

Лабораторная работа №2

**Дисциплина: Компьютерный практикум по статистическому
моделированию**

Манаева Варвара Евгеньевна

Содержание

1	Техническое оснащение:	6
2	Цели и задачи работы	7
2.1	Цель	7
2.2	Задачи [1]	7
3	Выполнение лабораторной работы	8
3.1	Повторение примеров	8
3.2	Самостоятельная работа	13
4	Выводы по проделанной работе	33
4.1	Вывод	33
	Список литературы	34

Список иллюстраций

3.1	Повторение примеров (1)	8
3.2	Повторение примеров (2)	9
3.3	Повторение примеров (3)	9
3.4	Повторение примеров (4)	10
3.5	Повторение примеров (5)	10
3.6	Повторение примеров (6)	11
3.7	Повторение примеров (7)	11
3.8	Повторение примеров (8)	12
3.9	Повторение примеров (9)	12
3.10	Повторение примеров (10)	13
3.11	Функции и формула из задания 1	14
3.12	Примеры выполнения операций	14
3.13	Примеры выполнения операций	15
3.14	Массивы для пункта 3 (1)	15
3.15	Массивы для пункта 3 (2)	16
3.16	Массивы для пункта 3 (3)	16
3.17	Массивы для пункта 3 (4)	17
3.18	Массивы для пункта 3 (5)	17
3.19	Массивы для пункта 3 (6)	18
3.20	Массивы для пункта 3 (7)	18
3.21	Массивы для пункта 3 (8)	19
3.22	Массивы для пункта 3 (9)	19
3.23	Массивы для пункта 3 (10)	20
3.24	Массивы для пункта 3 (11)	20
3.25	Массивы для пункта 3 (12)	21
3.26	Массивы для пункта 3 (13)	21
3.27	Массивы для пункта 3 (14)	22
3.28	Массивы для пункта 3 (15)	22
3.29	Массивы для пункта 3 (16)	23
3.30	Массивы для пункта 3 (17)	23
3.31	Массивы для пункта 3 (18)	24
3.32	Массивы для пункта 3 (19)	24
3.33	Массивы для пункта 3 (20)	25
3.34	Массивы для пункта 3 (21)	25
3.35	Массивы для пункта 3 (22)	26
3.36	Массивы для пункта 3 (23)	26
3.37	Массивы для пункта 3 (24)	27

3.38	Массивы для пункта 3 (25)	27
3.39	Массивы для пункта 3 (26)	28
3.40	Массивы для пункта 3 (27)	28
3.41	Массивы для пункта 3 (28)	29
3.42	Массивы для пункта 3 (29)	29
3.43	Массивы для пункта 3 (30)	30
3.44	Массивы для пункта 3 (31)	30
3.45	Массив натуральных чисел	31
3.46	Работа с простыми числами	31
3.47	Вычисление выражений (1)	32
3.48	Вычисление выражений (2)	32

Список таблиц

1 Техническое оснащение:

- Персональный компьютер с операционной системой Windows 10;
- Планшет для записи видеосопровождения и голосовых комментариев;
- Microsoft Teams, использующийся для записи скринкаста лабораторной работы;
- Приложение Rucharm для редактирования файлов формата *md*;
- *pandoc* для конвертации файлов отчётов и презентаций.

2 Цели и задачи работы

2.1 Цель

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

2.2 Задачи [1]

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 Повторение примеров

1. Реализация кортежей (3.1, 3.2)
2. Реализация словарей (3.2, 3.3)
3. Реализация множеств (3.4, 3.5, 3.6)
4. Реализация массивов (3.6, 3.7, 3.8, 3.9, 3.10)

Реализация кортежей

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

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

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

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

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

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

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

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

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

[5]: true

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

[6]: "555-3344"
```

Рис. 3.1: Повторение примеров (1)


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

[7]: ("867-5309", "333-5544")

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

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

Реализация словарей

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

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

[10]: # создать множество из 11 символьных значений:
B = Set{"abracadabra"}

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

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

[11]: false

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

[12]: true
```

Рис. 3.2: Повторение примеров (2)

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

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

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

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

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

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

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

[16]: true

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

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

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

[18]: 2
```

Рис. 3.3: Повторение примеров (3)

```

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

[19]: Any[]

[20]: # создание пустого массива с конкретным типом:
empty_array_2 = (Int64){}
empty_array_3 = (Float64){}

[20]: Float64[]

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

[21]: 3-element Vector{Int64}:
 1
 2
 3

[22]: # вектор-строка:
b = [1 2 3]
b

[22]: 1x3 Matrix{Int64}:
 1 2 3

[23]: # ячеечные массивы (матрицы):
A = [[1, 2, 3] [4, 5, 6] [7, 8, 9]]
B = [[1 2 3]; [4 5 6]; [7 8 9]]
A, B

[23]: ([1 4 7; 2 5 8; 3 6 9], [1 2 3; 4 5 6; 7 8 9])

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

[24]: 1x8 Matrix{Float64}:
 0.768374  0.411788  0.648862  0.193864  ...  0.899928  0.363381  0.0207368

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

[25]: 2x3 Matrix{Float64}:
 0.757099  0.32779  0.884303
 0.423174  0.958955  0.52504

```

Рис. 3.4: Повторение примеров (4)

```

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

[26]: 4x3x2 Array{Float64, 3}:
[:, :, 1] =
 0.547701  0.481862  0.52237
 0.876654  0.0863517  0.103491
 0.0755708  0.234026  0.886443
 0.488524  0.0821956  0.953272

[:, :, 2] =
 0.771658  0.431458  0.124003
 0.517059  0.229559  0.178669
 0.241137  0.348298  0.679719
 0.951206  0.870145  0.157296

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

[27]: 10-element Vector{Float64}:
 1.0
 1.4142135623730951
 1.7320508075688772
 2.0
 2.23606797749979
 2.449489742783178
 2.6457513110645907
 2.8284271247461903
 3.0
 3.1622776601683795

[28]: # массив с элементами вида  $3^i \cdot 2^j$ , где  $i$  - нечётное число от 1 до 9 (включительно)
ar_1 = [3^i * 2 for i in 1:2:9]
ar_1

[28]: 5-element Vector{Int64}:
 3
 27
 75
 147
 243

```

Рис. 3.5: Повторение примеров (5)

```
[29]: # массив квадратной элементов, если квадрат не делится на 5 или 4:
ar_2 = [i**2 for i in range(1,10) if (i**2%5!=0 && i**2%4!=0)]
ar_2

[29]: 4-element Vector{Int64}:
      1
      9
     49
     81

Реализация массивов

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

[30]: 5-element Vector{Float64}:
      1.0
      1.0
      1.0
      1.0
      1.0

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

[31]: 2x3 Matrix{Float64}:
      1.0  1.0  1.0
      1.0  1.0  1.0

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

[32]: 4-element Vector{Float64}:
      0.0
      0.0
      0.0
      0.0

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

[33]: 3x2 Matrix{Float64}:
      3.5  3.5
      3.5  3.5
      3.5  3.5
```

Рис. 3.6: Повторение примеров (6)

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

[34]: 3x6 Matrix{Int64}:
      1  2  1  2  1  2
      1  2  1  2  1  2
      1  2  1  2  1  2

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

[35]: 2x6 Matrix{Int64}:
      1  3  5  7  9  11
      2  4  6  8  10  12

[36]: # транспонирование
b'

[36]: 6x2 adjoint{::Matrix{Int64}} with eltype Int64:
      1  2
      3  4
      5  6
      7  8
      9  10
     11  12

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

[37]: 1x8 Matrix{Float64}:
      0.768374  0.411788  0.648862  0.193864  - 0.899928  0.363381  0.0207368

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

[38]: 10x5 Matrix{Int64}:
      12  15  10  17  20
      10  17  11  16  15
      12  12  10  20  14
      14  20  12  16  10
      10  17  16  14  10
      11  13  10  19  17
      11  17  14  12  14
      18  17  10  14  20
      10  17  17  10  14
      16  18  12  12  20
```

Рис. 3.7: Повторение примеров (7)

```
[39]: # выбор всех значений строки 0 столбце 2:
ar[:, 2]

[39]: 10-element Vector{Int64}:
 15
 17
 12
 20
 17
 13
 17
 17
 17
 18

[40]: # выбор всех значений 0 столбцах 2 и 5:
ar[:, [2, 5]]

[40]: 10x2 Matrix{Int64}:
 15  20
 17  15
 12  14
 20  10
 17  10
 13  17
 17  14
 17  20
 17  14
 18  20

[41]: # все значения строк 0 столбцах 2, 3 и 4:
ar[:, 2:4]

[41]: 10x3 Matrix{Int64}:
 15  10  17
 17  11  16
 12  10  20
 20  12  16
 17  16  14
 13  10  19
 17  14  12
 17  10  14
 17  17  10
 18  12  12
```

Рис. 3.8: Повторение примеров (8)

```
[42]: # значения 0 строках 2, 4, 6 и 0 столбцах 1 и 5:
ar[[2, 4, 6], [1, 5]]

[42]: 3x2 Matrix{Int64}:
 10  15
 14  10
 11  17

[43]: # значения 0 строке 1 от столбца 3 до последнего столбца:
ar[1, 3:end]

[43]: 3-element Vector{Int64}:
 10
 17
 20

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

[44]: 10x5 Matrix{Int64}:
 10  12  10  10  10
 10  13  10  12  10
 10  15  10  12  14
 11  17  10  14  14
 11  17  11  14  14
 12  17  12  16  15
 12  17  12  16  17
 14  17  14  17  20
 16  18  16  19  20
 18  20  17  20  20

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

[45]: 10x5 Matrix{Int64}:
 10  12  15  17  20
 10  11  15  16  17
 10  12  12  14  20
 10  12  14  16  20
 10  10  14  16  17
 10  11  13  17  19
 11  12  14  14  17
 10  14  17  18  20
 10  10  14  17  17
 12  12  16  18  20
```

Рис. 3.9: Повторение примеров (9)

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

```
[46]: 10x5 BitMatrix:
```

```
0 1 0 1 1
0 1 0 1 1
0 0 0 1 0
0 1 0 1 0
0 1 1 0 0
0 0 0 1 1
0 1 0 0 0
1 1 0 0 1
0 1 1 0 0
1 1 0 0 1
```

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

```
[47]: 22-element Vector{CartesianIndex{2}}:
```

```
CartesianIndex{2}(0, 1)
CartesianIndex{2}(10, 1)
CartesianIndex{2}(1, 2)
CartesianIndex{2}(2, 2)
CartesianIndex{2}(4, 2)
CartesianIndex{2}(5, 2)
CartesianIndex{2}(7, 2)
CartesianIndex{2}(8, 2)
CartesianIndex{2}(9, 2)
CartesianIndex{2}(10, 2)
CartesianIndex{2}(5, 3)
CartesianIndex{2}(9, 3)
CartesianIndex{2}(1, 4)
CartesianIndex{2}(2, 4)
CartesianIndex{2}(3, 4)
CartesianIndex{2}(4, 4)
CartesianIndex{2}(6, 4)
CartesianIndex{2}(1, 5)
CartesianIndex{2}(2, 5)
CartesianIndex{2}(6, 5)
CartesianIndex{2}(8, 5)
CartesianIndex{2}(10, 5)
```

