

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Компьютерный практикум

по математическому моделированию

Студент: Ли Тимофей Александрович

Группа: НФИбд-01-18

МОСКВА

2021 г.

Постановка задачи

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

Выполнение работы

Сначала выполнил все примеры к лабораторной работе №2:

Для начала ознакомился с кортежами.

[1]: <code>()</code>	[6]: <code>length(x2)</code>
[1]: <code>()</code>	[6]: <code>3</code>
[2]: <code>fav=("python","julia","r") fav</code>	[7]: <code>x2[1], x2[2]</code>
[2]: <code>("python", "julia", "r")</code>	[7]: <code>(1, 2.0)</code>
[3]: <code>x1=(1,2,3) x1</code>	[8]: <code>c=x1[2]+x1[3]</code>
[3]: <code>(1, 2, 3)</code>	[8]: <code>5</code>
[4]: <code>x2=(1,2.0,"tmp")</code>	[9]: <code>x3.a, x3.b, x3[2]</code>
[4]: <code>(1, 2.0, "tmp")</code>	[9]: <code>(2, 3, 3)</code>
[5]: <code>x3=(a=2,b=1+2)</code>	[10]: <code>in("tmp", x2), 0 in x2</code>
[5]: <code>(a = 2, b = 3)</code>	[10]: <code>(true, false)</code>

Создал пустой кортеж, затем кортежи с элементами разных типов. Создал кортеж с именованными элементами. Далее провел простые операции над кортежами – нашел длину, обратился к элементам через индексы и имена, проверил вхождение элементов в кортеж.

Далее ознакомился со словарями.

```
[11]: phonebook = Dict{"Иванов И.И." => ("867-5309", "333-5544"), "Бухгалтерия" => "555-2368"}
[11]: Dict{String, Any} with 2 entries:
      "Бухгалтерия" => "555-2368"
      "Иванов И.И." => ("867-5309", "333-5544")
[12]: keys(phonebook)
[12]: KeySet for a Dict{String, Any} with 2 entries. Keys:
      "Бухгалтерия"
      "Иванов И.И."
[13]: values(phonebook)
[13]: ValueIterator for a Dict{String, Any} with 2 entries. Values:
      "555-2368"
      ("867-5309", "333-5544")
[14]: pairs(phonebook)
[14]: Dict{String, Any} with 2 entries:
      "Бухгалтерия" => "555-2368"
      "Иванов И.И." => ("867-5309", "333-5544")
```

```

[15]: haskey(phonebook, "Иванов И.И.")

[15]: true

[16]: phonebook["Сидоров П.С."] = "555-3344"

[16]: "555-3344"

[17]: pop!(phonebook, "Иванов И.И.")

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

[18]: phonebook

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

[19]: a=Dict{"foo"=>0.0, "bar"=>42.0}
      b=Dict{"baz"=>17, "bar"=>13.0}
      merge(a,b), merge(b,a)

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

```

Создал словарь, вызвал ключи, значения, пары ключей и значений. Проверил вхождение ключа в словарь, добавил элемент в словарь, удалил элемент словаря, затем создал два словаря и объединил их.

Затем ознакомился с множествами.

[20]: A=Set([1,3,4,5])	[23]: s3=Set([1,2,2,3,1,2,3,2,1]) s4=Set([2,3,1]) issetequal(s3,s4)	[27]: issubset(S1,s4)
[20]: Set{Int64} with 4 elements: 5 4 3 1	[23]: true	[27]: true
[21]: B=Set("abracadabra")	[24]: c=union(S1,S2)	[28]: push!(s4,99)
[21]: Set{Char} with 5 elements: 'a' 'd' 'r' 'k' 'b'	[24]: Set{Int64} with 4 elements: 4 2 3 1	[28]: Set{Int64} with 4 elements: 2 99 3 1
[22]: S1=Set([1,2]) S2=Set([3,4]) issetequal(S1,S2)	[25]: d=intersect(S1,s3)	[29]: pop!(s4)
[22]: false	[25]: Set{Int64} with 2 elements: 2 1	[29]: 2
	[26]: e=setdiff(s3,S1)	
	[26]: Set{Int64} with 1 element: 3	

Создал множества из чисел, символов. Проверил множества на эквивалентность, вычислил объединения, пересечения и разности множеств. Проверил вхождение множеств друг в друга, добавил и удалил элемент из множества.

