Лабораторная работа №3

Управляющие структуры

Тазаева Анастасия Анатольевна

Содержание

1 Цель работы **5**

2 Задание **6**

3 Выполнение лабораторной работы **7**
3.1 Циклы while и for 7
3.2 Условные выражения 8
3.3 Функции 9
3.4 Сторонние библиотеки (пакеты) в Julia 12
3.5 Самостоятельная работа 13

4 Выводы **19**

Список иллюстраций

3.1 Циклы while и for. Примеры. Часть 1 7
3.2 Циклы while и for. Примеры. Часть 2 8
3.3 Циклы while и for. Примеры. Часть 3 8
3.4 Условные выражения. Примеры 9
3.5 Функции. Примеры. Часть 1 10
3.6 Функции. Примеры. Часть 2 11
3.7 Функции. Примеры. Часть 3 12
3.8 Функции. Примеры. Часть 4 12
3.9 Сторонние библиотеки. Пример 13
3.10 Самостоятельная работа. Задание 1.1 13
3.11 Самостоятельная работа. Задание 1.2 14
3.12 Самостоятельная работа. Задание 1.3 14
3.13 Самостоятельная работа. Задание 2 14
3.14 Самостоятельная работа. Задание 3 14
3.15 Самостоятельная работа. Задание 4 15
3.16 Самостоятельная работа. Задание 5 15
3.17 Самостоятельная работа. Задание 6 15
3.18 Самостоятельная работа. Задание 7.1 16
3.19 Самостоятельная работа. Задание 7.2 16
3.20 Самостоятельная работа. Задание 8.1 16
3.21 Самостоятельная работа. Задание 8.2 17
3.22 Самостоятельная работа. Задание 9 17
3.23 Самостоятельная работа. Задание 10 17
3.24 Самостоятельная работа. Задание 11 18

Список таблиц

1 Цель работы

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

2 Задание

1. Используя Jupyter Lab, повторите примеры из раздела 3.2.
2. Выполните задания для самостоятельной работы (раздел 3.4).

3 Выполнение лабораторной работы

3.1 Циклы while и for

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

```
while <условие>
    <тело цикла>
end
```

Примеры представлены на рис. 3.1 - 3.3 :

Рис. 3.1: Циклы while и for. Примеры. Часть 1

```
#Рассмотренные выше примеры, но с использованием цикла for:
for n in 1:2:10
println(n)
end
```

```
1
3
5
7
9
```

```
myfriends = ["Ted", "Robyn", "Barney", "Lily", "Marshall"]
for friend in myfriends
    println("Hi $friend, it's great to see you!")
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!
```

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

```
5×5 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
```

Рис. 3.2: Циклы while и for. Примеры. Часть 2

```
# Другая реализация этого же примера:
# инициализация массива m x n из нулей:B = fill(0, (m, n))
B = fill(0, (m, n))
for i in 1:m, j in 1:n
    B[i, j] = i + j
end
B
```

```
5×5 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
```

```
# Ещё одна реализация этого же примера:
C = [i + j for i in 1:m, j in 1:n]
C
```

```
5×5 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
```

Рис. 3.3: Циклы while и for. Примеры. Часть 3

3.2 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
```

Примеры представлены на рис. 3.4 :

Рис. 3.4: Условные выражения. Примеры

3.3 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
function f(x)
x^2
end
```

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»:

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

Примеры представлены на рис. 3.5 - 3.8 :

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
sayhi("Nastya")
f(9)
```

```
Hi Nastya, it's great to see you!
81
```

```
# 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
sayhi("Nastya")
f(9)
```

```
Hi Nastya, it's great to see you!
81
```

```
# Наконец, можно объявить выше определённые функции как «анонимные»:
sayhi3 = name -> println("Hi $name, it's great to see you!")
f3 = x -> x^2
sayhi("Nastya")
f(9)
```

```
Hi Nastya, it's great to see you!
81
```

Рис. 3.5: Функции. Примеры. Часть 1

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

```
3-element Vector{Int64}:
 3
 5
 2
```

```
sort!(v)
v
```

```
3-element Vector{Int64}:
 2
 3
 5
```

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

```
3-element Vector{Int64}:
 1
 4
 9
```

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

```
3-element Vector{Int64}:
  1
  8
 27
```

```
f(x) = x^2
broadcast(f, [1, 2, 3])
```

```
3-element Vector{Int64}:
 1
 4
 9
```

Рис. 3.6: Функции. Примеры. Часть 2

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

```
3-element Vector{Int64}:
 1
 4
 9
```

```
# Задаём матрицу A:
A = [i + 3*j for j in 0:2, i in 1:3]
```

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

