

Лабораторная работа №2

Компьютерный практикум по статистическому анализу данных

Николаев Д. И.

15 ноября 2023

Российский университет дружбы народов, Москва, Россия

Прагматика выполнения

- Получение навыков работы в Jupyter Notebook;
- Освоение особенностей языка Julia;
- Применение полученных знаний на практике в дальнейшем.

Цели

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

Задачи

1. Используя Jupyter Lab, повторить примеры из раздела 2.2.
2. Выполнить задания для самостоятельной работы (раздел 2.4).

Выполнение работы. Повторение
примеров

Повторение примеров (1)

```
Лабораторная работа № 2. Структуры данных

Раздел 2.2. Повтор заданий.

Кортежи

[1]: # пустой кортеж:
      ()

[1]: ()

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

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

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

[3]: (1, 2, 3)

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

[7]: (1, 2.0, "tmp")

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

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

Рис. 1: Работа с кортежами 1

Повторение примеров (2)

Примеры операций над кортежами:

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

```
[9]: 3
```

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

```
[10]: (1, 2.0, "tmp")
```

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

```
[11]: 5
```

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

```
[12]: (2, 3, 3)
```

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

```
[14]: (true, false)
```

Повторение примеров (3)

▼ Словари

Примеры словарей и операций над ними:

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

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

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

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

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

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

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

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

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

```
[21]: (true, false)
```

Повторение примеров (4)

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

[21]: (true, false)

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

[23]: "555-3344"

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

[25]: ("867-5309", "333-5544")

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

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

Рис. 4: Работа со словарями 2

▼ Множества ¶

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

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

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

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

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

```
[29]: false
```

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

```
[30]: true
```

Повторение примеров (6)

[31]: *# объединение множеств:*

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

[31]: Set{Int64} with 4 elements:

```
4  
2  
3  
1
```

[32]: *# пересечение множеств:*

```
D = intersect(S1, S3)
```

[32]: Set{Int64} with 2 elements:

```
2  
1
```

[33]: *# разность множеств:*

```
E = setdiff(S3, S1)
```

[33]: Set{Int64} with 1 element:

```
3
```

[34]: *# проверка вхождения элементов одного множества в другое:*

```
issubset(S1, S4)
```

[34]: true

[35]: *# добавление элемента в множество:*

```
push!(S4, 99)
```

[35]: Set{Int64} with 4 elements:

```
2  
99  
3  
1
```

Повторение примеров (7)

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

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

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

```
[36]: 2
```

Рис. 7: Работа с множествами 3

Массивы

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

```
[37]: Any[]
```

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

```
[40]: Float64[]
```

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

```
[41]: 3-element Vector{Int64}:  
 1  
 2  
 3
```

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

```
[42]: 1×3 Matrix{Int64}:  
 1  2  3
```

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

```
[45]: 3×3 Matrix{Int64}:  
 1  4  7  
 2  5  8  
 3  6  9
```


Повторение примеров (9)

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

```
[46]: 3x3 Matrix{Int64}:  
      1  2  3  
      4  5  6  
      7  8  9
```

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

```
[50]: 1x8 Matrix{Float64}:  
      0.679921  0.340091  0.821479  0.729865  ...  0.343236  0.891298  0.400566
```

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

```
[52]: 2x3 Matrix{Float64}:  
      0.26461  0.555366  0.0877469  
      0.0027883  0.635858  0.1202
```

Рис. 9: Работа с массивами 2

Повторение примеров (10)

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

```
[53]: 4x3x2 Array{Float64, 3}:  
[:, :, 1] =  
 0.209757  0.148723  0.945074  
 0.371645  0.298075  0.576505  
 0.272831  0.610135  0.891197  
 0.911689  0.798986  0.378871  
  
[:, :, 2] =  
 0.874182  0.992663  0.486733  
 0.074272  0.106636  0.441866  
 0.615368  0.453661  0.41649  
 0.431464  0.24774  0.572056
```

Примеры массивов, заданных некоторыми функциями через включение:

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

```
[54]: 10-element Vector{Float64}:  
 1.0  
 1.4142135623730951  
 1.7320508075688772  
 2.0  
 2.23606797749979  
 2.449489742783178  
 2.6457513110645907  
 2.8284271247461903  
 3.0  
 3.1622776601683795
```

Повторение примеров (11)

```
[55]: # 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]
```

```
[55]: 5-element Vector{Int64}:  
      3  
      27  
      75  
      147  
      243
```

