

Компьютерный практикум по статистическому анализу данных

Лабораторная работа № 2. Julia. Структуры данных

Сунгурова Мариян

Содержание

1	Введение	4
2	Теоретическое введение	5
3	Выполнение лабораторной работы	6
4	Выводы	25
	Список литературы	26

Список иллюстраций

3.1	Примеры. Кортежи	7
3.2	Примеры. Кортежи	7
3.3	Примеры. Словари	8
3.4	Примеры. Словари	8
3.5	Примеры. Множества	9
3.6	Примеры. Множества	9
3.7	Примеры. Массивы	10
3.8	Примеры. Массивы	10
3.9	Примеры. Массивы	11
3.10	Примеры. Массивы	12
3.11	Примеры. Массивы	13
3.12	Примеры. Массивы	14
3.13	Примеры. Массивы	15
3.14	Задание 1	16
3.15	Задание 2	17
3.16	Задание 2	17
3.17	Задание 2	18
3.18	Задание 3	18
3.19	Задание 3	19
3.20	Задание 3	20
3.21	Задание 3	21
3.22	Задание 3	21
3.23	Задание 3	22
3.24	Задание 4	22
3.25	Задание 5	23
3.26	Задание 6	24

1 Введение

Цель работы

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

Задачи

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

2 Теоретическое введение

Julia — высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений.[1]. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков, однако имеет некоторые существенные отличия.

Для выполнения заданий была использована официальная документация Julia[2].

3 Выполнение лабораторной работы

Рассмотрим несколько структур данных, реализованных в Julia.

Несколько функций (методов), общих для всех структур данных:

- `isempty()` — проверяет, пуста ли структура данных;
- `length()` — возвращает длину структуры данных;
- `in()` — проверяет принадлежность элемента к структуре;
- `unique()` — возвращает коллекцию уникальных элементов структуры,
- `reduce()` — свёртывает структуру данных в соответствии с заданным бинарным оператором;
- `maximum()` (или `minimum()`) — возвращает наибольший (или наименьший) результат вызова функции для каждого элемента структуры данных.

Выполним примеры из лабораторной работы для действий над кортежами (рис. fig. 3.1 - fig. 3.2)

```
Кортежи

Примеры кортежей

()
[92] ✓ 0.0s
... ()

favoritelang = ("Python", "Julia", "R")
[93] ✓ 0.0s
... ("Python", "Julia", "R")

x1 = (1, 2, 3)
x2 = (1, 2.0, "tmp")
x3 = (a=2, b=1+2)
[94] ✓ 0.0s
... (a = 2, b = 3)

Операции над кортежами

length(x2)
[95] ✓ 0.0s
... 3

x2[1], x2[2], x2[3]
[96] ✓ 0.0s
... (1, 2.0, "tmp")
```

Рис. 3.1: Примеры. Кортежи

```
Операции над кортежами

length(x2)
[97] ✓ 0.0s
3

x2[1], x2[2], x2[3]
[98] ✓ 0.0s
(1, 2.0, "tmp")

x = x1[1] + x1[3]
[99] ✓ 0.0s
4

x3.a, x3.b, x3[2]
[100] ✓ 0.0s
(2, 3, 3)

in("tmp", x2), 0 in x2
[101] ✓ 0.0s
(true, false)
```

Рис. 3.2: Примеры. Кортежи

Также со словарями(рис. fig. 3.3 - fig. 3.4)

Словари

Примеры словарей

```

phonebook = Dict("Иванов И.И." -> ("867-5309", "333-5544"), "Бухгалтерия" -> "555-2368")
[180] ✓ 0.0s
... Dict{String, Any} with 2 entries:
  "Бухгалтерия" -> "555-2368"
  "Иванов И.И." -> ("867-5309", "333-5544")

keys(phonebook)
[181] ✓ 0.0s
... KeySet for a Dict{String, Any} with 2 entries. Keys:
  "Бухгалтерия"
  "Иванов И.И."

values(phonebook)
[182] ✓ 0.0s
... ValueIterator for a Dict{String, Any} with 2 entries. Values:
  "555-2368"
  ("867-5309", "333-5544")

pairs(phonebook)
[183] ✓ 0.0s
... Dict{String, Any} with 2 entries:
  "Бухгалтерия" -> "555-2368"
  "Иванов И.И." -> ("867-5309", "333-5544")

