Отчёт по лабораторной работе №3
Управляющие структуры

Статический анализ данных

Выполнила: Коняева Марина Александровна,
НФИбд-01-21, 1032217044

Содержание

Список иллюстраций

Цели лабораторной работы

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

Теоретическое введение

Julia — высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений. Эффективен также и для написания программ общего назначения.[1]

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

Такие же результаты можно получить при использовании цикла for. Синтаксис for for in end

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

Задачи лабораторной работы

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

Выполнение лабораторной работы

Циклы while и for

1. Изучим информацию о циклах while и for, а также их синтаксис.

2. Повторим примеры с циклом while, а именно использование цикла для формирования элементов массива.

```
# пока n<10 прибавить к n единицу и распечатать значение:
n = 0
while n < 10
n += 1
println(n)
end
```

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

Рис. 1: Формирования элементов массива

3. Повторим примеры с циклом while, а именно при работе со строковыми элементами массива, подставляя имя из массива в заданную строку приветствия и выводя получившуюся конструкцию на экран.

```
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
```

```
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!
```

Рис. 2: while при работе со строковыми элементами массива

4. Повторим примеры с циклом for, а именно использование цикла для формирования элементов массива.

```
for n in 1:2:10
println(n)
end
```

```
1
3
5
7
9
```

Рис. 3: Формирования элементов массива

5. Повторим примеры с циклом for, а именно при работе со строковыми элементами массива, подставляя имя из массива в заданную строку приветствия и выводя получившуюся конструкцию на экран.

```
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!
```

Рис. 4: for при работе со строковыми элементами массива

6. Рассмотрим три примера 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
```

```
# 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
```

```
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
```

```
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. 5: Создания двумерного массива

Условные выражения

7. Изучим информацию об условных выражениях, а также их синтаксис. Синтаксис условных выражений с ключевым словом: if <условие 1> <действие 1> elseif <условие 2> <действие 2> else <действие 3> end

8. Повторим примеры с условными выражениями, а именно пусть для заданного числа требуется вывести слово «Fizz», если делится на 3, «Buzz», если делится на 5, и «FizzBuzz», если делится на 3 и 5.

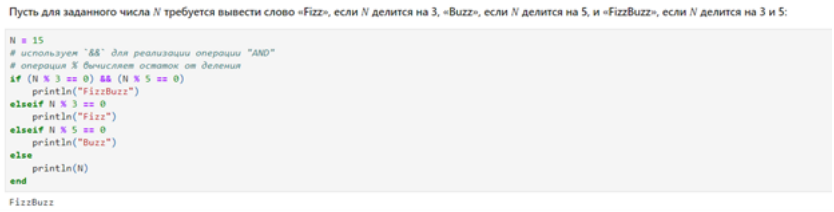

Рис. 6: Примеры с условными выражениями

9. Повторим примеры с условными выражениями с тернарными операторами. 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

Рис. 7: Пример с условными выражениями с тернарными операторами

Функции

10. Изучим информацию о функция и повторим примеры их задания, использования и вызов.

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

# функция возведения в квадрат:
function f(x)
    x^2
end
```

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

```
sayhi("C-3PO")
f(42)
```

```
Hi C-3PO, it's great to see you!
1764
```

Рис. 8: Функции (задание и вызов) 1

11. Повторим примеры с функциями, задание, использования и вызов, повторим альтернативу, можно объявить любую из выше определённых функций в одной строке, а также выполним пример, где можно объявить выше определённые функции как «анонимные».

```
sayhi2(name) = println("Hi $name, it's great to see you!")
f2(x) = x^2
```

```
f2 (generic function with 1 method)
```

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

```
#5 (generic function with 1 method)
```

Рис. 9: Функции (задание и вызов) 2-3

12. Выполним примеры с sort и sort!, по соглашению в Julia функции, сопровождаемые восклицательным знаком, изменяют свое содержимое, а функции без восклицатель-

ного знака не делают этого. 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.

```
# 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
```

Рис. 10: Примеры с sort и sort!

13. Повторим примеры с функцией map, 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.

```
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
```

Рис. 11: Примеры с map

14. Повторим примеры с функцией map, в map можно передать и анонимно заданную, а не именованную функцию.

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

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

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

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

Рис. 12: Примеры с map

15. Выполним примеры с функцией broadcast, 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 вызоваmap, например применение функции f к элементам массива [1, 2, 3]: f(x) = x^2 broadcast(f, [1, 2, 3]) или f.([1, 2, 3])

```
# 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
```

Рис. 13: Примеры с broadcast

16. Выполним примеры с функцией broadcast, точечный синтаксис для broadcast() позволяет записать относительно сложные составные поэлементные выражения в форме, близкой к математической записи.

