

# Лабораторная работа № 2

## Структуры данных

Старовойтов Е. С.

21 ноября 2024

## Информация

### Докладчик

- Старовойтов Егор Сергеевич
- студент кафедры ТВиК
- Российский университет дружбы народов
- [1032212281@pfur.ru](mailto:1032212281@pfur.ru)

## Вводная часть

### Цели и задачи

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

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

### Результаты

Поставленные боевые задачи были выполнены, все цели достигнуты.

## Выполнение лабораторной работы

### Задание 1

Были повторены все примеры.

```
lab2.ipynb
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 x +
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
[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("abracadabra")

[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_starovoirov.ipynb
[31]: 2

[32]: # Массивы

[33]: # 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 $1 \times 8$)
# со значениями, случайно распределёнными на интервале [0, 1]:
c = rand(1,8)
# многомерный массив $2 \times 3$ (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 $3^k \times 2$,
# где $k$ - 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 $2 \times 3$ из единиц:
ones(2,3)
# одномерный массив из 4 нулей:
zeros(4)
# заполнить массив $3 \times 2$ цифрами 3 и 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 $2 \times 6$
a = collect(1:12)
b = reshape(a,(2,6))
# транспонирование
b'
# транспонирование
c = transpose(b)
# массив $10 \times 5$ 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}:
      1
      9
      49
```

## Массивы

## Задание 2 - самостоятельная работа

### Задачи 1 и 2

```
lab2_starovoitov.ipynb
[37]:
# Задания для самостоятельного выполнения

[38]: # 1
A = Set{[0, 3, 4, 9]}
B = Set{[1, 3, 4, 7]}
C = Set{[0, 1, 2, 4, 7, 8, 9]}

P = union(union(intersect(A, B), intersect(A, C)), intersect(B, C))

[38]: Set{Int64} with 6 elements:
0
4
7
9
3
1

[41]: # 2 - ЭТО НЕВОЗМОЖНО
A = Set{1, 2, 3}
B = Set{"a", "b", "c"}
union(A, B)
intersect(A, B)

MethodError: no method matching Set{::Int64, ::Int64, ::Int64}
The type `Set` exists, but no method is defined for this combination of argument types
when trying to construct it.

Closest candidates are:
  Set{::Any}
    @ Base set.jl:48
  Set{()
    @ Base set.jl:48

Stacktrace:
 [1] top-level scope
      @ In[41]:2
```

