, 399, 252, 57, 373, 5
61, 291, 960, 670, 525, 379, 439, 456, 408, 184, 117, 90, 231, 554, 845, 61
0, 788, 566, 258, 319, 193, 946, 3, 215, 942, 225, 211, 251, 80, 698, 642, 2
42, 562, 66, 796, 338, 72, 382, 752, 424, 855, 507, 457, 786, 77, 748, 537,
70, 883, 207, 195, 613, 732, 930, 735, 647, 126, 418, 651, 746, 894, 309, 88
4, 852, 237, 548, 325, 572, 811, 35, 577, 905, 300, 553, 879, 560, 709, 582,
266, 450, 63, 209, 743, 792, 638, 161, 799, 791]
```

– сформируйте вектор $(\mid x_1 - \overline{x} \mid^{\frac{1}{2}}, \mid x_2 - \overline{x} \mid^{\frac{1}{2}}, \mid x_n - \overline{x} \mid^{\frac{1}{2}})$, где $\overline{x}$ обозначает среднее значение вектора x = (x1, x2, ... , xn);

In [84]:
```
vector4 = Vector{Float16}()
for i in 1:1:n
    append!(vector4, (abs(x[i] - mean(x)))^(0.5))
end
print(vector4)
```

```
Float16[21.02, 6.055, 13.22, 11.766, 17.8, 21.11, 3.918, 7.46, 11.55, 12.83,
17.56, 7.39, 9.57, 19.06, 14.75, 4.965, 20.78, 12.11, 21.81, 16.27, 16.81, 1
1.9, 17.3, 13.43, 18.23, 15.25, 17.81, 16.88, 17.5, 6.883, 16.78, 21.69, 20.
36, 17.66, 12.38, 22.44, 11.08, 12.945, 22.05, 11.46, 9.414, 15.35, 9.36, 2
0.75, 22.61, 10.71, 8.56, 14.02, 21.73, 13.51, 6.113, 10.42, 6.973, 10.375,
21.16, 5.625, 8.92, 17.42, 11.65, 20.98, 19.48, 14.72, 10.74, 11.38, 11.805,
16.69, 21.62, 13.24, 15.516, 6.508, 16.62, 19.25, 19.94, 15.22, 3.693, 16.8
3, 7.234, 21.78, 16.02, 8.15, 11.84, 11.89, 9.02, 18.62, 16.17, 18.9, 13.88,
19.64, 11.06, 6.586, 19.58, 17.33, 10.47, 21.89, 8.85, 15.16, 20.02, 12.984,
17.17, 6.03, 21.4, 15.016, 22.02, 16.52, 21.31, 9.51, 16.2, 9.2, 20.4, 16.6
2, 15.38, 20.19, 14.5, 5.445, 21.3, 20.06, 12.42, 15.67, 11.375, 8.625, 17.9
4, 20.53, 6.273, 17.56, 2.94, 12.34, 12.234, 14.55, 20.39, 19.52, 21.39, 12.
67, 10.28, 16.27, 20.22, 7.977, 13.586, 19.42, 20.67, 6.215, 19.17, 4.4, 5.5
35, 17.27, 20.27, 16.14, 19.27, 7.023, 17.19, 9.47, 14.41, 20.44, 19.89, 19.
89, 20.62, 16.75, 17.11, 11.195, 16.4, 15.63, 21.03, 15.73, 17.77, 12.625, 1
9.56, 18.77, 19.02, 10.51, 8.34, 12.78, 18.4, 20.81, 4.727, 16.08, 17.7, 13.
47, 20.45, 2.154, 20.16, 13.37, 9.68, 18.42, 19.73, 2.088, 3.557, 20.25, 11.
29, 21.94, 14.84, 19.9, 19.1, 14.164, 15.83, 13.555, 16.98, 7.77, 2.314, 12.
75, 9.19, 17.98, 18.14, 21.28, 21.9, 14.33, 9.45, 20.42, 20.89, 11.6, 13.69
5, 8.086, 19.69, 9.305, 19.75, 7.9, 14.62, 19.78, 14.12, 11.59, 21.44, 5.16,
21.73, 12.59, 16.72, 13.28, 8.51, 14.875, 20.7, 19.77, 9.71, 22.28, 17.73,
9.4, 14.89, 15.734, 6.137, 20.61, 16.69, 22.17, 15.98, 17.45, 12.266, 18.08,
8.695, 17.64, 14.69, 19.86, 17.06, 20.58, 17.42, 20.9]
```

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

In [85]:
```
count_numbers = 0
max = maximum(y)
for i in 1:1:n
    if abs(y[i] - max) < 200
        count_numbers += 1
    end
end
print(count_numbers)
```

```
47
```

– определите, сколько чётных и нечётных элементов вектора x;

In [86]:
```
even_numbers = 0
odd_numbers = 0
for i in 1:1:n
    if mod(x[i], 2) == 0
        even_numbers += 1
    else
        odd_numbers += 1
    end
end
println("Количество четных чисел = ", even_numbers)
print("Количество нечетных чисел = ", odd_numbers)
```

```
Количество четных чисел = 113
Количество нечетных чисел = 137
```

– определите, сколько элементов вектора x кратны 7;

In [87]:
```
count_seven = 0
for i in 1:1:n
    if mod(x[i], 7) == 0
        count_seven += 1
    end
end
print("Количество элементов кратных 7 = ", count_seven)
```

```
Количество элементов кратных 7 = 32
```

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

In [88]:
```
x_new = x[sortperm(y)]
print(x_new)
```

