

# Лабораторная работа № 2

## Структуры данных

---

Беличева Д. М.

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Беличева Дарья Михайловна
- студентка
- Российский университет дружбы народов
- 1032216453@pfur.ru
- <https://dmbelicheva.github.io/ru/>



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.
2. Выполните задания для самостоятельной работы.

# Выполнение лабораторной работы

```
Кортежи

[29]: # пустой кортеж:
      ()

[29]: ()

[30]: # кортеж из элементов типа String:
      favoritelang = ("Python", "Julia", "R")

[30]: ("Python", "Julia", "R")

[33]: # кортеж из целых чисел:
      x1 = (1, 2, 3)
      # кортеж из элементов разных типов:
      x2 = (1, 2.0, "tmp")
      # именованный кортеж:
      x3 = (a=2, b=1+2)

      print(x1, '\n', x2, '\n', x3)

      (1, 2, 3)
      (1, 2.0, "tmp")
      (a = 2, b = 3)

[34]: # длина кортежа x2:
      length(x2)

[34]: 3

[35]: # обратиться к элементам кортежа x2:
      x2[1], x2[2], x2[3]

[35]: (1, 2.0, "tmp")

[36]: # произвести какую-либо операцию (сложение) с вторым и третьим элементами кортежа x1:
      c = x1[2] + x1[3]

[36]: 5
```

Рис. 1: Примеры использования кортежей

```
▼ Словари
```

```
[39]: # создать словарь с именем phonebook:
phonebook = Dict("Иванов И.И." => ("867-5309", "333-5544"), "Бухгалтерия" => "555-2368")

[39]: Dict{String, Any} with 2 entries:
      "Бухгалтерия" => "555-2368"
      "Иванов И.И." => ("867-5309", "333-5544")

[41]: # вывести ключи словаря:
keys(phonebook)

[41]: KeySet for a Dict{String, Any} with 2 entries. Keys:
      "Бухгалтерия"
      "Иванов И.И."

[43]: # вывести значения элементов словаря:
values(phonebook)

[43]: ValueIterator for a Dict{String, Any} with 2 entries. Values:
      "555-2368"
      ("867-5309", "333-5544")

[45]: # вывести заданные в словаре пары "ключ - значение":
pairs(phonebook)

[45]: Dict{String, Any} with 2 entries:
      "Бухгалтерия" => "555-2368"
      "Иванов И.И." => ("867-5309", "333-5544")

[46]: # проверка вхождения ключа в словарь:
haskey(phonebook, "Иванов И.И.")

[46]: true

[47]: # добавить элемент в словарь:
phonebook["Сидоров П.С."] = "555-3344"

[47]: "555-3344"
```

Рис. 2: Примеры использования словарей

```
Множества

[55]: # создать множество из четырёх целочисленных значений:
A = Set([1, 3, 4, 5])

[55]: Set(Int64) with 4 elements:
  5
  4
  3
  1

[57]: # создать множество из 11 символьных значений:
B = Set("abracadabra")

[57]: Set(Char) with 5 elements:
  'a'
  'd'
  'r'
  'k'
  'b'

[58]: # проверка эквивалентности двух множеств:
S1 = Set([1,2]);
S2 = Set([3,4]);
Issetequal(S1,S2)

[58]: false

[60]: S3 = Set([1,2,2,3,3,1,2,3,2,1]);
S4 = Set([2,3,1]);
Issetequal(S3,S4)

[60]: true

[61]: # объединение множеств:
C = union(S1,S2)

[61]: Set(Int64) with 4 elements:
  4
  2
  3
  1

[63]: # пересечение множеств
D = intersect(S1,S3)
```

Рис. 3: Примеры использования множеств



```
Массивы

70]: # создание пустого массива с абстрактным типом:
empty_array_1 = []

70]: Any[]

71]: # создание пустого массива с конкретным типом:
empty_array_2 = (Int64)[]
empty_array_3 = (Float64)[]

71]: Float64[]

72]: # вектор-столбец:
a = [1, 2, 3]

72]: 3-element Vector{Int64}:
 1
 2
 3

73]: # вектор-строка:
b = [1 2 3]

73]: 1x3 Matrix{Int64}:
 1 2 3

76]: # многомерные массивы (матрицы):
A = [[1, 2, 3] [4, 5, 6] [7, 8, 9]]
B = [[1 2 3]; [4 5 6]; [7 8 9]]
A

76]: 3x3 Matrix{Int64}:
 1 4 7
 2 5 8
 3 6 9

77]: B

77]: 3x3 Matrix{Int64}:
 1 2 3
 4 5 6
 7 8 9
```