### Задачи 1 и 2

## Задача 3

```
Stacktrace:
[1] top-level scope
@ In[41]:2

[42]: # 3. Создание массивов разными способами

[54]: N = 30
arr1 = collect{1:N}
arr2 = reverse(arr1)
arr3 = vcat{arr1, arr2[2:end]}
tmp = [4, 6, 3]
arr5 = fill(tmp[1], 10)
arr6 = repeat(tmp, 10)
arr7 = [fill(tmp[1], 1); fill(tmp[2], 10); fill(tmp[3], 10)]
arr8 = [fill(tmp[1], 10); fill(tmp[2], 20); fill(tmp[3], 30)]
arr9 = [2 * tmp[i] for i in 1:3 for _ in 1:(i == 3 ? 4 : 1)]
println("arr = ", arr9)
println("Count of 6 in arr9 = ", count(x == 6, arr9))
arr9 = [8, 12, 6, 6, 6, 6]
Count of 6 in arr9 = 4

[57]: x_values = 3:0.1:6
y_values = x_values * cos(x_values)
println("Mean y = ", sum(y_values) / length(y_values))
Mean y = -0.00218125833785592

[58]: vector11 = [(0.1 * i, 0.2 * j) for i in 3:3:36, j in 1:3:34]

[58]: 12x12 Matrix{Tuple{Float64, Float64}}:
(0.3, 0.2) (0.3, 0.8) (0.3, 1.4) ... (0.3, 5.6) (0.3, 6.2) (0.3, 6.8)
(0.6, 0.2) (0.6, 0.8) (0.6, 1.4) ... (0.6, 5.6) (0.6, 6.2) (0.6, 6.8)
(0.9, 0.2) (0.9, 0.8) (0.9, 1.4) ... (0.9, 5.6) (0.9, 6.2) (0.9, 6.8)
(1.2, 0.2) (1.2, 0.8) (1.2, 1.4) ... (1.2, 5.6) (1.2, 6.2) (1.2, 6.8)
(1.5, 0.2) (1.5, 0.8) (1.5, 1.4) ... (1.5, 5.6) (1.5, 6.2) (1.5, 6.8)
(1.8, 0.2) (1.8, 0.8) (1.8, 1.4) ... (1.8, 5.6) (1.8, 6.2) (1.8, 6.8)
(2.1, 0.2) (2.1, 0.8) (2.1, 1.4) ... (2.1, 5.6) (2.1, 6.2) (2.1, 6.8)
(2.4, 0.2) (2.4, 0.8) (2.4, 1.4) ... (2.4, 5.6) (2.4, 6.2) (2.4, 6.8)
(2.7, 0.2) (2.7, 0.8) (2.7, 1.4) ... (2.7, 5.6) (2.7, 6.2) (2.7, 6.8)
(3.0, 0.2) (3.0, 0.8) (3.0, 1.4) ... (3.0, 5.6) (3.0, 6.2) (3.0, 6.8)
(3.3, 0.2) (3.3, 0.8) (3.3, 1.4) ... (3.3, 5.6) (3.3, 6.2) (3.3, 6.8)
(3.6, 0.2) (3.6, 0.8) (3.6, 1.4) ... (3.6, 5.6) (3.6, 6.2) (3.6, 6.8)

[60]: vector12 = [2 * i / j for i in 1:25]

[60]: 25-element Vector{Float64}:
 2.0
 2.0
 2.0000000000000005
 4.0
 6.4
18.000000000000006
18.285714285714285
32.0
56.888888888888886
102.4
186.18181818181818
341.3333333333333
630.1538461538462
1170.2857142857142
2184.3333333333333
4096.0
7710.117647058823
14503.555555555555
27594.105263157893
52428.8
99864.38895238895
198050.18181818182
384722.888955517
699050.6666666666
1.34217728e6

[61]: vector13 = ["fn$i" for i in 1:30]

[61]: 30-element Vector{String}:
"fn1"
"fn2"
"fn3"
"fn4"
"fn5"
"fn6"
"fn7"
"fn8"
"fn9"
"fn10"
"fn11"
"fn12"
"fn13"
⋮
"fn19"
"fn20"
"fn21"
"fn22"
"fn23"
"fn24"
"fn25"
"fn26"
"fn27"
"fn28"
"fn29"
"fn30"

[63]: n = 250
x = rand(0:999, n)
y = rand(0:999, n)
```