Далее ознакомился с массивами.

```

[31]: ea1=[]
[31]: Any[]
[32]: ea2=(Integer)[]
[32]: Integer[]
[33]: ea3=(Float64)[]
[33]: Float64[]
[34]: a=[1,2,3]
[34]: 3-element Vector{Int64}:
 1
 2
 3
[35]: b=[1 2 3]
[35]: 1x3 Matrix{Int64}:
 1 2 3

[36]: A=[[1,2,3] [4,5,6] [7,8,9]]
[36]: 3x3 Matrix{Int64}:
 1 4 7
 2 5 8
 3 6 9
[37]: B=[[1 2 3]; [4 5 6]; [7 8 9]]
[37]: 3x3 Matrix{Int64}:
 1 2 3
 4 5 6
 7 8 9
[38]: c=rand(1,8)
[38]: 1x8 Matrix{Float64}:
 0.385775 0.840777 0.814119 0.915637 ... 0.121126 0.585728 0.0588356
[39]: C=rand(2,3)
[39]: 2x3 Matrix{Float64}:
 0.143131 0.892835 0.0349428
 0.101308 0.258833 0.23488

```

Создал пустой массив любого типа, пустой целочисленный массив и пустой массив действительных чисел. Создал целочисленные вектор и строку, научился создавать матрицы, а также делать массивы разных размерностей из случайных чисел.

```

[40]: D=rand(4,3,2)
[40]: 4x3x2 Array{Float64, 3}:
[:, :, 1] =
 0.167084 0.729376 0.424649
 0.546267 0.519413 0.484973
 0.571437 0.787869 0.622602
 0.841251 0.451533 0.331373

[:, :, 2] =
 0.589348 0.795452 0.415582
 0.112638 0.162374 0.290339
 0.555702 0.292073 0.0871626
 0.836485 0.283858 0.358148
[41]: roots=[sqrt(i) for i in 1:10]
[41]: 10-element Vector{Float64}:
 1.0
 1.4142135623730951
 1.7320508075688772
 2.0
 2.23606797749979
 2.449489742783178
 2.6457513110645907
 2.8284271247461903
 3.0
 3.1622776601683795

[42]: ar1=[3*i^2 for i in 1:2:9]
[42]: 5-element Vector{Int64}:
 3
 27
 75
 147
 243
[43]: ar2=[i^2 for i in 1:10 if (i%5!=0 && i%4!=0)]
[43]: 4-element Vector{Int64}:
 1
 9
 49
 81
[44]: ones(5)
[44]: 5-element Vector{Float64}:
 1.0
 1.0
 1.0
 1.0
 1.0

[45]: ones(2,3)
[45]: 2x3 Matrix{Float64}:
 1.0 1.0 1.0
 1.0 1.0 1.0
[46]: zeros(4)
[46]: 4-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
[47]: fill(3.5,(3,2))
[47]: 3x2 Matrix{Float64}:
 3.5 3.5
 3.5 3.5
 3.5 3.5
[48]: repeat([1 2],3,3)
[48]: 3x6 Matrix{Int64}:
 1 2 1 2 1 2
 1 2 1 2 1 2
 1 2 1 2 1 2

```

Научился создавать массивы с определенными элементами с помощью цикла for, освоил функции ones, zeros, fill и repeat.

<pre>[49]: a = collect(1:12) b = reshape(a,(2,6)) [49]: 2x6 Matrix{Int64}: 1 3 5 7 9 11 2 4 6 8 10 12 [50]: b'</pre>	<pre>[52]: ar=rand(10:20, 10, 5) [52]: 10x5 Matrix{Int64}: 12 10 19 16 12 16 18 19 12 11 17 11 19 19 16 17 20 17 16 19 14 10 10 20 15 20 11 20 15 20 14 20 10 16 17 14 12 12 10 16 10 13 19 13 14 12 11 18 11 17 [53]: ar[:,2]</pre>	<pre>[54]: ar[:,[2,5]] [54]: 10x2 Matrix{Int64}: 10 12 18 11 11 16 20 19 10 15 11 20 20 17 12 16 13 14 11 17 [55]: ar[:,2:4]</pre>
<pre>[51]: c=transpose(b) [51]: 6x2 transpose(::Matrix{Int64}) with eltype Int64: 1 2 3 4 5 6 7 8 9 10 11 12</pre>	<pre>[53]: 10-element Vector{Int64}: 10 18 11 20 10 11 20 12 13 11</pre>	<pre>[55]: 10x3 Matrix{Int64}: 10 19 16 18 19 12 11 19 19 20 17 16 10 10 20 11 20 15 20 10 16 12 12 10 13 19 13 11 18 11</pre>

Ознакомился с функциями collect и reshape, научился транспонировать матрицы, научился делать срезы.

