Лабораторная работа № 2

Измерение и тестирование пропускной способности сети. Интерактивный эксперимент

Старовойтов Егор Сергеевич

Содержание

Цель работы

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

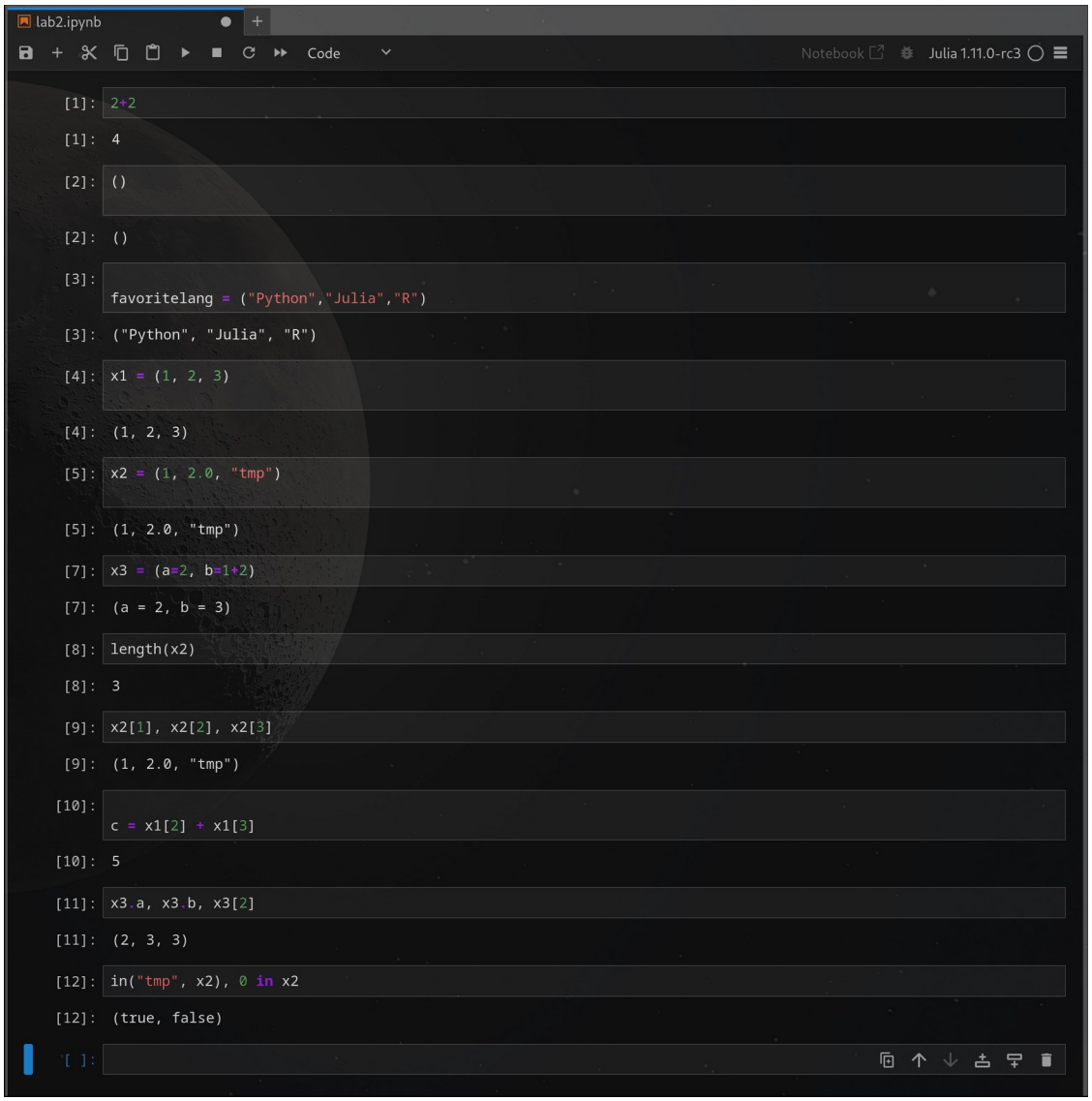
Задание

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

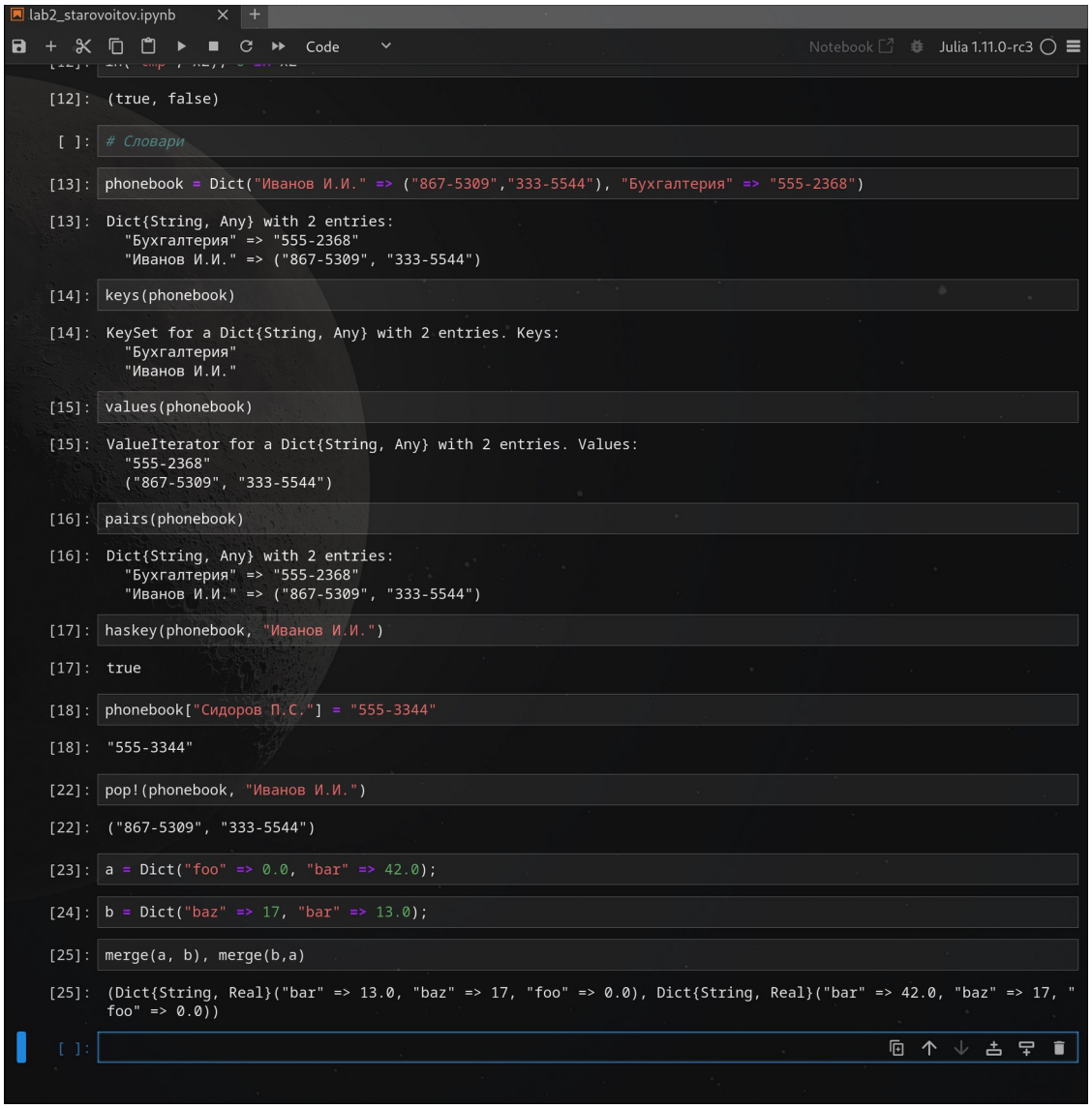
Выполнение лабораторной работы

Задание 1

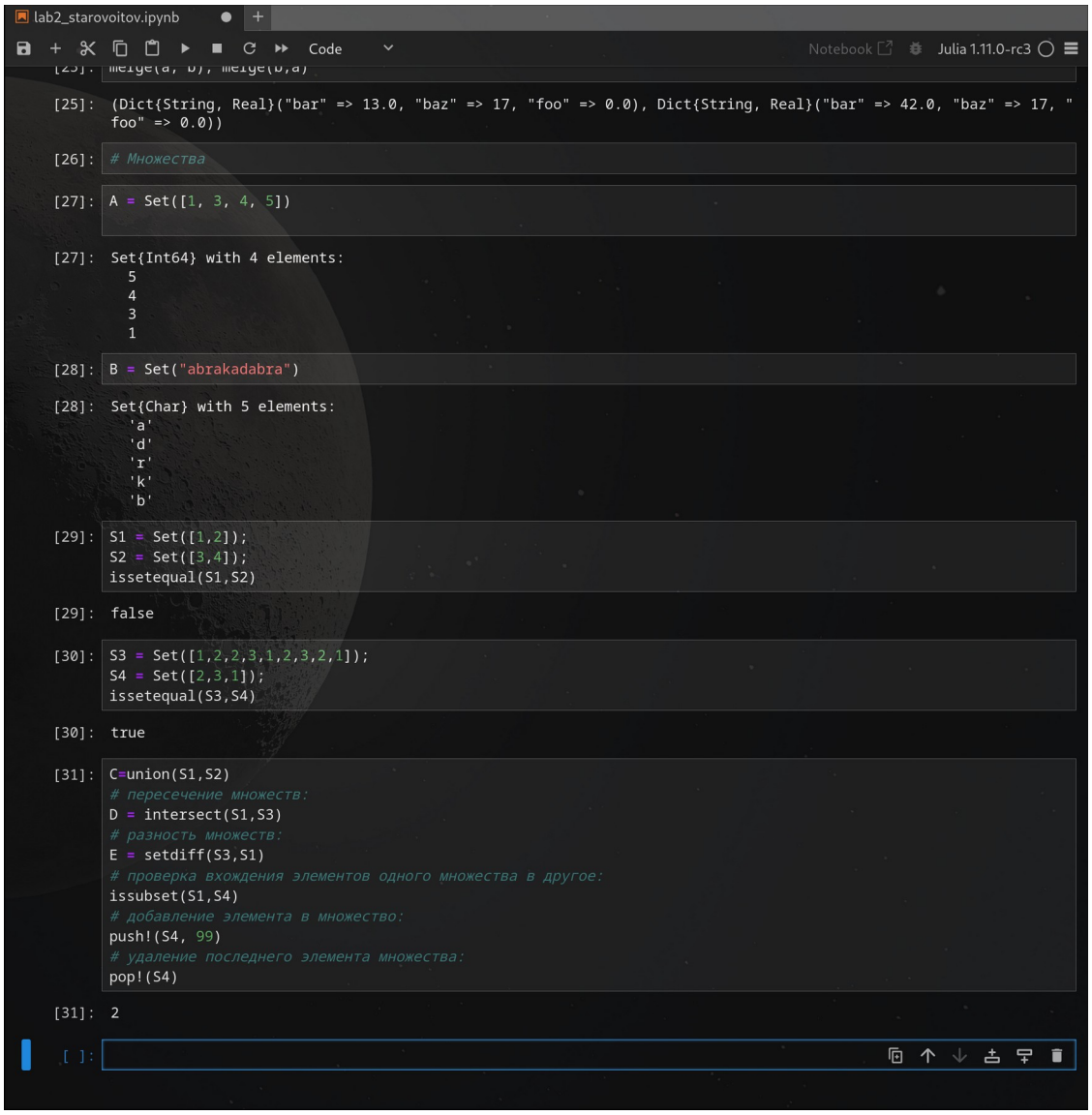
Были повторены все примеры.


lab2.ipynb
Code
Notebook
Julia 1.11.0-rc3
[1]: 2+2
[1]: 4
[2]: ()
[2]: ()
[3]: favoritelang = ("Python","Julia","R")
[3]: ("Python", "Julia", "R")
[4]: x1 = (1, 2, 3)
[4]: (1, 2, 3)
[5]: x2 = (1, 2.0, "tmp")
[5]: (1, 2.0, "tmp")
[7]: x3 = (a=2, b=1+2)
[7]: (a = 2, b = 3)
[8]: length(x2)
[8]: 3
[9]: x2[1], x2[2], x2[3]
[9]: (1, 2.0, "tmp")
[10]: c = x1[2] + x1[3]
[10]: 5
[11]: x3.a, x3.b, x3[2]
[11]: (2, 3, 3)
[12]: in("tmp", x2), 0 in x2
[12]: (true, false)
[ ]:

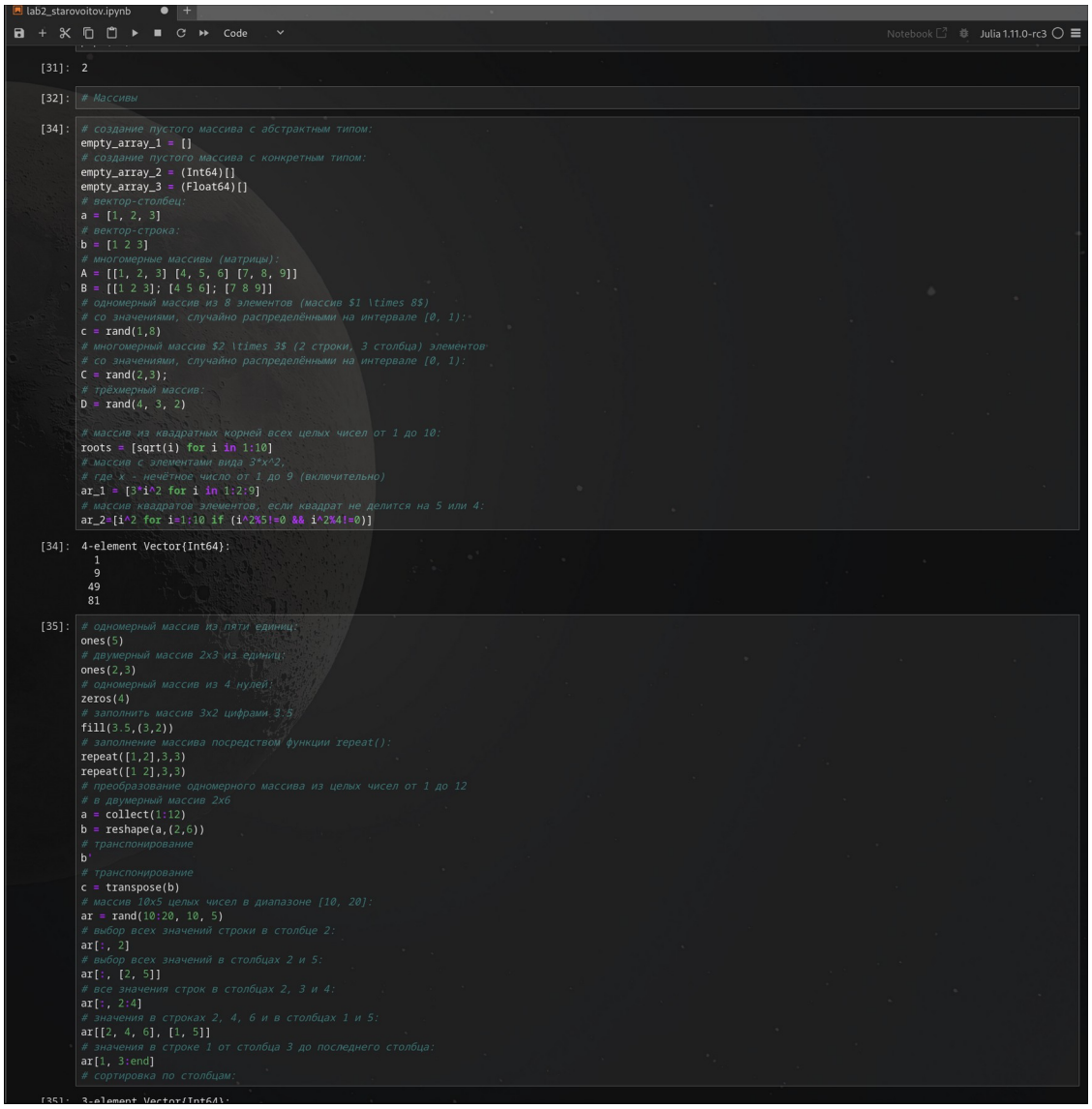
Кортежи


lab2_starovoitov.ipynb
Code
Notebook
Julia 1.11.0-rc3
[12]: (true, false)
[ ]: # Словари
[13]: phonebook = Dict("Иванов И.И." => ("867-5309","333-5544"), "Бухгалтерия" => "555-2368")
[13]: Dict{String, Any} with 2 entries:
"Бухгалтерия" => "555-2368"
"Иванов И.И." => ("867-5309", "333-5544")
[14]: keys(phonebook)
[14]: KeySet for a Dict{String, Any} with 2 entries. Keys:
"Бухгалтерия"
"Иванов И.И."
[15]: values(phonebook)
[15]: ValueIterator for a Dict{String, Any} with 2 entries. Values:
"555-2368"
("867-5309", "333-5544")
[16]: pairs(phonebook)
[16]: Dict{String, Any} with 2 entries:
"Бухгалтерия" => "555-2368"
"Иванов И.И." => ("867-5309", "333-5544")
[17]: haskey(phonebook, "Иванов И.И.")
[17]: true
[18]: phonebook["Сидоров П.С."] = "555-3344"
[18]: "555-3344"
[22]: pop!(phonebook, "Иванов И.И.")
[22]: ("867-5309", "333-5544")
[23]: a = Dict("foo" => 0.0, "bar" => 42.0);
[24]: b = Dict("baz" => 17, "bar" => 13.0);
[25]: merge(a, b), merge(b,a)
[25]: (Dict{String, Real}("bar" => 13.0, "baz" => 17, "foo" => 0.0), Dict{String, Real}("bar" => 42.0, "baz" => 17, "foo" => 0.0))
[ ]:

Словари


lab2_starovoitov.ipynb
Code
Notebook
Julia 1.11.0-rc3
[25]: merge(a, b), merge(b,a)
[25]: (Dict{String, Real}("bar" => 13.0, "baz" => 17, "foo" => 0.0), Dict{String, Real}("bar" => 42.0, "baz" => 17, "foo" => 0.0))
[26]: # Множества
[27]: A = Set([1, 3, 4, 5])
[27]: Set{Int64} with 4 elements:
5
4
3
1
[28]: B = Set("abrakadabra")
[28]: Set{Char} with 5 elements:
'a'
'd'
'r'
'k'
'b'
[29]: S1 = Set([1,2]);
S2 = Set([3,4]);
issetequal(S1,S2)
[29]: false
[30]: S3 = Set([1,2,2,3,1,2,3,2,1]);
S4 = Set([2,3,1]);
issetequal(S3,S4)
[30]: true
[31]: C=union(S1,S2)
# пересечение множеств:
D = intersect(S1,S3)
# разность множеств:
E = setdiff(S3,S1)
# проверка вхождения элементов одного множества в другое:
issubset(S1,S4)
# добавление элемента в множество:
push!(S4, 99)
удаление последнего элемента множества:
pop!(S4)
[31]: 2
[ ]:

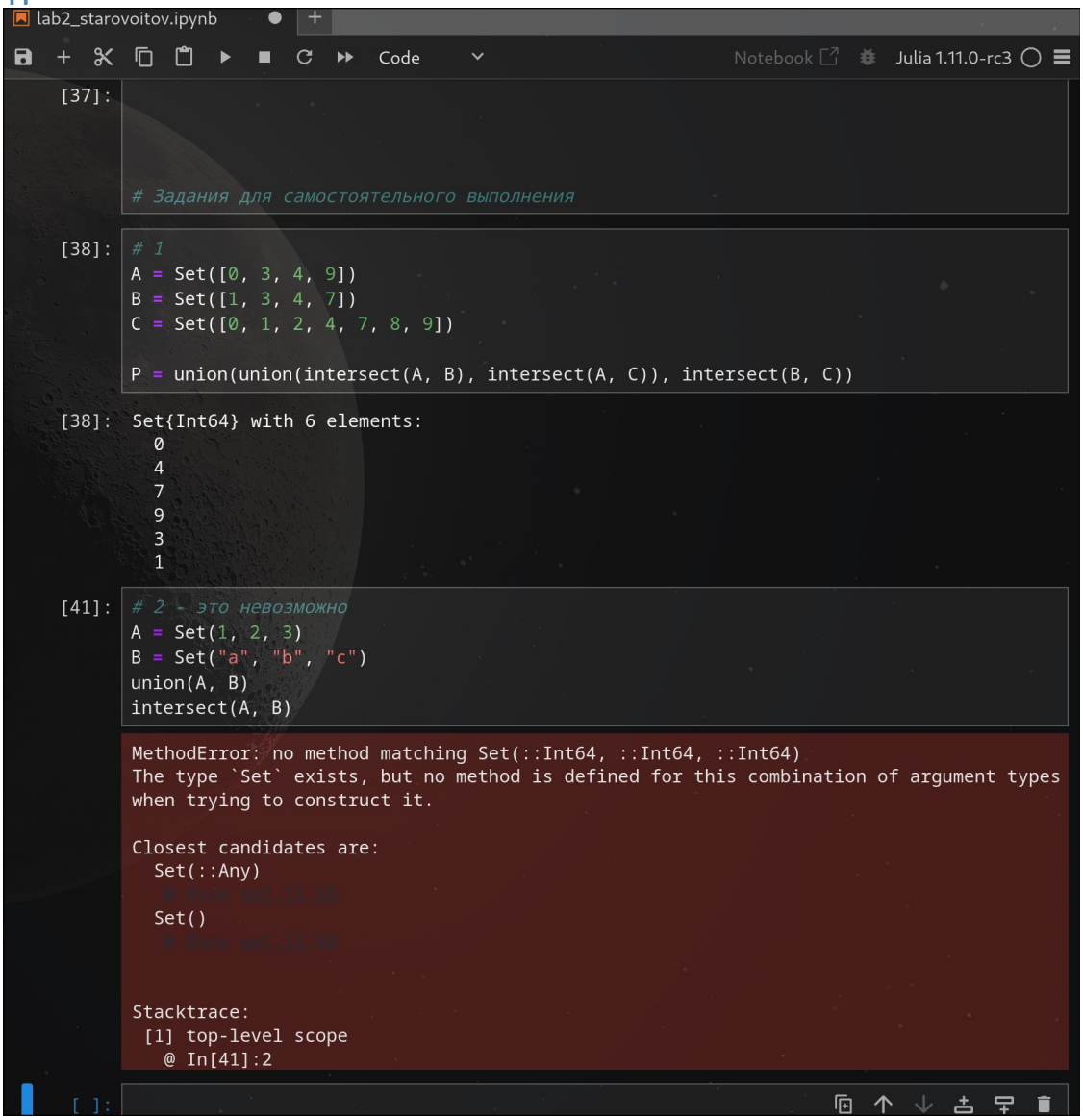
Множества


lab2_starovoitov.ipynb
Code
Notebook
Julia 1.11.0-rc3
[31]: 2
[32]: # Массивы
[34]: # создание пустого массива с абстрактным типом:
empty_array_1 = []
# создание пустого массива с конкретным типом:
empty_array_2 = (Int64)[]
empty_array_3 = (Float64)[]
# вектор-столбец:
a = [1, 2, 3]
# вектор-строка:
b = [1 2 3]
# многомерные массивы (матрицы):
A = [[1, 2, 3] [4, 5, 6] [7, 8, 9]]
B = [[1 2 3]; [4 5 6]; [7 8 9]]
# одномерный массив из 8 элементов (массив 1×8)
# со значениями, случайно распределёнными на интервале [0, 1):
c = rand(1,8)
# многомерный массив 2×3 (2 строки, 3 столбца) элементов
# со значениями, случайно распределёнными на интервале [0, 1):
C = rand(2,3);
# трёхмерный массив:
D = rand(4, 3, 2)
# массив из квадратных корней всех целых чисел от 1 до 10:
roots = [sqrt(i) for i in 1:10]
# массив с элементами вида 3*x^2,
# где x - нечётное число от 1 до 9 (включительно)
ar_1 = [3*i^2 for i in 1:2:9]
# массив квадратов элементов, если квадрат не делится на 5 или 4:
ar_2=[i^2 for i=1:10 if (i^2%5!=0 && i^2%4!=0)]
[34]: 4-element Vector{Int64}:
1
9
49
81
[35]: # одномерный массив из пяти единиц:
ones(5)
# двумерный массив 2x3 из единиц:
ones(2,3)
# одномерный массив из 4 нулей:
zeros(4)
# заполнить массив 3x2 цифрами 3.5
fill(3.5,(3,2))
# заполнение массива посредством функции repeat():
repeat([1,2],3,3)
repeat([1 2],3,3)
# преобразование одномерного массива из целых чисел от 1 до 12
# в двумерный массив 2x6
a = collect(1:12)
b = reshape(a,(2,6))
# транспонирование
b'
# транспонирование
c = transpose(b)
# массив 10x5 целых чисел в диапазоне [10, 20]:
ar = rand(10:20, 10, 5)
# выбор всех значений строки в столбце 2:
ar[:, 2]
выбор всех значений в столбцах 2 и 5:
ar[:, [2, 5]]
все значения строк в столбцах 2, 3 и 4:
ar[:, 2:4]
значения в строках 2, 4, 6 и в столбцах 1 и 5:
ar[[2, 4, 6], [1, 5]]
значения в строке 1 от столбца 3 до последнего столбца:
ar[1, 3:end]
сортировка по столбцам:
[35]: 3-element Vector{Int64}:

Массивы

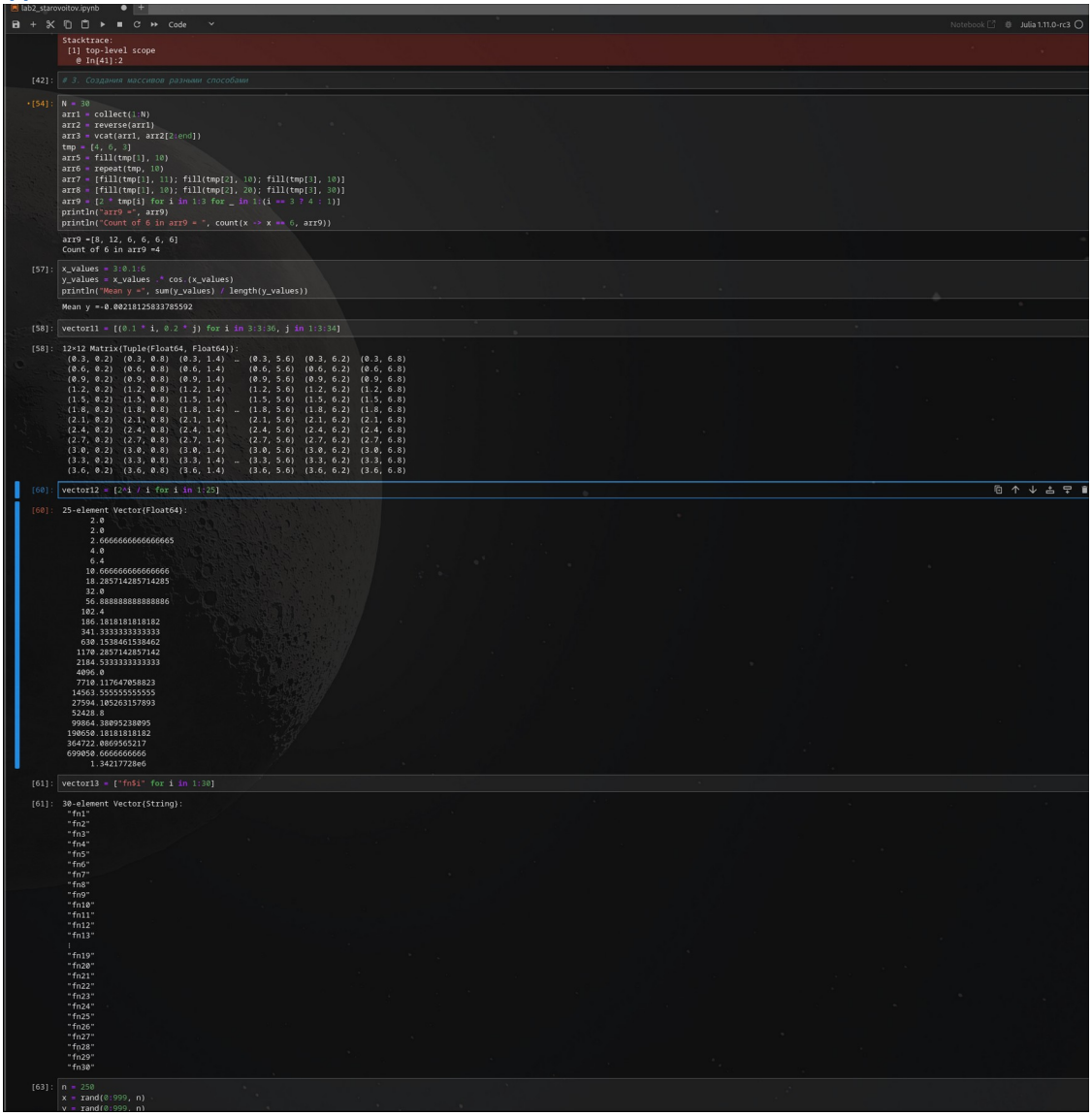
Задание 2 - самостоятельная работа

Задачи 1 и 2



Задачи 1 и 2

Задача 3



Часть 1


Code
Notebook
Julia 1.11.0-rc3
"tn30"
[63]:
n = 250
x = rand(0:999, n)
y = rand(0:999, n)
println([y[i + 1] - x[i] for i in 1:(n - 1)])
println()
println([x[i] + 2 * x[i + 1] - x[i + 2] for i in 1:(n - 2)])
println()
println([sin(y[i]) / cos(x[i + 1]) for i in 1:(n - 1)])
println()
println(sum(exp(-x[i + 1]) / (x[i] + 10) for i in 1:(n - 1)))
[1, 824, -206, 162, -152, 72, -371, 505, -330, 458, 277, -265, -257, 191, -713, -525, 120, 233, 905, -122, -32, 409, 189, -78, 896, -347, -85, -435, -389, -316, -231, -602, 17, -241, -312, -78, 311, 632, 144, -429, 703,
0, -319, -192, -265, 745, 50, 240, 183, -366, 337, 755, -41, 755, -890, -201, -125, -285, -273, -179, -396, -134, 240, 12, 484, -375, -317, 387, -13, -588, -549, -605, 766, -186, 168, 512, 204, 406, 477, -305, 245, 494,
37, 529, 10, 80, -371, 76, 247, 39, 363, -214, 197, -705, -489, -687, -264, 15, 406, 142, -514, -83, -47, 162, -152, -550, -43, 551, 437, -130, -192, -37, -49, -15, -367, 145, 40, 615, 544, 151, -691, 96, -121, 129, -224
-334, -200, -148, -687, 445, 628, 383, 849, 115, 883, 205, -448, -217, -36, -488, 166, 98, 666, 394, -273, -680, -736, 412, -195, -116, -461, -304, 517, -471, 88, -192, 144, -400, 898, 116, -690, -129, -289, 640, -387,
0, -341, 561, -307, -275, -705, -31, -177, -859, -164, -99, -145, 182, 586, -202, 135, 688, -348, -20, -168, -177, -231, -585, -393, -81, 6, 128, -52, 149, 38, -679, 309, -402, -216, 127, -226, 753, -58, -123, 546, -251,
-264, 565, 139, -558, -214, 561, -741, 76, -498, -133, 179, -489, -104, 807, 291, -215, 496, -169, 175, -56, 26, -230, 523, -200, 701, -219, 745, 335, -48, 14, 120, -366, 879, -311, -436, -429, -282, -318, 25, 468, -381,
6, -47]
[957, 323, 509, 712, 1095, 2213, 449, 1383, 705, 478, 918, 1328, 365, 1195, 1563, 752, 721, 27, 528, 739, 133, 241, 917, -313, 1179, 1662, 964, 986, 1230, 1718, 2515, 1017, 607, 189, 2271, 767, 33, 1416, 1423, 385, 503,
90, 922, 1301, 497, 810, 121, 414, 1515, 1817, 778, 355, -562, 1392, 1965, 399, 1355, 2182, 1543, 1013, 879, 1127, 680, -439, 1210, 1431, 62, 445, 938, 1704, 2027, 2036, 2214, 1341, -57, 1242, 1101, -114, 665, 428, 295,
59, 237, -10, 1610, 1521, 762, 484, 1762, 1270, 2023, 816, 1603, 1232, 1982, 1173, 324, 713, -328, 1849, 606, 1345, 1928, 686, 957, 1123, 522, 775, 1291, 1216, -32, 1744, 445, 2101, 353, 1622, 1013, 479, 29, 1665, 64, 17
, 1131, 1007, 1354, 1443, 1410, 2184, 662, 350, 1197, 640, 375, -221, 331, 1010, 1506, 1633, 2054, 511, 650, 607, 203, 1829, 1397, 2095, 625, 791, 465, 1027, 2064, 746, 1869, 459, 888, 9, 793, -288, 868, 1208, 792, 2003, 2233
201, 664, 1092, 1607, 328, 833, 1135, 1490, 2169, 483, 1203, 2249, 1509, 738, -134, -36, 1744, 1470, 75, 2037, 842, -365, 1542, 1209, 1010, 1596, 1260, 1722, 2082, 987, 1485, -2, 1545, 1138, 1120, 433, 1337, 1012, 380,
4, 1652, 710, 1830, 1555, -324, 1092, 1769, 947, -153, 1291, 1859, 1983, 1675, 651, 642, 1815, 263, 812, 900, -37, 1250, 1764, 1023, 955, 580, 388, 2058, 484, 1098, -3, 666, 1441, 837, 862, 2452, 663, 83, 936, 1661, 1053
1088, 1214, -51, 1235, 2375, 654]
[-1.1965501521808246, 0.47993681683553563, 0.10608161896204343, -1.7295461467467896, -3.443451483656467, -0.5334097243735564, -4.143461622997187, -0.816725214391953, -1.7637041113292977, 0.2067137830650559, -0.5612188476
054, 4.592033297993962, -1.1055087649764395, -1.0248163611050265, -0.13770483885212384, -1.0496228702185453, -0.7374982333814005, 0.3274495664492821, -2.3946772029638829, -6.611425665692199, -1.0484443216663282, 1.79985366
7682315, 0.8254069961826832, -2.2300032793487077, 0.7406249539812506, 1.4371565755544764, -1.2421010981044949, 0.9566953962953, -1.0260798312334423, -0.8179092694004706, 1.6037168508124238, 0.9940752491944955, -2.00583466
690106, -1.0298707238164222, 0.44693798052117642, -3.6681676268619983, -1.1424797807948728, 1.8809863112950418, 3.1366107608284723, 0.9390487395010883, -0.0088735091769721 14, 1.4135752952379133, -1.0761598381858553, -2.752539
998833923, -0.7646551716908174, 0.7046318892142087, -0.1609277824516172, 0.2942759413411697, -0.8381854211472411, 2.1700284109144498, -11.959587428836797, -1.9719457401167597, -0.7461102143752637, 1.0219671821988607, -0
5114844085809865, -3.758056824625595, -0.0976148088616503 2, -1.1631764716736872, -0.8525978849598506, 1.513840294149254, -0.3492990434734832, -0.7271166295995658, 1.2352567412107724, 0.8686764757860022, 0.2353308690960566
-0.3698004730514054 5, 0.7696183569093369, 0.14374796930140363, -0.07044799038515497, 1.1976444262021377, -0.9955831492685733, -0.07639573836943443, 21.218973791457785, -2.4449249025646031, 0.8440950228721437, -0.8458186145
939, 0.3431972006352582, -1.8696322217977103, -0.2764708085655632, -0.8249158962869273, -141.73154093011178, 0.2547923538940568 7, -0.7978335226700753, 0.8782083122847 48, -1.0174828555982432, -2.5221176711042745, 1.0045114
