

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ

### ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

дисциплина: Компьютерный практикум

по математическому моделированию

Студент: Абрамян Артём \_

Группа: НПИбд-01-20

МОСКВА

2023 г.

## **Постановка задачи**

Освоить применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.

## **Выполнение работы**

### **1. Повторите примеры из раздела 3.2**

#### **1.1 Циклы while и for**

Для различных операций, связанных с перебором индексируемых элементов структур данных, традиционно используются циклы while и for. Синтаксис while <condition>

<body>

end

Примеры:

```
• begin
•   n = 0
•   while n < 10
•     n += 1
•     println(n)
•   end
• end
```

```
1
2
3
4
5
6
7
8
9
10
```

```
• begin
•   myfriends = ["Ted", "Robyn", "Barney", "Lily", "Marshall"]
•   i = 1
•   while i <= length(myfriends)
•     friend = myfriends[i]
•     println("Hi $friend, it's great to see you!")
•     i += 1
•   end
• end
```

```
Hi Ted, it's great to see you!
Hi Robyn, it's great to see you!
Hi Barney, it's great to see you!
Hi Lily, it's great to see you!
Hi Marshall, it's great to see you!
```

Такие же результаты можно получить при использовании цикла for.

Синтаксис for

for <переменная> in <диапазон>

<тело цикла>

end

Рассмотренные выше примеры, но с использованием цикла for:

```

• begin
•   for n in 1:2:10
•     println(n)
•   end
•   myfriends = ["Ted", "Robyn", "Barney", "Lily", "Marshall"]
•   for friend in myfriends
•     println("Hi $friend, it's great to see you!")
•   end
• end

```

```

1
3
5
7
9
Hi Ted, it's great to see you!
Hi Robyn, it's great to see you!
Hi Barney, it's great to see you!
Hi Lily, it's great to see you!
Hi Marshall, it's great to see you!

```

Пример использования цикла `for` для создания двумерного массива, в котором значение каждой записи является суммой индексов строки и столбца:

```

5x5 Matrix{Int64}:
 2  3  4  5  6
 3  4  5  6  7
 4  5  6  7  8
 5  6  7  8  9
 6  7  8  9 10

```

```

• begin
•   # инициализация массива m x n из нулей:
•   m, n = 5, 5
•   A = fill{0, (m, n)}
•   # формирование массива, в котором значение каждой записи
•   # является суммой индексов строки и столбца:
•   for i in 1:m
•     for j in 1:n
•       A[i, j] = i + j
•     end
•   end
•   A
• end

```

Другая реализация этого же примера:

5x5 Matrix{Int64}:

```
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
6 7 8 9 10
```

```
• begin
•     # инициализация массива m x n из нулей:
•     B = fill{0, (m, n)}
•     for i in 1:m, j in 1:n
•         B[i, j] = i + j
•     end
•     B
•
•     C = [i + j for i in 1:m, j in 1:n]
•     C
•
• end
```

## 1.2 Условные выражения

Довольно часто при решении задач требуется проверить выполнение тех или иных условий. Для этого используют условные выражения.

Синтаксис условных выражений с ключевым словом:

if <условие 1>

<действие 1>

elseif <условие 2>

<действие 2>

else

<действие 3>

end

Например, пусть для заданного числа  $N$  требуется вывести слово «Fizz», если  $N$  делится на 3, «Buzz», если  $N$  делится на 5, и «FizzBuzz», если  $N$  делится на 3 и 5:

```
. begin
.   N = 10
.   # используем '&&' для реализации операции "AND"
.   # операция % вычисляет остаток от деления
.   if (N % 3 == 0) && (N % 5 == 0)
.     println("FizzBuzz")
.   elseif N % 3 == 0
.     println("Fizz")
.   elseif N % 5 == 0
.     println("Buzz")
.   else
.     println(N)
.   end
. end
```



Buzz



Синтаксис условных выражений с тернарными операторами:

$a ? b : c$

(если выполнено  $a$ , то выполнить  $b$ , если нет, то  $c$ ).