Рис. 3.10: Повторение примеров (10)

3.2 Самостоятельная работа

1. Пересечение и объединение множеств $A = \{0, 3, 4, 9\}$, $B = \{1, 3, 4, 7\}$ и $C = \{0, 1, 2, 4, 7, 8, 9\}$ (3.11).

```
[93]: function peresech(x, y)
      z = []
      for i in x
        for j in y
          if i == j
            append!(z, i)
          end
        end
      end
      return z
end

function obeydin(x, y...)
  z = x
  for i in y
    for j in i
      if findfirst(isequal(j), z) == nothing
        append!(z, j)
      end
    end
  end
  return sort(z)
end

A = [0, 3, 4, 9]
B = [1, 3, 4, 7]
C = [0, 1, 2, 4, 7, 8, 9]
peresech(A,B), obeydin(A,B)

[93]: (Any{3, 4}, [0, 1, 3, 4, 7, 9])

Высчитаем формулу из задания:  $P = (A \cap B) \cup (A \cap C) \cup (B \cap C)$  (разделена скобками для упрощения восприятия)

[94]: t1 = peresech(A,B)
      t2 = peresech(A,C)
      t3 = peresech(B,C)
      P = obeydin(t1,t2,t3)

[94]: 6-element Vector{Any}:
      0
      1
      3
      4
      7
      9
```

Рис. 3.11: Функции и формула из задания 1

2. Примеры с выполнением операций над множествами элементов разных типов (??)

2. Примеры с выполнением операций над множествами элементов разных типов

```
[63]: t = Set{[1 2 3; 4 5 6; "By" "Ka" "Бака"]}
```

```
[63]: Set{Any} with 9 elements:
      5
      4
      "Ka"
      6
      2
      "Бака"
      "By"
      3
      1
```

```
[64]: 1 in t
[64]: true
```

```
[65]: 0 in t
[65]: false
```

```
[66]: [1, 2, 3] in t
[66]: false
```

```
[67]: intersect(Set{[1, 2, 3, 4, 7, 8, 10]}, t)
```

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

```
[68]: union(Set{[1, 2, 3, 4, 7, 8, 10]}, t)
```

```
[68]: Set{Any} with 12 elements:
      5
      "Ka"
      7
      "Бака"
      8
      "By"
      1
      4
```

Рис. 3.12: Примеры выполнения операций

```

[69]: setdiff(Set([1, 2, 3, 4, 7, 8, 10]), t)
[69]: Set{Int64} with 3 elements:
      7
      10
      8

[70]: setdiff(t, Set([1, 2, 3, 4, 7, 8, 10]))
[70]: Set{Any} with 5 elements:
      5
      "Ka"
      6
      "Бака"
      "By"

```

Рис. 3.13: Примеры выполнения операций

3. Создать разными способами несколько видов массивов (3.14, 3.15)

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

```

[70]: N = 100
      t = zeros{Int64, N}
      for i in 1:N
          t[i] = i
      end
      t

[70]: 100-element Vector{Int64}:
       1
       2
       3
       4
       5
       6
       7
       8
       9
      10
      11
      12
      13
       ⋮
      89
      90
      91
      92
      93
      94
      95
      96
      97
      98
      99
     100

[71]: t = collect(1:N)
      t

[71]: 100-element Vector{Int64}:
       1
       2
       3
       4
       5
       6
       7
       ⋮

```

Рис. 3.14: Массивы для пункта 3 (1)

```
[72]: t = [i for i in 1:N]

[72]: 100-element Vector{Int64}:
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
⋮
89
90
91
92
93
94
95
96
97
98
99
100

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

[170]: t = zeros{Int64, N}
      for i in N:-1:1
          t[N-i+1] = i
      end
      t

[170]: 100-element Vector{Int64}:
100
 99
 98
 97
 96
 95
 94
 93
```

Рис. 3.15: Массивы для пункта 3 (2)

```
[107]: t = collect(N:-1:1)
      t

[107]: 100-element Vector{Int64}:
100
 99
 98
 97
 96
 95
 94
 93
 92
 91
 90
 89
 88
 ⋮
12
11
10
 9
 8
 7
 6
 5
 4
 3
 2
 1

[125]: t = [N-i+1 for i in 1:N]

[125]: 100-element Vector{Int64}:
100
 99
 98
 97
 96
 95
 94
 93
 92
 91
 90
 89
 88
 ⋮
 2
 1
```

Рис. 3.16: Массивы для пункта 3 (3)

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

```
[130]: t = zeros(Int64, 2*N-1)
for i in 1:N
    t[i] = i
end
for i in 1:N-1
    t[i+N] = N-i
end
t
```

```
[130]: 199-element Vector{Int64}:
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
⋮
12
11
10
 9
 8
 7
 6
 5
 4
 3
 2
 1
```

```
[128]: t = cat([i for i in 1:N], [i for i in N-1:-1:1], dims=1)
```

```
[128]: 199-element Vector{Int64}:
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
⋮
12
11
10
 9
 8
 7
 6
 5
 4
 3
 2
 1
```

Рис. 3.17: Массивы для пункта 3 (4)

4. массив с именем `tmp` вида (4, 6, 3)

```
[58]: t1 = 4; t2 = 6; t3 = 3
tmp = [t1, t2, t3]
```

```
[58]: 3-element Vector{Int64}:
 4
 6
 3
```

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

```
[59]: 3-element Vector{Int64}:
 4
 6
 3
```

5. массив, в котором первый элемент массива `tmp` повторяется 10 раз

```
[123]: t1 = zeros(Int64, 10)
for i in 1:10
    t1[i] = tmp[1]
end
t1
```

```
[123]: 10-element Vector{Int64}:
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
```

```
[65]: t1 = [tmp[1] for i in 1:10]
t1
```

```
[65]: 10-element Vector{Int64}:
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
```

Рис. 3.18: Массивы для пункта 3 (5)

```
[124]: t1 = fill(tmp[1], 10)

[124]: 10-element Vector{Int64}:
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4

6. массив, в котором все элементы массива tmp повторяются 10 раз

[112]: t2 = zeros{Int64, 30}
      for i in 0:9
          t2[3*i+1] = tmp[1]
          t2[3*i+2] = tmp[2]
          t2[3*i+3] = tmp[3]
      end
      t2

[112]: 30-element 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
 4
```

Рис. 3.19: Массивы для пункта 3 (6)

```
[113]: t2 = repeat(tmp, 10)
      t2

[113]: 30-element 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
 4

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

[114]: t3 = [tmp[1] for i in 1:11]
      for i in 1:10
          append!(t3, tmp[2])
          append!(t3, tmp[3])
      end
      t3

[114]: 31-element Vector{Int64}:
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
 6
 6
 6
 6
 6
 6
 6
 6
 6
 6
 6
 6
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
```

Рис. 3.20: Массивы для пункта 3 (7)

Рис. 3.21: Массивы для пункта 3 (8)

```
[119]: t3 = [tmp[1] for i in 1:10]
for i in 1:20
    append!(t3, tmp[2])
    append!(t3, tmp[3])
end
for i in 1:10
    append!(t3, tmp[3])
end
t3
```

```
[119]: 60-element Vector{Int64}:
 4
 4
 4
 4
 4
 4
 4
 4
 6
 3
 6
 :
 6
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
```

```
[121]: t3 = repeat(tmp, 10)
for i in 1:10
    append!(t3, tmp[2])
    append!(t3, tmp[3])
end
for i in 1:10
    append!(t3, tmp[3])
end
```

Рис. 3.22: Массивы для пункта 3 (9)

Рис. 3.23: Массивы для пункта 3 (10)

```

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

```

```

144]: t2 = []
      tmpsize = size(tmp)[1]
      for i in 1:4
          for j in 1:tmpsize
              append!(t2, 2^(tmp[j]))
          end
      end
      t2

```

```

144]: 12-element Vector{Any}:
      16
      64
      8
      16
      64
      8
      16
      64
      8
      16
      64
      8

```

```

145]: t2 = fill(2, 4*tmpsize)
      for i in 0:3
          for j in 1:3
              t2[i*tmpsize+j] = t2[i*tmpsize+j] ^ tmp[j]
          end
      end
      t2

```

```

145]: 12-element Vector{Int64}:
      16
      64
      8
      16
      64
      8
      16
      64
      8
      16
      64
      8

```

Рис. 3.24: Массивы для пункта 3 (11)

```
[148]: t2 = repeat(fill(2, tmpsize).^tmp, 4)

[148]: 12-element Vector{Int64}:
      16
       64
        8
       16
       64
        8
       16
       64
        8
       16
       64
        8

Сосчитаем количество 6 в векторе

[155]: size(findall(isequal("6"), split(join(string.(t2)), "")))[1]

[155]: 8

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

[159]: t1 = collect(3:0.1:6)
      t = ones(size(t1)[1])
      for i in 1:size(t1)[1]
          t[i] = exp(t1[i])*cos(t1[i])
      end
      t

[159]: 31-element Vector{Float64}:
      -19.884530844146987
      -22.178753389342127
      -24.490696732801293
      -26.77318244299338
      -28.969237768893574
      -31.01186439374516
      -32.819774760338504
      -34.30336011037369
      -35.35719361853035
      -35.86283371230767
      -35.68773248011913
      -34.68504225166807
      -32.693695428321746
      :
```

Рис. 3.25: Массивы для пункта 3 (12)

```
[83]: t = [exp(i)*cos(i) for i in 3:0.1:6]

[83]: 31-element Vector{Float64}:
      -19.884530844146987
      -22.178753389342127
      -24.490696732801293
      -26.77318244299338
      -28.969237768893574
      -31.01186439374516
      -32.819774760338504
      -34.30336011037369
      -35.35719361853035
      -35.86283371230767
      -35.68773248011913
      -34.68504225166807
      -32.693695428321746
      :
      25.046704998273004
      42.099201062533839
      61.99663027669454
      84.92906736250268
      111.0615860420258
      140.5250759527875
      173.40577640857734
      209.73349424783467
      249.46844055885668
      292.4867067371223
      338.5643778585117
      387.36034029093076

Среднее значение вектора y

[84]: sum(t)/size(t)[1]

[84]: 53.11374594642971

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

[85]: t1 = collect(3:3:36)
      t2 = collect(1:3:34)
      t = zeros((size(t1)[1], 2))
      for i in 1:size(t1)[1]
          t[i,1] = 0.1*t1[i]
          t[i,2] = 0.2*t2[i]
      end
      t
```

