

Отчет по лабораторной работе №2

**Дисциплина: Компьютерный практикум по статистическому анализу
данных**

Лобанова Полина Иннокентьевна

Содержание

1	Цель работы	6
2	Задание	7
3	Выполнение лабораторной работы	8
4	Выводы	32
	Список литературы	33

Список иллюстраций

3.1	Примеры создания кортежей и операций над ними	8
3.2	Примеры создания кортежей и операций над ними	9
3.3	Примеры создания словарей и операций над ними	9
3.4	Примеры создания словарей и операций над ними	10
3.5	Примеры создания множеств и операций над ними	10
3.6	Примеры создания множеств и операций над ними	11
3.7	Примеры создания множеств и операций над ними	11
3.8	Примеры создания массивов и операций над ними	12
3.9	Примеры создания массивов и операций над ними	12
3.10	Примеры создания массивов и операций над ними	13
3.11	Примеры создания массивов и операций над ними	13
3.12	Примеры создания массивов и операций над ними	14
3.13	Примеры создания массивов и операций над ними	14
3.14	Примеры создания массивов и операций над ними	15
3.15	Примеры создания массивов и операций над ними	15
3.16	Примеры создания массивов и операций над ними	16
3.17	Примеры создания массивов и операций над ними	16
3.18	Задание 1	17
3.19	Задание 2	17
3.20	Задание 3.1	17
3.21	Задание 3.2	18
3.22	Задание 3.3	18
3.23	Задание 3.4	18
3.24	Задание 3.5	18
3.25	Задание 3.6	19
3.26	Задание 3.7	19
3.27	Задание 3.8	19
3.28	Задание 3.9	20
3.29	Задание 3.10	20
3.30	Задание 3.11	20
3.31	Задание 3.12	21
3.32	Задание 3.13	22
3.33	Задание 3.14	23
3.34	Задание 3.15	23
3.35	Задачи	24
3.36	Задание 3.16	24
3.37	Задание 3.17	24

3.38 Задание 3.18	25
3.39 Задание 3.19	25
3.40 Задачи	26
3.41 Задание 3.20	26
3.42 Задание 3.21	26
3.43 Задание 3.22	27
3.44 Задание 3.23	27
3.45 Задание 3.24	27
3.46 Задание 3.25	28
3.47 Задание 4	29
3.48 Задание 5	30
3.49 Задачи	30
3.50 Задание 6	31

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 раздела 2.2.
2. Выполните задания для самостоятельной работы (раздел 2.4).

3 Выполнение лабораторной работы

1. Повторила примеры создания кортежей и операций над ними.

```
•[2]: #пустой кортеж
      ()

[2]: ()

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

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

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

[4]: (1, 2, 3)

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

[6]: (1, 2.0, "tmp")

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

[7]: (a = 2, b = 3)

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

[9]: 3

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

Рис. 3.1: Примеры создания кортежей и операций над ними


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

[14]: 5

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

[15]: (2, 3, 3)

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

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

Рис. 3.2: Примеры создания кортежей и операций над ними

2. Повторила примеры словарей и операций над ними.

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

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

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

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

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

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

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

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

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

[33]: true
```

Рис. 3.3: Примеры создания словарей и операций над ними

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

[34]: Dict{String, Any} with 3 entries:
"Сидоров П.С." => "555-3344"
"Бухгалтерия" => "555-2368"
"Иванов И.И." => ("867-5309", "333-5544")

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

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

[38]: # Объединение словарей (функция merge()):
a = Dict{"foo" => 0.0, "bar" => 42.0};
b = Dict{"baz" => 17, "bar" => 13.0};
print(merge(a, b), '\n', merge(b, a))

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 создания словарей и операций над ними

3. Повторила примеры множеств и операций над ними.

```
[63]: # создать множество из четырёх целочисленных значений:
A = Set{[1, 3, 4, 5]}

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

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

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

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

[65]: false

[66]: S3 = Set{[1, 2, 2, 3, 1, 2, 3, 2, 1]};
S4 = Set{[2, 3, 1]};
issetequal(S3,S4)

[66]: true
```

Рис. 3.5: Примеры создания множеств и операций над ними

```

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

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

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

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

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

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

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

[70]: true

```

Рис. 3.6: Примеры создания множеств и операций над ними

```

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

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

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

[72]: 2

```

Рис. 3.7: Примеры создания множеств и операций над ними

4. Повторила примеры массивов и операций над ними.

