

Отчёт по лабораторной работе №2

Структуры данных

Козлов Всеволод Павлович НФИбд-02-22

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	20
5	Список литературы	21

Список иллюстраций

3.1	Примеры кортежей	7
3.2	Примеры операций над кортежами	8
3.3	Словари	8
3.4	Словари	9
3.5	Примеры множеств и операций над ними	9
3.6	Примеры множеств и операций над ними	10
3.7	Примеры множеств и операций над ними	10
3.8	Примеры множеств и операций над ними	11
3.9	Примеры множеств и операций над ними	11
3.10	Операции для работы с массивами	12
3.11	Операции для работы с массивами	12
3.12	Операции для работы с массивами	13
3.13	Операции для работы с массивами	13
3.14	Операции для работы с массивами	14
3.15	Операции для работы с массивами	14
3.16	Самостоятельная работа. Задание 1-2	15
3.17	Самостоятельная работа. Задание 3.1-3.3	15
3.18	Самостоятельная работа. Задание 3.4-3.8	16
3.19	Самостоятельная работа. Задание 3.9-3.11	16
3.20	Самостоятельная работа. Задание 3.12-3.13	17
3.21	Самостоятельная работа. Задание 3.14	17
3.22	Самостоятельная работа. Задание 3.14.1-3.14.5	18
3.23	Самостоятельная работа. Задание 3.14.6-3.14.9	18
3.24	Самостоятельная работа. Задание 3.14.10-3.14.13	18
3.25	Самостоятельная работа. Задание 4-5	19
3.26	Самостоятельная работа. Задание 6	19

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 отчета.
2. Выполните задания для самостоятельной работы.

3 Выполнение лабораторной работы

Примеры кортежей (рис. 3.1)

Примеры кортежей	
[2]:	<code># пустой кортеж: ()</code>
[2]:	<code>()</code>
[3]:	<code># кортеж из элементов типа String: favoritelang = ("Python", "Julia", "R")</code>
[3]:	<code>("Python", "Julia", "R")</code>
[4]:	<code># кортеж из целых чисел: x1 = (1, 2, 3)</code>
[4]:	<code>(1, 2, 3)</code>
[5]:	<code># кортеж из элементов разных типов: x2 = (1, 2.0, "tmp")</code>
[5]:	<code>(1, 2.0, "tmp")</code>
[6]:	<code># именованный кортеж: x3 = (a=2, b=1+2)</code>
[6]:	<code>(a = 2, b = 3)</code>

Рис. 3.1: Примеры кортежей

Примеры операций над кортежами (рис. 3.2)

Примеры операций над кортежами

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

[7]: 3

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

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

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

[9]: 5

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

[10]: (2, 3, 3)

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

[11]: (True, False)
```

Рис. 3.2: Примеры операций над кортежами

Словари (рис. 3.3)

```
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")
```

Рис. 3.3: Словари

Словари (рис. 3.4)


```

[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})

```

Рис. 3.4: Словари

Примеры множеств и операций над ними (рис. 3.5)

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

```

Рис. 3.5: Примеры множеств и операций над ними

Примеры множеств и операций над ними (рис. 3.6)

```

[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

```

Рис. 3.6: Примеры множеств и операций над ними

Примеры множеств и операций над ними (рис. 3.7)

```

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

```

Рис. 3.7: Примеры множеств и операций над ними

Примеры множеств и операций над ними (рис. 3.8)

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

[36]: 1x8 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]: 2x3 Matrix{Float64}:
 0.915162  0.635159  0.0699625
 0.512862  0.71221  0.792008

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

[39]: 4x3x2 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
```

Рис. 3.8: Примеры множеств и операций над ними

Примеры множеств и операций над ними (рис. 3.9)

```
[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
```

Рис. 3.9: Примеры множеств и операций над ними

Операции для работы с массивами (рис. 3.10)

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 2х3 из единиц:
      ones(2,3)

[45]: 2x3 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 3х2 цифрами 3.5
      fill(3.5,(3,2))

[47]: 3x2 Matrix{Float64}:
      3.5  3.5
      3.5  3.5
      3.5  3.5
```

Рис. 3.10: Операции для работы с массивами

Операции для работы с массивами (рис. 3.11)

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

[48]: 3x6 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 2х6
      a = collect(1:12)
      b = reshape(a,(2,6))

[49]: 2x6 Matrix{Int64}:
      1  3  5  7  9  11
      2  4  6  8  10 12

[51]: # транспонирование
      b'

[51]: 6x2 adjoint{::Matrix{Int64}} with eltype Int64:
      1  2
      3  4
      5  6
      7  8
      9  10
      11 12
```

Рис. 3.11: Операции для работы с массивами

Операции для работы с массивами (рис. 3.12)

```

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

[52]: 6x2 transpose(::Matrix{Int64}) with eltype Int64:
      1  2
      3  4
      5  6
      7  8
      9 10
     11 12