Рис. 3.26: Массивы для пункта 3 (13)

```
[168]: 12x2 Matrix(Float64):
 0.001  0.2
 1.0e-6  0.0016
 1.0e-9  1.28e-5
 1.0e-12  1.024e-7
 1.0e-15  8.192e-10
 1.0e-18  6.5536e-12
 1.0e-21  5.24288e-14
 1.0e-24  4.1943e-16
 1.0e-27  3.35544e-18
 1.0e-30  2.68435e-20
 1.0e-33  2.14748e-22
 1.0e-36  1.71799e-24

[158]: t = zeros((size(collect(3:3:36))[1], 2))
for i in 0:size(collect(3:3:36))[1]-1
    t[i+1, 1] = 0.1*(3^(i+1))
    t[i+1, 2] = 0.2*(3^(i+1))
end
t

[158]: 12x2 Matrix(Float64):
 0.001  0.2
 1.0e-6  0.0016
 1.0e-9  1.28e-5
 1.0e-12  1.024e-7
 1.0e-15  8.192e-10
 1.0e-18  6.5536e-12
 1.0e-21  5.24288e-14
 1.0e-24  4.1943e-16
 1.0e-27  3.35544e-18
 1.0e-30  2.68435e-20
 1.0e-33  2.14748e-22
 1.0e-36  1.71799e-24

[167]: t = hcat(fill(0.1, size(collect(3:3:36))[1]), fill(0.2, size(collect(1:3:34))[1]))
t[:, 1] = t[:, 1].^collect(3:3:36)
t[:, 2] = t[:, 2].^collect(1:3:34)
t

[167]: 12x2 Matrix(Float64):
 0.001  0.2
 1.0e-6  0.0016
 1.0e-9  1.28e-5
 1.0e-12  1.024e-7
 1.0e-15  8.192e-10
 1.0e-18  6.5536e-12
 1.0e-21  5.24288e-14
```

Рис. 3.27: Массивы для пункта 3 (14)

12. вектор с элементами $\frac{2^i}{i}, i = 1, 2, \dots, M, M = 25$

```
[92]: M = 25
t = [2^i/i for i in 1:M]

[92]: 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
 52428.0
 99864.38095238095
 190650.18181818182
 364722.0869565217
 699850.6666666666
 1.34217728e6

[93]: t = zeros(M)
for i in 1:M
    t[i] = 2^i/i
end
t

[93]: 25-element Vector{Float64}:
 2.0
 2.0
 2.6666666666666665
 4.0
 6.4
 10.666666666666666
 18.285714285714285
 32.0
 56.888888888888886
```

Рис. 3.28: Массивы для пункта 3 (15)

```
[95]: t1 = collect(1:M)
      t = fill(2, M).^t1
      t = t./t1
      t

[95]: 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
  99864.30895238095
 198650.18181818182
 364722.0869565217
 699050.6666666666
   1.34217728e6

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

```
[88]: N = 30
      t = [join(["fn", string(i)]) for i in 1:N]

[88]: 30-element Vector{String}:
      "fn1"
      "fn2"
      "fn3"
      "fn4"
      "fn5"
      "fn6"
      "fn7"
      "fn8"
      "fn9"
```

Рис. 3.29: Массивы для пункта 3 (16)

```
[93]: t1 = string.(collect(1:N))
      t2 = fill("fn", N)
      t = join.([t2[i], t1[i]] for i in 1:N)
```

```
[93]: 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"
```

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 $0, 1, \dots, 999$ и на их основе создать следующие векторы:

```
[94]: n = 250
      x, y = rand(0:999, n), rand(0:999, n)

[94]: ([693, 40, 527, 700, 380, 516, 122, 461, 711, 822, ..., 30, 434, 188, 205, 405, 66, 851, 102, 792, 310], [618, 341, 265, 868, 455, 922, 95, 127, 773, 547, ..., 671, 71, 161, 168, 764, 337, 92, 210, 676, 488])
```

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

Рис. 3.30: Массивы для пункта 3 (17)

```

[135]: t = zeros(Int64, n-1)
       for i in 1:n-1
           t[i] = y[i+1]*x[i]
       end
       t

[135]: 249-element Vector{Int64}:
        -352
         225
         341
        -245
         542
        -421
           5
         312
        -164
         -69
         258
         313
        -452
           ⋮
         864
        -760
        -316
          41
        -273
         -20
         559
         -68
           26
        -641
         574
        -304

[133]: t = [y[i+1]*x[i] for i in 1:n-1]

[133]: 249-element Vector{Int64}:
        -352
         225
         341
        -245
         542
        -421
           5
         312
        -164
         -69
         258
         313
        -452
           ⋮
         864
        -760
        -316
          41
        -273
         -20
         559
         -68
           26
        -641
         574
        -304

```

Рис. 3.31: Массивы для пункта 3 (18)

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

```

[136]: t = zeros(Int64, n-2)
       for i in 1:n-2
           t[i] += x[i]
           t[i] += 2*x[i+1]
           t[i] -= x[i+2]
       end
       t

[136]: 248-element Vector{Int64}:
         246
         394
        1547
         944
        1290
         299
         333
        1061
        2237
         585
         200
        2102
         662
           ⋮
        -522
        1071
        2912
         613
         710
         605
         193
         949
        -314
        1666
         263
        1376

[137]: t = [x[i-2]+2*x[i-1]-x[i] for i in 3:n]

[137]: 248-element Vector{Int64}:
         246
         394
        1547
         944
        1290
         299
         333
        1061

```

Рис. 3.32: Массивы для пункта 3 (19)

сформируйте вектор $(\frac{\sin(y_1)}{\cos(x_2)}, \frac{\sin(y_2)}{\cos(x_2)}, \dots, \frac{\sin(y_n)}{\cos(x_n)})$;

```
[140]: t = zeros(n-1)
for i in 1:n-1
    t[i] = sin(y[i])/cos(x[i+1])
end
t

[140]: 249-element Vector{Float64}:
-1.1686666955889562
 1.403971127134412
-1.0054338709854956
-0.8029208442815684
 0.7113973991349912
 1.151809485186219
-0.9952706263038429
 1.8003280593915159
 0.3671201374214753
 1.876844639736422
 4.397047895093343
 0.8382507817721872
14.217599614817727
 ⋮
 0.02873301086942607
-0.517666243302685
 3.948599418178391
-1.075697503087067
 1.0811088877801183
 1.0045457607416504
 1.0333681049401044
 0.5590361307058382
 0.80602240574234
-7.672989813951116
 0.49251216587829283
 1.0072070483804159

[141]: t = [sin(y[i-1])/cos(x[i]) for i in 2:n]

[141]: 249-element Vector{Float64}:
-1.1686666955889562
 1.403971127134412
-1.0054338709854956
-0.8029208442815684
 0.7113973991349912
 1.151809485186219
-0.9952706263038429
 1.8003280593915159
 0.3671201374214753
 1.876844639736422
 4.397047895093343
 0.8382507817721872
14.217599614817727
 ⋮
 0.02873301086942607
-0.517666243302685
 3.948599418178391
-1.075697503087067
 1.0811088877801183
 1.0045457607416504
 1.0333681049401044
 0.5590361307058382
 0.80602240574234
-7.672989813951116
 0.49251216587829283
 1.0072070483804159
```

Рис. 3.33: Массивы для пункта 3 (20)

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

```
[143]: summ = 0
for i in 1:n-1
    summ += exp(-x[i+1])/(x[i]+10)
end
summ

[143]: 0.0005550928475862947

[144]: sum([exp(-x[i])/(x[i-1]+10) for i in 2:n])

[144]: 0.0005550928475862946
```

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

```
[146]: t = []
ind = []
for i in 1:n
    if y[i] > 600
        append!(t, y[i]); append!(ind, i)
    end
end
t

[146]: 93-element Vector{Any}:
 618
 868
 922
 773
 753
 786
 686
 972
 672
 912
 793
 808
 655
 ⋮
 951
 710
 781
 ⋮
```

Рис. 3.34: Массивы для пункта 3 (21)

```
[147]: ind
[147]: 93-element Vector{Any}:
 1
 4
 6
 9
11
13
15
18
19
24
28
31
32
:
220
222
227
228
230
232
236
237
239
241
245
249

[130]: t = y[findall(x->x>600, y)]
[130]: 93-element Vector{Int64}:
618
868
922
773
753
786
686
972
672
912
793
888
655
:
951
--~
```

Рис. 3.35: Массивы для пункта 3 (22)

```
[145]: findall(x->x>600, y)
[145]: 93-element Vector{Int64}:
 1
 4
 6
 9
11
13
15
18
19
24
28
31
32
:
220
222
227
228
230
232
236
237
239
241
245
249

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

[151]: hcat(ind, y[ind], x[ind])
[151]: 93x3 Matrix{Any}:
 1 618 699
 4 868 700
 6 922 516
 9 773 711
11 753 118
13 786 864
15 888 400
18 972 751
19 672 15
24 912 265
28 793 10
```

Рис. 3.36: Массивы для пункта 3 (23)

```
[152]: hcat(findall(temp->temp>600, y), y[findall(temp->temp>600, y)], x[findall(temp->temp>600, y)])
```

```
[152]: 93x3 Matrix{Int64}:
 1  618  693
 4  868  700
 6  922  516
 9  773  711
11  753  118
13  786  864
15  686  480
18  972  751
19  672   15
24  912  265
28  793   10
31  808  119
32  659  277
 ⋮
220 951  473
222 710  684
227 781  178
228 697  588
230 800  505
232 908  640
236 952  134
237 932  282
239 986  968
241 671   30
245 764  405
249 676  792
```

сформируйте вектор $(|x_1 - \bar{x}|^{\frac{1}{2}}, |x_2 - \bar{x}|^{\frac{1}{2}}, \dots, |x_n - \bar{x}|^{\frac{1}{2}})$

```
[129]: x_mean = 0
for i in x
    x_mean += 1/n
end
t = [abs(i-x_mean)^(1/2) for i in x]
```

```
[129]: 250-element Vector{Float64}:
 14.913752042997094
 20.750421682462264
  7.511324783285557
15.14661678395542
 9.517352573063581
 6.7394361781977
18.670297265978387
 3.0951575081084295
```

Рис. 3.37: Массивы для пункта 3 (24)

```
[128]: t = abs.(x.-sum(x)/size(x)[1]).^(1/2)
```

```
[128]: 250-element Vector{Float64}:
 14.913752042997094
 20.750421682462264
  7.511324783285557
15.14661678395542
 9.517352573063581
 6.7394361781977
18.670297265978387
 3.0951575081084295
15.50548290121917
18.746199614855275
18.777113729218343
 1.5556349186104097
19.834817871611527
 ⋮
22.30291460773681
22.72487623728675
20.989997617913158
 6.048140210014975
16.810116001979285
16.29662541755194
 8.098147936411138
20.11417410683322
19.504358487271507
19.198437436416537
17.928198092700318
12.672016414130782
```

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

(здесь написано для случая, когда максимальное значение включено)

