

Лабораторная Работа №3

Управляющие структуры

Козлов В.П.

Российский университет дружбы народов им. Патриса Лумумбы, Москва, Россия

- Козлов Всеволод Павлович
- НФИбд-02-22
- Российский университет дружбы народов
- [1132226428@pfur.ru]

Выполнение лабораторной работы

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

1. Используя Jupyter Lab, повторите примеры из отчета.
2. Выполните задания для самостоятельной работы.

Цикл while

```
[1]: # Пример 1: Цикл while для чисел от 1 до 10
println("Цикл while для чисел от 1 до 10:")
n = 0
while n < 10
    n += 1
    println(n)
end
```

Цикл while для чисел от 1 до 10:

1
2
3
4
5
6
7
8
9
10

```
[2]: # Пример 2: Цикл while для работы с массивом строк
println("\nЦикл while для приветствий:")
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
```

Цикл while для приветствий:

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!

Цикл for

```
[3]: # Пример 3: Цикл for с шагом 2
println("\nЦикл for с шагом 2:")
for n in 1:2:10
    println(n)
end
```

Цикл for с шагом 2:

1
3
5
7
9

```
[4]: # Пример 4: Цикл for для работы с массивом строк
println("\nЦикл for для приветствий:")
myfriends = ["Ted", "Robyn", "Barney", "Lily", "Marshall"]
for friend in myfriends
    println("Hi $friend, it's great to see you!")
end
```

Цикл for для приветствий:

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!

Figure 2: Цикл for

```
[5]: # Пример 5: Создание двумерного массива с вложенными циклами
println("\nСоздание массива A с вложенными циклами:")
m, n = 5, 5
A = fill{0, (m, n)}

for i in 1:m
    for j in 1:n
        A[i, j] = i + j
    end
end
println("Массив A:")
println(A)
```

Создание массива A с вложенными циклами:

Массив A:

[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]

```
[6]: # Пример 6: Создание массива B с одним циклом
println("\nСоздание массива B с одним циклом:")
B = fill{0, (m, n)}

for i in 1:m, j in 1:n
    B[i, j] = i + j
end
println("Массив B:")
println(B)
```

Создание массива B с одним циклом:

Массив B:

[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]

Figure 3: Массивы

```
7]: # Пример 7: Создание массива C с помощью генератора
println("\nСоздание массива C с помощью генератора:")
C = [i + j for i in 1:m, j in 1:n]
println("Массив C:")
println(C)

# ===== РАЗДЕЛ 3.2.2 - УСЛОВНЫЕ ВЫРАЖЕНИЯ =====|
```

Figure 4: Массивы

Условное выражение

```
[8]: # Пример 8: Условное выражение if-elseif-else для FizzBuzz
function fizzbuzz(N)
    println("Проверка числа $N:")
    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

# Тестирование функции FizzBuzz
fizzbuzz(15)
fizzbuzz(9)
fizzbuzz(10)
fizzbuzz(7)
```

```
Проверка числа 15:
FizzBuzz
Проверка числа 9:
Fizz
Проверка числа 10:
Buzz
Проверка числа 7:
7
```

Figure 5: Условное выражение

Условное выражение

```
[9]: # Пример 9: Тернарный оператор
println("\nТернарный оператор:")
x = 5
y = 10

result = (x > y) ? x : y
println("Большее число между $x и $y: $result")

# Эквивалент с if-else
if x > y
    result2 = x
else
    result2 = y
end
println("Проверка (if-else): $result2")

Тернарный оператор:
Большее число между 5 и 10: 10
Проверка (if-else): 10
```

Figure 6: Условное выражение

Функции примеры

[11]: # ===== РАЗДЕЛ 3.2.3 - ФУНКЦИИ =====

```
println("\n=== 3.2.3 ФУНКЦИИ ===")
```

Пример 10: Объявление функции с function и end

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

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

```
println("Вызов функций:")
sayhi("С-3Р0")
println("f(42) = $(f(42))")
```

=== 3.2.3 ФУНКЦИИ ===

Вызов функций:

Hi С-3Р0, it's great to see you!

f(42) = 1764

[12]: *# Пример 11: Однострочное объявление функций*
sayhi2(name) = println("Hi \$name, it's great to see you!")
f2(x) = x^2

```
println("\nОднострочные функции:")
sayhi2("R2-D2")
println("f2(5) = $(f2(5))")
```

Однострочные функции:

Hi R2-D2, it's great to see you!