8925013, 1.0425044869376736, 0.9133285053999225, 2.2314951746382863, 0.2624323171595442 4, 0.7245237925981212, -0.8183340471398600 8, -1.4528714660337765, 0.7309946440373015, 0.1910255343591777 67, 0.8580537570498287, -1.0461
088349338, 0.9765157442468614, 0.5328333260334598, -6.266367306572841, 1.2075335818910 71, -0.4179805437999386 5, -0.7455568966228852, 1.0639676362483885, -0.8476646596450207, -0.6662103662409736, 1.2288725302398902, -3.780
7228526865, -1.9299314159931007, -0.18553771819529433, 2.8628372307874 41, 0.7126969791560416, -0.8883637295793333, 1.0042342336873367, -0.5381662570800098, 1.0885733245351332, 0.4757682962870373, -0.4637425306616717, -0.
4676739248 1654, -0.7933268795784939, -14.379725353377195, -1.8310310788616236, 0.1491654969762844 8, 3.3233158621372 61, 3.1719485590919676, 0.7886797781979604, 1.0687439571292883, 0.6635508653814436, 0.5971616185693633,
-1.6268153395817884, 6.741744701381635, 10.347645315184534, 0.8599421405567781, 0.4275392438165121, 0.6400565055183126, 1.1833648789036 3, -2.7198278974286096, -0.8555916141995532, -0.2452319632984942, 0.2652852334599865,
1.16697185410 4187, -0.9005856502124 51, 1.5775615587185559, 1.0077784273762531, -1.4289964728442317, 0.9284659817700886, 1.5832179471729935, 2.14746252520736, -2.4502196888379197, -1.3560878416995355, -1.0399378362599918,
1.8634208934115 67, -14.529031838518636, -3.5910877655301 95, -1.8162690966404 38, 0.8302324082244891, 0.5696381711523013, 1.2435496467324 01, 2.2652313319197 65, 0.8015983238398113, -1.0118468175654913, -0.3385169034688515 5,
.9385332089011131, 1.0513474108943401, -1.2166838672507 485, 1.2126146553037955, 2.6266367029576187, 0.6200639426938807, 1.6693880979450777, -0.9563995310155062, 1.8790367229290679, -0.7733178378236564, 0.2314016643463871
0.8114859092257876, 1.2809186534566363, -2.2545862089712907, 0.5276494788728492, -0.8136635753180914, -28.165243413640106, 0.8272612609998985, 1.0688804837282 57, -12.257982645269921, 0.3847773438458905, 6.583946832664772
0.9674834000952934, 0.7998536558010491, 1.0880259189591686, 3.4027826017029 85, 1.7729431916908718, 29.862644150160833, -1.0326305591802 68, -0.8291190159754673, -4.596802699041375, -0.9756415841907077, -2.0687465627188777,
.5896377721854 89, 0.0090887434766 1188, 0.0786670755270416, 0.9214396572622071, 3.0674925584784836, 0.5442129699249518, -0.3573673736713845 5, -1.0006565236538454, 0.8911492456770379, 0.7157828466496273, 0.3063435697100644
-0.0354339092269108, -1.5104861354301002, 1.8556582085791157, 0.8572162866206093, 1.9886100529654698, -3.5203680735668 71, 0.5990072381106434, -12.005647755350742, -1.1008170661635994, -0.4455585615458416, -3.2341976606
669, -0.9403840442168782, 0.8881845243469593, -0.9668192880710349 3, -0.8983922218946964, 0.4070127208862 37, 0.4071058046520524, -0.8617868168129903, 23.254061519195456, -0.6739263885160835, 0.6338843356104689, -0.6637167
383341, 2.5902598258724647, 0.3274346070024774, -2.3127471013961 97, -2.2172731129880254, -0.6632025225652218, -1.0693467931864464, -0.9420012141073014, -0.8201057249428748, 0.4475973082067209, 0.7788825825813559, -0.8392
631629358 04, -0.5073937877710617, 0.1299839027038403, -0.7906916230957457, -0.17752386207451318, -0.3304277202910816, -1.0646557885814447, -0.7253641804698 38, -0.9747873783067 22, -1.6586526947369444]
1.9851068704085825e-12
println()
println(filter(x -> x > 600, y))
[689, 933, 974, 777, 667, 657, 680, 961, 986, 702, 898, 989, 899, 937, 894, 908, 718, 935, 923, 880, 667, 617, 670, 623, 650, 872, 638, 838, 912, 941, 765, 971, 983, 688, 744, 827, 818, 860, 966, 845, 610, 898, 826, 811,
67, 989, 874, 995, 940, 683, 615, 804, 966, 843, 722, 749, 928, 885, 848, 702, 744, 783, 715, 910, 684, 968, 723, 602, 919, 915, 759, 688, 772, 858, 838, 882, 672, 637, 959, 816, 948, 864, 822, 741, 665, 838, 985, 641, 7
, 791, 959, 789, 926, 691, 918]
idxs = findall(x -> x > 600, y)
println(idxs)
[1, 2, 3, 7, 9, 11, 12, 20, 26, 32, 37, 39, 40, 42, 43, 47, 50, 52, 53, 55, 59, 60, 64, 75, 77, 78, 79, 83, 84, 87, 90, 91, 92, 93, 100, 104, 105, 109, 110, 114, 116, 118, 119, 120, 124, 132, 133, 134, 136, 137, 139, 140
144, 145, 146, 154, 160, 161, 163, 165, 169, 173, 176, 180, 181, 183, 187, 191, 192, 193, 195, 198, 201, 203, 205, 206, 207, 209, 210, 213, 215, 221, 222, 224, 225, 226, 230, 231, 232, 234, 235, 238, 240, 247, 249]
println(x[idxs])
[932, 150, 275, 835, 710, 403, 527, 353, 864, 912, 74, 755, 451, 534, 799, 390, 789, 168, 156, 968, 890, 550, 164, 335, 668, 232, 31, 575, 35, 520, 932, 620, 902, 401, 154, 656, 258, 529, 592, 264, 117, 211, 267, 266, 41
491, 146, 143, 478, 682, 840, 879, 449, 995, 610, 501, 769, 708, 659, 582, 652, 284, 408, 886, 432, 917, 487, 913, 787, 405, 140, 512, 236, 66, 336, 923, 352, 820, 620, 954, 839, 531, 307, 834, 663, 396, 841, 86, 529, 62
628, 915, 346, 942, 311]
mean_x = sum(x) / length(x)
println(mean_x)
500.54
println(abs (x - mean_x ,^ 0.5))
