

Лабораторная работа №2

Структуры данных

Тазаева Анастасия Анатольевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Кортежи	7
3.2	Словари	8
3.3	Множества	10
3.4	Массивы	11
3.5	Самостоятельная работа	19
4	Выводы	23

Список иллюстраций

3.1	Примеры кортежей	7
3.2	Примеры операций над кортежами	8
3.3	Примеры словарей и операций над ними. Часть 1	9
3.4	Примеры словарей и операций над ними. Часть 2	9
3.5	Примеры множеств и операций над ними. Часть 1	10
3.6	Примеры множеств и операций над ними. Часть 2	11
3.7	Примеры множеств и операций над ними. Часть 3	11
3.8	Примеры массивов. Часть 1	12
3.9	Примеры массивов. Часть 2	13
3.10	Примеры массивов, заданных некоторыми функциями через включение	14
3.11	Некоторые операции для работы с массивами. Часть 1	15
3.12	Некоторые операции для работы с массивами. Часть 2	15
3.13	Некоторые операции для работы с массивами. Часть 3	16
3.14	Некоторые операции для работы с массивами. Часть 4	17
3.15	Некоторые операции для работы с массивами. Часть 5	18
3.16	Некоторые операции для работы с массивами. Часть 6	19
3.17	Самостоятельная работа. Задание 1 и 2	19
3.18	Самостоятельная работа. Задание 3. Подпункты 3.1, 3.2, 3.3, 3.4 . .	20
3.19	Самостоятельная работа. Задание 3. Подпункты 3.5, 3.6, 3.7, 3.8 . .	20
3.20	Самостоятельная работа. Задание 3. Подпункты 3.9, 3.10	20
3.21	Самостоятельная работа. Задание 3. Подпункт 3.11	21
3.22	Самостоятельная работа. Задание 3. Подпункты 3.12, 3.13, 3.14 . .	21
3.23	Самостоятельная работа. Задание 3. Подпункт 3.14	21
3.24	Самостоятельная работа. Задание 3. Подпункт 3.14	22
3.25	Самостоятельная работа. Задание 4 и 5	22
3.26	Самостоятельная работа. Задание 6. Подпункт 6.1, 6.2, 6.3	22

Список таблиц

1 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

3.1 Кортежи

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

(element1, element2, ...)

Примеры представлены на рис. 3.1 и 3.2 :

Кортежи

Примеры кортежей

```
[3]: () # Пустой кортеж
[3]: ()

[5]: favoritelang = ("python", "julia", "R") #Кортеж из элементов типа String
[5]: ("python", "julia", "R")

[13]: x1 = (1, 2, 3) # кортеж из целых чисел
[13]: (1, 2, 3)

[14]: x2 = (1, 2.0, "tmp") # Кортеж из элементов разных типов
[14]: (1, 2.0, "tmp")

[15]: x3 = (a=2, b=1+2) # Именованный кортеж
[15]: (a = 2, b = 3)
```

Рис. 3.1: Примеры кортежей

Примеры операций над кортежами

```
[16]: length(x2) # длина кортежа x2
[16]: 3

[17]: x2[1], x2[2], x2[3] # обратиться к элементам кортежа x2
[17]: (1, 2.0, "tmp")

[18]: c = x1[2] + x1[3] # произвести какую-либо операцию (сложение)
      # с вторым и третьим элементами кортежа x1
[18]: 5

[19]: x3.a, x3.b, x3[2] # обращение к элементам именованного кортежа x3
[19]: (2, 3, 3)

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

Рис. 3.2: Примеры операций над кортежами

3.2 Словари

Словарь — неупорядоченный набор связанных между собой по ключу данных.

Синтаксис определения словаря:

```
Dict(key1 => value1, key2 => value2, ...)
```

Примеры представлены на рис. 3.3 и 3.4 :

Примеры словарей и операций над ними

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

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

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

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

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

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

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

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

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

[29]: true
```

Рис. 3.3: Примеры словарей и операций над ними. Часть 1

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

[30]: "555-3344"

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

[31]: ("867-5309", "333-5544")

[32]: keys(phonebook)

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

[33]: # 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)

[33]: (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: Примеры словарей и операций над ними. Часть 2

3.3 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.