Такая запись эквивалентна записи условного выражения с ключевым словом:

if  $a$

$b$

else

$c$

end

Пример использования тернарного оператора:

10

```
- begin  
-   x = 5  
-   y = 10  
-   (x > y) ? x : y  
- end
```

### 1.3 Функции

Julia дает нам несколько разных способов написать функцию. Первый требует ключевых слов `function` и `end`:

```
function sayhi(name)

println("Hi $name, it's great to see you!")

end
```

# функция возведения в квадрат:

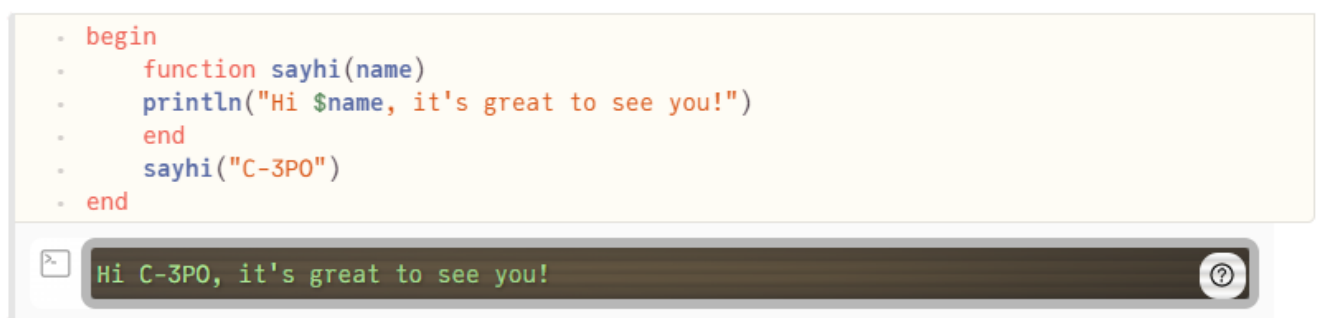
```
function f(x)

x^2

end
```

Вызов функции осуществляется по её имени с указанием аргументов, например:

```
sayhi("С-ЗРО")
```



```
• begin
•   function sayhi(name)
•       println("Hi $name, it's great to see you!")
•   end
•   sayhi("С-ЗРО")
• end
```

Hi С-ЗРО, it's great to see you!

```
f(42)
```

В качестве альтернативы, можно объявить любую из выше определённых функций

в одной строке:

```
sayhi2(name) = println("Hi $name, it's great to see you!")
```



$$f2(x) = x^2$$

Наконец, можно объявить выше определённые функции как «анонимные»:

```
25
. begin
.   sayhi3 = name -> println("Hi $name, it's great to see you!")
.   f3 = x -> x^2
.   sayhi3("Artem")
.   f3(5)
. end
```

```
> Hi Artem, it's great to see you!
```

По соглашению в Julia функции, сопровождаемые восклицательным знаком, изменяют свое содержимое, а функции без восклицательного знака не делают этого.

Например, сравните результат применения `sort` и `sort!`:

```
► [2, 3, 5]
```

```
. begin
.   # задаём массив v:
.   v = [3, 5, 2]
.   sort(v)
.   v
.   sort!(v)
.   v
. end
```

Функция `sort(v)` возвращает отсортированный массив, который содержит те же элементы, что и массив `v`, но исходный массив `v` остаётся без изменений. Если же использовать `sort!(v)`, то отсортировано будет содержимое исходного массива `v`. В программировании под функцией высшего порядка понимается функция, принимающая в качестве аргументов другие функции или возвращающая другую функцию в качестве результата. Основная идея состоит в том, что функции имеют тот же статус, что и другие объекты данных. В Julia функция `map` является функцией высшего порядка, которая принимает функцию в качестве одного из своих входных аргументов и применяет эту

функцию к каждому элементу структуры данных, которая ей передаётся также в качестве аргумента. Например, в случае выполнения выражения:

```
map(f, [1, 2, 3])
```

на выходе получим массив, в котором функция  $f$  была применена ко всем элементам

массива  $[1, 2, 3]$ :

```
[f(1), f(2), f(3)]
```

Если  $f(x) = x^2$

, то получим следующий результат:

```
f(x) = x^2
```

```
map(f, [1, 2, 3])
```

3-element Array{Int64,1}:

1

4

9

То есть в квадрат возведены все элементы массива  $[1, 2, 3]$ , но не сам массив (вектор).

```
► [1, 4, 9]
. begin
.   f(x) = x^2
.   map(f, [1, 2, 3])
. end
```