<pre>[56]: ar[[2,4,6],[1,5]] [56]: 3x2 Matrix{Int64}: 16 11 17 19 20 20 [57]: ar[1,3:end] [57]: 3-element Vector{Int64}: 19 16 12 [58]: sort(ar, dims=1) [58]: 10x5 Matrix{Int64}: 10 10 10 10 11 12 10 10 11 12 12 11 12 12 14 14 11 17 13 15 14 11 18 15 16 14 12 19 16 16 16 13 19 16 17 17 18 19 16 17 17 20 19 19 19 20 20 20 20 20</pre>	<pre>[59]: sort(ar,dims=2) [59]: 10x5 Matrix{Int64}: 10 12 12 16 19 11 12 16 18 19 11 16 17 19 19 16 17 17 19 20 10 10 14 15 20 11 15 20 20 20 10 14 16 17 20 10 12 12 14 16 10 13 13 14 19 11 11 12 17 18 [60]: ar .> 14 [60]: 10x5 BitMatrix: 0 0 1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0 1 1 1 0 1 0 1 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 1</pre>	<pre>[61]: findall(ar .> 14) [61]: 27-element Vector{CartesianIndex{2}}: CartesianIndex(2, 1) CartesianIndex(3, 1) CartesianIndex(4, 1) CartesianIndex(6, 1) CartesianIndex(2, 2) CartesianIndex(4, 2) CartesianIndex(7, 2) CartesianIndex(1, 3) CartesianIndex(2, 3) CartesianIndex(3, 3) CartesianIndex(4, 3) CartesianIndex(6, 3) CartesianIndex(9, 3) ⋮ CartesianIndex(3, 4) CartesianIndex(4, 4) CartesianIndex(5, 4) CartesianIndex(6, 4) CartesianIndex(7, 4) CartesianIndex(3, 5) CartesianIndex(4, 5) CartesianIndex(5, 5) CartesianIndex(6, 5) CartesianIndex(7, 5)</pre>
---	---	---

Научился сортировать матрицы по столбцам и строкам, нашел элементы матрицы, удовлетворяющие условию.

Далее выполнил поставленные задачи.

1. Нашел результат операции над множествами.

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

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

```
[63]: P=intersect(A,B)
      P=union(P,A)
      P=intersect(P,B)
      P=union(P,A)
      P=intersect(P,C)
      P=union(P,B)
      P=intersect(P,C)
```

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

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

```
[64]: t1=Set("laboratory")
      t2=Set([1.0, 2.3, 4,9])
      t3=Set([2,6,9])
```

```
[64]: Set{Int64} with 3 elements:
      6
      2
      9
```

```
[65]: issetequal(t2,t3)
```

```
[65]: false
```

```
[66]: union(t2,t3)
```

```
[66]: Set{Float64} with 6 elements:
      4.0
      6.0
      2.0
      2.3
      9.0
      1.0
```

```
[67]: union(t1,t3)
```

```
[67]: Set{Any} with 10 elements:
      6
      2
      'a'
      'l'
      'r'
      't'
      'o'
      'y'
      9
      'b'
```

```
[68]: pop!(t1)
```

```
[68]: 'a': ASCII/Unicode U+0061 (category Ll: Letter, lowercase)
```

```
[69]: t1
```

```
[69]: Set{Char} with 6 elements:
      'l'
      'r'
      't'
      'o'
      'y'
      'b'
```

```
[70]: push!(t2, 2)
```

```
[70]: Set{Float64} with 5 elements:
      4.0
      2.0
      2.3
      9.0
      1.0
```

```
[71]: push!(t3, 3.0)
```

```
[71]: Set{Int64} with 4 elements:
      6
      2
      9
      3
```

```
[72]: push!(t3, 4.2)
```

```
InexactError: Int64(4.2)
```

Как видно, все операции работают как нужно, единственное что – нельзя добавить в множество элемент другого типа, который не конвертируется (здесь – float64 с ненулевым значением после запятой не добавляется в множество int64).