```
[85, 93, 87, 329, 174, 380, 394, 875, 615, 524, 535, 708, 626, 935, 287, 44
9, 898, 540, 270, 141, 794, 664, 474, 757, 475, 576, 687, 280, 365, 984, 76
6, 92, 622, 461, 916, 401, 102, 8, 999, 51, 527, 865, 79, 400, 962, 272, 97,
569, 378, 463, 173, 663, 105, 341, 208, 358, 769, 713, 403, 531, 640, 771, 6
21, 859, 492, 247, 124, 945, 668, 911, 749, 787, 776, 171, 779, 179, 398, 27
6, 483, 941, 62, 458, 915, 255, 817, 627, 346, 617, 176, 641, 510, 274, 840,
493, 871, 223, 628, 303, 432, 925, 203, 928, 166, 878, 827, 20, 107, 327, 32
3, 669, 352, 969, 693, 451, 805, 530, 149, 320, 550, 503, 46, 71, 629, 412,
867, 304, 256, 578, 764, 603, 906, 98, 598, 155, 803, 353, 479, 240, 947, 1
5, 979, 295, 921, 967, 192, 271, 560, 161, 237, 610, 258, 537, 974, 638, 61
9, 215, 291, 424, 613, 77, 242, 456, 72, 418, 251, 786, 193, 12, 647, 17, 55
3, 942, 439, 63, 894, 561, 66, 319, 752, 433, 325, 205, 852, 373, 845, 855,
231, 709, 80, 90, 670, 792, 399, 746, 450, 57, 300, 35, 572, 195, 70, 743, 3
82, 209, 42, 651, 554, 991, 698, 507, 732, 3, 566, 313, 879, 117, 735, 408,
457, 884, 548, 379, 799, 225, 252, 56, 905, 642, 748, 184, 960, 788, 577, 39
6, 883, 930, 266, 126, 946, 309, 525, 811, 791, 796, 207, 562, 211, 582, 33
8, 73]
```

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

In [89]:
```
x_sort = sort(x, rev=true)
print(x_sort[1:10])
```

```
[999, 991, 984, 979, 974, 969, 967, 962, 960, 947]
```

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

In [90]:
```
x_unique = unique(x)
print(x_unique)
length(x_unique)
```

```
[46, 451, 313, 626, 171, 42, 503, 432, 621, 323, 179, 433, 396, 124, 270, 46
3, 56, 341, 12, 223, 205, 346, 787, 668, 155, 255, 805, 203, 794, 535, 769,
17, 73, 176, 641, 991, 365, 320, 974, 619, 399, 252, 400, 57, 999, 373, 561,
291, 960, 670, 525, 379, 439, 380, 935, 456, 408, 184, 352, 928, 867, 271, 6
03, 358, 627, 766, 20, 663, 247, 530, 764, 117, 90, 256, 474, 771, 540, 962,
231, 554, 628, 629, 569, 141, 749, 845, 295, 102, 610, 531, 871, 788, 378, 9
67, 566, 258, 87, 319, 193, 524, 946, 713, 3, 215, 942, 578, 225, 403, 71, 2
11, 251, 80, 698, 458, 941, 85, 642, 242, 617, 562, 166, 66, 527, 796, 479,
640, 338, 276, 72, 107, 945, 327, 382, 752, 79, 424, 303, 865, 915, 449, 85
5, 507, 457, 786, 77, 748, 859, 537, 192, 398, 280, 70, 92, 883, 62, 207, 19
5, 613, 757, 732, 930, 735, 803, 647, 105, 840, 126, 598, 418, 651, 149, 92
1, 510, 746, 174, 669, 906, 483, 894, 309, 394, 827, 98, 492, 475, 898, 615,
969, 708, 884, 852, 287, 237, 304, 776, 548, 493, 325, 572, 811, 817, 35, 8,
693, 577, 905, 51, 353, 300, 553, 875, 401, 878, 550, 274, 879, 687, 622, 94
7, 461, 15, 329, 208, 664, 560, 709, 916, 97, 582, 984, 173, 576, 266, 240,
450, 63, 209, 979, 743, 792, 638, 161, 412, 799, 272, 93, 779, 911, 791, 92
5]
```

Out[90]: 250

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

In [91]:
```
# зададим в цикле получение квадратов всех целых чисел от 1 до 100
squares = [i^2 for i=1:1:100]
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 32
4, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024, 10
89, 1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025,
2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249, 336
4, 3481, 3600, 3721, 3844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5
041, 5184, 5329, 5476, 5625, 5776, 5929, 6084, 6241, 6400, 6561, 6724, 6889,
7056, 7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836, 9025, 921
6, 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.

In [92]:
```
# использую пакет Primes и получим первые 168 простых чисел
using Primes
n = 1000
myprimes = primes(n)
print(myprimes)
```

```
[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, 1
57, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 23
9, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 33
1, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 42
1, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 50
9, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 61
3, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 70
9, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 82
1, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 91
9, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]
```

In [93]:
```
# определим наименьшее 89-е число
println("89-е наименьшее простое число = ", myprimes[89])
print("Срез массива с 89-го до 99-го элемента = ", myprimes[89:99])
```

```
89-е наименьшее простое число = 461
Срез массива с 89-го до 99-го элемента = [461, 463, 467, 479, 487, 491, 499,
503, 509, 521, 523]
```

6. Вычислите следующие выражения:

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

In [94]:
```
sum((i^3)+4*(i^2) for i=10:1:100)
```

Out[94]: 26852735

6.2 $\sum_{i=1}^{M}(\frac{2^i}{i}+\frac{3^i}{i^2})$, M = 25;

In [95]:
```
M = 25
sum(((2^i)/i) + ((3^i)/(i^2)) for i=1:1:M)
```

Out[95]: 2.1291704368143802e9

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

In [96]:
```
_sum = 1
temp = 1
for i in 3:2:39
    temp *= (i-1)/i
    _sum += temp
end
print(_sum)
```

```
6.976346137897618
```