```
[56]: # 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)]
```

```
[56]: 4-element Vector{Int64}:  
      1  
      9  
      49  
      81
```

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

```
[57]: 5-element Vector{Float64}:  
      1.0  
      1.0  
      1.0  
      1.0  
      1.0
```

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

```
[58]: 2x3 Matrix{Float64}:  
      1.0  1.0  1.0  
      1.0  1.0  1.0
```

Повторение примеров (12)

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

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

```
# 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
```

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

```
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)
```

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

Повторение примеров (13)

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

```
[64]: 2x6 Matrix{Int64}:  
 1  3  5  7  9 11  
 2  4  6  8 10 12
```

```
[65]: # транспонирование  
b'
```

```
[65]: 6x2 adjoint(::Matrix{Int64}) with eltype Int64:  
 1  2  
 3  4  
 5  6  
 7  8  
 9 10  
11 12
```

```
[66]: # транспонирование  
c = transpose(b)
```

```
[66]: 6x2 transpose(::Matrix{Int64}) with eltype Int64:  
 1  2  
 3  4  
 5  6  
 7  8  
 9 10  
11 12
```

Повторение примеров (14)

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

```
[67]: 10x5 Matrix{Int64}:  
 18 19 11 16 15  
 12 16 11 17 12  
 12 19 20 12 14  
 11 16 20 15 12  
 14 18 10 15 18  
 17 18 16 17 12  
 19 13 18 10 13  
 18 11 12 12 18  
 12 20 17 20 11  
 16 19 10 20 19
```

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

```
[68]: 10-element Vector{Int64}:  
 19  
 16  
 19  
 16  
 18  
 18  
 13  
 11  
 20  
 19
```

Повторение примеров (15)

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

```
[69]: 10x2 Matrix{Int64}:  
 19 15  
 16 12  
 19 14  
 16 12  
 18 18  
 18 12  
 13 13  
 11 18  
 20 11  
 19 19
```

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

```
[70]: 10x3 Matrix{Int64}:  
 19 11 16  
 16 11 17  
 19 20 12  
 16 20 15  
 18 10 15  
 18 16 17  
 13 18 10  
 11 12 12  
 20 17 20  
 19 10 20
```

Повторение примеров (16)

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

```
[71]: 3x2 Matrix{Int64}:  
      12  12  
      11  12  
      17  12
```

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

```
[72]: 3-element Vector{Int64}:  
      11  
      16  
      15
```

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

```
[73]: 10x5 Matrix{Int64}:  
      11  11  10  10  11  
      12  13  10  12  12  
      12  16  11  12  12  
      12  16  11  15  12  
      14  18  12  15  13  
      16  18  16  16  14  
      17  19  17  17  15  
      18  19  18  17  18  
      18  19  20  20  18  
      19  20  20  20  19
```


Повторение примеров (17)

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

```
[74]: 10x5 Matrix{Int64}:  
 11 15 16 18 19  
 11 12 12 16 17  
 12 12 14 19 20  
 11 12 15 16 20  
 10 14 15 18 18  
 12 16 17 17 18  
 10 13 13 18 19  
 11 12 12 18 18  
 11 12 17 20 20  
 10 16 19 19 20
```

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

```
[75]: 10x5 BitMatrix:  
 1 1 0 1 1  
 0 1 0 1 0  
 0 1 1 0 0  
 0 1 1 1 0  
 0 1 0 1 1  
 1 1 1 1 0  
 1 0 1 0 0  
 1 0 0 0 1  
 0 1 1 1 0  
 1 1 0 1 1
```

Повторение примеров (18)

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

```
[76]: 29-element Vector{CartesianIndex{2}}:  
 CartesianIndex{1, 1}  
 CartesianIndex{6, 1}  
 CartesianIndex{7, 1}  
 CartesianIndex{8, 1}  
 CartesianIndex{10, 1}  
 CartesianIndex{1, 2}  
 CartesianIndex{2, 2}  
 CartesianIndex{3, 2}  
 CartesianIndex{4, 2}  
 CartesianIndex{5, 2}  
 CartesianIndex{6, 2}  
 CartesianIndex{9, 2}  
 CartesianIndex{10, 2}  
 ⋮  
 CartesianIndex{9, 3}  
 CartesianIndex{1, 4}  
 CartesianIndex{2, 4}  
 CartesianIndex{4, 4}  
 CartesianIndex{5, 4}  
 CartesianIndex{6, 4}  
 CartesianIndex{9, 4}  
 CartesianIndex{10, 4}  
 CartesianIndex{1, 5}  
 CartesianIndex{5, 5}  
 CartesianIndex{8, 5}  
 CartesianIndex{10, 5}
```

Выполнение работы.

Самостоятельная работа

Задание 1

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

