Лабораторная 2

Структуры данных

Шалыгин Г. Э.

Российский университет дружбы народов, Москва, Россия

Информация

Докладчик

- Шалыгин Георгий Эдуардович
- студент НФИ-02-20
- Российский университет дружбы народов

Вводная часть

Цели и задачи

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

Материалы и методы

- Процессор `pandoc` для входного формата Markdown
- Результирующие форматы
  - `pdf`
  - `html`
- Автоматизация процесса создания: `Makefile`
- Компилятор Julia
- OpenModelica

Результаты

Кортежи

Повторим примеры кода работы с кортежами

In [7]:
```
()
favoritelang = ("Python","Julia","R")
# кортеж из целых чисел:
x1 = (1, 2, 3)
# кортеж из элементов разных типов:
x2 = (1, 2.0, "tmp")
# именованный кортеж:
x3 = (a=2, b=1+2)
#Примеры операций над кортежами:
# длина кортежа x2:
@show length(x2)
# обратиться к элементам кортежа x2:
@show x2[1], x2[2], x2[3]
# произвести какую-либо операцию (сложение)
# с вторым и третьим элементами кортежа x1:
@show c = x1[2] + x1[3]
# обращение к элементам именованного кортежа x3:
@show x3.a, x3.b, x3[2]
# проверка вхождения элементов tmp и 0 в кортеж x2
# (два способа обращения к методу in()):
@show in("tmp", x2), 0 in x2
```

```
length(x2) = 3
(x2[1], x2[2], x2[3]) = (1, 2.0, "tmp")
```

Словари

Примеры код работы со словарями

```
In [10]: phonebook = Dict("Иванов И.И." => ("867-5309","333-5544"),
             "Бухгалтерия" => "555-2368")
         # вывести ключи словаря:
         @show keys(phonebook)
         # вывести значения элементов словаря:
         @show values(phonebook)
         # вывести заданные в словаре пары "ключ - значение":
         @show pairs(phonebook)
         # проверка вхождения ключа в словарь:
         @show haskey(phonebook, "Иванов И.И.")
         # добавить элемент в словарь:
         phonebook["Сидоров П.С."] = "555-3344"
         # удалить ключ и связанные с ним значения из словаря
         pop!(phonebook, "Иванов И.И.")
         @show phonebook
         # Объединение словарей (функция merge()):
         a = Dict("foo" => 0.0, "bar" => 42.0);
         b = Dict("baz" => 17, "bar" => 13.0);
         @show merge(a, b), merge(b,a)

keys(phonebook) = ["Бухгалтерия", "Иванов И.И."]
values(phonebook) = Any["555-2368", ("867-5309", "333-5544")]
pairs(phonebook) = Dict{String, Any}("Бухгалтерия" => "555-2368", "Иванов И.И." => ("867-5309", "333-5544"))
haskey(phonebook, "Иванов И.И.") = true
phonebook = Dict{String, Any}("Сидоров П.С." => "555-3344", "Бухгалтерия" => "555-2368")
(merge(a, b), merge(b, a)) = (Dict{String, Real}("bar" => 13.0, "baz" => 17, "foo" => 0.0), Dict{String, Real}("bar" =>
"baz" => 17, "foo" => 0.0))
```

Figure 2: Словари

Работа с множествами

```
In [11]: @show A = Set([1, 3, 4, 5])
         @show B = Set("abrakadabra")
         # проверка эквивалентности двух множеств:
         S1 = Set([1,2]);
         S2 = Set([3,4]);
         @show issetequal(S1,S2)
         S3 = Set([1,2,2,3,1,2,3,2,1]);
         S4 = Set([2,3,1]);
         @show issetequal(S3,S4)
         #обьединение
         @show C=union(S1,S2)
         # пересечение множеств:
         @show D = intersect(S1,S3)
         # разность множеств:
         @show E = setdiff(S3,S1)
         # проверка вхождения элементов одного множества в другое:
         @show issubset(S1,S4)
         # добавление элемента в множество:
         push!(S4, 99)
         # удаление последнего элемента множества:
         pop!(S4)
         @show S4

         A = Set([1, 3, 4, 5]) = Set([5, 4, 3, 1])
         B = Set("abrakadabra") = Set(['a', 'd', 'r', 'k', 'b'])
         issetequal(S1, S2) = false
         issetequal(S3, S4) = true
         C = union(S1, S2) = Set([4, 2, 3, 1])
         D = intersect(S1, S3) = Set([2, 1])
         E = setdiff(S3, S1) = Set([3])
         issubset(S1, S4) = true
         S4 = Set([99, 3, 1])
```

