

Лабораторная работа № 3.

Управляющие структуры

Оразгелдиев Язгелди

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	16

Список иллюстраций

3.1	Циклы while и for	7
3.2	Условные выражения	8
3.3	Функции	8
3.4	Colors	9
3.5	задание 1	10
3.6	Задание 1	10
3.7	Задание 2 и 3	11
3.8	Задание 4 и 5	12
3.9	Задание 6	12
3.10	Задание 7	13
3.11	Задание 8 и 9	14
3.12	Задание 10	15
3.13	Задание 11	15

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.

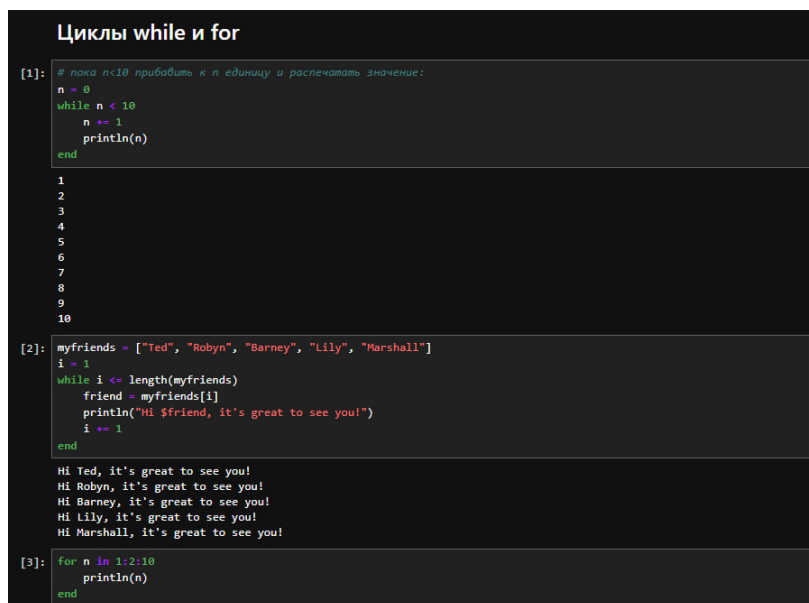
2 Задание

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

3 Выполнение лабораторной работы

Перед началом выполнения заданий, мы повторяем примеры из раздела методики

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



```
Циклы while и for

[1]: # 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

[2]: 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!

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

Рисунок 3.1: Циклы `while` и `for`

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

Условные выражения

```

# используем "&&" для реализации операции "AND"
# операция % вычисляет остаток от деления
N = 15
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

FizzBuzz

# a ? b : c (означает if (a): b; else: c)
x = 5
y = 10
(x > y) ? x : y

10

```

Рисунок 3.2: Условные выражения

Julia дает нам несколько разных способов написать функцию.

Функции

```

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)

# Вызов функции
sayhi("С-3Р0")
f(42)

Hi С-3Р0, it's great to see you!
1764

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

3-element Vector{Int64}:
 3
 5
 2

Функция sort(v) возвращает отсортированный массив, который содержит те же элементы, что и массив v, но исходный массив v остаётся без изменений. Если же испол зовать sort!(v), то отсортировано будет содержимое исходного массива v.

sort!(v)
v

```

Рисунок 3.3: Функции

Подключение сторонних библиотек


```
Сторонние библиотеки (пакеты) в Julia

julia> import Pkg
julia> Pkg.add("Example")

Updating registry at `C:\Users\Yazgeldi\.julia\registries\General.toml`
Resolving package versions...
Installing Example - v0.5.5
Updating `C:\Users\Yazgeldi\.julia\environments\v1.10\Project.toml`
  [7f70cd16] + Example v0.5.5
Updating `C:\Users\Yazgeldi\.julia\environments\v1.10\Manifest.toml`
  [7f70cd16] + Example v0.5.5
Precompiling packages...
 703.4 ms ✓ Example
1 dependency successfully precompiled in 2 seconds. 11 already precompiled.

julia> Pkg.add("Colors")
using Colors

Resolving package versions...
Installing ColorTypes - v0.12.1
Installing Colors - v0.13.1
Updating `C:\Users\Yazgeldi\.julia\environments\v1.10\Project.toml`
  [c34428d8] + Colors v0.13.1
Updating `C:\Users\Yazgeldi\.julia\environments\v1.10\Manifest.toml`
  [c34428d8] + ColorTypes v0.12.1
  [c34428d8] + Colors v0.13.1
  [c34428d8] + FixedPointNumbers v0.8.5
  [c34428d8] + Base64 v1.2.2
  [c34428d8] + LinearAlgebra
  [c34428d8] + SparseArrays v1.10.0
  [c34428d8] + Statistics v1.10.0
  [c34428d8] + CompilerSupportLibraries_jll v1.1.1+0
  [c34428d8] + OpenBLAS_jll v0.3.23+4
  [c34428d8] + SuiteSparse_jll v6.2.14+1
  [c34428d8] + liblapacke_jll v3.11.0+0
Precompiling packages...
 485.0 ms ✓ Colors
2302.3 ms ✓ LinearAlgebra
1543.5 ms ✓ Base64
3509.1 ms ✓ Colors
```

Рисунок 3.4: Colors

Задание для самостоятельной работы

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

