

Лабораторная работа № 2

Структуры данных

Беличева Дарья Михайловна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Примеры использования кортежей	7
4.2	Примеры использования словарей	8
4.3	Примеры использования множеств	9
4.4	Примеры использования массивов	10
4.5	Примеры использования массивов	11
4.6	Примеры использования массивов	12
4.7	Задание №1. Работа с множествами	13
4.8	Задание №2. Примеры операций над множествами элементов раз- ных типов	13
4.9	Задание №3. Работа с массивами	14
4.10	Задание №3. Работа с массивами	14
4.11	Задание №3. Работа с массивами	14
4.12	Задание №3. Работа с массивами	15
4.13	Задание №3. Работа с векторами	15
4.14	Задание №3. Работа с векторами	15
4.15	Задание №3. Работа с векторами	16
4.16	Задание №3. Работа с векторами	16
4.17	Задание №3. Работа с векторами	16
4.18	Задание №3. Работа с векторами	17
4.19	Задание №3. Работа с векторами	17
4.20	Задание №4	17
4.21	Задание №5. Работа с пакетом Primes	18
4.22	Задание №6	18

1 Цель работы

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

2 Задание

1. Используя Jupyter Lab, повторите примеры.
2. Выполните задания для самостоятельной работы.

3 Теоретическое введение

Julia – высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений [1]. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков, однако имеет некоторые существенные отличия.

Для выполнения заданий была использована официальная документация Julia [2].

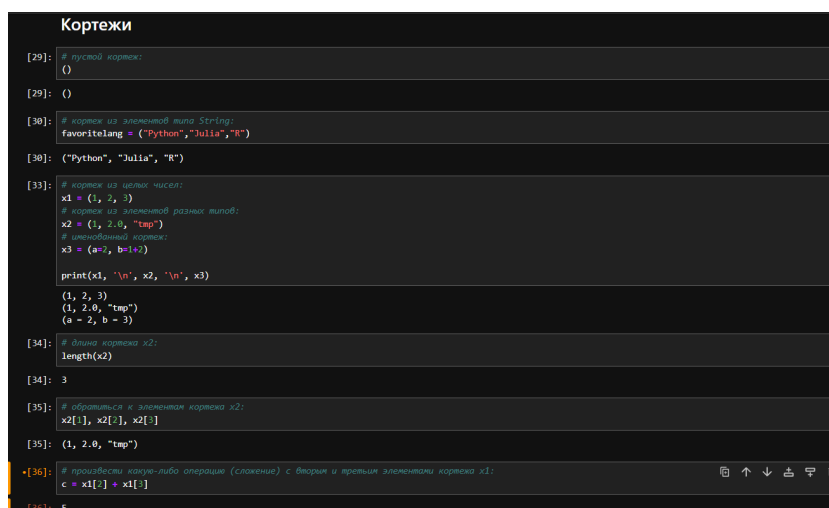
Рассмотрим несколько структур данных, реализованных в Julia. Несколько функций (методов), общих для всех структур данных:

- `isempty()` – проверяет, пуста ли структура данных;
- `length()` – возвращает длину структуры данных;
- `in()` – проверяет принадлежность элемента к структуре;
- `unique()` – возвращает коллекцию уникальных элементов структуры,
- `reduce()` – свёртывает структуру данных в соответствии с заданным бинарным оператором;
- `maximum()` (или `minimum()`) – возвращает наибольший (или наименьший) результат вызова функции для каждого элемента структуры данных.

4 Выполнение лабораторной работы

Для начала выполним примеры из раздела про кортежи (рис. 4.1).