Figure 3: Множества

Массивы

Создание массивов

```
# массив из квадратных корней всех целых чисел от 1 до 10:
@show roots = [sqrt(i) for i in 1:10]
# массив с элементами вида 3*x^2,
# где x - нечётное число от 1 до 9 (включительно)
@show ar_1 = [3*i^2 for i in 1:2:9]
# массив квадратов элементов, если квадрат не делится на 5 или 4:
@show ar_2=[i^2 for i=1:10 if (i^2%5!=0 && i^2%4!=0)]
```

```
roots = [sqrt(i) for i = 1:10] = [1.0, 1.4142135623730951, 1.7320508075688772, 2.0, 2.23606797749979, 2.449489742783178, 2.6457
513110645907, 2.8284271247461903, 3.0, 3.1622776601683795]
ar_1 = [3 * i ^ 2 for i = 1:2:9] = [3, 27, 75, 147, 243]
ar_2 = [i ^ 2 for i = 1:10 if i ^ 2 % 5 != 0 && i ^ 2 % 4 != 0] = [1, 9, 49, 81]
```

Figure 4: Массивы

Массивы

Создание массивов

```
# одномерный массив из пяти единиц:
ones(5)
# двумерный массив 2x3 из единиц:
@show ones(2,3)
# одномерный массив из 4 нулей:
@show zeros(4)
# заполнить массив 3x2 цифрами 3.5
@show fill(3.5,(3,2))
# заполнение массива посредством функции repeat():
@show repeat([1,2],3,3)
@show repeat([1 2],3,3)
# преобразование одномерного массива из целых чисел от 1 до 12
# в двумерный массив 2x6
a = collect(1:12)
b = reshape(a,(2,6))
# транспонирование
@show b''
# транспонирование
@show c = transpose(b)
```

```
ones(2, 3) = [1.0 1.0 1.0; 1.0 1.0 1.0]
zeros(4) = [0.0, 0.0, 0.0, 0.0]
fill(3.5, (3, 2)) = [3.5 3.5; 3.5 3.5; 3.5 3.5]
repeat([1, 2], 3, 3) = [1 1 1; 2 2 2; 1 1 1; 2 2 2; 1 1 1; 2 2 2]
```