f2(5) = 25

Функции примеры

```
[13]: # Пример 12: Анонимные функции
sayhi3 = name -> println("Hi $name, it's great to see you!")
f3 = x -> x^2

println("\nАнонимные функции:")
sayhi3("BB-8")
println("f3(3) = $(f3(3))")
```

Анонимные функции:
Hi BB-8, it's great to see you!
f3(3) = 9

```
[14]: # Пример 13: Функции с восклицательным знаком
println("\nФункции с восклицательным знаком:")
v = [3, 5, 2]
println("Исходный массив v: $v")

# Функция sort без изменения исходного массива
v_sorted = sort(v)
println("После sort(v): v_sorted = $v_sorted, исходный v = $v")

# Функция sort! с изменением исходного массива
v_original = copy(v)
sort!(v)
println("После sort!(v): изменённый v = $v")

# Восстановление исходного массива для демонстрации
v = [3, 5, 2]
```

Функции с восклицательным знаком:
Исходный массив v: [3, 5, 2]
После sort(v): v_sorted = [2, 3, 5], исходный v = [3, 5, 2]
После sort!(v): изменённый v = [2, 3, 5]

Функции примеры

```
[15]: # Пример 14: Функции высшего порядка - map
println("\nФункция map:")
f_square(x) = x^2
result_map = map(f_square, [1, 2, 3])
println("map(f_square, [1, 2, 3]) = $result_map")

Функция map:
map(f_square, [1, 2, 3]) = [1, 4, 9]

[16]: # Пример 15: map с анонимной функцией
result_map_anon = map(x -> x^3, [1, 2, 3])
println("map(x -> x^3, [1, 2, 3]) = $result_map_anon")

map(x -> x^3, [1, 2, 3]) = [1, 8, 27]

[18]: # Пример 16: Функция broadcast
println("\nФункция broadcast:")
result_broadcast = broadcast(f_square, [1, 2, 3])
println("broadcast(f_square, [1, 2, 3]) = $result_broadcast")
# Точечный синтаксис
result_dot = f_square.([1, 2, 3])
println("f_square.([1, 2, 3]) = $result_dot")

Функция broadcast:
broadcast(f_square, [1, 2, 3]) = [1, 4, 9]
f_square.([1, 2, 3]) = [1, 4, 9]

[ ]:
```

Figure 9: Функции примеры

Функции примеры

```
[19]: # Пример 17: Сравнение map и broadcast для матриц
println("\nСравнение map и broadcast для матриц:")
A_matrix = [i + 3*j for j in 0:2, i in 1:3]
println("Матрица A:")
println(A_matrix)

# Обычное применение функции (матричное умножение)
result_matrix = f_square(A_matrix)
println("f_square(A) (матричное умножение):")
println(result_matrix)

# Поэлементное применение с broadcast
result_broadcast_matrix = f_square.(A_matrix)
println("f_square.(A) (поэлементное возведение в квадрат):")
println(result_broadcast_matrix)
```

Сравнение map и broadcast для матриц:

Матрица A:

[1 2 3; 4 5 6; 7 8 9]

f_square(A) (матричное умножение):

[30 36 42; 66 81 96; 102 126 150]

f_square.(A) (поэлементное возведение в квадрат):

[1 4 9; 16 25 36; 49 64 81]

Figure 10: Функции примеры

```
[20]: # Пример 18: Сложные выражения с точечным синтаксисом
println("\nСложные выражения с точечным синтаксисом:")

# Способ 1: Явный вызов broadcast
result1 = broadcast(x -> x + 2 * f_square(x) / x, A_matrix)
println("broadcast(x -> x + 2 * f_square(x) / x, A):")
println(result1)

# Способ 2: Точечный синтаксис
result2 = A_matrix .+ 2 .* f_square.(A_matrix) ./ A_matrix
println("A .+ 2 .* f_square.(A) ./ A:")
println(result2)

# Способ 3: Макрос @.
using LinearAlgebra
result3 = @. A_matrix + 2 * f_square(A_matrix) / A_matrix
println("@. A + 2 * f_square(A) / A:")
println(result3)
```

```
Сложные выражения с точечным синтаксисом:
broadcast(x -> x + 2 * f_square(x) / x, A):
[3.0 6.0 9.0; 12.0 15.0 18.0; 21.0 24.0 27.0]
A .+ 2 .* f_square.(A) ./ A:
[3.0 6.0 9.0; 12.0 15.0 18.0; 21.0 24.0 27.0]
@. A + 2 * f_square(A) / A:
[3.0 6.0 9.0; 12.0 15.0 18.0; 21.0 24.0 27.0]
```