Кортеж (Tuple) – структура данных (контейнер) в виде неизменяемой индексированной последовательности элементов какого-либо типа (элементы индексируются с единицы).



```
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")
      # упорядоченный кортеж:
      x3 = (a=2, b=4+2)

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

      (1, 2, 3)
      (1, 2.0, "tmp")
      (a = 2, b = 6)

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

[34]: 3

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

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

[36]: # произвести какую-либо операцию (сложение) с вторым и третьим элементами кортежа x1:
      c = x1[1] + x1[2]

[36]: 5
```

Рис. 4.1: Примеры использования кортежей

Теперь выполним примеры из раздела про словари (рис. 4.2).

Словарь – неупорядоченный набор связанных между собой по ключу данных.

```
▼ 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. 4.2: Примеры использования словарей

Выполним примеры из раздела про множества (рис. 4.3).

Множество, как структура данных в Julia, соответствует множеству, как математическому объекту, то есть является неупорядоченной совокупностью элементов какого-либо типа. Возможные операции над множествами: объединение, пересечение, разность; принадлежность элемента множеству.


```
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([1,2,2,3,1,2,3,2,1]);
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. 4.3: Примеры использования множеств

Выполним примеры из раздела про массивы (рис. 4.3-4.6).

Массив — коллекция упорядоченных элементов, размещённая в многомерной сетке. Векторы и матрицы являются частными случаями массивов.

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. 4.4: Примеры использования массивов

```

[79]: # многомерный массив $2 \times 3$ (2 строки, 3 столбца) элементов
      # со значениями, случайно распределёнными на интервале [0, 1):
      C = rand(2,3)

[79]: 2x3 Matrix{Float64}:
      0.133083  0.227036  0.691054
      0.548747  0.356681  0.149355

[81]: # трёхмерный массив:
      D = rand(4, 3, 2)

[81]: 4x3x2 Array{Float64, 3}:
     [:, :, 1] =
      0.225261  0.174248  0.699222
      0.389464  0.924887  0.235429
      0.238028  0.330586  0.909227
      0.919654  0.948145  0.326892

     [:, :, 2] =
      0.486618  0.91015  0.328157
      0.487809  0.87158  0.803319
      0.422613  0.760968  0.97449
      0.433011  0.779108  0.13275

[82]: # массив из квадратных корней всех целых чисел от 1 до 10:
      roots = [sqrt(i) for i in 1:10]

[82]: 10-element Vector{Float64}:
      1.0
      1.4142135623730951
      1.7320508075688772
      2.0
      2.23606797749979
      2.449489742783178
      2.6457513110645907
      2.8284271247461903
      3.0
      3.1622776601683795

[83]: # 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]

[83]: 5-element Vector{Int64}:
      3
      27
      75
      147

```

Рис. 4.5: Примеры использования массивов

```

[100]: # транспонирование
b*

[100]: 6x2 adjoint(::Matrix{Int64}) with eltype Int64:
 1  2
 3  4
 5  6
 7  8
 9 10
11 12

[101]: # транспонирование
c = transpose(b)

[101]: 6x2 transpose(::Matrix{Int64}) with eltype Int64:
 1  2
 3  4
 5  6
 7  8
 9 10
11 12

[102]: # массив 10x5 целых чисел в диапазоне [10, 20]:
ar = rand(10:20, 10, 5)

[102]: 10x5 Matrix{Int64}:
15 16 17 13 13
11 11 10 19 10
18 12 15 10 17
18 19 13 15 19
15 12 20 14 19
14 14 10 18 17
12 15 18 16 14
13 11 15 20 10
18 17 18 13 11
19 20 19 13 15

[103]: # выбор всех значений строки в столбце 2:
ar[:, 2]

[103]: 10-element Vector{Int64}:
16
11
12

```

Рис. 4.6: Примеры использования массивов

Перейдем к выполнению заданий.

1. Даны множества: $A = 0, 3, 4, 9$, $B = 1, 3, 4, 7$, $C = 0, 1, 2, 4, 7, 8, 9$. Найдем $P = A \cap B \cup A \cap B \cup A \cap C \cup B \cap C$ (рис. 4.7).

```

Задание 1

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

[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. 4.7: Задание №1. Работа с множествами

2. Приведем свои примеры с выполнением операций над множествами элементов разных типов (рис. 4.8).

