Отчет по лабораторной работе №2

Структуры данных

Легиньких Галина Андреевна

Содержание

Список иллюстраций

Список таблиц

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 из раздела 2.2.
2. Выполните задания для самостоятельной работы (раздел 2.4).

3 Выполнение лабораторной работы

1. Для начала я повторила все примеры из документа. В начале был дан перечень функций (методов), которые я использовала при повторении примеров. (рис. 3.1)

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.

Рис. 3.1: Перечень методов

2. Рассмотрела примеры кортежей. Комментарии ко всем примерам на скриншотах. (рис. 3.2)

```
[2]: # пустой кортеж:
     ()

[2]: ()

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

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

[6]: # кортеж из целых чисел:
     x1 = (1, 2, 3)

[6]: (1, 2, 3)

[8]: # кортеж из элементов разных типов:
     x2 = (1, 2.0, "tmp")

[8]: (1, 2.0, "tmp")

[10]: # именованный кортеж:
      x3 = (a=2, b=1+2)

[10]: (a = 2, b = 3)
```

Рис. 3.2: Кортежи

3. Повторила примеры операция над кортежами. (рис. 3.3)

```
[12]: # длина кортежа x2:
      length(x2)

[12]: 3

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

[20]: (1, 2.0, "tmp")

[22]: # с вторым и третьим элементами кортежа x1:
      c = x1[2] + x1[3]

[22]: 5

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

[24]: (2, 3, 3)

[26]: in("tmp", x2)

[26]: true

[28]: 0 in x2

[28]: false
```

Рис. 3.3: Операции над кортежами

4. Перешла к словарям и операциями над ними. Повторила примеры операция над словарями. (рис. 3.4) (рис. 3.5)

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

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

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

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

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

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

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

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

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

[40]: true
```

Рис. 3.4: Словари 1

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

[42]: "555-3344"

[46]: # удалить ключ и связанные с ним значения из словаря
pop!(phonebook, "Иванов И.И.")

[46]: ("867-5309", "333-5544")

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

[48]: Dict{String, Any} with 2 entries:
  "Сидоров П.С." => "555-3344"
  "Бухгалтерия"  => "555-2368"

[50]: # Объединение словарей (функция merge()):
a = Dict("foo" => 0.0, "bar" => 42.0);
b = Dict("baz" => 17, "bar" => 13.0);
merge(a, b), merge(b,a)

[50]: (Dict{String, Real}("bar" => 13.0, "baz" => 17, "foo" => 0.0), Dict{String, Re
al}("bar" => 42.0, "baz" => 17, "foo" => 0.0))
```

Рис. 3.5: Словари 2

5. Далее множества и операции над ними. (рис. 3.6) (рис. 3.7) (рис. 3.8)

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

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

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

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

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

[56]: false

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

[58]: true
```

Рис. 3.6: Множества 1

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

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

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

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

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

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

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

[68]: true
```

Рис. 3.7: Множества 2

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

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

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

[72]: 2
```

Рис. 3.8: Множества 3

6. Повторила примеры массивов и операции для работы с ними. (рис. 3.9) (рис. 3.10) (рис. 3.11) (рис. 3.12) (рис. 3.13) (рис. 3.14) (рис. 3.15) (рис. 3.16)

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

[74]: Any[]

[78]: # создание пустого массива с конкретным типом:
      empty_array_2 = (Int64)[]

[78]: Int64[]

[80]: empty_array_3 = (Float64)[]

[80]: Float64[]

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

[82]: 3-element Vector{Int64}:
       1
       2
       3

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

[84]: 1x3 Matrix{Int64}:
       1  2  3

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

[88]: 3x3 Matrix{Int64}:
       1  4  7
       2  5  8
       3  6  9

[90]: B = [[1 2 3]; [4 5 6]; [7 8 9]]

[90]: 3x3 Matrix{Int64}:
       1  2  3
       4  5  6
       7  8  9
```

Рис. 3.9: Массивы 1

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

```
[92]: 1x8 Matrix{Float64}:
       0.330444  0.602191  0.31871  0.625844  …  0.297523  0.494738  0.18971
```