```
# Вызываем функцию f возведения в квадрат
f(A)
```

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

```
B = f.(A)
```

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

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

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

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

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

Рис. 3.7: Функции. Примеры. Часть 3

```
[44]: broadcast(x -> x + 2 * f(x) / x, A)
```

```
[44]: 3×3 Matrix{Float64}:
        3.0   6.0   9.0
       12.0  15.0  18.0
       21.0  24.0  27.0
```

Рис. 3.8: Функции. Примеры. Часть 4

3.4 Сторонние библиотеки (пакеты) в Julia

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.

Примеры представлены на рис. 3.9 :

Рис. 3.9: Сторонние библиотеки. Пример

3.5 Самостоятельная работа

Выполнение заданий можно просмотреть на рис. 3.10 - 3.24 :

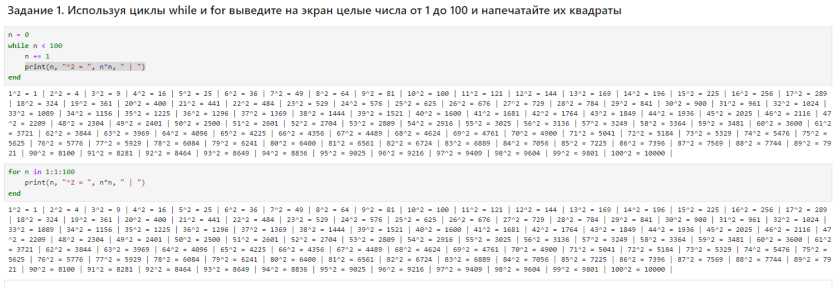

Рис. 3.10: Самостоятельная работа. Задание 1.1

Задание 1. Используя циклы while и for создайте словарь squares, который будет содержать целые числа в качестве ключей и квадраты в качестве их пар-значений

```
squares = Dict()
n = 0
while n<10
    n += 1
    squares[n] = n^n
end
pairs(squares)
```

```
Dict{Any, Any} with 10 entries:
  5  => 25
  4  => 16
  6  => 36
  7  => 49
  2  => 4
  10 => 100
  9  => 81
  8  => 64
  3  => 9
  1  => 1
```

```
for n in 1:10
    squares[n] = n^n
end
pairs(squares)
```

```
Dict{Any, Any} with 10 entries:
  5  => 25
  4  => 16
  6  => 36
  7  => 49
  2  => 4
  10 => 100
  9  => 81
  8  => 64
  3  => 9
  1  => 1
```

Рис. 3.11: Самостоятельная работа. Задание 1.2

Задание 1. Используя циклы while и for создайте массив squares_arr, содержащий квадраты всех чисел от 1 до 100

```
squares_arr=[]
n = 0
while n<100
    n += 1
    push!(squares_arr, [n^n])
end
print(squares_arr)
```

```
Any[[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], [1681], [1764], [1849], [1936], [2025], [2116], [2209], [2304], [2401], [2500], [2601], [2704], [2809], [2916], [3025], [3136], [3249], [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]]
```

```
squares_arr_2 = [i^i for i in 1:100]
print(squares_arr_2)
```

```
[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, 1681, 1764, 1849, 1936, 2025, 2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249, 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]
```

Рис. 3.12: Самостоятельная работа. Задание 1.3

Задание 2. Напишите условный оператор, который печатает число, если число чётное, и строку «нечётное», если число нечётное. Перепишите код, используя тернарный оператор.

```
N=1
if N % 2 == 0
    print(N)
else
    print("нечётное")
end
```

```
нечётное
```

```
N % 2 == 0 ? print(N) : print("нечётное")
```

```
нечётное
```

Рис. 3.13: Самостоятельная работа. Задание 2

Задание 3. Напишите функцию add_one, которая добавляет 1 к своему входу.

```
function add_one(N)
    N = N + 1
    println(N)
end
add_one(10)
```

```
11
```

Рис. 3.14: Самостоятельная работа. Задание 3

Задание 4. Используйте map() или broadcast() для задания матрицы A, каждый элемент которой увеличивается на единицу по сравнению с предыдущим.

```
# Задаём матрицу A:
A = [i + j for j in 0:4, i in 0:4]
```

```
5×5 Matrix{Int64}:
 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
```

```
map(x -> x+1, A)
```

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

Рис. 3.15: Самостоятельная работа. Задание 4

Задание 5. Задайте матрицу A следующего вида (1 1 3; 5 2 6; -2 -1 -3). Найдите A^3. Замените третий столбец матрицы A на сумму второго и третьего столбцов.