```

[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("Elements 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}}

```

Рис. 4.8: Задание №2. Примеры операций над множествами элементов разных типов

3. Создадим массивы разными способами, используя циклы (рис. 4.9-4.19):

[illegible]

Рис. 4.9: Задание №3. Работа с массивами

```
[4]: 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  $y = \exp(\cos(x)) \cdot \cos(x)$  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. 4.10: Задание №3. Работа с массивами

```

136:] vect = []
      for i in 1:134
          1 ← j ← 2
          push!(vect, ((0-1), (0-2)))
      end
      print(vect)

Any{ (0.010000000000000002, 0.2), (1.0000000000000006e-06, 0.001000000000000003), (1.000000000000005e-9, 1.2800000000000006e-5), (1.000000000000008e-2, 1.0240000000000000e-7), (1.0000000000000009e-15, 8.1520000000000005e-10), (1.0000000000000008e-18, 6.5536000000000005e-12), (1.0000000000000012e-21, 5.2423000000000005e-14), (1.0000000000000012e-24, 4.1943840000000005e-16), (1.0000000000000015e-27, 3.3554120000000044e-18), (1.0000000000000017e-30, 2.8352456000000004e-20), (1.0000000000000011e-33, 2.1474836400000004e-22), (1.000000000000002e-36, 1.7177069104000033e-24) }

3.12) вектор с элементами

$$\frac{2^i}{i}, i = 1, 2, \dots, M, M = 25;$$


[78:] vect_0 = [(i-1)/i for i in 1:25]
      show(vect_0)

[2.0, 2.0, 2.6666666666666665, 4.0, 6.4, 10.666666666666666, 18.285714285714285, 32.0, 56.888888888888886, 102.4, 186.18181818181818, 341.3333333333333, 630.1538461538462, 1170.2857142857142, 2184.5333333333333, 4096.0, 7710.117647058823, 14563.555555555555, 27594.105263157893, 52428.8, 99864.38095238095, 190850.18181818182, 364722.0869565217, 699050.6666666666, 1.34217728e6]

3.13) вектор вида ["fn1", "fn2", ..., "fnN"], N = 30:

[31:] vect1 = []
      for i in 1:30
          push!(vect1, "fn"i)
      end
      print(vect1)

Any{ "fn1", "fn2", "fn3", "fn4", "fn5", "fn6", "fn7", "fn8", "fn9", "fn10", "fn11", "fn12", "fn13", "fn14", "fn15", "fn16", "fn17", "fn18", "fn19", "fn20", "fn21", "fn22", "fn23", "fn24", "fn25", "fn26", "fn27", "fn28", "fn29", "fn30" }

