Лабораторная Работа №2

Структуры данных

Козлов В.П.

Российский университет дружбы народов им. Патриса Лумумбы, Москва, Россия

Докладчик

- Козлов Всеволод Павлович
- НФИбд-02-22
- Российский университет дружбы народов
- [1132226428@pfur.ru]

Выполнение лабораторной работы

Цель работы

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

Задание

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

Примеры кортежей

Figure 1: Примеры кортежей

Примеры операций над кортежами

Figure 2: Примеры операций над кортежами

Словари

Словари

[12]:
```
# создать словарь с именем phonebook:
phonebook = Dict("Иванов И.И." => ("867-5309","333-5544"), "Бухгалтерия" => "555-2368")
```

[12]:
```
Dict{String, Any} with 2 entries:
  "Бухгалтерия" => "555-2368"
  "Иванов И.И." => ("867-5309", "333-5544")
```

[13]:
```
# вывести ключи словаря:
keys(phonebook)
```

[13]:
```
KeySet for a Dict{String, Any} with 2 entries. Keys:
  "Бухгалтерия"
  "Иванов И.И."
```

[15]:
```
# вывести значения элементов словаря:
values(phonebook)
```

[15]:
```
ValueIterator for a Dict{String, Any} with 2 entries. Values:
  "555-2368"
  ("867-5309", "333-5544")
```

[16]:
```
# вывести заданные в словаре пары "ключ - значение":
pairs(phonebook)
```

[16]:
```
Dict{String, Any} with 2 entries:
  "Бухгалтерия" => "555-2368"
  "Иванов И.И." => ("867-5309", "333-5544")
```

Figure 3: Словари

Словари

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

[17]: true

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

[18]: "555-3344"

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

[20]: ("867-5309", "333-5544")

[21]: # 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)

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

Figure 4: Словари

Примеры множеств и операций над ними

Множества Примеры множеств и операций над ними

[22]:
```
# создать множество из четырёх целочисленных значений:
A = Set([1, 3, 4, 5])
```

[22]:
```
Set{Int64} with 4 elements:
  5
  4
  3
  1
```

[23]:
```
# создать множество из 11 символьных значений:
B = Set("abrakadabra")
```

[23]:
```
Set{Char} with 5 elements:
  'a'
  'd'
  'r'
  'k'
  'b'
```

[24]:
```
# проверка эквивалентности двух множеств:
S1 = Set([1,2]);
S2 = Set([3,4]);
issetequal(S1,S2)
S3 = Set([1,2,2,3,1,2,3,2,1]);
S4 = Set([2,3,1]);
issetequal(S3,S4)
```

[24]:
```
true
```

Figure 5: Примеры множеств и операций над ними

Примеры множеств и операций над ними

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

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

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

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

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

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

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

[28]: true

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

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

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

[30]: 2
```

Примеры множеств и операций над ними

Массивы

[31]:
```
# создание пустого массива с абстрактным типом:
empty_array_1 = []
```

[31]:
```
Any[]
```

[32]:
```
# создание пустого массива с конкретным типом:
empty_array_2 = (Int64)[]
empty_array_3 = (Float64)[]
```

[32]:
```
Float64[]
```

[33]:
```
# вектор-столбец:
a = [1, 2, 3]
```

[33]:
```
3-element Vector{Int64}:
 1
 2
 3
```

[34]:
```
# вектор-строка:
b = [1 2 3]
```

[34]:
```
1×3 Matrix{Int64}:
 1  2  3
```

[35]:
```
# многомерные массивы (матрицы):
A = [[1, 2, 3] [4, 5, 6] [7, 8, 9]]
B = [[1 2 3]; [4 5 6]; [7 8 9]]
```

[35]:
```
3×3 Matrix{Int64}:
 1  2  3
 4  5  6
 7  8  9
```


Figure 7: Примеры множеств и операций над ними

Примеры множеств и операций над ними

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

[36]: 1×8 Matrix{Float64}:
       0.215133  0.300672  0.242618  0.316867  …  0.504034  0.0442552  0.696184

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

[40]: 2×3 Matrix{Float64}:
       0.915162  0.635159  0.0699625
       0.512862  0.71221   0.792008

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

[39]: 4×3×2 Array{Float64, 3}:
      [:, :, 1] =
       0.635051   0.288559  0.0872019
       0.734769   0.768252  0.557425
       0.16587    0.483326  0.156759
       0.0789743  0.254288  0.237161

      [:, :, 2] =
       0.977329   0.0397281  0.211892
       0.548963   0.942451   0.828726
       0.0302323  0.629465   0.344746
       0.739478   0.0360661  0.768606
```

