Отчёт по лабораторной работе №2

Структуры данных

Ким Реачна

Содержание

Список иллюстраций

1 Цель работы

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

2 Выполнение лабораторной работы

2.1 Кортежи

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

Синтаксис определения кортежа: `(element1, element2, ...)`

Рис. 2.1: Примеры кортежей

```
[6]: # Примеры операций над кортежами:
     # длина кортежа x2:
     length(x2)

[6]: 3

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

[7]: (1, 2.0, "tmp")

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

[8]: 5

[9]: # обращение к элементам именованного кортежа x3:
     x3.a, x3.b, x3[2]

[9]: (2, 3, 3)

[10]: # проверка вхождения элементов tmp и 0 в кортеж x2
      # (два способа обращения к методу in()):
      in("tmp", x2), 0 in x2

[10]: (true, false)
```

Рис. 2.2: Примеры операций над кортежами

2.2 Словари

Словарь — неупорядоченный набор связанных между собой по ключу данных.

Синтаксис определения словаря: `Dict(key1 => value1, key2 => value2, ...)`

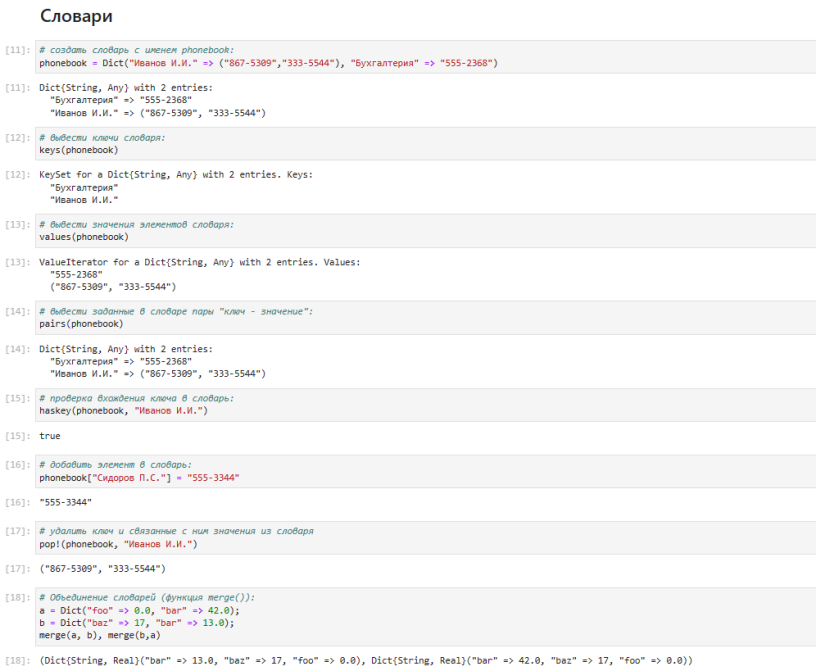


Рис. 2.3: Примеры словарей и операций над словарями

2.3 Множества

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.

Синтаксис определения множества: `Set([itr])`, где `itr` — набор значений, сгенерированных данным итерируемым объектом или пустое множество.

Множества

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

[19]: Set{Int64} with 4 elements:
        5
        4
        3
        1

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

[20]: Set{Char} with 5 elements:
        'a'
        'd'
        'r'
        'k'
        'b'

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

[21]: false

[22]: S3 = Set([1,2,2,3,1,2,3,2,1]);
      S4 = Set([2,3,1]);
      issetequal(S3,S4)

[22]: true

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

[23]: Set{Int64} with 4 elements:
        4
        2
        3
        1
```

Рис. 2.4: Примеры и операции над множествами

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

[24]: Set{Int64} with 2 elements:
        2
        1

[25]: # разность множеств:
      E = setdiff(S3,S1)

[25]: Set{Int64} with 1 element:
        3

[26]: # проверка вхождения элементов одного множества в другое:
      issubset(S1,S4)

[26]: true

[27]: # добавление элемента в множество:
      push!(S4, 99)

[27]: Set{Int64} with 4 elements:
        2
        99
        3
        1

[28]: # удаление последнего элемента множества:
      pop!(S4)

[28]: 2
```

Рис. 2.5: Примеры и операции над множествами

2.4 Массивы

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

Общий синтаксис одномерных массивов:

```
array_name_1 = [element1, element2, ...]
array_name_2 = [element1 element2 ...]
```

Некоторые операции для работы с массивами:

– `length(A)` — число элементов массива A;

– `ndims(A)` — число размерностей массива A;

– `size(A)` — кортеж размерностей массива A;

– `size(A, n)` — размерность массива A в заданном направлении;

– `copy(A)` — создание копии массива A;

– `ones(), zeros()` — создать массив с единицами или нулями соответственно;

– `fill(value,array_name)` — заполнение массива заранее определенным значением;

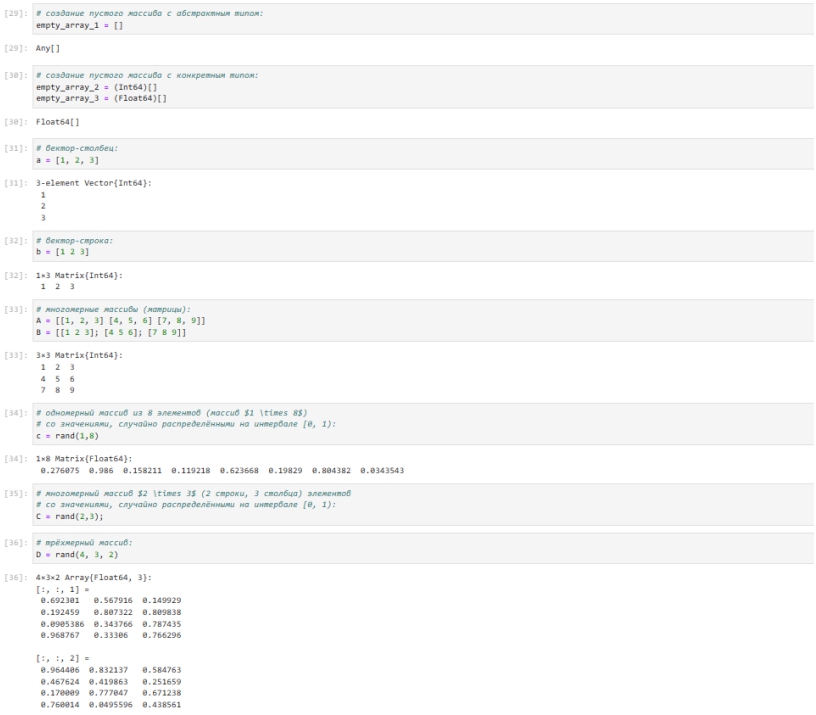
– `sort()` — сортировка элементов;

– `collect()` — вернуть массив всех элементов в коллекции или итераторе;

– `reshape()` — изменение размера массива;

– `transpose()` — транспонирование массива;