```

Рис. 4.11: Задание №3. Работа с массивами

```
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:

x = []
for i in 1:250
    push!(x, rand(0:999))
end
print(x)

Any[865, 447, 231, 344, 371, 315, 496, 440, 901, 88, 809, 809, 273, 378, 840, 767, 731, 984, 156, 756, 269, 403, 879, 426, 208, 187, 817, 995, 358, 777,
131, 88, 657, 834, 359, 849, 34, 442, 117, 489, 905, 936, 107, 69, 80, 16, 70, 3, 465, 117, 865, 431, 996, 25, 947, 282, 682, 658, 870, 169, 400, 910, 3
69, 569, 981, 999, 864, 954, 326, 38, 72, 703, 782, 198, 50, 12, 104, 761, 876, 422, 93, 52, 921, 22, 576, 165, 839, 407, 697, 409, 168, 931, 466, 926,
126, 945, 309, 426, 928, 311, 48, 0, 375, 616, 170, 14, 239, 152, 175, 650, 906, 5, 669, 611, 496, 158, 746, 887, 758, 395, 774, 968, 271, 476, 676, 79
8, 29, 859, 84, 689, 300, 100, 843, 683, 979, 161, 330, 181, 375, 772, 158, 809, 0, 161, 972, 32, 787, 67, 677, 675, 36, 347, 0, 930, 572, 885, 182, 58
2, 208, 985, 834, 622, 672, 311, 470, 437, 850, 890, 331, 29, 480, 112, 300, 884, 841, 835, 170, 405, 890, 304, 331, 122, 317, 256, 790, 306, 108, 684,
904, 156, 629, 533, 18, 613, 291, 383, 445, 730, 55, 486, 35, 685, 682, 844, 55, 401, 666, 984, 977, 326, 924, 417, 536, 546, 38, 434, 295, 483, 107, 66
5, 837, 730, 765, 839, 189, 287, 332, 35, 204, 910, 586, 334, 468, 0, 264, 846, 553, 180, 753, 460, 629, 632, 199, 865, 751, 900, 308, 61, 968, 541]

y = []
for i in 1:250
    push!(y, rand(0:999))
end
print(y)

Any[951, 946, 959, 706, 484, 438, 956, 736, 713, 891, 287, 56, 125, 256, 378, 189, 222, 996, 521, 857, 846, 573, 528, 612, 717, 985, 211, 678, 565, 553,
128, 320, 387, 740, 772, 209, 10, 10, 593, 968, 125, 321, 253, 577, 733, 793, 163, 981, 314, 102, 536, 38, 815, 193, 406, 150, 883, 873, 570, 710, 221,
248, 170, 47, 396, 98, 729, 385, 48, 418, 250, 656, 74, 974, 617, 848, 568, 491, 235, 52, 669, 183, 153, 224, 878, 465, 613, 838, 338, 972, 209, 522, 86
2, 416, 848, 512, 249, 576, 264, 858, 927, 103, 203, 227, 540, 666, 569, 747, 711, 273, 563, 305, 389, 682, 125, 594, 678, 426, 658, 57, 128, 254, 141,
4, 14, 920, 934, 880, 150, 643, 811, 787, 395, 797, 382, 49, 395, 464, 578, 678, 269, 883, 678, 93, 670, 367, 822, 596, 749, 498, 910, 338, 346, 72, 71
1, 122, 289, 671, 722, 52, 117, 349, 688, 521, 298, 140, 571, 217, 758, 814, 214, 273, 490, 150, 393, 710, 971, 350, 551, 883, 364, 198, 116, 652, 962,
841, 680, 379, 35, 567, 381, 669, 628, 378, 139, 157, 263, 63, 779, 804, 867, 661, 923, 424, 254, 214, 415, 267, 670, 797, 137, 766, 108, 288, 114, 173,
343, 249, 914, 93, 599, 487, 705, 355, 84, 979, 550, 157, 747, 760, 765, 433, 667, 102, 405, 445, 715, 564, 790, 626, 114, 636, 422, 126, 129, 847, 144,
608, 821, 684]
```

