

# **Управляющие структуры**

**Лабораторная работа № 3**

**Шулуужук Айраана НПИбд-02-22**

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Циклы for, while . . . . .	6
2.2	Условные выражения . . . . .	7
2.3	Функции . . . . .	8
2.4	Сторонние библиотеки (пакеты) в Julia . . . . .	9
2.5	Выполнение самостоятельной работы . . . . .	10
<b>3</b>	<b>Выводы</b>	<b>18</b>

# Список иллюстраций

2.1	цикл while . . . . .	6
2.2	цикл for . . . . .	7
2.3	условные выражения и тернарный оператор . . . . .	7
2.4	функции . . . . .	8
2.5	функции, сопровождаемые восклицательным знаком . . . . .	8
2.6	map(), broadcast . . . . .	9
2.7	сторонние библиотеки (пакеты) в Julia . . . . .	9
2.8	1 пункт . . . . .	10
2.9	2 пункт . . . . .	10
2.10	3 пункт . . . . .	11
2.11	2 задание . . . . .	11
2.12	3 задание . . . . .	11
2.13	4 задание . . . . .	12
2.14	5 задание . . . . .	12
2.15	6 задание . . . . .	13
2.16	7 задание . . . . .	13
2.17	7 задание . . . . .	14
2.18	8 задание . . . . .	14
2.19	8 задание . . . . .	15
2.20	9 задание . . . . .	15
2.21	10 задание . . . . .	16
2.22	10 задание . . . . .	16
2.23	10 задание . . . . .	16
2.24	10 задание . . . . .	17
2.25	11 задание . . . . .	17

## **Список таблиц**

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

### 2.1 Циклы for, while

Для различных операций, связанных с перебором индексируемых элементов структур данных, традиционно используются циклы `while` и `for` (рис. 2.1) (рис. 2.2)

Циклы `while` и `for`

```
[416]: # Формирование 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

[3]: # Демонстрация использования while при работе со строковыми элементами массива:
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. 2.1: цикл `while`

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

1
3
5
7
9

[7]: # Рассмотренные выше примеры, но с использованием цикла for:
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. 2.2: цикл for

## 2.2 Условные выражения

Довольно часто при решении задач требуется проверить выполнение тех или иных условий. Для этого используют условные выражения (рис. 2.3)

```
[13]: # 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
N = 100

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

[14]: # Пример использования тернарного оператора:

x = 5
y = 10

(x > y) ? x : y

[14]: 10
```

Рис. 2.3: условные выражения и тернарный оператор

## 2.3 Функции

Julia дает нам несколько разных способов написать функцию. Первый требует ключевых слов `function` и `end` (рис. 2.4) (рис. 2.5) (рис. 2.6)

```
function sayhi(name)
    println("Hi $name, it's great to see you!")
end
# функция возведения в квадрат:
function f(x)
    x^2
end

# Вызов функции осуществляется по её имени с указанием аргументов, например:
sayhi("C-3PO")
f(42)

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

]: # 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("C-3PO")
f(42)
```

Рис. 2.4: функции

```
[27]: # Сравнение результата применения sort и sort!:

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

[27]: 3-element Vector{Int64}:
      3
      5
      2

[28]: sort!(v)
v

[28]: 3-element Vector{Int64}:
      2
      3
      5

[31]: # 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^3
map(f, [1, 2, 3])

[31]: 3-element Vector{Int64}:
      1
      8
     27
```

Рис. 2.5: функции, сопровождаемые восклицательным знаком



```
[31]: # 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^3
map(f, [1, 2, 3])

[31]: 3-element Vector{Int64}:
      1
      8
     27

[418]: # 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

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

[418]: 3-element Vector{Int64}:
      1
      4
      9
```

Рис. 2.6: map(), broadcast

## 2.4 Сторонние библиотеки (пакеты) в Julia

При первом использовании пакета в вашей текущей установке Julia вам необходимо использовать менеджер пакетов, чтобы явно его добавить (рис. 2.7)

```
[177]: ## Добавим и загрузим пакет Colors:
import Pkg
Pkg.add("Colors")
using Colors

# Затем создадим палитру из 100 разных цветов:
palette = distinguishable_colors(100)

# а затем определим матрицу 3 × 3 с элементами в форме случайного цвета из палитры, используя функцию rand:
rand(palette, 3, 3)
```

Resolving package versions...  
No Changes to `C:\Users\airan\.julia\environments\v1.11\Project.toml`  
No Changes to `C:\Users\airan\.julia\environments\v1.11\Manifest.toml`