Создание массивов

```
# Массив 10x5 целых чисел в диапазоне [10, 20]:
@show ar = rand(10:20, 10, 5)
# Выбор всех значений строки в столбце 2:
ar[:, 2]
# Выбор всех значений в столбцах 2 и 5:
ar[:, [2, 5]]
# Все значения строк в столбцах 2, 3 и 4:
ar[:, 2:4]
# значения в строках 2, 4, 6 и в столбцах 1 и 5:
ar[[2, 4, 6], [1, 5]]
# значения в строке 1 от столбца 3 до последнего столбца:
@show ar[1, 3:end]
# сортировка по столбцам:
@show sort(ar,dims=1)
# сортировка по строкам:
@show sort(ar,dims=2)
# поэлементное сравнение с числом
# (результат - массив логических значений):
@show ar .> 14
# Возврат индексов элементов массива, удовлетворяющих условию:
@show findall(ar .> 14)
```

```
ar = rand(10:20, 10, 5) = [15 16 20 10 13; 17 10 15 13 18; 18 10 20 13 12; 15 11 20 14 16; 11 14 16 20 19; 10 15 13 15 15; 13 1
3 10 18 12; 12 18 16 13 11; 11 12 15 18 13; 13 11 20 14 17]
ar[1, 3:end] = [20, 10, 13]
sort(ar, dims = 1) = [10 10 10 10 11; 11 10 13 13 12; 11 11 15 13 12; 12 11 15 13 13; 13 12 16 14 13; 13 13 16 14 15; 15 14 20
15 16; 15 15 20 18 17; 17 16 20 18 18; 18 18 20 20 19]
sort(ar, dims = 2) = [10 13 15 16 20; 10 13 15 17 18; 10 12 13 18 20; 11 14 15 16 20; 11 14 16 19 20; 10 13 15 15 15; 10 12 13
13 18; 11 12 13 16 18; 11 12 13 15 18; 11 13 14 17 20]
ar .> 14 = Bool[1 1 1 0 0; 1 0 1 0 1; 1 0 1 0 0; 1 0 1 0 1; 0 0 1 1 1; 0 1 0 1 1; 0 0 0 1 0; 0 1 1 0 0; 0 0 1 1 0; 0 0 1 0 1]
findall(ar .> 14) = CartesianIndex{2}[CartesianIndex(1, 1), CartesianIndex(2, 1), CartesianIndex(3, 1), CartesianIndex(4, 1), C
artesianIndex(1, 2), CartesianIndex(6, 2), CartesianIndex(8, 2), CartesianIndex(1, 3), CartesianIndex(2, 3), CartesianIndex(3,
```

Figure 6: Массивы

Задания для самостоятельного выполнения

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

In [33]:
```
A = Set([0,3,4,9])
B = Set([1, 3, 4, 7])
C = Set([0, 1, 2, 4, 7, 8, 9])
@show P = union(intersect(A, B), intersect(A, B), intersect(A, C), intersect(C, B))
```

```
P = union(intersect(A, B), intersect(A, B), intersect(A, C), intersect(C, B)) = Set([0, 4, 7, 9, 3, 1])
```

Задания для самостоятельного выполнения

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

```
A = Set([1, 3, 4, 5])
B = Set("abrakadabra1")
@show union(A, B)
@show intersect(A, B)
@show setdiff(A, B)
@show issubset(A, union(A, B))
```

```
union(A, B) = Set(Any[5, 4, '1', 'a', 'd', 'r', 'k', 3, 1, 'b'])
intersect(A, B) = Set{Any}()
setdiff(A, B) = Set([5, 4, 3, 1])
issubset(A, union(A, B)) = true
```

Задания для самостоятельного выполнения

3.4) массив с именем tmp вида (4, 6, 3);

3.5) массив, в котором первый элемент массива tmp повторяется 10 раз;

3.6) массив, в котором все элементы массива tmp повторяются 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 $2^{tmp[i]}$, i = 1, 2, 3, где элемент $2^{tmp[3]}$ встречается 4 раза; посчитайте в полученном векторе, сколько раз встречается цифра 6, и выведите это значение на экран;

```
tmp = [4, 6, 3]
@show a5 = fill(tmp[1], 10)
@show a5 = repeat([tmp[1]], inner=10)
@show a6 = repeat(tmp, 10)
@show a7 = [fill(tmp[1], 11); fill(tmp[2], 10); fill(tmp[3], 10)]
@show a8 = [fill(tmp[1], 10); fill(tmp[2], 20); fill(tmp[3], 30)]
```

Задания для самостоятельного выполнения

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

```
@show a8 = [tmp[i]  for i in 1:3 for _ in 1:10*i]
@show a9 = [2^tmp[i] for i in 1:3 for _ in 1:(i==3 ? 4 : 1)]
@show count(q -> q=='6', string(a9))
```

```
a8 = [tmp[i] for i = 1:3 for _ = 1:10i] = [4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
a9 = [2 ^ tmp[i] for i = 1:3 for _ = 1:if i == 3
                        4
                    else
                        1
                    end] = [16, 64, 8, 8, 8, 8]
count((q->begin
            #= In[79]:3 =#
            q == '6'
        end), string(a9)) = 2
```