Рис. 4.12: Задание №3. Работа с массивами

```
- сформируйте вектор (y2 - x1, ..., yn - xn - 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;-400;-112;450;235;-54;-542;580;166;145;11;271;-300;-152;-672;355;-389;-284;-84;-762;-174;-425;-266;-13
1;124;-251;-556;-339;583;-100;-6;-198;452;-145;807;92;-132;-573;-75;887;25;-398;-46;0;142;-5;-230;806;-298;-394;-68;637;468;588;96;231;-466;120;-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;31
2;141;280;377;-203;66;-766;194;-200;-271;481;-322;41;156;129;1400;131;-557;-232;-10;299;705;-489;307;149;905;121;-228;-340;307;275;136;-14;538;50;108;1040;-
911;37;70;58;270;199;-141;151;-252;-632;-290;-690;-538;308;-768;-307;-536;-121;671;-182;-347;-90;523;127;380;339;156;-155;-456;-309;56;-176;712;-600;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, ..., xn-2 + 2xn-1 - xn):

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, 1224, -261, 1547, 1388, 336, 1440, 1469, 1338, 1540, 812, 615, 1278, 1450, 927, 320, 654, -391, 1168, 1244, 1055, 1849, 2234, 1457, 568, 696, 852,
8, 1392, 1719, 418, 404, 1570, 369, -400, 1888, 1115, 444, -56, 1121, 2203, 1145, 540, 1397, 2204, 1672, 1143, 760, 776, 1748, 2425, 947, 1685, 994, 80,
715, 612, 1365, -2645, 836, 958, -211, 1553, 1211, 900, 1659, -115, 986, 1553, 213, 1310, 4839, 1220, 2292, 871, 1865, 1407, -181, 1211, 586, 1417, 40, 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, 980,
-336, 710, 2723, 487, 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, -269, 11159, 1802, 1802, 1730, 1677, 2067, 1513, 1699, 2298, 952, -435, 570, 1315, 974, -161, 766, 2602, 490, 502, 1501, 1122, -213, 1460, 1389, 1739]
```

Рис. 4.13: Задание №3. Работа с векторами

```
- сформируйте вектор (sin(y1)/cos(x2), sin(y2)/cos(x3), ..., sin(yn-1)/cos(xn))

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

[-0.9623545641825781, 1.0272114737656448, 1.0073561631552175, 2.3434076638962242, -1.30775360875392, 2.184835663849713, 1.0084071069229527, 0.1004723151
0774891, 1.0430160728779123, -0.1336538942205555, -1.1073572256540407, 1.2854249007330012, -0.3891499399263308, -0.1340199420565114, 0.37518130648249026,
0.5211174574923891, 0.8192266067120235, -0.7739595953525381, 0.579179483244327, 1.0010269534259488, 1.0483867675849603, 0.9234825652611104, 1.0521006340
006278, -0.8250978811490015, 0.48025128257012945, -0.23338063729593764, 0.8956624065272996, -1.05912745608187594, -1.164995693676858, -0.1194102051669962
7, 0.824235631192083, -1.5891474636969543, 4.402163461591519, 1.842792092140669, 0.00021907845920782221, -0.9719941107813822, 0.6440606324567041, 0.2
92894392570258923, 0.7217155857465929, 1.707443558924714, -1.4309572503119243, 0.4746858562001765, -0.826676029918168, 4.2249204664685673, -0.99960025106
86977, 0.9998761269247262, 1.334623990179329, -4.175700504083413, 0.5278999414515335, -1.1606456192464394, -0.7886930718610737, -7.155133085915265, -0.
835351782871281, -0.01112213498037299, 0.2511274519742956, 4.124678738372672, 0.6323106968070983, 1.1652027119095312, 0.3902726259980732, -16.571822322
053066, 3.9139763145521704, 0.2726781265536376, 6.364726598261218, 0.82827976015391944, -2.767702123370766, -1.191735542929732, -0.5842792032249161,
2.2516010893595585, -1.2122130256441008, 0.76415532772509, 0.76774880651778, 0.8356845390083437, 2.765119354679163, -0.995179018210317, 0.369081946
5396002, 0.7804725597357736, -1.003022441691108, 0.44113038470293574, 1.251384209872519, -0.9601093065112769, 0.166938217673871, -2.40251471675830606, 0.
38031780804736825, -0.923980402603263, 0.9915769726806085, 2.9215697138829064, 0.43360251389751053, 0.413456195951658367, 1.462664099816396, -1.1247961857
7062121, -1.4722612782185576, 0.8341505228230625, 10.653408239567194, -0.6588946697449217, 0.6635332931842858, 1.24967353181302858, -10.79362648171079, 0.
2161895883338852, -0.0346412945474173, 4.644007453106214, -0.5301481819779847, 0.9830537400933795, -0.05944111444465744, 1.2359969293654358, -2.258006
8001851443, -0.9163766829420628, -1.404004371933541, -0.680249339260237, -0.21148759440843556, 6.540723070583012, 0.907116527673966, -0.99704005338513
2, -1.817291349833192, -0.6921076018118204, -1.8786374581202472, -1.8033462662240189, 0.8637623610253509, 8.856243661130548, -1.0263818211109407, -1.80
```