```
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
```

```
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
```

Рис. 14: Примеры с broadcast

Сторонние библиотеки (пакеты) в Julia

17. Изучим информацию о сторонних библиотеках. 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

18. Добавим необходимые пакеты для дальнейшего использования.

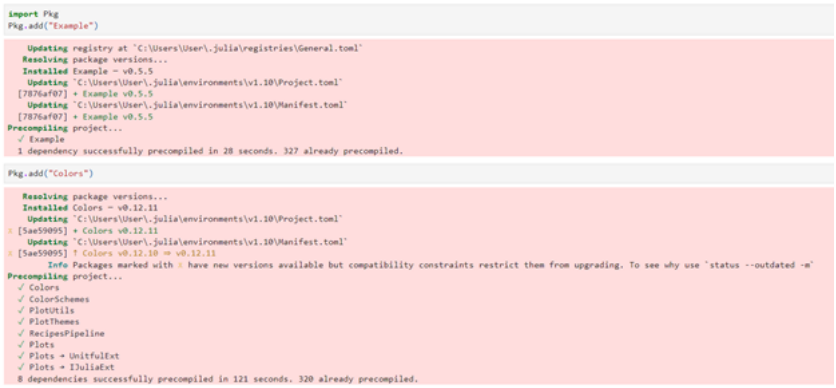

Рис. 15: Сторонние библиотеки (пакеты)

18. Повторим примеры со сторонними библиотеками (пакетами), создадим палитру из 100 разных цветов, а затем определим матрицу 3 × 3 с элементами в форме случайного цвета из палитры, используя функцию rand.

Рис. 16: Пример со сторонними библиотеками

Самостоятельная работа

19. Выполним 1 задание для самостойтельной работы: Используя циклы while и for:

– выведите на экран целые числа от 1 до 100 и напечатайте их квадраты;

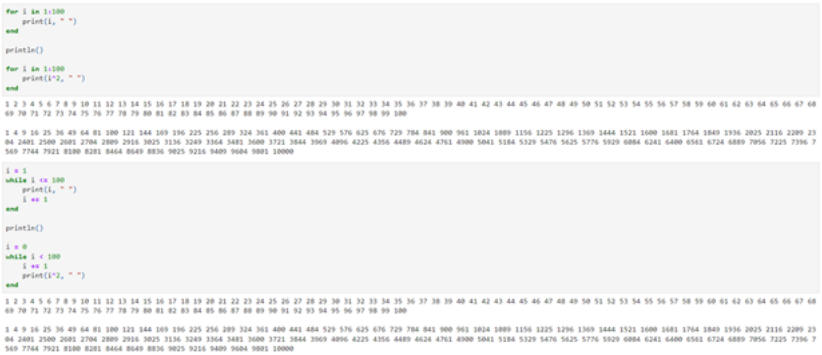

Рис. 17: 1 задания: часть 1

– создайте словарь squares, который будет содержать целые числа в качестве ключей и квадраты в качестве их пар-значений;

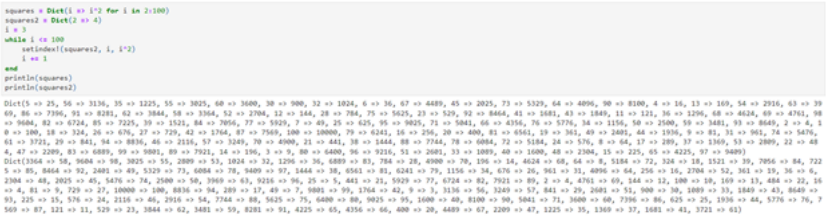

Рис. 18: 1 задания: часть 2

```
squares_arr = [i^2 for i in 1:100]
squares_arr2 = zeros(100)
i = 1
while i <= 100
    squares_arr2[i] = i^2
    i += 1
end
hcat(squares_arr, squares_arr2)
```

```
100×2 Matrix{Float64}:
     1.0      1.0
     4.0      4.0
     9.0      9.0
    16.0     16.0
    25.0     25.0
    36.0     36.0
    49.0     49.0
    64.0     64.0
    81.0     81.0
   100.0    100.0
   121.0    121.0
   144.0    144.0
   169.0    169.0
     ⋮
  7921.0   7921.0
  8100.0   8100.0
  8281.0   8281.0
  8464.0   8464.0
  8649.0   8649.0
  8836.0   8836.0
  9025.0   9025.0
  9216.0   9216.0
  9409.0   9409.0
  9604.0   9604.0
  9801.0   9801.0
 10000.0  10000.0
```

Рис. 19: 1 задания: часть 3

20. Выполним 2 задание для самостойтельной работы: напишите условный оператор, который печатает число, если число чётное, и строку «нечётное», если число нечётное. Перепишите код, используя тернарный оператор.

