

Лабораторная работа № 2

Структуры данных

Оразгелдиев Язгелди

2025-09-27

Содержание I

Информация

Информация

Докладчик

► Оразгелдиев Язгелди

Докладчик

- ▶ Оразгелдиев Язгелди
- ▶ 1032225075@pfur.ru

Докладчик

- ▶ Оразгелдиев Язгелди
- ▶ 1032225075@pfur.ru
- ▶ <https://github.com/YazgeldiOrazgeldiyev>

Цели

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

Задачи

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

Содержание исследования

```
[1]: # пустой кортеж:
    ()

[1]: ()

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

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

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

[3]: (1, 2, 3)

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

[4]: (1, 2.0, "top")

[5]: # анонимный кортеж:
    x3 = (a-2, b-1+2)

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

Рисунок 1: Примеры использования кортежей

Содержание исследования

```
[6]: # Длина кортежа x2:
length(x2)

[6]: 3

[7]: # обратиться к элементам кортежа x2:
x2[1], x2[2], x2[-1]

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

[8]: # произвести какие-либо операции (сложения)
# с первым и вторым элементами кортежа x2:
c = x2[1] + x2[2]

[8]: 5

[9]: # обращение к элементу именованного кортежа x3:
x3.a, x3.b, x3[1]

[9]: (2, 3, 1)

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

[10]: (True, False)
```

Рисунок 2: Примеры использования кортежей

Содержание исследования

```

[11]: # словарь словарь с именем phonebook:
phonebook = Dict{"Иванов И.И." => ("867-5389", "333-5544"), "бухгалтерия" => "555-2368"}

[11]: Dict{String, Any} with 2 entries:
      "бухгалтерия" => "555-2368"
      "Иванов И.И." => ("867-5389", "333-5544")

[12]: # Вывести ключи словаря:
keys(phonebook)

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

[13]: # Вывести значения элементов словаря:
values(phonebook)

[13]: valueiterator for a Dict{String, Any} with 2 entries. values:
      "555-2368"
      ("867-5389", "333-5544")

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

[14]: Dict{String, Any} with 2 entries:
      "бухгалтерия" => "555-2368"
      "Иванов И.И." => ("867-5389", "333-5544")

[15]: # проверить наличие ключа в словаре:
haskey(phonebook, "Иванов И.И.")

[15]: true

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

[16]: "555-3344"

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

[18]: ("867-5389", "333-5544")

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

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

```

Рисунок 3: Примеры использования словарей

Содержание исследования

```
[20]: # создать множество из четырех натуральных чисел:  
A = Set([1, 3, 4, 5])  
  
[20]: set(int64) with 4 elements:  
5  
4  
3  
1  
  
[21]: # создать множество из 11 символьных значений:  
B = Set("abracadabra")  
  
[21]: set(char) with 5 elements:  
'a'  
'd'  
'r'  
'k'  
'b'  
  
[22]: # проверка эквивалентности двух множеств:  
S1 = Set([1,2]);  
S2 = Set([3,4]);  
isetequal(S1,S2)  
S3 = Set([1,2,3,4,5,6,7,8,9,10]);  
S4 = Set([2,3,11]);  
isetequal(S3,S4)  
  
[22]: true  
  
[23]: # объединение множеств:  
C = union(S1,S2)  
  
[24]: # пересечение множеств:  
D = intersect(S1,S3)  
  
[24]: set(int64) with 2 elements:  
2  
1  
  
[25]: # разность множеств:  
E = setdiff(S3,S1)  
  
[25]: set(int64) with 1 element:  
3  
  
[26]: # проверка вхождения элементов одного множества в другое:  
issubset(S1,S4)  
  
[26]: true  
  
[27]: # добавление элемента в множество:  
push!(S4, 10)
```

Рисунок 4: Примеры использования множеств

Содержание исследования

```

empty_array_1 = []

Any[]

# создание пустого массива с конкретным типом:
empty_array_2 = (int64)[]
empty_array_3 = (float64)[]

float64[]

# вектор-столбец:
a = [1, 2, 3]

3-element Vector{Int64}:
 1
 2
 3

# вектор-строка:
b = [1 2]

1x3 Matrix{Int64}:
 1 2 3

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

3x3 Matrix{Int64}:
 1 2 3
 4 5 6
 7 8 9

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

1x8 Matrix{Float64}:
0.37157 0.868185 0.700679 0.952377 ... 0.383572 0.442995 0.450085