```
Самостоятельная работа

Задача 1

# Часть 1: Вывод целых чисел от 1 до 100 и их квадратов
for i in 1:100
    println("$i, ${i^2}")
end

# Часть 2: Создание словаря с квадратами чисел
squares = Dict{i => i^2 for i in 1:100}

# Часть 3: Создание массива с квадратами чисел
squares_arr = [i^2 for i in 1:100]

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
19, 361
```

Рисунок 3.5: задание 1

```
100-element Vector{Int64}:
 1
 4
 9
16
25
36
49
64
81
100
121
144
169
 ⋮
7921
8100
8281
8464
8649
8836
9025
9216
9409
9604
9801
10000
```

Рисунок 3.6: Задание 1

2. Написали условный оператор, который печатает число, если число чётное, и строку «нечётное», если число нечётное. Переписали код, используя тернарный оператор.
3. Написали функцию `add_one`, которая добавляет 1 к своему входу.

```
Задача 2

# Функция для проверки четности числа и вывода результата
function check_even_odd(n)
    if n % 2 == 0
        println(n)
    else
        println("нечётное")
    end
end
# Тернарный оператор
n % 2 == 0 ? println(n) : println("нечётное")
end

# Пример использования функции check_even_odd с числом 4 и 5
check_even_odd(4)
check_even_odd(5)

4
4
нечётное
нечётное

Задача 3

# Функция add_one
add_one = x -> x + 1

# Пример использования функции add_one с числом 5
add_one_result = add_one(546)

647
```

Рисунок 3.7: Задание 2 и 3

4. Использовал `map()` или `broadcast()` для задания матрицы \mathbb{N} , каждый элемент которой увеличивается на единицу по сравнению с предыдущим.
5. Задайте матрицу \mathbb{N} следующего вида: – Нашли \mathbb{N}
 - Заменяли третий столбец матрицы \mathbb{N} на сумму второго и третьего столбцов.

Задача 4

```
# Создание матрицы A размером 5x5, где каждый элемент увеличивается на 1
A = reshape(1:25, 5, 5)

# Использование map или broadcast для увеличения каждого элемента на 1
A_map = map(x -> x + 1, A)
A_broadcast = A .+ 1
```

```
5x5 Matrix{Int64}:
 2  7 12 17 22
 3  8 13 18 23
 4  9 14 19 24
 5 10 15 20 25
 6 11 16 21 26
```

Задача 5

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

# Вычисление A^3
A_cubed = A^3

# Замена третьего столбца на сумму второго и третьего столбцов
A[:, 3] = A[:, 2] + A[:, 3]
```

```
3-element Vector{Int64}:
 4
 8
-4
```

Рисунок 3.8: Задание 4 и 5

6. Создали матрицу \mathbb{B} с элементами $\mathbb{B}_{11} = 10, \mathbb{B}_{12} = -10, \mathbb{B}_{13} = 10, \mathbb{B}_{ij} = 1, 2, \dots, 15$.
Вычислили матрицу $\mathbb{C} = \mathbb{B}\mathbb{B}^T$.

Задача 6

```
# Создание матрицы B размером 15x3
B = [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]

# Вычисление матрицы C = B^T * B
C = transpose(B) * B
```

```
3x3 Matrix{Int64}:
 1500  -1500  1500
 -1500  1500  -1500
 1500  -1500  1500
```

Рисунок 3.9: Задание 6

7. Создали матрицу \mathbb{A} размерности 6×6 , все элементы которой равны нулю, и матрицу \mathbb{B} , все элементы которой равны 1. Используя цикл while или for

и закономерности расположения элементов, создайте следующие матрицы размерности 6×6 :

Задача 7

```
# Создание матриц Z, E размером 6x6
Z = zeros(Int, 6, 6)
E = ones(Int, 6, 6)

# Создание матриц Z1, Z2, Z3, Z4
Z1, Z2, Z3, Z4 = copy(Z), copy(Z), copy(Z), copy(Z)

# Заполнение матриц Z1 и Z2
for i in 1:6
    for j in 1:6
        if i == j || i == 7 - j
            Z1[i, j] = 1 - (i + j) % 2
            Z2[i, j] = (i + j) % 2
        end
    end
end

# Заполнение матриц Z3 и Z4
for i in 1:6
    for j in 1:6
        if (i + j) % 2 == 1
            Z3[i, j] = 1
            Z4[7 - i, 7 - j] = 1
        end
    end
end

println("Z1 = ", Z1)
println("Z2 = ", Z2)
println("Z3 = ", Z3)
println("Z4 = ", Z4)

Z1 = [1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0; 0 0 0 1 0 0; 0 0 0 0 1 0; 0 0 0 0 0 1]
Z2 = [0 0 0 0 0 1; 0 0 0 0 1 0; 0 0 0 1 0 0; 0 0 1 0 0 0; 0 1 0 0 0 0; 1 0 0 0 0 0]
Z3 = [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; 1 0 1 0 1 0]
Z4 = [0 1 0 1 0 1; 1 0 1 0 1 0; 0 1 0 1 0 1; 1 0 1 0 1 0]
```