```
function chet(x)
    if x%2 == 0
        println(x)
    else
        println("Нечетное")
    end
end
chet(25), chet(32)
```

```
Нечетное
32
(nothing, nothing)
```

```
# Перепишите код, используя тернарный оператор.
function chet2(x)
    x%2 == 0 ? println(x) : println("Нечетное")
end
chet(25), chet(32)
```

```
Нечетное
32
(nothing, nothing)
```

Рис. 20: Выполнение 2 задания

21. Выполним 3 задание для самостойтельной работы: напишите функцию add_one, которая добавляет 1 к своему входу.

```
function add_one(x)
    return x+1
end
add_one(5)
```

```
6
```

Рис. 21: Выполнение 3 задания

22. Выполним 4 задание для самостойтельной работы: используйте map() или broadcast() для задания матрицы □, каждый элемент которой увеличивается на единицу по сравнению с предыдущим.

```
A = zeros(Int64, 3, 5)
t = map!(x -> x+1, A, 0:14); display(A)
t = broadcast(add_one, reshape(0:14, 3, 5))
```

```
3×5 Matrix{Int64}:
 1  4  7  10  13
 2  5  8  11  14
 3  6  9  12  15
3×5 Matrix{Int64}:
 1  4  7  10  13
 2  5  8  11  14
 3  6  9  12  15
```

Рис. 22: Задание 4

23. Выполним 5 задание для самостойтельной работы: Задайте матрицу A следующего вида:

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

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

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

Рис. 23: Задание 5

- Найдите A^3

```
display(A^2)
display(A^3)
```

```
3×3 Matrix{Int64}:
  0   0   0
  3   3   9
 -1  -1  -3
3×3 Matrix{Int64}:
 0  0  0
 0  0  0
 0  0  0
```

Рис. 24: Задание 5: часть 1

- Замените третий столбец матрицы A на сумму 2-го и 3-го столбцов

```
A[:, 3] = A[:, 2] + A[:, 3]
A
```

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

Рис. 25: Задание 5: часть 2

24. Выполним 6 задание для самостойтельной работы: создайте матрицу B с элементами $B_{i1} = 10, B_{i2} = -10, B_{i3} = 10, \quad i = 1, 2, ..., 15.$

```
B = fill(10, (15,3))
B[:, 2] = -B[:, 2]
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
```

Рис. 26: Задание 6

- Вычислите матрицу $C = B^T B$.

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

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

Рис. 27: Задание 6

25. Выполним 7 задание для самостойтельной работы: создайте матрицу Z размерности 6×6, все элементы которой равны нулю, и матрицу E, все элементы которой равны

1.

```
Z = zeros(Int64, 6, 6); display(Z)
E = ones(Int64, 6, 6)
```

```
6×6 Matrix{Int64}:
 0  0  0  0  0  0
 0  0  0  0  0  0
 0  0  0  0  0  0
 0  0  0  0  0  0
 0  0  0  0  0  0
 0  0  0  0  0  0
6×6 Matrix{Int64}:
 1  1  1  1  1  1
 1  1  1  1  1  1
 1  1  1  1  1  1
 1  1  1  1  1  1
 1  1  1  1  1  1
 1  1  1  1  1  1
```

Рис. 28: Задание 7

- Используя цикл `while` или `for` и закономерности расположения элементов, создайте следующие матрицы размерности 6×6:

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

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

```
Z1, Z2, Z3, Z4 = zeros(Int64, 6, 6), zeros(Int64, 6, 6), zeros(Int64, 6, 6), zeros(Int64, 6, 6)
for i in 1:6, j in 1:6
    if i == j+1 || i == j-1
        Z1[i,j] = 1
    end
    if i == j || i == j-2 || i == j+2
        Z2[i,j] = 1
    end
    if i == 7-j || i == 5-j || i == 9-j
        Z3[i,j]= 1
    end
    if (i+j)%2 == 0
        Z4[i,j] = 1
    end
end
display(Z1); display(Z2); display(Z3); display(Z4)
```