```
[77]: A = Set([0, 3, 4, 9])
      B = Set([1, 3, 4, 7])
      C = Set([0, 1, 2, 4, 7, 8, 9])

[77]: Set{Int64} with 7 elements:
      0
      4
      7
      2
      9
      8
      1

[91]: P = A ∩ B ∪ A ∩ B ∪ A ∩ C ∪ B ∩ C

[91]: Set{Int64} with 6 elements:
      0
      4
      7
      9
      3
      1

[92]: P1 = A ∩ (B ∪ (A ∩ (B ∪ (A ∩ (C ∪ (B ∩ C)))))

[92]: Set{Int64} with 4 elements:
      0
      4
      9
      3

[94]: print(P), print(P1)

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

[94]: (nothing, nothing)
```

Рис. 19: Задание 1

Задание 2 (1)

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

```
[95]: A1 = Set("ajkhewrhglewha")

[95]: Set{Char} with 9 elements:
      'w'
      'a'
      'e'
      'j'
      'h'
      'g'
      'k'
      'r'
      'l'

[96]: B1 = Set([1 2 3 4 5 6])

[96]: Set{Int64} with 6 elements:
      5
      4
      6
      2
      3
      1

[97]: C1 = Set(["wgegetr", "agsegqe", "wggertgert"])

[97]: Set{String} with 3 elements:
      "wgegetr"
      "wggertgert"
      "agsegqe"
```

Рис. 20: Задание2. Примеры операций с множествами разных типов 1

Задание 2 (2)

```
[98]: A1 ∪ B1  
[98]: Set[Any]()  
[99]: A1 ∪ B1  
[99]: Set[Any] with 15 elements:  
5  
  'w'  
  'a'  
  'e'  
1  
4  
6  
2  
  'j'  
  'h'  
  'g'  
  'k'  
  'r'  
  'l'  
3  
[101]: setdiff(A1, C1)  
[101]: Set[Char] with 9 elements:  
  'w'  
  'a'  
  'e'  
  'j'  
  'h'  
  'g'  
  'k'  
  'r'  
  'l'  
[102]: issubset(A1, B1)  
[102]: false
```

3. Создайте разными способами:

```
[104]: N = 50;
```

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

```
[126]: B31 = [i for i in range(1, N)];  
C31 = collect(1:N);  
# Классическим циклом  
#=D31 = (Int64)[]  
for i in 1:N  
    push!(D31, i)  
end  
print(D31)=#  
A31 = [i for i in 1:N]
```

Рис. 22: Задание 3. Пункт 1 (1)

Задание 3. Пункт 1 (2)

[126]: 50-element Vector{Int64}:

1
2
3
4
5
6
7
8
9
10
11
12
13
:
39
40
41
42
43
44
45
46
47
48
49
50

Задание 3. Пункт 2

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

```
[127]: B32 = [i for i in range(N, 1, step=-1)];  
C32 = collect(N:-1:1);  
# Классическим циклом  
# D32 = (Int64){}  
for i in 1:N  
    push!(D32, i)  
end  
print(D32)=#  
A32 = [i for i in N:-1:1]
```

```
[127]: 50-element Vector{Int64}:
```

```
50  
49  
48  
47  
46  
45  
44  
43  
42  
41  
40  
39  
38  
⋮  
12  
11  
10  
9  
8  
7  
6  
5  
4  
3  
2
```

Задание 3. Пункт 3, 4

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

```
[131]: B33_1 = [1 for i in range(1, N)];
      B33_2 = [1 for i in range(N-1, 1, step=-1)];
      B33 = [B33_1; B33_2];
      C33 = vcat(B33_1, B33_2);
      #C32 = collect(N:-1:1);
      # Классический цикл
      D33 = zeros{Int64, []}
      for i = 1:N
          push!(D33, i)
      end
      for i = N-1:-1:1
          push!(D33, i)
      end
      print(D33)
      #A32 = {i for i in N:-1:1}
```

[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, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 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]

3.4) массив с именем tmp вида {4, 6, 3};

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

[132]: 3-element Vector{Int64}:
4
6
3

Рис. 25: Задание 3. Пункт 3,4

Задание 3. Пункт 5

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