Рис. 4.14: Задание №3. Работа с векторами

```

– вычислите  $\sum_{i=1}^{n-1} \frac{e^{-x_{i+1}}}{x_i + 10}$ 
1]: summa = Base.sum([(exp(-x[i+1]))/(x[i]+10) for i in 1:249])
2]: 7.415576748637432e-5

– выберите элементы вектора y, значения которых больше 600, и выведите на экран определите индексы этих элементов:
1]: y_sorted = sort(y)
for i in y_sorted
    if i > 600
        print(i, "; ")
    end
end
println("\n", "Индексы:")
show(findall(y -> 600))

612; 613; 617; 626; 628; 636; 642; 652; 656; 658; 661; 666; 667; 668; 669; 669; 670; 670; 671; 678; 678; 678; 678; 680; 682; 684; 688; 705; 706; 710; 71
0; 711; 711; 713; 715; 717; 722; 729; 733; 736; 740; 747; 747; 749; 758; 760; 765; 766; 772; 779; 787; 790; 793; 797; 797; 804; 811; 814; 815; 821; 822;
838; 841; 846; 847; 848; 848; 857; 858; 862; 867; 873; 878; 883; 883; 883; 891; 898; 910; 914; 920; 923; 927; 934; 946; 951; 956; 959; 962; 968; 971; 97
2; 974; 979; 981; 985; 996;
Индексы:
[1, 2, 3, 4, 7, 8, 9, 10, 18, 20, 21, 24, 25, 26, 28, 34, 35, 40, 45, 46, 48, 53, 57, 58, 60, 67, 72, 74, 75, 76, 81, 85, 87, 88, 90, 93, 95, 100, 101,
106, 108, 109, 114, 117, 119, 126, 127, 128, 130, 131, 132, 134, 140, 142, 143, 145, 147, 149, 151, 155, 158, 159, 163, 169, 170, 176, 177, 180, 184, 18
5, 186, 187, 192, 193, 199, 200, 201, 202, 203, 209, 210, 212, 219, 223, 226, 229, 230, 231, 233, 237, 239, 240, 242, 246, 248, 249, 250]

