

Лабораторная работа №2

Структуры данных

Чемоданова Ангелина Александровна

11 сентября 2025

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

- Чемоданова Ангелина Александровна
- Студентка НФИбд-02-22
- Российский университет дружбы народов имени Патриса Лумумбы
- 1132226443@pfur.ru
- <https://github.com/aachemodanova>



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.
2. Выполните задания для самостоятельной работы.

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

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

Кортежи

```
#пустой кортеж
()
✓ 0.0s
()

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

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

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

# именованный кортеж:
x3 = (a=2, b=1+2)
(a = 2, b = 3)
```

Рис. 1: Примеры кортежей

Кортежи

```
# длина кортежа x2:  
length(x2)  
[157] ... 3  
  
# обратиться к элементам кортежа x2:  
x2[1], x2[2], x2[3]  
[158] ... (1, 2.0, "tmp")  
  
# произвести какую-либо операцию (сложение)  
# с вторым и третьим элементами кортежа x1:  
x1[2] + x1[3]  
[159] ... 5  
  
# обращение к элементам именованного кортежа x3:  
x3.a, x3.b, x3[2]  
[160] ... (2, 3, 3)  
  
▷ ...  
# проверка вхождения элементов tmp и 0 в кортеж x2  
# (два способа обращения к методу in()):  
in("tmp", x2), 0 in x2  
[161] ... (true, false)
```

Рис. 2: Примеры операций над кортежами

Словарь – неупорядоченный набор связанных между собой по ключу данных.

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

Словари

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

Julia

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

```
# вывести ключи словаря:  
keys(phonebook)
```

Julia

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

```
# вывести значения элементов словаря:  
values(phonebook)
```

Julia

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

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

Julia

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

Рис. 3: Примеры словарей и операций над ними

Словари

```
# проверка наличия ключа в словаре:  
haskey(phonebook, "Иванов И.И.")
```

Julia

true

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

Julia

"555-3344"

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

Julia

("867-5309", "333-5544")

```
phonebook
```

Julia

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

```
# 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)
```

Julia

(Dict{String, Real}("bar" => 13.0, "baz" => 17, "foo" => 0.0), Dict{String, Real}("bar" => 42.0, "baz" => 17, "foo" => 0.0))

Рис. 4: Примеры словарей и операций над ними

Множество, как структура данных в Julia, соответствует множеству, как математическому объекту, то есть является неупорядоченной совокупностью элементов какого-либо типа. Возможные операции над множествами: объединение, пересечение, разность; принадлежность элемента множеству.

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

Множества

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

Julia

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

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

Julia

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

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

Julia

```
false
```

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

Julia

```
true
```

Рис. 5: Примеры множеств и операций над ними

Множества

```
C=union(S1,S2)
```

Julia

```
Set{Int64} with 4 elements:
```

```
4  
2  
3  
1
```

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

Julia

```
Set{Int64} with 2 elements:
```

```
2  
1
```

```
# разность множеств:  
E = setdiff(S3,S1)
```

Julia

```
Set{Int64} with 1 element:
```

```
3
```

```
# проверка вхождения элементов одного множества в другое:  
issubset(S1,S4)
```

Julia

```
true
```

Множества

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

Julia

```
Set{Int64} with 4 elements:  
2  
99  
3  
1
```

```
# удаление последнего элемента множества:  
pop!(S4)
```

Julia

```
2
```

```
S4
```

Julia

```
Set{Int64} with 3 elements:  
99  
3  
1
```

Рис. 7: Примеры множеств и операций над ними

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 ...]`

Массивы

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

Julia

Any[]

```
empty_array_2 = (Int64)[]
```

Julia

Int64[]

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

Julia

Float64[]

