

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

дисциплина: «Компьютерный практикум по статистическому
анализу данных»

Работу выполнил:

Снимщиков Иван Игоревич

Группа: НПИбд-02-21

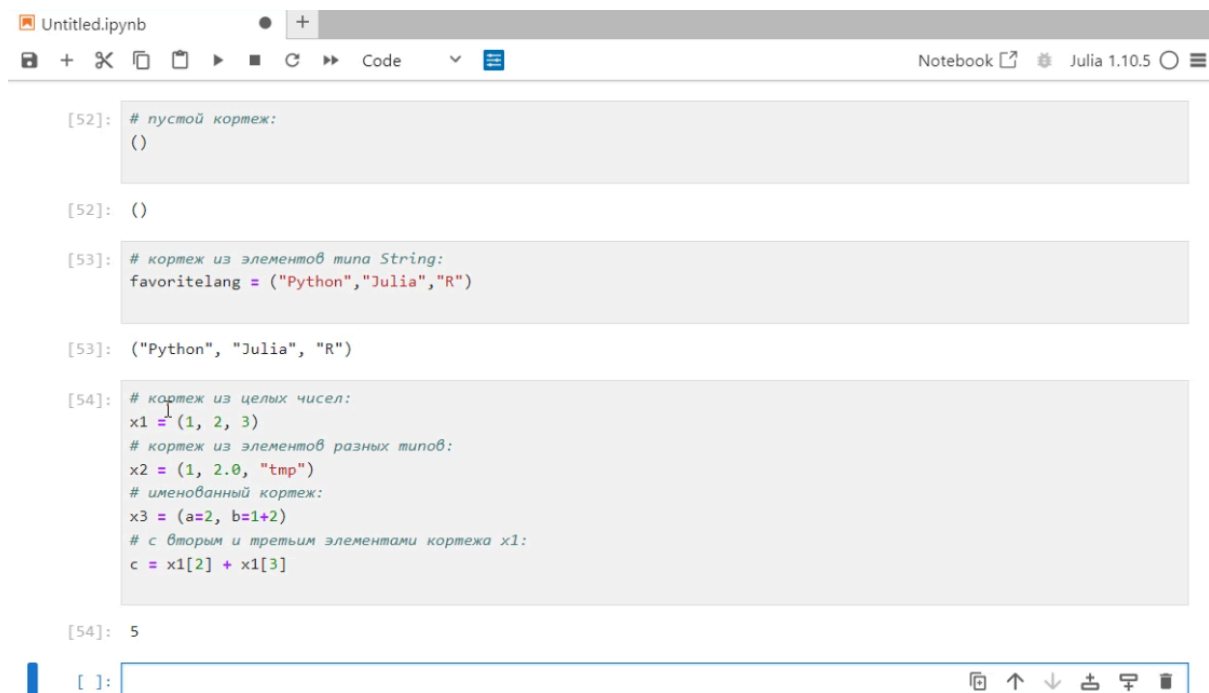
МОСКВА

2024 г.

Цели работы: Основная цель работы — изучить несколько структур данных, реализованных в Julia, научиться применять их и операции над ними для решения задач.

Ход работы:

Первым делом я повторил примеры из файла лабораторной работы №2



The screenshot shows a Julia notebook interface with the title 'Untitled.ipynb'. The toolbar includes icons for file operations, a 'Code' dropdown, and a 'Julia 1.10.5' version indicator. The notebook contains three code cells. The first cell, labeled '[52]:', contains a comment '# пустой кортеж:' followed by an empty tuple '()'. The second cell, also labeled '[52]:', contains an empty tuple '()'. The third cell, labeled '[53]:', contains a comment '# кортеж из элементов типа String:', a variable assignment 'favoritelang = ("Python", "Julia", "R")', and the output '("Python", "Julia", "R")'. Below these, a fourth cell labeled '[54]:' contains several comments and code lines: '# кортеж из целых чисел:', 'x1 = (1, 2, 3)', '# кортеж из элементов разных типов:', 'x2 = (1, 2.0, "tmp")', '# именованный кортеж:', 'x3 = (a=2, b=1+2)', '# с вторым и третьим элементами кортежа x1:', and 'c = x1[2] + x1[3]'. The output of this cell is '5'. At the bottom, there is an input prompt '[]:' followed by an empty text box and a set of icons for cell manipulation.

```
[52]: # пустой кортеж:
      ()

[52]: ()

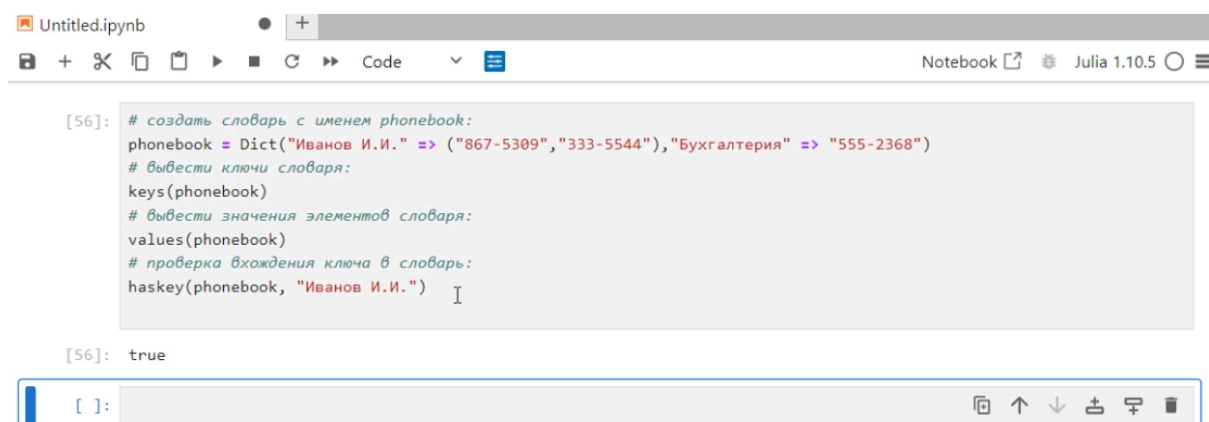
[53]: # кортеж из элементов типа String:
      favoritelang = ("Python", "Julia", "R")

[53]: ("Python", "Julia", "R")

[54]: # кортеж из целых чисел:
      x1 = (1, 2, 3)
      # кортеж из элементов разных типов:
      x2 = (1, 2.0, "tmp")
      # именованный кортеж:
      x3 = (a=2, b=1+2)
      # с вторым и третьим элементами кортежа x1:
      c = x1[2] + x1[3]

[54]: 5

[ ]:
```



The screenshot shows a Julia notebook interface with the title 'Untitled.ipynb'. The toolbar includes icons for file operations, a 'Code' dropdown, and a 'Julia 1.10.5' version indicator. The notebook contains one code cell labeled '[56]:' with the following content: a comment '# создать словарь с именем phonebook:', a dictionary assignment 'phonebook = Dict{"Иванов И.И." => ("867-5309", "333-5544"), "Бухгалтерия" => "555-2368"}', a comment '# вывести ключи словаря:', 'keys(phonebook)', a comment '# вывести значения элементов словаря:', 'values(phonebook)', a comment '# проверка вхождения ключа в словарь:', and 'haskey(phonebook, "Иванов И.И.")'. The output of this cell is 'true'. At the bottom, there is an input prompt '[]:' followed by an empty text box and a set of icons for cell manipulation.

```
[56]: # создать словарь с именем phonebook:
      phonebook = Dict{"Иванов И.И." => ("867-5309", "333-5544"), "Бухгалтерия" => "555-2368"}
      # вывести ключи словаря:
      keys(phonebook)
      # вывести значения элементов словаря:
      values(phonebook)
      # проверка вхождения ключа в словарь:
      haskey(phonebook, "Иванов И.И.")

[56]: true

[ ]:
```

```
[59]: # массив из квадратных корней всех целых чисел от 1 до 10:  
roots = [sqrt(i) for i in 1:10]
```

```
[59]: 10-element Vector{Float64}:  
 1.0  
 1.4142135623730951  
 1.7320508075688772  
 2.0  
 2.23606797749979  
 2.449489742783178  
 2.6457513110645907  
 2.8284271247461903  
 3.0  
 3.1622776601683795
```

```
[ ]:
```



Далее я перешел к выполнению самостоятельной работы из файла лабораторной работы №2

Пункт первый:

```
[61]: #объединение пересечений заданных множеств  
A = Set([0, 3, 4, 9])  
B = Set([1, 3, 4, 7])  
C = Set([0, 1, 2, 4, 7, 8, 9])  
  
P = union(intersect(A, B), intersect(A, C), intersect(B, C))
```

```
[61]: Set{Int64} with 6 elements:  
 0  
 4  
 7  
 9  
 3  
 1
```

```
[ ]:
```