В `map` можно передать и анонимно заданную, а не именованную функцию:

```

▶ [1, 8, 27]

- begin
-     x -> x^3
-     map(x -> x^3, [1, 2, 3])
- end

```

Функция `broadcast` — ещё одна функция высшего порядка в Julia, представляющая собой обобщение функции `map`. Функция `broadcast()` будет пытаться привести все объекты к общему измерению, `map()` будет напрямую применять данную функцию поэлементно. Синтаксис для вызова `broadcast` такой же, как и для вызова `map`, например применение функции `f` к элементам массива `[1, 2, 3]`:

```

3×3 Matrix{Int64}:
 30  36  42
 66  81  96
102 126 150

- begin
-     f(x) = x^2
-     broadcast(f, [1, 2, 3])
-
-     f.([1, 2, 3])
-     # Задаём матрицу A:
-     A = [i + 3*j for j in 0:2, i in 1:3]
-     f(A)
-
- end

```

```

3×3 Matrix{Int64}:
 1  4  9
16 25 36
49 64 81

- begin
-     f(x) = x^2
-     broadcast(f, [1, 2, 3])
-
-     f.([1, 2, 3])
-     # Задаём матрицу A:
-     A = [i + 3*j for j in 0:2, i in 1:3]
-     f(A)
-     B = f.(A)
-
- end

```

Точечный синтаксис для `broadcast()` позволяет записать относительно сложные

составные поэлементные выражения в форме, близкой к математической записи.

Например, запись

`A .+ 2 .* f(A) ./ A`

или запись

`@. A + 2 * f(A) / A`

аналогичны записи

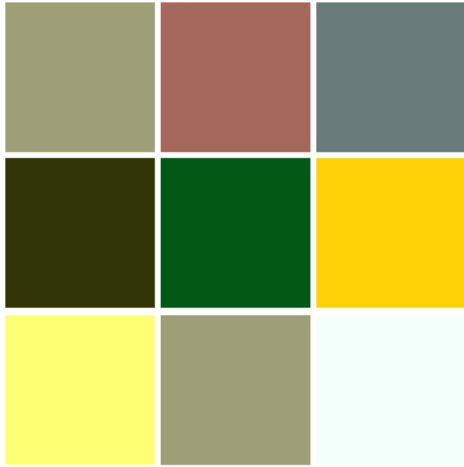
`broadcast(x -> x + 2 * f(x) / x, A)`

```
3×3 Matrix{Float64}:
 3.0  6.0  9.0
12.0 15.0 18.0
21.0 24.0 27.0

- begin
-   f(x) = x^2
-   broadcast(f, [1, 2, 3])
-
-   f.([1, 2, 3])
-   # Задаём матрицу A:
-   A = [i + 3*j for j in 0:2, i in 1:3]
-   f(A)
-   B = f.(A)
-
-   broadcast(x -> x + 2 * f(x) / x, A)
-
- end
```

## 1.4 Сторонние библиотеки (пакеты) в Julia

Julia имеет более 2000 зарегистрированных пакетов, что делает их огромной частью экосистемы Julia. Есть вызовы функций первого класса для других языков, обеспечивающие интерфейсы сторонних функций. Можно вызвать функции из Python или R, например, с помощью PyCall или Rcall. С перечнем доступных в Julia пакетов можно ознакомиться на страницах следующих ресурсов: — <https://julialang.org/packages/> — <https://juliahub.com/ui/Home> — <https://juliaobserver.com/> — <https://github.com/svaksha/Julia.jl> При первом использовании пакета в вашей текущей установке Julia вам необходимо использовать менеджер пакетов, чтобы явно его добавить: `import Pkg` `Pkg.add("Example")` При каждом новом использовании Julia (например, в начале нового сеанса в REPL или открытии блокнота в первый раз) нужно загрузить пакет, используя ключевое слово `using`: Например, добавим и загрузим пакет `Colors`:



```
begin
  using Colors ✓
  # Затем создадим палитру из 100 разных цветов:
  palette = distinguishable_colors(100)

  rand(palette, 3, 3)
end
```