Figure 11: Функции примеры

Сторонние библиотеки

[23]: # ===== РАЗДЕЛ 3.2.4 - СТОРОННИЕ БИБЛИОТЕКИ =====

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

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

```
palette = distinguishable_colors(100)
rand(palette, 3, 3)
```

```
Resolving package versions...
No Changes to `C:\Users\vsvld\.julia\environments\v1.11\Project.toml`
No Changes to `C:\Users\vsvld\.julia\environments\v1.11\Manifest.toml`
Resolving package versions...
Installed FixedPointNumbers v0.8.5
Installed ColorTypes v0.12.1
Installed Reexport v1.2.2
Installed Statistics v1.11.1
Installed Colors v0.13.1
Updating `C:\Users\vsvld\.julia\environments\v1.11\Project.toml`
[5ae59095] + Colors v0.13.1
Updating `C:\Users\vsvld\.julia\environments\v1.11\Manifest.toml`
[3da002f7] + ColorTypes v0.12.1
[5ae59095] + Colors v0.13.1
[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...
 937.4 ms ✓ Reexport
1406.9 ms ✓ Statistics
3742.8 ms ✓ FixedPointNumbers
2413.8 ms ✓ ColorTypes
1473.2 ms ✓ ColorTypes + StyledStringsExt
7243.8 ms ✓ Colors
6 dependencies successfully precompiled in 16 seconds. 43 already precompiled.
```

[23]:



Самостоятельная работа. Задание 1

```
[1]: # ===== ЗАДАНИЕ 1 =====
println("=== ЗАДАНИЕ 1 ===")

# 1.1 Вывод чисел от 1 до 100 и их квадратов
println("Числа от 1 до 100 и их квадраты:")
for i in 1:100
    println("$i -> $(i^2)")
end

=== ЗАДАНИЕ 1 ===
Числа от 1 до 100 и их квадраты:
1 -> 1
2 -> 4
3 -> 9
4 -> 16
5 -> 25
6 -> 36
7 -> 49
8 -> 64
9 -> 81
10 -> 100
11 -> 121
12 -> 144
13 -> 169
14 -> 196
15 -> 225
16 -> 256
17 -> 289
18 -> 324
```

Figure 13: Самостоятельная работа. Задание 1

Самостоятельная работа. Задание 1

```
[2]: # 1.2 Создание словаря squares
      printIn("\nСловарь squares:")
      squares = Dict{Int, Int}{}
      for i in 1:100
          squares[i] = i^2
      end
      printIn(squares)
```

Словарь squares:

Dict{5 => 25, 56 => 3136, 35 => 1225, 55 => 3025, 60 => 3600, 30 => 900, 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, 23 => 529, 92 => 8464, 41 => 1681, 43 => 1849, 11 => 121, 36 => 1296, 68 => 4624, 69 => 4761, 98 => 9604, 82 => 6724, 85 => 7225, 39 => 1521, 84 => 7056, 77 => 5929, 7 => 49, 25 => 625, 95 => 9025, 71 => 5041, 66 => 4356, 76 => 5776, 34 => 1156, 50 => 2500, 59 => 3481, 93 => 8649, 2 => 4, 10 => 100, 18 => 324, 26 => 676, 27 => 729, 42 => 1764, 87 => 7569, 100 => 10000, 79 => 6241, 16 => 256, 20 => 400, 81 => 6561, 19 => 361, 49 => 2401, 44 => 1936, 9 => 81, 31 => 961, 74 => 5476, 61 => 3721, 29 => 841, 94 => 8836, 46 => 2116, 57 => 3249, 70 => 4900, 21 => 441, 38 => 1444, 88 => 7744, 78 => 6084, 72 => 5184, 24 => 576, 8 => 64, 17 => 289, 37 => 1369, 1 => 1, 53 => 2809, 22 => 484, 47 => 2209, 83 => 6889, 99 => 9801, 89 => 7921, 14 => 196, 3 => 9, 80 => 6400, 96 => 9216, 51 => 2601, 33 => 1089, 40 => 1600, 48 => 2304, 15 => 225, 65 => 4225, 97 => 9409}

```
[3]: # 1.3 Создание массива squares_arr
      printIn("\nМассив squares_arr:")
      squares_arr = [i^2 for i in 1:100]
      printIn(squares_arr)
```

Массив squares_arr:

[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]