```
[120]: y_max = 0
for i in y
    if i > y_max
        y_max = i
    end
end
counte = 0
for i in y
    if y_max-i<=200
        counte += 1
    end
end
counte
```

Рис. 3.38: Массивы для пункта 3 (25)

```
[120]: 43

[121]: ymax = maximum(y)
size(findall(x-> ymax-x<=200, y))[1]

[121]: 43

Здесь написано для случая, когда ymax не входит в список элементов, которые подходят под условие

[122]: ymax = 0
for i in y
    if i > ymax
        ymax = i
    end
end
counte = 0
for i in y
    if 0<ymax-i<=200
        counte += 1
    end
end
counte

[122]: 42

[118]: ymax = maximum(y)
size(findall(x-> 0<ymax-x<=200, y))[1]

[118]: 42

определите, сколько чётных и нечётных элементов вектора x;

[116]: counte = [0, 0]
for i in x
    if i % 2 == 0
        counte[1] += 1
    else
        counte[2] += 1
    end
end
counte[1], counte[2]

[116]: (127, 123)
```

Рис. 3.39: Массивы для пункта 3 (26)

```
[112]: size(findall(iseven, x))[1], size(findall(isodd, x))[1]

[112]: (127, 123)

определите, сколько элементов вектора x кратны 7;

[110]: counte = 0
for i in x
    if i % 7 == 0
        counte += 1
    end
end
counte

[110]: 41

[107]: size(findall(x->x%7 == 0, x))[1]

[107]: 41

отсортируйте элементы вектора x в порядке возрастания элементов вектора y;

[106]: num = trunc(Int, n/2)
orde = []
yordered = zeros{Int64, 2, num}
ycopy = copy(y)
for i in 1:num
    tm = [-1, 1000]
    imm = [0, 0]
    for j in 1:size(ycopy)[1]
        if ycopy[j] > tm[1]
            imm[1] = j
            tm[1] = ycopy[j]
        end
        if ycopy[j] < tm[2]
            imm[2] = j
            tm[2] = ycopy[j]
        end
    end
    yordered[1, i] = tm[2]; yordered[2, num-i+1] = tm[1]
    imm = sort(imm, rev=true)
    for k in imm
        ycopy = deleteat!(ycopy, k)
    end
end
```

Рис. 3.40: Массивы для пункта 3 (27)

```

yordered = vcat(yordered[1, :], yordered[2, :])
ycopy = copy(yordered)
while size(ycopy)[1] > 0
    temp = ycopy[1]
    if size(ycopy)[1] > 1
        while ycopy[2] != temp
            deleteat!(ycopy, 2)
        end
    end
    append!(orde, findall(isequal(temp), y))
    deleteat!(ycopy, 1)
end
hcat(yordered, y[orde], x[orde])

[106]: 250x3 Matrix{Int64}:
 6  6  541
 8  8  457
12 12  309
14 14  593
15 15  799
22 22  763
23 23  663
23 23  204
30 30  343
31 31  137
33 33  147
57 57  985
71 71  434
 1
950 950  900
951 951  473
952 952  134
953 953  448
960 960  366
969 969  276
972 972  751
972 972  740
986 986  968
993 993  546
993 993  797
998 998  623

[105]: orde = sortperm(y)
hcat(sort(y), y[orde], x[orde])

[105]: 250x3 Matrix{Int64}:
 6  6  541
 8  8  457
 1  1  100

```

Рис. 3.41: Массивы для пункта 3 (28)

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

```

[97]: t = sort(x, rev=true)[1:10]

[97]: 10-element Vector{Int64}:
 999
 994
 987
 985
 970
 969
 968
 964
 961
 950

[98]: t = []
xstr = join(x, "-")
while size(t)[1] < 10
    temp = 0
    for j in 1:size(x)[1]
        if x[j] >= temp && !(x[j] in t)
            temp = x[j]
        end
    end
    for j in 1:count(string(temp), xstr)
        append!(t, temp)
    end
end
t = t[1:10] # На случай, если дубликатов последнего из top-10 окажется больше одного

[98]: 10-element Vector{Any}:
 999
 994
 987
 985
 970
 969
 968
 964
 961
 950

```

Рис. 3.42: Массивы для пункта 3 (29)

Другой вариант решения будет, если мы должны вывести топ-10 без повторяющихся значений

```
[99]: t = unique(sort(x, rev=true))[1:10]

[99]: 10-element Vector{Int64}:
 999
 994
 987
 985
 970
 969
 968
 964
 961
 950

[100]: t = []
xstr = join(x, "-")
while size(t)[1] < 10
    temp = 0
    for j in 1:size(x)[1]
        if x[j] >= temp && !(x[j] in t)
            temp = x[j]
        end
    end
    append!(t, temp)
end
t

[100]: 10-element Vector{Any}:
 999
 994
 987
 985
 970
 969
 968
 964
 961
 950
```

Рис. 3.43: Массивы для пункта 3 (30)

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

```
[132]: x_unique = []
for i in x
    if !(i in x_unique)
        append!(x_unique, i)
    end
end
x_unique

[132]: 217-element Vector{Any}:
 693
 40
 527
 700
 380
 516
 122
 461
 711
 822
 118
 473
 864
 1
 202
 968
 987
 30
 434
 188
 205
 405
 66
 851
 102
 310

[131]: unique(x)

[131]: 217-element Vector{Int64}:
 693
 40
 527
 700
 380
 516
 122
 461
```

Рис. 3.44: Массивы для пункта 3 (31)

4. Создать массив квадратов натуральных чисел от 1 до 100 (3.45)

4. Создать массив квадратов натуральных чисел от 1 до 100

```
[153]: t = [i**2 for i in 1:100]
[153]: 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. 3.45: Массив натуральных чисел

5. С помощью пакета `Primes` сгенерировать массив первых 168 простых чисел, определим 89-е простое число, создадим срез массива с 89-го по 99-й элементов (3.46)

5. С помощью пакета `Primes` сгенерировать массив первых 168 простых чисел, определим 89-е простое число, создадим срез массива с 89-го по 99-й элементов.

```
[156]: using Primes
      t = [prime(i) for i in 1:168]
[156]: 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
[159]: t[89], t[89:99]
[159]: (461, [461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523])
```

Рис. 3.46: Работа с простыми числами

6. Реализовать формулы (3.47, 3.48)

6. Реализовать формулы

- $\sum_{i=1}^{100} (i^3 + 4i^2)$;
- $\sum_{i=1}^M (\frac{2^i}{i} + \frac{3^i}{i^2})$, $M = 25$;
- $1 + \frac{2}{3} + (\frac{2}{3}\frac{4}{5}) + \dots + (\frac{2}{3}\frac{4}{5} \dots \frac{38}{39}) = 1 + \sum_{i=1}^{19} \prod_{j=1}^i \frac{2j}{2j+1}$

```
[130]: t = [1^3+4*1^2 for i in 10:100]

[130]: 91-element Vector{Int64}:
 1400
 1815
 2304
 2873
 3528
 4275
 5120
 6069
 7128
 8303
 9600
11025
12584
   ⋮
73653
761400
786695
812544
838953
865928
893475
921600
950309
979608
1009503
1040000

[131]: sum(t)

[131]: 26852735

[132]: M = 25
        t = [2^i/i + 3^i/(i^2) for i in 1:M]

[132]: 25-element Vector{Float64}:
 5.0
 4.25
 5.6666666666666666
```

Рис. 3.47: Вычисление выражений (1)

```
299.8888888888889
692.89
1650.206611570248
4031.8958333333335
18064.01775147929
25573.188775510203
65957.45333333334
172247.25390625
454561.89273356396
1.2103058055555555e6
3.24715495567867e6
8.7693898025e6
2.3819486156462584e7
6.502755020041322e7
1.7832914331001893e8
4.910281870572917e8
1.3570039523888e9

[133]: sum(t)

[133]: 2.1291704368143802e9

[134]: t = cat([1], [prod([2j/(2j+1) for j in 1:i]) for i in 1:19], dims=1)

[134]: 20-element Vector{Float64}:
 1.0
 0.6666666666666666
 0.5333333333333333
 0.45714285714285713
 0.4063492063492063
 0.36940836940836935
 0.34099234099234094
 0.3182595182595182
 0.2995383781266054
 0.2837731927515209
 0.27026818357287707
 0.25850974080883893
 0.24816935117648536
 0.2389778937255044
 0.23073727670048702
 0.2232941387424066
 0.2165276496896066
 0.2103411454127607
 0.20465624959079418
 0.19940865344744046

[135]: sum(t)

[135]: 6.976346137897619
```

Рис. 3.48: Вычисление выражений (2)

4 Выводы по проделанной работе

4.1 Вывод

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

Были записаны скринкасты выполнения и защиты лабораторной работы.

Ссылки на скринкасты:

- Выполнение, Youtube
- Выполнение, Rutube
- Защита презентации, Youtube
- Защита презентации, Rutube

Список литературы

1. Лабораторная работа № 2 [Электронный ресурс]. Российский Университет Дружбы Народов имени Патрису Лумумбы, 2023. URL: <https://esystem.rudn.ru/mod/resource/view.php?id=1069831>.