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

Julia

```
3-element Vector{Int64}:  
1  
2  
3
```

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

Julia

```
1x3 Matrix{Int64}:  
1 2 3
```

Рис. 8: Примеры массивов

Массивы

```
A = [[1, 2, 3] [4, 5, 6] [7, 8, 9]]
```

Julia

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

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

Julia

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

Рис. 9: Примеры массивов

Массивы

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

Julia

```
1x8 Matrix{Float64}:
0.323291  0.175305  0.872976  0.418621 ... 0.939439  0.715507  0.924749
```

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

Julia

```
2x3 Matrix{Float64}:
0.397739  0.373398  0.0950988
0.399855  0.983419  0.790952
```

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

Julia

```
4x3x2 Array{Float64, 3}:
[:, :, 1] =
0.795965  0.378973  0.0517636
0.531543  0.0751465 0.350175
0.621869  0.532999  0.133856
0.220307  0.395279  0.65738

[:, :, 2] =
0.18628  0.0360285 0.637506
0.333688 0.631937  0.860791
0.739044  0.582138  0.606298
0.278239  0.655225  0.697197
```

Рис. 10: Примеры массивов

Массивы

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

Julia

```
10-element Vector{Float64}:  
1.0  
1.4142135623730951  
1.7320508075688772  
2.0  
2.23660797749979  
2.449489742783178  
2.6457513110645907  
2.8284271247461903  
3.0  
3.1622776601683795
```

```
ar_1 = [3*i^2 for i in 1:2:9]
```

Julia

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

```
# 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)]
```

Julia

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

Рис. 11: Примеры массивов, заданных некоторыми функциями через включение

Массивы

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

Julia

```
5-element Vector{Float64}:  
1.0  
1.0  
1.0  
1.0  
1.0
```

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

Julia

```
2x3 Matrix{Float64}:  
1.0 1.0 1.0  
1.0 1.0 1.0
```

```
# одномерный массив из 4 нулей:  
zeros(4)
```

Julia

```
4-element Vector{Float64}:  
0.0  
0.0  
0.0  
0.0
```

Рис. 12: Некоторые операции для работы с массивами

Массивы

```
# заполнить массив 3x2 цифрами 3.5
fill(3.5,(3,2))

3x2 Matrix{Float64}:
3.5  3.5
3.5  3.5
3.5  3.5
```

Julia

Рис. 13: Некоторые операции для работы с массивами

Массивы

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

Julia

```
6x3 Matrix{Int64}:
1 1 1
2 2 2
1 1 1
2 2 2
1 1 1
2 2 2
```

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

Julia

```
3x6 Matrix{Int64}:
1 2 1 2 1 2
1 2 1 2 1 2
1 2 1 2 1 2
```

```
# преобразование одномерного массива из целых чисел от 1 до 12
# в двумерный массив 2х6
a = collect(1:12)
b = reshape(a,(2,6))
```

Julia

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

Рис. 14: Некоторые операции для работы с массивами

Массивы

```
# транспонирование  
b'
```

Julia

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

```
# транспонирование  
c = transpose(b)
```

Julia

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

Рис. 15: Некоторые операции для работы с массивами

Массивы

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

Julia

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



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

Julia

```
10-element Vector{Int64}:  
11  
16  
14  
10  
20  
15  
10  
15  
16  
15
```

Рис. 16: Некоторые операции для работы с массивами

Массивы

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

Julia

```
10x2 Matrix{Int64}:  
11 13  
16 18  
14 12  
10 13  
20 18  
15 17  
10 10  
15 11  
16 11  
15 11
```

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

Julia

```
10x3 Matrix{Int64}:  
11 20 10  
16 11 12  
14 14 15  
10 13 15  
20 20 19  
15 19 14  
10 19 20  
15 15 10  
16 11 19  
15 20 19
```

Рис. 17: Некоторые операции для работы с массивами

Массивы

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

Julia

```
3x2 Matrix{Int64}:  
10 18  
14 13  
13 17
```

Рис. 18: Некоторые операции для работы с массивами

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

Julia

```
3-element Vector{Int64}:  
20  
10  
13
```

```
sort(ar,dims=1)
```

Julia

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

Рис. 19: Некоторые операции для работы с массивами

Массивы

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

Julia

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

Рис. 20: Некоторые операции для работы с массивами

```
# логическое сражение с числом  
# (результат - массив логических значений):  
ar .> 14
```

