Отчёт по лабораторной работе №3

Управляющие структуры

Гань Чжаолун

Цель работы

Основная цель работы — освоить применение циклов, функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.

Выполнение лабораторной работы

Циклы while и for

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

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

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

```
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!
```

```
# рассмотрим те же самые примеры с циклом 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!
```

Рисунок 1. Циклы while и for -1

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
```

```
# Другая реализация этого же примера:
# 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
```

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

Рисунок 2. Циклы while и for -2

Условные выражения

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

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.

```
# используем `&&` для реализации операции "AND"
# операция % вычисляет остаток от деления
# допустим N равно 5
N = 5

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
```

```
Buzz
```

```
# пример использования тернарного оператора
x = 5
y = 10
(x > y) ? x : y
```

```
10
```

Рисунок 3. Условные выражения

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
sayhi("Nastya")
```

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

```
# функция возведения в квадрат:
function f(x)
    x^2
end
f(4)
```

```
16
```

```
# в качестве альтернативы
sayhi2(name) = println("Hi $name, it's great to see you!")
sayhi2("Sasha")
```

```
Hi Sasha, it's great to see you!
```

```
f2(x) = x^2
f2(42)
```

```
1764
```

Рисунок 4. Функции

Также можно объявить вышеперечисленные функции как анонимные

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

```
Hi Bernard, it's great to see you!
```

```
f3 = x -> x^2
f3(34)
```

```
1156
```

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

```
# 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
```

Рисунок 5. объявить функции как анонимные -1

> 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.

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

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

```
# работа map с анонимной функцией
map(x -> x^3, [1, 2, 3])
```

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

Рисунок 6. объявить функции как анонимные -2

Функция broadcast() будет пытаться привести все объекты к общему измерению, map() будет напрямую применять данную функцию поэлементно.

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

```
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
```

```
# аналогичная запись
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
```

Рисунок 7. объявить функции как анонимные -3

Сторонние библиотеки в Julia

```
import Pkg
Pkg.add("Example")
```

```
    Updating registry at `C:\Users\GanZL\.julia\registries\General.toml`
   Resolving package versions...
   Installed Example ─ v0.5.5
    Updating `C:\Users\GanZL\.julia\environments\v1.11\Project.toml`
  [7876af07] + Example v0.5.5
    Updating `C:\Users\GanZL\.julia\environments\v1.11\Manifest.toml`
  [7876af07] + Example v0.5.5
Precompiling project...
   1600.0 ms  ✓ Example
  1 dependency successfully precompiled in 3 seconds. 45 already precompiled.
```

```
Pkg.add("Colors")
using Colors
```

```
   Resolving package versions...
   Installed FixedPointNumbers ─ v0.8.5
   Installed Reexport ────────── v1.2.2
   Installed ColorTypes ──────── v0.12.0
   Installed Colors ──────────── v0.13.0
   Installed Statistics ──────── v1.11.1
    Updating `C:\Users\GanZL\.julia\environments\v1.11\Project.toml`
  [5ae59095] + Colors v0.13.0
    Updating `C:\Users\GanZL\.julia\environments\v1.11\Manifest.toml`
  [3da002f7] + ColorTypes v0.12.0
  [5ae59095] + Colors v0.13.0
  [53c48c17] + FixedPointNumbers v0.8.5
  [189a3867] + Reexport v1.2.2
  [10745b16] + Statistics v1.11.1
  [37e2e46d] + LinearAlgebra v1.11.0
  [e66e0078] + CompilerSupportLibraries_jll v1.1.1+0
  [4536629a] + OpenBLAS_jll v0.3.27+1
  [8e850b90] + libblastrampoline_jll v5.11.0+0
Precompiling project...
    386.8 ms  ✓ Reexport
    499.1 ms  ✓ Statistics
   1443.3 ms  ✓ FixedPointNumbers
   1060.9 ms  ✓ ColorTypes
    494.8 ms  ✓ ColorTypes → StyledStringsExt
   2448.9 ms  ✓ Colors
  6 dependencies successfully precompiled in 6 seconds. 47 already precompiled.
```

Рисунок 8. Сторонние библиотеки в Julia -1

```
palette = distinguishable_colors(100)
```

