Компьютерный практикум по статистическому анализу данных

Отчёт по лабораторной работе №3: Управляющие структуры

Кармацкий Никита Сергеевич

Содержание

1 Цель работы **5**

2 Выполнение лабораторной работы **6**
2.1 Циклы while и for 6
2.2 Условные выражения 12
2.3 Функции 14
2.4 Сторонние библиотеки 20
2.5 Самостоятельная работа 23

3 Вывод **52**

4 Список литературы. Библиография **53**

List of Figures

2.1 Примеры использования цикла while 7
2.2 Примеры использования цикла for 9
2.3 Примеры использования цикла for для создания двумерного массива 11
2.4 Пример использования условного выражения 13
2.5 Пример способ написания функции 15
2.6 Сравнение результатов вывода 17
2.7 Пример использования функций map() и broadcast() 19
2.8 Пример использвания стронних библиотек 22
2.9 Выполнение подпунктов задания №1 24
2.10 Выполнение подпунктов задания №1 25
2.11 Выполнение подпунктов задания №1 26
2.12 Выполнение подпунктов задания №1 27
2.13 Выполнение задания №2 29
2.14 Выполнение задания №3 31
2.15 Выполнение задания №4 33
2.16 Выполнение задания №5 35
2.17 Выполнение задания №6 37
2.18 Выполнение задания №7 39
2.19 Выполнение задания №7 40
2.20 Выполнение подпунктов задания №8 42
2.21 Выполнение подпунктов задания №8 43
2.22 Выполнение подпунктов задания №8 44
2.23 Выполнение задания №9 46
2.24 Выполнение подпунктов задания №10 48
2.25 Выполнение подпунктов задания №10 49
2.26 Выполнение задания №11 51

List of Tables

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 while и for

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

Синтаксис while

```
while <условие>
    <тело цикла>
end
```

Пример использования цикла while (рис. [-fig@:001]):

Циклы while и for

[3]:
```
n = 0
while n < 10
    n+=1
    println(n)
end
```

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

[5]:
```
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: Примеры использования цикла while

Такие же результаты можно получить при использование цикла for

Синтаксис for

```
for <переменная> in <диапазон>
    <тело цикла>
end
```

Пример использования цикла for (рис. [-fig@:002]):

```
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!
```

Рис. 2.2: Примеры использования цикла for

Пример использования цикла for для создания двумерного массива, где значение каждой записи является суммой индексов строки и столбца(рис. [-fig@:003]):

```
[11]: 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
      A
```

```
[11]: 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
```

```
[17]: B = fill(0, (m,n))
      for i in 1:m, j in 1:n
          B[i,j] = i + j
      end
      B
```

```
[17]: 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
```

```
[19]: C = [i + j for i in 1:m, j in 1:n]
      C
```

```
[19]: 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
```

Рис. 2.3: Примеры использования цикла for для создания двумерного массива

2.2 Условные выражения

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

Синтаксис условных выражений с ключевым словом:

```
if <условие 1>
  <Действие 1>
elseif <Условие 2>
  <Действие 2>
else
  <Действие3>
end
```

Пример использования условного выражения(рис.[-fig@:004]):

Условные выражения

[26]:
```
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
```

[28]:
```
x = 5
y = 10

(x > y) ? x : y
```

[28]:
```
10
```

Рис. 2.4: Пример использования условного выражения

2.3 Функции

Julia дает нам несколько разных способов написать функцию.

Пример способ написания функции(рис.[-fig@:005]):

Функции

[31]:
```
function sayhi(name)
    println("Hi $name, it's great to see you!")
end

function f(x)
    x^2
end
sayhi("C-3PO")
f(4)
```

```
Hi C-3PO, it's great to see you!
```

[31]: 16

[33]:
```
sayhi2(name) = println("Hi $name, it's great to see you!")
f2(x) = x^2

sayhi("C-3PO")
f(4)
```

```
Hi C-3PO, it's great to see you!
```

[33]: 16

Рис. 2.5: Пример способ написания функции

По соглашению в Julia функции, сопровождаемые восклицательным знаком, изменяют свое содержимое, а функции без восклицательного знака не делают этого(рис.[-fig@:006]):