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

```
[96]: 2x3 Matrix{Float64}:
       0.471745  0.656126  0.0519266
       0.680595  0.976129  0.0554209
```

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

```
[98]: 4x3x2 Array{Float64, 3}:
      [:, :, 1] =
       0.973478  0.657723  0.387081
       0.505183  0.217103  0.303282
       0.808302  0.593603  0.200853
       0.51385   0.957914  0.351662

      [:, :, 2] =
       0.423366   0.21554   0.55696
       0.0284772  0.551653  0.659923
       0.802924   0.685677  0.0436171
       0.775933   0.437156  0.631416
```

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

```
[100]: 10-element Vector{Float64}:
        1.0
        1.4142135623730951
        1.7320508075688772
        2.0
        2.23606797749979
        2.449489742783178
        2.6457513110645907
        2.8284271247461903
        3.0
        3.1622776601683795
```


Рис. 3.10: Массивы 2

```
[106]: # 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]
```

```
[106]: 5-element Vector{Int64}:
          3
         27
         75
        147
        243
```

```
[108]: # 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)]
```

```
[108]: 4-element Vector{Int64}:
         1
         9
        49
        81
```

Рис. 3.11: Массивы 3

```
[110]: # одномерный массив из пяти единиц:
       ones(5)

[110]: 5-element Vector{Float64}:
        1.0
        1.0
        1.0
        1.0
        1.0

[112]: # двумерный массив 2x3 из единиц:
       ones(2,3)

[112]: 2×3 Matrix{Float64}:
        1.0  1.0  1.0
        1.0  1.0  1.0

[114]: # одномерный массив из 4 нулей:
       zeros(4)

[114]: 4-element Vector{Float64}:
        0.0
        0.0
        0.0
        0.0

[116]: # заполнить массив 3x2 цифрами 3.5
       fill(3.5,(3,2))

[116]: 3×2 Matrix{Float64}:
        3.5  3.5
        3.5  3.5
        3.5  3.5

[118]: # заполнение массива посредством функции repeat():
       repeat([1,2],3,3)

[118]: 6×3 Matrix{Int64}:
        1  1  1
        2  2  2
        1  1  1
        2  2  2
        1  1  1
        2  2  2
```

Рис. 3.12: Массивы 4

```
[120]: repeat([1 2],3,3)

[120]: 3×6 Matrix{Int64}:
        1  2  1  2  1  2
        1  2  1  2  1  2
        1  2  1  2  1  2

[126]: a = collect(1:12)
       b = reshape(a,(2,6))

[126]: 2×6 Matrix{Int64}:
        1  3  5  7   9  11
        2  4  6  8  10  12

[128]: # транспонирование
       b'

[128]: 6×2 adjoint(::Matrix{Int64}) with eltype Int64:
         1   2
         3   4
         5   6
         7   8
         9  10
        11  12

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

[130]: 6×2 transpose(::Matrix{Int64}) with eltype Int64:
         1   2
         3   4
         5   6
         7   8
         9  10
        11  12
```

Рис. 3.13: Массивы 5

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

```
[132]: 10×5 Matrix{Int64}:
        20  19  13  19  11
        20  14  14  20  18
        19  10  13  19  12
        10  16  15  10  19
        11  18  12  13  12
        16  10  20  20  15
        15  19  16  13  10
        17  14  19  13  10
        12  20  19  18  16
        11  11  10  11  20
```

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

```
[134]: 10-element Vector{Int64}:
        19
        14
        10
        16
        18
        10
        19
        14
        20
        11
```

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

```
[136]: 10×2 Matrix{Int64}:
        19  11
        14  18
        10  12
        16  19
        18  12
        10  15
        19  10
        14  10
        20  16
        11  20
```

Рис. 3.14: Массивы 6

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

```
[138]: 10×3 Matrix{Int64}:
        19  13  19
        14  14  20
        10  13  19
        16  15  10
        18  12  13
        10  20  20
        19  16  13
        14  19  13
        20  19  18
        11  10  11
```

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

```
[140]: 3×2 Matrix{Int64}:
        20  18
        10  19
        16  15
```

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

```
[142]: 3-element Vector{Int64}:
        13
        19
        11
```