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

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

4x3x2 Array{Float64, 3}:
[:, 1, 1] =
 0.101011 0.862076 0.188035
 0.691591 0.886606 0.505946
 0.307612 0.031682 0.846561
 0.267509 0.399429 0.996038

```

Рисунок 5: Примеры использования массивов

Содержание исследования

```
# произвольный массив:
D = rand(4, 2)

4x2 Array{Float64, 2}:
[:, 1, 1] =
 0.101011  0.862076  0.188035
 0.691591  0.036664  0.905946
 0.307612  0.0336582  0.846561
 0.287509  0.339469  0.996038

[:, 1, 2] =
 0.84059  0.697702  0.0540585
 0.673059  0.447209  0.231141
 0.207027  0.224450  0.735207
 0.809726  0.465222  0.481309

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

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

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

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

# массив квадратных элементов, если квадрат не делится на 5 или 4:
ar_2 = [i^2 for i in 1:10 if (i^2%5!=0 || i^2%4!=0)]

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

Рисунок 6: Примеры использования массивов

Содержание исследования

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

5-element Vector{Float64}:
 1.0
 1.0
 1.0
 1.0
 1.0

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

2x3 Matrix{Float64}:
 1.0  1.0  1.0
 1.0  1.0  1.0

# одномерный массив из 4 нулей:
zeros(4)

4-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0

# заполнить массив 3x2 значениями 3.5
fill(3.5, (3,2))

3x2 Matrix{Float64}:
 3.5  3.5
 3.5  3.5
 3.5  3.5

# заполнение массива посредством функции repeat():
repeat([1,2],3,3)
repeat([1: 2],1,5)

3x5 Matrix{Int64}:
 1  2  1  2  1  2
 1  2  1  2  1  2
 1  2  1  2  1  2
```

Рисунок 7: Примеры использования массивов

Содержание исследования

```
# Выполнение самостоятельной работы:  
  
# Задание #1.  
  
A = Set([0, 3, 4, 9])  
B = Set([1, 3, 4, 7])  
C = Set([0, 1, 2, 4, 7, 8, 9])  
  
P = union(intersect(A, B), intersect(A, C), intersect(B, C)) # union(intersect(A, B), intersect(A, C)) - intersect(A, B)  
println(P)  
  
Set([0, 4, 7, 9, 3, 1])
```

Рисунок 8: Задание №1. Работа с множествами

Содержание исследования

```
Set1 = Set([1,2,3], "Hello", "Geeks"])
println(Set1)

Set(Any[2, "Hello", "Geeks", 3, 1])

println("Elements of set:")
for i in Set1
    println(i)
end

Elements of set:
2
Hello
Geeks
3
1

println(in("Hello", Set1))

true

Set1 = push!(Set1, "welcome")
println("\nSet after adding one element:\n", Set1)

Set after adding one element:
Set(Any["welcome", 2, "Hello", "Geeks", 3, 1])

for i in 1:5
    push!(Set1,i)
end
println("\nSet after adding range of elements:\n", Set1)

Set after adding range of elements:
Set(Any[5, 4, "welcome", 2, "Hello", "Geeks", 3, 1])
```

Рисунок 9: Задание №2. Примеры операций над множествами элементов разных типов

Содержание исследования