```
6×6 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
6×6 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
6×6 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
6×6 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
```

Рис. 29: Задание 7: часть 1

26. Выполним 8 задание для самостойтельной работы: 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 результирующей матрицы C имеют вид $C_{ij} = \sum_{k=1}^{M} A_{ik} B_{kj}$ (или в тензорном виде $C_i^j = \sum_{k=1}^{M} A_k^i B_j^k$)

```
function outer(A, B, operation)
    if size(A)[2] != size(B)[1]
        println("Size incompatability!")
        return
    end
    answ = zeros(size(A)[1], size(B)[2])
    for i in 1:size(A)[1], j in 1:size(B)[2]
        answ[i,j] = sum(operation(A[i,k], B[k,j]) for k in 1:size(A)[2])
    end
    return answ
end
mtrx1, mtrx2 = rand(1:10, 4, 6), rand(1:10, 6, 3); display(mtrx1); display(mtrx2)
display(outer(mtrx1, mtrx1', *))
display(outer(mtrx2', mtrx2, +))
display(outer(mtrx1, mtrx2, %))
display(outer(mtrx1, mtrx2, /))
display(outer(mtrx1, mtrx2, -))
display(outer(mtrx1, mtrx2, div))
```

Рис. 30: Задание 8: часть 1

```
4×6 Matrix{Int64}:
  7  10  10  9  7  5
  5   6  10  9  1  5
  4   2   8  8  8  7
 10   3   5  5  8  6
6×3 Matrix{Int64}:
  8   4  5
  5  10  3
  4   6  6
 10   8  4
  5   6  7
  9  10  4
4×4 Matrix{Float64}:
 404.0  308.0  291.0  281.0
 308.0  268.0  227.0  201.0
 291.0  227.0  261.0  232.0
 281.0  201.0  232.0  259.0
3×3 Matrix{Float64}:
 82.0  85.0  70.0
 85.0  88.0  73.0
 70.0  73.0  58.0
4×3 Matrix{Float64}:
 25.0  14.0   9.0
 23.0  18.0   7.0
 24.0  13.0  12.0
 20.0  23.0   9.0
4×3 Matrix{Float64}:
 8.23056  7.20833  10.9
 5.98056  5.30833   8.30952
 6.07778  5.56667   7.69286
 5.86667  6.19167   7.72619
4×3 Matrix{Float64}:
  7.0   4.0  19.0
 -5.0  -8.0   7.0
 -4.0  -7.0   8.0
 -4.0  -7.0   8.0
4×3 Matrix{Float64}:
 5.0  5.0  9.0
 3.0  3.0  7.0
 3.0  4.0  5.0
 3.0  3.0  6.0
```

Рис. 31: Задание 8: часть 1 (вывод)

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

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

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

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

В каждом случае ваше решение должно быть легко обобщаемым на случай создания матриц большей размерности, но той же структуры.

```
n = 5
A1 = Int64.(outer(hcat([[i//n for i in 0:n-1] for j in 1:n]...), hcat([[i//n for i in 0:n-1] for j in 1:n]...)', +))
display(A1)
t2 = hcat([[j-1 for i in 1:n] for j in 1:n]...)'
A2 = Int64.(outer(t2, hcat([[i==j ? i : 1 for i in 1:n] for j in 1:n]...), ^))-hcat([[(n-1)*i for i in 0:n-1] for j in 1:n]...)
display(A2)
#=
Матрицы A3 и A4 аналогичны по своей структуре первой матрице, единственное,
чем они отличаются - каждый элемент теперь представлен остатком от деления
его предыдущего значения на сторону матрицы
=#
t3 = outer(hcat([[i//n for i in 0:n-1] for j in 1:n]...), hcat([[i//n for i in 0:n-1] for j in 1:n]...)', +)
A3 = Int64.(outer(t3, hcat([[i==j ? n/1 : 1.0 for i in 1:n] for j in 1:n]...), mod))
display(A3)
n = 10
t4 = outer(hcat([[i//n for i in 0:n-1] for j in 1:n]...), hcat([[i//n for i in 0:n-1] for j in 1:n]...)', +)
A4 = Int64.(outer(t4, hcat([[i==j ? n/1 : 1.0 for i in 1:n] for j in 1:n]...), mod)) # Остаток от деления на n
display(A4)
n = 9
t5 = outer(hcat([[i==j-1 ? j-1 : 0 for i in 0:n-1] for j in 1:n]...), hcat([[i==j-1 ? (n-i) : 0 for i in 0:n-1] for j in 1:n]...)', +)
A5 = Int64.(outer(round.(t5), hcat([[i==j ? n/1 : 1.0 for i in 1:n] for j in 1:n]...), mod)) # Остаток от деления на n
display(A5)
```

Рис. 32: Задание 8: часть 2

```
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
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
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
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
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. 33: Задание 8: часть 2 (вывод)

27. Выполним 9 задание для самостойтельной работы: 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.