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

```
[146]: 10×5 Matrix{Int64}:
        10  10  10  10  10
        11  10  12  11  10
        11  11  13  13  11
        12  14  13  13  12
        15  14  14  13  12
        16  16  15  18  15
        17  18  16  19  16
        19  19  19  19  18
        20  19  19  20  19
        20  20  20  20  20
```

Рис. 3.15: Массивы 7

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

[148]: 10×5 Matrix{Int64}:
        11  13  19  19  20
        14  14  18  20  20
        10  12  13  19  19
        10  10  15  16  19
        11  12  12  13  18
        10  15  16  20  20
        10  13  15  16  19
        10  13  14  17  19
        12  16  18  19  20
        10  11  11  11  20

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

[150]: 10×5 BitMatrix:
        1  1  0  1  0
        1  0  0  1  1
        1  0  0  1  0
        0  1  1  0  1
        0  1  0  0  0
        1  0  1  1  1
        1  1  1  0  0
        1  0  1  0  0
        0  1  1  1  1
        0  0  0  0  1

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

[152]: 26-element Vector{CartesianIndex{2}}:
        CartesianIndex(1, 1)
        CartesianIndex(2, 1)
        CartesianIndex(3, 1)
        CartesianIndex(6, 1)
        CartesianIndex(7, 1)
        CartesianIndex(8, 1)
        CartesianIndex(1, 2)
        CartesianIndex(4, 2)
        CartesianIndex(5, 2)
        CartesianIndex(7, 2)
        CartesianIndex(9, 2)
        CartesianIndex(4, 3)
        CartesianIndex(6, 3)
```

Рис. 3.16: Массивы 8

7. Перешла к выполнению самостоятельных заданий. Дублировать сами задания я не буду, это масимально проблематично. Я просто буду придерживаться их нумерации. Все объяснения в видео выполнение.

- Задания 1 и 2 (рис. 3.17)

```
[160]: A = Set([0,3,4,9])
       B = Set([1,3,4,7])
       C = Set([0,1,2,4,7,8,9])
       P = union(intersect(A,B),intersect(A,B),intersect(A,C),intersect(B,C))
       println("P = $P")

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

[176]: A = Set([0,3,4,9,"a"])
       B = Set(["a","b","c",3])
       C = union(A,B)
       println("C = $C")
       D = intersect(A,B)
       println("D = $D")

       C = Set(Any[0, 4, "c", 9, "b", 3, "a"])
       D = Set(Any["a", 3])
```

Рис. 3.17: Задания 1 и 2

- Задания 3.1 - 3.5 (рис. 3.18)

```
[180]: N = 22
       array1 = collect(1:N)
       println("array1 = $array1")

       array1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2
       0, 21, 22]

[182]: N = 22
       array2 = collect(N:-1:1)
       println("array2 = $array2")

       array2 = [22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5,
       4, 3, 2, 1]

[188]: array3 = vcat(array1, collect(N-1:-1:1))
       println("array3 = $array3")

       array3 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2
       0, 21, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4,
       3, 2, 1]

[190]: tmp = [4,6,3]

[190]: 3-element Vector{Int64}:
        4
        6
        3

[196]: array4 = fill(tmp[1],10)
       println("array4 = $array4")

       array4 = [4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
```

Рис. 3.18: Задания 3.1-3.5

- Задания 3.6 - 3.10 (рис. 3.19)

```
[222]: array5 = repeat(tmp,10)
       println("array5 = $array5")
```

```
array5 = [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]
```

```
[210]: array6 = vcat(fill(tmp[1],11),fill(tmp[2],10),fill(tmp[3],10))
       println("array6 = $array6")
```

```
array6 = [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]
```

```
[212]: array7 = vcat(fill(tmp[1],10),fill(tmp[2],20),fill(tmp[3],30))
       println("array7 = $array7")
```

```
array7 = [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, 3]
```

```
[236]: array8 = [2^tmp[i] for i in 1:3]
       array8 = vcat(array8[1:2],fill(array8[3],4))
       count6 = count(x ->x==6,array8)
       println("array8 = $array8")
       println("Количество 6 в массиве array8: $count6")
```