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

```
[73]: Any[]
```

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

```
[76]: Float64[]
```

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

```
[77]: 3-element Vector{Int64}:  
 1  
 2  
 3
```

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

```
[78]: 1×3 Matrix{Int64}:  
 1  2  3
```

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

```
[79]: 3×3 Matrix{Int64}:  
 1  4  7  
 2  5  8  
 3  6  9
```

Рис. 3.8: Примеры создания массивов и операций над ними

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

```
[80]: 3×3 Matrix{Int64}:  
 1  2  3  
 4  5  6  
 7  8  9
```

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

```
[81]: 1×8 Matrix{Float64}:  
 0.0208143  0.437316  0.840603  0.912453 ... 0.152981  0.225446  0.752386
```

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

```
[82]: 2×3 Matrix{Float64}:  
 0.842003  0.69886  0.00400994  
 0.974391  0.969261  0.703589
```

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

```
[83]: 4×3×2 Array{Float64, 3}:  
[:, :, 1] =  
 0.95166  0.966101  0.243843  
 0.512624  0.740958  0.790691  
 0.699691  0.565528  0.487775  
 0.0969832  0.339082  0.22116
```

```
[:, :, 2] =  
 0.979651  0.610078  0.267444  
 0.909902  0.736681  0.105966  
 0.914645  0.707693  0.962816  
 0.695719  0.875981  0.823412
```

Рис. 3.9: Примеры создания массивов и операций над ними

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

[84]: 10-element Vector{Float64}:
 1.0
 1.4142135623730951
 1.7320508075688772
 2.0
 2.23606797749979
 2.449489742783178
 2.6457513110645907
 2.8284271247461903
 3.0
 3.1622776601683795

[85]: # 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]

[85]: 5-element Vector{Int64}:
 3
 27
 75
 147
 243

[87]: # 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)]

[87]: 4-element Vector{Int64}:
 1
 9
 49
 81
```

Рис. 3.10: Примеры создания массивов и операций над ними

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

[91]: 5-element Vector{Float64}:
 1.0
 1.0
 1.0
 1.0
 1.0

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

[92]: 2x3 Matrix{Float64}:
 1.0  1.0  1.0
 1.0  1.0  1.0

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

[93]: 4-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0

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

[94]: 3x2 Matrix{Float64}:
 3.5  3.5
 3.5  3.5
 3.5  3.5
```

Рис. 3.11: Примеры создания массивов и операций над ними

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

[97]: 3x6 Matrix{Int64}:
      1  2  1  2  1  2
      1  2  1  2  1  2
      1  2  1  2  1  2

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

[99]: 2x6 Matrix{Int64}:
      1  3  5  7  9 11
      2  4  6  8 10 12

100]: # транспонирование
      b'

100]: 6x2 adjoint(::Matrix{Int64}) with eltype Int64:
      1  2
      3  4
      5  6
      7  8
      9 10
      11 12

101]: c = transpose(b)

101]: 6x2 transpose(::Matrix{Int64}) with eltype Int64:
      1  2
      3  4
      5  6
      7  8
      9 10
      11 12
```

Рис. 3.12: Примеры создания массивов и операций над ними

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

[102]: 10x5 Matrix{Int64}:
      11 12 14 14 12
      11 14 16 14 13
      20 12 14 14 17
      14 10 17 11 19
      17 11 19 19 15
      12 19 13 13 14
      20 13 16 15 12
      19 10 20 17 15
      18 20 12 11 14
      17 14 11 14 14

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

[103]: 10-element Vector{Int64}:
      12
      14
      12
      10
      11
      19
      13
      10
      20
      14
```