Figure 14: Самостоятельная работа. Задание 1

Самостоятельная работа. Задание 2

```
[4]: # ===== ЗАДАНИЕ 2 =====
println("\n=== ЗАДАНИЕ 2 ===")

# 2.1 Условный оператор
function check_even_odd(n)
    if n % 2 == 0
        println(n)
    else
        println("нечётное")
    end
end

println("Проверка чётности (условный оператор):")
check_even_odd(4)
check_even_odd(7)
```

```
=== ЗАДАНИЕ 2 ===
Проверка чётности (условный оператор):
4
нечётное
```

```
[5]: # 2.2 Тернарный оператор
function check_even_odd_ternary(n)
    result = n % 2 == 0 ? n : "нечётное"
    println(result)
end

println("Проверка чётности (тернарный оператор):")
check_even_odd_ternary(4)
check_even_odd_ternary(7)
```

```
Проверка чётности (тернарный оператор):
4
нечётное
```

Figure 15: Самостоятельная работа. Задание 2

Самостоятельная работа. Задание 3-4

```
[6]: # ----- ЗАДАНИЕ 3 -----
println("\n=== ЗАДАНИЕ 3 ===")

# 3. Функция add_one
add_one(x) = x + 1

println("Функция add_one:")
println("add_one(5) = $(add_one(5))")
println("add_one(-3) = $(add_one(-3))")

=== ЗАДАНИЕ 3 ===
Функция add_one:
add_one(5) = 6
add_one(-3) = -2

[7]: # ----- ЗАДАНИЕ 4 -----
println("\n=== ЗАДАНИЕ 4 ===")

# 4. Использование map/broadcast для увеличения элементов матрицы на 1
A = [1 2 3; 4 5 6; 7 8 9]
println("Исходная матрица A:")
println(A)

A_plus_one = map(x -> x + 1, A)
println("Матрица A после увеличения всех элементов на 1 (map):")
println(A_plus_one)

# Альтернативный вариант с broadcast
A_plus_one_broadcast = A .+ 1
println("Матрица A после увеличения всех элементов на 1 (broadcast):")
println(A_plus_one_broadcast)

=== ЗАДАНИЕ 4 ===
Исходная матрица A:
[1 2 3; 4 5 6; 7 8 9]
Матрица A после увеличения всех элементов на 1 (map):
[2 3 4; 5 6 7; 8 9 10]
```

Самостоятельная работа. Задание 5

```
[8]:  
# ===== ЗАДАНИЕ 5 =====  
println("\n=== ЗАДАНИЕ 5 ===")  
  
# 5.1 Задание матрицы A  
A = [1 1 3; 5 2 6; -2 -1 -3]  
println("Матрица A:")  
println(A)  
  
# 5.2 Нахождение A^3  
A_cubed = A^3  
println("A^3:")  
println(A_cubed)  
  
# 5.3 Замена третьего столбца на сумму второго и третьего  
A_modified = copy(A)  
A_modified[:, 3] = A[:, 2] + A[:, 3]  
println("Матрица A с заменённым третьим столбцом:")  
println(A_modified)  
  
=== ЗАДАНИЕ 5 ===  
Матрица A:  
[1 1 3; 5 2 6; -2 -1 -3]  
A^3:  
[0 0 0; 0 0 0; 0 0 0]  
Матрица A с заменённым третьим столбцом:  
[1 1 4; 5 2 8; -2 -1 -4]
```

Figure 17: Самостоятельная работа. Задание 5

Самостоятельная работа. Задание 6

```
[9]: # ----- ЗАДАНИЕ 6 -----  
println("\n=== ЗАДАНИЕ 6 ===")  
  
# 6. Создание матрицы B и вычисление C = B^T * B  
B = zeros(15, 3)  
for i in 1:15  
    B[i, 1] = 10  
    B[i, 2] = -10  
    B[i, 3] = 10  
end  
  
println("Матрица B:")  
println(B)  
  
C = B' * B  
println("Матрица C = B^T * B:")  
println(C)  
  
=== ЗАДАНИЕ 6 ===  
Матрица B:  
[10.0 -10.0 10.0; 10.0 -10.0 10.0; 10.0 -10.0 10.0; 10.0 -10.0 10.0; 10.0 -10.0 10.0; 10.0 -10.0 10.0; 10.0 -10.0 10.0; 10.0 -10.0 10.0; 10.0 -10.0 10.0; 10.0 -10.0 10.0; 10.0 -10.0 10.0; 10.0 -10.0 10.0; 10.0 -10.0 10.0; 10.0 -10.0 10.0; 10.0 -10.0 10.0]  
Матрица C = B^T * B:  
[1500.0 -1500.0 1500.0; -1500.0 1500.0 -1500.0; 1500.0 -1500.0 1500.0]
```