```
array8 = [16, 64, 8, 8, 8, 8]
Количество 6 в массиве array8: 0
```

```
[254]: x = 3:0.1:6
       y = exp.(x).*cos.(x)
       mean_y = sum(y)/length(y)
       println("mean_y = $mean_y")
```

```
mean_y = 53.11374594642971
```

Рис. 3.19: Задания 3.6-3.10

- Задание 3.11 (рис. 3.20)

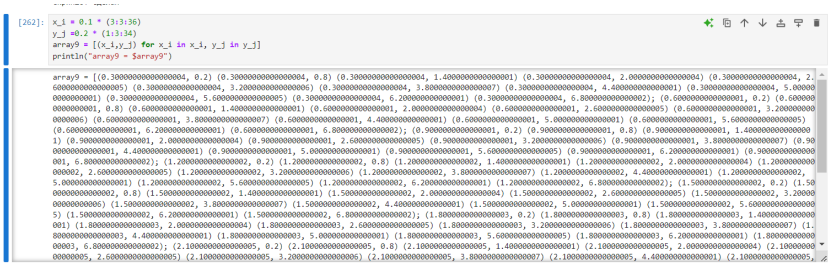

Рис. 3.20: Задание 3.11

- Задания 3.12 - 3.13 (рис. 3.21)

```
[268]: M = 25
       array10 = [2^i/i for i in 1:M]
       println("array10 = $array10")
```

```
array10 = [2.0, 2.0, 2.6666666666666665, 4.0, 6.4, 10.666666666666666, 18.2857
14285714285, 32.0, 56.888888888888886, 102.4, 186.1818181818182, 341.333333333
3333, 630.1538461538462, 1170.2857142857142, 2184.5333333333333, 4096.0, 7710.
117647058823, 14563.555555555555, 27594.105263157893, 52428.8, 99864.380952380
95, 190650.18181818182, 364722.0869565217, 699050.6666666666, 1.34217728e6]
```

```
[274]: N = 30
       array11 = ["fn$i" for i in 1:N]
       println("array11 = $array11")
```

```
array11 = ["fn1", "fn2", "fn3", "fn4", "fn5", "fn6", "fn7", "fn8", "fn9", "fn1
0", "fn11", "fn12", "fn13", "fn14", "fn15", "fn16", "fn17", "fn18", "fn19", "f
n20", "fn21", "fn22", "fn23", "fn24", "fn25", "fn26", "fn27", "fn28", "fn29",
"fn30"]
```

Рис. 3.21: Задания 3.12-3.13

- Задание 3.14.1 (рис. 3.22)

```
[282]: N = 250
       x = rand(0:999,N)
       y = rand(0:999,N)
```

```
250-element Vector{Int64}: ●●●
```

```
[288]: # Задание 1
       v = y[2:end].-x[1:end-1]
       println("$v")
```

```
[-367, -81, -251, 172, -64, 561, -782, 703, 360, 190, -646, -618, -379, -58, -
94, 276, 288, 360, -496, 652, -108, -494, -64, -254, 81, -428, 293, -401, 37,
333, 480, -374, -75, -632, -517, 220, 159, 194, 20, 609, 43, -361, 459, 77, 13
2, -556, -167, 659, 853, -70, 829, -224, 85, 51, -49, 414, -864, 536, 717, -18
2, 352, 296, 190, 562, -503, -207, 239, -732, 19, -187, 841, 589, -138, -407,
-684, -84, 16, 57, -537, -13, 466, -613, -682, 367, -97, 182, -891, -910, 164,
-449, 284, 466, 387, 206, 699, 501, 459, 333, -123, 77, -503, 527, -633, -484,
-53, -381, 213, -81, -343, -919, 98, -242, 22, 770, -490, -121, -309, 168, -40
5, -758, 768, -429, 603, -299, -852, 304, 615, 742, -135, 117, 897, 266, -377,
-311, 781, -548, 211, -686, 100, 151, -81, -267, -518, -386, -349, -110, -304,
123, -783, 170, -444, -90, 271, -290, 136, -752, 95, -271, 713, 818, -832, 24
9, -415, 571, -603, -169, 91, -441, 182, 207, -622, 799, -537, 197, -39, -62,
-339, 692, 321, 37, 18, 280, 274, -345, -315, 199, -442, -847, 222, 132, -550,
110, -159, 192, 28, 819, -33, -601, -592, -209, 136, 597, -605, 319, 47, -149,
277, -227, 810, 120, 292, -679, 88, -294, -324, 2, -11, 491, 79, 156, 20, -17
5, -414, -192, 639, -116, -599, -560, 766, -63, 198, -460, -877, 138, 354, 11
8, 642, 27, -82, 349, 322, -556, -25, -80, -14, 107, 438, -48, -234]
```

Рис. 3.22: Задание 3.14.1

- Задание 3.14.2 (рис. 3.23)