▶ 94.7 ms

## 2. Задания для самостоятельной работы

### 2.1 Используя циклы while и for:

```
. begin
.   # – выведите на экран целые числа от 1 до 100 и напечатайте их квадраты;
.   for i in 1:1:100
.       print(i)
.       print(" ")
.   end
.   println()
.   for i in 1:1:100
.       print(i^2)
.       print(" ")
.   end
. end
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 2
9 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 8
2 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361 400 441 484 529
576 625 676 729 784 841 900 961 1024 1089 1156 1225 1296 1369 1444 1521 1600 16
81 1764 1849 1936 2025 2116 2209 2304 2401 2500 2601 2704 2809 2916 3025 3136 3
249 3364 3481 3600 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 9025 9216 9409 9604 9801 10000
```

► Dict{5 ⇒ 25, 4 ⇒ 16, 6 ⇒ 36, 7 ⇒ 49, 2 ⇒ 4, 10 ⇒ 100, 9 ⇒ 81, 8 ⇒ 64, 3 ⇒ 9, 1 ⇒

```
. begin
.   #– создайте словарь squares, который будет содержать целые числа в качестве
.   ключей и # квадраты в качестве их пар-значений;
.   squares = Dict{Int64, Int64}{}
.   for i in 1:1:10
.       push!(squares, i => i^2)
.   end
.   pairs(squares)
. end
```

▶ [1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, ...]

```
begin
  # - создайте массив squares_arr, содержащий квадраты всех чисел от 1 до 100.
  squares_arr = [];
  i = 1
  while (i <= 100)
    push!(squares_arr, i^2)
    i += 1;
  end
  squares_arr
end
```

47.8  $\mu s$

**2.2 Напишите условный оператор, который печатает число, если число чётное, и строку «нечётное», если число нечётное. Перепишите код, используя тернарный оператор.**

```
. begin
.   begin
.     x = 2
.     if (x % 2 == 0)
.       println("even")
.     else
.       println("odd")
.     end
.   end
.
.   (x % 2 == 0) ? println("even") : println("odd")
. end
```



```
even
even
```





### 2.3 Напишите функцию `add_one`, которая добавляет 1 к своему входу.

```
3
. begin
.   function add_one(x)
.       x + 1
.   end
.
.   add_one(2)
. end
```

### 2.4 Используйте `map()` или `broadcast()` для задания матрицы $A$ , каждый элемент которой увеличивается на единицу по сравнению с предыдущим.

```
3x3 Matrix{Int64}:
 2  3  4
 5  6  7
 8  9 10

. begin
.   A = [i + 3*j for j in 0:2, i in 1:3]
.   map(l -> l + 1, A)
. end
```

### 2.5

5. Задайте матрицу  $A$  следующего вида:

$$A = \begin{pmatrix} 1 & 1 & 3 \\ 5 & 2 & 6 \\ -2 & -1 & -3 \end{pmatrix}.$$

- Найдите  $A^3$ .
- Замените третий столбец матрицы  $A$  на сумму второго и третьего столбцов.

```
3x3 Matrix{Int64}:
 1  1  4
 5  2  8
-2 -1 -4

. begin
.   A = [1 1 3; 5 2 6; -2 -1 -3]
.   A^3
.   for i in 7:1:9
.       A[i] += A[i - 3]
.   end
.   A
. end
```

## 2.6

- Замените третий столбец матрицы  $A$  на сумму второго и третьего столбцов.
6. Создайте матрицу  $B$  с элементами  $B_{i1} = 10, B_{i2} = -10, B_{i3} = 10, i = 1, 2, \dots, 15$ .  
Вычислите матрицу  $C = B^T B$ .

```
3x3 Matrix{Int64}:
 1500  -1500  1500
-1500   1500 -1500
 1500  -1500  1500

- begin
-   B = Array{Int64, 2}(undef, 15, 3)
-
-   for i in 1:1:15
-       B[i,1] = 10
-       B[i,2] = -10
-       B[i,3] = 10
-   end
-
-   B
-
-   C = B' * B
- end
```

## 2.7

Создайте матрицу  $Z$  размерности  $6 \times 6$ , все элементы которой равны нулю, и матрицу  $E$ , все элементы которой равны 1. Используя цикл `while` или `for` и закономерности расположения элементов, создайте следующие матрицы размерности  $6 \times 6$ :

$$Z_1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad Z_2 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix},$$

$$Z_3 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \quad Z_4 = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

```

6x6 Matrix{Int64}:
0 1 0 0 0 0
1 0 1 0 0 0
0 1 0 1 0 0
0 0 1 0 1 0
0 0 0 1 0 1
0 0 0 0 1 0