3. 3.1, 3.2, 3.3 Для N=24 создал массивы 1:N, N:1, 1:N:1

```
[4]: N=24
      t5=collect(1:N)

[2]: t4=[i for i in N:-1:1]

[3]: append!(t5,t4[2:end])

[1]: 24-element Vector{Int64}:
      1
      2
      3
      4
      5
      6
      7
      8
      9
     10
     11
     12
     13
     14
     15
     16
     17
     18
     19
     20
     21
     22
     23
     24

[2]: 24-element Vector{Int64}:
      24
      23
      22
      21
      20
      19
      18
      17
      16
      15
      14
      13
      12
      11
      10
      9
      8
      7
      6
      5
      4
      3
      2
      1

[3]: 47-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
```

3.4 – 3.9 Создал массив $\text{tmp} = [4, 6, 3]$, далее создал массив, в котором первый элемент tmp повторяется 10 раз, затем массив, в котором все элементы tmp повторяются 10 раз. Далее к полученному массиву добавил первый элемент tmp и получил массив, куда $\text{tmp}[1]$ входит 11 раз, а $\text{tmp}[2]$ и $\text{tmp}[3]$ по 10 раз. Далее с помощью функций `fill` и `append` создал массив, в который элементы tmp входят 10, 20 и 30 раз соответственно. Далее создал массив элементов $2^{\text{tmp}[i]}$, после чего сделал такой же массив, но $2^{\text{tmp}[3]}$ в него входит четыре раза. Затем прошелся по полученному массиву и подсчитал количество цифр 6 в его элементах.

[illegible]

3.10-3.13 Создал массив элементов $e^x \cos(x)$ для x от 3 до 6 с шагом 0.1, затем посчитал среднее для этого массива. Далее создал вектор (x^i, y^j) для данных x, y, i, j , вектор $2^i/i$ при i от 1 до 25, а также вектор ("fn1", "fn2", ... "fn30").

```
[85]: y=[cos(x)*exp(x) for x in 3:0.1:6] [88]: summy=0
[85]: 31-element Vector{Float64}:
-19.884530844146987
-21.178753389342127
-24.490696732801293
-26.77318244299338
-28.969237768093574
-31.011186439374516
-32.819774760338504
-34.30336011037369
-35.35719361853035
-35.86283371230767
-35.68773248011913
-34.68504225166807
-32.693695428321746
:

[88]: for i in 1:length(y)
      summy+=y[i]
      end
      print("y mean = ", summy/length(y))
      y mean = 53.11374594642971

[89]: arr311=[0.1^i 0.2^(i-2)] for i in 3:3:36
[89]: 12-element Vector{Matrix{Float64}}:
[0.0010000000000000002 0.2]
[1.0000000000000004e-6 0.0016000000000000003]
[1.0000000000000005e-9 1.2800000000000005e-5]
[1.0000000000000006e-12 1.0240000000000006e-7]
[1.0000000000000009e-15 8.192000000000005e-10]
[1.000000000000001e-18 6.553600000000005e-12]
[1.0000000000000012e-21 5.242880000000005e-14]
[1.0000000000000014e-24 4.194304000000005e-16]
[1.0000000000000015e-27 3.3554432000000048e-18]
[1.0000000000000017e-30 2.684354560000004e-20]
[1.0000000000000018e-33 2.1474836480000035e-22]
[1.000000000000002e-36 1.717986918400003e-24]

[90]: M=25
arr312=[(2^i)/i for i in 1:M]
[90]: 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
:

[113]: N=30
arr313=["f"+string(i) for i in 1:N]
[113]: 30-element Vector{String}:
"f1"
"f12"
"f13"
"f14"
"f15"
"f16"
"f17"
"f18"
"f19"
"f110"
"f118"
"f112"
"f113"
:
```

3.14 Создал массивы x и y размером 250, содержащие случайные целые числа в диапазоне 0-999. Далее сформировал вектор вида $x_i + 2x_{i+1} - x_{i+2}$, вектор вида

$$\sin(y_i)/\cos(x_{i+1}), \text{ также вычислил } \sum_{i=1}^{n-1} \frac{e^{-x_{i+1}}}{x_i + 10}.$$

```
[162]: N=250
x314=rand(0:999, N)
y314=rand(0:999, N)

[5]: t3141=[y314[i+1]-x314[i] for i in 1:N-1]
[5]: 249-element Vector{Int64}:
-191
-63
-79
563
533
573
-483
58
-609
383
-686
-164
-773
:

[164]: t3142=[x314[i]+2*x314[i+1]-x314[i+2] for i in 1:N-2]
[164]: 248-element Vector{Int64}:
1126
1339
1465
2518
253
1039
1443
623
-92
1394
682
614
10
:

[165]: t3143=[sin(y314[i])/cos(x314[i+1]) for i in 1:N-1]
[165]: 249-element Vector{Float64}:
4.380285486474569
-0.6815204438976953
2.9100592596446186
-4.503192558366575
0.7230623961879205
-156.5782350868654
23.65045512229104
1.1145805765484145
0.9884868255616273
-1.3362478719992334
0.16586660815822404
-0.13095549863225464
0.9593392937879452
:

[166]: t3144=0
for i in 1:N-1
    t3144+=(exp(-x314[i+1])/(x314[i]+10))
end
t3144
[166]: 0.0006578070827282885
```