```
[300]: # Задание 2
v_2 = x[1:end-2].+ 2 .*x[2:end-1].-x[3:end]
println("$v_2")
```

```
[2006, 904, 1299, 1988, 370, 2131, 747, -175, 824, 1038, 1588, 2342, 799, 150
6, 327, 1281, 345, 1135, 817, 278, 772, 1523, 745, 1035, 1730, -41, 1062, 108
5, 897, 21, 1671, 1200, 1000, 2167, 1052, -11, -308, 1407, 266, 748, 1590, 33
6, 106, 810, 1696, 1440, 613, -552, 1895, 407, 364, 1639, 1293, 2430, 603, 192
3, 1056, -4, 1131, 469, 480, 1250, 339, 1300, 1194, 689, 1909, 826, 1020, 272,
-625, 1290, 1768, 2532, 612, 809, 1515, 2263, 1820, -131, 593, 2012, 416, 126
8, 408, 1339, 2814, 708, 1109, 455, 857, -43, 1313, 835, -147, 725, 152, 293,
830, 1638, 276, 993, 2079, 16, 1303, 1009, 962, 1068, 2126, 1481, 799, 1708, 4
57, 306, 1698, 2177, 564, 458, 2301, 438, 1252, 375, 729, 1985, 2008, 711, 30
7, 693, 339, -129, 228, 1180, 1889, -226, 1668, 321, 973, 1351, 1482, 1121, 10
9, 1142, 1737, 1363, 2368, 1134, 614, 2091, 224, 1086, 1711, -130, 1320, 504,
1589, 769, 1133, 514, -837, 1029, 1801, 1450, -72, 984, 2230, 1206, 1728, 168
0, 1290, 2500, 254, 1455, 685, -29, 1216, 2141, 872, 1157, 33, 990, 525, -1, 1
63, 2020, 629, 985, 2025, 1185, 1087, 1711, 295, 1680, 1816, 1183, 575, -339,
731, 1452, 2019, 1091, -448, 1641, 1236, -87, 1091, 709, 834, -20, 1093, 1293,
1866, 724, 889, 1716, 1135, 1758, 1188, 895, 1488, 1982, 700, 1540, 2006, 457,
1405, 2217, 1973, 868, -273, 571, 1721, 2616, 1152, 549, 1004, 501, 520, 762,
208, -659, 1003, 1404, 218, 1677, 871, 338, 808, 1545]
```

Рис. 3.23: Задание 3.14.2

- Задание 3.14.3 (рис. 3.24)

```
[306]: # Задание 3
v_3 = sin.(y[1:end-1]).*cos.(x[2:end])
println("$v_3")
```

