

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

дисциплина: «Компьютерный практикум по статистическому
анализу данных»

Работу выполнил:

Снимщиков Иван Игоревич

Группа: НПИбд-02-21

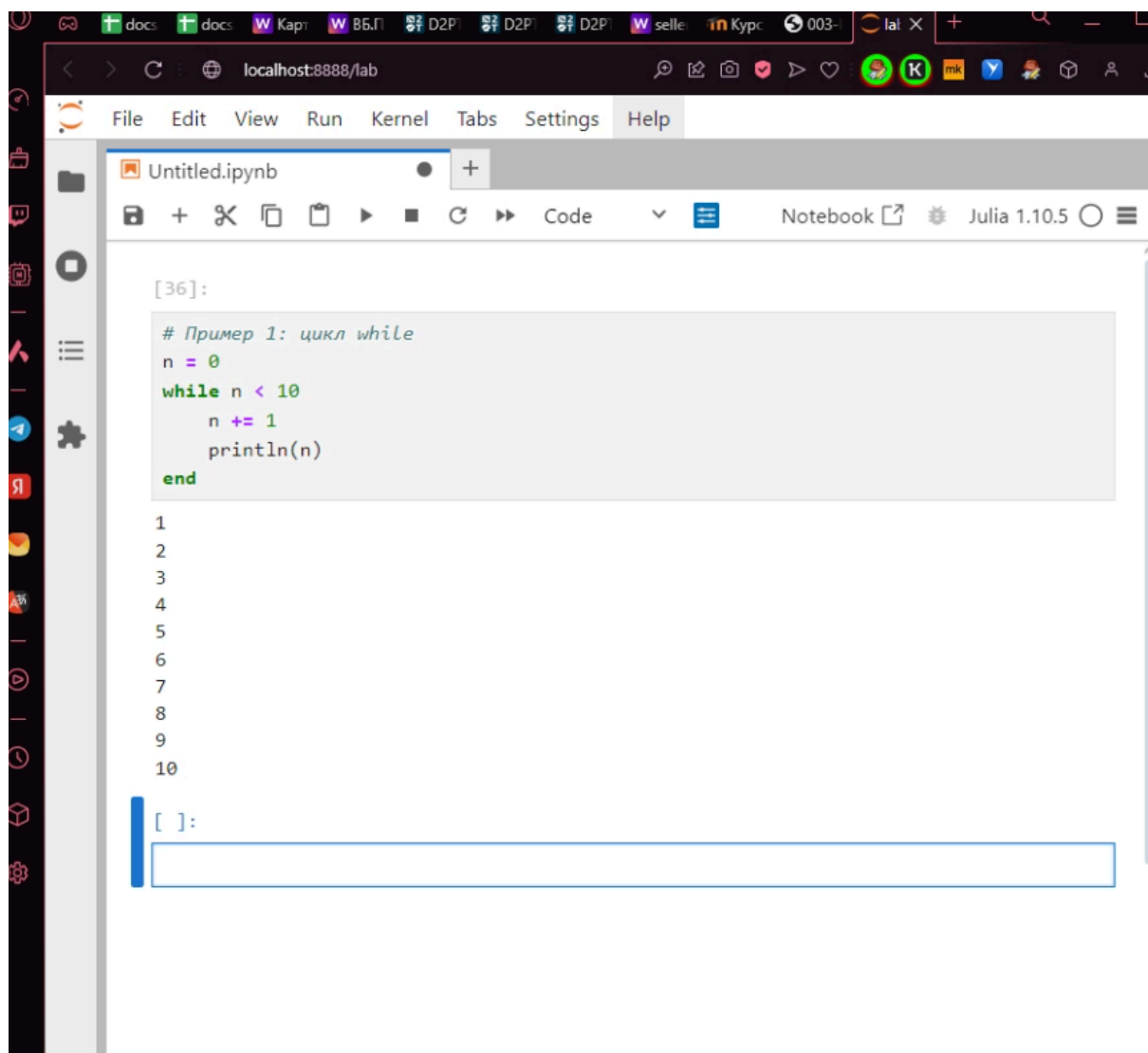
МОСКВА

2024 г.

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

Ход работы:

Первым делом я повторил примеры из файла лабораторной работы



The screenshot shows a web-based Julia REPL interface. The browser address bar displays 'localhost:8888/lab'. The interface includes a menu bar with 'File', 'Edit', 'View', 'Run', 'Kernel', 'Tabs', 'Settings', and 'Help'. Below the menu is a toolbar with icons for file operations and execution. The main area shows a code cell with the following Julia code:

```
[36]:  
  
# Пример 1: цикл while  
n = 0  
while n < 10  
    n += 1  
    println(n)  
end
```

The output of the code is displayed below the code cell, showing the numbers 1 through 10 on separate lines. At the bottom of the code cell, there is an input prompt '[]:' followed by an empty text box for entering new code.

[39]:

```
# Пример 4: цикл for с массивом строк
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!

[]:

[43]:

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

sayhi("Ваня")
```

Hi Ваня, it's great to see you!