```
[35]: v = [3, 5, 2]
      sort(v)
      v
```

```
[35]: 3-element Vector{Int64}:
       3
       5
       2
```

```
[37]: sort!(v)
      v
```

```
[37]: 3-element Vector{Int64}:
       2
       3
       5
```

Рис. 2.6: Сравнение результатов вывода

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.

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.

Пример использования функций map() и broadcast()(рис.[-fig@:007]):

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

```
[39]: 3-element Vector{Int64}:
        1
        8
       27
```

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

```
[41]: 3-element Vector{Int64}:
        1
        8
       27
```

Рис. 2.7: Пример использования функций map() и broadcast()

2.4 Сторонние библиотеки

Julia имеет более 2000 зарегистрированных пакетов, что делает их огромной частью экосистемы Julia. Есть вызовы функций первого класса для других языков, обеспечивающие интерфейсы сторонних функций. Можно вызвать функции из Python или R, например, с помощью PyCall или Rcall.

С перечнем доступных в Julia пакетов можно ознакомиться на страницах следующих ресурсов: - https://julialang.org/packages/ - https://juliahub.com/ui/Home - https://juliaobserver.com/ - https://github.com/svaksha/Julia.jl

При первом использовании пакета в вашей текущей установке Julia вам необходимо использовать менеджер пакетов, чтобы явно его добавить:

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

При каждом новом использовании Julia (например, в начале нового сеанса в REPL или открытии блокнота в первый раз) нужно загрузить пакет, используя ключевое слово using:

Например, 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)
```

Пример испольвания стронних библиотек (рис.[-fig@:008]):

[45]:
```
import Pkg
Pkg.add("Colors")
using Colors
palette = distinguishable_colors(100)
rand(palette, 3, 3)
```

```
  Resolving package versions...
 No Changes to `~/.julia/environments/v1.11/Project.toml`
 No Changes to `~/.julia/environments/v1.11/Manifest.toml`
```

[45]:

Рис. 2.8: Пример использвания стронних библиотек

2.5 Самостоятельная работа

Выполнения здания №1 (рис.[-fig@:009] - рис.[-fig@:0012]):

№1. Используя циклы while и for:

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

[52]:
```
n = 1
while n <= 100
    println("$n^2 = ", n^2)
    n += 1
end
```

```
1^2 = 1
2^2 = 4
3^2 = 9
4^2 = 16
5^2 = 25
6^2 = 36
7^2 = 49
8^2 = 64
9^2 = 81
10^2 = 100
11^2 = 121
12^2 = 144
13^2 = 169
14^2 = 196
15^2 = 225
16^2 = 256
17^2 = 289
18^2 = 324
19^2 = 361
20^2 = 400
21^2 = 441
22^2 = 484
23^2 = 529
24^2 = 576
```

Рис. 2.9: Выполнение подпунктов задания №1

[54]:
```
for n in 1:100
    println("$n^2 = ", n^2)
end
```

```
1^2 = 1
2^2 = 4
3^2 = 9
4^2 = 16
5^2 = 25
6^2 = 36
7^2 = 49
8^2 = 64
9^2 = 81
10^2 = 100
11^2 = 121
12^2 = 144
13^2 = 169
14^2 = 196
15^2 = 225
16^2 = 256
17^2 = 289
18^2 = 324
19^2 = 361
20^2 = 400
21^2 = 441
22^2 = 484
23^2 = 529
24^2 = 576
25^2 = 625
26^2 = 676
27^2 = 729
```

Рис. 2.10: Выполнение подпунктов задания №1

1.2) Создайте словарь squares, который будет содержать целые числа в качестве ключей и квадраты в качестве их пар-значений:

[58]:
```
squeres = Dict()
for n in 1:100
    squeres[n] = n^2
end
println(squeres)
```

```
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 => 73
96, 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, 3
9 => 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 => 280
9, 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. 2.11: Выполнение подпунктов задания №1

1.3) Создайте массив squares_arr, содержащий квадраты всех чисел от 1 до 100:

```
[60]: 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, 1
936, 2025, 2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249, 3364, 3481, 3600, 3721, 3
844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5041, 5184, 5329, 5476, 5625, 5776, 5929, 6084, 6241, 6
400, 6561, 6724, 6889, 7056, 7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409, 9
604, 9801, 10000]
```

Рис. 2.12: Выполнение подпунктов задания №1

Выполнения здания №2 (рис.[-fig@:013]):

№2. Напишите условный оператор, который печатает число, если число чётное, и строку «нечётное», если число нечётное. Перепишите код, используя тернарный оператор:

```
[67]: n = 42
if n % 2 ==0
    println(n)
else
    println("Нечетное число")
end
```

```
42
```

```
[69]: println(n % 2 == 0 ? n : "нечётное")
```

```
42
```

Рис. 2.13: Выполнение задания №2

Выполнения здания №3 (рис.[-fig@:014]):

№3. Напишите функцию add_one, которая добавляет 1 к своему входу:

```
[72]: function add_one(x)
          return x + 1
      end
      println(add_one(4))
```

```
5
```

Рис. 2.14: Выполнение задания №3

Выполнения здания №4 (рис.[-fig@:015]):

№4. Используйте map() или broadcast() для задания матрицы A, каждый элемент которой увеличивается на единицу по сравнению с предыдущим:

```
[75]: #map

A = reshape(1:9, 3, 3)
B = map(x -> x + 1, A)
println(B)
```

```
[2 5 8; 3 6 9; 4 7 10]
```

```
[77]: B = broadcast(x -> x + 1, A)
println(B)
```

```
[2 5 8; 3 6 9; 4 7 10]
```

Рис. 2.15: Выполнение задания №4

Выполнения здания №5 (рис.[-fig@:016]):

№5. Задайте матрицу A следующего вида. Найдите A^3. Замените третий столбец матрицы A на сумму второго и третьего столбцов:

```
(82): A = [1 1 3; 5 2 6; -2 -1 -3]

println(map(x -> x^3, A))
```

```
[1 1 27; 125 8 216; -8 -1 -27]
```

```
(88): A[:,3] = A[:,2] + A[:,3]
println(A)
```

```
[1 1 4; 5 2 8; 8 -1 -4]
```

Рис. 2.16: Выполнение задания №5

Выполнения здания №6 (рис.[-fig@:017]):

№6. Создайте матрицу B с элементами $Bi1 = 10$, $Bi2 = -10$, $Bi3 = 10$, $i = 1, 2, \ldots, 15$. Вычислите матрицу $C = B^{\wedge}TB$:

[91]:
```
B = repeat([10 -10 10], 15, 1)
```

[91]:
```
15×3 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
```

[93]:
```
C = B' * B
println(C)
```

```
[1500 -1500 1500; -1500 1500 -1500; 1500 -1500 1500]
```

Рис. 2.17: Выполнение задания №6

Выполнения здания №7 (рис.[-fig@:018] - рис.[-fig@:019]):

№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:

[184]:
```
n = 6
Z1 = zeros(Int, n, n)
Z2 = zeros(Int, n, n)
Z3 = zeros(Int, n, n)
Z4 = zeros(Int, n, n)

function print_matrix(mat)
    for row in eachrow(mat)
        println(row)
    end
    println()
end

# Построение Z1: 1 вокруг главной диагонали
for i in 1:n
    for j in 1:n
        if abs(i - j) == 1
            Z1[i, j] = 1
        end
    end
end

# Построение Z2: 1 на главной диагонали и вокруг нее
for i in 1:n
    for j in 1:n
        if abs(i - j) <= 1
            Z2[i, j] = 1
        end
    end
end

# Построение Z3: 1 на побочной диагонали и вокруг нее
for i in 1:n
    for j in 1:n
        if abs((i + j) - (n + 1)) <= 1
            Z3[i, j] = 1
        end
    end
end

# Построение Z4: 1 на позициях с четной суммой индексов
for i in 1:n
    for j in 1:n
        if (i + j) % 2 == 0
            Z4[i, j] = 1
        end
    end
end