Figure 18: Самостоятельная работа. Задание 6

Самостоятельная работа. Задание 7

```
[10]: # ===== ЗАДАНИЕ 7 =====
println("\n=== ЗАДАНИЕ 7 ===")

# 7. Создание матриц Z1, Z2, Z3, Z4
n = 6
Z = zeros{Int, n, n}
E = ones{Int, n, n}

# Матрица Z1
Z1 = zeros{Int, n, n}
for i in 1:n
    for j in 1:n
        if abs(i - j) == 1
            Z1[i, j] = 1
        end
    end
end
println("Матрица Z1:")
println(Z1)

=== ЗАДАНИЕ 7 ===
Матрица Z1:
[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]
```

Figure 19: Самостоятельная работа. Задание 7

Самостоятельная работа. Задание 7

```
[11]: # Матрица Z2
Z2 = zeros(Int, n, n)
for i in 1:n
    for j in 1:n
        if (i + j) % 2 == 1 && abs(i - j) <= 2
            Z2[i, j] = 1
        end
    end
end
println("Матрица Z2:")
println(Z2)

# Матрица Z3
Z3 = zeros(Int, n, n)
for i in 1:n
    for j in 1:n
        if (i == 1 && (j == 3 || j == 6)) ||
            (i == 2 && (j == 3 || j == 5)) ||
            (i >= 3 && i <= 6 && abs(i - j) == 1)
            Z3[i, j] = 1
        end
    end
end
println("Матрица Z3:")
println(Z3)

Матрица Z2:
[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]
Матрица Z3:
[0 0 1 0 0 1; 0 0 1 0 1 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]
```

Figure 20: Самостоятельная работа. Задание 7

Самостоятельная работа. Задание 7

```
[12]: # Матрица Z4
Z4 = zeros(Int, n, n)
for i in 1:n
    for j in 1:n
        if (i + j) % 2 == 1 && !(i == 1 && j == 5) && !(i == 1 && j == 6)
            Z4[i, j] = 1
        end
    end
end
println("Матрица Z4:")
println(Z4)

Матрица Z4:
[0 1 0 1 0 0; 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]
```

Figure 21: Самостоятельная работа. Задание 7

Самостоятельная работа. Задание 8

```
[13]: # ===== ЗАДАНИЕ 8 =====
println("\n--- ЗАДАНИЕ 8 ---")

# 8.1 Функция outer (аналог функции из R)
function outer(x, y, operation)
    result = similar(x, size(x, 1), size(y, 2))
    for i in 1:size(x, 1)
        for j in 1:size(y, 2)
            result[i, j] = operation(x[i], y[j])
        end
    end
    return result
end

# Тестирование функции outer
A_test = [1, 2, 3]
B_test = [4, 5, 6]
println("Тест функции outer (сложение):")
println(outer(A_test, B_test, +))

# 8.2 Создание матриц A1-A5 с помощью outer

# Матрица A1
A1 = outer(0:4, 0:4, +)
println("Матрица A1:")
println(A1)

=== ЗАДАНИЕ 8 ===
Тест функции outer (сложение):
[5; 6; 7;;]
Матрица A1:
[0; 1; 2; 3; 4;;]
```

Figure 22: Самостоятельная работа. Задание 8

Самостоятельная работа. Задание 8

```
[14]: # Матрица A2
A2 = outer(0:4, 0:4, (x,y) -> x^y)
println("Матрица A2:")
println(A2)

# Матрица A3
A3 = outer(0:4, 0:4, (x,y) -> (x + y) % 5)
println("Матрица A3:")
println(A3)

# Матрица A4
A4 = outer(0:9, 0:9, (x,y) -> (x + y) % 10)
println("Матрица A4:")
println(A4)

# Матрица A5
A5 = outer(0:8, 0:8, (x,y) -> (9 - abs(x - y)) % 10)
println("Матрица A5:")
println(A5)
```

Матрица A2:

```
[1; 1; 1; 1; 1; 1;;]
Матрица A3:
```

```
[0; 1; 2; 3; 4;;]
Матрица A4:
```

```
[0; 1; 2; 3; 4; 5; 6; 7; 8; 9;;]
Матрица A5:
```

```
[9; 8; 7; 6; 5; 4; 3; 2; 1;;]
```