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

[53]: 10x5 Matrix{Int64}:
      20 18 18 12 19
      15 20 16 15 10
      14 12 11 13 14
      12 15 20 19 10
      17 13 19 15 16
      13 15 20 16 11
      11 16 19 16 13
      13 11 11 20 16
      20 12 13 17 11
      16 18 16 10 20

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

[54]: 10-element Vector{Int64}:
      18
      20
      12
      15
      13
      15
      16
      11
      12
      18

```

Рис. 3.12: Операции для работы с массивами

Операции для работы с массивами (рис. 3.13)

```

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

[55]: 10x2 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]: 10x3 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 8 столбцах 1 и 5:
      ar[[2, 4, 6], [1, 5]]

[57]: 3x2 Matrix{Int64}:
      15 10
      12 10
      13 11

```

Рис. 3.13: Операции для работы с массивами

Операции для работы с массивами (рис. 3.14)

```

[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]: 10x5 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]: 10x5 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

```

Рис. 3.14: Операции для работы с массивами

Операции для работы с массивами (рис. 3.15)

```

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

[61]: 10x5 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)

```

Рис. 3.15: Операции для работы с массивами

Самостоятельная работа. Задание 1-2 (рис. 3.16)

```
[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, 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, Int64, Float64, String}}([2.5, 1, "world", 2, 3.7, "hello"])
Пересечение: Set{Any{::Symbol}}([:symbol])
Разность set1-set2: Set{Any{String, Float64}}([2.5, 1])
Разность set2-set1: Set{Any{String, Float64}}([2, 3.7, "world"])
Принадлежность :symbol set1: true
```

Рис. 3.16: Самостоятельная работа. Задание 1-2

Самостоятельная работа. Задание 3.1-3.3 (рис. 3.17)

```
[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]
```

Рис. 3.17: Самостоятельная работа. Задание 3.1-3.3

Самостоятельная работа. Задание 3.4-3.8 (рис. 3.18)

Рис. 3.18: Самостоятельная работа. Задание 3.4-3.8

Рис. 3.19: Самостоятельная работа. Задание 3.9-3.11


```
[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"]
```

Рис. 3.20: Самостоятельная работа. Задание 3.12-3.13

Самостоятельная работа. Задание 3.14. Векторы целочисленного типа длины $n=250$ как случайные выборки из совокупности 0,..999 (рис. 3.21)

```
[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
```

Рис. 3.21: Самостоятельная работа. Задание 3.14

Самостоятельная работа. Задание 3.14.1-3.14.5 (рис. 3.22)

```
[91]: # Вектор  $y_2 - x_2, \dots, y_n - x_{n-1}$ 
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  $x_1 + 2x_2 - x_3, x_2 + 2x_3 - x_4, \dots$ 
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  $\sin(y_1)/\cos(x_{1+1})$ 
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, -1.4219100205712436, -2.787213484786365, -1.0486185570049187, 0.33723601025409156]

[94]: # Сумма  $e^{-(x_{i+1})/(x_i + 10)}$ 
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: 0.00017065210565897441

[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
```

Рис. 3.22: Самостоятельная работа. Задание 3.14.1-3.14.5

Самостоятельная работа. Задание 3.14.6-3.14.9 (рис. 3.23)

```
[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  $|x_i - \bar{x}|^{1/2}$ 
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
```

Рис. 3.23: Самостоятельная работа. Задание 3.14.6-3.14.9

Самостоятельная работа. Задание 3.14.10-3.14.13 (рис. 3.24)

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

Рис. 3.24: Самостоятельная работа. Задание 3.14.10-3.14.13

Самостоятельная работа. Задание 4-5 (рис. 3.25)

```
[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]
```

Рис. 3.25: Самостоятельная работа. Задание 4-5

Самостоятельная работа. Задание 6 (рис. 3.26)

```
[106]: # Задание 6: Вычисления сумм
# 6.1 Сумма  $i^3 + 4i^2$  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  $(2i/i) + (3i/i^2)$  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  $1 + 2/3 + 24/35 + 246/357 + \dots$ 
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
```

Рис. 3.26: Самостоятельная работа. Задание 6

4 Выводы

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

5 Список литературы

1. Julia 1.5 Documentation. — 2020. — URL: <https://docs.julialang.org/en/v1/>.
2. Klok H.,Nazarathy Y. Statistics with Julia: Fundamentals for Data Science,Machine Learning and Artificial Intelligence. — 2020. — URL: <https://statisticswithjulia.org/>.
3. Ökten G. First Semester in Numerical Analysis with Julia. — Florida State University, 2019. — DOI: 10.33009/jul.
4. Антонюк В. А. Язык Julia как инструмент исследователя. — М. : Физический факультет МГУ им. М. В. Ломоносова, 2019.
5. Шиндин А. В. Язык программирования математических вычислений Julia. Базовое руководство. — Нижний Новгород : Нижегородский госуниверситет, 2016.