```

Рис. 4.15: Задание №3. Работа с векторами

```

– определите значения вектора x, соответствующие значениям вектора y, значения которых больше 600 (под соответствием понимается расположение на
аналогичных индексных позициях):
1]: for i in 1:length(y)
    if (y[i] > 600)
        print(x[i], "; ")
    end
end

438; 473; 781; 896; 839; 56; 183; 183; 774; 377; 94; 161; 510; 52; 260; 595; 899; 978; 828; 927; 370; 303; 538; 642; 765; 760; 147; 769; 718; 699; 525;
609; 650; 334; 916; 527; 890; 157; 786; 930; 727; 20; 312; 214; 351; 262; 88; 356; 446; 363; 159; 77; 141; 676; 336; 235; 112; 358; 999; 915; 237; 4; 63
3; 193; 365; 903; 129; 537; 870; 139; 457; 404; 347; 760; 154; 159; 963; 407; 464; 195; 482; 320; 726; 934; 655; 273; 671; 514; 30; 37; 359; 502; 6
98; 551; 227; 31;

– сформируйте вектор  $(|x_1 - \bar{x}|^{1/2}, |x_2 - \bar{x}|^{1/2}, \dots, |x_n - \bar{x}|^{1/2})$ , где  $\bar{x}$  обозначает среднее значение вектора  $x = (x_1, x_2, \dots, x_n)$ :
2]: using Statistics
3]: vect = [(abs(x[i] - mean(x)))^(1/2) for i in 1:n]
show(vect)

[13.54237793083727, 12.59380800234782, 17.62396096228087, 12.907517189606992, 10.74225302527445, 11.899747896489236, 20.233536517376294, 20.8231601828
3488, 8.242001174092104, 21.73407276031404, 21.12080557247205, 11.535381018446588, 5.440095370836696, 16.114116949693607, 19.328838537960247, 14.812022
144190847, 11.419106795191995, 16.32164207425221, 8.520798084686668, 17.84365433424443, 19.83421286565212, 17.3090872765460314, 19.662248091202592, 21.41
0184492432567, 13.550055350440454, 12.025140331821495, 17.98899663683386, 14.94643770267685, 18.963227573385286, 20.11477069220527, 20.745987563864006,
18.06643296281809, 7.912268953972684, 20.312656153245936, 21.227435078218942, 21.71165585578401, 14.580946471337175, 21.693409137339387, 15.152689530245
119, 16.28087980807955, 21.34080837208017, 13.914125438920081, 14.040768681369812, 17.20816283627732, 18.61730377901181, 19.990097540536376, 12.43261809
6869787, 12.42561806060782, 21.785407960379568, 15.940790236942966, 18.89480743200676, 17.82732074325953, 13.401641680479565, 8.209586648432641, 12.14933
869274086, 13.970411926967956, 8.449615375861793, 20.726890746081526, 15.407660432395309, 9.879473670191139, 14.092693142192516, 20.28289201051116, 2
0.435361508913903, 15.830476935329523, 15.63169864106256, 3.550211261319527, 20.213955575295003, 2.1456933611306126, 18.751106634009963, 19.611323259790

```

Рис. 4.16: Задание №3. Работа с векторами

```

– определите, сколько элементов вектора y отстоят от максимального значения не более, чем на 200:
1]: length([i for i in y if abs(maximum(y) - i) <= 200])
2]: 44

– определите, сколько чётных и нечётных элементов вектора x:
1]: print("Количество четных элементов вектора x: ", count(true for i in x if i%2 == 0))
print("\n", "Количество нечетных элементов вектора x: ", count(true for i in x if i%2 != 0))

Количество четных элементов вектора x: 123
Количество нечетных элементов вектора x: 127

– определите, сколько элементов вектора x кратны 7:
1]: print("Количество элементов вектора x кратны 7: ", count(true for i in x if i%7 == 0))

Количество элементов вектора x кратны 7: 42

– отсортируйте элементы вектора x в порядке возрастания элементов вектора y:
1]: print(x)

Any[673, 331, 179, 323, 605, 348, 890, 56, 420, 962, 936, 357, 460, 228, 116, 709, 620, 756, 417, 808, 883, 190, 103, 948, 306, 345, 166, 713,
920, 816, 427, 77, 39, 961, 277, 19, 260, 755, 945, 296, 687, 191, 143, 90, 644, 644, 15, 744, 817, 811, 310, 557, 342, 685, 561, 60, 727, 39,
1, 72, 239, 734, 477, 81, 485, 138, 105, 404, 782, 244, 121, 637, 151, 210, 994, 560, 50, 446, 694, 145, 258, 416, 292, 193, 752, 223, 854, 8,
4, 919, 249, 276, 529, 820, 215, 377, 357, 346, 552, 260, 566, 140, 636, 708, 539, 509, 856, 267, 342, 70, 886, 539, 307, 596, 905, 763, 120,
11, 515, 385, 330, 844, 120, 685, 455, 706, 169, 515, 232, 600, 197, 154, 88, 457, 841, 407, 514, 841, 248, 872, 255, 789, 34, 744, 882, 597,
60, 313, 170, 770, 366, 695, 885, 327, 753, 70, 807, 646, 816, 206, 704, 799, 256, 507, 364, 893, 335, 873, 240, 119, 833, 60, 670, 737, 763,
407, 836, 790, 62, 570, 336, 365, 995, 27, 767, 921, 812, 255, 43, 988, 551, 669, 322, 958, 983, 240, 536, 510, 372, 260, 535, 966, 485, 801,
64, 391, 188, 743, 859, 8, 282, 171, 797, 897, 341, 449, 126, 858, 944, 633, 344, 627, 449, 912, 292, 181, 519, 581, 758, 783, 529, 855, 58,
68, 626, 704]

1]: print(y)

Any[951, 946, 959, 706, 484, 438, 956, 736, 713, 891, 287, 56, 125, 256, 378, 189, 222, 996, 521, 857, 846, 573, 528, 612, 717, 985, 211, 678,
128, 320, 387, 740, 772, 209, 10, 10, 593, 968, 125, 321, 253, 577, 733, 793, 163, 981, 314, 102, 536, 38, 815, 193, 406, 150, 883, 873, 570,
248, 170, 47, 396, 98, 729, 385, 48, 418, 250, 656, 74, 974, 617, 848, 568, 491, 235, 52, 669, 183, 153, 224, 878, 465, 613, 838, 338, 972, 2,
2, 416, 848, 512, 249, 576, 264, 858, 927, 103, 293, 227, 540, 666, 568, 747, 711, 273, 563, 305, 389, 682, 125, 594, 678, 426, 658, 57, 128,
4, 18, 920, 934, 898, 150, 643, 611, 787, 395, 797, 382, 49, 395, 464, 578, 678, 269, 883, 678, 93, 670, 367, 822, 596, 749, 490, 516, 338, 3,
1, 113, 380, 721, 333, 63, 612, 440, 692, 531, 209, 149, 321, 312, 703, 814, 313, 323, 409, 169, 202, 216, 321, 359, 513, 892, 264, 108, 114,