```

```

- begin
-     Z = zeros(Int64, 6,6)
-     E = ones(Int64, 6,6)
-
-     Z1 = zeros(Int64, 6,6)
-     for i in 1:1:6
-         if (i != 1)
-             Z1[i, i - 1] = E[i, i-1]
-         end
-
-         if (i != 6)
-             Z1[i, i + 1] = E[i, i+1]
-         end
-     end
-     Z1
- end

```

```

6x6 Matrix{Int64}:
1 0 1 0 0 0
0 1 0 1 0 0
1 0 1 0 1 0
0 1 0 1 0 1
0 0 1 0 1 0
0 0 0 1 0 1

```

```

- begin
-     Z2 = zeros(Int64, 6,6)
-     for i in 1:1:6
-         Z2[i,i] = 1
-         if (i+2 <= 6)
-             Z2[i, i+2] = E[i,i+2]
-         end
-
-         if (i-2 >= 1)
-             Z2[i, i-2] = E[i,i-2]
-         end
-     end
-     Z2
- end

```

```
6x6 Matrix{Int64}:
```

```
0 0 0 1 0 1
0 0 1 0 1 0
0 1 0 1 0 1
1 0 1 0 1 0
0 1 0 1 0 0
1 0 1 0 0 0
```

```
. begin
.   Z3 = zeros(Int64, 6,6)
.   for i in 1:1:6
.       Z3[i,7-i] = 1
.       if (7-i + 2 <= 6)
.           Z3[i, 9-i] = E[i,9-i]
.       end
.
.       if (7-i-2 >= 1)
.           Z3[i, 5-i] = E[i,5-i]
.       end
.   end
.   Z3
. end
```

```
6x6 Matrix{Int64}:
```

```
1 0 1 0 1 0
0 1 0 1 0 1
1 0 1 0 1 0
0 1 0 1 0 1
1 0 1 0 1 0
0 1 0 1 0 1
```

```
. begin
.   Z4 = zeros(Int64, 6,6)
.   for i in 1:1:6
.       Z4[i,i] = 1
.       if (i + 2 <= 6)
.           Z4[i, i + 2] = E[i,i+2]
.       end
.
.       if (i - 2 >= 1)
.           Z4[i, i - 2] = E[i,i-2]
.       end
.
.       if (i + 4 <= 6)
.           Z4[i, i + 4] = E[i,i+4]
.       end
.
.       if (i - 4 >= 1)
.           Z4[i, i - 4] = E[i,i-4]
.       end
.
.   end
.   Z4
. end
```

**2.8 В языке R есть функция `outer()`. Фактически, это матричное умножение с возможностью изменить применяемую операцию (например, заменить произведение на сложение или возведение в степень).**

- Напишите свою функцию, аналогичную функции `outer()` языка R. Функция должна иметь следующий интерфейс: `outer(x, y, operation)`. Таким образом, функция вида `outer(A, B, *)` должна быть эквивалентна произведению матриц  $A$  и  $B$  размерностями  $L \times M$  и  $M \times N$  соответственно, где элементы резуль-

тирующей матрицы  $C$  имеют вид  $C_{ij} = \sum_{k=1}^M A_{ik} B_{kj}$  (или в тензорном виде  $C_i^j = \sum_{k=1}^M A_k^i B_j^k$ ).

`outer` (generic function with 1 method)

```

• function outer(x, y, operation)
•   if (ndims(x) == 1)
•     x = reshape(x, (size(x, 1), size(x, 2)))
•   end
•   if (ndims(y) == 1)
•     y = reshape(y, (size(y, 1), size(y, 2)))
•   end
•   c = zeros(size(x)[1], size(y)[2])
•
•   for i in 1:size(x)[1], j in 1:size(y)[2], k in 1:size(x)[2]
•     c[i, j] += operation(x[i, k], y[k, j])
•   end
•   return c
• end