Часть 1



```
103: 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+1] + 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]) / (x[i]+10) for i in 1:(n-1)))

[1, 824, -206, 162, -152, 72, -371, 585, -330, 458, 277, -265, -257, 191, -713, -525, 120, 233, 985, -122, -32, 409, 189, -78, 896, -347, -85, -435, -389, -316, -231, -602, 17, -241, -312, -78, 311, 632, 144, -429, 783,
0, -219, -192, -205, 745, 58, 248, 183, -366, 337, 755, -41, 755, -896, -201, -125, -205, -278, -179, -396, -134, 248, 12, 484, -375, -317, 387, -113, -588, -549, -685, -766, -186, 189, 512, 284, 486, 477, 385, 245, 494,
37, 525, 18, 88, -371, 76, 247, 39, 383, -214, 197, -785, -489, -657, -264, 15, 486, 142, -514, -83, -47, 162, -152, -558, -43, 551, 437, -119, -102, -37, -49, -19, -307, 145, 48, 615, 544, 151, -691, 96, -121, 129, -224,
-334, -388, -148, -687, 445, 628, 383, 849, 115, 883, 285, -448, -217, -36, -488, 166, 98, 666, 394, -273, -688, -736, 412, -195, -116, -461, -384, 517, -471, 88, -192, 344, -488, 898, 116, -698, -129, -289, 648, -387,
0, 241, 561, -387, -275, -785, -31, -177, -859, -164, -99, -145, 182, 586, -202, 1185, 888, -348, -248, -168, -177, -231, -585, -393, -81, 6, 128, -32, 149, 88, -879, 389, -482, -216, 127, -226, 753, -58, -123, 546, -251,
-264, 565, 139, -558, -214, 561, -741, 76, -488, -133, 179, -489, -184, 807, 291, -215, 486, -169, 175, -56, 36, -238, 523, -288, 781, -219, 745, 335, -48, 14, 128, -366, 879, -311, -436, -429, -282, -318, 25, 468, -281,
6, -47]

[957, 323, 589, 712, 1895, 2213, 449, 1383, 785, 478, 918, 1328, 365, 1195, 1563, 752, 721, 27, 528, 739, 133, 241, 917, -313, 1179, 1662, 964, 986, 1230, 1718, 2515, 1817, 687, 189, 2271, 767, 33, 1416, 1423, 385, 583,
98, 922, 1381, 497, 818, 121, 414, 1515, 1817, 778, 355, -562, 1392, 1905, 399, 1355, 2182, 1543, 1813, 879, 1127, 688, -439, 1218, 1431, 62, 445, 938, 1704, 2827, 2836, 2214, 1341, -57, 1242, 1181, -114, 645, 428, 295,
59, 237, -48, 1618, 1521, 762, 484, 1762, 1278, 2823, 816, 1688, 1232, 1982, 1173, 724, 328, 1848, 686, 1345, 1392, 686, 957, 1123, 522, 775, 1291, 1216, -32, 3744, 445, 2181, 353, 1622, 1013, 479, 29, 1665, 64, 12,
1131, 1887, 1354, 1441, 158, 2184, 662, 598, 1197, 648, 375, -221, 331, 1818, 1586, 333, 2554, 311, 656, 687, 283, 1829, 1397, 2895, 625, 791, 465, 1027, 2864, 746, 1869, 459, 888, 9, 793, -288, 868, 1288, 2889, 2233,
281, 664, 1892, 1607, 328, 813, 1155, 1490, 2169, 483, 1283, 2249, 1589, 738, -134, -36, 1744, 1478, 75, 2837, 842, -365, 1542, 1289, 1818, 1596, 1268, 1722, 2882, 987, 1485, -2, 1545, 1138, 1128, 432, 1337, 1812, -380,
4, 1652, 78, 1839, 1555, 324, 1892, 1769, 947, -153, 1291, 1859, 1983, 1675, 651, 642, 1815, 263, 812, 988, -37, 1258, 1764, 1823, 955, 588, 388, 2858, 464, 1898, -3, 666, 1441, 837, 662, 2452, 663, 83, 936, 1661, 1853]

1888, 1214, -51, 1235, 275, 654]

1.1965581521888346, 0.4705936183553559, 1.0688161896284343, 1.7205461467478996, 3.44345148356467, 0.534907243725564, 4.143461622097187, 0.816725143201953, 1.763784111202977, 0.286713783858559, 0.5612188476
854, 4.59283329793962, 1.10378867401058265, -1.0248163611058265, -0.13774883833814805, -0.04962287821058265, -0.37490233814805, -0.3274455664492821, -0.394677282863829, -0.611425656492821, 1.7988358