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

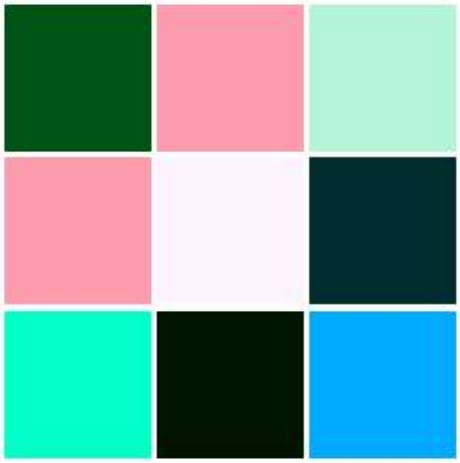

Рисунок 9. Сторонние библиотеки в Julia -2

Задания для самостоятельного выполнения

1. Используя циклы while и for:
   - выведите на экран целые числа от 1 до 100 и напечатайте их квадраты;
   - создайте словарь squares, который будет содержать целые числа в качестве ключей и квадраты в качестве их пар-значений;
   - создайте массив squares_arr, содержащий квадраты всех чисел от 1 до 100.

```
for i in 1:1:100
    println(i)
end
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

Рисунок 10. Код и результат Задания 1-1

Сначала я вывел числа от 1 до 100, применив цикл `for`. Это позволило последовательно пройти все значения от 1 до 100 и напечатать их, что является простым и эффективным способом организации итераций по фиксированному диапазону.

```julia
i = 1
while i <= 100
    println(i^2)
    i += 1
end
```

```
1
4
9
16
25
36
49
64
81
100
121
144
169
196
225
256
289
324
361
```

Рисунок 11. Код и результат Задания 1-2

Затем с помощью цикла `while` я вывел квадраты всех чисел от 1 до 100. Цикл `while` позволил мне возводить числа в квадрат, пока условие (`i <= 100`) выполнялось. В результате, были напечатаны квадраты всех чисел от 1 до 100.

```julia
# инициализируем словарь
squares = Dict{Int64, Int64}()
# занесем значения в словарь
for i in 1:1:100
    push!(squares, i => i^2)
end
pairs(squares)
```

```
32 => 1024
6  => 36
67 => 4489
45 => 2025
73 => 5329
64 => 4096
90 => 8100
4  => 16
13 => 169
54 => 2916
63 => 3969
86 => 7396
91 => 8281
62 => 3844
58 => 3364
52 => 2704
12 => 144
28 => 784
75 => 5625
⋮  => ⋮
```

Рисунок 12. Код и результат Задания 1-3

Я создал словарь `squares`, используя цикл `for`, чтобы добавить в словарь числа в качестве ключей и их квадраты как значения. В итоге словарь `squares` содержал все числа от 1 до 100 и соответствующие им квадраты, что позволило иметь прямую связь между числом и его квадратом для дальнейшего использования.

```julia
# создадим массив из целых чисел от 1 до 100
numbers = []
for i in 1:1:100
    append!(numbers, i)
end

# возведем в квадрат какое число из массива numbers
squares_arr = []
i = 1
while i <= length(numbers)
    append!(squares_arr, numbers[i]^2)
    i += 1
end
squares_arr
```

```
100-element Vector{Any}:
     1
     4
     9
    16
    25
    36
    49
    64
    81
   100
   121
   144
   169
     ⋮
  7921
  8100
  8281
  8464
```

Рисунок 13. Код и результат Задания 1-4

Наконец, я создал массив `squares_arr`, содержащий квадраты чисел от 1 до 100. Сначала я инициализировал массив чисел от 1 до 100, а затем с помощью цикла `while` возводил каждое число в квадрат и добавлял в новый массив `squares_arr`. В результате массив `squares_arr` включал все квадраты чисел, что упрощает доступ к ним при необходимости.

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

```julia
N = 5
if N % 2 == 0
    println("Четное число")
else
    println("Нечетное число")
end
```

```
Нечетное число
```

```julia
N = 4
(N % 2 == 0) ? println("Четное число") : println("Нечетное число")
```

```
Четное число
```

Рисунок 14. Код и результат Задания 2

В первом шаге я написал условный оператор `if-else` для проверки четности числа. Здесь я проверяю, делится ли число `N` на 2 без остатка. Если остаток равен нулю (`N % 2 == 0`), то программа выводит `"Четное число"`, иначе — `"Нечетное число"`.

На втором шаге я переписал тот же условный оператор, используя тернарный оператор, Тернарный оператор в Julia имеет следующую структуру: `(условие) ? значение_если_истинно : значение_если_ложно`. Здесь я проверяю, является ли число `N` четным, и в зависимости от результата либо вывожу `"Четное число"`, либо `"Нечетное число"`.

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