Julia

```
10×5 BitMatrix:  
0 0 1 0 0  
0 1 0 0 1  
1 0 0 1 0  
0 0 0 1 0  
1 1 1 1 1  
0 1 1 0 1  
1 0 1 1 0  
0 1 1 0 0  
1 1 0 1 0  
0 1 1 1 0
```

Рис. 21: Некоторые операции для работы с массивами

Массивы

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

Julia

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

Рис. 22: Некоторые операции для работы с массивами

Самостоятельная работа

Выполнение заданий №1 и №2:

№1. Даны множества: $A = \{0, 3, 4, 9\}$, $B = \{1, 3, 4, 7\}$, $C = \{0, 1, 2, 4, 7, 8, 9\}$. Найти $P = A \cap B \cup A \cap C \cup A \cap B \cap C$.

```
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, C), intersect(B, C))
println(P)

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

Julia

№2. Приведите свои примеры с выполнением операций над множествами элементов разных типов.

```
set1 = Set(["hot", "dog"])
set2 = Set(["hot", "cold", "warm"])
intersection = intersect(set1, set2)
println(intersection)

Set(["hot"])
```

Julia

```
set3 = Set([1, 2, 3])
set4 = Set([3, 8, 9])
res = setdiff(set3, set4)
println(res)

Set([2, 1])
```

Julia

Рис. 23: Решение заданий №1 и №2

Самостоятельная работа

№3. Создайте разными способами:

3.1) массив $(1, 2, 3, \dots, N - 1, N)$, N выберите больше 20

```
ar_3_1 = collect(1:27)
println(ar_3_1)
```

Julia

```
[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]
```

3.2) массив $(N, N - 1, \dots, 2, 1)$, N выберите больше 20

```
ar_3_2 = collect(27:-1:1)
println(ar_3_2)
```

Julia

```
[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]
```

3.3) массив $(1, 2, 3, \dots, N - 1, N, N - 1, \dots, 2, 1)$, N выберите больше 20

```
ar_3_3 = vcat(collect(1:27), collect(27:-1:1))
println(ar_3_3)
```

Julia

```
[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, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8
```

Рис. 24: Выполнение подпунктов задания №3

Самостоятельная работа

3.4) массив с именем tmp вида (4, 6, 3)

```
tmp = [4, 6, 3]
println(tmp)
```

Julia

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

3.5) массив, в котором первый элемент массива tmp повторяется 10 раз

```
ar_3_5 = fill(tmp[1], 10)
println(ar_3_5)
```

Julia

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

3.6) массив, в котором все элементы массива tmp повторяются 10 раз

```
ar_3_6 = repeat(tmp, 10)
println(ar_3_6)
```

Julia

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

3.7) массив, в котором первый элемент массива tmp встречается 11 раз, второй элемент — 10 раз, третий элемент — 10 раз

```
ar_3_7 = vcat(fill(tmp[1], 11), fill(tmp[2], 10), fill(tmp[3], 10))
println(ar_3_7)
```

Julia

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

Рис. 25: Выполнение подпунктов задания №3

Самостоятельная работа

3.8) массив, в котором первый элемент массива tmp встречается 10 раз подряд, второй элемент — 20 раз подряд, третий элемент — 30 раз подряд

```
ar_3_8 = vcat(fill(tmp[1], 10), fill(tmp[2], 20), fill(tmp[3], 30))
println(ar_3_8)
```

Julia

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

```

result_array = [2^tmp[i] for i in 1:3]
result_array = vcat(result_array, repeat([2^tmp[3]], 3))

# Преобразуем массив в строку для поиска цифры '6'
result_string = join(result_array, "")
count_6 = count(x -> x == '6', result_string)