[909.6272487163515, 127.62724871635 14, 252.62724871635 14, 354.62724871635 14, 497.62724871635 14, 682.62724871635 15, 812.62724871635 15, 139.62724871635 14, 687.62724871635 15, 176.62724871635 14, 380.62724871635 14, 504.62724871
63514, 516.62724871635 15, 254.62724871635 14, 705.62724871635 15, 515.62724871635 15, 218.62724871635 14, 245.62724871635 14, 33.62724871635 14, 330.62724871635 14, 211.62724871635 14, 59.62724871635 14, 242.62724871635 14, 348.62
487163514, 67.62724871635 14, 841.62724871635 15, 616.62724871635 15, 457.62724871635 14, 612.62724871635 15, 741.62724871635 15, 910.62724871635 15, 889.62724871635 15, 219.62724871635 14, 356.62724871635 14, 370.62724871635 14, 9
.62724871635 15, 51.62724871635 14, 334.62724871635 14, 732.62724871635 15, 428.62724871635 14, 211.62724871635 14, 511.62724871635 14, 776.62724871635 15, 319.62724871635 14, 538.62724871635 14, 140.62724871635 14, 367.62724871635
, 110.62724871635 14, 512.62724871635 15, 766.62724871635 15, 575.62724871635 15, 145.62724871635 14, 133.62724871635 14, 102.62724871635 14, 945.62724871635 15, 646.62724871635 15, 318.62724871635 14, 929.62724871635 15, 867.62724
163515, 527.62724871635 15, 424.62724871635 14, 488.62724871635 14, 487.62724871635 14, 141.62724871635 14, 55.62724871635 14, 736.62724871635 15, 363.62724871635 14, 77.62724871635 14, 581.62724871635 14, 680.62724871635 15, 969.62
248716351 5, 960.62724871635 15, 908.62724871635 15, 786.62724871635 15, 812.62724871635 14, 115.62724871635 14, 645.62724871635 15, 209.62724871635 14, 8.62724871635 14, 1403, 385.62724871635 14, 159.62724871635 14, 321.62724871635 14,
52.62724871635 15, 12.62724871635 1403, 385.62724871635 14, 838.62724871635 15, 497.62724871635 14, 357.62724871635 14, 495.62724871635 14, 909.62724871635 15, 597.62724871635 15, 879.62724871635 15, 378.62724871635 14, 865.62724871
3515, 551.62724871635 15, 781.62724871635 15, 257.62724871635 14, 168.62724871635 14, 315.62724871635 14, 131.62724871635 14, 951.62724871635 15, 230.62724871635 14, 851.62724871635 15, 633.62724871635 15, 235.62724871635 14, 543.62
2487163515, 410.62724871635 14, 286.62724871635 14, 586.62724871635 14, 569.62724871635 15, 399.62724871635 14, 197.62724871635 14, 871.62724871635 15, 241.62724871635 14, 954.62724871635 15, 94.62724871635 14, 835.62724871635 15,
8.62724871635 14, 244.62724871635 14, 243.62724871635 14, 747.62724871635 15, 118.62724871635 14, 965.62724871635 15, 302.62724871635 14, 664.62724871635 15, 679.62724871635 15, 714.62724871635 15, 762.62724871635 14, 762.62724871
15, 96.62724871635 14, 338.62724871635 14, 468.62724871635 14, 123.62724871635 14, 120.62724871635 14, 34.62724871635 14, 455.62724871635 14, 659.62724871635 15, 809.62724871635 15, 817.62724871635 15, 856.62724871635 15, 21.62724871
6351403, 433.62724871635 14, 277.62724871635 14, 426.62724871635 14, 972.62724871635 15, 587.62724871635 15, 795.62724871635 15, 128.62724871635 14, 472.62724871635 14, 327.62724871635 14, 707.62724871635 15, 768.62724871635 15, 20
62724871635 14, 478.62724871635 14, 142.62724871635 14, 349.62724871635 14, 1.3727512836648597, 382.62724871635 14, 15.62724871635 1403, 746.62724871635 15, 685.62724871635 15, 954.62724871635 15, 636.62724871635 15, 39.62724871635
, 559.62724871635 15, 539.62724871635 15, 591.62724871635 15, 160.62724871635 14, 629.62724871635 15, 631.62724871635 15, 802.62724871635 15, 791.62724871635 15, 261.62724871635 14, 876.62724871635 15, 856.62724871635 15, 385.62724
163514, 163.62724871635 14, 19.62724871635 1403, 381.62724871635 14, 863.62724871635 15, 409.62724871635 14, 257.62724871635 14, 894.62724871635 15, 54.62724871635 14, 206.62724871635 14, 877.62724871635 15, 464.62724871635 14, 642
27248716351 5, 784.62724871635 15, 660.62724871635 15, 890.62724871635 15, 764.62724871635 15, 382.62724871635 14, 587.62724871635 15, 117.62724871635 14, 869.62724871635 15, 356.62724871635 14, 489.62724871635 14, 260.62724871635 14
622.62724871635 15, 213.62724871635 14, 82.62724871635 14, 43.62724871635 14, 938.62724871635 15, 313.62724871635 14, 900.62724871635 15, 329.62724871635 14, 49.62724871635 14, 797.62724871635 15, 597.62724871635 15, 268.62724871635
4, 232.62724871635 14, 931.62724871635 15, 849.62724871635 15, 816.62724871635 15, 544.62724871635 15, 275.62724871635 14, 489.62724871635 14, 657.62724871635 15, 34.62724871635 14, 508.62724871635 14, 284.62724871635 14, 222.62724
163514, 811.62724871635 15, 640.62724871635 15, 373.62724871635 14, 409.62724871635 14, 282.62724871635 14, 439.62724871635 14, 818.62724871635 15, 63.62724871635 14, 506.62724871635 14, 23.62724871635 1403, 601.62724871635 15, 605
27248716351 5, 416.62724871635 14, 646.62724871635 15, 892.62724871635 15, 24.62724871635 1403, 323.62724871635 14, 633.62724871635 15, 699.62724871635 15, 416.62724871635 14, 524.62724871635 15, 422.62724871635 14, 200.62724871635
, 919.62724871635 15, 849.62724871635 15, 288.62724871635 14, 817.62724871635 15]
println(count(x -> maximum(y) - x <= 200, y))
55