7842315, 0.8254809961826832, -2.23888279347877, 0.7486249539812586, 1.4371565755544784, -1.24218189818444949, 0.9560563962959, 1.6208798312334423, 0.817989269488476, 1.0837165881424238, 0.99487240194855, -1.0858344
698186, -1.82887723816422, 0.446937888211764, 1.6681876286119815, 1.1424879788748729, 1.888986311258419, 3.13681876286119815, 0.93848739518883, 0.088873589156972114, 1.415372925379133, 1.0761598318185533, -2.75223
99383923, -0.764555171088874, 0.7484318892142887, 0.29427594134116947, 0.838184151472411, 0.278048189144948, -11.895878248386797, -1.0719457481167597, -0.7481162181988687, 1.0219678121988687, 0.51844888889865,
-3.7588588482625595, -0.897614888615882, -1.163176471738672, -0.852597884959586, 1.51184824494254, -0.349298843474832, 0.7271166295995658, 1.235256741287724, 0.886674578788882, 0.233388898984566,
0.9698884738154845, 0.7696113568891369, 0.1437479693814839, 0.8784874988831489, 1.197644268211377, -0.995831494268713, 0.8784874988831489, 21.218973791457745, -2.448248482248481, 0.448988232721487, 0.858181458
939, 0.341879288352582, -1.0606322217877813, -0.27647088855562, -0.8249158862869273, -141.7315488381178, 0.2547923580485687, -0.7978335226788753, 0.87828831284748, -1.017482855862432, -2.522171671842745, -1.0485114
8928813, 1.04258488867736, 0.91328583999225, 2.2314551746382863, 0.262432171559424, 0.724523725981212, 0.818148471986588, -1.452871466813765, 0.789946483718815, 0.1918253431591767, 0.8588537578482817, -1.48418
88848328, 0.876515742446814, 0.328331246034588, -0.266367386572841, 1.28733381391971, -0.437088437999385, 0.45554868628852, 1.8029676362483885, 0.4766466396458287, 0.662183662489736, 1.228872382398982, -3.748
722528865, -1.929931419931887, -0.18553771818529413, 0.268283728787441, 0.7126969791568416, -0.088367325979313, 0.084242336871367, 0.538166267888898, 1.088573245351332, 0.4757862826789373, -0.46763752481654,
0.793328795784939, -14.37972535377195, -1.831818718616236, 0.1491649679782848, 3.2231586217261, 1.171948558919676, 0.7886797781979684, 1.068743957122883, 0.63558463814436, 0.577161152938313,
1.028615395817884, 0.74174878138155, 18.7474831514854, 0.89942148507781, 0.429392438316511, 0.64885859131126, 1.0334847898363, -2.71827897428686, 0.835591316290532, 0.2452316623499842, 0.26582334699865,
1.16897158141817, -0.08958506212451, 1.57095357185559, 1.08778473762531, 1.428964728442317, 0.92688571770886, 1.5832179471729935, 2.14746252528736, -2.458219888379197, -2.58878481693355, -0.38316823699918,
1.863428893411567, -14.52963813818636, -3.59187765538195, -1.81636996640438, 0.838214882248891, 0.59688711523813, 1.24354646724281, 2.265231331919765, 0.881598328389113, -1.018488175654913, -0.338136846885155,
3.885312888881131, 1.851241418942481, -2.24683867527485, 1.22614653587955, 2.62667829576187, 0.62863362638887, 1.469388876548777, -0.59635831835882, 1.878894722288878, -0.77317837821654, 0.21481644443831,
1.0342288652582, 1.2889186583456363, -2.2545862889712887, 0.5278494788728492, -0.8186357259318914, -28.16524813640186, 0.827261268998985, 1.068888483728257, -12.257882645269921, 0.38477384885895, 6.38396468326477