Рис. 4: Примеры использования массивов

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

### Задание 1

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

```
[118]: A = Set([0, 3, 4, 9]); B = Set([1, 3, 4, 7]); C = ([0, 1, 2, 4, 7, 8, 9]);
```

```
[124]: union(intersect(A,B), intersect(A,C), intersect(B,C))
```

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

Рис. 5: Задание №1. Работа с множествами

Приведем свои примеры с выполнением операций над множествами элементов разных типов

```
[5]: Set1 = Set([1, 2, 3, "Hello", "Geeks"])
      println(Set1)
      Set{Any{2, "Hello", "Geeks", 3, 1}}

Доступ к элементам набора невозможен по ссылке на значение индекса, поскольку наборы неупорядочены и элементы не имеют фиксированного индекса. Но
можно перебирать элементы набора с помощью цикла for или спросить, присутствует ли указанное значение в наборе, используя ключевое слово in().

[12]: println("\nElements of set: ")
      for i in Set1
          println(i)
      end

      Elements of set:
      2
      Hello
      Geeks
      3
      1

[10]: print(in("Hello", Set1))

      true

[14]: Set1 = push!(Set1, "Welcome")
      println("\nSet after adding one element:\n", Set1)

      Set after adding one element:
      Set{Any{"Welcome", 2, "Hello", "Geeks", 3, 1}}

[15]: for i in 1:5
          push!(Set1, i)
      end
      println("\nSet after adding range of elements:\n", Set1)

      Set after adding range of elements:
      Set{Any{5, 4, "Welcome", 2, "Hello", "Geeks", 3, 1}}
```

## Выполнение лабораторной работы

3.1) массив (1, 2, 3, ..., N - 1, N), N выберите больше 20;

3.2) массив  $(N, N - 1, \dots, 2, 1)$ ,  $N$  выберите больше 20;

3.3) массив  $(1, 2, 3, \dots, N-1, N, N-1, \dots, 2, 1)$ .  $N$  выберите больше 20;

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

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

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

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

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

элемент — 20 раз подряд, третий элемент — 30 раз подряд;

```
1): b_1 = [i for i in 1:21]
print(b_1, '\n')
c = []
c_1 = reverse([i for i in 1:21])
print(c_1, '\n')
print(vcat(b_1, c_1), '\n')
tmp = [4 6 3]
print(tmp, '\n')
print(vcat(fill.(tmp, [10 2 1])...), '\n')
print(vcat(fill.(tmp, [10 10 10])...), '\n')
print(vcat(fill.(tmp, [11, 10, 10])...), '\n')
print(vcat(fill.(tmp, [10, 20, 30])...))
```

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

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

[4 6 3]

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

[4, 6, 6]

[illegible][illegible][illegible][illegible]

Рис. 7: Задание №3. Работа с массивами

3.9) массив из элементов вида  $2^{tmp[i]}$ ,  $i = 1, 2, 3$ , где элемент  $2^{tmp[0]}$  встречается 4 раза; посчитайте в полученном векторе, сколько раз встречается цифра 6, и выведите это значение на экран;

```
84]: tmp_deg = []
      for i in 1:3
          push!(tmp_deg, 2^tmp[i])
      end
      print(vcat(fill(tmp_deg, [1, 1, 4])...))
      count = 0
      for i in tmp_deg
          if '6' in string(i)
              count = count + 1
          end
      end
      println("\n", count)

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

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

```
[9]: y(x) = exp(x)*cos(x)
      Y = [y(x) for x in 3:0.1:6]
      mean(Y)

[9]: 53.11374594642971
```

Рис. 8: Задание №3. Работа с массивами

# Выполнение лабораторной работы

```
- сформируйте вектор (y2 = x1, ..., ym = xm-1):