```
[139]: B35 = fill(tmp[1], 10);  
       C35 = repeat([tmp[1]], 10);  
       A35 = [tmp[1] for i in 1:10]
```

```
[139]: 10-element Vector{Int64}:  
       4  
       4  
       4  
       4  
       4  
       4  
       4  
       4  
       4  
       4
```

Рис. 26: Задание 3. Пункт 5

Задание 3. Пункт 6

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

```
[151]: B36 = fill(tmp, 10);  
C36_1 = repeat(tmp, 10); # В один столбец (30 x 1)
```

```
[152]: C36_2 = repeat(tmp, 1, 10) # В строки (3 x 10)
```

```
[152]: 3x10 Matrix{Int64}:  
 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
```

```
[153]: A36 = [tmp for i in 1:10]
```

```
[153]: 10-element Vector{Vector{Int64}}:  
 [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]
```

Задание 3. Пункт 7

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

```
[155]: B37_1 = fill(tmp[1], 11);  
       B37_2 = fill(tmp[2], 10);  
       B37_3 = fill(tmp[3], 10);  
       B37 = [B37_1; B37_2; B37_3]
```

```
[155]: 31-element Vector{Int64}:  
       4  
       4  
       4  
       4  
       4  
       4  
       4  
       4  
       4  
       4  
       4  
       6  
       6  
       :  
       6  
       6  
       3  
       3  
       3  
       3  
       3  
       3  
       3  
       3  
       3  
       3
```

Рис. 28: Задание 3. Пункт 7

Задание 3. Пункт 8 (1)

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

```
[158]: B38_1 = fill(tmp[1], 10);
      B38_2 = fill(tmp[2], 20);
      B38_3 = fill(tmp[3], 30);
      B38 = [B38_1; B38_2; B38_3]
```

```
[158]: 60-element Vector{Int64}:
```

4
4
4
4
4
4
4
4
4
4
6
6
6
:
3
3
3
3
3
3
3
3
3
3
3

Рис. 29: Задание 3. Пункт 8 (1)

Задание 3. Пункт 1 (2)

```
[169]: A38 = (Int64)[
  for i in 1:10
    push!(A38, tmp[1])
  end
  for i in 1:20
    push!(A38, tmp[2])
  end
  for i in 1:30
    push!(A38, tmp[3])
  end
  A38
```

```
[169]: 60-element Vector{Int64}:
```

4
4
4
4
4
4
4
4
4
4
6
6
6
:
3
3
3
3
3
3
3
3
3
3

Задание 3. Пункт 9

3.9) массив из элементов вида 2^{n+i} , $i = 1, 2, 3$, где элемент 2^{n+i} встречается 4 раза; посчитайте в полученном векторе, сколько раз встречается цифра 6, и выведите это значение на экран;

```
[189]: A39 = [2**(tmp[i] * 3 + 1)] for i in 0:11]
```

```
[189]: 12-element Vector{Int64}:
```

```
16
64
8
16
64
8
16
64
8
16
64
8
```

```
[191]: function digit_num(d, a)
      # Находим число цифр d в числе a
      n = 0
      while a > 0
          if a % 10 == d
              n += 1
          end
          a = a ÷ 10
      end
      return n
  end
```

```
[191]: digit_num (generic function with 1 method)
```

```
[198]: N1 = sum([digit_num(6, a) for a in A39]);
      N2 = reduce(+, [digit_num(6, a) for a in A39])
```

```
[198]: 8
```

Рис. 31: Задание 3. Пункт 9

Задание 3. Пункт 10

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

```
[203]: A310 = [exp(x)*cos(x) for x in 3:0.1:6]
```

```
[203]: 31-element Vector{Float64}:
```

```
-19.884530844146987  
-22.178753389342127  
-24.490696732801293  
-26.77318244299338  
-28.969237768093574  
-31.011186439374516  
-32.819774760338504  
-34.30336011037369  
-35.35719361853035  
-35.86283371230767  
-35.68773248011913  
-34.68504225166807  
-32.693695428321746  
⋮  
25.046704998273004  
42.09920106253839  
61.99663027669454  
84.92906736250268  
111.0615860420258  
140.5250750527875  
173.40577640857734  
209.73349424783467  
249.46844055885668  
292.4867067371223  
338.5643778585117  
387.36034029093076
```

```
[204]: Avg_310 = sum(A310)/length(A310)
```

```
[204]: 53.11374594642971
```

Задание 3. Пункт 11

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