Повторение примеров

Реализация кортежей

```
In [1]: # пустой кортеж:  
( )
```

```
Out[1]: ( )
```

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

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

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

```
Out[3]: (1, 2, 3)
```

```
In [4]: # кортеж из элементов разных типов:  
x2 = (1, 2.0, "tmp")
```

```
Out[4]: (1, 2.0, "tmp")
```

```
In [5]: # именованный кортеж:  
x3 = (a=2, b=1+2)
```

```
Out[5]: (a = 2, b = 3)
```

```
In [6]: # длина кортежа x2:  
length(x2)
```

```
Out[6]: 3
```

```
In [7]: # обратиться к элементам кортежа x2:  
x2[1], x2[2], x2[3]
```

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

```
In [8]: # произвести какую-либо операцию (сложение) с вторым и третьим элементами кортежа x1:  
c = x1[2] + x1[3]
```

```
Out[8]: 5
```

```
In [9]: # обращение к элементам именованного кортежа x3:  
x3.a, x3.b, x3[2]
```

```
Out[9]: (2, 3, 3)
```

```
In [10]: # проверка вхождения элементов tmp и 0 в кортеж x2 (два способа обращения к методу in()):  
in("tmp", x2), 0 in x2
```

```
Out[10]: (true, false)
```

Реализация словарей

```
In [11]: # создать словарь с именем phonebook:  
phonebook = Dict{"Иванов И.И." => ("867-5309", "333-5544"), "Бухгалтерия" => "555-2368"}  
phonebook
```

```
Out[11]: Dict{String, Any} with 2 entries:  
  "Бухгалтерия" => "555-2368"  
  "Иванов И.И." => ("867-5309", "333-5544")
```

```
In [12]: # вывести ключи словаря:  
keys(phonebook)
```

```
Out[12]: KeySet for a Dict{String, Any} with 2 entries. Keys:  
  "Бухгалтерия"  
  "Иванов И.И."
```

```
In [13]: # вывести значения элементов словаря:  
values(phonebook)
```

```
Out[13]: ValueIterator for a Dict{String, Any} with 2 entries. Values:
         "555-2368"
         ("867-5309", "333-5544")
```

```
In [14]: # вывести заданные в словаре пары "ключ - значение":
pairs(phonebook)
```

```
Out[14]: Dict{String, Any} with 2 entries:
         "Бухгалтерия" => "555-2368"
         "Иванов И.И." => ("867-5309", "333-5544")
```

```
In [15]: # проверка вхождения ключа в словарь:
haskey(phonebook, "Иванов И.И.")
```

```
Out[15]: true
```

```
In [16]: # добавить элемент в словарь:
phonebook["Сидоров П.С."] = "555-3344"
```

```
Out[16]: "555-3344"
```

```
In [17]: # удалить ключ и связанные с ним значения из словаря
pop!(phonebook, "Иванов И.И.")
```

```
Out[17]: ("867-5309", "333-5544")
```

```
In [18]: # 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)
```

```
Out[18]: (Dict{String, Real}{"bar" => 13.0, "baz" => 17, "foo" => 0.0}, Dict{String, Real}{"bar" => 42.0, "baz" => 17, "foo" => 0.0})
```

Реализация множеств

```
In [19]: # создать множество из четырёх целочисленных значений:
A = Set{[1, 3, 4, 5]}
```

```
Out[19]: Set{Int64} with 4 elements:
         5
         4
         3
         1
```

```
In [20]: # создать множество из 11 символьных значений:
B = Set{"abracadabra"}
```

```
Out[20]: Set{Char} with 5 elements:
         'a'
         'd'
         'r'
         'k'
         'b'
```

```
In [21]: # проверка эквивалентности двух множеств:
S1 = Set{[1,2]};
S2 = Set{[3,4]};
issetequal(S1,S2)
```

```
Out[21]: false
```

```
In [22]: S3 = Set{[1,2,2,3,1,2,3,2,1]};
S4 = Set{[2,3,1]};
issetequal(S3,S4)
```

```
Out[22]: true
```

```
In [23]: # объединение множеств:
C=union(S1,S2)
C
```

```
Out[23]: Set{Int64} with 4 elements:
         4
         2
         3
         1
```

```
In [24]: # пересечение множеств:
D = intersect(S1,S3)
```

```
D
```

```
Out[24]: Set{Int64} with 2 elements:  
      2  
      1
```

```
In [25]: # разность множеств:  
E = setdiff(S3,S1)  
E
```

```
Out[25]: Set{Int64} with 1 element:  
      3
```

```
In [26]: # проверка вхождения элементов одного множества в другое:  
issubset(S1,S4)
```

```
Out[26]: true
```

```
In [27]: # добавление элемента в множество:  
push!(S4, 99)
```

```
Out[27]: Set{Int64} with 4 elements:  
      2  
      99  
      3  
      1
```

```
In [28]: # удаление последнего элемента множества:  
pop!(S4)
```

```
Out[28]: 2
```

Реализация массивов

```
In [29]: # создание пустого массива с абстрактным типом:  
empty_array_1 = []
```

```
Out[29]: Any[]
```

```
In [30]: # создание пустого массива с конкретным типом:  
empty_array_2 = (Int64)[]  
empty_array_3 = (Float64)[]
```

```
Out[30]: Float64[]
```

```
In [31]: # вектор-столбец:  
a = [1, 2, 3]  
a
```

```
Out[31]: 3-element Vector{Int64}:  
      1  
      2  
      3
```

```
In [32]: # вектор-строка:  
b = [1 2 3]  
b
```

```
Out[32]: 1×3 Matrix{Int64}:  
      1  2  3
```

```
In [33]: # многомерные массивы (матрицы):  
A = [[1, 2, 3] [4, 5, 6] [7, 8, 9]]  
B = [[1 2 3]; [4 5 6]; [7 8 9]]  
A, B
```

```
Out[33]: ([1 4 7; 2 5 8; 3 6 9], [1 2 3; 4 5 6; 7 8 9])
```

```
In [34]: # одномерный массив из 8 элементов (массив $1 \times 8$) со значениями, случайно распределёнными на интервале [0,1]  
c = rand(1,8)  
c
```

```
Out[34]: 1×8 Matrix{Float64}:  
      0.0469965  0.599488  0.252129  0.707828  ...  0.56991  0.376603  0.154653
```

```
In [35]: # многомерный массив $2 \times 3$ (2 строки, 3 столбца) элементов со значениями, случайно распределёнными на интервале [0,1]  
C = rand(2,3)  
C
```

```
Out[35]: 2×3 Matrix{Float64}:  
  0.630296  0.934137  0.81798  
  0.797798  0.703873  0.988166
```

```
In [36]: # трёхмерный массив:  
D = rand(4, 3, 2)  
D
```

```
Out[36]: 4×3×2 Array{Float64, 3}:  
[:, :, 1] =  
  0.290743  0.596179  0.916233  
  0.331316  0.0893986  0.585883  
  0.848618  0.759715  0.616475  
  0.303702  0.878446  0.192259  
  
[:, :, 2] =  
  0.367442  0.330485  0.831731  
  0.834432  0.0939463  0.911308  
  0.936972  0.10629  0.0536688  
  0.820039  0.00735729  0.132789
```

```
In [37]: # массив из квадратных корней всех целых чисел от 1 до 10:  
roots = [sqrt(i) for i in 1:10]  
roots
```

```
Out[37]: 10-element Vector{Float64}:  
 1.0  
 1.4142135623730951  
 1.7320508075688772  
 2.0  
 2.23606797749979  
 2.449489742783178  
 2.6457513110645907  
 2.8284271247461903  
 3.0  
 3.1622776601683795
```

```
In [38]: # 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]  
ar_1
```

```
Out[38]: 5-element Vector{Int64}:  
 3  
 27  
 75  
 147  
 243
```

```
In [39]: # 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)]  
ar_2
```

```
Out[39]: 4-element Vector{Int64}:  
 1  
 9  
 49  
 81
```

```
In [40]: # одномерный массив из пяти единиц:  
ones(5)
```

```
Out[40]: 5-element Vector{Float64}:  
 1.0  
 1.0  
 1.0  
 1.0  
 1.0
```

```
In [41]: # двумерный массив 2×3 из единиц:  
ones(2,3)
```

```
Out[41]: 2×3 Matrix{Float64}:  
 1.0  1.0  1.0  
 1.0  1.0  1.0
```

```
In [42]: # одномерный массив из 4 нулей:  
zeros(4)
```

```
Out[42]: 4-element Vector{Float64}:  
 0.0  
 0.0  
 0.0  
 0.0
```

```
In [43]: # заполнить массив 3x2 цифрами 3.5
fill(3.5,(3,2))
```

```
Out[43]: 3x2 Matrix{Float64}:
 3.5  3.5
 3.5  3.5
 3.5  3.5
```

```
In [44]: # заполнение массива посредством функции repeat():
repeat([1 2],3,3)
```

```
Out[44]: 3x6 Matrix{Int64}:
 1  2  1  2  1  2
 1  2  1  2  1  2
 1  2  1  2  1  2
```

```
In [45]: # преобразование одномерного массива из целых чисел от 1 до 12 в двумерный массив 2x6
a = collect(1:12)
b = reshape(a,(2,6))
```

```
Out[45]: 2x6 Matrix{Int64}:
 1  3  5  7  9 11
 2  4  6  8 10 12
```

```
In [46]: # транспонирование
b'
```

```
Out[46]: 6x2 adjoint{::Matrix{Int64}} with eltype Int64:
 1  2
 3  4
 5  6
 7  8
 9 10
11 12
```

```
In [47]: # транспонирование
c = transpose(b)
c
```

```
Out[47]: 1x8 Matrix{Float64}:
 0.0469965  0.599488  0.252129  0.707828  ...  0.56991  0.376603  0.154653
```

```
In [48]: # массив 10x5 целых чисел в диапазоне [10, 20]:
ar = rand(10:20, 10, 5)
ar
```

```
Out[48]: 10x5 Matrix{Int64}:
17 12 11 12 14
14 13 14 16 20
14 10 19 11 17
17 13 10 17 16
16 10 15 18 15
17 18 16 20 16
11 14 13 15 11
12 19 19 15 20
17 18 20 13 15
11 16 19 11 15
```

```
In [49]: # выбор всех значений строки в столбце 2:
ar[:, 2]
```

```
Out[49]: 10-element Vector{Int64}:
12
13
10
13
10
18
14
19
18
16
```

```
In [50]: # выбор всех значений в столбцах 2 и 5:
ar[:, [2, 5]]
```

```
Out[50]: 10x2 Matrix{Int64}:  
12 14  
13 20  
10 17  
13 16  
10 15  
18 16  
14 11  
19 20  
18 15  
16 15
```

```
In [51]: # все значения строк в столбцах 2, 3 и 4:  
ar[:, 2:4]
```

```
Out[51]: 10x3 Matrix{Int64}:  
12 11 12  
13 14 16  
10 19 11  
13 10 17  
10 15 18  
18 16 20  
14 13 15  
19 19 15  
18 20 13  
16 19 11
```

```
In [52]: # значения в строках 2, 4, 6 и в столбцах 1 и 5:  
ar[[2, 4, 6], [1, 5]]
```

```
Out[52]: 3x2 Matrix{Int64}:  
14 20  
17 16  
17 16
```

```
In [53]: # значения в строке 1 от столбца 3 до последнего столбца:  
ar[1, 3:end]
```

```
Out[53]: 3-element Vector{Int64}:  
11  
12  
14
```

```
In [54]: # сортировка по столбцам:  
sort(ar,dims=1)
```

```
Out[54]: 10x5 Matrix{Int64}:  
11 10 10 11 11  
11 10 11 11 14  
12 12 13 12 15  
14 13 14 13 15  
14 13 15 15 15  
16 14 16 15 16  
17 16 19 16 16  
17 18 19 17 17  
17 18 19 18 20  
17 19 20 20 20
```

```
In [55]: # сортировка по строкам:  
sort(ar,dims=2)
```

```
Out[55]: 10x5 Matrix{Int64}:  
11 12 12 14 17  
13 14 14 16 20  
10 11 14 17 19  
10 13 16 17 17  
10 15 15 16 18  
16 16 17 18 20  
11 11 13 14 15  
12 15 19 19 20  
13 15 17 18 20  
11 11 15 16 19
```

```
In [56]: # поэлементное сравнение с числом (результат - массив логических значений):  
ar .> 14
```



```
Out[56]: 10x5 BitMatrix:
```

```
1 0 0 0 0
0 0 0 1 1
0 0 1 0 1
1 0 0 1 1
1 0 1 1 1
1 1 1 1 1
0 0 0 1 0
0 1 1 1 1
1 1 1 0 1
0 1 1 0 1
```

```
In [57]: # возврат индексов элементов массива, удовлетворяющих условию:
findall(ar .> 14)
```

```
Out[57]: 29-element Vector{CartesianIndex{2}}:
```

```
CartesianIndex{1, 1}
CartesianIndex{4, 1}
CartesianIndex{5, 1}
CartesianIndex{6, 1}
CartesianIndex{9, 1}
CartesianIndex{6, 2}
CartesianIndex{8, 2}
CartesianIndex{9, 2}
CartesianIndex{10, 2}
CartesianIndex{3, 3}
CartesianIndex{5, 3}
CartesianIndex{6, 3}
CartesianIndex{8, 3}
:
CartesianIndex{5, 4}
CartesianIndex{6, 4}
CartesianIndex{7, 4}
CartesianIndex{8, 4}
CartesianIndex{2, 5}
CartesianIndex{3, 5}
CartesianIndex{4, 5}
CartesianIndex{5, 5}
CartesianIndex{6, 5}
CartesianIndex{8, 5}
CartesianIndex{9, 5}
CartesianIndex{10, 5}
```

```
In [58]: findall(x->x>14, ar)
```

```
Out[58]: 29-element Vector{CartesianIndex{2}}:
```

```
CartesianIndex{1, 1}
CartesianIndex{4, 1}
CartesianIndex{5, 1}
CartesianIndex{6, 1}
CartesianIndex{9, 1}
CartesianIndex{6, 2}
CartesianIndex{8, 2}
CartesianIndex{9, 2}
CartesianIndex{10, 2}
CartesianIndex{3, 3}
CartesianIndex{5, 3}
CartesianIndex{6, 3}
CartesianIndex{8, 3}
:
CartesianIndex{5, 4}
CartesianIndex{6, 4}
CartesianIndex{7, 4}
CartesianIndex{8, 4}
CartesianIndex{2, 5}
CartesianIndex{3, 5}
CartesianIndex{4, 5}
CartesianIndex{5, 5}
CartesianIndex{6, 5}
CartesianIndex{8, 5}
CartesianIndex{9, 5}
CartesianIndex{10, 5}
```