Примеры представлены на рис. 3.5, 3.6, 3.7 :

Примеры множеств и операций над ними

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

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

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

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

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

```
[40]: false
```

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

```
[41]: true
```

Рис. 3.5: Примеры множеств и операций над ними. Часть 1

```

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

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

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

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

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

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

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

[45]: true

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

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

```

Рис. 3.6: Примеры множеств и операций над ними. Часть 2

```

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

[47]: 2

[48]: S4

[48]: Set{Int64} with 3 elements:
      99
      3
      1

```

Рис. 3.7: Примеры множеств и операций над ними. Часть 3

3.4 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 представлены на рис. 3.8, 3.9, 3.10 :

Примеры массивов

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

```
[49]: Any[]
```

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

```
[52]: Int64[]
```

```
[51]: empty_array_3 = (Float64)[]
```

```
[51]: Float64[]
```

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

```
[53]: 3-element Vector{Int64}:
 1
 2
 3
```

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

```
[55]: 1×3 Matrix{Int64}:
 1  2  3
```

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

```
[57]: 3×3 Matrix{Int64}:
 1  4  7
 2  5  8
 3  6  9
```

Рис. 3.8: Примеры массивов. Часть 1

```

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

[58]: 3x3 Matrix{Int64}:
      1  2  3
      4  5  6
      7  8  9

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

[59]: 1x8 Matrix{Float64}:
      0.310043  0.950794  0.915795  0.962301 ... 0.185676  0.892984  0.766257

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

[66]: 2x3 Matrix{Float64}:
      0.203864  0.758838  0.393973
      0.738799  0.988954  0.64076

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

[67]: 4x3x2 Array{Float64, 3}:
     [:, :, 1] =
      0.227332  0.844477  0.464529
      0.708991  0.0160303  0.854196
      0.760585  0.115136  0.984205
      0.283116  0.597728  0.772792

     [:, :, 2] =
      0.0107221  0.446021  0.0808428
      0.472339  0.710554  0.627792
      0.337875  0.181368  0.0408687
      0.903711  0.882048  0.893641

```

Рис. 3.9: Примеры массивов. Часть 2

▼ Примеры массивов, заданных некоторыми функциями через включение

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

[68]: 10-element Vector{Float64}:
 1.0
 1.4142135623730951
 1.7320508075688772
 2.0
 2.23606797749979
 2.449489742783178
 2.6457513110645907
 2.8284271247461903
 3.0
 3.1622776601683795

[69]: # 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]

[69]: 5-element Vector{Int64}:
 3
 27
 75
 147
 243

[70]: # 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)]

[70]: 4-element Vector{Int64}:
 1
 9
 49
 81
```

Рис. 3.10: Примеры массивов, заданных некоторыми функциями через включение

Примеры операций над массивами представлены на рис. 3.11 - 3.16 :

Некоторые операции для работы с массивами

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

```
[71]: 5-element Vector{Float64}:
 1.0
 1.0
 1.0
 1.0
 1.0
```

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

```
[73]: 2x3 Matrix{Float64}:
 1.0  1.0  1.0
 1.0  1.0  1.0
```

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

```
[74]: 4-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
```

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

```
[75]: 3x2 Matrix{Float64}:
 3.5  3.5
 3.5  3.5
 3.5  3.5
```

Рис. 3.11: Некоторые операции для работы с массивами. Часть 1

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

```
[95]: 3x2 Matrix{Float64}:
 3.5  3.5
 3.5  3.5
 3.5  3.5
```

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

```
[96]: 3x6 Matrix{Int64}:
 1  2  1  2  1  2
 1  2  1  2  1  2
 1  2  1  2  1  2
```

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

```
[81]: 2x6 Matrix{Int64}:
 1  3  5  7  9 11
 2  4  6  8 10 12
```

```
[83]: # транспонирование
b'
```

```
[83]: 6x2 adjoint{::Matrix{Int64}} with eltype Int64:
 1  2
 3  4
 5  6
 7  8
 9 10
```

Рис. 3.12: Некоторые операции для работы с массивами. Часть 2

```

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

[84]: 6x2 transpose(::Matrix{Int64}) with eltype Int64:
      1  2
      3  4
      5  6
      7  8
      9 10
     11 12