```
[209]: x = 0.1
      y = 0.2
      A311 = [(x^i, y^j) for i in 3:3:36 for j in 1:3:34] # j - внутренний цикл
```

```
[209]: 144-element Vector{Tuple{Float64, Float64}}:
      (0.0010000000000000002, 0.2)
      (0.0010000000000000002, 0.0016000000000000003)
      (0.0010000000000000002, 1.2800000000000006e-5)
      (0.0010000000000000002, 1.0240000000000006e-7)
      (0.0010000000000000002, 8.192000000000005e-10)
      (0.0010000000000000002, 6.5536000000000055e-12)
      (0.0010000000000000002, 5.2428800000000056e-14)
      (0.0010000000000000002, 4.194304000000005e-16)
      (0.0010000000000000002, 3.3554432000000044e-18)
      (0.0010000000000000002, 2.6843545600000004e-20)
      (0.0010000000000000002, 2.1474836480000004e-22)
      (0.0010000000000000002, 1.7179869184000035e-24)
      (1.0000000000000004e-6, 0.2)
      ⋮
      (1.0000000000000002e-36, 0.2)
      (1.0000000000000002e-36, 0.0016000000000000003)
      (1.0000000000000002e-36, 1.2800000000000006e-5)
      (1.0000000000000002e-36, 1.0240000000000006e-7)
      (1.0000000000000002e-36, 8.192000000000005e-10)
      (1.0000000000000002e-36, 6.5536000000000055e-12)
      (1.0000000000000002e-36, 5.2428800000000056e-14)
      (1.0000000000000002e-36, 4.194304000000005e-16)
      (1.0000000000000002e-36, 3.3554432000000044e-18)
      (1.0000000000000002e-36, 2.6843545600000004e-20)
      (1.0000000000000002e-36, 2.1474836480000004e-22)
      (1.0000000000000002e-36, 1.7179869184000035e-24)
```

Задание 3. Пункт 12

3.12) вектор с элементами $\frac{2^i}{i}, i = 1, 2, \dots, M, M = 25$;

```
[210]: M = 25  
A312 = [2^i / i for i in 1:M]
```

```
[210]: 25-element Vector{Float64}:
```

```
 2.0  
 2.0  
 2.6666666666666665  
 4.0  
 6.4  
10.666666666666666  
18.285714285714285  
32.0  
56.888888888888886  
102.4  
186.1818181818182  
341.3333333333333  
630.1538461538462  
1170.2857142857142  
2184.5333333333333  
4096.0  
7710.117647058823  
14563.555555555555  
27594.105263157893  
52428.8  
99864.38095238095  
190650.18181818182  
364722.0869565217  
699050.6666666666  
1.34217728e6
```

Задание 3. Пункт 13

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

```
[212]: N = 30  
A313 = ["fn%i" for i in 1:N]
```

```
[212]: 30-element Vector{String}:
```

```
"fn1"  
"fn2"  
"fn3"  
"fn4"  
"fn5"  
"fn6"  
"fn7"  
"fn8"  
"fn9"  
"fn10"  
"fn11"  
"fn12"  
"fn13"  
⋮  
"fn19"  
"fn20"  
"fn21"  
"fn22"  
"fn23"  
"fn24"  
"fn25"  
"fn26"  
"fn27"  
"fn28"  
"fn29"  
"fn30"
```

Задание 3. Пункт 14. Подпункт 1 (1)

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 $0, 1, \dots, 999$; на его основе:

```
[223]: x = [rand(0:999) for i in 1:n]
```

```
[223]: 250-element Vector{Int64}:
```

```
304
903
949
531
245
968
953
414
692
717
46
582
146
⋮
919
718
407
231
59
246
756
199
322
939
498
297
```

Рис. 36: Задание 3. Пункт 14. Подпункт 1 (1)

Задание 3. Пункт 14. Подпункт 1 (2)

```
[224]: y = [rand(0:999) for i in 1:n]
```

```
[224]: 250-element Vector{Int64}:
```

```
752
```

```
859
```

```
49
```

```
809
```

```
904
```

```
814
```

```
511
```

```
168
```

```
40
```

```
411
```

```
447
```

```
86
```

```
448
```

```
⋮
```

```
987
```

```
285
```

```
804
```

```
612
```

```
602
```

```
410
```

```
36
```

```
677
```

```
887
```

```
614
```

```
92
```

```
391
```

Задание 3. Пункт 14. Подпункт 2

– сформируйте вектор $(y_2 - x_1, \dots, y_n - x_{n-1})$;