```
function add_one(X)
    X + 1
end
add_one(5)
```

```
6
```

Рисунок 15. Код и результат Задания 3

- Я определил функцию `add_one`, которая принимает один аргумент `X`.
- В теле функции я добавил `1` к значению `X` и возвращаю результат. Поскольку Julia автоматически возвращает последнее вычисленное выражение в функции, явное использование `return` не требуется.
- Я вызвал функцию `add_one(5)`, чтобы протестировать ее работу. В данном случае значение `5` передается в функцию как аргумент `X`.
- Вызов функции `add_one(5)` возвращает значение `6`, так как к аргументу `5` добавляется `1`.

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

```
X = fill(1, 4 * 4)
Y = collect(0:(length(X) - 1))
X = reshape(map(+, X, Y), (4, 4))
```

```
4×4 Matrix{Int64}:
 1  5   9  13
 2  6  10  14
 3  7  11  15
 4  8  12  16
```

Рисунок 16. Код и результат Задания 4

- `X = fill(1, 4 * 4)`: Я создал одномерный массив из 16 элементов, каждый из которых равен `1`.
- `Y = collect(0:(length(X) - 1))`: Я создал массив `Y`, который содержит значения от `0` до `15`. Эти значения будут добавляться к каждому элементу матрицы `X`, чтобы получить последовательность, увеличивающуюся на единицу с каждым шагом.
- `X = reshape(..., (4, 4))`: Я преобразовал полученный одномерный массив в матрицу размером `4x4`. В итоге получена матрица размером `4x4`, в которой каждый элемент увеличивается на единицу по сравнению с предыдущим.

5. Задайте 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
```

```
# Возведение в степень 3
A^3
```

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

```
# заменим 3 столбец матрицы A на сумму второго и третьего столбца
for i in 7:1:9
    A[i] += A[i-3]
end
A
```

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

Рисунок 17. Код и результат Задания 5

- создал матрицу `A` размером `3x3`, используя синтаксис в Julia: `A = [1 1 3; 5 2 6; -2 -1 -3]`.
- применил оператор `^` для возведения матрицы в степень 3 (`A^3`), что означает умножение матрицы на саму себя трижды.
- заменил третий столбец матрицы `A` на сумму второго и третьего столбцов с использованием цикла `for`:
  - В цикле `for i in 7:1:9` я перебираю индексы элементов третьего столбца, начиная с 7-го элемента и заканчивая 9-м.
  - Каждому элементу третьего столбца (`A[i]`) я добавляю соответствующий элемент второго столбца (`A[i-3]`), чтобы получить нужный результат.

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

```
# объявим матрицу B
B = Array{Int32, 2}(undef, 15, 3)

# заполним матрицу в соответствии с заданием
for i in 1:1:15
    B[i, 1] = 10
    B[i, 2] = -10
    B[i, 3] = 10
end
B
```

```
15×3 Matrix{Int32}:
 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
```

```
# вычислим матрицу
C = (B')*B
C
```

```
3×3 Matrix{Int32}:
  1500  -1500   1500
 -1500   1500  -1500
  1500  -1500   1500
```

Рисунок 18. Код и результат Задания 6

- Я объявил матрицу `B` размером `15x3` с использованием `Array{Int32, 2}(undef, 15, 3)`, что означает создание двумерного массива типа `Int32` с размерами `15` строк и `3` столбца.

  Использование `undef` позволяет сначала выделить память для массива, а затем заполнить его необходимыми значениями.

- Я использовал цикл `for i in 1:1:15` для перебора всех строк матрицы.

  Заполнил все элементы в каждой строке значениями: `10`, `-10` и `10` для первого, второго и третьего столбцов соответственно.

- Я вычислил матрицу `C` как произведение транспонированной матрицы `B` и самой матрицы `B`: `C = (B') * B`.

  Здесь `B'` обозначает транспонирование матрицы `B`, а затем выполняется матричное умножение.