Самостоятельная работа

1. Пересечение и объединение множеств $A = \{0, 3, 4, 9\}$, $B = \{1, 3, 4, 7\}$ и $C = \{0, 1, 2, 4, 7, 8, 9\}$

```
In [59]: function peresech(x, y)
```

```

    z = []
    for i in x
        for j in y
            if i == j
                append!(z, i)
            end
        end
    end
    return z
end

function obyedin(x, y...)
    z = x
    for i in y
        for j in i
            if findfirst(isequal(j), z) == nothing
                append!(z, j)
            end
        end
    end
    return sort(z)
end

A = [0, 3, 4, 9]
B = [1, 3, 4, 7]
C = [0, 1, 2, 4, 7, 8, 9]
peresech(A,B), obyedin(A,B)

```

Out[59]: (Any[3, 4], [0, 1, 3, 4, 7, 9])

```
In [60]: a = Set([0, 3, 4, 9]); b = Set(B); c = Set(C)
a, b, c
```

Out[60]: (Set([0, 4, 9, 3]), Set([4, 7, 3, 1]), Set([0, 4, 7, 2, 9, 8, 1]))

```
In [61]: intersect(a,b), union(a,b)
```

Out[61]: (Set([4, 3]), Set([0, 4, 7, 9, 3, 1]))

Вычисляем формулу из задания: $P = (A \cap B) \cup (A \cap B) \cup (A \cap C) \cup (B \cap C)$ (разделена скобками для упрощения восприятия)

```
In [62]: t1 = peresech(A,B)
t2 = peresech(A,C)
t3 = peresech(B,C)
P = obyedin(t1,t1,t2,t3)
```

Out[62]: 6-element Vector{Any}:
0
1
3
4
7
9

2. Примеры с выполнением операций над множествами элементов разных типов

```
In [63]: t = Set([1 2 3; 4 5 6; "Бу" "Ка" "Бака"])
```

Out[63]: Set{Any} with 9 elements:
5
4
"Ka"
6
2
"Бака"
"Бу"
3
1

```
In [64]: 1 in t
```

Out[64]: true

```
In [65]: 0 in t
```

Out[65]: false

```
In [66]: [1, 2, 3] in t
```

```
Out[66]: false
```

```
In [67]: intersect(Set([1, 2, 3, 4, 7, 8, 10]), t)
```

```
Out[67]: Set{Any} with 4 elements:  
4  
2  
3  
1
```

```
In [68]: union(Set([1, 2, 3, 4, 7, 8, 10]), t)
```

```
Out[68]: Set{Any} with 12 elements:  
5  
"Ka"  
7  
"Бака"  
8  
"Бу"  
1  
4  
6  
2  
10  
3
```

```
In [69]: setdiff(Set([1, 2, 3, 4, 7, 8, 10]), t)
```

```
Out[69]: Set{Int64} with 3 elements:  
7  
10  
8
```

```
In [70]: setdiff(t, Set([1, 2, 3, 4, 7, 8, 10]))
```

```
Out[70]: Set{Any} with 5 elements:  
5  
"Ka"  
6  
"Бака"  
"Бу"
```

3. Создать разными способами несколько видов массивов

1. массив (1, 2, 3, ...N – 1, N), N выберите больше 20

```
In [71]: N = 100  
t = zeros(Int64, N)  
for i in 1:N  
    t[i] = i  
end  
t
```

```
Out[71]: 100-element Vector{Int64}:  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
⋮  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100
```

```
In [72]: t = collect(1:N)
t
```

```
Out[72]: 100-element Vector{Int64}:
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
 ⋮
89
90
91
92
93
94
95
96
97
98
99
100
```

```
In [73]: t = [i for i in 1:N]
```

```
Out[73]: 100-element Vector{Int64}:
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
 ⋮
89
90
91
92
93
94
95
96
97
98
99
100
```

2. массив (N, N – 1..., 2, 1), N выберите больше 20

```
In [74]: t = zeros{Int64, N}
for i in N:-1:1
    t[N-i+1] = i
end
t
```

```
Out[74]: 100-element Vector{Int64}:
 100
  99
  98
  97
  96
  95
  94
  93
  92
  91
  90
  89
  88
   ⋮
  12
  11
  10
   9
   8
   7
   6
   5
   4
   3
   2
   1
```

```
In [75]: t = collect(N:-1:1)
t
```

```
Out[75]: 100-element Vector{Int64}:
 100
  99
  98
  97
  96
  95
  94
  93
  92
  91
  90
  89
  88
   ⋮
  12
  11
  10
   9
   8
   7
   6
   5
   4
   3
   2
   1
```

```
In [76]: t = [N-i+1 for i in 1:N]
```

```
Out[76]: 100-element Vector{Int64}:
 100
  99
  98
  97
  96
  95
  94
  93
  92
  91
  90
  89
  88
   ⋮
  12
  11
  10
   9
   8
   7
   6
   5
   4
   3
   2
   1
```

3. массив (1, 2, 3, ..., N - 1, N, N - 1, ..., 2, 1), N выберите больше 20

```
In [77]: t = zeros{Int64, 2*N-1}
for i in 1:N
    t[i] = i
end
for i in 1:N-1
    t[i+N] = N-i
end
t
```

```
Out[77]: 199-element Vector{Int64}:
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
   ⋮
 12
 11
 10
  9
  8
  7
  6
  5
  4
  3
  2
  1
```

```
In [78]: t = cat([i for i in 1:N], [i for i in N-1:-1:1], dims=1)
```

```
Out[78]: 199-element Vector{Int64}:
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
⋮
12
11
10
9
8
7
6
5
4
3
2
1
```

```
In [79]: t = vcat(collect(1:N), collect(N:-1:1))
```

```
Out[79]: 200-element Vector{Int64}:
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
⋮
12
11
10
9
8
7
6
5
4
3
2
1
```

4. массив с именем `tmp` вида (4, 6, 3)

```
In [80]: t1 = 4; t2 = 6; t3 = 3
tmp = [t1, t2, t3]
```

```
Out[80]: 3-element Vector{Int64}:
```

```
4
6
3
```

```
In [81]: tmp = [4, 6, 3]
```

```
Out[81]: 3-element Vector{Int64}:
```

```
4
6
3
```

5. массив, в котором первый элемент массива `tmp` повторяется 10 раз

```
In [82]: t1 = zeros{Int64, 10}
for i in 1:10
    t1[i] = tmp[1]
end
```

```
t1
```

```
Out[82]: 10-element Vector{Int64}:  
 4  
 4  
 4  
 4  
 4  
 4  
 4  
 4  
 4  
 4
```

```
In [83]: t1 = [tmp[1] for i in 1:10]  
t1
```

```
Out[83]: 10-element Vector{Int64}:  
 4  
 4  
 4  
 4  
 4  
 4  
 4  
 4  
 4  
 4
```

```
In [84]: t1 = fill(tmp[1], 10)
```

```
Out[84]: 10-element Vector{Int64}:  
 4  
 4  
 4  
 4  
 4  
 4  
 4  
 4  
 4  
 4
```

6. массив, в котором все элементы массива `tmp` повторяются 10 раз

```
In [85]: t2 = zeros{Int64, 30}  
for i in 0:9  
    t2[3*i+1] = tmp[1]  
    t2[3*i+2] = tmp[2]  
    t2[3*i+3] = tmp[3]  
end  
t2
```

```
Out[85]: 30-element 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
```

```
In [86]: t2 = repeat(tmp, 10)
```



```
t2
```

```
Out[86]: 30-element 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
```

7. массив, в котором первый элемент массива `tmp` встречается 11 раз, второй элемент — 10 раз, третий элемент — 10 раз

```
In [87]: t3 = [tmp[1] for i in 1:11]
for i in 1:10
    append!(t3, tmp[2])
    append!(t3, tmp[3])
end
t3
```

```
Out[87]: 31-element Vector{Int64}:
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
 6
 3
 6
 3
 6
 3
 6
 3
 6
 3
 6
 3
 6
 3
 6
 3
 6
 3
 6
 3
```

```
In [88]: t3 = repeat(tmp, 10)
append!(t3, tmp[1])
t3
```

```
Out[88]: 31-element 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
 4
 6
 3
 4
```

```
In [89]: t3 = cat([tmp[1] for i in 1:11], [tmp[2] for i in 1:10], [tmp[3] for i in 1:10], dims=1)
```

```
Out[89]: 31-element Vector{Int64}:
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
 6
 6
 6
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
```