Рис. 3.13: Примеры создания массивов и операций над ними

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

```
[104]: 10x2 Matrix{Int64}:  
12 12  
14 13  
12 17  
10 19  
11 15  
19 14  
13 12  
10 15  
20 14  
14 14
```

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

```
[106]: 10x3 Matrix{Int64}:  
12 14 14  
14 16 14  
12 14 14  
10 17 11  
11 19 19  
19 13 13  
13 16 15  
10 20 17  
20 12 11  
14 11 14
```

Рис. 3.14: Примеры создания массивов и операций над ними

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

```
[107]: 3x2 Matrix{Int64}:  
11 13  
14 19  
12 14
```

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

```
[109]: 3-element Vector{Int64}:  
14  
14  
12
```

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

```
[110]: 10x5 Matrix{Int64}:  
11 10 11 11 12  
11 10 12 11 12  
12 11 13 13 13  
14 12 14 14 14  
17 12 14 14 14  
17 13 16 14 14  
18 14 16 14 15  
19 14 17 15 15  
20 19 19 17 17  
20 20 20 19 19
```

Рис. 3.15: Примеры создания массивов и операций над ними

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

[111]: 10x5 Matrix{Int64}:
 11 12 12 14 14
 11 13 14 14 16
 12 14 14 17 20
 10 11 14 17 19
 11 15 17 19 19
 12 13 13 14 19
 12 13 15 16 20
 10 15 17 19 20
 11 12 14 18 20
 11 14 14 14 17

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

[112]: 10x5 BitMatrix:
 0 0 0 0 0
 0 0 1 0 0
 1 0 0 0 1
 0 0 1 0 1
 1 0 1 1 1
 0 1 0 0 0
 1 0 1 1 0
 1 0 1 1 1
 1 1 0 0 0
 1 0 0 0 0
```

Рис. 3.16: Примеры создания массивов и операций над ними

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

[113]: 20-element Vector{CartesianIndex{2}}:
 CartesianIndex(3, 1)
 CartesianIndex(5, 1)
 CartesianIndex(7, 1)
 CartesianIndex(8, 1)
 CartesianIndex(9, 1)
 CartesianIndex(10, 1)
 CartesianIndex(6, 2)
 CartesianIndex(9, 2)
 CartesianIndex(2, 3)
 CartesianIndex(4, 3)
 CartesianIndex(5, 3)
 CartesianIndex(7, 3)
 CartesianIndex(8, 3)
 CartesianIndex(5, 4)
 CartesianIndex(7, 4)
 CartesianIndex(8, 4)
 CartesianIndex(3, 5)
 CartesianIndex(4, 5)
 CartesianIndex(5, 5)
```

Рис. 3.17: Примеры создания массивов и операций над ними

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


```
[115]:
#1
A = Set([0, 3, 4, 9]);
B = Set([1, 3, 4, 7]);
C = Set([0, 1, 2, 4, 7, 8, 9]);
x1 = intersect(A,B);
x2 = union(x1,A);
x3 = intersect(x2, B);
x4 = union(x3, A);
x5 = intersect(x4, C);
x6 = union(x5, B);
P = intersect(x6, C)

[115]:
Set{Int64} with 5 elements:
 0
 4
 7
 9
 1
```

Рис. 3.18: Задание 1

6. Привела свои примеры с выполнением операций над множествами элементов разных типов.

```
#2
D = Set(["ad", "fg"]);
E = Set(["er", "ad"]);
x7 = union(D, E);
x8 = intersect(D, E);
x9 = setdiff(D, E);
print("Объединение: ", x7, '\n', "Пересечение: ", x8, '\n', "Разность: ", x9)

Объединение: Set(["ad", "er", "fg"])
Пересечение: Set(["ad"])
Разность: Set(["fg"])
```

Рис. 3.19: Задание 2

7. Создала разными способами:

массив (1, 2, 3, ... N - 1, N), N = 25;

```
#3
#3.1
N = 25;
ar1 = [i for i in 1:N];
ar2 = collect(1:N);
print(ar1, '\n', ar2)

[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]
[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]
```

Рис. 3.20: Задание 3.1

массив (N, N - 1 ..., 2, 1);

```
#3.2
ar3 = [i for i in N:-1:1];
ar4 = collect(N:-1:1);
print(ar3, '\n', ar4)

[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]
[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]
```