Figure 8: Примеры множеств и операций над ними

Примеры множеств и операций над ними

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

```
[41]: 10-element Vector{Float64}:
 1.0
 1.4142135623730951
 1.7320508075688772
 2.0
 2.23606797749979
 2.449489742783178
 2.6457513110645907
 2.8284271247461903
 3.0
 3.1622776601683795
```

```
[42]: # 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]
```

```
[42]: 5-element Vector{Int64}:
   3
  27
  75
 147
 243
```

```
[43]: # 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)]
```

```
[43]: 4-element Vector{Int64}:
  1
  9
 49
 81
```

Figure 9: Примеры множеств и операций над ними

Операции для работы с массивами

Операции для работы с массивами

[44]:
```
# одномерный массив из пяти единиц:
ones(5)
```

[44]:
```
5-element Vector{Float64}:
 1.0
 1.0
 1.0
 1.0
 1.0
```

[45]:
```
# двумерный массив 2x3 из единиц:
ones(2,3)
```

[45]:
```
2×3 Matrix{Float64}:
 1.0  1.0  1.0
 1.0  1.0  1.0
```

[46]:
```
# одномерный массив из 4 нулей:
zeros(4)
```

[46]:
```
4-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
```

[47]:
```
# заполнить массив 3x2 цифрами 3.5
fill(3.5,(3,2))
```

[47]:
```
3×2 Matrix{Float64}:
 3.5  3.5
 3.5  3.5
 3.5  3.5
```


Figure 10: Операции для работы с массивами

Операции для работы с массивами

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

```
[48]: 3×6 Matrix{Int64}:
       1  2  1  2  1  2
       1  2  1  2  1  2
       1  2  1  2  1  2
```

```
[49]: # преобразование одномерного массива из целых чисел от 1 до 12
      # в двумерный массив 2x6
      a = collect(1:12)
      b = reshape(a,(2,6))
```

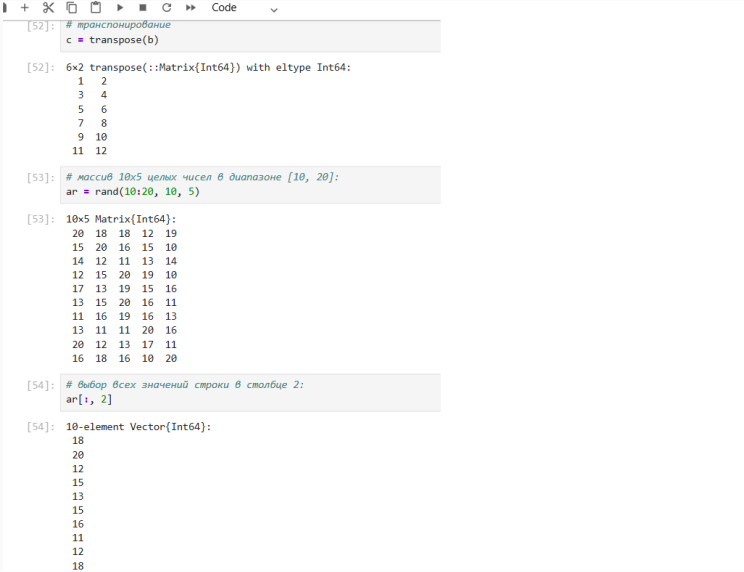
```
[49]: 2×6 Matrix{Int64}:
       1  3  5  7   9  11
       2  4  6  8  10  12
```

```
[51]: # транспонирование
      b'
```

```
[51]: 6×2 adjoint(::Matrix{Int64}) with eltype Int64:
        1   2
        3   4
        5   6
        7   8
        9  10
       11  12
```