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

```
In [90]: t3 = [tmp[1] for i in 1:10]
for i in 1:20
    append!(t3, tmp[2])
    append!(t3, tmp[3])
end
for i in 1:10
    append!(t3, tmp[3])
end
t3
```

Out[90]: 60-element Vector{Int64}:

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

```
In [91]: t3 = repeat(tmp, 10)
for i in 1:10
    append!(t3, tmp[2])
    append!(t3, tmp[3])
end
for i in 1:10
    append!(t3, tmp[3])
end
t3
```

Out[91]: 60-element Vector{Int64}:

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

```
In [92]: t3 = cat([tmp[1] for i in 1:10], [tmp[2] for i in 1:20], [tmp[3] for i in 1:30], dims=1)
```



```
Out[95]: 12-element Vector{Int64}:
 16
 64
  8
 16
 64
  8
 16
 64
  8
 16
 64
  8
```

Сосчитаем количество 6 в векторе

```
In [96]: size(findall(isequal("6"), split(join(string.(t2)), "")))[1]
```

```
Out[96]: 8
```

```
In [97]: count("6", join(string.(t2)))
```

```
Out[97]: 8
```

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

```
In [98]: t1 = collect(3:0.1:6)
t = ones(size(t1)[1])
for i in 1:size(t1)[1]
    t[i] = exp(t1[i])*cos(t1[i])
end
t
```

```
Out[98]: 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
```

```
In [99]: t = [exp(i)*cos(i) for i in 3:0.1:6]
```

```
Out[99]: 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
```

Среднее значение вектора y

```
In [100...] sum(t)/size(t)[1]
```

```
Out[100...] 53.11374594642971
```

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

```
In [101...] t1 = collect(3:3:36)
t2 = collect(1:3:34)
t = zeros((size(t1)[1], 2))
for i in 1:size(t1)[1]
    t[i,1] = 0.1^t1[i]
    t[i,2] = 0.2^t2[i]
end
t
```

```
Out[101...] 12x2 Matrix{Float64}:
 0.001  0.2
 1.0e-6  0.0016
 1.0e-9  1.28e-5
 1.0e-12 1.024e-7
 1.0e-15 8.192e-10
 1.0e-18 6.5536e-12
 1.0e-21 5.24288e-14
 1.0e-24 4.1943e-16
 1.0e-27 3.35544e-18
 1.0e-30 2.68435e-20
 1.0e-33 2.14748e-22
 1.0e-36 1.71799e-24
```

```
In [102...] t = zeros((size(collect(3:3:36))[1], 2))
for i in 0:size(collect(3:3:36))[1]-1
    t[i+1, 1] = 0.1^(3*(i+1))
    t[i+1, 2] = 0.2^(3*i+1)
end
t
```

```
Out[102...] 12x2 Matrix{Float64}:
 0.001  0.2
 1.0e-6  0.0016
 1.0e-9  1.28e-5
 1.0e-12 1.024e-7
 1.0e-15 8.192e-10
 1.0e-18 6.5536e-12
 1.0e-21 5.24288e-14
 1.0e-24 4.1943e-16
 1.0e-27 3.35544e-18
 1.0e-30 2.68435e-20
 1.0e-33 2.14748e-22
 1.0e-36 1.71799e-24
```

```
In [103...] t = hcat(fill(0.1, size(collect(3:3:36))[1]), fill(0.2, size(collect(1:3:34))[1]))
```

```
t[:, 1] = t[:, 1].^collect(3:3:36)
t[:, 2] = t[:, 2].^collect(1:3:34)
t
```

```
Out[103... 12x2 Matrix{Float64}:
 0.001    0.2
 1.0e-6    0.0016
 1.0e-9    1.28e-5
 1.0e-12   1.024e-7
 1.0e-15   8.192e-10
 1.0e-18   6.5536e-12
 1.0e-21   5.24288e-14
 1.0e-24   4.1943e-16
 1.0e-27   3.35544e-18
 1.0e-30   2.68435e-20
 1.0e-33   2.14748e-22
 1.0e-36   1.71799e-24
```

12. вектор с элементами 2^i , $i = 1, 2, \dots, M$, $M = 25$

```
In [104... M = 25
t = [2^i/i for i in 1:M]
```

```
Out[104... 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
```

```
In [105... t = zeros(M)
for i in 1:M
    t[i] = 2^i/i
end
t
```

```
Out[105... 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
```

```
In [151...] t = fill(2, M).^collect(1:M)./collect(1:M)
t
```

```
Out[151...] 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
```

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

```
In [107...] N = 30
t = [join(["fn", string(i)]) for i in 1:N]
```

```
Out[107...] 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"
```

```
In [108...] t1 = string.(collect(1:N))
t2 = fill("fn", N)
t = join.([t2[i],t1[i]] for i in 1:N)
```



```

Out[108.. 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"

```

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 $0, 1, \dots, 999$ и на их основе создать следующие векторы:

```

In [109.. n = 250
x, y = rand(0:999, n), rand(0:999, n)

```

```

Out[109.. ([599, 798, 936, 146, 933, 901, 959, 887, 739, 822 ... 745, 213, 881, 223, 854, 871, 95, 349, 956, 822], [131,
787, 473, 713, 637, 68, 823, 807, 781, 60 ... 797, 702, 489, 842, 234, 934, 675, 366, 512, 25])

```

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

```

In [110.. t = zeros{Int64, n-1}
for i in 1:n-1
    t[i] = y[i+1]-x[i]
end
t

```

```

Out[110.. 249-element Vector{Int64}:
 188
-325
-223
 491
-865
 -78
-152
-106
-679
-463
-916
 777
 660
 ⋮
-151
 337
  30
 -43
 276
 -39
  11
  80
-196
 271
 163
-931

```

```

In [111.. t = [y[i+1]-x[i] for i in 1:n-1]

```

```
Out{111... 249-element Vector{Int64}:
 188
-325
-223
 491
-865
 -78
-152
-106
-679
-463
-916
 777
 660
   ⋮
-151
 337
  30
 -43
 276
 -39
  11
  80
-196
 271
 163
-931
```

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

```
In [112... t = zeros{Int64, n-2}
for i in 1:n-2
    t[i] += x[i]
    t[i] += 2*x[i+1]
    t[i] -= x[i+2]
end
t
```

```
Out{112... 248-element Vector{Int64}:
 1259
 2524
  295
 1111
 1776
 1932
 1994
 1543
 1402
 2567
 1199
  546
  317
   ⋮
 2448
  350
  906
 2044
  290
 1752
  473
 1060
 2501
  712
 -163
 1439
```

```
In [113... t = [x[i-2]+2*x[i-1]-x[i] for i in 3:n]
```

```

Out[113... 248-element Vector{Int64}:
 1259
 2524
 295
 1111
 1776
 1932
 1994
 1543
 1402
 2567
 1199
 546
 317
  ⋮
 2448
 350
 906
 2044
 290
 1752
 473
 1060
 2501
 712
 -163
 1439

```

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

```

In [114... t = zeros(n-1)
for i in 1:n-1
    t[i] = sin(y[i])/cos(x[i+1])
end
t

```

```

Out[114... 249-element Vector{Float64}:
 -0.8121140868985631
  1.0187492027089577
 11.69548660066726
 -0.14125880958717973
 -0.8418797459810125
  1.3081300760049281
 -0.20260769269220813
 -0.5074105256897286
  2.085501215598547
 -0.4482791364055335
 -0.7773105356425388
 -1.1515959007373768
 -1.2158591632370919
  ⋮
  1.313422206114909
 -0.009903527710614974
 -1.1057240212368369
 -1.0157669033063539
 -4.600714349902583
  0.8871602467130308
  0.06100228839913869
 -1.4033598600537407
 -1.1115935429148585
 -0.44589167993107853
  1.7333003856221905
  0.17437395879240855

```

```

In [115... t = [sin(y[i-1])/cos(x[i]) for i in 2:n]

```

```
Out[115...] 249-element Vector{Float64}:
 -0.8121140868985631
  1.0187492027089577
 11.69548660066726
 -0.14125880958717973
 -0.8418797459810125
  1.3081300760049281
 -0.20260769269220813
 -0.5074105256897286
  2.085501215598547
 -0.4482791364055335
 -0.7773105356425388
 -1.1515959007373768
 -1.2158591632370919
      ⋮
  1.313422206114909
 -0.009903527710614974
 -1.1057240212368369
 -1.0157669033063539
 -4.600714349902583
  0.8871602467130308
  0.06100228839913869
 -1.4033598600537407
 -1.1115935429148585
 -0.44589167993107853
  1.7333003856221905
  0.17437395879240855
```

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

```
In [116...] summ = 0
for i in 1:n-1
    summ += exp(-x[i+1])/(x[i]+10)
end
summ
```

```
Out[116...] 2.1267687767329432e-8
```

```
In [117...] sum([exp(-x[i])/(x[i-1]+10) for i in 2:n])
```

```
Out[117...] 2.1267687767329432e-8
```

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

```
In [118...] t = []
ind = []
for i in 1:n
    if y[i] > 600
        append!(t, y[i]); append!(ind, i)
    end
end
t
```

```
Out[118.. 111-element Vector{Any}:  
 787  
 713  
 637  
 823  
 807  
 781  
 994  
 876  
 878  
 746  
 824  
 707  
 980  
  ⋮  
 858  
 665  
 684  
 738  
 915  
 727  
 732  
 797  
 702  
 842  
 934  
 675
```

```
In [119.. ind
```

```
Out[119.. 111-element Vector{Any}:  
 2  
 4  
 5  
 7  
 8  
 9  
 13  
 14  
 15  
 17  
 20  
 21  
 24  
  ⋮  
229  
230  
231  
233  
234  
237  
239  
241  
242  
244  
246  
247
```

```
In [120.. t = y[findall(x->x>600, y)]
```

```
Out[120...] 111-element Vector{Int64}:  
 787  
 713  
 637  
 823  
 807  
 781  
 994  
 876  
 878  
 746  
 824  
 707  
 980  
  ⋮  
 858  
 665  
 684  
 738  
 915  
 727  
 732  
 797  
 702  
 842  
 934  
 675
```

```
In [121...] findall(x->x>600, y)
```

```
Out[121...] 111-element Vector{Int64}:  
 2  
 4  
 5  
 7  
 8  
 9  
13  
14  
15  
17  
20  
21  
24  
  ⋮  
229  
230  
231  
233  
234  
237  
239  
241  
242  
244  
246  
247
```

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

```
In [122...] hcat(ind, y[ind], x[ind])
```