```
[225]: y_minus_x = [y[i] - x[i-1] for i in 2:n]
```

```
[225]: 249-element Vector{Int64}:
```

555

-854

-140

373

569

-457

-785

-374

-281

-270

40

-134

761

⋮

237

-634

86

205

371

351

-210

-79

688

292

-847

-107

Задание 3. Пункт 14. Подпункт 3

– сформируйте вектор $(x_1 + 2x_2 - x_3, x_2 + 2x_3 - x_4, \dots, x_{n-2} + 2x_{n-1} - x_n)$;

```
[226]: x1_plus_2x2_minus_x3 = [x[i] + 2*x[i+1] - x[i+2] for i in 1:n-2]
```

```
[226]: 248-element Vector{Int64}:
```

1161

2270

1766

53

1228

2460

1089

1081

2080

227

1064

28

1026

⋮

1245

1870

1948

1301

810

103

-205

1559

832

-96

1702

1638

Задание 3. Пункт 14. Подпункт 4

– сформируйте вектор $(\frac{\sin(y_1)}{\cos(x_2)}, \frac{\sin(y_2)}{\cos(x_3)}, \dots, \frac{\sin(y_{n-1})}{\cos(x_n)})$

```
[227]: siny_divide_cosx = [sin(y[i-1]) / cos(x[i]) for i in 2:n]
```

```
[227]: 249-element Vector{Float64}:
```

```
 4.440762796604472
-1.0031818070105352
 0.9561508827440894
-1.000182463515962
-0.7591972657062622
 0.7056098258205616
 1.1435236257292338
-1.5102475163473301
 0.9883937960184693
-1.2067373827245433
-1.1250376112108684
-10.998864247308298
-1.5481986411729924
 ⋮
11.86408519204383
-3.5315825458429986
 4.747080802129639
-2.641994690524218
-0.743544523690401
-1.606294475559757
-2.3130027041569248
 2.1028939413502497
-75.48444446193533
-0.9296670498176479
17.096364201961297
 6.538592779718017
```

Задание 3. Пункт 14. Подпункт 5

– вычислите $\sum_{i=1}^{n-1} \frac{e^{-x_{i+1}}}{x_i + 10}$

```
[228]: S314_array = [exp(-x[i+1])/(x[i]+10) for i in 1:n-1]
```

```
[228]: 249-element Vector{Float64}:
 0.0
 0.0
 2.5574756191608885e-234
 7.3224205570088654e-110
 0.0
 0.0
 1.6537056299290693e-183
 6.931906024237895e-304
 5.81459356e-315
 1.448503075316893e-23
 3.1075884977133967e-255
 6.617347406166209e-67
 0.0
 :
 0.0
 1.616390463e-315
 2.3989129343706285e-180
 1.1424537443123088e-103
 9.876624102466392e-29
 2.1120702587699017e-109
 0.0
 4.910993186809479e-90
 6.871167586659815e-143
 0.0
 5.547301870445639e-220
 2.0355190089610133e-132
```

```
[229]: S314 = sum(S314 array)
```

Задание 3. Пункт 14. Подпункт 6 (1)

– выберите элементы вектора y , значения которых больше 600, и выведите на экран; определите индексы этих элементов;

```
[235]: y_bigger_than_600 = [y_ for y_ in y if y_ > 600]
```

```
[235]: 113-element Vector{Int64}:
```

```
752  
859  
809  
904  
814  
907  
664  
629  
778  
870  
690  
835  
622  
⋮  
759  
837  
852  
630  
887  
987  
804  
612  
602  
677  
887  
614
```

Рис. 42: Задание 3. Пункт 14. Подпункт 6 (1)

Задание 3. Пункт 14. Подпункт 6 (2)

```
[236]: indexes_y_bigger_than_600 = findall(y .> 600)
```

```
[236]: 113-element Vector{Int64}:
```

```
 1  
 2  
 4  
 5  
 6  
14  
15  
16  
17  
18  
19  
20  
21  
 ⋮  
227  
229  
232  
233  
235  
239  
241  
242  
243  
246  
247  
248
```

Задание 3. Пункт 14. Подпункт 7

– определите значения вектора x, соответствующие значениям вектора u, значения которых больше 600 (под соответствием понимается расположение на аналогичных индексных позициях);

```
[238]: x_with_indexes_as_y_bigger_than_600 * [x[i] for i in indexes_y_bigger_than_600]
```