```

3. Создайте разные массивы

int a[5] = { 1, 2, 3, 4, 5 };
printf("%i\n", a[0]);
int b[10];
for (int i = 0; i < 10; i++)
    printf("%i\n", b[i]);
printf("%i\n", a[0]);
int c[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
printf("%i\n", c[0]);
printf("%i\n", c[9]);
printf("%i\n", c[5]);
printf("%i\n", c[10]);
printf("%i\n", c[11]);
printf("%i\n", c[12]);
printf("%i\n", c[13]);
printf("%i\n", c[14]);
printf("%i\n", c[15]);
printf("%i\n", c[16]);
printf("%i\n", c[17]);
printf("%i\n", c[18]);
printf("%i\n", c[19]);
printf("%i\n", c[20]);
printf("%i\n", c[21]);
printf("%i\n", c[22]);
printf("%i\n", c[23]);
printf("%i\n", c[24]);
printf("%i\n", c[25]);
printf("%i\n", c[26]);
printf("%i\n", c[27]);
printf("%i\n", c[28]);
printf("%i\n", c[29]);
printf("%i\n", c[30]);
printf("%i\n", c[31]);
printf("%i\n", c[32]);
printf("%i\n", c[33]);
printf("%i\n", c[34]);
printf("%i\n", c[35]);
printf("%i\n", c[36]);
printf("%i\n", c[37]);
printf("%i\n", c[38]);
printf("%i\n", c[39]);
printf("%i\n", c[40]);
printf("%i\n", c[41]);
printf("%i\n", c[42]);
printf("%i\n", c[43]);
printf("%i\n", c[44]);
printf("%i\n", c[45]);
printf("%i\n", c[46]);
printf("%i\n", c[47]);
printf("%i\n", c[48]);
printf("%i\n", c[49]);
printf("%i\n", c[50]);
printf("%i\n", c[51]);
printf("%i\n", c[52]);
printf("%i\n", c[53]);
printf("%i\n", c[54]);
printf("%i\n", c[55]);
printf("%i\n", c[56]);
printf("%i\n", c[57]);
printf("%i\n", c[58]);
printf("%i\n", c[59]);
printf("%i\n", c[60]);
printf("%i\n", c[61]);
printf("%i\n", c[62]);
printf("%i\n", c[63]);
printf("%i\n", c[64]);
printf("%i\n", c[65]);
printf("%i\n", c[66]);
printf("%i\n", c[67]);
printf("%i\n", c[68]);
printf("%i\n", c[69]);
printf("%i\n", c[70]);
printf("%i\n", c[71]);
printf("%i\n", c[72]);
printf("%i\n", c[73]);
printf("%i\n", c[74]);
printf("%i\n", c[75]);
printf("%i\n", c[76]);
printf("%i\n", c[77]);
printf("%i\n", c[78]);
printf("%i\n", c[79]);
printf("%i\n", c[80]);
printf("%i\n", c[81]);
printf("%i\n", c[82]);
printf("%i\n", c[83]);
printf("%i\n", c[84]);
printf("%i\n", c[85]);
printf("%i\n", c[86]);
printf("%i\n", c[87]);
printf("%i\n", c[88]);
printf("%i\n", c[89]);
printf("%i\n", c[90]);
printf("%i\n", c[91]);
printf("%i\n", c[92]);
printf("%i\n", c[93]);
printf("%i\n", c[94]);
printf("%i\n", c[95]);
printf("%i\n", c[96]);
printf("%i\n", c[97]);
printf("%i\n", c[98]);
printf("%i\n", c[99]);

```

Рисунок 10: Задание №3. Работа с массивами

Содержание исследования

```
tmp_deg = []
for i in 1:3
    push!(tmp_deg, 2*tmp[i])
end
print(vcat(fill(tmp_deg, [1, 1, 4])...))
count = 0
for i in tmp_deg
    if '6' in string(i)
        count = count+1
    end
end
println('\n', count)

[16, 64, 8, 8, 8, 8]
2

using Statistics
y(x) = exp(x)*cos(x)
Y = [y(x) for x in 3:0.1:6]
mean(Y)

[ Info: Precompiling Statistics [10745b16-79ce-11e8-11f9-7d13ad32a3b2]
53.11374594642971
```

Рисунок 11: Задание №3. Работа с массивами

Содержание исследования

```

// Функция 3.12: Возврат пары (x1, y1), x = 0..3, y = 0..3, M = 30, p = 0.2, f = 1, A, B = 1, M.
x_vector = 0..3 * collect{1::10}
y_vector = 0..3 * collect{1::10}
xy_pairs = [(x, y) for x in x_vector, y in y_vector]