# Вывод результатов
println("Матрица Z1:")
print_matrix(Z1)
println("Матрица Z2:")
```

Рис. 2.18: Выполнение задания №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]

Матрица Z2:
[1, 1, 0, 0, 0, 0]
[1, 1, 1, 0, 0, 0]
[0, 1, 1, 1, 0, 0]
[0, 0, 1, 1, 1, 0]
[0, 0, 0, 1, 1, 1]
[0, 0, 0, 0, 1, 1]

Матрица Z3:
[0, 0, 0, 0, 1, 1]
[0, 0, 0, 1, 1, 1]
[0, 0, 1, 1, 1, 0]
[0, 1, 1, 1, 0, 0]
[1, 1, 1, 0, 0, 0]
[1, 1, 0, 0, 0, 0]

Матрица Z4:
[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. 2.19: Выполнение задания №7

Выполнения здания №8 (рис.[-fig@:020] - рис.[-fig@:022]):

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

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

```
[118]: outer (generic function with 1 method)
```

Рис. 2.20: Выполнение подпунктов задания №8

8.2) Используя написанную вами функцию outer(), создайте матрицы следующей структуры:

[135]:
```
A1 = outer(0:4, 0:4, +)

function safe_pow(x, y)
    x == 0 && y == 0 ? 0 : x^y
end

A2 = [j ==1 ? i : safe_pow(i, j) for i in 0:4, j in 1:5]

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

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

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

function print_matrix(name, mat)
    println("\nМатрица $name:")
    for row in eachrow(mat)
        println(row)
    end
end

print_matrix("A1", A1)
print_matrix("A2", A2)
print_matrix("A3", A3)
print_matrix("A4", A4)
print_matrix("A5", A5)
```

Рис. 2.21: Выполнение подпунктов задания №8

```
Матрица A1:
[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]

Матрица A2:
[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]

Матрица A3:
[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]

Матрица A4:
[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]

Матрица A5:
[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. 2.22: Выполнение подпунктов задания №8

Выполнения здания №9 (рис.[-fig@:023]):

№9. Решите следующую систему линейных уравнений с 5 неизвестными:

```
[138]: A = [ 1 2 3 4 5;
             2 1 2 3 4;
             3 2 1 3 4;
             4 3 2 1 2;
             5 4 3 2 1]
b = [7, -1, -3, 5, -6]

x = A \ b

print(x)
```

```
[-3.9166666666666687, 4.305555555555559, 2.3888888888888875, -8.194444444444445, 5.583333333333334]
```

Рис. 2.23: Выполнение задания №9

Выполнения здания №10 (рис.[-fig@:024] - рис.[-fig@:025]):

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

[152]:
```
function print_matrix(name, mat)
    println("\nМатрица $name:")
    for row in eachrow(mat)
        println(row)
    end
end

M = rand(1:10, 6, 10)

print_matrix(M)
```

```
[9, 9, 6, 9, 2, 8, 7, 5, 9, 1]
[7, 3, 3, 5, 7, 2, 10, 7, 7, 5]
[10, 9, 7, 4, 7, 5, 6, 7, 4, 3]
[5, 10, 9, 8, 2, 9, 7, 5, 6, 7]
[2, 9, 3, 6, 7, 2, 7, 9, 1, 3]
[1, 4, 8, 5, 2, 1, 5, 9, 5, 1]
```

Рис. 2.24: Выполнение подпунктов задания №10

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

[155]:
```
N = 4
greater_then_N = sum(M .> N, dims=2)
println(greater_then_N)
```

```
[8; 7; 7; 9; 5; 5;;]
```

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

[160]:
```
M_value = 7
rows = findall( x -> count(==(M_value), x) ==2, eachrow(M))
println(rows)
```

```
[4, 5]
```

10.3) Определите все пары столбцов матрицы M, сумма элементов которых больше K (например, $K = 75$):

[165]:
```
k = 75
col_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!(col_pairs, (i,j))
        end
    end
end
println(col_pairs)
```

```
Any[(1, 2), (1, 7), (1, 8), (2, 3), (2, 4), (2, 7), (2, 8), (2, 9), (3, 7), (3, 8), (4, 7), (4, 8), (7, 8)]
```

Рис. 2.25: Выполнение подпунктов задания №10

Выполнения здания №11 (рис.[-fig@:026]):

№11. Вычислите

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

```
[172]: 216749980
```

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

```
[174]: 1958839480
```

Рис. 2.26: Выполнение задания №11

3 Вывод

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

4 Список литературы. Библиография

[1] Mininet: https://mininet.org/