Figure 23: Самостоятельная работа. Задание 8

Самостоятельная работа. Задание 9

```
[15]: # ===== ЗАДАНИЕ 9 =====
println("\n--- ЗАДАНИЕ 9 ---")

# 9. Решение системы линейных уравнений
using LinearAlgebra

# Матрица коэффициентов A
A_system = [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]

# Вектор правой части
b = [7, -1, -3, 5, 17]

# Решение системы
x = A_system \ b
println("Решение системы уравнений:")
println("x = $x")

# Проверка решения
println("Проверка: A * x = $(A_system * x)")
println("Ожидаемый результат: $b")

=== ЗАДАНИЕ 9 ===
Решение системы уравнений:
x = [-1.999999999999987, 2.999999999999996, 4.999999999999998, 2.000000000000001, -4.0]
Проверка: A * x = [6.999999999999964, -1.0, -2.999999999999964, 5.0, 17.0]
Ожидаемый результат: [7, -1, -3, 5, 17]
```

Figure 24: Самостоятельная работа. Задание 9

Самостоятельная работа. Задание 10

```
[17]: # ===== ЗАДАНИЕ 10 =====
println("\n== ЗАДАНИЕ 10 ==")

using Random

# 10.1 Создание случайной матрицы M
Random.seed!(123) # для воспроизводимости результатов
M = rand{Int, 6, 10}
println("Матрица M:")
println(M)

# 10.2 Число элементов > N в каждой строке
N = 4
count_greater_than_N = [count(x -> x > N, M[i, :]) for i in 1:size(M, 1)]
println("Число элементов > $N в каждой строке: $count_greater_than_N")

# 10.3 Строки, где число M встречается ровно 2 раза
M_target = 7
rows_with_two_M = [i for i in 1:size(M, 1) if count(x -> x == M_target, M[i, :]) == 2]
println("Строки, где число $M_target встречается ровно 2 раза: $rows_with_two_M")

# 10.4 Пары столбцов с суммой элементов > K
K = 75
column_pairs = []
for i in 1:size(M, 2)
    for j in i+1:size(M, 2)
        if sum(M[:, i] + M[:, j]) > K
            push!(column_pairs, (i, j))
        end
    end
end
println("Пары столбцов с суммой элементов > $K: $column_pairs")

=== ЗАДАНИЕ 10 ===
Матрица M:
[6 1 1 7 4 3 4 2 7; 6 10 5 3 2 1 1 9 6 10; 9 6 4 3 1 6 1 3 9 8; 2 4 2 10 4 4 10 10 3 7; 6 6 8 9 5 2 4 9 9 9; 4 9 6 3 8 3 8 8 4 5]
Число элементов > 4 в каждой строке: [3, 6, 5, 4, 8, 6]
Строки, где число 7 встречается ровно 2 раза: [1]
Пары столбцов с суммой элементов > 75: Any{([1, 8], (1, 10), (2, 8), (2, 10), (4, 8), (4, 10), (8, 9), (8, 10), (9, 10])}
```

Figure 25: Самостоятельная работа. Задание 10

Самостоятельная работа. Задание 11

```
[18]: # ----- ЗАДАНИЕ 11 -----  
println("\n== ЗАДАНИЕ 11 ==")  
  
# 11.1 Вычисление первой суммы  
sum1 = 0.0  
for i in 1:20  
    for j in 1:5  
        sum1 += i^4 / (3 + j)  
    end  
end  
println("Первая сумма: $sum1")  
  
# 11.2 Вычисление второй суммы  
sum2 = 0.0  
for i in 1:20  
    for j in 1:5  
        sum2 += i^4 / (3 + i*j)  
    end  
end  
println("Вторая сумма: $sum2")  
  
# Альтернативный вариант с использованием генераторов  
sum1_alt = sum(i^4 / (3 + j) for i in 1:20 for j in 1:5)  
sum2_alt = sum(i^4 / (3 + i*j) for i in 1:20 for j in 1:5)  
  
println("Первая сумма (альтернативный расчет): $sum1_alt")  
println("Вторая сумма (альтернативный расчет): $sum2_alt")  
  
--- ЗАДАНИЕ 11 ---  
Первая сумма: 639215.2833333334  
Вторая сумма: 89912.02146097136  
Первая сумма (альтернативный расчет): 639215.2833333334  
Вторая сумма (альтернативный расчет): 89912.02146097136
```

1.

Figure 26: Самостоятельная работа. Задание 11

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