Рис. 3.21: Задание 3.2

массив $(1, 2, 3, \dots, N - 1, N, N - 1, \dots, 2, 1)$;

```
[146]:
#3.3
ar5 = vcat([i for i in 1:N], [i for i in N-1:-1:1]);
print(ar5)

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

Рис. 3.22: Задание 3.3

массив с именем tmp вида $(4, 6, 3)$;

```
[147]:
#3.4
tmp = [4,6,3]

[147]:
3-element Vector{Int64}:
 4
 6
 3
```

Рис. 3.23: Задание 3.4

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

```
[153]:
#3.5
ar6 = fill(tmp[1], (1, 10))

[153]:
1×10 Matrix{Int64}:
 4 4 4 4 4 4 4 4 4 4
```

Рис. 3.24: Задание 3.5

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

```
[154]:
#3.6
ar7=vcats(fill(tmp[1], (1, 10)), fill(tmp[2], (1, 10)), fill(tmp[3], (1, 10)))

[154]:
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
```

Рис. 3.25: Задание 3.6

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

```
[157]:
#3.7
ar8=vcats(fill.(tmp, [11,10,10]))

[157]:
3-element Vector{Vector{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]
```

Рис. 3.26: Задание 3.7

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

```
[158]:
#3.8
ar9=vcats(fill.(tmp, [10,20,30]))

[158]:
3-element Vector{Vector{Int64}}:
 [4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
 [6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6]
 [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

Рис. 3.27: Задание 3.8

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

```
[167]:
#3.9
tmp1 = [2^(tmp[i]) for i in 1:3];
ar10 = vcat(fill.(tmp1, [3,2,4]));
count=0
for i in ar10
    if i==6
        count+=1
    else
        continue
    end
end
print(count)
```

Рис. 3.28: Задание 3.9

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

```
[186]:
#3.10
y = [exp(x)*cos(x) for x in 3:0.1:6]
sum=0
for i in y
    sum+=i
end
avg=sum/length(y)
```

Рис. 3.29: Задание 3.10

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

```
[199]:
#3.11
ar11=[]
for j in 1:3:34
    i=j+2
    push!(ar11, (0.1^i, 0.2^j))
end
print(ar11)

Any[(0.0010000000000000002, 0.2), (1.0000000000000004e-6, 0.0016000000000000003), (1.00000000000000005e-9, 1.28000000000000005e-5), (1.00000000000000006e-12, 1.02400000000000006e-7), (1.00000000000000009e-15, 8.19200000000000005e-10), (1.0000000000000001e-18, 6.553600000000000055e-12), (1.00000000000000012e-21, 5.242800000000000056e-14), (1.00000000000000014e-24, 4.19430400000000005e-16), (1.00000000000000015e-27, 3.355443200000000048e-18), (1.00000000000000017e-30, 2.6843545600000004e-20), (1.00000000000000018e-33, 2.147483648000000035e-22), (1.0000000000000002e-36, 1.7179869184000003e-24)]
```

Рис. 3.30: Задание 3.11

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

```
[200]:
#3.12
ar12=[(2^i)/i for i in 1:25]

[200]:
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
```

Рис. 3.31: Задание 3.12

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

```
[209]:
#3.13
ar13=[]
for i in 1:30
    push!(ar13, "fn$i")
end
ar13
```

```
[209]:
30-element Vector{Any}:
 "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"
```

Рис. 3.32: Задание 3.13

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:

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

```
[212]:
#3.14
x = rand(0:999, 1, 250);
y = rand(0:999, 1, 250);
v1 = [y[i]-x[i-1] for i in 2:250]
```

```
[212]:
249-element Vector{Int64}:
 550
  13
  77
 381
-472
 801
-574
 694
 783
 124
-200
-489
-734
   ⋮
 193
 415
  48
 507
-504
 126
 576
-571
-449
 187
-277
-397
```