```
[177]:
```




Рис. 2.7: сторонние библиотеки (пакеты) в Julia

## 2.5 Выполнение самостоятельной работы

### 1. Используя циклы while и for:

- выведите на экран целые числа от 1 до 100 и напечатайте их квадраты (рис. 2.8).

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

```
[196]: n = 1
while n <= 100
    println("${n^2}")
    n += 1
end

1
4
9
16
25
36
49
64
81
100
121
```

Рис. 2.8: 1 пункт

- создайте словарь squares, который будет содержать целые числа в качестве ключей и квадраты в качестве их пар-значений (рис. 2.9).

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

```
[203]: squares = Dict{<
for n in 1:100
    squares[n] = n^2
end
println(squares)

Dict{Any, Any}{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}
```

Рис. 2.9: 2 пункт

- создайте массив squares\_arr, содержащий квадраты всех чисел от 1 до 100. (рис. 2.10).

– создайте массив `squares_arr`, содержащий квадраты всех чисел от 1 до 100.

```
[206]: squares_arr = [n**2 for n in 1:100]
println(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, 19
36, 2025, 2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249, 3364, 3481, 3600, 3721, 384
4, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5041, 5184, 5329, 5476, 5625, 5776, 5929, 6084, 6241, 640
0, 6561, 6724, 6889, 7056, 7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409, 960
4, 9801, 10000]
```

Рис. 2.10: 3 пункт

2. Напишите условный оператор, который печатает число, если число чётное, и строку «нечётное», если число нечётное. Перепишите код, используя тернарный оператор (рис. 2.11).

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

```
[217]: n = 100
if n % 2 == 0
    println(n)
else
    println("нечетное")
end

100

[219]: println(n % 2 == 0 ? n : "нечетное")

100
```

Рис. 2.11: 2 задание

3. Напишите функцию `add_one`, которая добавляет 1 к своему входу (рис. 2.12)

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

```
]: function add_one(i)
    return i + 1
end
add_one(9)

]: 10
```

Рис. 2.12: 3 задание

4. Используйте `map()` или `broadcast()` для задания матрицы  $\square$ , каждый элемент которой увеличивается на единицу по сравнению с предыдущим (рис. 2.13).

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

```
[231]: A = reshape(1:16, 4, 4)
A

[231]: 4x4 reshape(::UnitRange{Int64}, 4, 4) with eltype Int64:
 1  5   9  13
 2  6  10  14
 3  7  11  15
 4  8  12  16

[233]: B = map(x -> x + 1, A)
B

[233]: 4x4 Matrix{Int64}:
 2  6  10  14
 3  7  11  15
 4  8  12  16
 5  9  13  17

[235]: C = broadcast(x -> x + 1, B)
C

[235]: 4x4 Matrix{Int64}:
 3  7  11  15
 4  8  12  16
 5  9  13  17
 6 10  14  18
```

Рис. 2.13: 4 задание

5. Задайте матрицу *A* следующего вида, найти куб, заменить 3 столбец на сумму 2 и 3 столбцов (рис. 2.14)

5. Задайте матрицу *A* следующего вида:

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

```
[429]: 3x3 Matrix{Int64}:
 1  1  3
 5  2  6
-2 -1 -3
```

```
[431]: # найдем матрицу в кубе
B = map(x -> x^3, A)
B
```

```
[431]: 3x3 Matrix{Int64}:
 1  1  27
125  8  216
-8 -1 -27
```

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

```
[433]: 3x3 Matrix{Int64}:
 1  1  4
 5  2  8
-2 -1 -4
```

Рис. 2.14: 5 задание

6. Создайте матрицу  $B$  с элементами  $B_{i1} = 10, B_{i2} = -10, B_{i3} = 10, i = 1, 2, \dots, 15$ . Вычислите матрицу  $C = BTB$  (рис. 2.15)

6. Создайте матрицу  $B$  с элементами  $B_{i1} = 10, B_{i2} = -10, B_{i3} = 10, i = 1, 2, \dots, 15$ . Вычислите матрицу  $C = BTB$ .