Пункт второй:

```
[62]: set1 = Set([1, 2, "abc"])  
      set2 = Set(["abc", 4, 5])  
  
      # Операции над множествами  
      union_result = union(set1, set2) # Объединение  
      intersect_result = intersect(set1, set2) # Пересечение  
      setdiff_result = setdiff(set1, set2) # Разность
```

```
[62]: Set{Any} with 2 elements:  
 2  
 1
```

```
[ ]:
```

Пункт третий: (со всеми пунктами можно ознакомиться в записи выполнения лабораторной работы, в отчете указал примеры выполненной работы на нескольких пунктах)

```
[64]: N = 25  
array1 = collect(20:N)  
  
[64]: 6-element Vector{Int64}:  
 20  
 21  
 22  
 23  
 24  
 25
```

```
[69]: N = 25  
array1 = collect(21:N)  
array2 = reverse(array1)
```

```
[69]: 5-element Vector{Int64}:  
 25  
 24  
 23  
 22  
 21
```

```
[ ]:
```

```
Untitled.ipynb  
array4 = repeat([tmp[1]], 10)  
array5 = repeat(tmp, 10)  
array6 = vcat(repeat([tmp[1]], 11), repeat([tmp[2]], 10), repeat([tmp[3]], 10))  
  
[74]: 31-element Vector{Int64}:  
 4  
 4  
 4  
 4  
 4  
 4  
 4  
 4  
 4  
 4  
 6  
 6  
 6  
 6  
 6  
 3  
 3  
 3  
 3  
 3  
 3  
 3  
 3  
 3  
 3  
 3
```

```
[77]: N = 25
array1 = collect(21:N)
tmp = [4, 6, 3]
array4 = repeat([tmp[1]], 10)
array5 = repeat(tmp, 10)
array6 = vcat(repeat([tmp[1]], 11), repeat([tmp[2]], 10), repeat([tmp[3]], 10))
array7 = vcat(repeat([tmp[1]], 10), repeat([tmp[2]], 20), repeat([tmp[3]], 30))
array8 = [2 * tmp[i] for i in 1:3]
array8 = vcat(array8, repeat([array8[3]], 4))
count_6 = count(x -> x == 6, array8) # Количество шестерок
using Statistics
array9 = [exp(x) * cos(x) for x in 3:0.1:6]
mean_value = mean(array9)
```

```
[77]: 53 11374594647971
```

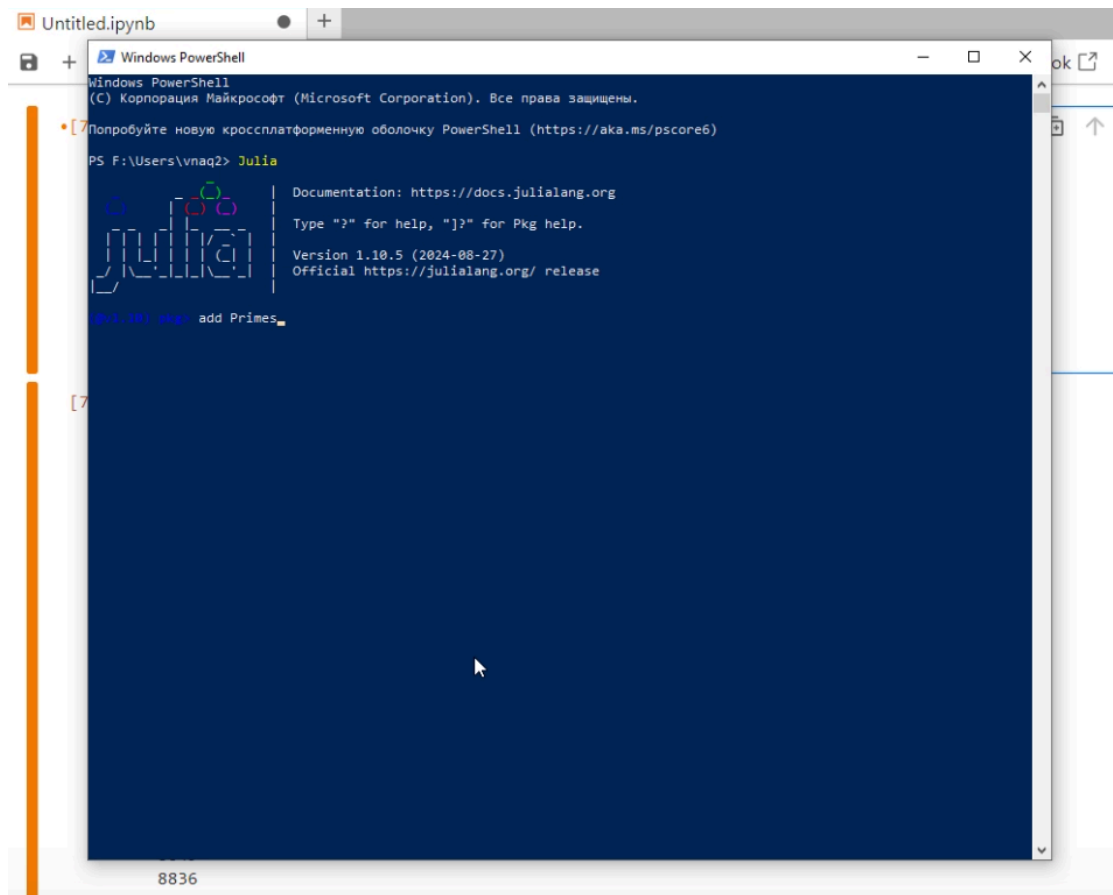
Пункт четвертый:

```
[78]: squares = [i^2 for i in 1:100]
```

```
[78]: 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
```

