Лабораторная работа №2

Дисциплина: Компьютерный практикум по статистическому моделированию

Манаева Варвара Евгеньевна

Содержание

Список иллюстраций

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 Pycharm 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)

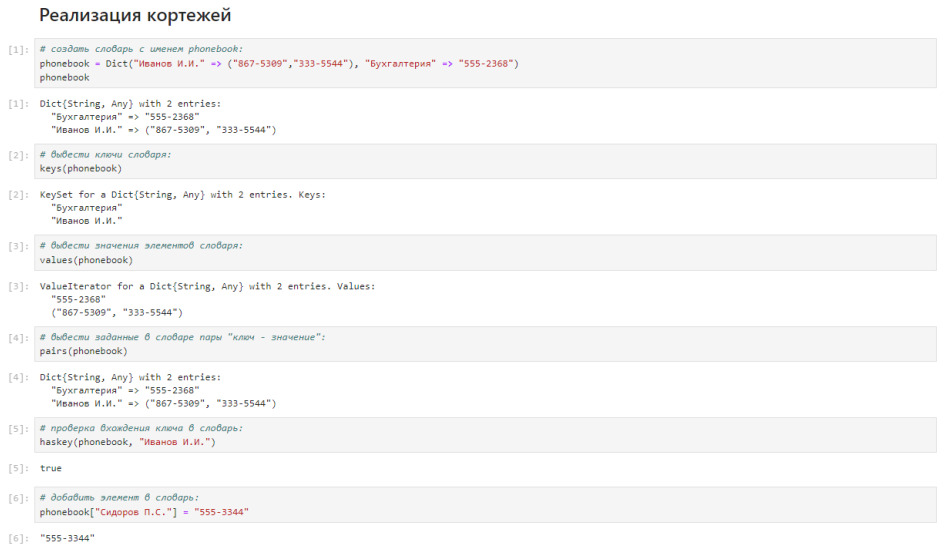

Рис. 3.1: Повторение примеров (1)

[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), merge(b,a)
```

[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("abrakadabra")
```

[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([1,2,2,3,1,2,3,2,1]);
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:
  2
  99
  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]: 1×3 Matrix{Int64}:
       1  2  3

[23]: # 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

[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]: 1×8 Matrix{Float64}:
       0.768374  0.411788  0.648862  0.193864  …  0.899928  0.363381  0.0207368

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

[25]: 2×3 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]: 4×3×2 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*x^2, где x - 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 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
```

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

```
[36]: # транспонирование
b'
```

```
[36]: 6×2 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]: 1×8 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]: 10×5 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 в столбце 2:
ar[:, 2]
```

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

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

```
[40]: 10×2 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 в столбцах 2, 3 и 4:
ar[:, 2:4]
```

```
[41]: 10×3 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 в строках 2, 4, 6 и в столбцах 1 и 5:
ar[[2, 4, 6], [1, 5]]
```

```
[42]: 3×2 Matrix{Int64}:
 10  15
 14  10
 11  17
```

```
[43]: # значения в 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]: 10×5 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]: 10×5 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]: 10×5 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(8, 1)
       CartesianIndex(10, 1)
       CartesianIndex(1, 2)
       CartesianIndex(2, 2)
       CartesianIndex(4, 2)
       CartesianIndex(5, 2)
       CartesianIndex(7, 2)
       CartesianIndex(8, 2)
       CartesianIndex(9, 2)
       CartesianIndex(10, 2)
       CartesianIndex(5, 3)
       CartesianIndex(9, 3)
       CartesianIndex(1, 4)
       CartesianIndex(2, 4)
       CartesianIndex(3, 4)
       CartesianIndex(4, 4)
       CartesianIndex(6, 4)
       CartesianIndex(1, 5)
       CartesianIndex(2, 5)
       CartesianIndex(6, 5)
       CartesianIndex(8, 5)
       CartesianIndex(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 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)
```

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

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

[94]:
```
t1 = peresech(A,B)
t2 = peresech(A,C)
t3 = peresech(B,C)
P = obyedin(t1,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; "Бу" "Ка" "Бака"])
```

[63]:
```
Set{Any} with 9 elements:
  5
  4
  "Ка"
  6
  2
  "Бака"
  "Бу"
  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
  "Ка"
  7
  "Бака"
  8
  "Бу"
  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
        "Ка"
        6
        "Бака"
        "Ву"
```

Рис. 3.13: Примеры выполнения операций

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

1. массив $(1, 2, 3, \ldots N-1, N)$. 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 $(N, N-1 \ldots, 2, 1)$. 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
         ⋮
```

Рис. 3.16: Массивы для пункта 3 (3)

3. массив $(1, 2, 3, \ldots, N-1, N, N-1, \ldots, 2, 1)$. 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
        ⋮
```

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
```

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
 ⋮
 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
 ⋮
 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

```
[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
```

Рис. 3.20: Массивы для пункта 3 (7)

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

```
[115]: 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
```

```
[117]: 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)
```

```
[117]: 31-element Vector{Int64}:
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
 6
 6
 ⋮
```

Рис. 3.21: Массивы для пункта 3 (8)

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

```
[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
 4
 4
 4
 6
 3
 6
 ⋮
 6
 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])
```

Рис. 3.22: Массивы для пункта 3 (9)

```
end
t3
```

[121]:
```
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
```

[120]:
```
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)
```

[120]:
```
60-element Vector{Int64}:
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
 6
 6
 6
 ⋮
 3
```