37]: for i in 1:249
      j = i + 1
      print(y[j] x[i], ",")
    end

9;22;43;310;106;606;-21;696;391;-68;431;302;-22;579;-684;-180;407;-8;166;165;-259;353;-520;-465;457;274;-541;-727;-388;-191;568;733;-211;-319;-318;493;4
90;65;-242;75;48;-85;-47;-425;732;105;771;-811;356;-420;-112;450;235;-54;-542;580;166;145;11;27;-320;-152;-672;355;-389;-284;-84;-762;-174;-425;-266;-13
1;124;-251;-556;-339;583;-108;-6;-198;452;-145;807;92;-132;-573;-75;887;25;-398;46;0;142;-5;-230;806;-258;-394;68;637;468;588;96;233;-468;210;-201;53
4;77;27;-571;-250;447;-488;505;112;-94;-531;-254;-45;323;-211;785;118;-934;-95;-665;-430;-92;-741;188;-375;-139;157;-299;133;-120;519;-401;-333;-380;11
2;131;230;377;-203;66;-766;194;-290;-271;48;-322;41;156;129;580;33;-557;-232;-10;299;705;-689;207;478;505;22;-228;-340;307;275;136;-14;538;549;188;500;-
391;37;70;58;279;199;-141;151;-252;-632;-290;-650;-538;208;-768;-307;-536;-121;671;-182;-347;-99;523;127;308;330;556;-155;-456;-399;56;-178;712;-669;42
2;44;785;357;190;-442;-385;-151;469;-353;187;389;-58;-147;-426;582;205;181;29;156;179;-671;-973;-816;-299;150;-505;314;-3;-530;-572;252;-234;406;651;-11
7;254;

- сформируйте вектор (x1 + 2x2 - x3, x2 + 2x3 - x4, ..., xm-2 + 2xm-1 - xm):

83]: vect_2 = [x[i] + 2*x[i+1] - x[i+2] for i in 1:248]
      show(vect_2)

[1734, 935, 116, 677, 152, 672, 543, -7, 225, 790, 741, 657, -87, 991, 2047, 313, 634, 1958, 777, 52, 555, 1794, 353, 359, 1205, 2167, 936, 1502, 1324,
976, -657, 1875, 382, 1588, 1208, 968, -68, 576, 1851, 811, 832, 911, -259, 978, 247, 1388, 603, 1605, 1690, 1799, 110, 17, 2169, 665, 1503, 883, -412,
747, 1294, -261, 1547, 1388, 336, 1440, 1460, 1338, 1940, 812, 615, 1728, 1450, 927, 320, 654, -391, 1168, 1244, 1055, 1849, 2234, 1457, 568, 696, 852,
8, 1392, 1719, 418, 404, 1570, 369, -400, 1888, 1213, 444, -56, 1121, 2203, 1145, 540, 1397, 2204, 1672, 1143, 760, 776, 1748, 2425, 947, 1605, 994, 80,
715, 612, 1305, 2645, 836, 358, -211, 1551, 1281, 900, 1659, -119, 986, 1553, 213, 1310, 1839, 1220, 2292, 571, 1865, 1407, -181, 1213, 586, 1417, 48, 1
200, 2438, 1889, -143, 891, 1481, 1917, 595, 1060, 791, 1437, 911, -360, 969, 1148, 822, 734, 462, 1545, 1653, 1021, 130, 1655, 830, 583, 546, 104, 900,
-336, 710, 2723, 407, 710, 1855, 1890, 2667, 749, 278, 900, 1216, 1613, 1689, 2173, 509, 2057, 499, 1264, 555, 761, 1089, 526, -29, 1223, 1779, 2488, 13
94, 318, 866, 1582, 747, 1702, 1770, 980, 1046, 2233, 988, -312, 1276, 1646, 2025, 249, 43, 1630, 1810, 2098, 362, 758, 1256, 981, 913, 1317, 1390, 201
9, -265, 1119, 1802, 1082, 1730, 1677, 2067, 1513, 1699, 2298, 952, -435, 570, 1315, 974, -161, 766, 2602, 490, 502, 1501, 1122, -213, 1460, 1389, 1739]
```

Рис. 9: Задание №3. Работа с векторами

# Выполнение лабораторной работы

```
- сформируйте вектор  $(\frac{\sin(y_1)}{\cos(x_2)}, \frac{\sin(y_2)}{\cos(x_3)}, \dots, \frac{\sin(y_n - 1)}{\cos(x_n)})$ :
```