Задания для самостоятельного выполнения

3.10) вектор значений $y = e^x cos(x)$ в точках x = 3, 3.1, 3.2, … , 6, найдите среднее значение y;

3.11) вектор вида (x^i, y^j), x = 0.1, i = 3, 6, 9, … , 36, y = 0.2, j = 1, 4, 7, … , 34;

3.12) вектор с элементами $\frac{y^i}{i}$, i = 1, 2, … , M, M = 25;

3.13) вектор вида ("fn1", "fn2", …, "fnN"), N = 30;

```
xx = 3:0.1:6
@show a10 = [exp(x)*cos(x) for x in xx]
@show a11 = [(0.1^i, 0.2^(i-2)) for i in 3:3:36]
```

```
a10 = [exp(x) * cos(x) for x = xx] = [-19.884530844146987, -22.178753389342127, -24.490696732801293, -26.77318244299338, -28.969237768093574, -31.011186439374516, -32.819774760338504, -34.303360110373769, -35.357193618530 35, -35.862833712307 67, -35.687732 48011913, -34.68504225166807, -32.693695428321746, -29.538816297262983, -25.032529229039966, -18.975233154958957, -11.157417389 647478, -1.3620985182057503, 10.632038010191998, 25.046704998273004, 42.099201062538 39, 61.99663027669454, 84.92906736250268, 111.06158604202 58, 140.52507505278 75, 173.40577640857734, 209.73349424783467, 249.46844055885668, 292.4867067371223, 338.5643778 585117, 387.36034029093076]
a11 = [(0.1 ^ i, 0.2 ^ (i - 2)) for i = 3:3:36] = [(0.001000000000000000 2, 0.2), (1.0000000000000004e-6, 0.0016000000000000 03), (1.0000000000000005e-9, 1.2800000000000006e-5), (1.0000000000000008e-12, 1.0240000000000006e-7), (1.0000000000000009e-15, 8.192000000000005e-10), (1.0000000000000008e-18, 6.553600000000055e-12), (1.0000000000000012e-21, 5.242880000000056e-14), (1.0000000000000012e-24, 4.194304000000005e-16), (1.0000000000000015e-27, 3.355443200000044e-18), (1.0000000000000017e-30, 2.6843 54560000004e-20), (1.0000000000000018e-33, 2.1474836480000 04e-22), (1.0000000000000002e-36, 1.7179869184000035e-24)]
```

Задания для самостоятельного выполнения

3.12) вектор с элементами $\frac{2^i}{i}$, i = 1, 2, … , M, M = 25;

3.13) вектор вида ("fn1", "fn2", …, "fnN"), N = 30;

```
@show a12 = [2^i/i for i in 1:25]
@show a13 = ["fn$i" for i in 1:30]
```

```
a12 = [2 ^ i / i for i = 1:25] = [2.0, 2.0, 2.6666666666666665, 4.0, 6.4, 10.666666666666666, 18.285714285714285, 32.0, 56.8888
88888888886, 102.4, 186.18181818181182, 341.3333333333333, 630.1538461538462, 1170.2857142857142, 2184.5333333333333, 4096.0, 77
10.1176470588823, 14563.555555555555, 27594.105263157893, 52428.8, 99864.38095238095, 190650.18181818182, 364722.0869565217, 699
050.6666666666, 1.34217728e6]
a13 = ["fn$(i)" for i = 1:30] = ["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"]
```

Задания для самостоятельного выполнения

векторы x и y целочисленного типа длины n = 250 как случайные выборки из совокупности 0, 1, … , 999; на его основе:

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

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

– сформируйте вектор $(\frac{sin(y1)}{cos(x2)}, \dots, \frac{sin(yn-1)}{cos(xn)})$;