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(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
```

```
E = ones(Int64, 6, 6)
```

```
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
```

Рисунок 19. Код и результат Задания 7-1

```
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
```

```
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
```

```
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
```

```
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
```

Рисунок 20. Код и результат Задания 7-2

- z1: Создана с элементами равными `1`, расположенными по соседству с главной диагональю. Цикл добавляет единицы в элементы, которые находятся рядом с главной диагональю.

- z2: Создана с элементами главной диагонали равными `1`, а также с добавлением единиц через две позиции от главной диагонали.

```
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
```

```
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
```

```
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
```

```
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
```

Рисунок 21. Код и результат Задания 7-3

- z3: Создана с зеркальным отражением элементов относительно главной диагонали и с дополнительными значениями, отстоящими на две позиции.
- z4: Создана с элементами главной диагонали, а также с добавлением единиц через две и четыре позиции от диагонали. Это обеспечивает равномерное распределение `1` вдоль диагонали и по сторонам.

8. Напишите свою функцию, аналогичную функции outer() языка R. Функция должна иметь следующий интерфейс: outer(x,y,operation).

```
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
```

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

```
# проверим на данной матрице работу функции
X = [[1 1 3]; [5 2 6]; [-2 -1 -3]]
```

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

```
X * X
```

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

```
outer(X, X, *)
```

```
3×3 Matrix{Float64}:
  0.0   0.0   0.0
  3.0   3.0   9.0
 -1.0  -1.0  -3.0
```

Рисунок 22. Код и результат Задания 8-1

1. Создание функции `outer`:
    - Функция `outer` принимает три аргумента: `x`, `y`, и `operation`.
    - Сначала проверяется, являются ли входные данные векторами (`ndims(x) == 1`). Если это так, то вектора преобразуются в матрицы с использованием функции `reshape`.
    - Создается пустая результирующая матрица `c` размером `(size(x)[1], size(y)[2])`, которая будет содержать результаты.
    - Далее используется вложенный цикл `for`, который проходит по всем элементам и выполняет операцию, переданную в аргументе `operation`.
2. Проверка работы функции:
    - В качестве проверки, я использовал функцию `outer` для произведения матрицы `X` самой на себя и убедился, что результат эквивалентен обычному матричному умножению (`X * X`).

```
# Построим первую матрицу
A1 = outer(collect(0:4), collect(0:4)', +)
```

```
5×5 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
```

```
# построим вторую матрицу
A2 = outer(collect(0:4), collect(1:5)', ^)
```

```
5×5 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
```

```
# построим третью матрицу
# возьмем остаток от деления на 5
A3 = outer(collect(0:4), collect(0:4)', +).%5
```

```
5×5 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
```

Рисунок 23. Код и результат Задания 8-2

- A1: Матрица, созданная на основе векторов `0:4` и операции сложения. Эта операция фактически создает таблицу сложения для чисел от 0 до 4.
- A2: Матрица, созданная с использованием возведения в степень для элементов векторов `0:4` и `1:5`.
- A3: Сложение элементов с последующим взятием остатка от деления на `5`. Это создает матрицу, где значения периодически повторяются с шагом `5`.

```
# построим четвертую матрицу
# возьмем остаток от деления на 10
A4 = outer(collect(0:9), collect(0: 9)', +).%10
```

```
10×10 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
```

```
# построим пятую матрицу
# возьмем остаток от деления на 9
A5 = outer(collect(0:8), collect(-9:-1)', -).%9
```

```
9×9 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
```

Рисунок 24. Код и результат Задания 8-3

- A4: Матрица размером `10x10`, элементы которой — это сумма чисел с последующим взятием остатка от деления на `10`.
- A5: Используется операция вычитания, затем берется остаток от деления на `9` между значениями из диапазонов `0:8` и `-9:-1`.

9. Решите следующую систему линейных уравнений с 5 неизвестными.

```
# зададим 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))


# обозначим ответы
answers = [7; -1; -3; 5; 17]
```

```
Int32[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
```

```
# с помощью обратной матрицы получим решение данной матрицы
array_system_inv = inv(array_system)
round.(Int, array_system_inv)
x_n = round.(Int, array_system_inv * answers)
```