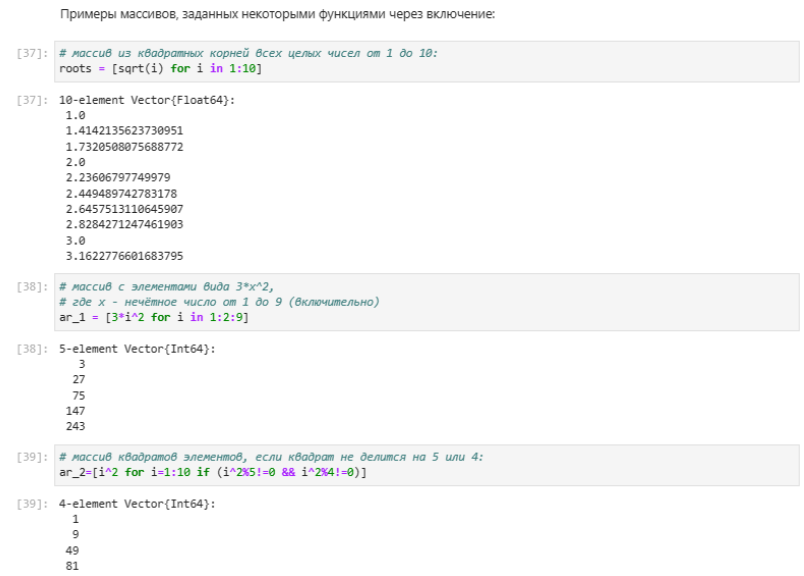
Примеры массивов:



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

[29]: Any[]

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

[30]: Float64[]

[31]: # вектор-столбец:
      a = [1, 2, 3]

[31]: 3-element Vector{Int64}:
       1
       2
       3

[32]: # вектор-строка:
      b = [1 2 3]

[32]: 1×3 Matrix{Int64}:
       1  2  3

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

[33]: 3×3 Matrix{Int64}:
       1  2  3
       4  5  6
       7  8  9

[34]: # одномерный массив из 8 элементов (массив $1 \times 8$)
      # со значениями, случайно распределёнными на интервале [0, 1):
      c = rand(1,8)

[34]: 1×8 Matrix{Float64}:
       0.276075  0.986  0.158211  0.119218  0.623668  0.19829  0.804382  0.0343543

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

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

[36]: 4×3×2 Array{Float64, 3}:
      [:, :, 1] =
       0.692301   0.567916  0.149929
       0.192459   0.807322  0.809838
       0.0905386  0.343766  0.787435
       0.968767   0.33306   0.766296

      [:, :, 2] =
       0.964406  0.832137   0.584763
       0.467624  0.419863   0.251659
       0.170009  0.777047   0.671238
       0.760014  0.0495596  0.438561
```

Рис. 2.6: Примеры массивов

Примеры массивов, заданных некоторыми функциями через включение:



```
Примеры массивов, заданных некоторыми функциями через включение:

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

[37]: 10-element Vector{Float64}:
       1.0
       1.4142135623730951
       1.7320508075688772
       2.0
       2.23606797749979
       2.449489742783178
       2.6457513110645907
       2.8284271247461903
       3.0
       3.1622776601683795

[38]: # 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]

[38]: 5-element Vector{Int64}:
          3
         27
         75
        147
        243

[39]: # 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)]

[39]: 4-element Vector{Int64}:
        1
        9
       49
       81
```

Рис. 2.7: Примеры массивов, заданных некоторыми функциями через включение

Некоторые операции для работы с массивами:

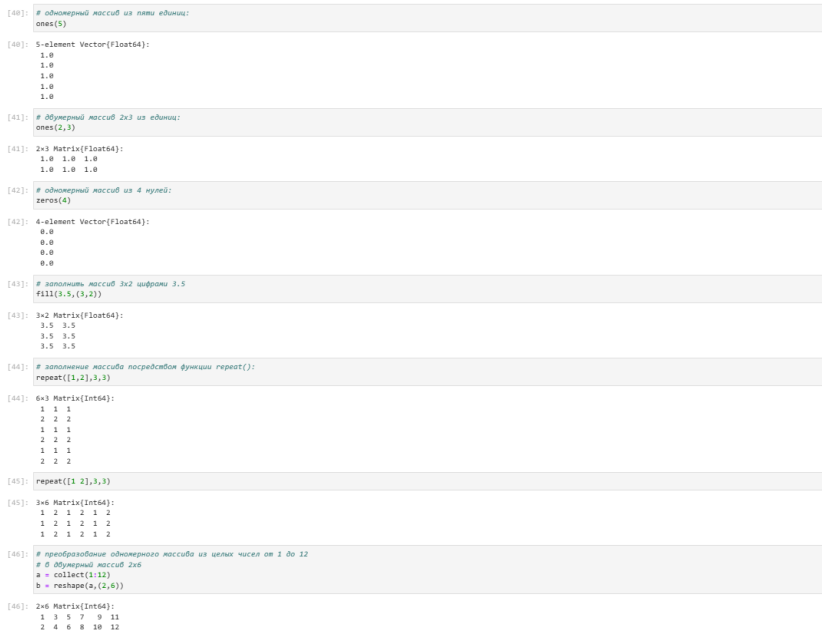


Рис. 2.8: Примеры операций над массивами

```
[47]: # транспонирование
      b'

[47]: 6×2 adjoint(::Matrix{Int64}) with eltype Int64:
       1   2
       3   4
       5   6
       7   8
       9  10
      11  12

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

[48]: 6×2 transpose(::Matrix{Int64}) with eltype Int64:
       1   2
       3   4
       5   6
       7   8
       9  10
      11  12

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

[49]: 10×5 Matrix{Int64}:
      12  12  14  14  14
      20  17  11  16  13
      19  17  10  19  11
      16  13  10  10  17
      16  17  16  17  14
      20  15  16  15  16
      16  12  13  10  12
      18  11  20  15  13
      14  13  16  15  13
      11  16  13  15  19

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

[50]: 10-element Vector{Int64}:
      12
      17
      17
      13
      17
      15
      12
      11
      13
      16

[51]: # выбор всех значений в столбцах 2 и 5:
      ar[:, [2, 5]]

[51]: 10×2 Matrix{Int64}:
      12  14
      17  13
      17  11
      13  17
      17  14
      15  16
      12  12
```

Рис. 2.9: Примеры операций над массивами

```
[52]: # все значения строк в столбцах 2, 3 и 4:
      ar[:, 2:4]

[52]: 10×3 Matrix{Int64}:
      12  14  14
      17  11  16
      17  10  19
      13  10  10
      17  16  17
      15  16  15
      12  13  10
      11  20  15
      13  16  15
      16  13  15

[53]: # значения в строках 2, 4, 6 и в столбцах 1 и 5:
      ar[[2, 4, 6], [1, 5]]

[53]: 3×2 Matrix{Int64}:
      20  13
      16  17
      20  16

[54]: # значения в строке 1 от столбца 3 до последнего столбца:
      ar[1, 3:end]

[54]: 3-element Vector{Int64}:
      14
      14
      14

[55]: # сортировка по столбцам:
      sort(ar,dims=1)

[55]: 10×5 Matrix{Int64}:
      11  11  10  10  11
      12  12  10  10  12
      14  12  11  14  13
      16  13  13  15  13
      16  13  13  15  13
      16  15  14  15  14
      18  16  16  15  14
      19  17  16  16  16
      20  17  16  17  17
      20  17  20  19  19

[56]: # сортировка по строкам:
      sort(ar,dims=2)

[56]: 10×5 Matrix{Int64}:
      12  12  14  14  14
      11  13  16  17  20
      10  11  17  19  19
      10  10  13  16  17
      14  16  16  17  17
      15  15  16  16  20
      10  12  12  13  16
      11  13  15  18  20
      13  13  14  15  16
      11  13  15  16  19
```

Рис. 2.10: Примеры операций над массивами

```
[57]: # поэлементное сравнение с числом
      # (результат - массив логических значений):
      ar .> 14

[57]: 10×5 BitMatrix:
       0  0  0  0  0
       1  1  0  1  0
       1  1  0  1  0
       1  0  0  0  1
       1  1  1  1  0
       1  1  1  1  1
       1  0  0  0  0
       1  0  1  1  0
       0  0  1  1  0
       0  1  0  1  1