Рис. 3.33: Задание 3.14

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

```
[214]:
v2 = [x[i-2]+2*x[i-1]-x[i] for i in 3:250]
```

```
[214]:
248-element Vector{Int64}:
 101
-133
 101
2062
 470
1521
 916
 -78
 761
1214
 953
1589
2285
   ⋮
 262
 162
2115
 654
2080
1137
-721
1369
1656
 310
1483
1762
```

Рис. 3.34: Задание 3.15

- сформируйте вектор $\left(\frac{\sin(y_1)}{\cos(x_2)}, \frac{\sin(y_2)}{\cos(x_3)}, \dots, \frac{\sin(y_{n-1})}{\cos(x_n)} \right)$;
- вычислите $\sum_{i=1}^{n-1} \frac{e^{-x_{i+1}}}{x_i + 10}$;

Рис. 3.35: Задачи

```
[215]:
v3 = [(sin(y[i-1]))/(cos(x[i])) for i in 2:250]
```

```
[215]:
```

```
249-element Vector{Float64}:
```

```
-1.754722687152709
 0.9924804445211151
 0.8887114139147112
 2.2097483587123703
 1.2926131931680243
 0.8865820542365037
-1.1181275586399602
 1.2556847312195138
-0.3504773299757346
 0.3233601793275197
-0.6605551156470216
-4.110448071676383
-1.3054656540484193
 ⋮
 0.1366529552109628
-0.9904019338058403
-0.49407478784638204
-0.26896935770095953
-1.1523421912916259
-2.2997286020434804
 0.5028973046701645
-0.03779959921553782
 0.6240594659965716
 0.6037466961066857
 8.521144605305865
-0.7578279342271558
```

Рис. 3.36: Задание 3.16

```
[220]:
sum=0
for i in 1:249
    sum+=(exp(-x[i+1]))/x[i]+10
end
print(sum)
```

```
2490.000551954167
```

Рис. 3.37: Задание 3.17

– выбрала элементы вектора y , значения которых больше 600, и вывела на экран; определила индексы этих элементов;


```

[228]:
for i in 1:250
    if y[i]>600
        print(y[i], ", ")
    else
        continue
    end
end

print('\n', "Индексы: ", findall(y .> 600))