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 $x = 3, 3.1, 3.2, \ldots, 6$. найдите среднее значение y

```
[139]: 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
```

```
[139]: 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
   ⋮
```

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
 -28.969237768093574
 -31.011186439374516
 -32.819774760338504
 -34.30336011037369
 -35.35719361853035
 -35.86283371230767
 -35.68773248011913
 -34.68504225166807
 -32.693695428321746
   ⋮
  25.046704998273004
  42.09920106253839
  61.996630276694454
  84.92906736250268
 111.06158604202 58
 140.5250750527875
 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^j, y^j). $x = 0.1, \quad i = 3, 6, 9, \ldots, 36, \quad y = 0.2, \quad j = 1, 4, 7, \ldots, 34$

```
[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]:
```
12×2 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]:
```
12×2 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]:
```
12×2 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 $\frac{2^i}{i}, i = 1, 2, \ldots, M$. $M = 25$

[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
   52428.8
   99864.38095238095
  190650.18181818182
  364722.0869565217
  699050.6666666666
       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
    99864.38095238095
   190650.18181818182
   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 $x = (x_1, x_2, \ldots, x_n)$ и $y = (y_1, y_2, \ldots, y_n)$ целочисленного типа длины $n = 250$ как случайные выборки из совокупности $0, 1, \ldots, 999$ 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 $(y_2 - x_1, \ldots, y_n - x_{n-1})$;

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
```

Рис. 3.31: Массивы для пункта 3 (18)

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

[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 $(\frac{sin(y_1)}{cos(x_2)}, \frac{sin(y_2)}{cos(x_3)}, \ldots, \frac{sin(y_{n-1})}{cos(x_n)})$;

[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
 -1.0654338709854956
 -0.8029208442815684
  0.7113973991349912
  1.151809485186219
 -0.9952706263038429
  1.8003280593915159
  0.3671201374214753
  1.876844639736422
  4.397047895093343
  0.8382507817721072
 14.217599614817727
  ⋮
  0.028733010869426O7
 -0.517666243302685
  3.948539418170391
 -1.075697503087067
  1.0811088877801183
  1.0045457607416504
  1.0333681049401044
  0.5590361307058382
  0.80602284057423 4
 -7.672989813051116
  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
 -1.0654338709854956
 -0.8029208442815684
  0.7113973991349912
  1.151809485186219
 -0.9952706263038429
  1.8003280593915159
```

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
 808
 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]: 93×3 Matrix{Any}:
   1  618  693
   4  868  700
   6  922  516
   9  773  711
  11  753  118
  13  786  864
  15  686  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]: 93×3 Matrix{Int64}:
     1  618  693
     4  868  700
     6  922  516
     9  773  711
    11  753  118
    13  786  864
    15  686  400
    18  972  751
    19  672   15
    24  912  265
    28  793   10
    31  808  119
    32  655  277
     ⋮
   220  951  473
   222  710  684
   227  781  178
   228  697  589
   230  800  505
   232  908  640
   236  952  134
   237  932  202
   239  986  968
   241  671   30
   245  764  405
   249  676  792
```

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

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

```
[129]: 250-element Vector{Float64}:
 14.913752042997094
 20.750421682462264
  7.51132478328557
 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
  7.51132478328557
 15.14661678395542
  9.517352573063581
  6.7394361781977
 18.670297265978387
  3.0951575081084295
 15.50548290121917
 18.746199614855275
 18.777113729218343
  1.5556349186104097
 19.834817871611527
     ⋮
 22.30291460773681
 22.724876237286675
 20.989997617913158
  6.048140210014975
 16.810116001979285
 16.29662541755194
  8.098147936411138
 20.11417410683322
 19.504358487271507
 19.198437436416537
 17.92819009270038
 12.672016414130782
```

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

(здесь написано для случая, когда максимальное значение влючено)

```
[120]: 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
```

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]:
```
250×3 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  284
 30   30  343
 31   31  137
 33   33  147
 57   57  985
 71   71  434
  ⋮
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  673
```

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

[105]:
```
250×3 Matrix{Int64}:
  6    6  541
  8    8  457
```

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 выписать top-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
   ⋮
 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

- $\sum_{i=10}^{100}(i^3+4i^2)$;
- $\sum_{i=1}^{M}(\frac{2^i}{i}+\frac{3^i}{i^2}), M=25$;
- $1+\frac{2}{3}+(\frac{2}{3}\frac{4}{5})+\ldots+(\frac{2}{3}\frac{4}{5}\cdots\frac{38}{39})=1+\sum_{i=1}^{19}\prod_{j=1}^{i}\frac{2j}{2j+1}$

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

```
[130]: 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
```

```
[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
               5.666666666666666
```

Рис. 3.47: Вычисление выражений (1)

```
          299.8888888888889
          692.89
         1650.206611570248
         4031.895833333335
        10064.01775147929
        25573.188775510203
        65957.45333333334
       172247.25390625
       454561.89273356396
            1.2103058055555555e6
            3.247154955678 67e6
            8.7693898025e6
            2.3819486156462584e7
            6.502755020041322e7
            1.7832914331001893e8
            4.910281070572917e8
            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
        0.2995383701266054
        0.2837731927515209
        0.27026018357287707
        0.25850974080883893
        0.24816935117648536
        0.2389778937255044
        0.23073727670048702
        0.2232941387424068
        0.2165276496896066
        0.21034114541276507
        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.