```
[253]: B = repeat([10 -10 10], 15, 1)
B
[253]: 15x3 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
 10 -10 10

[255]: C = B' * B
[255]: 3x3 Matrix{Int64}:
 1500 -1500 1500
-1500 1500 -1500
 1500 -1500 1500
```

Рис. 2.15: 6 задание

7. Создайте матрицу  $Z$  размерности  $6 \times 6$ , все элементы которой равны нулю, и матрицу  $E$ , все элементы которой равны 1. Используя цикл `while` или `for` и закономерности расположения элементов, создайте следующие матрицы размерности  $6 \times 6$  (рис. 2.16) (рис. 2.17)

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

```
[441]: Z = fill{0, 6, 6}
[441]: 6x6 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
```

Рис. 2.16: 7 задание

```
[443]: Z1 = fill(0, 6, 6)
for i in 1:6
    for j in 1:6
        if abs(i - j) == 1
            Z1[i, j] = 1
        end
    end
end
Z1

[443]: 6x6 Matrix{Int64}:
 0  1  0  0  0  0
 1  0  1  0  0  0
 0  1  0  1  0  0
 0  0  1  0  1  0
 0  0  0  1  0  1
 0  0  0  0  1  0

[445]: Z4 = copy(Z)
for i in 1:6
    for j in 1:6
        if (i + j) % 2 == 0
            Z4[i, j] = 1
        end
    end
end
Z4

[445]: 6x6 Matrix{Int64}:
 1  0  1  0  1  0
 0  1  0  1  0  1
 1  0  1  0  1  0
 0  1  0  1  0  1
 1  0  1  0  1  0
 0  1  0  1  0  1
```

Рис. 2.17: 7 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).
- 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  $C$  (рис. 2.18) (рис. 2.19)

– 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  $C_{ij} = \sum_{k=1}^M A_{ik}B_{kj}$

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

325]: outer (generic function with 1 method)
```

Рис. 2.18: 8 задание

```
[361]: A1 = outer(0:4, 0:4, +)

[361]: 5x5 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

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

[363]: 5x5 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

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

[365]: 10x10 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

[367]: A5 = outer(0:8, 0:8, (x, y) -> mod(x - y, 9))

[367]: 9x9 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. 2.19: 8 задание

9. Решите следующую систему линейных уравнений с 5 неизвестными (рис. 2.20)

9. Решите следующую систему линейных уравнений с 5 неизвестными:

```
[370]: 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, 17]
       C = A \ B

[370]: 5-element Vector{Float64}:
 -2.0000000000000036
  3.0000000000000058
  4.9999999999999998
  1.9999999999999991
 -3.9999999999999999
```

Рис. 2.20: 9 задание

10. Создайте матрицу М размерности  $6 \times 10$ , элементами которой являются целые числа, выбранные случайным образом с повторениями из совокупности 1, 2, ..., 10 (рис. 2.21)

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

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

J> 6x10 Matrix{Int64}:
 2  5  6  5  10  2  3  10  7  5
 5  2  9  5  4  7  7  2  10  5
 4  4  1  6  8  7  2  9  6  1
 5  2  5  2  2  6  1  10  3  9
 1  4  6  6  8  2  4  7  3  5
 6  3  1  4  8  1  3  5  3  10
```

Рис. 2.21: 10 задание

– Найдите число элементов в каждой строке матрицы  $M$ , которые больше числа  $N$  (например,  $N = 4$ ) (рис. 2.22)

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

```
J> N = 4
A = sum(M .> N, dims = 2)

J> 6x1 Matrix{Int64}:
 7
 7
 5
 5
 5
 4
```

Рис. 2.22: 10 задание

– Определите, в каких строках матрицы  $M$  число  $M$  (например,  $M = 7$ ) встречается ровно 2 раза? (рис. 2.23)

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

```
J> M_var = 7
B = findall(x -> count(==(M_var), x) == 2, eachrow(M))

J> 1-element Vector{Int64}:
 2
```

Рис. 2.23: 10 задание

– Определите все пары столбцов матрицы  $M$ , сумма элементов которых больше  $K$  (например,  $K = 75$ ) (рис. 2.24)



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

```
] : K = 75
pairs = []
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, (i, j))
        end
    end
end
println(pairs)
Any[(5, 8), (8, 10)]
```

Рис. 2.24: 10 задание

## 11. Вычислите (рис. 2.25)

### 11. Вычислите:

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

```
[408]: 21679980
```

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

```
[410]: 195839490
```

Рис. 2.25: 11 задание

## **3 Выводы**

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