[58]: # возврат индексов элементов массива, удовлетворяющих условию:
      findall(ar .> 14)

[58]: 26-element Vector{CartesianIndex{2}}:
       CartesianIndex(2, 1)
       CartesianIndex(3, 1)
       CartesianIndex(4, 1)
       CartesianIndex(5, 1)
       CartesianIndex(6, 1)
       CartesianIndex(7, 1)
       CartesianIndex(8, 1)
       CartesianIndex(2, 2)
       CartesianIndex(3, 2)
       CartesianIndex(5, 2)
       CartesianIndex(6, 2)
       CartesianIndex(10, 2)
       CartesianIndex(5, 3)
       CartesianIndex(6, 3)
       CartesianIndex(8, 3)
       CartesianIndex(9, 3)
       CartesianIndex(2, 4)
       CartesianIndex(3, 4)
       CartesianIndex(5, 4)
       CartesianIndex(6, 4)
       CartesianIndex(8, 4)
       CartesianIndex(9, 4)
       CartesianIndex(10, 4)
       CartesianIndex(4, 5)
       CartesianIndex(6, 5)
       CartesianIndex(10, 5)
```

Рис. 2.11: Примеры операций над массивами

2.5 Задания для самостоятельного выполнения

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

```
[60]: # Задания 1
      # Определение множеств
      A = Set([0, 3, 4, 9])
      B = Set([1, 3, 4, 7])
      C = Set([0, 1, 2, 4, 7, 8, 9])

      # Вычисление пересечений и объединений
      P = union(intersect(A, B), intersect(A, B), intersect(A, C), intersect(B, C))

      # вывод
      println("P: ",P)

      P: Set([0, 4, 7, 9, 3, 1])
```

Рис. 2.12: Задача с множествами

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

```
[61]: # Задания 2
      # Создание множество
      set1 = Set([1, "hello", 3.14, [1, 2, 3]])
      set2 = Set([1, 2, "world", 3.14])

      intersection_set = intersect(set1, set2)
      union_set = union(set1, set2)
      isequal_set = issetequal(set1, set2)
      sub_set = issubset(set1, set2)

      println("Множество 1 : ", set1)
      println("Множество 2 : ", set2)
      println("Пересечение : ", intersection_set)
      println("Объединение : ", union_set)
      println("Эквивалентности множеств : ", isequal_set)
      println("Подмножество : ", sub_set)

Множество 1 : Set(Any["hello", 3.14, [1, 2, 3], 1])
Множество 2 : Set(Any[2, 3.14, 1, "world"])
Пересечение : Set(Any[3.14, 1])
Объединение : Set(Any[2, "hello", 3.14, [1, 2, 3], 1, "world"])
Эквивалентности множеств : false
Подмножество : false
```

Рис. 2.13: Задача с множествами

3. Создайте разными способами:

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

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

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

```
[62]: # Задания 3
      # Задания 3.1 до 3.3
      N = 25
      array_3_1 = collect(1:N)
      array_3_2 = reverse(collect(1:N))
      array_3_3 = [1:N; N-1:-1:1]

      println("Задания 3.1 : ", array_3_1)
      println("Задания 3.2 : ", array_3_2)
      println("Задания 3.3 : ", array_3_3)

Задания 3.1 : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 2
4, 25]
Задания 3.2 : [25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3,
2, 1]
Задания 3.3 : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 2
4, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Рис. 2.14: Задача с множествами 3.1 до 3.3

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;

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

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

[63]:
```
# Задания 3.4 до 3.9
tmp = [4, 6, 3]

array_3_5 = fill(tmp[1], 10)
array_3_6 = repeat([tmp], 10)
array_3_7 = vcat(fill(tmp[1], 11), fill(tmp[2], 10), fill(tmp[3], 10))
array_3_8 = vcat(fill(tmp[1], 10), fill(tmp[2], 20), fill(tmp[3], 30))
array_3_9 = repeat(2 .^ tmp, inner = [1, 1, 4])
count_six = count(x -> x == '6', string(array_3_9))

println("Задания 3.5 : ", array_3_5)
println("Задания 3.6 : ", array_3_6)
println("Задания 3.7 : ", array_3_7)
println("Задания 3.8 : ", array_3_8)
println("Задания 3.9 : ", array_3_9)
println("Количество цифр 6 в результате 3.9 : ", count_six)
```

```
Задания 3.5 : [4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
Задания 3.6 : [[4, 6, 3], [4, 6, 3], [4, 6, 3], [4, 6, 3], [4, 6, 3], [4, 6, 3], [4, 6, 3], [4, 6, 3], [4, 6, 3], [4, 6, 3]]
Задания 3.7 : [4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3]
Задания 3.8 : [4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
Задания 3.9 : [16; 64; 8;;; 16; 64; 8;;; 16; 64; 8;;; 16; 64; 8]
Количество цифр 6 в результате 3.9 : 8
```

[64]:
```
# Задания 3.10
using Statistics;
x_values = 3:0.1:6
y_values = [exp(x) * cos(x) for x in x_values]
mean_y = mean(y_values)
println("Задания 3.10. Среднее значение y : ", mean_y)
```

```
Задания 3.10. Среднее значение y : 53.11374594642971
```