Часть 2


lab2_starovoitov.ipynb
Code
Julia 1.11.0-rc3
8.6272487163514, 244.6272487163514, 243.6272487163514, 747.6272487163515, 118.6272487163514, 965.6272487163515, 392.6272487163514, 664.6272487163515, 679.6272487163515, 714.6272487163515, 710.6272487163515, 762.6272487
15, 96.6272487163514, 338.6272487163514, 468.6272487163514, 123.6272487163514, 128.6272487163514, 34.6272487163514, 455.6272487163514, 659.6272487163515, 809.6272487163515, 817.6272487163515, 856.6272487163515, 21.62724
6351403, 433.6272487163514, 277.6272487163514, 426.6272487163514, 972.6272487163515, 587.6272487163515, 795.6272487163515, 128.6272487163514, 472.6272487163514, 327.6272487163514, 707.6272487163515, 760.6272487163515, 2
6272487163514, 478.6272487163514, 142.6272487163514, 349.6272487163514, 1.372751283648597, 382.6272487163514, 15.627248716351403, 746.6272487163515, 685.6272487163515, 954.6272487163515, 636.6272487163515, 39.6272487163
, 559.6272487163515, 539.6272487163515, 591.6272487163515, 160.6272487163514, 629.6272487163515, 631.6272487163515, 802.6272487163515, 791.6272487163515, 261.6272487163514, 876.6272487163515, 856.6272487163515, 385.6272
163514, 163.6272487163514, 19.627248716351403, 381.6272487163514, 863.6272487163515, 409.6272487163514, 257.6272487163514, 894.6272487163515, 54.6272487163514, 286.6272487163514, 877.6272487163515, 464.6272487163514, 64
272487163515, 784.6272487163515, 660.6272487163515, 890.6272487163515, 764.6272487163515, 382.6272487163514, 587.6272487163515, 117.6272487163514, 869.6272487163515, 356.6272487163514, 489.6272487163514, 260.627248716351
622.6272487163515, 213.6272487163514, 82.6272487163514, 43.6272487163514, 938.6272487163515, 313.6272487163514, 900.6272487163515, 329.6272487163514, 49.6272487163514, 797.6272487163515, 597.6272487163515, 268.627248716
4, 232.6272487163514, 931.6272487163515, 849.6272487163515, 816.6272487163515, 544.6272487163515, 275.6272487163514, 489.6272487163514, 657.6272487163515, 34.6272487163514, 508.6272487163514, 284.6272487163514, 222.6272
163514, 811.6272487163515, 640.6272487163515, 373.6272487163514, 409.6272487163514, 282.6272487163514, 439.6272487163514, 818.6272487163515, 63.6272487163514, 586.6272487163514, 23.627248716351403, 601.627248716351 5, 60
272487163515, 416.6272487163514, 646.6272487163515, 892.6272487163515, 24.627248716351403, 323.6272487163514, 633.6272487163515, 699.6272487163515, 416.6272487163514, 524.6272487163515, 422.6272487163514, 200.627248716
, 919.6272487163515, 849.6272487163515, 288.6272487163514, 817.6272487163515]
[73]:
println(count(x -> maximum(y) - x <= 200, y))
55
[75]:
evens = count(iseven, x)
odds = count(!iseven, x)
println(evens, " ", odds)
133 117
[76]:
count(x -> x % 7 == 0, x)
[76]: 34
[77]:
x[sortperm(y)]
[77]: 250-element Vector{Int64}:
818
38
961
105
241
538
191
433
977
234
608
165
656
353
592
449
917
620
275
902
841
864
755
491
143
[78]:
sort(x, rev=true)[1:10]
[78]: 10-element Vector{Int64}:
995
992
988
983
977
977
976
974
968
961
[79]:
unique(x)
[79]: 228-element Vector{Int64}:
932
150
275
377
520
705
835
162
710
199
483
527
539
624
628
439
915
47
346
722
547
445
223
942
311

Часть 3

Задача 4

```
# Задание 4
[2^i / i for i in 1:25]
```

```
25-element Vector{Float64}:
      2.0
      2.0
      2.6666666666666665
      4.0
      6.4
     10.666666666666666
     18.285714285714285
     32.0
     56.888888888888886
    102.4
    186.1818181818182
    341.3333333333333
    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
```

Задача 4

Задача 5

```
# Задание 5
import Pkg; Pkg.add("Primes")
using Primes
myprimes = primes(1000)
println("89th prime = ", myprimes[89])
println("Slice from 89th to 99th = ", myprimes[89:99])
```

```
89th prime = 461
Slice from 89th to 99th = [461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]
   Resolving package versions...
  No Changes to `~/.julia/environments/v1.11/Project.toml`
  No Changes to `~/.julia/environments/v1.11/Manifest.toml`
```

Задача 5

Задача 6

```julia
# Задание 6
# 6.1
expr1 = sum(i^3 + 4 * i^2 for i in 10:100)
println("Sum 6.1 =", expr1)

# 6.2
expr2 = sum((2^i / i) + (3^i / i^2) for i in 1:25)
println("Sum 6.2 =", expr2)

# 6.3
expr3 = sum(prod(2:2:(2 * i - 1)) / prod(1:2:i) for i in 1:19)
println("Sum 6.3 =", expr3)
```

```
Sum 6.1 =26852735
Sum 6.2 =2.1291704368143802e9
Sum 6.3 =-1.914760197961644e11
```

Задача 6

Выводы

Я изучил несколько структур данных, реализованных в Julia, научился применять их и операции над ними для решения задач, выполнив все предложенные задания.

Список литературы