12x12 Matrix{Tuple{Float64, Float64}}:
(0.3, 0.2) (0.3, 0.4) (0.3, 1.6) - (0.3, 5.0) (0.3, 6.2) (0.3, 6.4)
(0.4, 0.2) (0.4, 0.4) (0.4, 1.6) - (0.4, 5.0) (0.4, 6.2) (0.4, 6.4)
(0.9, 0.2) (0.9, 0.4) (0.9, 1.6) - (0.9, 5.0) (0.9, 6.2) (0.9, 6.4)
(1.2, 0.2) (1.2, 0.4) (1.2, 1.6) - (1.2, 5.0) (1.2, 6.2) (1.2, 6.4)
(1.5, 0.2) (1.5, 0.4) (1.5, 1.6) - (1.5, 5.0) (1.5, 6.2) (1.5, 6.4)
(1.6, 0.2) (1.6, 0.4) (1.6, 1.6) - (1.6, 5.0) (1.6, 6.2) (1.6, 6.4)
(2.3, 0.2) (2.3, 0.4) (2.3, 1.6) - (2.3, 5.0) (2.3, 6.2) (2.3, 6.4)
(2.4, 0.2) (2.4, 0.4) (2.4, 1.6) - (2.4, 5.0) (2.4, 6.2) (2.4, 6.4)
(2.7, 0.2) (2.7, 0.4) (2.7, 1.6) - (2.7, 5.0) (2.7, 6.2) (2.7, 6.4)
(3.0, 0.2) (3.0, 0.4) (3.0, 1.6) - (3.0, 5.0) (3.0, 6.2) (3.0, 6.4)
(3.3, 0.2) (3.3, 0.4) (3.3, 1.6) - (3.3, 5.0) (3.3, 6.2) (3.3, 6.4)
(3.6, 0.2) (3.6, 0.4) (3.6, 1.6) - (3.6, 5.0) (3.6, 6.2) (3.6, 6.4)

// Функция 3.12: Возврат 2-элементный P(1)/2, 1 = 1, 2, ..., N, N = 20.
N = 20
work1 = [1..N * rand{Float64}]

20-element Vector{Float64}:
 2.0
 2.0
 2.0000000000000005
 4.0
 6.4
10.000000000000005
10.000000000000005
10.000000000000005
10.000000000000005
10.000000000000005
10.000000000000005
10.000000000000005
10.000000000000005
10.000000000000005
10.000000000000005
10.000000000000005
10.000000000000005
10.000000000000005
10.000000000000005
10.000000000000005
10.000000000000005

```

Рисунок 12: Задание №3. Работа с массивами

Содержание исследования

```
# Space 3.13: Reverse Bits ("01", "101", ... "100"), N = 30.  
N = 30  
arr13 = ["01" for i in 1:N]  
  
30-element Vector{String}:  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"  
"01"
```

Рисунок 13: Задание №3. Работа с массивами

Содержание исследования

```
# Рисунок 2.14.  
# Генерируем векторы x и y:  
n = 200  
x = rand(0:100, n)  
y = rand(0:100, n)  
vec3 = y[i+end] - x[i+end-1] * norm(y[2:end], p=2)  
  
140-element Vector{Int64}:  
285  
-60  
-18  
507  
-520  
31  
-476  
865  
342  
419  
215  
-520  
86  
1  
353  
487  
641  
-495  
498  
504  
575  
313  
588  
-562  
574  
598
```

Рисунок 14: Задание №3. Работа с векторами

Содержание исследования

```
vec2 = x[1:1000,1] * x[1:1000,1] - x[1:1000,2] * x[1:1000,2]
248-element Vector{Float64}:
 1788
 1138
 -137
 1808
 633
 1291
 167
 678
 295
 -249
 1494
 679
 371
 1
 1764
 1675
 88
 1112
 1278
 44
 -145
 1088
 25
 1448
 1801
 527

vec3 = sin.(y[1:1000,1]) ./ cos.(x[1:1000,1]) * bump(sin(x1)/cos(x2), sin(x2)/cos(x3), ..., sin(xn-1)/cos(xn))
4
249-element Vector{Float64}:
 71.148214190199498
 0.47514888227189735
 -29.4898984178815
 0.48949818458129844
 -1.9784027265777896
 2.5628513644536282
 -187.94581088413168
 0.54488524799468297
 -0.4943483811699885
 0.8713095704118881
```

Рисунок 15: Задание №3. Работа с векторами

```

new_exp = sum(exp(x[i:-ind]), x[i:-ind] - 1) * 0.0 # Rescaled system.
print(new_exp)
y_gt_600 = y[-600]
indices_y_gt_600 = findall(y == 600) # Rescaled y == 600.
print(indices_y_gt_600)

2005: 73739393017
61, 63, 64, 70, 71, 76, 77, 78, 80, 82, 83, 84, 85, 87, 88, 96, 87, 89, 100, 101, 102, 100, 100,
0, 112, 134, 117, 123, 134, 125, 127, 138, 132, 135, 138, 140, 145, 144, 144, 140, 153, 157, 158, 162, 16
1, 162, 163, 164, 165, 173, 176, 180, 184, 186, 188, 180, 181, 193, 195, 196, 198, 200, 201, 203, 20
4, 205, 208, 212, 216, 216, 222, 223, 224, 226, 227, 228, 229, 233, 237, 237, 239, 240, 241, 243, 245, 246
x

for y_gt_600 in x[y == 600]: # Coordinates/indices where y == y_gt_600.
    print(new_exp[y_gt_600])