Решение будет осуществлено методом Гаусса.

```
function gauss_method(mtrx, vec)
    if size(mtrx)[1] != size(mtrx)[2]
        println("Size incompatability!"), size(mtrx)[1] >= size(mtrx)[2] ? "More" : "Less", "equations than value"
        return
    end
    if size(mtrx)[1] != size(vec)[1]
        println("Size incompatability!"), size(mtrx)[1] >= size(vec)[1] ? "More" : "Less", "answers than equations"
        return
    end
    n = size(mtrx)[1]
    max_el, max_row = 0, 0
    for i in 1:n
        max_el = abs(mtrx[i,i])
        max_row = i
        for k in i:n
            if abs(mtrx[k,k]) > max_el
                max_el = abs(mtrx[k,k])
                max_row = k
            end
        end
        mtrx[i,:], mtrx[max_row,:] = mtrx[max_row,:], mtrx[i,:]
        vec[i], vec[max_row] = vec[max_row], vec[i]

        for k in i+1:n
            c = -mtrx[k,i] / mtrx[i,i]
            mtrx[k,:] = mtrx[k,:].+(c.*mtrx[i,:])
            mtrx[k,i] = 0
            vec[k] += c*vec[i]
        end
    end
    answ = zeros(n)
    for i in n:-1:1
        answ[i] = vec[i] /  mtrx[i,i]
        for k in i-1:-1:1
            vec[k] -= mtrx[k,i]* answ[i]
        end
    end
    return answ
end
```

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

Рис. 34: Задание 9: метод Гаусса

```
m = 5
A = hcat([[abs(i-j)+1 for j in 1:m] for i in 1:m]...); display(A)
y = [7, -1, -3, 5, 17]; display(y)
x = gauss_method(A,y)
```

```
5×5 Matrix{Int64}:
 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
5-element Vector{Int64}:
  7
 -1
 -3
  5
 17
5-element Vector{Float64}:
 -2.0
  3.0
  5.0
  2.0
 -4.0
```

Рис. 35: Задание 9: метод Гаусса (решение)

28. Выполним 10 задание для самостойтельной работы: создайте матрицу M размерности 6×10, элементами которой являются целые числа, выбранные случайным образом с повторениями из совокупности $1, 2, \ldots, 10$.

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

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

Рис. 36: Задание 10

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

```
N = 4
[size(findall(x -> x>N, M[i,:]))[1] for i in 1:6]
```

```
6-element Vector{Int64}:
  6
  5
 10
  9
  4
  6
```

Рис. 37: Задание 10: часть 1

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

```
T = 7
findall(x -> x==2, [size(findall(x -> x==T, M[i,:]))[1] for i in 1:6])
```

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

Рис. 38: Задание 10: часть 2

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

```
K = 75
findall(x -> x>K, hcat([[i==j ? -1 : sum(M[:,i])+sum(M[:,j]) for j in 1:6] for i in 1:6]...))
```

```
10-element Vector{CartesianIndex{2}}:
 CartesianIndex(2, 1)
 CartesianIndex(5, 1)
 CartesianIndex(1, 2)
 CartesianIndex(5, 2)
 CartesianIndex(5, 4)
 CartesianIndex(1, 5)
 CartesianIndex(2, 5)
 CartesianIndex(4, 5)
 CartesianIndex(6, 5)
 CartesianIndex(5, 6)
```

Рис. 39: Задание 10: часть 3

29. Выполним 11 задание для самостойтельной работы: вычислите:

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

```
sum([sum([i^4/(3+j)] for j in 1:5) for i in 1:20])
```

```
1-element Vector{Float64}:
 639215.2833333334
```

Рис. 40: Задание 11: часть 1

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

```
sum([sum([i^4/(3+i*j)] for j in 1:5) for i in 1:20])
```

```
1-element Vector{Float64}:
 89912.02146097136
```

Рис. 41: Задание 11: часть 1

Выводы по проделанной работе

Вывод

В результате выполнения работы мы освоили применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.

Были записаны скринкасты выполнения , создания отчета, презентации и защиты лабораторной работы.

Список литературы

- Julia: https://ru.wikipedia.org/wiki/Julia
- https://julialang.org/packages/
- https://juliahub.com/ui/Home
- https://juliaobserver.com/
- https://github.com/svaksha/Julia.jl