```
5-element Vector{Int64}:
 -2
  3
  5
  2
 -4
```

Рисунок 25. Код и результат Задания 9

1. Задание коэффициентов матрицы `A`:
    - Я задал матрицу `array_system` размером `5x5`, используя цикл `for`. Каждый элемент `A[i, j]` был вычислен как `1 + abs(i - j)`, что создает матрицу со специфической симметричной структурой.
    - Использование цикла позволяет легко обобщить решение на матрицы большей размерности (например, `nxn`), сохраняя ту же закономерность при формировании элементов.
2. Задание вектора ответов:
    - Вектор ответов (`answers`) представляет собой значения правой части системы уравнений: `[7, -1, -3, 5, 17]`.
3. Решение системы с помощью обратной матрицы:
    - Я вычислил обратную матрицу `array_system_inv` с помощью функции `inv()`.
    - Далее для нахождения решения `x` я умножил обратную матрицу на вектор ответов (`x_n = array_system_inv * answers`).
    - Значения были округлены с использованием функции `round.(Int, ...)`, чтобы получить целые числа в результате.

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

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

```
N = 6
for i in 1:1:6
    count = 0
    for j in 1:1:10
        if M[i, j] > N
            count += 1
        end
    end
    println("Количество элементов больше ", N, " в строке ", i, " ровно ", count)
end
```

```
Количество элементов больше 6 в строке 1 ровно 4
Количество элементов больше 6 в строке 2 ровно 2
Количество элементов больше 6 в строке 3 ровно 2
Количество элементов больше 6 в строке 4 ровно 2
Количество элементов больше 6 в строке 5 ровно 3
Количество элементов больше 6 в строке 6 ровно 5
```

Рисунок 26. Код и результат Задания 10-1

1. Создание матрицы `M`:
    - Я создал матрицу `M` размером `6x10`, заполненную случайными целыми числами от `1` до `10`, используя функцию `rand(1:10, 6, 10)`.
2. Определение количества элементов больше `N` в каждой строке:
    - Использовал цикл `for`, чтобы пройти по каждой строке матрицы и посчитать количество элементов, которые больше заданного числа `N` (в данном случае `N = 6`).
    - Результат выводится для каждой строки, показывая количество элементов, которые превышают `N`.

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

```
M_ = 7

for i in 1:1:6
    count = 0
    for j in 1:1:10
        if M[i, j] == M_
            count += 1
        end
    end
    if count == 2
        println("Число ", M_, " встречается в строке ", i, " ровно 2 раза")
    end
end
```

```
Число 7 встречается в строке 1 ровно 2 раза
Число 7 встречается в строке 3 ровно 2 раза
Число 7 встречается в строке 6 ровно 2 раза
```

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

```
function sum_75(matrix, K)
    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
    # теперь просуммируем все значения matrix_new попарно
    for i in 1:1:10
        for j in i+1:1:10
            sum = matrix_new[i] + matrix_new[j]
            if sum > K
                println("Столбцы - ", i, " и ", j)
            end
        end
    end
end
```

```
sum_75 (generic function with 1 method)
```

```
sum_75(M, 75)
```

```
Столбцы - 5 и 6
```

Рисунок 27. Код и результат Задания 10-2

1. Определение строк, где число `M_` встречается ровно два раза:
    - В следующем цикле я посчитал, сколько раз число `M_` (в данном случае `M_ = 7`) встречается в каждой строке матрицы `M`.
    - Если число `7` встречается ровно два раза, выводится номер этой строки.
2. Определение пар столбцов, сумма элементов которых больше `K`:
    - Я создал функцию `sum_75(matrix, K)`, которая сначала вычисляет сумму элементов каждого столбца матрицы.
    - Затем эти суммы попарно суммируются, и проверяется, превышает ли эта сумма заданное значение `K` (в данном случае `K = 75`).
    - Выводятся все пары столбцов, сумма которых больше `K`.

11. Вычислите:

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

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

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

```
639215.2833333334
```

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

```
89912.02146097136
```

Рисунок 28. Код и результат Задания 11

1. Первая сумма:
    - Я использовал два вложенных цикла для вычисления этой суммы.
    - Внешний цикл проходит по значениям `i` от `1` до `20`, а внутренний — по значениям `j` от `1` до `5`.
    - Для каждой комбинации `i` и `j` вычисляется дробь `(i^4) / (3 + j)` и добавляется к переменной `first_sum`.
2. Вторая сумма:
    - Аналогично первой сумме, я использовал вложенные циклы для вычисления.
    - Для каждой комбинации `i` и `j` я вычисляю дробь `(i^4) / (3 + i * j)` и добавляю к переменной `second_sum`.

Вывод

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