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

[85]: 10x5 Matrix{Int64}:
      15 19 16 16 13
      16 13 17 20 16
      13 16 14 14 11
      15 14 17 12 10
      18 14 12 17 20
      17 18 17 19 16
      13 10 11 19 15
      14 11 19 20 20
      12 11 18 18 13
      13 10 13 10 20

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

[86]: 10-element Vector{Int64}:
      19
      13
      16
      14
      14

```

Рис. 3.13: Некоторые операции для работы с массивами. Часть 3


```

[87]: # Выбор всех значений в столбцах 2 и 5:
      ar[:, [2, 5]]

[87]: 10x2 Matrix{Int64}:
      19  13
      13  16
      16  11
      14  10
      14  20
      18  16
      10  15
      11  20
      11  13
      10  20

[88]: # Все значения строк в столбцах 2, 3 и 4:
      ar[:, 2:4]

[88]: 10x3 Matrix{Int64}:
      19  16  16
      13  17  20
      16  14  14
      14  17  12
      14  12  17
      18  17  19
      10  11  19
      11  19  20
      11  18  18
      10  13  10

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

[89]: 3x2 Matrix{Int64}:
      16  16
      15  10
      17  16

```

Рис. 3.14: Некоторые операции для работы с массивами. Часть 4

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

[90]: 3-element Vector{Int64}:
 16
 16
 13

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

[91]: 10x5 Matrix{Int64}:
 12 10 11 10 10
 13 10 12 12 11
 13 11 13 14 13
 13 11 14 16 13
 14 13 16 17 15
 15 14 17 18 16
 15 14 17 19 16
 16 16 17 19 20
 17 18 18 20 20
 18 19 19 20 20

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

[92]: 10x5 Matrix{Int64}:
 13 15 16 16 19
 13 16 16 17 20
 11 13 14 14 16
 10 12 14 15 17
 12 14 17 18 20
 16 17 17 18 19
 10 11 13 15 19
 11 14 19 20 20
 11 12 13 18 18
 10 10 13 13 20
```

Рис. 3.15: Некоторые операции для работы с массивами. Часть 5

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

[93]: 10x5 BitMatrix:
 1 1 1 1 0
 1 0 1 1 1
 0 1 0 0 0
 1 0 1 0 0
 1 0 0 1 1
 1 1 1 1 1
 0 0 0 1 1
 0 0 1 1 1
 0 0 1 1 0
 0 0 0 0 1

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

[94]: 27-element Vector{CartesianIndex{2}}:
 CartesianIndex{1, 1}
 CartesianIndex{2, 1}
 CartesianIndex{4, 1}
 CartesianIndex{5, 1}
 CartesianIndex{6, 1}
 CartesianIndex{1, 2}
 CartesianIndex{3, 2}
 CartesianIndex{6, 2}
 CartesianIndex{1, 3}
 CartesianIndex{2, 3}
 CartesianIndex{4, 3}
 CartesianIndex{6, 3}
 CartesianIndex{8, 3}
      :
 CartesianIndex{2, 4}
 CartesianIndex{5, 4}
 CartesianIndex{6, 4}
```

Рис. 3.16: Некоторые операции для работы с массивами. Часть 6

3.5 Самостоятельная работа

Выполнение заданий можно просмотреть на рис. 3.17 - 3.26 :

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

```
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))
print(P)

Set{0, 4, 7, 9, 3, 1}
```

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

```
N = Set{2002, 2003, 2004, 2005} # выпускники кружка Олены рога
T = Set{2002, 2000, 2001, 2005, 1998, 1997, 1930, 2009, 1992, 1993, 2003, 2004} # выпускники школы творчества
# входят ли выпускники кружка Олены рога в список выпускников школы творчества
issubset(N,T)

true

N = Set{2002, 2003, 2004, 2005} # выпускники кружка Олены рога
S = Set{2000, 1993, 1994, 1995} # выпускники кружка Олены коты
R = Set{1930, 2009, 2004, 2005} # выпускники кружка Олены хвосты
T = union(N,S,R) # выпускники школы творчества по годам
print(T)

Set{2004, 2005, 1995, 2002, 1930, 1994, 2003, 1993, 2009, 2000}
```

Рис. 3.17: Самостоятельная работа. Задание 1 и 2