Рис. 2.15: Задача с множествами 3.4 до 3.10

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

```
[65]: # # Задания 3.11
x_vector = [0.1^i for i in 3:3:36]
y_vector = [0.2^j for j in 1:3:34]
result_3_11 = [(x, y) for x in x_vector, y in y_vector]
println("Задания 3.11 : ", result_3_11)
```

```
Задания 3.11 : [(0.0010000000000000002, 0.2) (0.0010000000000000002, 0.0016000000000000003) (0.0010000000000000002, 1.2800000000000006e-5) (0.0010000000000000002, 1.0240000000000006e-7) (0.0010000000000002, 8.192000000000005e-10) (0.0010000000000000002, 6.553600000000055e-12) (0.0010000000000002, 5.2428800000000056e-14) (0.0010000000000000002, 4.194304000000005e-16) (0.0010000000000000002, 3.3554432000000044e-18) (0.0010000000000000002, 2.684354560000004e-20) (0.0010000000000000002, 2.147483648000004e-22) (0.0010000000000000002, 1.7179869184000035e-24); (1.0000000000000004e-6, 0.2) (1.0000000000000004e-6, 0.0016000000000000003) (1.0000000000000004e-6, 1.2800000000000006e-5) (1.0000000000000004e-6, 1.0240000000000006e-7) (1.0000000000000004e-6, 8.192000000000005e-10) (1.0000000000000004e-6, 6.553600000000055e-12) (1.0000000000000004e-6, 5.2428800000000056e-14) (1.0000000000000004e-6, 4.194304000000005e-16) (1.0000000000000004e-6, 3.3554432000000044e-18) (1.0000000000000004e-6, 2.684354560000004e-20) (1.0000000000000004e-6, 2.147483648000004e-22) (1.0000000000000004e-6, 1.7179869184000035e-24); (1.0000000000000005e-9, 0.2) (1.0000000000000005e-9, 0.0016000000000000003) (1.0000000000000005e-9, 1.2800000000000006e-5) (1.0000000000000005e-9, 1.0240000000000006e-7) (1.0000000000000005e-9, 8.192000000000005e-10) (1.0000000000000005e-9, 6.553600000000055e-12) (1.0000000000000005e-9, 5.2428800000000056e-14) (1.0000000000000005e-9, 4.194304000000005e-16) (1.0000000000000005e-9, 3.3554432000000044e-18) (1.0000000000000005e-9, 2.684354560000004e-20) (1.0000000000000005e-9, 2.147483648000004e-22) (1.0000000000000005e-9, 1.7179869184000035e-24); (1.0000000000000008e-12, 0.2) (1.0000000000000008e-12, 0.0016000000000000003) (1.0000000000000008e-12, 1.2800000000000006e-5) (1.0000000000000008e-12, 1.0240000000000006e-7) (1.0000000000000008e-12, 8.192000000000005e-10) (1.0000000000000008e-12, 6.553600000000055e-12) (1.0000000000000008e-12, 5.2428800000000056e-14) (1.0000000000000008e-12, 4.194304000000005e-16) (1.0000000000000008e-12, 3.3554432000000044e-18) (1.0000000000000008e-12, 2.684354560000004e-20) (1.0000000000000008e-12, 2.147483648000004e-22) (1.0000000000000008e-12, 1.7179869184000035e-24); (1.0000000000000009e-15, 0.2) (1.0000000000000009e-15, 0.0016000000000000003) (1.0000000000000009e-15, 1.2800000000000006e-5) (1.0000000000000009e-15, 1.0240000000000006e-7) (1.0000000000000009e-15, 8.192000000000005e-10) (1.0000000000000009e-15, 6.553600000000055e-12) (1.0000000000000009e-15, 5.2428800000000056e-14) (1.0000000000000009e-15, 4.194304000000005e-16) (1.0000000000000009e-15, 3.3554432000000044e-18) (1.0000000000000009e-15, 2.684354560000004e-20) (1.0000000000000009e-15, 2.147483648000004e-22) (1.0000000000000009e-15, 1.7179869184000035e-24); (1.0000000000000008e-18, 0.2) (1.0000000000000008e-18, 0.0016000000000000003) (1.0000000000000008e-18, 1.2800000000000006e-5) (1.0000000000000008e-18, 1.0240000000000006e-7) (1.0000000000000008e-18, 8.192000000000005e-10) (1.0000000000000008e-18, 6.553600000000055e-12) (1.0000000000000008e-18, 5.2428800000000056e-14) (1.0000000000000008e-18, 4.194304000000005e-16) (1.0000000000000008e-18, 3.3554432000000044e-18) (1.0000000000000008e-18, 2.684354560000004e-20) (1.0000000000000008e-18, 2.147483648000004e-22) (1.0000000000000008e-18, 1.7179869184000035e-24); (1.0000000000000012e-21, 0.2) (1.0000000000000012e-21, 0.0016000000000000003) (1.0000000000000012e-21, 1.2800000000000006e-5) (1.0000000000000012e-21, 1.0240000000000006e-7) (1.0000000000000012e-21, 8.192000000000005e-10) (1.0000000000000012e-21, 6.553600000000055e-12) (1.0000000000000012e-21, 5.2428800000000056e-14) (1.0000000000000012e-21, 4.194304000000005e-16) (1.0000000000000012e-21, 3.3554432000000044e-18) (1.0000000000000012e-21, 2.684354560000004e-20) (1.0000000000000012e-21, 2.1474
```

Рис. 2.16: Задача с множествами 3.11

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

3.13. вектор вида $("fn1", "fn2", ..., "fnN"), N = 30$

Рис. 2.17: Задача с множествами 3.12 и 3.13

3.14. векторы $x = (x_1, x_2, ..., x_n)$ и $y = (y_1, y_2, ..., y_n)$ целочисленного типа длины $n = 250$ как случайные выборки из совокупности $0, 1, ..., 999$; на его основе:

3.14.1. сформируйте вектор $(y_2 - x_1, ..., y_n - x_{n-1})$

3.14.2. сформируйте вектор $(x_1 + 2x_2 - x_3, x_2 + 2x_3 - x_4, ..., x_{n-2} + 2x_{n-1} - x_n)$

```
[68]: # Задания 3.14
      # Создание векторов x и y целочисленного типа
      n = 250
      x = rand(0:999, n)
      y = rand(0:999, n)

      # Формирование векторов
      vector1 = y[2:end] .- x[1:end-1]
      vector2 = [x[i] + 2x[i+1] - x[i+2] for i in 1:n-2]
      vector3 = [sin(y[i]) / cos(x[i+1]) for i in 1:n-1]
      sum_expo = sum([exp(-x[i+1]) / (x[i] +10)] for i in 1:n-1)

[68]: 1-element Vector{Float64}:
       1.4713467696164503e-5

[69]: # Вывод
      println("Задания 3.14.1 : ", vector1)

      Задания 3.14.1 : [55, -123, -72, -83, 668, -58, 58, -446, -284, -925, 313, -65, 640, -62, 2, 514, -162, 419, 395, 890, 76, -529, -
      277, 586, -179, 370, 154, -621, 485, 45, 834, -82, 141, -490, -190, -370, -902, -487, 158, 822, -59, 377, -337, -30, -317, -80, -1
      1, 32, 124, -240, -431, -558, -405, -301, 48, 21, 93, -1, 728, 395, -63, -129, -356, -691, -368, 328, 431, 57, 228, -148, 698, -15
      1, -170, 10, -199, 296, -454, -657, 325, -83, -464, -403, 222, 321, 692, 80, -170, 181, 682, -47, -266, -22, -405, -130, -754, -27
      1, -636, -98, 554, -95, 250, -799, 17, -291, -251, 7, 664, -150, 465, 83, -227, 486, 215, 62, 85, -332, 596, -105, -389, 419, 441,
      -763, 881, 54, -470, 180, -769, -7, -478, -397, 360, -17, -48, 489, 53, -23, 316, -60, 238, 512, 529, -522, 127, 46, -670, -410,
      4, 153, 231, 51, 16, 30, -332, 265, -550, -800, 313, 323, -265, -279, -645, -597, -191, -632, -258, 325, -228, 179, 501, 148, -45
      9, 556, 422, 39, 117, -156, -270, -78, 231, -652, 635, -768, -262, 338, 338, -601, 751, -175, 272, 513, -213, -333, -444, -83, 32
      3, 123, -66, 421, 7, 713, -14, 393, 516, -130, -9, 344, -305, -706, -209, 194, -301, 786, 100, -490, -37, -587, 187, -317, 590, -1
      97, -596, 306, -485, -669, 230, -259, 525, 82, -617, 669, 418, 285, 573, 212, -269, -12, 89, 339, 585, -735, -387, -95, -211, 88,
      270, -103, 135, -939, -99]

[70]: println("Задания 3.14.2 : ", vector2)

      Задания 3.14.2 : [797, 559, 2126, 134, 1467, 677, 533, 868, 2205, 995, 1986, 1245, 636, 99, 477, 539, 608, 1311, 377, -562, 985, 1
      728, 264, 1836, 1200, 952, 1353, 707, 2256, 313, 1943, 458, 894, 1709, 873, 1614, 2050, 2573, 450, 735, 124, 165, 819, 2320, 1375,
      492, 1469, 459, 795, 1566, 2033, 1946, 1534, 1973, 1790, -148, 1839, 965, -174, 471, 1420, 1678, 2177, 1269, 995, 415, 1, 570, 159
      8, 997, 580, 157, 726, 2237, -10, 1103, 2009, 1467, -309, 1281, 1540, 1446, 1385, -110, 625, 2124, 1078, -122, 483, 1727, 2053, 15
      82, -156, 1625, 799, 1159, 2051, 1106, 787, -336, 1167, 1584, 1676, 1319, 1727, 56, 944, 1335, -122, 1077, 573, 1279, 1231, 1500,
      1830, 1150, 244, 1012, 73, -97, 2129, 411, 1006, 1277, 817, 1758, 1015, 1363, 2476, 238, 1839, 1451, -670, 1351, 1476, 744, 1916,
      1259, 100, -146, 1719, 1027, 173, 1645, 2241, 966, 638, -136, 1070, 1220, 961, 1687, 597, 1086, 2807, 562, 317, 957, 1603, 1724, 1
      782, 467, 1293, 1476, 373, 1749, 1511, 841, 460, 2136, 1262, 300, 1476, 1422, 1542, 1653, 933, 1195, 1874, 436, 1864, 974, 1316, 3
      55, 1973, 795, 453, 130, -93, 1155, 429, 1953, 1652, 25, 356, 634, 1199, 1067, 432, 575, 144, -155, 2017, 815, 558, 476, 1075, 277
      8, 405, 1217, 115, 339, 1516, 2084, 2270, 190, 1905, 391, 752, 1835, 396, 1002, 2416, 67, 1580, 1723, -240, 1554, 999, 785, 362, 7
      17, 55, 209, 852, 1477, 1320, 192, 1293, 1679, 468, 19, 1081, 303, 1659, 505, 1986, 517]
```