[]:

Далее я перешел к самостоятельной работе

While и For:

[1]:

```
# Вывод чисел от 1 до 100 и их квадратов
println("Числа от 1 до 100 и их квадраты:")
for n in 1:100
    println("Число: $n, Квадрат: $(n^2)")
end
```

Числа от 1 до 100 и их квадраты:

Число: 1, Квадрат: 1

Число: 2, Квадрат: 4

Число: 3, Квадрат: 9

Число: 4, Квадрат: 16

Число: 5, Квадрат: 25

Число: 6, Квадрат: 36

Число: 7, Квадрат: 49

Число: 8, Квадрат: 64

Число: 9, Квадрат: 81

[2]:

```
# Создание словаря squares
squares = Dict{<
for n in 1:100
    squares[n] = n^2
end
println("Словарь квадратов: ", squares)
```

Словарь квадратов: 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}

[]:

Проверка четности при помощи условного и тернарного операторов.

[4]:

```
# Условный оператор для проверки четности
n = 5
if n % 2 == 0
    println("$n - четное")
else
    println("$n - нечетное")
end
```

5 - нечетное

[6]:

```
# Тернарный оператор
n = 5
result = (n % 2 == 0) ? "$n - четное" : "$n - нечетное"
println(result)
```

5 - нечетное

[1]:

[7]:

```
# Функция, добавляющая 1 к числу
function add_one(x)
    return x + 1
end
println("add_one(5): ", add_one(5))
```

add_one(5): 6

[]:

Функции:

[7]:

```
# Функция, добавляющая 1 к числу
function add_one(x)
    return x + 1
end
println("add_one(5): ", add_one(5))
```

add_one(5): 6

Матрицы:

[10]:

```
# Задаем матрицу A
A = [
    1 1 3;
    5 2 6;
    -2 -1 -3
]
println("Исходная матрица A: ", A)

# Вычисление A^3
A_cubed = A^3
println("A^3: ", A_cubed)

# Замена третьего столбца на сумму второго и третьего
A[:, 3] = A[:, 2] + A[:, 3]
println("Матрица A после замены третьего столбца: ", A)
```

Исходная матрица A: [1 1 3; 5 2 6; -2 -1 -3]

A^3: [0 0 0; 0 0 0; 0 0 0]

Матрица A после замены третьего столбца: [1 1 4; 5 2 8; -2 -1 -4]

localhost:8889/lab

File Edit View Run Kernel Tabs Settings Help

Untitled.ipynb

Code Notebook Julia 1.10.5

```
for i in 1:6
    for j in 1:6
        if abs(i - j) <= 1
            Z2[i, j] = 1
        end
    end
end
println("Матрица Z2: ", Z2)

# Создание Z3
Z3 = copy(Z)
for i in 1:6
    for j in 1:6
        if (i + j) % 2 == 0
            Z3[i, j] = 1
        end
    end
end
println("Матрица Z3: ", Z3)

# Создание Z4
Z4 = [mod(i + j, 2) == 0 ? 1 : 0 for i in 1:6, j in 1:6]
println("Матрица Z4: ", Z4)
```

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: [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]

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]

[]:

Would you like to get notified about official Jupyter news?

[Open privacy policy](#) Yes No

Реализация функции Outer:

[]:

```
# Реализация функции outer
function outer(x, y, operation)
    return [operation(x[i], y[j]) for i in 1:length(x), j in 1:length(y)]
end

# Примеры работы
x = 0:4
y = 0:4
result_add = outer(x, y, +)
println("Результат сложения: ", result_add)
result_mul = outer(x, y, *)
println("Результат умножения: ", result_mul)
```

Решение системы:

[13]:

```
# Матрица A и вектор 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 Ax = y
x = A \ y
println("Решение системы: ", x)
```

Решение системы: [-2.0000000000000036, 3.0000000000000058, 4.999999999999998, 1.9999999999999991, -3.999999999999999]

[]:

Работа со случайной матрицей:

[14]:

```
# Создание случайной матрицы M
using Random
M = rand(1:10, 6, 10)
println("Матрица M: ", M)

# Число элементов в каждой строке больше 4
count_greater_4 = [sum(x -> x > 4, row) for row in eachrow(M)]
println("Элементы больше 4 в каждой строке: ", count_greater_4)

# Строки, где число 7 встречается ровно 2 раза
rows_with_two_7s = findall(row -> count(x -> x == 7, row) == 2, eachrow(M))
println("Строки, где 7 встречается ровно 2 раза: ", rows_with_two_7s)
```

Матрица M: [1 5 3 6 2 4 8 10 6 1; 3 1 8 2 1 9 4 8 3 1; 7 2 10 10 8 7 2 4 2 3; 10 7
1 10 5 9 1 3 7 3; 4 7 10 6 1 7 1 8 6 6; 7 5 7 3 8 10 10 4 4 8]
Элементы больше 4 в каждой строке: [5, 3, 5, 6, 7, 7]
Строки, где 7 встречается ровно 2 раза: [3, 4, 5, 6]

[]:

Решение математического примера:

[15]:

```
# Первая сумма
sum1 = sum(i^4 / (3 + j) for i in 1:20, j in 1:5)
println("Первая сумма: ", sum1)

# Вторая сумма
sum2 = sum(i^4 / (3 + i*j) for i in 1:20, j in 1:5)
println("Вторая сумма: ", sum2)
```

Первая сумма: 639215.2833333338

Вторая сумма: 89912.02146097131

[]:

Вывод: Я освоил применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.

