

Лабораторная работа № 3

Управляющие структуры

Беличева Дарья Михайловна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
5	Выводы	17
	Список литературы	18

Список иллюстраций

4.1	Выполнение примеров с циклами	8
4.2	Выполнение примеров с условными выражениями	9
4.3	Выполнение примеров со сторонними библиотеками	10
4.4	Задание №1	11
4.5	Задание №2	11
4.6	Задание №3	12
4.7	Задание №4	12
4.8	Задание №5	13
4.9	Реализация функции outer()	14
4.10	Проверка работы функции outer()	15
4.11	Решение систему линейных уравнений	16
4.12	Задание №10	16
4.13	Задание №11	16

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 (раздел 3.4)

3 Теоретическое введение

Julia – высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений [1]. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков, однако имеет некоторые существенные отличия.

Для выполнения заданий была использована официальная документация Julia [2].

4 Выполнение лабораторной работы

Для начала выполним примеры из лабораторной работы, чтобы познакомиться с циклами, условными операторами, функциями и работой со сторонними библиотеками (рис. 4.1-4.3).

```
Циклы while и for

: n = 0
  while n < 10
    n += 1
    println(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!

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

1
3
```

Рис. 4.1: Выполнение примеров с циклами


```
Условные выражения

3]: # 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

4]: x = 5
    y = 10
    (x > y) ? x : y

4]: 10

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

5]: sayhi (generic function with 1 method)

7]: sayhi("Dasha")

    Hi Dasha, it's great to see you!

9]: # функция возведения в квадрат:
    function f(x)
        x^2
    end
    f(42)
```

Рис. 4.2: Выполнение примеров с условными выражениями

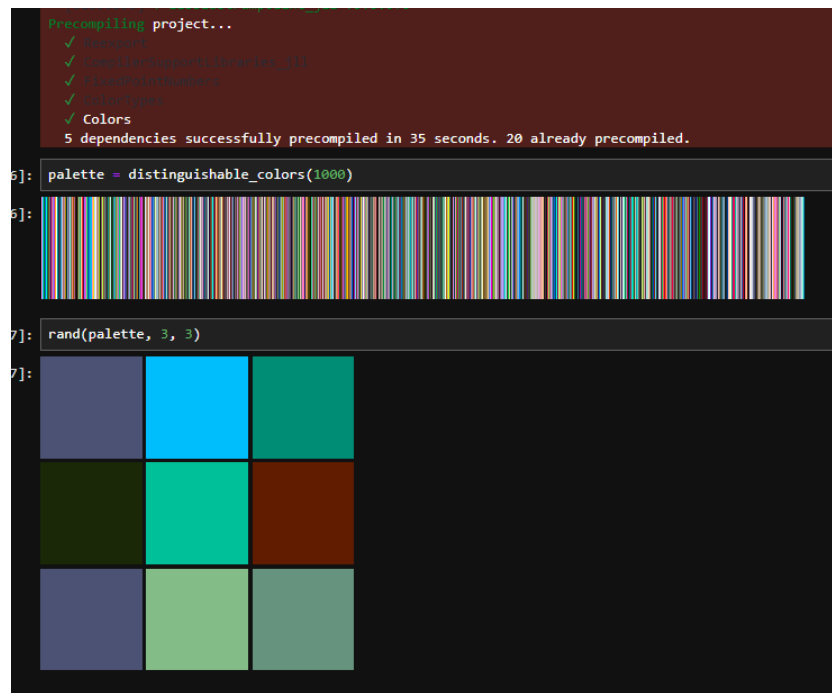


Рис. 4.3: Выполнение примеров со сторонними библиотеками

Теперь перейдем к выполнению заданий для самостоятельной работы.

Используя циклы `while` и `for` (рис. 4.4):

- выведем на экран целые числа от 1 до 100 и напечатаем их квадраты;
- создадим словарь `squares`, который будет содержать целые числа в качестве ключей и квадраты в качестве их пар-значений;
- создадим массив `squares_arr`, содержащий квадраты всех чисел от 1 до 100.

```
1. Используя циклы while и for:

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

68]: print([i for i in 1:100])
      print('\n', "Квадраты:")
      print('\n', [i^2 for i in 1:100])

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]
Квадраты:
[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, 6891, 7060, 7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409, 9604, 9801, 10000]

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

62]: squares = Dict()
      for i in 1:10
        squares[i] = i^2
      end
      print(squares)

Dict{Any, Any}(5 => 25, 4 => 16, 6 => 36, 7 => 49, 2 => 4, 10 => 100, 9 => 81, 8 => 64, 3 => 9, 1 => 1)

66]: squares = Dict{< i -> i^2 for i in 1:10}
      show(squares)

Dict{5 => 25, 4 => 16, 6 => 36, 7 => 49, 2 => 4, 10 => 100, 9 => 81, 8 => 64, 3 => 9, 1 => 1)

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

68]: squares_arr = [i^2 for i in 1:100]
      show(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, 6891, 7060, 7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409, 9604, 9801, 10000]
```

Рис. 4.4: Задание №1

Напишем условный оператор, который печатает число, если число чётное, и строку «нечётное», если число нечётное. Перепишем код, используя тернарный оператор (рис. 4.5).

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

65]: a = readline()
      a = parse{Int64, a}
      if a%2 == 0
        print(a)
      else
        print("Нечетное")
      end

stdin> 7
Нечетное

79]: a = readline()
      a = parse{Int64, a}
      (a%2 == 0) ? a : "Нечетное"

stdin> 2

79]: 2
```

Рис. 4.5: Задание №2

Напишем функцию `add_one`, которая добавляет 1 к своему входу (рис. 4.6).