```

Рис. 3.3: Примеры. Словари

```

haskey(phonebook, "Иванов И.И.")
[184] ✓ 0.0s
... true

phonebook["Сидоров"] = "555-3344"
[185] ✓ 0.0s
... "555-3344"

pop!(phonebook, "Иванов И.И.")
[186] ✓ 0.0s
... ("867-5309", "333-5544")

a = Dict{"foo" => 0.0, "bar" => 42.0};
b = Dict{"baz" => 17, "bar" => 13.0};
merge(a, b), merge(b, a)
[187] ✓ 0.0s
... (Dict{String, Real}{"bar" => 13.0, "baz" => 17, "foo" => 0.0}, Dict{String, Real}{"bar" => 42.0, "baz" => 17, "foo" => 0.0})

```

Рис. 3.4: Примеры. Словари

Рассмотрим также примеры операций над множествами(рис. fig. 3.5 - fig. 3.6)

Множества

```

A = Set([1, 2, 4, 5])
B = Set("abcd")

S1 = Set([1, 2])
S2 = Set([3, 4])
isetequal(S1, S2)
[108] ✓ 0.0s
... false

S3 = Set([1, 2, 2, 3, 1, 2, 3, 2, 1])
S4 = Set([2, 3, 1])
isetequal(S3, S4)
[109] ✓ 0.0s
... true

C = union(S1, S2)
[110] ✓ 0.0s
... Set{Int64} with 4 elements:
    4
    2
    3
    1

D = intersect(S1, S3)
[111] ✓ 0.0s
... Set{Int64} with 2 elements:
    2
    1

```

Рис. 3.5: Примеры. Множества

```

D = intersect(S1, S3)
[111] ✓ 0.0s
... Set{Int64} with 2 elements:
    2
    1

E = setdiff(S3, S1)
[112] ✓ 0.0s
... Set{Int64} with 1 element:
    3

issubset(S1, S4)
[113] ✓ 0.0s
... true

push!(S4, 99)
[114] ✓ 0.0s
... Set{Int64} with 4 elements:
    2
    99
    3
    1

pop!(S4)
[115] ✓ 0.0s
... 2

```

Рис. 3.6: Примеры. Множества

И с массивами(рис. fig. 3.7 - fig. 3.11)

Массивы

```

emp_arr_1 = []
emp_arr_2 = (Int64)[]
emp_arr_3 = (Float64)[]
[116] ✓ 0.0s
... Float64[]

a = [1, 2, 3]
b = [1 2 3]
[117] ✓ 0.0s
... 1x3 Matrix{Int64}:
 1  2  3

A = [[1, 2, 3] [4, 5, 6] [7, 8, 9]]
B = [[1 2 3]; [4 5 6]; [7 8 9]]
[118] ✓ 0.0s
... 3x3 Matrix{Int64}:
 1  2  3
 4  5  6
 7  8  9

c = rand(1, 8)
[119] ✓ 0.0s
... 1x8 Matrix{Float64}:
 0.0118475  0.144387  0.0256888  0.457047 ... 0.19639  0.998878  0.471298

```

Рис. 3.7: Примеры. Массивы

```

c = rand(1, 8)
[119] ✓ 0.0s
... 1x8 Matrix{Float64}:
 0.0118475  0.144387  0.0256888  0.457047 ... 0.19639  0.998878  0.471298

C = rand(2, 3)
[120] ✓ 0.0s
... 2x3 Matrix{Float64}:
 0.0462157  0.172209  0.180259
 0.41104   0.386364  0.296203

D = rand(4, 3, 2)
[121] ✓ 0.0s
... 4x3x2 Array{Float64, 3}:
[:, :, 1] =
 0.784008  0.339395  0.89673
 0.0690803 0.436972  0.199708
 0.388326  0.298149  0.747105
 0.409187  0.152635  0.638116

[:, :, 2] =
 0.0863516 0.27734  0.723226
 0.538446  0.704094  0.896067
 0.107992  0.100161  0.131123
 0.504906  0.889275  0.891341

roots = [sqrt(i) for i in 1:10]
[122] ✓ 0.0s
... 10-element Vector{Float64}:
 1.0
 1.4142135623730951
 1.7320508075688772
 2.0
 2.23606797749979
 2.449489742783178