```
[0.4029224223996743, -0.4160525100761176, -0.8294347664172707, -0.3972786371
4159684, 0.26574207970227975, 0.3688142567950392, -0.024299298261488755, -0.
4379133316583163, 0.06009117582982427, 0.017910008932158505, 0.0414173842275
38816, -0.2933180815054248, 0.8733343192469714, -0.79975113632628, 0.8135514
027936551, 0.40553798683489667, 0.11439120967997221, -0.009558459089234288,
0.23579359175926173, 0.11705858179908164, -0.3092422184769499, -0.4438110196
037077, 0.10563474675199663, -0.46895063178845237, -0.3816402767447685, -0.1
4115757153361375, 0.751279823655116, -0.00999356873491156, -0.10556652028436
453, 0.637971279118843, -0.4458545776029028, -0.3107632418752964, -0.2150773
8009600433, -0.07564611393509427, -0.667809796342176, 0.29187728658670753,
0.14131639036636512, -0.801009917851022, -0.09816253709283593, 0.17122273455
249876, -0.4597029325307745, -0.3877004762036667, 0.2852719381875076, -0.336
08484656959203, -0.17498016095160673, 0.1667334224717661, -0.491144207200378
66, 0.9776924482680401, -0.223651018334295, -0.9703614178337387, -0.78672391
95472487, -0.2852100569603724, -0.4908992755476883, -0.7033723608996801, -0.
039203134448666655, -0.978216842660593, 0.46547300678914116, 0.24529154982459
106, 0.3200709232246895, -0.18968569152133147, 0.5528413678944817, 0.9322384
901546129, -0.24722724976320182, -0.382476436331816, -0.04848397344701281, -
```

Рис. 3.24: Задание 3.14.3

- Задание 3.14.4 - 3.14.5 (рис. 3.25)

```
[316]: # Задание 4
sum1 = sum(exp.(-x[2:end])./(x[1:end-1].+ 10))
```

```
[316]: 0.011994226986598393
```

```
[322]: # Задание 5
y_600 = y[y.>600]
ind_y_600 = findall(y.>600)
println("Элементы y > 600: $y_600")
println("Индексы этих элементов: $ind_y_600")
```

```
Элементы y > 600: [777, 976, 670, 845, 720, 820, 933, 902, 707, 727, 759, 686,
705, 635, 748, 809, 960, 846, 873, 959, 757, 944, 676, 703, 909, 724, 737, 83
4, 749, 858, 762, 850, 748, 915, 617, 872, 867, 814, 833, 623, 719, 621, 602,
709, 927, 818, 871, 814, 918, 771, 957, 997, 675, 800, 716, 778, 869, 664, 73
8, 819, 949, 747, 797, 680, 919, 939, 853, 772, 965, 785, 707, 775, 830, 790,
972, 704, 995, 896, 936, 821, 978, 697, 984, 806, 999, 869, 954, 911, 702, 82
6, 793, 796]
Индексы этих элементов: [3, 5, 6, 7, 9, 11, 18, 21, 26, 31, 32, 40, 41, 42, 4
6, 49, 50, 51, 52, 54, 55, 56, 57, 59, 60, 63, 64, 65, 68, 72, 73, 74, 78, 79,
86, 93, 95, 96, 98, 101, 103, 108, 112, 114, 115, 117, 122, 124, 127, 128, 12
9, 132, 133, 136, 140, 141, 147, 149, 160, 161, 163, 165, 167, 168, 170, 171,
173, 177, 179, 180, 190, 191, 194, 195, 197, 203, 210, 211, 212, 218, 219, 22
0, 221, 222, 226, 227, 230, 232, 236, 238, 246, 248]
```

Рис. 3.25: Задание 3.14.4-3.14.5

- Задание 3.14.6 - 3.14.7 (рис. 3.26)

```
[334]: # Задание 6
mean_x = sum(x)/length(x)
v_6 = abs.(x .- mean_x).^(1/2)
println("$v_6")
```

```
[14.481574500032792, 18.780734809905603, 8.960133927570503, 17.2834024428062
2, 15.12335941515641, 14.875617634236233, 20.657105315120994, 22.09714913738
874, 16.919929077865547, 11.167631799087934, 12.23584896931962, 19.917730794
445436, 19.123702570370625, 14.25776981157993, 13.06583330675851, 18.9019575
70579825, 11.820152283282987, 18.662368552785576, 10.426696504646138, 15.977
609333063567, 14.43204767176162, 7.7920472277829536, 12.3173049000177, 12.17
7191794498434, 10.987083325432643, 8.289511445193861, 21.477523134663365, 1
4.06115215762919, 10.829773774183835, 10.54912318631269, 15.042739112276061,
20.632886371033987, 5.940033669938243, 12.873072671277825, 17.51330922470108
2, 19.423799834223992, 22.119764917376497, 18.283435125818123, 12.6773814330
8783, 20.230768645802858, 9.31214261059183, 5.168752267230459, 20.9591030342
42663, 13.314803791269325, 10.52216707717569, 15.643401164708397, 9.23493367
5993563, 18.848978752176468, 19.957053890792597, 20.266129378842916, 21.4775
23134663365, 9.576847080328683, 19.20197906466935, 14.16742743055351, 22.084
293060906433, 15.597563912355032, 20.216725748745766, 18.392498470844032, 1
7.699830507663062, 7.050957381802843, 18.283435125818123, 8.791131895268094,
6.458792456798717, 15.273637418768327, 15.707195803197973, 5.41146930140049
8, 2.171635328502465, 17.36997409324493, 17.037722852541062, 9.7099948506680
```