# Выводим массив и количество цифр 6
println("Результирующий массив: ", result_array)
println("Количество цифры 6: ", count_6)

```

100

Результирующий массив: [16, 64, 8, 8, 8, 8]
Количество цифры 6: 2

Рис. 26: Выполнение подпунктов задания №3

Самостоятельная работа

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

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

Julia

Среднее значение y: 53.11374594642971

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

```
map(3:3:36, 1:3:34) do i, j  
|> [0.1^i, 0.2^j]  
end
```

Julia

```
12-element Vector{Vector{Float64}}:  
[0.001000000000002, 0.2]  
[1.00000000000004e-6, 0.0016000000000003]  
[1.00000000000005e-9, 1.28000000000005e-5]  
[1.00000000000006e-12, 1.02400000000006e-7]  
[1.00000000000009e-15, 8.1920000000005e-10]  
[1.00000000000001e-18, 6.55360000000005e-12]  
[1.000000000000012e-21, 5.24288000000005e-14]  
[1.000000000000014e-24, 4.194384000000005e-16]  
[1.000000000000015e-27, 3.3554432000000048e-18]  
[1.000000000000017e-30, 2.684354560000004e-20]  
[1.000000000000018e-33, 2.1474836480000035e-22]  
[1.00000000000002e-36, 1.717986918400003e-24]
```

Рис. 27: Выполнение подпунктов задания №3

Самостоятельная работа

3.12) вектор с элементами $(2^i)/i$, $i = 1, 2, \dots, M$, $M = 25$

```
vector12 = [2^i / i for i in 1:25]
println(vector12)
```

Julia

```
[2.0, 2.0, 2.6666666666666665, 4.0, 6.4, 10.666666666666666, 18.285714285714285, 32.0, 56.88888888888886, 102.4, 186.18181818182, 341.3333333333333, 630.1538461538462, 1
```



>

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

```
vector13 = ["fn$i" for i in 1:30]
println(vector13)
```

Julia

```
["fn1", "fn2", "fn3", "fn4", "fn5", "fn6", "fn7", "fn8", "fn9", "fn10", "fn11", "fn12", "fn13", "fn14", "fn15", "fn16", "fn17", "fn18", "fn19", "fn20", "fn21", "fn22", "fn2
```



>

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

```
using Statistics
n = 250
x = [rand(0:999) for i=1:n];
y = [rand(0:999) for i=1:n];
```

Julia

Рис. 28: Выполнение подпунктов задания №3

Самостоятельная работа

```

tmp1 = [y[i+1]-x[i] for i=1:n-1];
print("1: ", tmp1, "\n\n")

tmp2 = [x[i]*2*x[i+1]-x[i+2] for i=1:n-2];
print("2: ", tmp2, "\n\n")

tmp3 = [sin(y[i])/cos(x[i+1]) for i=1:n-1];
print("3: ", tmp3, "\n\n")

tmp4 = sum([exp(-x[i+1])/(x[i]+10) for i=1:n-1]);
print("4: ", tmp4, "\n\n")

tmp5_ind = findall(y,>600);
print("5: ", tmp5_ind, "\n\n")

tmp5_el = filter(e->e>600, y);
print("5: ", tmp5_el, "\n\n")

tmp6 = [x[i] for i in tmp5_ind];
print("6: ", tmp6, "\n\n")

```

100

```

1: [-261, -99, -202, -276, -913, -780, 744, 506, -123, -482, 44, -48, -97, 93, 600, -84, -110, 501, -275, -308, -370, 651, -854, 208, -281, 262, 432, 847, 398, -477, 485, -2: [1482, 428, 1261, 1909, 2632, 722, 364, 952, 1112, 1277, 2018, 134, 1663, 374, 1695, 910, 493, 582, 1549, 2436, 497, 1248, 1972, 1852, 1554, 1251, 86, -105, 1827, 341, 43: [-0.9093335356025565, 0.5066427288381242, 1.2689784283664929, 0.34158622435736347, -0.9243600368201255, 2.779469333021577, 5.170284470625539, -1.969314354105376, 1.364614: 0.0007084828382995724