Рисунок 3.10: Задание 7

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). – Написали 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 \mathbb{R} и \mathbb{R} размерностями $n \times m$ и $m \times k$ соответственно – Используя написанную вами функцию `outer()`, создали матрицы следующей структуры
9. Решили следующую систему линейных уравнений с 5 неизвестными

Задача 8

```
function custom_outer(x, y, operation)
    result = zeros(length(x), length(y))
    for i in 1:length(x)
        for j in 1:length(y)
            result[i, j] = operation(x[i], y[j])
        end
    end
    return result
end

# Пример использования функции custom_outer
x = [1, 2, 3]
y = [4, 5, 6]
result = custom_outer(x, y, (a, b) -> a * b) # Пример с умножением
```

```
3×3 Matrix{Float64}:
 4.0  5.0  6.0
 8.0 10.0 12.0
12.0 15.0 18.0
```

Задача 9

```
using LinearAlgebra

# Определение матрицы коэффициентов A и вектора свободных членов y
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]
y = [7, -1, -3, 5, 17]

# Решение системы линейных уравнений
x = A \ y
```

```
5-element Vector{Float64}:
-2.0000000000000036
 3.0000000000000058
 4.999999999999998
 1.9999999999999991
-3.9999999999999999
```

Рисунок 3.11: Задание 8 и 9

- Создали матрицу \mathbb{M} размерности 6×10 , элементами которой являются целые числа, выбранные случайным образом с повторениями из совокупности 1, 2, ..., 10. – Нашли число элементов в каждой строке матрицы \mathbb{M} , которые больше числа \mathbb{N} (например, $\mathbb{N} = 4$). – Определили, в каких строках матрицы \mathbb{M} число \mathbb{N} (например, $\mathbb{N} = 7$) встречается ровно 2 раза? – Определили все пары столб

```
Задача 10

using Random

# Создание матрицы M размером 6x10 со случайными элементами от 1 до 10
M = rand(1:10, 6, 10)

# N задано как 4
N = 4
# Подсчет количества элементов в каждой строке, больших N
count_greater_than_N = [sum(row .> N) for row in eachrow(M)]

# M_value задано как 7
M_value = 7
# Определение строк, где M_value встречается ровно 2 раза
rows_with_M_value_twice = [sum(row .== M_value) == 2 for row in eachrow(M)]

# K задано как 75
K = 75
# Определение пар столбцов, сумма элементов которых больше K
column_pairs_sum_greater_than_K = []
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_sum_greater_than_K, (i, j))
        end
    end
end

# Вывод результатов
println("Матрица M:\n", M)
println("Количество элементов в каждой строке больше N (", N, "): ", count_greater_than_N)
println("Строки, где значение ", M_value, " встречается ровно 2 раза: ", rows_with_M_value_twice)
println("Пары столбцов, сумма элементов которых больше K (", K, "): ", column_pairs_sum_greater_than_K)

Матрица M:
[10 5 1 9 5 8 5 7 5 7; 9 8 4 3 6 2 7 3 2 2; 7 8 1 6 6 3 6 1 5 1; 3 7 2 9 7 6 1 2 9 4; 5 8 3 3 7 7 9 4 8 7; 6 10 7 9 7 5 6 5 8
5]
Количество элементов в каждой строке больше N (4): [9, 4, 6, 5, 7, 10]
Строки, где значение 7 встречается ровно 2 раза: Bool[1, 0, 0, 1, 0, 1]
Пары столбцов, сумма элементов которых больше K (75): Any[(1, 2), (1, 4), (1, 5), (1, 9), (2, 4), (2, 5), (2, 6), (2, 7), (2,
9), (4, 5), (4, 9)]
```

Рисунок 3.12: Задание 10

11. Вычислили суммы выражений

```
Задача 11

# Вычисление первой суммы
sum1 = sum([i^4 / (3 + j) for i in 1:20, j in 1:5])

639215.2833333334

# Вычисление второй суммы
sum2 = sum([i^4 / (3 + i * j) for i in 1:20, j in 1:5])

89912.02146097137
```

Рисунок 3.13: Задание 11

4 Выводы

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