```

- Используя написанную вами функцию `outer()`, создайте матрицы следующей структуры:

$$A_1 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 4 & 8 & 16 & 32 \\ 3 & 9 & 27 & 81 & 243 \\ 4 & 16 & 64 & 256 & 1024 \end{pmatrix}.$$

$$A_3 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 0 \\ 2 & 3 & 4 & 0 & 1 \\ 3 & 4 & 0 & 1 & 2 \\ 4 & 0 & 1 & 2 & 3 \end{pmatrix}, \quad A_4 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix},$$

$$A_5 = \begin{pmatrix} 0 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 1 & 0 & 8 & 7 & 6 & 5 & 4 & 3 & 2 \\ 2 & 1 & 0 & 8 & 7 & 6 & 5 & 4 & 3 \\ 3 & 2 & 1 & 0 & 8 & 7 & 6 & 5 & 4 \\ 4 & 3 & 2 & 1 & 0 & 8 & 7 & 6 & 5 \\ 5 & 4 & 3 & 2 & 1 & 0 & 8 & 7 & 6 \\ 6 & 5 & 4 & 3 & 2 & 1 & 0 & 8 & 7 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & 8 \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{pmatrix}.$$

```

A = 5x5 Matrix{Float64}:
 0.0  1.0  2.0  3.0  4.0
 1.0  2.0  3.0  4.0  5.0
 2.0  3.0  4.0  5.0  6.0
 3.0  4.0  5.0  6.0  7.0
 4.0  5.0  6.0  7.0  8.0

- A = outer(collect(0:4), collect(0:4)', +)

A2 = 5x5 Matrix{Float64}:
 0.0  0.0  0.0  0.0  0.0
 1.0  1.0  1.0  1.0  1.0
 2.0  4.0  8.0  16.0  32.0
 3.0  9.0  27.0  81.0  243.0
 4.0  16.0  64.0  256.0  1024.0

- A2 = outer(collect(0:4), collect(1:5)', ^)

A3 = 5x5 Matrix{Float64}:
 0.0  1.0  2.0  3.0  4.0
 1.0  2.0  3.0  4.0  0.0
 2.0  3.0  4.0  0.0  1.0
 3.0  4.0  0.0  1.0  2.0
 4.0  0.0  1.0  2.0  3.0

- A3 = outer(collect(0:4), collect(0:4)', +).%5

A4 = 10x10 Matrix{Float64}:
 0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0
 1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0  0.0
 2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0  0.0  1.0
 3.0  4.0  5.0  6.0  7.0  8.0  9.0  0.0  1.0  2.0
 4.0  5.0  6.0  7.0  8.0  9.0  0.0  1.0  2.0  3.0
 5.0  6.0  7.0  8.0  9.0  0.0  1.0  2.0  3.0  4.0
 6.0  7.0  8.0  9.0  0.0  1.0  2.0  3.0  4.0  5.0
 7.0  8.0  9.0  0.0  1.0  2.0  3.0  4.0  5.0  6.0
 8.0  9.0  0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0
 9.0  0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0

- A4 = outer(collect(0:9), collect(0: 9)') +).%10

A5 = 9x9 Matrix{Float64}:
 0.0  8.0  7.0  6.0  5.0  4.0  3.0  2.0  1.0
 1.0  0.0  8.0  7.0  6.0  5.0  4.0  3.0  2.0
 2.0  1.0  0.0  8.0  7.0  6.0  5.0  4.0  3.0
 3.0  2.0  1.0  0.0  8.0  7.0  6.0  5.0  4.0
 4.0  3.0  2.0  1.0  0.0  8.0  7.0  6.0  5.0
 5.0  4.0  3.0  2.0  1.0  0.0  8.0  7.0  6.0
 6.0  5.0  4.0  3.0  2.0  1.0  0.0  8.0  7.0
 7.0  6.0  5.0  4.0  3.0  2.0  1.0  0.0  8.0
 8.0  7.0  6.0  5.0  4.0  3.0  2.0  1.0  0.0

- A5 = outer(collect(0:8), collect(-9:-1)', -).%9

```

Q

Live Docs

Status

.

%

.

%

Any::DataType

Any

is the union of all types. It has the defining property `isa(x, Any) == true` for any `x`. `Any` therefore describes the entire universe of possible values. For example `Integer` is a subset of `Any` that includes `Int`, `Int8`, and other integer types.