[205, 635, 334, 300, 138, 212, 457, 501, 889, 18, 6, 536, 144, 112, 682, 331, 156, 32, 655, 304, 375, 87,
100, 101, 102, 100, 100, 153, 157, 158, 162, 163, 164, 165, 173, 176, 180, 184, 186, 188, 180, 181, 193, 195, 196, 198, 200, 201, 203, 204,
205, 208, 212, 216, 216, 222, 223, 224, 226, 227, 228, 229, 233, 237, 237, 239, 240, 241, 243, 245, 246]

vec = abs(x - x_mean) * 12 # norm (12 * d^1/2, 12 * d^1/2, ..., 12 * d^1/2)
vec

200-element Vector{Float64}
1.6208412598166126
1.2735848396709708
0.3014748315682789
1.9106847068682725
0.9658647914981427
0.6132638131292322
1.813078047091272929
0.4502825785811736
1.1640817837982507
1.129445758977706828
0.4141895933324229
5.406788112649648
2.073684761347442292
5.860286343285138
5.67378607895651e29
1.660181204054540632
1.87804821884773576616
1.5471565858404984625

```

Рисунок 16: Задание №3. Работа с векторами

Содержание исследования

```

count_within_200 = count(y >= minimum(y) - 200, y) # Значение y, исходя из которого не более
println(count_within_200)
even_count = count(iseven, x) # Количество четных и нечетных элементов в x.
odd_count = count(isodd, x)
println(even_count, odd_count)
multiples_of_7 = count(x -> x % 7 == 0, x) # Значение x, кратное 7.
println(multiples_of_7)

```

259
140110
39

```

sorted_x_by_y = sortperm(y) # Сортировка x по возрастанию y.

```

250-element Vector{Int64}:
172
129
854
380
941
840
886
74
762
841
22
684
423
1
928
939
992
224
987
231
182
748
798
152
937
646

Рисунок 17: Задание №3. Работа с векторами

Содержание исследования

```
top_10_x = sort(x, rev=TRUE)[1:10] # Top 10 maximum elements x
10-element vector [int64]:
906
992
987
982
982
969
965
965
968
974

unique_x = unique(x) # Уникальные элементы в x
223-element vector [int64]:
265
442
881
626
283
408
215
114
909
787
421
138
232
1
531
835
440
791
996
666
```

Рисунок 18: Задание №3. Работа с векторами

Содержание исследования

```
# Пункт 4. Создание массива squares.
squares = [i**2 for i in 1:100]
println(squares)

# Пункт 5. Работа с пакетом Primes.
using Primes

# Получаем первые 168 простых чисел
myprimes = primes(2, 1000) # Предполагаем, что первые 168 простых чисел находятся в этом диапазоне
eighty_ninth_prime = myprimes[89]
slice_89_to_99 = myprimes[89:99]
println(myprimes)
println(eighty_ninth_prime)
println(slice_89_to_99)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441, 484, 529, 5
76, 625, 676, 729, 784, 841, 900, 961, 1024, 1089, 1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764,
1849, 1936, 2025, 2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249, 3364, 3481, 360
0, 3721, 3844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5041, 5184, 5329, 5476, 5625, 5776, 5929,
6084, 6241, 6400, 6561, 6724, 6889, 7056, 7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836, 902
5, 9216, 9409, 9604, 9801, 10000]

[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, 10
3, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 22
3, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 34
7, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 46
3, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 60
7, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 74
3, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 88
3, 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]
```

```
# Пункт 6. Вычисление выражений.
sum_expr_6_1 = sum([i**3 + 4*i**2 for i in 10:100]) # Первая сумма
println(sum_expr_6_1)
M = 25
sum_expr_6_2 = sum([2 + 3/i for i in 1:M]) # Вторая сумма
println(sum_expr_6_2)
sum_expr_6_3 = 1 + sum([prod(2:i)/prod(3:i+1) for i in 2:38]) # Третья сумма
println(sum_expr_6_3)
```

Результаты

Мы изучили несколько структур данных, реализованных в Julia, научились применять их и операции над ними для решения задач.