```

Рис. 3.8: Примеры. Массивы

```
ar_1 = [3*i^2 for i in 1:2:9]
[123] ✓ 0.0s
... 5-element Vector{Int64}:
      3
     27
     75
    147
    243

ar_2=[i^2 for i=1:10 if (i^2%5!=0 && i^2%4!=0)]
[124] ✓ 0.1s
... 4-element Vector{Int64}:
      1
      9
     49
     81

ones(5)
[125] ✓ 0.0s
... 5-element Vector{Float64}:
 1.0
 1.0
 1.0
 1.0
 1.0

ones(2, 3)
[126] ✓ 0.0s
... 2x3 Matrix{Float64}:
 1.0  1.0  1.0
 1.0  1.0  1.0
```

Рис. 3.9: Примеры. Массивы

```
▷ fill(3.5, (3,2))
[128] ✓ 0.0s
... 3x2 Matrix{Float64}:
      3.5  3.5
      3.5  3.5
      3.5  3.5

repeat([1, 2], 3,3)
[129] ✓ 0.0s
... 6x3 Matrix{Int64}:
      1  1  1
      2  2  2
      1  1  1
      2  2  2
      1  1  1
      2  2  2

a = collect(1:12)
b = reshape(a, (2, 6))
[130] ✓ 0.0s
... 2x6 Matrix{Int64}:
      1  3  5  7  9 11
      2  4  6  8 10 12

b'
[131] ✓ 0.0s
... 6x2 adjoint(::Matrix{Int64}) with eltype Int64:
      1  2
      3  4
      5  6
      7  8
      9 10
     11 12
```

Рис. 3.10: Примеры. Массивы

```

c = transpose(b)
[132] ✓ 0.0s
... 6x2 transpose(::Matrix{Int64}) with eltype Int64:
 1  2
 3  4
 5  6
 7  8
 9 10
11 12

ar = rand(10:20, 10, 5)
[133] ✓ 0.0s
... 10x5 Matrix{Int64}:
17 15 18 17 18
19 16 13 16 14
12 14 17 17 18
17 14 10 14 18
20 10 17 18 17
12 17 16 20 11
12 10 19 13 16
16 13 10 11 11
15 19 20 19 20
11 19 15 19 20

ar[:, 2]
[134] ✓ 0.0s
... 10-element Vector{Int64}:
15
16
14
14
10
17
10
13
19
19
```

Рис. 3.11: Примеры. Массивы

```
ar[:, [2, 5]]
[135] ✓ 0.0s
... 10x2 Matrix{Int64}:
15 18
16 14
14 18
14 18
10 17
17 11
10 16
13 11
19 20
19 20

ar[[2, 4, 6], [1, 5]]
[136] ✓ 0.0s
... 3x2 Matrix{Int64}:
19 14
17 18
12 11

ar[1, 3:end]
[137] ✓ 0.0s
... 3-element Vector{Int64}:
18
17
18

sort(ar, dims=1)
[138] ✓ 0.0s
... 10x5 Matrix{Int64}:
11 10 10 11 11
12 10 10 13 11
12 13 13 14 14
12 14 15 16 16
```

Рис. 3.12: Примеры. Массивы


```
Задание 1

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

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

[142] ✓ 0.0s
... Set{Int64} with 7 elements:
0
4
7
2
9
8
1

union(intersect(A, B), intersect(A, C), intersect(B, C) )

[143] ✓ 0.0s
... Set{Int64} with 6 elements:
0
4
7
9
3
1
```

Рис. 3.14: Задание 1

2. Приведем свои примеры с выполнением операций над множествами элементов разных типов. (рис. fig. 3.16 - fig. 3.17)