5: [3, 5, 8, 9, 12, 13, 15, 16, 17, 19, 21, 23, 25, 27, 28, 29, 30, 34, 36, 38, 39, 40, 42, 45, 49, 51, 52, 57, 59, 63, 65, 66, 67, 70, 71, 74, 76, 79, 81, 85, 87, 88, 90, 5: [633, 664, 859, 919, 739, 680, 953, 798, 782, 911, 628, 905, 990, 952, 738, 898, 720, 691, 802, 886, 787, 753, 697, 857, 822, 733, 937, 846, 643, 739, 719, 929, 794, 6746: [318, 937, 413, 577, 728, 133, 190, 866, 227, 554, 877, 888, 480, 396, 51, 322, 800, 343, 423, 639, 850, 659, 776, 80, 322, 94, 49, 298, 27, 848, 331, 796, 674, 986, 271

```

Рис. 29: Выполнение подпунктов задания №3

Самостоятельная работа

```
mean_x = mean(x);
print(mean_x, "\n\n")
tmp7 = [abs(x[i]-mean(x))^(1/2) for i=1:n];
print(tmp7, "\n\n")
tmp8 = sum(y./maximum(y)-200);
print(tmp8, "\n\n")
tmp9_odd = sum(map(isodd, x));
print(tmp9_odd, "\n\n")
tmp9_even = sum(map(iseven, x));
print(tmp9_even, "\n\n")
tmp10 = count(true for i in x if is7 == 0)
print(tmp10, "\n\n")
tmp11 = x[sortperm(y)];
print(tmp11, "\n\n")
tmp12 = last(sort(x), 10);
print(tmp12, "\n\n")
tmp13 = collect(Set(x));
print(tmp13, "\n\n")
```

Julia

```
482.532
```

```
[12.10504027254763, 15.794556024149587, 12.827002767599295, 21.388501583794973, 21.318255088069474, 20.554026369546186, 19.17112411936243, 8.338585011859026, 9.719465005852
```

```
53
```

```
125
```

```
125
```

```
38
```

```
[336, 534, 857, 905, 67, 730, 782, 440, 860, 437, 939, 764, 150, 407, 732, 777, 224, 84, 732, 729, 717, 742, 289, 332, 114, 74, 480, 708, 795, 940, 410, 57, 63, 115, 581, 2
```

```
[940, 941, 942, 951, 963, 964, 965, 969, 984, 986]
```

```
[413, 429, 118, 689, 711, 719, 581, 648, 941, 699, 67, 614, 929, 115, 185, 420, 905, 861, 267, 717, 139, 525, 417, 584, 242, 948, 942, 658, 224, 430, 410, 874, 850, 730, 79]
```

Рис. 30: Выполнение подпунктов задания №3

Самостоятельная работа

№4. Создайте массив squares, в котором будут храниться квадраты всех целых чисел от 1 до 100.

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

Julia

```
[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]
```

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 Pkg
import Primes as pm
```

Julia

```
myprimes = [pm.prime(i) for i=1:168];
prime_89 = pm.prime(89);
prime_89_to_99 = myprimes[89:99];

println(myprimes, "\n", prime_89, "\n", prime_89_to_99)
```

Julia

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173,
461
[461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]
```

Рис. 31: Решение заданий №4, №5 и №6

Самостоятельная работа

6. Вычислите выражения:

6.1

```
sum1 = sum(i^3 + 4*i^2 for i in 10:100)
println(sum1)
```

Julia

26852735

6.2

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

Julia

2.1291704368143802e9

6.3

```
s = 1
sum3 = 1
for i=2:2:38
    s=i/(i+1)
    sum3+=s
end
println(sum3)
```

Julia

6.976346137897618

Рис. 32: Решение заданий №4, №5 и №6

Выводы

В результате выполнения данной лабораторной работы мы изучили несколько структур данных, реализованных в Julia, научились применять их и операции над ними для решения задач.