Рис. 2.18: Задача с множествами 3.14, 3.14.1, 3.14.2

3.14.3. сформируйте вектор $(\frac{sin(y_1)}{cos(x_2)}, \frac{sin(y_2)}{cos(x_3)}, ..., \frac{sin(y_{n-1})}{cos(x_n)})$

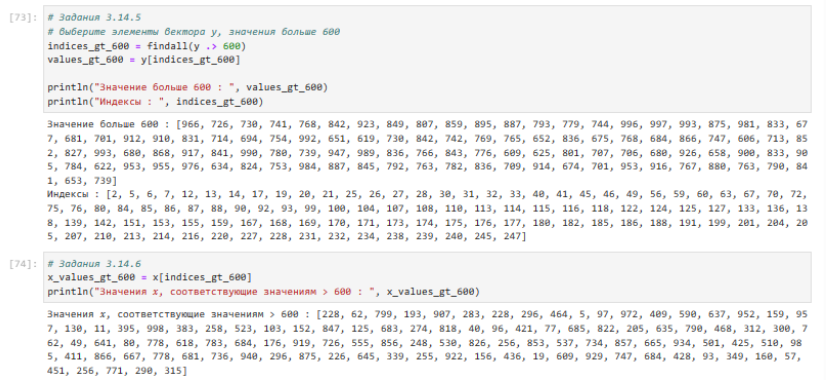
3.14.4. вычислите $\sum_{i=1}^{n-1} \frac{e^{-x_{i+1}}}{x_i+10}$

```
[71]: println("Задания 3.14.3 : ", vector3)

      Задания 3.14.3 : [-4.2808793402840335, 5.052782875536123, -24.33570707460792, 1.4823058801626103, -0.5644909464728004, -4.42217369
      5396321, -0.6831021302467 84, 0.37899742612235077, -1.003975230287573, -0.7462479256211241, 1.4126819734822 35, 1.0265051250681847,
      -0.22868519211337449, 0.7513406222135456, -1.1842200852760154, -1.2587009945728846, 52.3130101719231, 1.5353269869137294, 1.338042
      1566214038, 1.0535207594687757, 0.4669966167554453, -0.2557246067049243, -0.5545181845530521, 2.8653067561499657, 1.05827337487361
      04, 1.18933985022893, 0.1558268137 1560447, -1.1215925811396859, 0.2892751044267637, 0.3354055467418 6807, 2.390796679509297, -0.691
      3063611935918, 3.3187864067852946, 0.7333265084887703, -2.5715551468126248, 2.2643730220555 17, 2.0457289116613135, 0.6865071819837
      713, -222.24343113422444, 1.0993931262200791, 0.4601192366423918 5, 0.2340922473211 0814, 0.0357868699385 77666, -0.0512721818722744
      4, -1.0387472292383035, 0.7819933501169753, -0.9992744873818619, 0.9147156531528382, 0.4304044341885 4624, 1.3678290317705724, -0.8
      805159095798382, 0.5390234695353677, -8.398040384904005, 0.9967462374198038, 11.849672239158501, 0.9905095869885053, -0.4842709033 4
      516265, 0.7514029775846133, -2.432201139512577, -1.0180356680971498, 0.7173652249277354, 1.0618145305890143, 3.3835590719811535, 5
      9.401688852 09957, 223.79008925507 65, 1.1589798545700307, 0.2883984511937284, -0.2876202506596982 6, -1.2997607939902 63, 0.068810878
      69828703, -1.0315497911527314, 2.1320262463594624, -0.46296672964489566, -2.0760643117203 05, 0.9547880530821586, 0.1063853623568848
      42, 0.2699158356096983, -1.0354742183855676, -5.451340110823241, 2.6663074897100083, 1.6833716491725 83, -1.8342957453943 86, -0.811
      9078803755647, -1.7156977017721824, 0.5563759589745295, 1.3961836425320304, 1.4297940596838652, 0.9947743386968448, 0.4383181582 74
      3446, 3.554011780044503, 4.520778661085813, -0.4301630444972701 6, 1.1397380401063637, 0.2830316562303011, -1.1739310265830076, 1.5
      6005645226283, -0.2624132263905342 7, 0.6393761531412757, 34.508421290494105, -1.1193854731776434, -0.7166966651242712, -2.2374928 7
      46877005, 5.040272990088773 5, -0.6443984804160445, 0.4708193014755981, 0.9013357481921925, 0.3237071391125778, -0.1690951607757817
      4, -7.160715710837733, 0.5953759776777463, 2.5949971111152474, 1.866892253142524, 1.1009593280560437, -0.3449163520702633 3, 1.5271
      930810432672, 0.9379017672955017, 0.1415044721606903 4, 1.9694563370773377, -0.9510919109535582, -1.4184625051925 18, -10.9872094642
      1173, 1.4065253028237519, 0.9207872118767911, 0.7960875319599412, 0.7902339311580375, -2.0986526997503643, 1.0960771660585681, -0.8
      2442649731755979, 0.5764816317467157, 1.5447158279115927, 0.0595414409163925 9, 0.6053785061796935, 1.0777484950942513, -0.890099695
      7946974, -1.6680529123904337, 8.528831590456477, 0.9606063647282347, -8.294582391187639, 0.5820496339418 87, -0.7050245224767 06, 6.
      453136935798458, 1.0478326745406 48, -1.2301347486665706, -0.8714510204511889, -0.3761419395893711 4, -0.1587481384469785 5, 8.971457
      32933411, -0.7294749042577021, -0.8908384786358121, 1.0228858352258 42, -0.0455550305442186 5, 0.1876528400206289, 1.5931836497529 12
      6, 0.1557581866295931 5, 5.699152122831511, -1.1875288582695 59, -0.5355267393673 48, -0.5479426823335564, -0.1136873791721241 3, 1.026
      098926326431, 10.337441023571342, 0.3767151728499 7186, 1.4740852271915177, -0.5138307580331412, 3.3550589872594654, -1.3248153854 6
      1884, 1.6821244428616 5, -1.1949597606891434, 2.0282691100429373, 6.300084369270641, 224.46258945631556, 1.1445808044406331, 2.11531
      00206973514, -0.8286571016805676, 0.4986708417746924 5, 1.3168548966894 13, 0.0353880029100401 4, 0.6180042004154426, 0.8638693335790
      141, -0.9848752913083969, -0.9308476226890344, 0.0446565443782101 64, -0.7326646908221306, -1.2776501780867493, -12.9527248548246 0
      6, 0.7688991265256385, -0.6420386471957671, -0.8115970637475934, 2.0572978389365093, 1.7780640541917 66, 2.0383090726783393, 25.1574
      67158973393, 1.6962328541196354, -0.4755953350358426, -1.1103545822076935, 0.6045380824426 11, 0.4578784727622 1516, 0.4303640533 60
      76894, 1.7959341517427234, -0.4779592091974881, -1.5209971839936778, -1.2133411318425014, 10.6078441655466 75, 0.1872324894876 9688,
      -0.7995468421170088, -0.7841381811721081, -1.6432914312399 26, 4.7288449355547 79, 0.8260734213361532, -0.6612699013967798, -0.60911
      01680940102, 0.6181408957932567, 0.5387481447396844, 1.3580532911259615, -0.9578249296191558, 0.3376217920816 1125, -0.3623271398 17
      31934, 0.9661567998452543, -0.8973620407332 4, -1.1819402025300685, 0.0773727975891989 7, 0.5134478137267597, -0.9372556965098 27, 0.
      2965256259419 1356, -0.8744403629251 71, -0.8741991393591756, -1.2984755738362925, -1.0216486808635774, 2.1333009846026543, 1.022700
      3458457549, 0.9989740957433497, 0.6814138603903101, -0.9934616206666407, -77.80931234090802, 2.5138244302215162, -2.9651414860423 1
      6, 3.2569317858236007, -9.948786805419987, 3.8516596345697964, 1.1605895155331 82, -9.7629766527482 5, 1.5849588296781 83, 0.7432886 7
      8883334, -1.7567226312003477, -0.4367276554755504, 1.0719776008140873, -0.8100164837372955, -1.0906922396299186, -0.5309225446957 6
      56]

[72]: println("Задания 3.14.4 : ", sum_expo)

      Задания 3.14.4 : [1.4713467696164503e-5]
```