```
[238]: 113-element Vector{Int64}:
```

```
304
903
531
245
968
846
812
204
57
7
339
618
913
:
976
239
133
787
274
919
407
231
59
199
322
939
```

Рис. 44: Задание 3. Пункт 14. Подпункт 7

Задание 3. Пункт 14. Подпункт 8

– сформируйте вектор $(|x_1 - \bar{x}|, |x_2 - \bar{x}|, \dots, |x_n - \bar{x}|)$, где \bar{x} обозначает среднее значение вектора $x = (x_1, x_2, \dots, x_n)$

```
[239]: avg_x = sum(x)/length(x)
```

```
[239]: 500.776
```

```
[240]: abs_x_minus_avgx = [abs(x[i] - avg_x) for i in 1:n]
```

```
[240]: 250-element Vector{Float64}:
```

```
196.776
402.224
448.224
30.223999999999999
255.776
467.224
452.224
86.776000000000001
191.224
216.224
454.776
81.223999999999999
354.776
⋮
418.224
217.224
93.776000000000001
269.776
441.776
254.776
255.224
301.776
178.776
438.224
2.77600000000000105
203.776
```

Задание 3. Пункт 14. Подпункт 9 (1)

– определите, сколько элементов вектора y отстоят от максимального значения не более, чем на 200;

```
[241]: y_max = maximum(y)
```

```
[241]: 993
```

```
[242]: y_dist_less_than_maxy_by200 = [y_ for y_ in y if abs(y_ - y_max) <= 200]
```

```
[242]: 54-element Vector{Int64}:
```

```
859
```

```
809
```

```
904
```

```
814
```

```
907
```

```
870
```

```
835
```

```
852
```

```
803
```

```
924
```

```
798
```

```
881
```

```
866
```

```
⋮
```

```
952
```

```
836
```

```
934
```

```
978
```

```
941
```

```
950
```

```
837
```

```
852
```

```
887
```

```
987
```

```
804
```

```
887
```

```
[243]: number_of_y_dist_less_than_maxy_by200 = length(y_dist_less_than_maxy_by200)  
[243]: 54
```

Рис. 47: Задание 3. Пункт 14. Подпункт 9 (2)

Задание 3. Пункт 14. Подпункт 10

– определите, сколько чётных и нечётных элементов вектора x ;

```
[246]: number_of_odd_x = count(i->(i % 2 == 1), x)
```

```
[246]: 130
```

```
[248]: number_of_even_x = length([i for i in x if i % 2 == 0])
```

```
[248]: 120
```

Рис. 48: Задание 3. Пункт 14. Подпункт 10

– определите, сколько элементов вектора x кратны 7;

```
[249]: number_of_x_divisible_by_7 = length([x_ for x_ in x if x_ % 7 == 0])
```

```
[249]: 39
```

Рис. 49: Задание 3. Пункт 14. Подпункт 11

Задание 3. Пункт 14. Подпункт 12 (1)

– отсортируйте элементы вектора x в порядке возрастания элементов вектора y ;

```
[255]: y_ascending_order_indexes = sortperm(y) # Индексы отсортированного по возрастанию массива y
```

```
[255]: 250-element Vector{Int64}:
```

```
53  
174  
170  
29  
237  
245  
9  
214  
90  
130  
223  
22  
3  
⋮  
224  
57  
225  
185  
189  
138  
211  
78  
145  
239  
102  
120
```

Задание 3. Пункт 14. Подпункт 12 (2)

```
[262]: #x_sorted_by_indexes_of_y_ascending_order = [x[i] for i in y_ascending_order_indexes]  
x_sorted_by_indexes_of_y_ascending_order = x[sortperm(y)]
```

```
[262]: 250-element Vector{Int64}:
```

```
307  
40  
330  
285  
664  
756  
692  
102  
96  
710  
831  
733  
949  
⋮  
251  
704  
350  
332  
739  
593  
993  
700  
294  
919  
769  
884
```

Задание 3. Пункт 14. Подпункт 13

– выведите элементы вектора x , которые входят в десятку наибольших (top-10)

```
[258]: top_10_x = sort(x)[end-9:end]
```

```
[258]: 10-element Vector{Int64}:
```

```
 968
```

```
 970
```

```
 976
```

```
 977
```

```
 984
```

```
 985
```

```
 989
```

```
 993
```

```
 994
```

```
 995
```

Рис. 52: Задание 3. Пункт 14. Подпункт 13

Задание 3. Пункт 14. Подпункт 14

– сформируйте вектор, содержащий только уникальные (неповторяющиеся) элементы вектора x