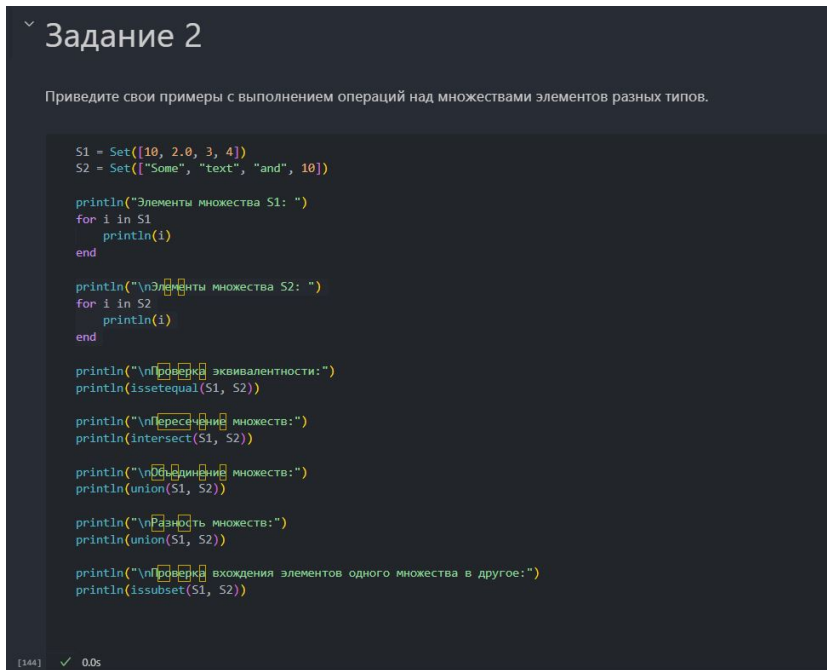


Рис. 3.15: Задание 2

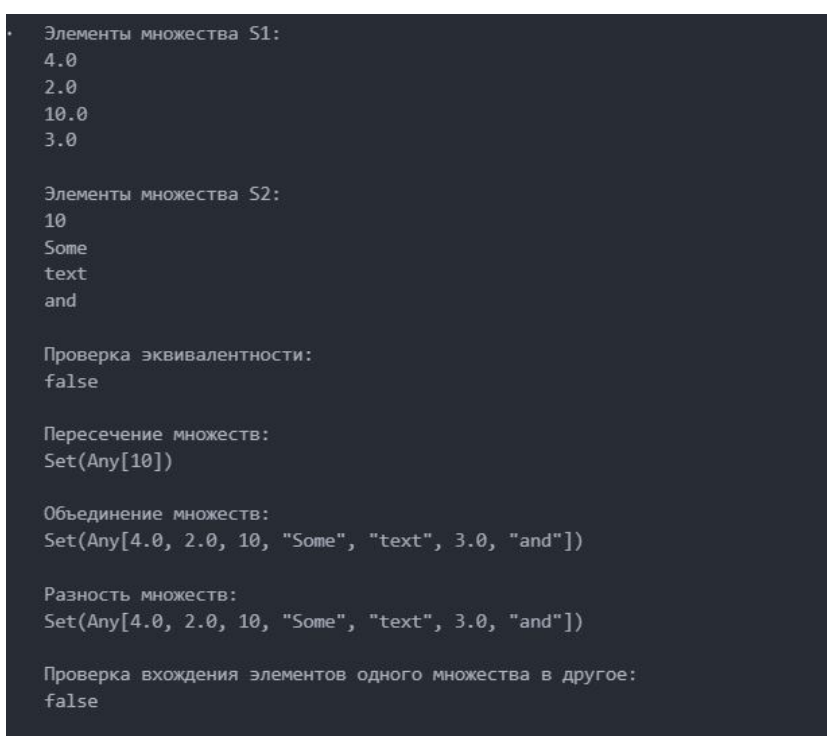


Рис. 3.16: Задание 2

```
println("\nдобавление элемента в множество S2:")
push!(S2, "end")

println("\nЭлементы множества S2: ")
for i in S2
    println(i)
end

println("\nудаление последнего элемента в множестве S2:")
pop!(S2)

println("\nЭлементы множества S2: ")
for i in S2
    println(i)
end
```

✓ 0.0s

добавление элемента в множество S2:

Элементы множества S2:

10
Some
text
and
end

удаление последнего элемента в множестве S2:

Элементы множества S2:

Some
text
and
end

Рис. 3.17: Задание 2

1. Создадим разными способами массивы и вектора. Для создания нужных массивов, используем генераторы и циклы(рис. fig. 3.18 - fig. 3.23)

Задание 3

Создайте разными способами:

- 3.1) массив $\{1, 2, 3, \dots, N-1, N\}$, N выберите больше 20;
- 3.2) массив $\{N, N-1, \dots, 2, 1\}$, N выберите больше 20;
- 3.3) массив $\{1, 2, 3, \dots, N-1, N, N-1, \dots, 2, 1\}$, N выберите больше 20;
- 3.4) массив с именем `tmp` вида $\{4, 6, 9\}$;
- 3.5) массив, в котором первый элемент массива `tmp` повторится 10 раз;
- 3.6) массив, в котором все элементы массива `tmp` повторятся 10 раз;
- 3.7) массив, в котором первый элемент массива `tmp` встречается 11 раз, второй элемент — 10 раз, третий элемент — 10 раз;
- 3.8) массив, в котором первый элемент массива `tmp` встречается 10 раз подряд, второй элемент — 20 раз подряд, третий элемент — 30 раз подряд;
- 3.9) массив из элементов вида $2 \cdot \text{tmp}[i]$, $i = 1, 2, 3$, где элемент $2 \cdot \text{tmp}[3]$ встречается 4 раза; посчитайте в полученном векторе, сколько раз встречается цифра 6, и выведите это значение на экран;

```
a1 = [1 for i in 1:27]
a2 = reverse([1 for i in 1:27])
a3 = vcat([1 for i in 1:27], reverse([1 for i in 1:27]))
tmp = [4, 6, 9]
tmp1 = fill(tmp[1], 10)
tmp2 = repeat(tmp, 10)
tmp3 = vcat(fill(tmp[1], 11), fill(tmp[2], 10), fill(tmp[3], 10))
tmp4 = vcat(fill(tmp[1], 10), fill(tmp[2], 20), fill(tmp[3], 30))
tmp5 = [2*i for i in vcat(fill...([1, 1, 4])...)]

c = 0
str_tmp5 = string(tmp5)
for i in str_tmp5
    if i=="6"
        c += 1
    end
end
println(c)
```

Рис. 3.18: Задание 3

```

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

function Y(x)
    exp(x)*cos(x)
end
[168] ✓ 0.2s

... Y (generic function with 1 method)

X = 3:0.1:6
res = [Y(x) for x in X ]
println(sum(res)/sizeof(res))
[170] ✓ 0.0s

... 6.639218243303714

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

is = 3:3:36
js = 1:3:34
x = 0.1
y = 0.2
v = [ x*is[k], y*js[k] for k in 1:length(is)]
[182] ✓ 0.6s

... 12-element Vector{Vector{Float64}}:
 [0.0018000000000000002, 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]

```

Рис. 3.19: Задание 3

```
вектор с элементами  $2^i i$ ,  $i = 1, 2, \dots, M$ ,  $M = 25$ ;  
  
M = 25  
is = 1:M  
v1 = [2^i/i for i in is]  
184] ✓ 0.0s  
... 25-element Vector{Float64}:  
2.0  
2.0  
2.6666666666666665  
4.0  
6.4  
10.666666666666666  
18.285714285714285  
32.0  
56.888888888888886  
102.4  
:  
7710.117647058823  
14563.555555555555  
27594.105263157893  
52428.8  
99864.38095238095  
190650.18181818182  
364722.0869565217  
699050.6666666666  
1.34217728e6  
  
вектор вида ("fn1", "fn2", ..., "fnN"),  $N = 30$ ;  
  
v2 = []  
for i=1:30  
    push!(v2, "fn"*string(i))  
end  
v2  
187] ✓ 0.4s  
... 30-element Vector{Any}:  
"fn1"  
"fn2"  
"fn3"  
"fn4"  
"fn5"  
"fn6"  
"fn7"  
"fn8"  
"fn9"  
"fn10"  
"fn11"  
"fn12"  
"fn13"  
"fn14"  
"fn15"  
"fn16"  
"fn17"  
"fn18"  
"fn19"  
"fn20"  
"fn21"  
"fn22"  
"fn23"  
"fn24"  
"fn25"  
"fn26"  
"fn27"  
"fn28"  
"fn29"  
"fn30"
```

Рис. 3.20: Задание 3

```

v2 = []
for i=1:30
    push!(v2, "fn"*string(i))
end
v2

[187] ✓ 0.4s

... 30-element Vector{Any}:
 "fn1"
 "fn2"
 "fn3"
 "fn4"
 "fn5"
 "fn6"
 "fn7"
 "fn8"
 "fn9"
 "fn10"
 ⋮
 "fn22"
 "fn23"
 "fn24"
 "fn25"
 "fn26"
 "fn27"
 "fn28"
 "fn29"
 "fn30"