```
# ЗАдаем матрицу A
A = [1 1 3; 5 2 6; -2 -1 -3]
```

```
3×3 Matrix{Int64}:
  1   1   3
  5   2   6
 -2  -1  -3
```

```
f(n)=n^3
f(A)
```

```
3×3 Matrix{Int64}:
 0  0  0
 0  0  0
 0  0  0
```

```
for i in 1:3
    A[i,3]=A[i,2]+A[i,1]
end
A
```

```
3×3 Matrix{Int64}:
  1   1   2
  5   2   7
 -2  -1  -3
```

Рис. 3.16: Самостоятельная работа. Задание 5

Задание 6. Создайте матрицу B с элементами $Bi1 = 10$, $Bi2 = -10$, $Bi3 = 10$, $i = 1, 2, \ldots, 15$. Вычислите матрицу $C = B^T*B$.

```
# ЗАдаем матрицу B
B = [j%2==1 ? x=10 : x=-10 for i in 1:15,j in 1:3]
B
```

```
15×3 Matrix{Int64}:
  10  -10   10
  10  -10   10
  10  -10   10
  10  -10   10
  10  -10   10
  10  -10   10
  10  -10   10
  10  -10   10
  10  -10   10
  10  -10   10
  10  -10   10
  10  -10   10
  10  -10   10
  10  -10   10
  10  -10   10
```

```
C=B'*B
```

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

Рис. 3.17: Самостоятельная работа. Задание 6

Задание 7. Создайте матрицу Z размерности 6 × 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 × 6:

```
Z = zeros(6,6)
E = ones(6,6)

Z1 = copy(Z)
for i in 1:6, j in 1:6
    if abs(i-j) == 1
        Z1[i,j]=Z[i,j]+E[i,j]
    end
    Z1[i,j] |> Int64
end
Z1
```

```
6×6 Matrix{Float64}:
 0.0  1.0  0.0  0.0  0.0  0.0
 1.0  0.0  1.0  0.0  0.0  0.0
 0.0  1.0  0.0  1.0  0.0  0.0
 0.0  0.0  1.0  0.0  1.0  0.0
 0.0  0.0  0.0  1.0  0.0  1.0
 0.0  0.0  0.0  0.0  1.0  0.0
```

```
Z2 = copy(Z)
for i in 1:6, j in 1:6
    if abs(i-j) == 2 || i == j
        Z2[i,j]=Z[i,j]+E[i,j]
    end
    Z2[i,j] |> Int64
end
Z2
```

```
6×6 Matrix{Float64}:
 1.0  0.0  1.0  0.0  0.0  0.0
 0.0  1.0  0.0  1.0  0.0  0.0
 1.0  0.0  1.0  0.0  1.0  0.0
 0.0  1.0  0.0  1.0  0.0  1.0
 0.0  0.0  1.0  0.0  1.0  0.0
 0.0  0.0  0.0  1.0  0.0  1.0
```

Рис. 3.18: Самостоятельная работа. Задание 7.1

```
Z3 = copy(Z)
for i in 1:6, j in 1:6
    if i+j == 5 || i+j == 7 || i+j == 9
        Z3[i,j]=Z[i,j]+E[i,j]
    end
    Z3[i,j] |> Int64
end
Z3
```

```
6×6 Matrix{Float64}:
 0.0  0.0  0.0  1.0  0.0  1.0
 0.0  0.0  1.0  0.0  1.0  0.0
 0.0  1.0  0.0  1.0  0.0  1.0
 1.0  0.0  1.0  0.0  1.0  0.0
 0.0  1.0  0.0  1.0  0.0  0.0
 1.0  0.0  1.0  0.0  0.0  0.0
```

```
Z4 = copy(Z)
for i in 1:6, j in 1:6
    if (i+j) %2 == 0
        Z4[i,j]=Z[i,j]+E[i,j]
    end
    Z4[i,j] |> Int64
end
Z4
```

```
6×6 Matrix{Float64}:
 1.0  0.0  1.0  0.0  1.0  0.0
 0.0  1.0  0.0  1.0  0.0  1.0
 1.0  0.0  1.0  0.0  1.0  0.0
 0.0  1.0  0.0  1.0  0.0  1.0
 1.0  0.0  1.0  0.0  1.0  0.0
 0.0  1.0  0.0  1.0  0.0  1.0
```

Рис. 3.19: Самостоятельная работа. Задание 7.2

Задание 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). Подпункт 1 (см.в файле лр)

```
function outer(x,y,operation)
    return [operation(xi, yi) for xi in x, yi in y]
end
```

```
outer (generic function with 1 method)
```

Задание 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). Подпункт 2 (см.в файле лр)