```
[259]: x_unique = unique(x)
```

```
[259]: 228-element Vector{Int64}:
```

```
304
903
949
531
245
968
953
414
692
717
46
582
146
⋮
919
718
407
231
59
246
756
199
322
939
498
297
```

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

```
[260]: squares = [i^2 for i in 1:100]
```

```
[260]: 100-element Vector{Int64}:
```

```
 1  
 4  
 9  
16  
25  
36  
49  
64  
81  
100  
121  
144  
169  
⋮  
7921  
8100  
8281  
8464  
8649  
8836  
9025  
9216  
9409  
9604  
9801  
10000
```

Рис. 54: Задание 4. Квадраты чисел от 1 до 100

Задание 5 (1)

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.

```
[261]: import Pkg
      Pkg.add("Primes")
      using Primes

      Updating registry at "C:\Users\User\.julia\registries\General.toml"
      Resolving package versions...
      Installed Primes v0.5.5
      Installed IntegerMathUtils v0.1.2
      Updating "C:\Users\User\.julia\environments\v1.8\Project.toml"
      [27ebfcd6] + Primes v0.5.5
      Updating "C:\Users\User\.julia\environments\v1.8\Manifest.toml"
      [18e54dd8] + IntegerMathUtils v0.1.2
      [27ebfcd6] + Primes v0.5.5
      Precompiling project...
      ✓ IntegerMathUtils
      ✓ Primes
      2 dependencies successfully precompiled in 31 seconds. 28% already precompiled.
```

Рис. 55: Задание 5. Подключение пакета Primes

Задание 5 (2)

```
[264]: myprimes = primes(1000)
```

```
[264]: 168-element Vector{Int64}:
```

```
 2
```

```
 3
```

```
 5
```

```
 7
```

```
11
```

```
13
```

```
17
```

```
19
```

```
23
```

```
29
```

```
31
```

```
37
```

```
41
```

```
⋮
```

```
919
```

```
929
```

```
937
```

```
941
```

```
947
```

```
953
```

```
967
```

```
971
```

```
977
```

```
983
```

```
991
```

```
997
```

```
[265]: myprimes[89]
```

```
[265]: 461
```

```
[266]: myprimes[89:99]
```

```
[266]: 11-element Vector{Int64}:
```

```
461
```

```
463
```

```
467
```

```
479
```

```
487
```

```
491
```

```
499
```

```
503
```

```
509
```

```
521
```

```
523
```

6. Вычислите следующие выражения:

$$6.1) \sum_{i=10}^{100} (i^3 + 4i^2)$$

```
[267]: S61 = sum([i^3 + 4*i^2 for i in 10:100])
```

```
[267]: 26852735
```

Рис. 58: Задание 6. Пункт 1

$$6.2) \sum_{i=1}^M \left(\frac{2^i}{i} + \frac{3^i}{i^2} \right), \text{ где } M = 25$$

```
[270]: S62 = reduce(+, [2^i / i + 3^i / i^2 for i in 1:M])
```

```
[270]: 2.1291704368143802e9
```

Рис. 59: Задание 6. Пункт 2

Задание 6. Пункт 3 (1)

$$6.3) 1 + \frac{2}{3} + (\frac{2}{3} \frac{4}{5}) + (\frac{2}{3} \frac{4}{5} \frac{6}{7}) + \dots + (\frac{2}{3} \frac{4}{5} \dots \frac{38}{39})$$

```
[279]: [1 - 1//i for i in 3:2:39]
```

```
[279]: 19-element Vector{Rational{Int64}}:
```

```
 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  
30//31  
32//33  
34//35  
36//37  
38//39
```

Задание 6. Пункт 3 (2)

```
[280]: cumprod([1 - 1//i for i in 3:2:39])
```

```
[280]: 19-element Vector{Rational{Int64}}:
```

```
      2//3
      8//15
     16//35
    128//315
    256//693
   1024//3003
   2048//6435
  32768//109395
 65536//230945
262144//969969
524288//2028117
4194304//16900975
8388608//35102025
33554432//145422675
67108864//300540195
2147483648//9917826435
4294967296//20419054425
17179869184//83945001525
34359738368//172308161025
```

```
[288]: S63 = 1 + reduce(+, cumprod([1 - 1/i for i in 3:2:39]))
```

```
[288]: 6.976346137897621
```

Результаты

В ходе работы я изучил структуры данных, реализованные в Julia, и операция над ними для решения практических задач