Figure 11: Операции для работы с массивами

Операции для работы с массивами



Операции для работы с массивами

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

[55]: 10×2 Matrix{Int64}:
       18  19
       20  10
       12  14
       15  10
       13  16
       15  11
       16  13
       11  16
       12  11
       18  20

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

[56]: 10×3 Matrix{Int64}:
       18  18  12
       20  16  15
       12  11  13
       15  20  19
       13  19  15
       15  20  16
       16  19  16
       11  11  20
       12  13  17
       18  16  10

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

[57]: 3×2 Matrix{Int64}:
       15  10
       12  10
       13  11
```

Figure 13: Операции для работы с массивами

Операции для работы с массивами

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

[58]: 3-element Vector{Int64}:
       18
       12
       19

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

[59]: 10×5 Matrix{Int64}:
       11  11  11  10  10
       12  12  11  12  10
       13  12  13  13  11
       13  13  16  15  11
       14  15  16  15  13
       15  15  18  16  14
       16  16  19  16  16
       17  18  19  17  16
       20  18  20  19  19
       20  20  20  20  20

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

[60]: 10×5 Matrix{Int64}:
       12  18  18  19  20
       10  15  15  16  20
       11  12  13  14  14
       10  12  15  19  20
       13  15  16  17  19
       11  13  15  16  20
       11  13  16  16  19
       11  11  13  16  20
       11  12  13  17  20
       10  16  16  18  20
```

Figure 14: Операции для работы с массивами

Операции для работы с массивами

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

```
[61]: 10×5 BitMatrix:
       1  1  1  0  1
       1  1  1  1  0
       0  0  0  0  0
       0  1  1  1  0
       1  0  1  1  1
       0  1  1  1  0
       0  1  1  1  0
       0  0  0  1  1
       1  0  0  1  0
       1  1  1  0  1
```

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

```
[62]: 29-element Vector{CartesianIndex{2}}:
       CartesianIndex(1, 1)
       CartesianIndex(2, 1)
       CartesianIndex(5, 1)
       CartesianIndex(9, 1)
       CartesianIndex(10, 1)
       CartesianIndex(1, 2)
       CartesianIndex(2, 2)
       CartesianIndex(4, 2)
       CartesianIndex(6, 2)
       CartesianIndex(7, 2)
       CartesianIndex(10, 2)
       CartesianIndex(1, 3)
       CartesianIndex(2, 3)
       ⋮
       CartesianIndex(10, 3)
       CartesianIndex(2, 4)
       CartesianIndex(4, 4)
       CartesianIndex(5, 4)
       CartesianIndex(6, 4)
       CartesianIndex(7, 4)
       CartesianIndex(8, 4)
       CartesianIndex(9, 4)
       CartesianIndex(1, 5)
       CartesianIndex(5, 5)
```


Figure 15: Операции для работы с массивами

Самостоятельная работа. Задание 1-2

[63]:
```
using Primes

# Задание 1: Множества
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("Задание 1: P = $P")
println()
```

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

[64]:
```
# Задание 2: Примеры операций над множествами разных типов
set1 = Set([1, 2.5, "hello", :symbol])
set2 = Set([2, 3.7, "world", :symbol])

println("Задание 2:")
println("Объединение: $(union(set1, set2))")
println("Пересечение: $(intersect(set1, set2))")
println("Разность set1-set2: $(setdiff(set1, set2))")
println("Разность set2-set1: $(setdiff(set2, set1))")
println("Принадлежность :symbol set1: $(in(:symbol, set1))")
println()
```

```
Задание 2:
Объединение: Set(Any[:symbol, 2, 3.7, "hello", 2.5, 1, "world"])
Пересечение: Set(Any[:symbol])
Разность set1-set2: Set(Any["hello", 2.5, 1])
Разность set2-set1: Set(Any[2, 3.7, "world"])
Принадлежность :symbol set1: true
```