```

765, 823, 969, 861, 961, 725, 904, 605, 787, 697, 607, 993, 883, 893, 715, 980, 874, 838, 916, 707,
931, 755, 917, 788, 768, 715, 861, 767, 910, 790, 842, 991, 754, 919, 851, 708, 804, 914, 834, 700,
694, 849, 811, 667, 944, 612, 615, 839, 847, 871, 862, 977, 639, 720, 901, 967, 832, 872, 707, 637,
912, 868, 824, 888, 943, 949, 728, 737, 624, 983, 972, 663, 867, 934, 873, 842, 616, 614, 724, 764,
850, 630, 858, 762, 603, 766, 793, 801, 979, 952, 964, 993, 831, 622, 706,
Индексы: CartesianIndex{2}[CartesianIndex{1, 1}, CartesianIndex{1, 5}, CartesianIndex{1, 7}, CartesianIndex{1, 9}, CartesianIndex{1, 10}, CartesianIndex{1, 11}, CartesianIndex{1, 20}, CartesianIndex{1, 22}, CartesianIndex{1, 24}, CartesianIndex{1, 31}, CartesianIndex{1, 32}, CartesianIndex{1, 33}, CartesianIndex{1, 36}, CartesianIndex{1, 37}, CartesianIndex{1, 40}, CartesianIndex{1, 41}, CartesianIndex{1, 45}, CartesianIndex{1, 46}, CartesianIndex{1, 47}, CartesianIndex{1, 49}, CartesianIndex{1, 52}, CartesianIndex{1, 55}, CartesianIndex{1, 58}, CartesianIndex{1, 59}, CartesianIndex{1, 62}, CartesianIndex{1, 63}, CartesianIndex{1, 64}, CartesianIndex{1, 67}, CartesianIndex{1, 68}, CartesianIndex{1, 71}, CartesianIndex{1, 72}, CartesianIndex{1, 73}, CartesianIndex{1, 75}, CartesianIndex{1, 76}, CartesianIndex{1, 78}, CartesianIndex{1, 79}, CartesianIndex{1, 85}, CartesianIndex{1, 87}, CartesianIndex{1, 91}, CartesianIndex{1, 92}, CartesianIndex{1, 94}, CartesianIndex{1, 96}, CartesianIndex{1, 98}, CartesianIndex{1, 100}, CartesianIndex{1, 101}, CartesianIndex{1, 102}, CartesianIndex{1, 103}, CartesianIndex{1, 106}, CartesianIndex{1, 110}, CartesianIndex{1, 111}, CartesianIndex{1, 113}, CartesianIndex{1, 114}, CartesianIndex{1, 115}, CartesianIndex{1, 117}, CartesianIndex{1, 119}, CartesianIndex{1, 121}, CartesianIndex{1, 125}, CartesianIndex{1, 128}, CartesianIndex{1, 135}, CartesianIndex{1, 141}, CartesianIndex{1, 142}, CartesianIndex{1, 143}, CartesianIndex{1, 145}, CartesianIndex{1, 146}, CartesianIndex{1, 150}, CartesianIndex{1, 159}, CartesianIndex{1, 165}, CartesianIndex{1, 166}, CartesianIndex{1, 167}, CartesianIndex{1, 168}, CartesianIndex{1, 169}, CartesianIndex{1, 172}, CartesianIndex{1, 174}, CartesianIndex{1, 181}, CartesianIndex{1, 182}, CartesianIndex{1, 184}, CartesianIndex{1, 186}, CartesianIndex{1, 187}, CartesianIndex{1, 192}, CartesianIndex{1, 193}, CartesianIndex{1, 194}, CartesianIndex{1, 201}, CartesianIndex{1, 211}, CartesianIndex{1, 215}, CartesianIndex{1, 222}, CartesianIndex{1, 225}, CartesianIndex{1, 228}, CartesianIndex{1, 229}, CartesianIndex{1, 233}, CartesianIndex{1, 238}, CartesianIndex{1, 240}, CartesianIndex{1, 241}, CartesianIndex{1, 242}, CartesianIndex{1, 245}, CartesianIndex{1, 249}]

Рис. 3.38: Задание 3.18

- определила значения вектора x , соответствующие значениям вектора y , значения которых больше 600 (под соответствием понимается расположение на аналогичных индексных позициях);

```

[228]:
for i in 1:250
    if y[i]>600
        print(x[i], ", ")
    else
        continue
    end
end

```

38, 894, 760, 178, 601, 619, 823, 456, 991, 227, 430, 285, 617, 993, 464, 458, 593, 47, 346, 760, 3
80, 257, 166, 549, 465, 753, 126, 18, 274, 333, 682, 213, 331, 737, 925, 768, 74, 298, 61, 441, 80
5, 378, 805, 313, 143, 470, 717, 507, 814, 991, 757, 881, 932, 547, 976, 699, 940, 831, 249, 259, 4
89, 17, 235, 828, 50, 477, 512, 341, 328, 480, 450, 449, 13, 263, 883, 667, 333, 716, 836, 652, 19
4, 16, 474, 249, 695, 829, 17, 322, 631, 9, 945, 324, 939, 935, 856,

Рис. 3.39: Задание 3.19

- сформируйте вектор $(|x_1 - \bar{x}|^{\frac{1}{2}}, |x_2 - \bar{x}|^{\frac{1}{2}}, \dots, |x_n - \bar{x}|^{\frac{1}{2}})$, где \bar{x} обозначает среднее значение вектора $x = (x_1, x_2, \dots, x_n)$;

Рис. 3.40: Задачи

```
[230]:
sum=0
for i in x
    sum+=i
end
avg=sum/length(x)
v4=[(abs(x[i]-avg))^0.5 for i in 1:250]
```

```
[230]:
250-element Vector{Float64}:
 21.590460856591275
 20.424201330774235
 19.827960056445544
  7.883400281604381
 19.744670166908335
 18.3343393663366
 15.995374331349673
 18.36159034506543
 18.059568101147935
  9.841341371987864
 10.716902537580529
 10.993270668913778
 20.294137084389668
  :
  6.697163578710018
 20.99647589477815
 13.421922366039823
 20.853105284345542
 19.54860608841459
 21.404392072656492
 20.756974731400526
  6.546143903092871
 11.45198672720153
 21.88268722072314
 18.75771841136336
 20.70874211534829
```