```
A1 = outer(0:4, 0:4, +)
```

```
5×5 Matrix{Int64}:
 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
```

```
A2 = [y == 1 ? x : x^y for x in 0:4, y in 1:5]
```

```
5×5 Matrix{Int64}:
 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
```

```
A3 = outer(0:4, 0:4, (x,y) -> mod(x+y,5))
```

```
5×5 Matrix{Int64}:
 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
```

```
A4 = outer(0:9, 0:9, (x,y) -> mod(x+y,10))
```

```
10×10 Matrix{Int64}:
 0  1  2  3  4  5  6  7  8  9
 1  2  3  4  5  6  7  8  9  0
 2  3  4  5  6  7  8  9  0  1
 3  4  5  6  7  8  9  0  1  2
 4  5  6  7  8  9  0  1  2  3
 5  6  7  8  9  0  1  2  3  4
 6  7  8  9  0  1  2  3  4  5
```

Рис. 3.20: Самостоятельная работа. Задание 8.1

```
A4 = outer(0:9, 0:9, (x,y) -> mod(x+y,10))
```

```
10×10 Matrix{Int64}:
 0  1  2  3  4  5  6  7  8  9
 1  2  3  4  5  6  7  8  9  0
 2  3  4  5  6  7  8  9  0  1
 3  4  5  6  7  8  9  0  1  2
 4  5  6  7  8  9  0  1  2  3
 5  6  7  8  9  0  1  2  3  4
 6  7  8  9  0  1  2  3  4  5
 7  8  9  0  1  2  3  4  5  6
 8  9  0  1  2  3  4  5  6  7
 9  0  1  2  3  4  5  6  7  8
```

```
A5 = outer(0:8, 0:8, (x,y) -> mod(x-y,9))
```

```
9×9 Matrix{Int64}:
 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
```

Рис. 3.21: Самостоятельная работа. Задание 8.2

Задание 9. Решите следующую систему линейных уравнений с 5 неизвестными

```
A = [1 2 3 4 5; 2 1 2 3 4; 3 2 1 2 3; 4 3 2 1 2; 5 4 3 2 1]
B = [7, -1, -3, 5, -6]

X = A \ B
for i in 1:5
    println("X$i = ", X[i], "\n" )
end
```

```
X1 = -3.9166666666666667

X2 = 3.0000000000000013

X3 = 5.0

X4 = -9.500000000000002

X5 = 5.583333333333335
```

Рис. 3.22: Самостоятельная работа. Задание 9

Задание 10. Создайте матрицу M размерности 6 × 10, элементами которой являются целые числа, выбранные случайным образом с повторениями из совокупности 1, 2, ... , 10.

```
M = rand(1:10, 6, 10)
```

```
6×10 Matrix{Int64}:
 10   6  3  8  7   4  5   7  3  4
 10   8  8  6  4  10  1  10  4  5
  9   1  9  7  3   3  3   4  1  5
 10   3  2  3  8  10  4   7  5  5
  7  10  3  7  9   6  5   8  5  3
 10   1  1  8  7   1  1   9  2  7
```

- Найдите число элементов в каждой строке матрицы M, которые больше числа N (например, N = 4).

```
N = 4
res = sum(M .> N, dims=2)
println(res)
```

```
[6; 7; 4; 6; 8; 5;;]
```

- Определите, в каких строках матрицы M число M(например,M = 7) встречается ровно 2 раза?

```
M_value = 7
rows_res = findall(x -> count(==(M_value), x) == 2, eachrow(M))
println(rows_res)
```

```
[1, 5, 6]
```

- Определите все пары столбцов матрицы M, сумма элементов которых больше K (например, K = 75).

```
K = 75
pairs_res = []
for i in 1:size(M, 2)-1
    for j in i+1:size(M, 2)
        if sum(M[:,i] .+ M[:,j]) > K
            push!(pairs_res, (i, j))
        end
    end
end
println(pairs_res)
```

```
Any[(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 8), (1, 9), (1, 10), (4, 5), (4, 8), (5, 8), (6, 8)]
```

Рис. 3.23: Самостоятельная работа. Задание 10

Задание 11. Вычислите

```
sum_1 = sum(i^4 * (3 + j) for i in 1:20 for j in 1:5)
println(sum_1)
```

```
21679980
```

```
sum_2 = sum(i^4 * (3 + i * j) for i in 1:20 for j in 1:5)
println(sum_2)
```

```
195839490
```

Рис. 3.24: Самостоятельная работа. Задание 11

4 Выводы

В ходе лабораторной работы мною было освоено применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.