```

Рис. 4.17: Задание №3. Работа с векторами


```

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

[143]: reverse(last(sort(x),10))

[143]: 10-element Vector{Any}:
 999
 996
 995
 985
 984
 984
 981
 979
 977
 972

```

Рис. 4.18: Задание №3. Работа с векторами

```

– 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, 59, 60, 62, 64, 68, 70, 72, 77, 81, 85, 88, 90, 103, 105, 113, 116, 119, 120, 129, 131, 136, 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, 244, 248, 249, 25, 255, 256, 258, 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, 407, 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, 649, 673, 678, 685, 685, 687, 684, 695, 704, 706, 709, 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, 59, 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, 70, 9, 713, 727, 731, 734, 737, 743, 744, 752, 753, 755, 756, 750, 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, 859, 872, 873, 882, 883, 885, 893, 897, 899, 901, 905, 912, 919, 920, 921, 936, 939, 944, 945, 948, 958, 961, 962, 966, 983, 987, 994, 99, 5, 998]

```

Рис. 4.19: Задание №3. Работа с векторами

4. Создадим массив `squares`, в котором будут храниться квадраты всех целых чисел от 1 до 100 (рис. 4.20).

```

Задание 4

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

1]: squares = [(i)^2 for i in 1:100]
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. 4.20: Задание №4

5. Подключим пакет `Primes` (функции для вычисления простых чисел). Сгенерируем массив `myprimes`, в котором будут храниться первые 168 простых чисел. Определим 89-е наименьшее простое число. Получим срез массива с 89-го до 99-го элемента включительно, содержащий наименьшие простые числа (рис. 4.21).

```
[31]: myprime2 = primes(prime(100))
      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. 4.21: Задание №5. Работа с пакетом Primes

6. Вычислим следующие выражения (рис. 4.22).

```
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)

2.1291704368143882e9

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. 4.22: Задание №6

5 Выводы

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

Список литературы

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).