```
[338]: # Задание 7
y_max = maximum(y)
y_200 = count(abs.(y.-y_max).<=200)
println("Количество элементов вектора y, которые отстоят от максимального значе
не более, чем на 200: $y_200")
```

```
Количество элементов вектора y, которые отстоят от максимального значения
не более, чем на 200: 49
```

Рис. 3.26: Задание 3.14.6-3.14.7

- Задание 3.14.8 - 3.14.10 (рис. 3.27)

[340]:
```
# Задание 8
count_1 = count(x_i -> x_i % 2 != 0, x)
count_2 = count(x_i -> x_i % 2 == 0, x)
println("Четных: $count_2. Нечетных: $count_2")
```

```
Четных: 147. Нечетных: 147
```

[348]:
```
# Задание 9
x_7 = count(x_i -> x_i % 7 == 0, x)
println("Количество элементов вектора x кратны 7: $x_7")
```

```
Количество элементов вектора x кратны 7: 36
```

[362]:
```
# Задание 10
s = sortperm(y)
x_by_y = x[s]
println("Элементы вектора x в порядке возрастания элементов вектора y: $x_by_y")
```

```
Элементы вектора x в порядке возрастания элементов вектора y: [902, 485, 843, 821, 228, 70
0, 82, 715, 812, 541, 504, 176, 766, 167, 939, 188, 616, 713, 692, 657, 142, 215, 273, 51
9, 182, 614, 753, 320, 983, 626, 913, 151, 699, 250, 732, 300, 192, 80, 44, 17, 103, 1, 18
5, 746, 477, 66, 804, 171, 912, 601, 566, 420, 25, 807, 100, 69, 17, 129, 411, 922, 676, 4
76, 687, 211, 756, 150, 203, 546, 510, 242, 871, 966, 54, 128, 19, 834, 388, 559, 282, 74
6, 508, 609, 238, 358, 465, 97, 826, 338, 618, 760, 703, 858, 16, 32, 858, 724, 964, 107,
666, 178, 874, 171, 580, 805, 671, 461, 778, 153, 616, 687, 979, 989, 348, 380, 210, 61, 6
45, 394, 936, 406, 408, 758, 35, 273, 938, 449, 247, 274, 794, 261, 581, 302, 661, 500, 73
7, 662, 722, 614, 954, 428, 990, 386, 328, 360, 92, 226, 184, 728, 702, 714, 470, 374, 13
6, 630, 148, 832, 357, 255, 491, 342, 628, 642, 532, 816, 284, 690, 914, 938, 96, 828, 42
0, 192, 888, 592, 574, 643, 157, 627, 750, 219, 547, 279, 272, 1, 887, 750, 858, 807, 993,
931, 988, 215, 790, 684, 425, 388, 44, 369, 211, 589, 589, 873, 418, 107, 56, 727, 724, 86
4, 655, 487, 287, 598, 79, 752, 932, 44, 859, 765, 173, 115, 348, 925, 680, 106, 597, 644,
297, 555, 981, 933, 156, 732, 544, 157, 771, 911, 262, 463, 126, 279, 706, 916, 464, 100,
734, 618, 786, 776, 409, 985]
```

Рис. 3.27: Задание 3.14.8-3.14.10

- Задание 3.14.11 - 3.14.12 (рис. 3.28)

[366]:
```
# Задание 11
top_10 = sort(x, rev=true)[1:10]
println("Топ 10 x: $top_10")
```