Рис. 2.19: Задача с множествами 3.14.3 и 3.14.4

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

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



```
[73]: # Задания 3.14.5
# Выберите элементы вектора y, значения больше 600
indices_gt_600 = findall(y .> 600)
values_gt_600 = y[indices_gt_600]

println("Значение больше 600 : ", values_gt_600)
println("Индексы : ", indices_gt_600)
```

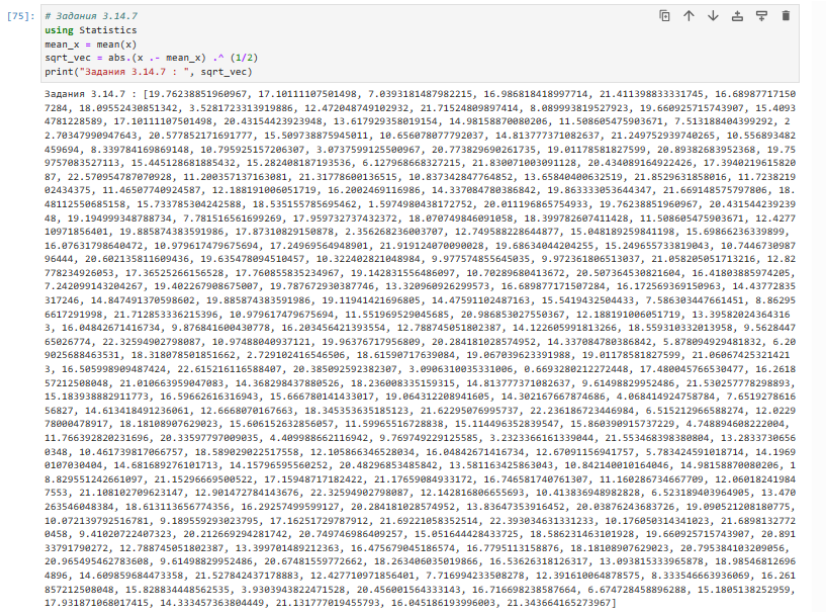
```
Значение больше 600 : [966, 726, 730, 741, 768, 842, 923, 849, 807, 859, 895, 887, 793, 779, 744, 996, 997, 993, 875, 981, 833, 677, 681, 701, 912, 910, 831, 714, 694, 754, 992, 651, 619, 730, 842, 742, 769, 765, 652, 836, 675, 768, 684, 866, 747, 606, 713, 852, 827, 993, 680, 868, 917, 841, 990, 780, 739, 947, 989, 836, 766, 843, 776, 609, 625, 801, 707, 706, 680, 926, 658, 900, 833, 905, 784, 622, 953, 955, 976, 634, 824, 753, 984, 887, 845, 792, 763, 782, 836, 709, 914, 674, 701, 953, 916, 767, 880, 763, 790, 841, 653, 739]
Индексы : [2, 5, 6, 7, 12, 13, 14, 17, 19, 20, 21, 25, 26, 27, 28, 30, 31, 32, 33, 40, 41, 45, 46, 49, 56, 59, 60, 63, 67, 70, 72, 75, 76, 80, 84, 85, 86, 87, 88, 90, 92, 93, 99, 100, 104, 107, 108, 110, 113, 114, 115, 116, 118, 122, 124, 125, 127, 133, 136, 138, 139, 142, 151, 153, 155, 159, 167, 168, 169, 170, 171, 173, 174, 175, 176, 177, 180, 182, 185, 186, 188, 191, 199, 201, 204, 205, 207, 210, 213, 214, 216, 220, 227, 228, 231, 232, 234, 238, 239, 240, 245, 247]
```

```
[74]: # Задания 3.14.6
x_values_gt_600 = x[indices_gt_600]
println("Значения x, соответствующие значениям > 600 : ", x_values_gt_600)
```

```
Значения x, соответствующие значениям > 600 : [228, 62, 799, 193, 907, 283, 228, 296, 464, 5, 97, 972, 409, 590, 637, 952, 159, 957, 130, 11, 395, 998, 383, 258, 523, 103, 152, 847, 125, 683, 274, 818, 40, 96, 421, 77, 685, 822, 205, 635, 790, 468, 312, 300, 762, 49, 641, 80, 778, 618, 783, 684, 176, 919, 726, 555, 856, 248, 530, 826, 256, 853, 537, 734, 857, 665, 934, 501, 425, 510, 985, 411, 866, 667, 778, 681, 736, 940, 296, 875, 226, 645, 339, 255, 922, 156, 436, 19, 609, 929, 747, 684, 428, 93, 349, 160, 57, 451, 256, 771, 290, 315]
```