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

]:
```
n = 250
x = rand(0:999, n)
y = rand(0:999, n)
v1 = [y[i]-x[i-1] for i in 2:n]
v2 = [x[i] + 2*x[i+1] - x[i+2] for i in 1:n-2]
v3 = [sin(y[i]) / cos(x[i+1]) for i in 1:n-1]
S = sum([exp(-x[i+1]) / (x[i] + 10) for i in 1:n-1])
```

]:
```
0.0025233553067527583
```

Задания для самостоятельного выполнения

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

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

– сформируйте вектор $(|x_1 - \bar{x}|^{0.5}, \ldots, |x_n - \bar{x}|^{0.5})$, где $\bar{x}$ обозначает среднее значение вектора x

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

```
@show y[y .> 600]
@show findall(y .> 600)
@show x[y .> 600]
v4 = [abs(x[i] - sum(x)/n)^0.5 for i in 1:n]
@show sum([abs(y[i] - maximum(y)) for i in 1:n] .<= 200)
```

```
y[y .> 600] = [984, 650, 869, 619, 714, 920, 750, 672, 805, 752, 696, 640, 926, 863, 787, 681, 776, 976, 949, 711, 852, 892, 707, 847, 622, 889, 739, 983, 923, 794, 874, 847, 689, 950, 855, 852, 948, 657, 850, 971, 670, 869, 604, 752, 702, 634, 669, 920, 913, 769, 999, 968, 736, 722, 858, 850, 804, 976, 850, 764, 684, 886, 889, 623, 717, 918, 758, 756, 787, 817, 747, 620, 962, 837, 841, 777, 765, 721, 829, 868, 763, 989, 877, 992, 888, 826, 946, 901, 640, 896, 702, 816, 736, 665, 860, 721, 904, 934, 960, 955, 739, 921, 945, 651, 997, 862]
findall(y .> 600) = [2, 3, 8, 9, 11, 12, 15, 18, 19, 22, 30, 33, 34, 35, 36, 42, 45, 46, 52, 53, 54, 56, 58, 61, 62, 63, 66, 67, 70, 73, 75, 77, 79, 82, 85, 88, 90, 100, 103, 107, 108, 109, 113, 114, 118, 119, 120, 121, 127, 128, 129, 130, 131, 133, 134, 140, 142, 143, 144, 145, 149, 151, 153, 155, 157, 161, 165, 166, 167, 171, 173, 174, 182, 183, 185, 189, 190, 191, 192, 193, 194, 195, 196, 201, 203, 204, 207, 208, 209, 212, 214, 215, 216, 217, 218, 220, 222, 226, 229, 230, 232, 237, 239, 240, 245, 248]
x[y .> 600] = [352, 348, 394, 102, 598, 649, 848, 368, 89, 914, 927, 589, 704, 29, 221, 16, 188, 2, 41, 931, 685, 93, 361, 36, 497, 563, 417, 594, 128, 344, 339, 182, 367, 865, 118, 97, 667, 369, 402, 383, 836, 927, 867, 451, 877, 823, 380, 481, 841, 767, 607, 733, 738, 482, 189, 50, 214, 116, 651, 412, 80, 115, 913, 180, 175, 932, 335, 812, 177, 503, 458, 525, 553, 760, 368, 943, 64, 561, 485, 737, 750, 22, 745, 814, 668, 336, 600, 943, 859, 85, 761, 816, 766, 747, 410, 435, 479, 277, 491, 635, 375, 19, 586, 294, 750, 463]
sum([abs(y[i] - maximum(y)) for i in 1:n] .<= 200) = 59
```

Задания для самостоятельного выполнения

– определите, сколько чётных и нечётных элементов вектора x;

– определите, сколько элементов вектора x кратны 7;