```
Топ 10 x: [993, 990, 989, 988, 985, 983, 981, 979, 966, 964]
```

[368]:
```
# Задание 12
un = unique(x)
println("Уникальные (неповторяющиеся) элементы вектора x: $un")
```

```
Уникальные (неповторяющиеся) элементы вектора x: [715, 858, 425, 804, 734, 284, 932, 17, 2
19, 630, 655, 902, 871, 302, 676, 148, 645, 157, 614, 250, 297, 566, 657, 357, 626, 574, 4
4, 703, 388, 394, 279, 931, 470, 671, 812, 128, 16, 171, 666, 96, 592, 532, 66, 328, 616,
750, 420, 150, 107, 916, 597, 874, 706, 993, 262, 914, 167, 192, 555, 428, 547, 272, 752,
476, 510, 807, 215, 411, 173, 988, 859, 938, 203, 732, 933, 461, 35, 662, 766, 182, 714, 3
42, 990, 983, 142, 559, 151, 406, 106, 661, 115, 56, 374, 79, 380, 546, 642, 699, 69, 821,
408, 628, 702, 964, 504, 491, 687, 544, 939, 724, 210, 580, 912, 103, 680, 211, 727, 936,
156, 80, 100, 409, 690, 609, 19, 873, 97, 746, 627, 282, 843, 826, 758, 979, 348, 541, 81
6, 82, 756, 508, 61, 760, 261, 778, 228, 465, 25, 1, 864, 700, 463, 176, 887, 966, 589, 73
7, 911, 54, 765, 129, 338, 834, 790, 273, 464, 519, 92, 178, 449, 913, 255, 794, 485, 643,
684, 300, 989, 598, 369, 153, 692, 753, 226, 888, 242, 136, 601, 247, 386, 185, 776, 644,
771, 320, 805, 581, 832, 487, 618, 828, 786, 418, 922, 722, 360, 985, 925, 188, 126, 713,
981, 954, 184, 287, 238, 32, 477, 274, 358, 728, 500]
```

Рис. 3.28: Задание 3.14.11-3.14.12

- Задание 4 (рис. 3.29)

```
[374]: squares = [i^2 for i in 1:100]
println("squares = $squares")
```

```
squares = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 3
61, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024, 1089, 1156, 1225, 12
96, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025, 2116, 2209, 2304, 2401, 2500, 26
01, 2704, 2809, 2916, 3025, 3136, 3249, 3364, 3481, 3600, 3721, 3844, 3969, 4096, 4225, 43
56, 4489, 4624, 4761, 4900, 5041, 5184, 5329, 5476, 5625, 5776, 5929, 6084, 6241, 6400, 65
61, 6724, 6889, 7056, 7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836, 9025, 92
16, 9409, 9604, 9801, 10000]
```

Рис. 3.29: Задание 4

- Задание 5 (рис. 3.30)

```
[411]: using Primes
```

```
[423]: myprimes = primes(1, 10000)[1:168]
println("89-е наименьшее простое число: $(myprimes[89])")
println("Срез массива с 89-го до 99-го элемента: $(myprimes[89:99])")
```

```
89-е наименьшее простое число: 461
Срез массива с 89-го до 99-го элемента: [461, 463, 467, 479, 487, 491, 499, 503, 509, 521,
523]
```

Рис. 3.30: Задание 5

- Задание 6 (рис. 3.31)

```
[427]: sum1 = sum(i^3 + 4* i^2 for i in 10:100)
println("Задание 6.1: Сумма = $sum1")
```

```
Задание 6.1: Сумма = 26852735
```

```
[429]: M = 25
sum2 = sum((2^i/i)+(3^i/i^2) for i in 1:M)
println("Задание 6.2: Сумма = $sum2")
```

```
Задание 6.2: Сумма = 2.1291704368143802e9
```

```
[431]: sum3 = sum(prod(2:i)/prod(i+1:2*i-1) for i in 1:38)
println("Задание 6.3: Сумма = $sum3")
```

```
Задание 6.3: Сумма = 22.37818348501321
```

Рис. 3.31: Задание 6

4 Вывод

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