Figure 16: Самостоятельная работа. Задание 1-2

Самостоятельная работа. Задание 3.1-3.3

[71]:
```
N = 25
M = 25
n = 250

# 3.1 Массив от 1 до N. больше 20
arr1 = collect(1:N)
elements_greater_than_20 = filter(x -> x > 20, arr1)
println("Элементы больше 20: $elements_greater_than_20")
```

```
Элементы больше 20: [21, 22, 23, 24, 25]
```

[72]:
```
# 3.2 Массив от N до 1. больше 20
arr2 = collect(N:-1:1)
elements_greater_than_20 = filter(x -> x > 20, arr2)
println("Элементы больше 20: $elements_greater_than_20")
```

```
Элементы больше 20: [25, 24, 23, 22, 21]
```

[73]:
```
# 3.3 Массив 1,2,...,N-1,N,N-1,...,2,1. больше 20
arr3 = vcat(collect(1:N), collect((N-1):-1:1))
elements_greater_than_20 = filter(x -> x > 20, arr3)
println("Элементы больше 20: $elements_greater_than_20")
```

```
Элементы больше 20: [21, 22, 23, 24, 25, 24, 23, 22, 21]
```

Figure 17: Самостоятельная работа. Задание 3.1-3.3

Самостоятельная работа. Задание 3.4-3.8

```
[75]: # 3.4 Массив tmp = (4, 6, 3)
      tmp = [4, 6, 3]
      println("3.4: tmp = $tmp")

      3.4: tmp = [4, 6, 3]

[76]: # 3.5 Первый элемент tmp повторяется 10 раз
      arr5 = fill(tmp[1], 10)
      println("3.5: $arr5")

      3.5: [4, 4, 4, 4, 4, 4, 4, 4, 4, 4]

[77]: # 3.6 Все элементы tmp повторяются 10 раз
      arr6 = repeat(tmp, 10)
      println("3.6: $arr6")

      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]

[78]: # 3.7 Элементы tmp повторяются 11, 10, 10 раз
      arr7 = vcat(fill(tmp[1], 11), fill(tmp[2], 10), fill(tmp[3], 10))
      println("3.7: $arr7")

      3.7: [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, 3]

[79]: # 3.8 Элементы tmp повторяются подряд 10, 20, 30 раз
      arr8 = vcat(fill(tmp[1], 10), fill(tmp[2], 20), fill(tmp[3], 30))
      println("3.8: $arr8")

      3.8: [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, 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]
```

Figure 18: Самостоятельная работа. Задание 3.4-3.8

Самостоятельная работа. Задание 3.9-3.11

[80]:
```
# 3.9 Массив 2^tmp[i] с повторениями
arr9 = vcat(fill(2^tmp[1], 1), fill(2^tmp[2], 1), fill(2^tmp[3], 4))
count_6 = count(x -> x == 6, arr9)
println("3.9: $arr9, количество цифр 6: $count_6")
```

```
3.9: [16, 64, 8, 8, 8, 8], количество цифр 6: 0
```

[82]:
```
using Statistics
# 3.10 Вектор y = e^x * cos(x) для x = 3:0.1:6
x_range = 3:0.1:6
y = [exp(x) * cos(x) for x in x_range]
y_mean = mean(y)
println("3.10: Среднее значение y: $y_mean")
```

```
3.10: Среднее значение y: 53.11374594642971
```

[83]:
```
# 3.11 Вектор (x^i, y^j)
x_val = 0.1
y_val = 0.2
arr11 = [x_val^i for i in 3:3:36]
arr11b = [y_val^j for j in 1:3:34]
println("3.11: Длина первого вектора: $(length(arr11)), длина второго: $(length(arr11b))")
```

```
3.11: Длина первого вектора: 12, длина второго: 12
```

Figure 19: Самостоятельная работа. Задание 3.9-3.11

Самостоятельная работа. Задание 3.12-3.13

[84]:
```
# 3.12 Вектор 2^i / i
arr12 = [2^i / i for i in 1:M]
println("3.12: Первые 5 элементов: $(arr12[1:5])")
```

```
3.12: Первые 5 элементов: [2.0, 2.0, 2.6666666666666665, 4.0, 6.4]
```