```
vect_3 = [sin(y[i])/cos(x[i+1]) for i in 1:249]
show(vect_3)
```

[ -0.9623545641825781, 1.0272114737656448, 1.0073561631552175, 2.3434876638962242, -1.30775360875392, 2.184835663849713, 1.0084071869229527, 0.1004723151  
0774891, 1.0430160728779123, -6.133653894220555, -1.1073572256540407, 1.2854249007330012, -0.3891499399263308, -0.1340199420565114, 0.37518130648249026,  
0.5211174574923891, 0.8192266067120235, -0.7739595953525381, 0.579179483244327, 1.0010269534259488, 1.0403867675849603, 0.9234825652611104, 1.0521006340  
006278, -0.8250978831490015, 0.48025128257012945, -0.23338063729593764, 0.8956624065272996, -3.0591274560187594, -1.264995693676858, -0.1194102051669962  
7, -0.8584235656119303, -1.5691647462690543, 4.499163461591518, 1.3842792029144968, 0.00021907845026787221, -0.9719941107813822, 0.6449696324563701, 0.2  
9289439257023503, 0.7217155857465929, 1.707443550924714, -1.4309572503119243, 0.4746585562001765, -0.8266760299918168, 4.2249204664685673, -0.99960025106  
86977, 0.9998761269247262, 1.3346239901979329, -4.175700504083413, 0.5278999414515335, -1.1606456192464394, -0.7886930718610737, -7.155133085915265, -0.  
8353517828717201, -0.01112213498037299, 0.2511274519742956, 4.124678738372672, 0.6323106968070983, 1.1652027119095312, 0.3902726259980732, -16.571282322  
05306, 3.9139763145521704, 0.2726781265536376, 6.364726598261218, 0.028279760153971944, -2.767702123370766, -1.1917355242929732, -0.5842792032249161,  
2.2516010893505585, -1.2122130256441008, 0.764195332773509, 0.7677248800651778, -0.8356845390003437, 2.768159854679163, -0.9951799018210317, 0.369801946  
5396002, 0.7807475597357736, -1.003022441691108, 0.44413030470293574, 1.251384209872519, -0.9601093065122769, 0.166998217673871, -2.4025147187503686, 0.  
38031720084736825, -0.929306402603263, 0.991576972680685, 2.9215697138829064, 0.43360251389751053, 0.41345619591658367, 1.462662499816396, -1.1247961857  
706121, -1.4726126782185576, 0.8341505228230625, 10.653488239567194, -0.6588946697449217, 0.6635332931842858, 1.2496735181302858, -10.79362648171079, 0.  
21618958853358852, -0.03546412945474173, 4.644007453106214, -0.5301481819779847, 0.9830537400933795, -0.05944111444405744, 1.2359969293654358, -2.258006  
8001851443, -0.9163766028420628, -1.404084371933541, -0.6802249339260237, -0.2814875048843556, 6.540723070583812, 0.9071160527673966, -0.997049055385913  
2, -1.8187293345099332, -0.6971074018318784, -3.8786374837024673, -1.093246567220189, 0.8637665030753502, -0.8562436621395488, -1.0769818771190467, -1.389

Рис. 10: Задание №3. Работа с векторами

# Выполнение лабораторной работы

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

88]: `show(sort(x))`

```
Any[8, 11, 15, 19, 22, 27, 34, 39, 41, 44, 50, 56, 58, 60, 62, 64, 68, 70, 72, 77, 81, 85, 88, 90, 103, 105, 113, 116, 119, 120, 120, 121, 126, 130, 138, 140, 143, 145, 151, 154, 166, 169, 170, 171, 173, 179, 181, 188, 188, 190, 191, 193, 197, 210, 215, 223, 228, 232, 239, 240, 240, 244, 248, 249, 25, 5, 255, 256, 258, 260, 260, 260, 267, 276, 277, 282, 286, 291, 292, 292, 296, 306, 307, 310, 313, 322, 323, 327, 330, 331, 335, 336, 341, 342, 342, 344, 345, 346, 348, 357, 357, 364, 365, 366, 372, 377, 385, 391, 392, 396, 404, 407, 416, 417, 420, 427, 446, 449, 449, 455, 457, 460, 460, 468, 477, 485, 48, 5, 487, 497, 507, 509, 510, 514, 515, 515, 519, 529, 529, 535, 536, 539, 539, 551, 552, 557, 560, 561, 566, 570, 581, 596, 597, 600, 605, 620, 626, 627, 633, 636, 637, 644, 644, 646, 669, 673, 678, 685, 685, 687, 694, 695, 704, 706, 708, 709, 713, 727, 731, 734, 737, 743, 744, 744, 752, 753, 755, 756, 75, 8, 763, 763, 767, 770, 782, 783, 784, 789, 790, 790, 797, 799, 801, 806, 807, 808, 811, 812, 816, 816, 817, 820, 833, 833, 836, 841, 841, 844, 854, 855, 856, 858, 859, 872, 873, 882, 883, 885, 893, 897, 899, 899, 901, 905, 912, 919, 920, 921, 936, 939, 944, 945, 948, 958, 961, 962, 966, 983, 987, 994, 99, 5, 998]
```

89]: `show(unique(sort(x)))`