Далее вывел все элементы y, большие 600, вместе с их индексами, вывел все элементы x, соответствующие найденным элементам y, сформировал вектор $(|x_1 - \bar{x}|^{\frac{1}{2}}, |x_2 - \bar{x}|^{\frac{1}{2}}, \dots, |x_n - \bar{x}|^{\frac{1}{2}})$ (модуль Statistics понадобился для импорта функции mean, чтобы не считать среднее вручную). Далее посчитал, сколько элементов y меньше максимального не более, чем на 200, вывел количество четных и нечетных элементов x, а также количество элементов x, кратных 7.


```
[167]: for i in 1:N
        if y314[i]>600
            println(i, " ", y314[i])
        end
    end
```

```
5 698
7 847
8 822
10 954
13 801
16 603
17 717
18 706
20 992
23 620
26 870
```

```
[168]: for i in 1:N
        if y314[i]>600
            println(x314[i])
        end
    end
```

```
872
721
454
203
356
398
485
...
```

```
[169]: using Statistics
        t3147=[sqrt(abs(x314[i]-mean(x314))) for i in 1:N]
```

```
[169]: 250-element Vector{Float64}:
 16.48199017109281
 16.892128344291017
 13.503481032681906
 17.898156329633508
 19.166220284657065
 21.299201862980688
 14.708636918491122
 7.117302859932265
 17.850938350686217
 17.368246889078932
 12.200000000000000
```

```
[170]: t3148=0
        maxy314=0
        for i in 1:N
            if y314[i]>maxy314
                maxy314=y314[i]
            end
        end
        for i in 1:N
            if maxy314-y314[i]<=200
                t3148+=1
            end
        end
        t3148
```

```
[170]: 51
```

```
[171]: xch=0
        xnc=0
        for i in 1:N
            if x314[i]%2==0
                xch+=1
            else
                xnc+=1
            end
        end
        print(xch, " ", xnc)
```

```
124 126
```

```
[172]: t31410=0
        for i in 1:N
            if x314[i]%7==0
                t31410+=1
            end
        end
        print(t31410)
```

```
32
```

Затем, вывел вектор элементов x, отсортированных в порядке возрастания элементов y. Для этого отсортировал y, при этом запоминая перестановки индексов, и вывел x, соответствуя новому порядку индексов. Далее вывел 10 самых больших элементов x (отсортировал и вывел 10 последних), а также составил вектор неповторяющихся элементов x.

```
[176]: #y314=rand(0:999, 250)
        a=collect(1:250)
        check=0
        for i in 1:N
            for j in 1:N-i
                if y314[j]>y314[j+1]
                    check=y314[j]
                    y314[j]=y314[j+1]
                    y314[j+1]=check
                    check=a[j]
                    a[j]=a[j+1]
                    a[j+1]=check
                end
            end
        end
        t31411=[]
        for i in 1:N
            append!(t31411, x314[a[i]])
        end
        t31411
```

```
[176]: 250-element Vector{Any}:
 233
 790
 687
 825
 872
 51
 771
```

```
[179]: t31412=sort(x314)
        print(t31412[241:250])

[949, 952, 956, 964, 967, 968, 977, 981, 989, 998]
```

```
[183]: t31413=[t31412[1]]
        for i in 1:N
            if t31412[i]!=t31413[end]
                append!(t31413, t31412[i])
            end
        end
        t31413
```

```
[183]: 217-element Vector{Int64}:
 1
 2
 6
 9
 10
 14
 23
 24
 35
 39
 40
 43
 45
 ...
```

4. Создал массив squares, содержащий первые 100 квадратов натуральных чисел.
5. Подключил модуль Primes, создал массив из 168 первых простых чисел, вывел 89 простое число, а также 89-99 простые числа.

```
[184]: squares=[i^2 for i in 1:100]
[184]: 100-element Vector{Int64}:
 1
 4
 9
16
25
36
49
64
81
100
121
144
169
:
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]
461
[461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]
```

6. Вычислил выражения

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

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

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

```
[194]: t61=0
for i in 10:100
    t61+=i^3+4*i^2
end
print(t61)
```

26852735

```
[198]: t62=0
M=25
for i in 1:M
    t62+= 2^i/i + 3^i/i^2
end
print(t62)
```

2.1291704368143802e9

```
[199]: t63=1
check=1
for i in 1:19
    check*=2*i/(2*i+1)
    t63+=check
end
print(t63)
```

6.976346137897618

Выводы

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