[85]:
```
# 3.13 Вектор "fn1", "fn2", ..., "fnN"
arr13 = ["fn$i" for i in 1:30]
println("3.13: $(arr13[1:3])...$(arr13[end-2:end])")
```

```
3.13: ["fn1", "fn2", "fn3"]...["fn28", "fn29", "fn30"]
```

Figure 20: Самостоятельная работа. Задание 3.12-3.13

Самостоятельная работа. Задание 3.14. Векторы целочисленного типа длины n=250 как случайные выьорки из совокупнсоти 0,..999

```
[89]: using Statistics
      using Random
      # 3.14 Работа со случайными векторами
      Random.seed!(123) # для воспроизводимости
      x = rand(0:999, n)
      y = rand(0:999, n)
```

```
[89]: 250-element Vector{Int64}:
       552
       310
       462
        96
       650
       753
       911
       670
       922
       767
       655
       824
       254
         ⋮
        33
       928
       505
       419
       324
       342
       736
        90
       766
```

Самостоятельная работа. Задание 3.14.1-3.14.5

[91]:
```
# Вектор y₂ - x₁, ..., yₙ - xₙ₋₁
diff_vec = [y[i+1] - x[i] for i in 1:n-1]
println("3.14.1: Первые 5 разностей: $(diff_vec[1:5])")
```
```
3.14.1: Первые 5 разностей: [-211, -124, -794, 460, 228]
```

[92]:
```
# Вектор x₁ + 2x₂ - x₃, x₂ + 2x₃ - x₄, ...
linear_comb = [x[i] + 2x[i+1] - x[i+2] for i in 1:n-2]
println("3.14.2: Первые 5 линейных комбинаций: $(linear_comb[1:5])")
```
```
3.14.2: Первые 5 линейных комбинаций: [803, 2176, 745, 850, 1261]
```

[93]:
```
# Вектор sin(yᵢ)/cos(xᵢ₊₁)
trig_vec = [sin(y[i]) / cos(x[i+1]) for i in 1:n-1]
println("3.14.3: Первые 5 тригонометрических отношений: $(trig_vec[1:5])")
```
```
3.14.3: Первые 5 тригонометрических отношений: [8.571950432842645, -1.421910020571243 6, -2.787213484786365, -1.048618557004918 7, 0.337236010254091 56]
```

[94]:
```
# Сумма e^(-xᵢ₊₁)/(xᵢ + 10)
sum_exp = sum(exp(-x[i+1]) / (x[i] + 10) for i in 1:n-1)
println("3.14.4: Сумма: $sum_exp")
```
```
3.14.4: Сумма: 0.000170652105658974 41
```

[95]:
```
# Элементы y > 600
y_gt_600 = y[y .> 600]
indices_gt_600 = findall(y .> 600)
println("3.14.5: Количество элементов y > 600: $(length(y_gt_600))")
```
```
3.14.5: Количество элементов y > 600: 114
```

Figure 22: Самостоятельная работа. Задание 3.14.1-3.14.5

Самостоятельная работа. Задание 3.14.6-3.14.9

[96]:
```
# Соответствующие значения x
x_corresponding = x[indices_gt_600]
println("3.14.6: Соответствующие значения x (первые 5): $(x_corresponding[1:5])")
```

```
3.14.6: Соответствующие значения x (первые 5): [525, 390, 44, 933, 580]
```

[97]:
```
# Вектор |x_i - x̄|^(1/2)
x_mean = mean(x)
sqrt_dev = [sqrt(abs(x[i] - x_mean)) for i in 1:n]
println("3.14.7: Первые 5 отклонений: $(sqrt_dev[1:5])")
```

```
3.14.7: Первые 5 отклонений: [5.741428393701344, 9.897676495016393, 20.049039877260956, 17.263719182146122, 6.079802628375365]
```