```

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

[66]: function add_one(x)
      x+1
      end

[66]: add_one (generic function with 1 method)

[68]: add_one(5)

[68]: 6

```

Рис. 4.6: Задание №3

Используем `map()` или `broadcast()` для задания матрицы \square , каждый элемент которой увеличивается на единицу по сравнению с предыдущим. (рис. 4.7)

```

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

[84]: A

[84]: 3x3 Matrix{Int64}:
      1 2 3
      4 5 6
      7 8 9

[87]: map(x -> (x + 1), A)

[87]: 3x3 Matrix{Int64}:
      2 3 4
      5 6 7
      8 9 10

```

Рис. 4.7: Задание №4

Зададим матрицу A. Найдем A^3 . Заменяем третий столбец матрицы \square на сумму второго и третьего столбцов (рис. 4.8).

```
5. Задайте матрицу A.

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

173]: 3x3 Matrix{Int64}:
      1  1  3
      5  2  6
     -2 -1 -3

      – Найдите A3.

174]: g(x) = x^3
      B = g.(A)

174]: 3x3 Matrix{Int64}:
      1  1  27
     125 8 216
     -8 -1 -27

      – Замените третий столбец матрицы A на сумму второго и третьего столбцов.

171]: for i in 1:3
      A[i,3] = A[i,1] + A[i,2]
      end
      A

171]: 3x3 Matrix{Int64}:
      1  1  2
      5  2  7
     -2 -1 -3

165]: m, n = 2, 3
      A = fill(1, (m, n))

165]: 2x3 Matrix{Int64}:
      1  1  1
      1  1  1
```

Рис. 4.8: Задание №5

Напишем свою функцию, аналогичную функции `outer()` языка R. Функция должна иметь следующий интерфейс: `outer(x, y, operation)` (рис. 4.9, 4.10).

```

[35]: function outer(x, y, f)
      if (f == "*")
          res = x .* y
      elseif (f == "+")
          res = x .+ y
      elseif (f == "^")
          res = x .^ y
      elseif (f == "-")
          res = x .- y
      elseif (f == "%")
          res = x .% y
      end
  end

[35]: outer (generic function with 1 method)

[25]: A1 = outer(0:4, 0:4, "+")
      A1

[25]: 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

[34]: A2 = outer(0:4, 1:5, "^")
      A2

[34]: 5x5 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

[37]: A3 = outer(0:4, 0:4, "+")
      A3_res = outer(A3, 5, "%")

[37]: 5x5 Matrix{Int64}:
      0  1  2  3  4
      1  2  3  4  0
      2  3  4  0  1

```

Рис. 4.9: Реализация функции outer()

```

: A3 = outer(0:4, 0:4, "+")
  A3_res = outer(A3, 5, "%")

: 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

: A3 = outer(0:9, 0:9, "+")
  A3_res = outer(A3, 10, "%")

: 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

: A3 = outer(0:9, 0:9, "-")
  A3 = abs.(A3)

: 10x10 Matrix{Int64}:
  0  1  2  3  4  5  6  7  8  9
  1  0  1  2  3  4  5  6  7  8
  2  1  0  1  2  3  4  5  6  7
  3  2  1  0  1  2  3  4  5  6
  4  3  2  1  0  1  2  3  4  5
  5  4  3  2  1  0  1  2  3  4
  6  5  4  3  2  1  0  1  2  3
  7  6  5  4  3  2  1  0  1  2
  8  7  6  5  4  3  2  1  0  1
  9  8  7  6  5  4  3  2  1  0

```

Рис. 4.10: Проверка работы функции outer()

Решим систему линейных уравнений с 5 неизвестными (рис. 4.11).

```

4]: 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]

x = A \ y
println("Решение системы: ")
println(x)

Решение системы:
[-2.00000000000000036, 3.00000000000000058, 4.999999999999998, 1.9999999999999991, -3.999999999999999]

```

Рис. 4.11: Решение систему линейных уравнений

В 10 задании произведем анализ количества элементов матрицы, удовлетворяющих необходимым условиям (рис. 4.12).

```

]: M = rand(1:10, 6, 10)
println(M)
N = 4
K = 75
count_1 = sum(M.>N)
println(count_1)
count_2 = [i for i=1:6 if sum(M[i,:].==2)]
println(count_2)
count_3 = [(i, j) for i = 1:6, j = 2:5 if (i != j && sum(M[:, i] + M[:, j]) > K)]
println(count_3)

[5 2 4 2 10 7 8 5 3 10; 9 4 9 7 10 8 7 3 8 8; 9 4 2 5 2 4 4 9 2 3; 4 5 6 2 9 6 7 3 9 2; 7 10 10 8 1 9 8 9 6 3; 1 4 9 3 6 2 4 9 3 9]
35
[2]
[(5, 3), (6, 3), (3, 5)]

```

Рис. 4.12: Задание №10

Вычислим выражения (рис. 4.13).

```

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

sum2 = sum(i^4 / (3 + i * j) for i in 1:20, j in 1:5)

println("Первая сумма: $sum1")
println("Вторая сумма: $sum2")

Первая сумма: 639215.2833333338
Вторая сумма: 89912.02146097131

```

Рис. 4.13: Задание №11

5 Выводы

В результате выполнения данной лабораторной работы я освоила применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.

Список литературы

1. JuliaLang [Электронный ресурс]. 2024 JuliaLang.org contributors. URL: <https://julialang.org/> (дата обращения: 11.10.2024).
2. Julia 1.11 Documentation [Электронный ресурс]. 2024 JuliaLang.org contributors. URL: <https://docs.julialang.org/en/v1/> (дата обращения: 11.10.2024).