Out[122... 111×3 Matrix{Any}:

```
2 787 798
4 713 146
5 637 933
7 823 959
8 807 887
9 781 739
13 994 216
14 876 103
15 878 105
17 746 736
20 824 473
21 707 430
24 980 775
⋮
229 858 971
230 665 132
231 684 38
233 738 246
234 915 749
237 727 799
239 732 117
241 797 745
242 702 213
244 842 223
246 934 871
247 675 95
```

In [123... `hcat(findall(temp->temp>600, y), y[findall(temp->temp>600, y)], x[findall(temp->temp>600, y)])`

Out[123... 111×3 Matrix{Int64}:

```
2 787 798
4 713 146
5 637 933
7 823 959
8 807 887
9 781 739
13 994 216
14 876 103
15 878 105
17 746 736
20 824 473
21 707 430
24 980 775
⋮
229 858 971
230 665 132
231 684 38
233 738 246
234 915 749
237 727 799
239 732 117
241 797 745
242 702 213
244 842 223
246 934 871
247 675 95
```

сформируйте вектор $(|x_1 - \bar{x}|^{12}, |x_2 - \bar{x}|^{12}, \dots, |x_n - \bar{x}|^{12})$

In [124...

```
x_mean = 0
for i in x
    x_mean += i/n
end
t = [abs(i-x_mean)^(1/2) for i in x]
```

```
Out[124... 250-element Vector{Float64}:
 9.215638881813891
16.850163203957404
20.54088605683796
19.1852026311947
20.467730699811348
19.67048550493861
21.093316477026555
19.31134381652401
14.997599807969273
17.54787736451335
21.608516839431623
17.235776744898967
17.264761799688984
 ⋮
19.926665551466456
15.903710258930147
15.196315342871772
17.35142645432934
19.15536478378838
17.060832336084896
18.43713643709348
18.892538209568347
20.471248129999303
12.848034869193032
21.02208362651048
17.54787736451335
```

```
In [125... t = abs.(x.-sum(x)/size(x)[1]).^(1/2)
```

```
Out[125... 250-element Vector{Float64}:
 9.215638881813891
16.850163203957404
20.54088605683796
19.1852026311947
20.467730699811348
19.67048550493861
21.093316477026555
19.31134381652401
14.997599807969273
17.54787736451335
21.608516839431623
17.235776744898967
17.264761799688984
 ⋮
19.926665551466456
15.903710258930147
15.196315342871772
17.35142645432934
19.15536478378838
17.060832336084896
18.43713643709348
18.892538209568347
20.471248129999303
12.848034869193032
21.02208362651048
17.54787736451335
```

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

(здесь написано для случая, когда максимальное значение включено)

```
In [126... ymax = 0
for i in y
    if i > ymax
        ymax = i
    end
end
counte = 0
for i in y
    if ymax-i<=200
        counte += 1
    end
end
counte
```

```
Out[126... 49
```

```
In [127... ymax = maximum(y)
size(findall(x-> ymax-x<=200, y))[1]
```

```
Out[127... 49
```


Здесь написано для случая, когда `ymax` не входит в список элементов, которые подходят под условие

```
In [128... ymax = 0
for i in y
    if i > ymax
        ymax = i
    end
end
counte = 0
for i in y
    if 0 < ymax - i <= 200
        counte += 1
    end
end
counte
```

Out[128... 48

```
In [129... ymax = maximum(y)
size(findall(x-> 0 < ymax - x <= 200, y))[1]
```

Out[129... 48

определите, сколько чётных и нечётных элементов вектора `x`;

```
In [130... counte = [0, 0]
for i in x
    if i % 2 == 0
        counte[1] += 1
    else
        counte[2] += 1
    end
end
counte[1], counte[2]
```

Out[130... (130, 120)

```
In [131... size(findall(iseven, x))[1], size(findall(isodd, x))[1]
```

Out[131... (130, 120)

определите, сколько элементов вектора `x` кратны 7;

```
In [132... counte = 0
for i in x
    if i % 7 == 0
        counte += 1
    end
end
counte
```

Out[132... 35

```
In [133... size(findall(x->x%7 == 0, x))[1]
```

Out[133... 35

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

```
In [134... num = trunc(Int, n/2)
orde = []
yordered = zeros{Int64, 2, num}
ycopy = copy(y)
for i in 1:num
    tm = [-1, 1000]
    imm = [0, 0]
    for j in 1:size(ycopy)[1]
        if ycopy[j] > tm[1]
            imm[1] = j
            tm[1] = ycopy[j]
        end
        if ycopy[j] < tm[2]
            imm[2] = j
            tm[2] = ycopy[j]
        end
    end
    yordered[1, i] = tm[2]; yordered[2, num-i+1] = tm[1]
    imm = sort(imm, rev=true)
    for k in imm
```

```

        ycopy = deleteat!(ycopy, k)
    end
end
yordered = vcat(yordered[1, :], yordered[2, :])
ycopy = copy(yordered)
while size(ycopy)[1] > 0
    temp = ycopy[1]
    if size(ycopy)[1] > 1
        while ycopy[2] == temp
            deleteat!(ycopy, 2)
        end
    end
    append!(orde, findall(isequal(temp), y))
    deleteat!(ycopy, 1)
end
hcat(yordered, y[orde], x[orde])

```

Out[134]... 250×3 Matrix{Int64}:

```

16  16  214
17  17  899
21  21  425
22  22  331
25  25  822
27  27  885
31  31  121
32  32  607
34  34  449
43  43  304
58  58  102
59  59   92
60  60  822
⋮
970 970  640
970 970   76
974 974  818
974 974  319
975 975  238
977 977  976
980 980  775
981 981  382
983 983  976
993 993  628
994 994  216
998 998  194

```

In [135]... orde = sortperm(y)
hcat(sort(y), y[orde], x[orde])

Out[135]... 250×3 Matrix{Int64}:

```

16  16  214
17  17  899
21  21  425
22  22  331
25  25  822
27  27  885
31  31  121
32  32  607
34  34  449
43  43  304
58  58  102
59  59   92
60  60  822
⋮
970 970  640
970 970   76
974 974  818
974 974  319
975 975  238
977 977  976
980 980  775
981 981  382
983 983  976
993 993  628
994 994  216
998 998  194

```

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

In [137]... t = []
xstr = join(x, "-")
while size(t)[1] < 10
 temp = 0
 for j in 1:size(x)[1]

```

        if x[j] >= temp && !(x[j] in t)
            temp = x[j]
        end
    end
    for j in 1:count(string(temp), xstr)
        append!(t, temp)
    end
end
t = t[1:10] # На случай, если дубликатов последнего из top-10 окажется больше одного

```

Out[137... 10-element Vector{Any}:

```

998
998
995
992
992
987
981
976
976
976

```

In [136... t = sort(x, rev=true)[1:10]

Out[136... 10-element Vector{Int64}:

```

998
998
995
992
992
987
981
976
976
976

```

Другой вариант решения будет, если мы должны выписать top-10 без повторяющихся значений

```

In [139... t = []
xstr = join(x, ".")
while size(t)[1] < 10
    temp = 0
    for j in 1:size(x)[1]
        if x[j] >= temp && !(x[j] in t)
            temp = x[j]
        end
    end
    append!(t, temp)
end
t

```

Out[139... 10-element Vector{Any}:

```

998
995
992
987
981
976
974
971
970
965

```

In [138... t = unique(sort(x, rev=true))[1:10]

Out[138... 10-element Vector{Int64}:

```

998
995
992
987
981
976
974
971
970
965

```

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

```

In [140... x_unique = []
for i in x
    if !(i in x_unique)
        append!(x_unique, i)
    end
end

```

```
end
end
x_unique
```

```
Out[140...] 218-element Vector{Any}:
 599
 798
 936
 146
 933
 901
 959
 887
 739
 822
 981
 217
 216
  ⋮
 799
 883
 117
 767
 213
 881
 223
 854
 871
  95
 349
 956
```

```
In [141...] unique(x)
```

```
Out[141...] 218-element Vector{Int64}:
 599
 798
 936
 146
 933
 901
 959
 887
 739
 822
 981
 217
 216
  ⋮
 799
 883
 117
 767
 213
 881
 223
 854
 871
  95
 349
 956
```

4. Создать массив квадратов натуральных чисел от 1 до 100

```
In [142...] t = [i^2 for i in 1:100]
```

```
Out[142... 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
```

5. С помощью пакета `Primes` сгенерировать массив первых 168 простых чисел, определим 89-е простое число, создадим срез массива с 89-го по 99-й элементов.

```
In [152... using Primes  
t = [prime(i) for i in 1:168]
```

```
Out[152... 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
```

```
In [153... t[89]
```

```
Out[153... 461
```

```
In [154... t[89:99]
```

```
Out[154...] 11-element Vector{Int64}:
 461
 463
 467
 479
 487
 491
 499
 503
 509
 521
 523
```

6. Реализовать формулы

- $\sum_{100i=10} (i^3 + 4i^2);$
- $\sum_{M=1} (2^i + 3i^2), M = 25;$
- $1 + 23 + (2345) + \dots + (2345 \dots 3839) = 1 + \sum_{19i=1} \prod_{ij=1}^{2j2j+1}$

```
In [145...] t = [i^3+4*i^2 for i in 10:100]
```

```
Out[145...] 91-element Vector{Int64}:
 1400
 1815
 2304
 2873
 3528
 4275
 5120
 6069
 7128
 8303
 9600
11025
12584
   ⋮
736653
761400
786695
812544
838953
865928
893475
921600
950309
979608
1009503
1040000
```

```
In [146...] sum(t)
```

```
Out[146...] 26852735
```

```
In [147...] M = 25
t = [2^i/i + 3^i/(i^2) for i in 1:M]
```

Out[147... 25-element Vector{Float64}:
5.0
4.25
5.6666666666666666
9.0625
16.12
30.916666666666664
62.91836734693877
134.515625
299.88888888888889
692.89
1650.206611570248
4031.8958333333335
10064.01775147929
25573.188775510203
65957.45333333334
172247.25390625
454561.89273356396
1.2103058055555555e6
3.24715495567867e6
8.7693898025e6
2.3819486156462584e7
6.502755020041322e7
1.7832914331001893e8
4.910281070572917e8
1.3570039523888e9

In [148... sum(t)

Out[148... 2.1291704368143802e9

In [149... t = cat([1], [prod([2j/(2j+1) for j in 1:i]) for i in 1:19], dims=1)

Out[149... 20-element Vector{Float64}:
1.0
0.6666666666666666
0.5333333333333333
0.45714285714285713
0.4063492063492063
0.36940836940836935
0.34099234099234094
0.3182595182595182
0.2995383701266054
0.2837731927515209
0.27026018357287707
0.25850974080883893
0.24816935117648536
0.2389778937255044
0.23073727670048702
0.2232941387424068
0.2165276496896066
0.2103411454127607
0.20465624959079418
0.19940865344744046

In [150... sum(t)

Out[150... 6.976346137897619

In []:

Processing math: 100%