[98]:
```
# Элементы y, отстоящие от максимума не более чем на 200
y_max = maximum(y)
close_to_max = count(y .>= y_max - 200)
println("3.14.8: Элементы y в пределах 200 от максимума: $close_to_max")
```

```
3.14.8: Элементы y в пределах 200 от максимума: 57
```

[99]:
```
# Чётные и нечётные элементы x
even_count = count(iseven, x)
odd_count = count(isodd, x)
println("3.14.9: Чётных: $even_count, нечётных: $odd_count")
```

```
3.14.9: Чётных: 129, нечётных: 121
```

Figure 23: Самостоятельная работа. Задание 3.14.6-3.14.9

Самостоятельная работа. Задание 3.14.10-3.14.13

[100]:
```
# Элементы x, кратные 7
multiple_7 = count(x -> x % 7 == 0, x)
println("3.14.10: Кратных 7: $multiple_7")
```

```
3.14.10: Кратных 7: 30
```

[101]:
```
# Сортировка x по возрастанию y
sorted_x_by_y = x[sortperm(y)]
println("3.14.11: Первые 5 отсортированных x: $(sorted_x_by_y[1:5])")
```

```
3.14.11: Первые 5 отсортированных x: [471, 799, 152, 34, 100]
```

[102]:
```
# Top-10 элементов x
top10_x = sort(x, rev=true)[1:10]
println("3.14.12: Top-10 x: $top10_x")
```

```
3.14.12: Top-10 x: [996, 993, 993, 986, 983, 974, 959, 953, 949, 943]
```

[103]:
```
# Уникальные элементы x
unique_x = unique(x)
println("3.14.13: Уникальных элементов: $(length(unique_x))")
println()
```

```
3.14.13: Уникальных элементов: 227
```

Figure 24: Самостоятельная работа. Задание 3.14.10-3.14.13

Самостоятельная работа. Задание 4-5

[104]:
```
# Задание 4: Квадраты чисел от 1 до 100
squares = [i^2 for i in 1:100]
println("Задание 4: Квадраты от 1 до 100 созданы, первые 5: $(squares[1:5])")
println()
```

```
Задание 4: Квадраты от 1 до 100 созданы, первые 5: [1, 4, 9, 16, 25]
```

[105]:
```
# Задание 5: Простые числа
myprimes = primes(1000)[1:168] # Первые 168 простых чисел
prime_89 = myprimes[89]
primes_slice = myprimes[89:99]
println("Задание 5:")
println("89-е наименьшее простое число: $prime_89")
println("Срез с 89-го по 99-й: $primes_slice")
println()
```

```
Задание 5:
89-е наименьшее простое число: 461
Срез с 89-го по 99-й: [461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]
```

Figure 25: Самостоятельная работа. Задание 4-5

Самостоятельная работа. Задание 6

[106]:
```
# Задание 6: Вычисления сумм
# 6.1 Сумма i³ + 4i² от 10 до 100
sum1 = sum(i^3 + 4i^2 for i in 10:100)
println("6.1: Сумма = $sum1")
```

```
6.1: Сумма = 26852735
```

[107]:
```
# 6.2 Сумма (2i/i) + (3i/i²) от 1 до M
sum2 = sum(2i/i + 3i/i^2 for i in 1:M)
println("6.2: Сумма = $sum2")
```

```
6.2: Сумма = 61.44787453326052
```

[108]:
```
# 6.3 Сумма ряда 1 + 2/3 + 24/35 + 246/357 + ...
function compute_series(n_terms)
    result = 1.0
    current_num = 2.0
    current_denom = 3.0

    for i in 2:n_terms
        term = current_num / current_denom
        result += term

        # Обновляем числитель и знаменатель для следующего члена
        current_num = current_num * 10 + (current_num % 10 + 2)
        current_denom = current_denom * 10 + (current_denom % 10 + 2)
    end

    return result
end

sum3 = compute_series(5) # 5 членов ряда
println("6.3: Сумма ряда (5 членов) = $sum3")
```

```
6.3: Сумма ряда (5 членов) = 3.7310346770728406
```


Figure 26: Самостоятельная работа. Задание 6

Выводы

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