```
Any[8, 11, 15, 19, 22, 27, 34, 39, 41, 44, 50, 56, 58, 60, 62, 64, 68, 70, 72, 77, 81, 85, 88, 90, 103, 105, 113, 116, 119, 120, 121, 126, 130, 138, 14, 0, 143, 145, 151, 154, 166, 169, 170, 171, 173, 179, 181, 188, 190, 191, 193, 197, 210, 215, 223, 228, 232, 239, 240, 244, 248, 249, 255, 256, 258, 260, 267, 276, 277, 282, 286, 291, 292, 296, 306, 307, 310, 313, 322, 323, 327, 330, 331, 335, 336, 341, 342, 344, 345, 346, 348, 357, 364, 365, 366, 372, 37, 7, 385, 391, 392, 396, 404, 407, 416, 417, 420, 427, 446, 449, 455, 457, 460, 468, 477, 485, 487, 497, 507, 509, 510, 514, 515, 519, 529, 535, 536, 539, 551, 552, 557, 560, 561, 566, 570, 581, 596, 597, 600, 605, 620, 626, 627, 633, 636, 637, 644, 646, 669, 673, 678, 685, 687, 694, 695, 704, 706, 708, 709, 713, 727, 731, 734, 737, 743, 744, 752, 753, 755, 756, 758, 763, 767, 770, 782, 783, 784, 789, 790, 797, 799, 801, 806, 807, 808, 811, 812, 816, 817, 820, 833, 836, 841, 844, 854, 855, 856, 858, 859, 872, 873, 882, 883, 885, 893, 897, 899, 901, 905, 912, 919, 920, 921, 936, 939, 944, 945, 948, 958, 96, 1, 962, 966, 983, 987, 994, 995, 998]
```

Рис. 11: Задание №3. Работа с векторами



## Задание 4

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

```
1]: squares = [(i)**2 for i in range(1,101)]  
    print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024, 1089, 1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025, 2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 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]
```

Рис. 12: Задание №4

```
[31]: myprime2 = primes(prime(168))
      show(myprime2)

[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, 307, 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, 659, 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]

[32]: prime(89)

[32]: 461

[35]: myprime3 = [prime(i) for i in 89:99]
      show(myprime3)

[461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]

[36]: primes(prime(89),prime(99))

[36]: 11-element Vector{Int64}:
      461
      463
      467
      479
      487
      491
      499
      503
      509
      521
      523
```

Рис. 13: Задание №5. Работа с пакетом Primes

## Задание №6

Вычислите следующие выражения:

```
1]: sum = 0
for i in 10:100
    sum = sum + (i^3 + 4*i^2)
end
print(sum)

26852735
```

```
4]: M = 25
sum = 0
for i in 1:M
    sum = sum + (2^i/i + 3^i/i^2)
end
print(sum)

2.1291704368143802e9
```

```
8]: N = 38
sum = 1
a_n = 1
for i in 2:2:N
    a_n *= i/(i+1)
    sum += a_n
end
print(sum)

6.976346137897618
```

Рис. 14: Задание №6

В результате выполнения данной лабораторной работы я изучила несколько структур данных, реализованных в Julia, научилась применять их и операции над ними для решения задач.

1. JuliaLang [Электронный ресурс]. 2024 JuliaLang.org contributors. URL: <https://julialang.org/> (дата обращения: 11.10.2024).
2. Julia 1.11 Documentation [Электронный ресурс]. 2024 JuliaLang.org contributors. URL: <https://docs.julialang.org/en/v1/> (дата обращения: 11.10.2024).