Рис. 2.20: Задача с множествами 3.14.5 и 3.14.6

3.14.7. сформируйте вектор $(|x_1 - \bar{x}|^{\frac{1}{2}}, |x_2 - \bar{x}|^{\frac{1}{2}}, ..., |x_n - \bar{x}|^{\frac{1}{2}})$, где $\bar{x}$ обозначает среднее значение вектора $x = (x_1, x_2, ..., x_n)$



```
[75]: # Задания 3.14.7
using Statistics
mean_x = mean(x)
sqrt_vec = abs.(x .- mean_x) .^ (1/2)
print("Задания 3.14.7 : ", sqrt_vec)
```

```
Задания 3.14.7 : [19.76238851960967, 17.10111107501498, 7.0393181487982215, 16.986818418997714, 21.411398833331745, 16.68987717150
7284, 18.09552430851342, 3.5281723313919886, 12.472048749102932, 21.71524809897414, 8.089993819527923, 19.660925715743907, 15.4093
4781228589, 17.10111107501498, 20.43154423923948, 13.617929358019154, 14.98158870080206, 11.508605475903671, 7.513188404399292, 2
2.70347990947643, 20.577852171691777, 15.509738875945011, 10.656078077792037, 14.813777371082637, 21.249752939740265, 10.556893482
459694, 8.339784169869148, 10.795925157206307, 3.0737599125500967, 20.773829690261735, 19.011785818275999, 20.89382683952368, 19.75
9757083527113, 15.445128681885432, 15.282408187193536, 6.127968668327215, 21.830071003091128, 20.434089164922426, 17.3940219615820
87, 22.570954787070928, 11.200357137163081, 21.31778600136515, 10.837342847764852, 13.658404006325119, 21.85296318580016, 11.7238219
02434375, 11.46507740924587, 12.188191006051719, 16.2002469116986, 14.337084780386842, 19.863333053644347, 21.669148575797806, 18.
48112550685158, 15.733785304242588, 18.535155785695462, 1.5974980438172752, 20.011196865754933, 19.76238851960967, 20.431544239239
48, 19.194999348788734, 7.781516561699269, 17.959732737432372, 18.070749846091058, 18.399782607411428, 11.508605475903671, 12.4277
10971856401, 19.885874383591986, 17.87310829150878, 2.356268236003707, 12.749588228644877, 15.048189259841198, 15.69866236339899,
16.07631798640472, 10.979617479675694, 17.24969564948901, 21.919124070090028, 19.68634044204255, 15.249655733819043, 10.7446730987
96444, 20.602135811609436, 19.635478094510457, 10.322402821048984, 9.977574855645035, 9.972361806513037, 21.058205051713216, 12.82
778234926053, 17.36525266156528, 17.760855835234967, 19.142831556486097, 10.70289680413672, 20.507364530821604, 16.41803885974205,
7.2420099143204267, 19.402267908675007, 19.787672930387746, 13.320960926299573, 16.689877171507284, 16.172569369150963, 14.43772835
317246, 14.847491370598602, 19.885874383591986, 19.119414211696805, 14.475911024871163, 15.541943250443 3, 7.586303447661451, 8.86295
6617291998, 21.712853336215396, 10.979617479675694, 11.551969529045685, 20.986853027550367, 12.188191006051719, 13.39582024364316
3, 16.04842671416734, 9.876841600430778, 16.203456421393554, 12.788745051802387, 14.122605991813266, 18.559310332013958, 9.5628447
65026774, 22.32594902798087, 10.97488040937121, 19.963767179568 09, 20.284181028574952, 14.337084780386842, 5.878094929481832, 6.20
9025688463531, 18.318078501851662, 2.7291024165465 06, 18.615907176390 84, 19.067039623391988, 19.011785818275 99, 21.0606742532142 1
3, 16.505998909487424, 22.615216116588407, 20.385092592382307, 3.0906310035331006, 0.6693280212272448, 17.480045766530477, 16.2618
57212508048, 21.010663959047083, 14.368298437880526, 18.236008335159315, 14.813777371082637, 9.614988229952486, 21.530257778298893,
15.183938882911773, 16.59662616316943, 15.666780141433017, 19.064312208941605, 14.302167667874686, 4.068414924758784, 7.6519278616
56827, 14.613418491236061, 12.666807016766 3, 18.345353635185123, 21.622950769957 37, 22.236186723446984, 6.515212966588274, 12.0229
78000478917, 18.181089076290 23, 15.606152632856057, 11.599655167288 38, 15.114496352839547, 15.860390915737229, 4.748894608222004,
11.766392820231696, 20.335977970089035, 4.409988662116942, 9.769749229125585, 3.2323366161339044, 21.553468398380804, 13.2833730656
0348, 10.461739817066757, 18.589029022517558, 12.105866346528034, 16.048426714167 34, 12.670911569417 57, 5.783424591018714, 14.1969
0107030404, 14.681689276101713, 14.157965955602 52, 20.48296853485842, 13.581163425863043, 10.842140010164046, 14.9815887008020 6, 1
8.829551242661097, 21.152966695005 22, 17.159487171824 22, 21.176590849331 72, 16.746581740761307, 11.160286734667709, 12.060182419847553, 21.108102709623147, 12.901472784143676, 22.325949027980 87, 12.142816806655693, 10.413836948982828, 6.523189403964905, 13.4702635460484 84, 18.613113656774356, 16.292574995991 27, 20.284181028574952, 13.836473539164 52, 20.038762436837 26, 19.090521208180775,
10.072139792516781, 9.189559293023795, 17.162517297879 12, 21.692210583525 14, 22.393034631331233, 10.176050314341023, 21.6898132772
0458, 9.410207224073 23, 20.212669294281742, 20.749746986409257, 15.051644428433725, 18.586231463101928, 19.660925715743907, 20.891
33791790272, 12.788745051802387, 13.399701489212363, 16.475679045186574, 16.77951133588 76, 18.181089076290 23, 20.795384103209056,
20.965495462783608, 9.614988299524 86, 20.674815597726 62, 18.263406035019866, 16.536263181263 17, 13.093815333965878, 18.985468126964896, 14.609859684473358, 21.527842437178883, 12.427710971856401, 7.716994233508278, 12.391610064878575, 8.333546663936069, 16.261
857212508048, 15.828834448562535, 3.9303943822471528, 20.456001564333143, 16.716698238587664, 6.674728458896288, 15.180513825295 9,
17.931871068017415, 14.333457363804449, 21.131777019455793, 16.045186193996003, 21.343664165273967]
```

Рис. 2.21: Задача с множествами 3.14.7

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

3.14.9. определите, сколько чётных и нечётных элементов вектора x;

3.14.10. определите, сколько элементов вектора x кратны 7;

3.14.11. отсортируйте элементы вектора x в порядке возрастания элементов вектора y;

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

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

Рис. 2.22: Задача с множествами 3.14.8 до 3.14.13

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

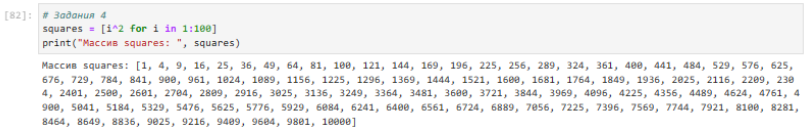


Рис. 2.23: Квадрат массива

5. Подключите пакет Primes (функции для вычисления простых чисел). Сгенерируйте массив myprimes, в котором будут храниться первые 168 простых чисел. Определите 89-е наименьшее простое число. Получите срез массива с 89-го до 99-го элемента включительно, содержащий наименьшие простые числа.