0.90743488892594, 0.79883658810481, 1.888825918959186, 3.482732607872887, 1.77294151688878, 28.86244138108393, 1.63263859182868, 0.829118815954673, -4.59688269941375, -2.08474562718877,
5.8863772185489, 0.808887434761188, 0.8786878752748816, 0.92438672622873, 3.674262587474836, 0.844212609249518, -0.3738737367138455, -1.088265238538454, 0.891182485778379, 0.15728466469273, 0.88634556718864,
-0.8543889226108, -1.5184861348381882, 1.835658285791157, 0.8572162866286893, 1.9888610529624688, -3.520368795366871, 0.598997281186434, -12.085647555308742, -1.188817861635594, 0.3388435815048489, -3.234191688
669, -0.948384821868782, 0.888185243484959, 0.888185243484959, -0.888352218494584, 0.48781273886237, 0.487185848526524, -0.861786818129883, 23.5486151915456, -0.87926385166835, 0.3388435815048489, -0.68716717
83141, 2.5982959258724657, 0.3254346874824774, -2.33274701394937, -2.33274701394937, -0.653282226652218, -1.86934679186464, -0.842882141873815, -0.828187248487148, 0.447973883867289, 0.778859282812559, -0.838
6312935884, -0.587393777716017, 0.1259938927838483, -0.7986916238957457, -0.1775282867451318, -0.3384277282918016, -1.0646557885814447, -0.7253614868938, -0.97487373887622, -0.5385626947369444]

1.98518687488525e-12

164: println()
println(filter(x -> x < 680, y))

[689, 933, 974, 777, 667, 657, 680, 961, 986, 782, 898, 989, 899, 937, 894, 988, 718, 935, 923, 888, 667, 617, 670, 623, 658, 872, 638, 838, 912, 941, 705, 971, 983, 688, 744, 827, 818, 868, 966, 845, 610, 898, 826, 811,
97, 989, 874, 995, 940, 683, 615, 884, 966, 843, 722, 749, 928, 928, 885, 848, 782, 744, 783, 715, 918, 884, 968, 723, 682, 919, 915, 759, 688, 772, 858, 838, 882, 672, 637, 959, 816, 848, 864, 822, 741, 665, 838, 985, 641, 7,
701, 599, 789, 926, 691, 918]

167: [idx = findall(x -> x < 680, y)
println(idx)]

[1, 2, 3, 7, 9, 1, 12, 28, 26, 32, 37, 165, 169, 42, 43, 47, 58, 52, 53, 55, 59, 68, 64, 75, 77, 78, 79, 83, 84, 85, 89, 91, 92, 93, 180, 184, 185, 189, 110, 114, 116, 118, 119, 120, 124, 132, 133, 134, 136, 137, 139, 148
144, 145, 146, 154, 168, 169, 167, 169, 173, 176, 168, 181, 183, 187, 191, 192, 193, 195, 198, 201, 203, 205, 286, 287, 289, 210, 213, 215, 221, 222, 224, 225, 226, 230, 231, 232, 234, 235, 236, 247, 249]

168: println(x[idx])

[932, 158, 275, 835, 710, 483, 527, 353, 864, 812, 74, 755, 451, 534, 799, 380, 789, 168, 156, 968, 886, 558, 164, 335, 668, 232, 31, 575, 35, 526, 932, 628, 902, 481, 154, 656, 258, 529, 502, 264, 117, 211, 267, 266, 43,
491, 146, 143, 478, 682, 848, 879, 449, 995, 618, 501, 769, 708, 659, 582, 652, 284, 488, 886, 432, 917, 487, 913, 787, 485, 148, 512, 236, 66, 336, 923, 352, 828, 628, 954, 839, 531, 387, 834, 663, 396, 841, 86, 529, 62,
628, 915, 346, 942, 311]

171: mean_x = sum(x) / length(x)
println(mean_x)

580.54

172: println(abs(x - mean_x * 0.5))

[989, 6272487163515, 127, 6272487163514, 252, 6272487163514, 354, 6272487163514, 497, 6272487163514, 682, 6272487163515, 812, 6272487163515, 139, 6272487163514, 687, 6272487163514, 176, 6272487163514, 380, 6272487163514, 584, 627248
63514, 516, 6272487163515, 254, 6272487163514, 785, 6272487163515, 515, 6272487163515, 218, 6272487163514, 245, 6272487163514, 33, 6272487163514, 338, 6272487163514, 211, 6272487163514, 59, 6272487163514, 242, 6272487163514, 348, 62