## 2.9

Решите следующую систему линейных уравнений с 5 неизвестными:

$$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 = 7, \\ 2x_1 + x_2 + 2x_3 + 3x_4 + 4x_5 = -1, \\ 3x_1 + 2x_2 + x_3 + 2x_4 + 3x_5 = -3, \\ 4x_1 + 3x_2 + 2x_3 + x_4 + 2x_5 = 5, \\ 5x_1 + 4x_2 + 3x_3 + 2x_4 + x_5 = 17, \end{cases}$$

рассмотрев соответствующее матричное уравнение  $Ax = y$ . Обратите внимание на особый вид матрицы  $A$ . Метод, используемый для решения данной системы уравнений, должен быть легко обобщаем на случай большего числа уравнений, где матрица  $A$  будет иметь такую же структуру.

► [-2, 3, 5, 2, -4]

```
• begin
•   # коэффициенты из системы уравнений
•   array_system = Array{Int64, 2}(undef, 5, 5)
•   m = 5
•   n = 5
•   for i in 1:1:m
•       for j in 1:1:n,
•           array_system[i, j] = 1 + abs(i - j)
•       end
•   end
•   println(round.(Int32, array_system))
•
•   answers = [7; -1; -3; 5; 17]
•
•   array_system_inverted = inv(array_system)
•   round.(Int, array_system_inverted)
•   x_n = round.(Int, array_system_inverted * answers)
• end
```

## 2.10

Создайте матрицу  $M$  размерности  $6 \times 10$ , элементами которой являются целые числа, выбранные случайным образом с повторениями из совокупности  $1, 2, \dots, 10$ .

- Найдите число элементов в каждой строке матрицы  $M$ , которые больше числа  $N$  (например,  $N = 4$ ).
- Определите, в каких строках матрицы  $M$  число  $M$  (например,  $M = 7$ ) встречается ровно 2 раза?
- Определите все пары столбцов матрицы  $M$ , сумма элементов которых больше  $K$  (например,  $K = 75$ ).

```
• begin
•   M = rand(1:10, 6, 10)
•   # - Найдите число элементов в каждой строке матрицы M, которые больше числа N
•   #(например, N = 4).
•   N = 4
•   for i in 1:1:6
•       count = 0
•       for j in 1:1:10
•           if M[i, j] > N
•               count += 1
•           end
•       end
•       println(count)
•   end
• end
```



```
6
6
5
4
3
5
```



```

• begin
•     N2 = 7
•     for i in 1:1:6
•         count = 0
•         for j in 1:1:10
•             if M[i, j] == N2
•                 count += 1
•             end
•         end
•         if count == 2
•             println(N2, " в строке ", i, " встречается 2 раза")
•         end
•     end
• end

```

```

7 в строке 1 встречается 2 раза
7 в строке 5 встречается 2 раза
7 в строке 6 встречается 2 раза

```

```

• begin
•     M10 = rand(1:10, 6, 10)
•     begin
•         function sumEqual75(matrix)
•             matrix_new = rand(10)
•             for i in 1:1:10
•                 sum = 0
•                 for j in 1:1:6
•                     sum += matrix[j, i]
•                 end
•                 matrix_new[i] = sum
•             end
•             for i in 1:1:10
•                 for j in i+1:1:10
•                     sum = matrix_new[i] + matrix_new[j]
•                     if sum > 75
•                         println(i, " ", j)
•                     end
•                 end
•             end
•         end
•         sumEqual75(M10)
•     end
• end

```

```

1 3
1 4
2 3
2 4
3 4
3 9
4 7
4 9
4 10

```



## 2.11

11. Вычислите:

$$-\sum_{i=1}^{20} \sum_{j=1}^5 \frac{i^4}{(3+j)},$$
$$-\sum_{i=1}^{20} \sum_{j=1}^5 \frac{i^4}{(3+ij)}.$$

```
• begin
•     sum1 = 0
•         for i in 1:1:20
•             for j in 1:1:5
•                 sum1 += (i^4)/(3+j)
•             end
•         end
•     println(sum1)
• end
```

639215.2833333334

```
• begin
•     sum2 = 0
•         for i in 1:1:20
•             for j in 1:1:5
•                 sum2 += (i^4)/(3+i*j)
•             end
•         end
•     println(sum2)
• end
```

89912.02146097136

## Выводы

В ходе выполнения лабораторной работы успешно удалось освоить применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.