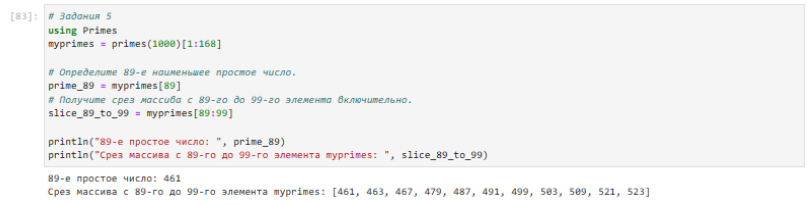


Рис. 2.24: Библиотека и вычисление простого числа

6. Вычислите следующие выражения:

6.1. $\sum_{i=10}^{100}(i^3 + 4i^2)$
6.2. $\sum_{i=1}^{M}(\frac{2^i}{i} + \frac{3^i}{i^2}), M = 25$
6.3. $1 + \frac{2}{3} + (\frac{2}{3}\frac{4}{5}) + (\frac{2}{3}\frac{4}{5}\frac{6}{7}) + ... + (\frac{2}{3}\frac{4}{5}...\frac{38}{39})$

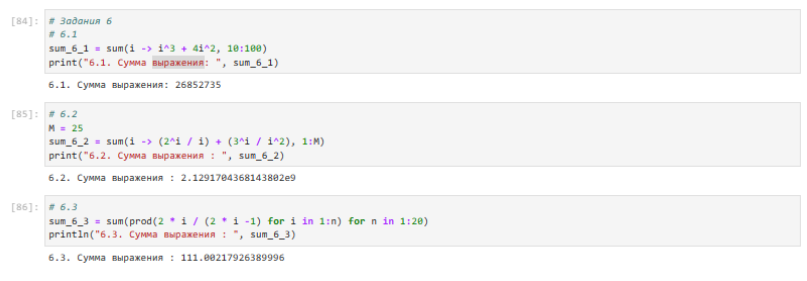


Рис. 2.25: Вычисляющее выражение

3 Листинги программы

```
# Задания 1
# Определение множеств
A = Set([0, 3, 4, 9])
B = Set([1, 3, 4, 7])
C = Set([0, 1, 2, 4, 7, 8, 9])

# Вычисление пересечений и объединений
P = union(intersect(A, B), intersect(A, B), intersect(A, C), intersect(B, C))

# вывод
println("P: ",P)

# Задания 2
# Создание множество
set1 = Set([1, "hello", 3.14, [1, 2, 3]])
set2 = Set([1, 2, "world", 3.14])

intersection_set = intersect(set1, set2)
union_set = union(set1, set2)
isequal_set = issetequal(set1, set2)
sub_set = issubset(set1, set2)
```

```
println("Множество 1 : ", set1)
println("Множество 2 : ", set2)
println("Пересечение : ", intersection_set)
println("Объединение : ", union_set)
println("Эквивалентности множеств : ", isequal_set)
println("Подмножество : ", sub_set)

# Задания 3
# Задания 3.1 до 3.3
N = 25
array_3_1 = collect(1:N)
array_3_2 = reverse(collect(1:N))
array_3_3 = [1:N; N-1:-1:1]

println("Задания 3.1 : ", array_3_1)
println("Задания 3.2 : ", array_3_2)
println("Задания 3.3 : ", array_3_3)

# Задания 3.4 до 3.9
tmp = [4, 6, 3]

array_3_5 = fill(tmp[1], 10)
array_3_6 = repeat([tmp], 10)
array_3_7 = vcat(fill(tmp[1], 11), fill(tmp[2], 10), fill(tmp[3], 10))
array_3_8 = vcat(fill(tmp[1], 10), fill(tmp[2], 20), fill(tmp[3], 30))
array_3_9 = repeat(2 .^ tmp, inner = [1, 1, 4])
count_six = count(x -> x == '6', string(array_3_9))

println("Задания 3.5 : ", array_3_5)
```

```
println("Задания 3.6 : ", array_3_6)
println("Задания 3.7 : ", array_3_7)
println("Задания 3.8 : ", array_3_8)
println("Задания 3.9 : ", array_3_9)
println("Количество цифр 6 в результате 3.9 : ", count_six)

# Задания 3.10
using Statistics;
x_values = 3:0.1:6
y_values = [exp(x) * cos(x) for x in x_values]
mean_y = mean(y_values)
println("Задания 3.10. Среднее значение y : ", mean_y)

# # Задания 3.11
x_vector = [0.1^i for i in 3:3:36]
y_vector = [0.2^j for j in 1:3:34]
result_3_11 = [(x, y) for x in x_vector, y in y_vector]
println("Задания 3.11 : ", result_3_11)

# Задания 3.12
M = 25
result_3_12 = [2^i / i for i in 1:M]
println("Задания 3.12 : ", result_3_12)

# Задания 3.13
N = 30
result_3_13 = ["fn$i" for i in 1:N]
println("Задания 3.13 : ", result_3_13)
```

```julia
# Задания 3.14
# Создание векторов x и y целочисленного типа
n = 250
x = rand(0:999, n)
y = rand(0:999, n)

# Формирование векторов
vector1 = y[2:end] .- x[1:end-1]
vector2 = [x[i] + 2x[i+1] - x[i+2] for i in 1:n-2]
vector3 = [sin(y[i]) / cos(x[i+1]) for i in 1:n-1]
sum_expo = sum([exp(-x[i+1]) / (x[i] +10)] for i in 1:n-1)

# Вывод
println("Задания 3.14.1 : ", vector1)
println("Задания 3.14.2 : ", vector2)
println("Задания 3.14.3 : ", vector3)
println("Задания 3.14.4 : ", sum_expo)

# Задания 3.14.5
# выберите элементы вектора y, значения больше 600
indices_gt_600 = findall(y .> 600)
values_gt_600 = y[indices_gt_600]

println("Значение больше 600 : ", values_gt_600)
println("Индексы : ", indices_gt_600)

# Задания 3.14.6
x_values_gt_600 = x[indices_gt_600]
println("Значения x, соответствующие значениям > 600 : ", x_values_gt_600)
```

```julia
# Задания 3.14.7
using Statistics
mean_x = mean(x)
sqrt_vec = abs.(x .- mean_x) .^ (1/2)
print("Задания 3.14.7 : ", sqrt_vec)

# Задания 3.14.8
count_below_200 = count(x -> x <= 200, values_gt_600)
println("Задания 3.14.8. Количество элементов <= 200 : ", count_below_200)

# Задания 3.14.9
even_count = count(iseven, x)
odd_count = count(isodd, x)
println("Задания 3.14.9 : Чётные : ", even_count)
println(" Нечётные : ", odd_count)

# 3.14.10
divisible_by_7_count = count(x -> x % 7 == 0, x)
println("Задания 3.14.10 : Количество элементов, кратных 7: ", divisible_by_7_cou

# 3.14.11
sorted_indices = sortperm(y)
sorted_x = x[sorted_indices]
println("Задания 3.14.11: Отсортированный x по возрастанию y : ", sorted_x)

# 3.14.12
top_10_x = sorted_x[1:10]
println("Задания 3.14.12: Элементы вектора x в десятке наибольших : ", top_10_x)
```

```
# 3.14.13
unique_x = unique(x)
println("Задания 3.14.13: Уникальные элементы x : ", unique_x)

# Задания 4
squares = [i^2 for i in 1:100]
print("Массив squares: ", squares)

# Задания 5
using Primes
myprimes = primes(1000)[1:168]

# Определите 89-е наименьшее простое число.
prime_89 = myprimes[89]
# Получите срез массива с 89-го до 99-го элемента включительно.
slice_89_to_99 = myprimes[89:99]

println("89-е простое число: ", prime_89)
println("Срез массива с 89-го до 99-го элемента myprimes: ", slice_89_to_99)

# Задания 6
# 6.1
sum_6_1 = sum(i -> i^3 + 4i^2, 10:100)
print("6.1. Сумма выражения: ", sum_6_1)
# 6.2
M = 25
sum_6_2 = sum(i -> (2^i / i) + (3^i / i^2), 1:M)
print("6.2. Сумма выражения : ", sum_6_2)
```

```julia
# 6.3
sum_6_3 = sum(prod(2 * i / (2 * i -1) for i in 1:n) for n in 1:20)
println("6.3. Сумма выражения : ", sum_6_3)
```

4 Вывод

Изучила несколько структур данных, реализованных в Julia, научиться применять их и операции над ними для решения задач.