487163514, 67, 6272487163514, 64, 6272487163515, 616, 6272487163515, 457, 6272487163514, 612, 6272487163515, 741, 6272487163515, 918, 6272487163515, 889, 6272487163515, 219, 6272487163514, 356, 6272487163514, 370, 6272487163514, 9,
6272487163515, 51, 6272487163514, 334, 6272487163514, 732, 6272487163515, 428, 6272487163514, 211, 6272487163514, 511, 6272487163514, 776, 6272487163515, 319, 6272487163514, 538, 6272487163515, 148, 6272487163514, 329, 6272487163515, 867, 62724
118, 6272487163514, 512, 6272487163515, 766, 6272487163515, 375, 6272487163515, 145, 6272487163514, 133, 6272487163514, 182, 6272487163514, 945, 6272487163515, 646, 6272487163515, 318, 6272487163514, 929, 6272487163515, 867, 62724
163515, 327, 6272487163515, 424, 6272487163514, 488, 6272487163514, 487, 6272487163514, 141, 6272487163514, 55, 6272487163514, 748, 6272487163515, 383, 6272487163514, 77, 6272487163514, 561, 6272487163514, 688, 6272487163515, 969, 6
272487163515, 986, 6272487163515, 988, 6272487163515, 786, 6272487163515, 312, 6272487163514, 115, 6272487163514, 645, 6272487163515, 289, 6272487163514, 885, 6272487163514, 159, 6272487163514, 321, 6272487163514, 487, 6272487163515, 12, 6272487163515, 12, 6272487163514, 385, 6272487163514, 838, 6272487163514, 357, 6272487163514, 495, 6272487163514, 989, 6272487163515, 597, 6272487163515, 879, 6272487163515, 374, 6272487163514, 865, 6272487163514,
3518, 851, 6272487163515, 781, 6272487163515, 257, 6272487163514, 168, 6272487163514, 155, 6272487163514, 131, 6272487163514, 931, 6272487163515, 238, 6272487163514, 863, 6272487163515, 631, 6272487163515, 235, 6272487163514, 543, 6
272487163515, 418, 6272487163514, 288, 6272487163514, 586, 6272487163514, 569, 6272487163515, 399, 6272487163514, 192, 6272487163514, 871, 6272487163514, 241, 6272487163514, 954, 6272487163515, 714, 6272487163515, 855, 6272487163515, 855, 6272487163515,
8, 6272487163514, 244, 6272487163514, 243, 6272487163514, 747, 6272487163515, 118, 6272487163514, 965, 6272487163514, 392, 6272487163514, 664, 6272487163515, 679, 6272487163515, 714, 6272487163515, 714, 6272487163515, 762, 627248716
15, 96, 6272487163514, 338, 6272487163514, 464, 6272487163514, 129, 6272487163514, 128, 6272487163514, 34, 6272487163514, 455, 6272487163514, 689, 6272487163515, 880, 6272487163515, 817, 6272487163515, 856, 6272487163515, 21, 627248
6351483, 433, 6272487163514, 277, 6272487163514, 426, 6272487163514, 972, 6272487163515, 587, 6272487163515, 795, 6272487163515, 128, 6272487163514, 472, 6272487163514, 327, 6272487163514, 787, 6272487163515, 768, 6272487163515, 28, 6272487163514, 478, 6272487163514, 478, 6272487163514, 349, 6272487163514, 137, 6272487163514, 15, 6272487163514, 389, 6272487163514, 746, 6272487163515, 685, 6272487163515, 954, 6272487163515, 636, 6272487163515, 39, 6272487163515, 129
599, 6272487163515, 539, 6272487163515, 591, 6272487163515, 168, 6272487163515, 629, 6272487163515, 631, 6272487163515, 882, 6272487163515, 793, 6272487163515, 261, 6272487163514, 876, 6272487163515, 856, 6272487163515, 385, 62724
6351483,
```



```
lab2_starovtov.py nb
+ Code