Рис. 3.41: Задание 3.20

- определила, сколько элементов вектора y отстоят от максимального значения не более, чем на 200;

```
[234]:
max1 = maximum(y)
count=0
for i in 1:250
    if max1 - y[i] < 200
        count+=1
    else
        continue
    end
end
print(count)
```

```
55
```

Рис. 3.42: Задание 3.21

- определила, сколько чётных и нечётных элементов вектора x ;

```
[239]:
count_ch=0
for i in 1:250
    if x[i]%2 ==0
        count_ch+=1
    else
        continue
    end
end
print(count_ch)

123

[240]:
count_nch=0
for i in 1:250
    if x[i]%2 !=0
        count_nch+=1
    else
        continue
    end
end
print(count_nch)

127
```

Рис. 3.43: Задание 3.22

– определила, сколько элементов вектора x кратны 7;

```
[242]:
count_7=0
for i in 1:250
    if x[i]%7 ==0
        count_7+=1
    else
        continue
    end
end
print(count_7)

30
```

Рис. 3.44: Задание 3.23

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

```
[253]:
reverse(last(sort(x,dims=2), 10))

[253]:
10-element Vector{Int64}:
 993
 993
 991
 991
 984
 983
 976
 973
 954
 951
```

Рис. 3.45: Задание 3.24

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

```
[254]:  
unique(x)  
  
[254]:  
218-element Vector{Int64}:  
 38  
 87  
111  
442  
894  
168  
760  
167  
178  
601  
619  
625  
916  
:  
724  
104  
945  
793  
324  
939  
 46  
935  
373  
983  
856  
933
```

Рис. 3.46: Задание 3.25

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

```
[255]:
#4
squares = [i^2 for i in 1:100]

[255]:
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. 3.47: Задание 4

9. Подключила пакет Primes (функции для вычисления простых чисел). Сгенерировала массив `myprimes`, в котором будут храниться первые 168 простых чисел. Определила 89-е наименьшее простое число. Получила срез массива с 89-го до 99-го элемента включительно, содержащий наименьшие простые числа.

```

import Primes

[287]:
myprimes = Primes.primes(Primes.prime(168))
myprimes[89]

[287]:
461

[288]:
Primes.prime(89)

[288]:
461

[289]:
myprimes[89:99]

[289]:
11-element Vector{Int64}:
 461
 463
 467
 479
 487
 491
 499
 503
 509
 521
 523

```

Рис. 3.48: Задание 5

10. Вычислила следующие выражения:

- 6.1) $\sum_{i=10}^{100} (i^3 + 4i^2);$
- 6.2) $\sum_{i=1}^M \left(\frac{2^i}{i} + \frac{3^i}{i^2} \right), M = 25;$
- 6.3) $1 + \frac{2}{3} + \left(\frac{2}{3} \frac{4}{5} \right) + \left(\frac{2}{3} \frac{4}{5} \frac{6}{7} \right) + \dots + \left(\frac{2}{3} \frac{4}{5} \dots \frac{38}{39} \right).$

Рис. 3.49: Задачи

```

[290]: #6.1
sum1=0
for i in 10:100
    sum1+=(i^3 + 4*i^2)
end
sum1

[290]: 26852735

[292]: #6.2
sum2=0
for i in 1:25
    sum2+=((2*i)/i) + (3*i/i^2)
end
sum2

[292]: 2.1291704368143802e9

[293]: #6.3
sum3=1
dr=1
for i in 2:2:38
    dr*= i/(i+1)
    sum3+=dr
end
sum3

[293]: 6.976346137897618

```

Рис. 3.50: Задание 6

4 Выводы

Я изучила несколько структур данных, реализованных в Julia, и научилась применять их и операции над ними для решения задач.

Список литературы