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

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

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

```
println(count(x -> x % 2 == 0, x), " четные")
println(count(x -> x % 2 == 1, x), " нечетные")
println(count(x -> x % 7 == 0, x), " делятся на 7")
@show x[sortperm(y)]
println("10 максимальных: ", sort(x, rev=true)[1:10])
unique_x = unique(x)
```

```
120 четные
130 нечетные
29 делятся на 7
x[sortperm(y)] = [564, 244, 889, 988, 88, 152, 491, 606, 933, 266, 21, 318, 668, 575, 815, 501, 769, 725, 221, 805, 610, 599, 8
61, 592, 718, 331, 233, 623, 286, 886, 871, 410, 671, 257, 596, 783, 747, 221, 86, 285, 277, 360, 280, 859, 360, 906, 79, 668,
397, 919, 522, 941, 711, 207, 645, 656, 410, 456, 577, 279, 524, 45, 968, 804, 90, 682, 66, 434, 481, 453, 377, 289, 258, 10, 3
65, 956, 359, 494, 745, 50, 976, 538, 526, 746, 963, 568, 374, 2, 392, 670, 357, 755, 662, 204, 14, 423, 775, 790, 364, 681, 57
2, 639, 401, 276, 552, 400, 678, 865, 153, 49, 926, 557, 323, 859, 454, 307, 1, 106, 296, 773, 673, 750, 308, 41, 171, 624, 97
9, 899, 22, 142, 674, 363, 650, 197, 980, 928, 774, 558, 733, 637, 905, 434, 35, 659, 867, 102, 525, 497, 180, 823, 589, 859, 3
48, 294, 369, 747, 380, 836, 368, 16, 80, 367, 927, 877, 761, 361, 931, 598, 175, 561, 435, 482, 738, 766, 417, 375, 458, 848,
914, 451, 812, 335, 750, 412, 64, 767, 188, 943, 221, 177, 344, 214, 89, 816, 503, 336, 485, 760, 368, 36, 182, 402, 50, 651, 6
85, 97, 118, 189, 410, 463, 29, 737, 394, 927, 339, 745, 115, 668, 563, 913, 93, 85, 943, 479, 841, 932, 649, 481, 19, 128, 70
4, 277, 586, 600, 667, 41, 865, 635, 491, 553, 733, 383, 2, 116, 594, 352, 22, 814, 750, 607]
10 максимальных: [988, 980, 979, 976, 968, 963, 956, 943, 943, 941]
```

Задания для самостоятельного выполнения

Создайте массив squares, в котором будут храниться квадраты всех целых чисел от 1 до 100.

```
: squares = [x^2 for x in 1:100]
```

```
: 100-element Vector{Int64}:
       1
       4
```

Задания для самостоятельного выполнения

Подключите пакет Primes (функции для вычисления простых чисел). Сгенерируйте массив myprimes, в котором будут храниться первые 168 простых чисел. Определите 89-е наименьшее простое число. Получите срез массива с 89-го до 99-го элемента включительно, содержащий наименьшие простые числа.

```
import Pkg; Pkg.add("Primes")
```

```
┌ Warning: could not download https://pkg.julialang.org/registries
│   exception = InterruptException:
└ @ Pkg.Registry C:\workdir\usr\share\julia\stdlib\v1.8\Pkg\src\Registry\Registry.jl:68
    Updating registry at `D:\julia\depot\registries\General.toml`
   Resolving package versions...
    Installed IntegerMathUtils ─ v0.1.2
    Installed Primes ──────────── v0.5.4
    Updating `D:\julia\depot\environments\v1.8\Project.toml`
  [27ebfcd6] + Primes v0.5.4
    Updating `D:\julia\depot\environments\v1.8\Manifest.toml`
  [18e54dd8] + IntegerMathUtils v0.1.2
  [27ebfcd6] + Primes v0.5.4
Precompiling project...
  ✓ IntegerMathUtils
  ✓ Primes
  2 dependencies successfully precompiled in 5 seconds. 272 already precompiled. 2 skipped during auto due to previous errors.
```

```
using Primes
```

Задания для самостоятельного выполнения

```
myprimes = primes(1000)[1:168]
println(myprimes[89])
println(myprimes[89:99])
```

```
461
[461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]
```

Вывод

Вывод

В ходе работы были изучены несколько структур данных, реализованных в Julia, научились применять их и операции над ними для решения задач.