8 6272487163514, 244 6272487163514, 248 6272487163515, 118 6272487163514, 965 6272487163515, 392 6272487163514, 694 6272487163515, 879 6272487163515, 714 6272487163515, 718 6272487163515, 762 62724871
15, 96 6272487163514, 138 6272487163514, 468 6272487163514, 121 6272487163514, 128 6272487163514, 34 6272487163514, 455 6272487163514, 659 6272487163515, 889 6272487163515, 817 6272487163515, 856 6272487163515, 21 62724
6351489, 433 6272487163514, 277 6272487163514, 426 6272487163514, 972 6272487163515, 587 6272487163515, 795 6272487163515, 128 6272487163514, 472 6272487163514, 327 6272487163514, 787 6272487163515, 700 6272487163515, 2
6272487163514, 478 6272487163514, 142 6272487163514, 349 6272487163514, 1 372791283648597, 382 6272487163514, 15 627248716351489, 746 6272487163515, 685 6272487163515, 954 6272487163515, 636 6272487163515, 39 6272487163
559 6272487163515, 510 6272487163515, 991 6272487163515, 108 6272487163514, 629 6272487163515, 631 6272487163515, 883 6272487163515, 701 6272487163515, 261 6272487163514, 816 6272487163515, 856 6272487163515, 385 6272
163514, 163 6272487163514, 19 627248716351489, 381 6272487163514, 863 6272487163515, 409 6272487163514, 257 6272487163514, 894 6272487163515, 54 6272487163514, 286 6272487163514, 877 6272487163514, 464 6272487163514, 64
272487163515, 784 6272487163515, 668 6272487163515, 898 6272487163515, 704 6272487163515, 382 6272487163514, 587 6272487163515, 117 6272487163514, 809 6272487163515, 356 6272487163514, 486 6272487163514, 268 62724871635
627 6272487163515, 213 6272487163514, 82 6272487163514, 43 6272487163514, 918 6272487163515, 313 6272487163514, 900 6272487163515, 329 6272487163514, 49 6272487163514, 797 6272487163515, 268 6272487163
4, 232 6272487163514, 931 6272487163515, 849 6272487163515, 816 6272487163515, 544 6272487163515, 275 6272487163514, 489 6272487163514, 657 6272487163514, 34 6272487163514, 588 6272487163514, 284 6272487163514, 222 6272
163514, 811 6272487163515, 648 6272487163515, 378 6272487163514, 409 6272487163514, 222 6272487163514, 439 6272487163514, 818 6272487163515, 63 6272487163514, 586 6272487163514, 23 627248716351489, 681 6272487163515, 64
272487163515, 416 6272487163514, 646 6272487163515, 892 6272487163515, 24 627248716351489, 323 6272487163514, 633 6272487163515, 699 6272487163515, 416 6272487163514, 524 6272487163515, 422 6272487163514, 280 6272487163
, 919 6272487163515, 849 6272487163515, 288 6272487163514, 817 6272487163515]

[73]: printIn(count(x -> maximum(y) - x == 280, y))

55

[75]: evens = count(!iseven, x)
odds = count(!iseven, x)
printIn(evens, " ", odds)

133 117

[76]: count(x -> x % 7 == 0, x)

[76]: 34

[77]: x[sortperm(y)]

[77]: 258-element Vector{Int64}:
 818
  88
 961
 185
 241
 538
 191
 433
 977
 234
 688
 165
 656
   1
 353
 592
 449
 917
 620
 275
 982
 841
 864
 755
 491
 143

[78]: sort(x, rev=true)[1:10]

[78]: 18-element Vector{Int64}:
 995
 992
 988
 983
 977
 977
 976
 974
 968
 961

[79]: unique(x)

[79]: 228-element Vector{Int64}:
 932
 158
 275
 377
 528
 785
 835
 165
 718
 199
 483
 527
 539
   1
 624
 624
 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

```
# 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