```
N = 29
Arr1 = collect(1:1:N)
print(Arr1)
```

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]

```
N = 29
Arr2 = collect(Ni-1:1)
print(Arr2)
```

[29, 28, 27, 26, 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]

```
N = 20
Arr3 = cat(collect(1:1:N),collect(N-1:1),dims=1)
print(Arr3)
```

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

```
tmp = [4, 6, 3]
print("tmp = ", tmp)

tmp = [4, 6, 3]
```

```
Arr3_5 = fill(tmp[1], 10)
print(Arr3_5)
```

```
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
```

```
Arr3_6 = repeat(tnp, 10)
print(Arr3_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]

```
Arr3_7 = cat(fill(tmp[1],11),fill(tmp[2],10),fill(tmp[3],10), dims = 1)
print(Arr3_7)
```

[4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3]

```
Arr3_8 = cat(fill(tmp[1],10),fill(tmp[2],20),fill(tmp[3],30), dims = 1)
print(Arr3_8)
```

[4, 4, 4, 4, 4, 4, 4, 4, 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]

```
#coordanu maccu8
Arr3_9_1 = [2*tmp[i] for i in 1:1:3]
Arr3_9 = cat(Arr3_9_1, repeat([2*tmp[3]],3), dims = 1)
print(Arr3_9)
```

```
res = count(x -> x == '6', string_split)
print("Цифра 6 встречается ", res, " раз")
```

Цифра 6 встречается 2 раз

```
using Statistics
y = [exp(x) * cos(x) for x in 3:0.1:6]
println(mean(y))
```



```

for i in 1:n
    if y[i]>600
        for j in 1:n
            if x[j]>my[i]
                print("y["~i~"]="~y[i]~"~x["~j~"]="~x[j]~", "\n")
            end
        end
    end
end
end
end

```

```

y[1]=613~x[53]=613
y[24]=893~x[119]=893
y[32]=752~x[111]=752
y[36]=897~x[117]=897
y[70]=862~x[75]=862
y[71]=928~x[112]=928
y[75]=828~x[49]=828
y[80]=929~x[190]=929
y[81]=658~x[144]=658
y[89]=776~x[250]=776
y[99]=820~x[142]=820
y[103]=978~x[122]=978
y[105]=617~x[215]=617
y[112]=856~x[44]=856
y[114]=981~x[150]=981
y[121]=656~x[16]=656
y[138]=978~x[122]=978
y[147]=928~x[179]=928
y[148]=940~x[48]=940
y[166]=656~x[16]=656
y[170]=863~x[230]=863
y[174]=667~x[156]=667
y[193]=944~x[55]=944
y[197]=863~x[230]=863
y[201]=720~x[63]=720
y[202]=882~x[211]=882
y[203]=764~x[226]=764
y[213]=856~x[44]=856

```

Рис. 3.24: Самостоятельная работа. Задание 3. Подпункт 3.14

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

```

squares = [i^2 for i in 1:100]
print(squares)

```

```

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024, 1089, 1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025, 2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249, 3364, 3481, 3600, 3721, 3844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5041, 5184, 5329, 5476, 5625, 5776, 5929, 6084, 6241, 6400, 6561, 6724, 6889, 7056, 7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409, 9604, 9801, 10000]

```

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.

```

using Primes

# Создаем список простых чисел до 1000 (или другого достаточно большого числа)
myprimes = primes(1000)

println(myprimes[89])
println(myprimes[89:99])

```

```

461
[461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]

```

6.1)

```

for i in 10:1:100
    res += i^3+4*i^2
end
print(res)

```

```

5.3705470004035644e7

```

Рис. 3.25: Самостоятельная работа. Задание 4 и 5

6.1)

```

for i in 10:1:100
    res += i^3+4*i^2
end
print(res)

```

```

5.3705470004035644e7

```

6.2)

```

sum2 = sum((2^i / i + 3^i / i^2) for i in 1:25)
println(sum2)

```

```

2.1291704368143802e9

```

6.3)

```

sum3 = sum(prod(2:2n) / prod(3:2n+1) for n in 1:19)
println(sum3)

```

```

12.84175745993532

```

Рис. 3.26: Самостоятельная работа. Задание 6. Подпункт 6.1, 6.2, 6.3

4 Выводы

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