```

Рис. 3.21: Задание 3

```

314
n = 250
x = [rand(0:999) for i=1:n]
y = [rand(0:999) for i=1:n]
import Plots
Plots.add!("Statistics")
using Statistics

315 ✓ 346

Resolving package versions...
Updating C:\Users\IP\julia\environments\v1.10\Project.toml
C:\Users\IP\julia\environments\v1.10\Manifest.toml
No Changes to 'C:\Users\IP\julia\environments\v1.10\Manifest.toml'

316
arr1 = [x[i+1] - x[i] for i=1:n-1]
println("3.14.3 ", arr1)
arr2 = [x[i] + 2*x[i+1] - x[i+2] for i=1:n-2]
println("3.14.2 ", arr2)
arr3 = [sin(y[i])/cos(x[i+1]) for i=1:n-1]
println("3.14.1 ", arr3)
arr4 = sum([x[i]*y[i+1])/(x[i] + 10) for i=1:n-1])
println("3.14.4 ", arr4)

317 ✓ 10s

3.14.1 [0, 181, -532, -699, -383, -111, -59, 362, 668, 12, -66, 363, 44, -13, 115, -23, 385, 379, -20, 91, 223, 656, 366, -207, -272, -121, -131, -629, 386, 184, 188, -521, 566, 44, -453, 229, -125, 236, -391, 639, -279, -13.14.2 [1409, 1893, 2228, 3686, 412, 1179, 882, -121, 758, 2389, 980, 1258, 3538, 2373, 532, 666, 1480, 1851, 2217, 833, -533, 1173, 999, 583, 1818, 1781, 2384, 959, 507, 236, 1893, 1151, -134, 1288, 18, 964, 1484, 1335, 31.14.3 [4.311802967081979, 1.4896398121118772, -1.184309812335362, -1.5086156489528664, 8.4911976011811679, 0.584528779115562, 8.7118481233417881, 18.43658287992175, -32.58886188928881, -1.7222173668189565, -1.143727646793.14.4) 0.801411182804294862

```

Рис. 3.22: Задание 3

массива с 89-го до 99-го элемента включительно, содержащий наименьшие простые числа.(рис. fig. 3.25)

```
Задание 5

Pkg.add("Primes")
import Primes as pm

[215] ✓ 17.6s

... Resolving package versions...
Installed IntegerMathUtils - v0.1.2
Installed Primes - v0.5.6
Updating `C:\Users\HP\.julia\environments\v1.10\Project.toml`
[27ebfcd6] + Primes v0.5.6
Updating `C:\Users\HP\.julia\environments\v1.10\Manifest.toml`
[18e54dd8] + IntegerMathUtils v0.1.2
[27ebfcd6] + Primes v0.5.6
Precompiling project...
✓ IntegerMathUtils
✓ Primes
2 dependencies successfully precompiled in 11 seconds. 315 already precompiled.

myprimes = [pm.prime(i) for i=1:168]
prime_89 = myprimes[89]
prime_89_to_99 = myprimes[89:99]

[217] ✓ 0.0s

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

Рис. 3.25: Задание 5

6. Вычислим выражения(рис. fig. 3.26)

Задание 6

```
# 6.1
res61 = sum([i^3 + 4*i^2 for i=10:100])
println("6.1) ", res61)

# 6.2
res62 = sum([ ((2^i)/(i) + (3^i)/(i^2) ) for i=1:25])
println("6.2) ", res62)

# 6.3
res63 = 1
tmp = 1
for i=1:2:38
    tmp*= i/(i+1)
    res63 = res63 + tmp
end
println("6.3) ", res63)
```

219] ✓ 0.1s

... 6.1) 26852735
6.2) 2.1291704368143802e9
6.3) 5.01482750478317

Рис. 3.26: Задание 6

4 Выводы

В результате выполнения данной лабораторной работы были изучены структуры данных, реализованных в Julia: словарь, массив, кортеж множество, также юыли получены практические навыки применения этих структур и операций над ними решения задач.

Список литературы

1. JuliaLang [Электронный ресурс]. 2024 JuliaLang.org contributors. URL: <https://julialang.org/> (дата обращения: 11.10.2024).
2. Julia 1.11 Documentation [Электронный ресурс]. 2024 JuliaLang.org contributors. URL: <https://docs.julialang.org/en/v1/> (дата обращения: 11.10.2024).