Подключение Primes:



```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

• [?] Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

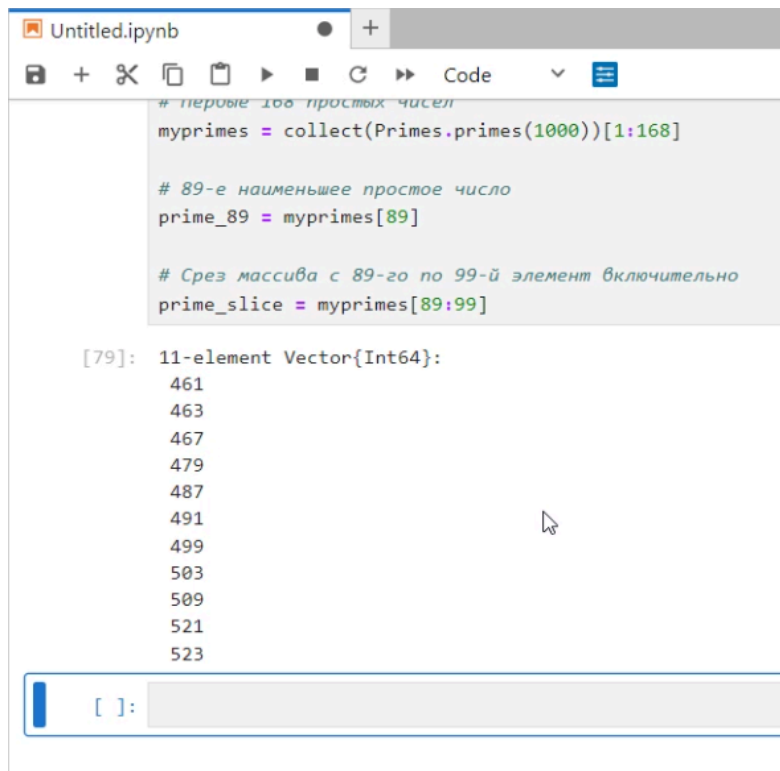
PS F:\Users\vnaq2> Julia

Documentation: https://docs.julialang.org
Type "?" for help, "]" for Pkg help.
Version 1.10.5 (2024-08-27)
Official https://julialang.org/ release

julia> add Primes_

[?] 
```

Пункт пятый:



```
Untitled.ipynb
# первые 1000 простых чисел
myprimes = collect(Primes.primes(1000))[1:1000]

# 89-е наименьшее простое число
prime_89 = myprimes[89]

# Срез массива с 89-го по 99-й элемент включительно
prime_slice = myprimes[89:99]

[79]: 11-element Vector{Int64}:
      461
      463
      467
      479
      487
      491
      499
      503
      509
      521
      523
```

Пункт шестой:

```
[80]: sum1 = sum(i^3 + 4 * i^2 for i in 10:100)
```

```
[80]: 26852735
```

```
[81]: sum2 = sum(2 * i / (i + 3 * i / i^2) for i in 1:25)
```

```
[81]: 45.79330094261042
```

```
[82]: # Функция для вычисления одного члена ряда с выводом промежуточных значений
function compute_term_debug(n)
    numerator = prod(2:2:2*n) # Числитель: произведение четных чисел
    denominator = prod(3:2:(2*n + 1)) # Знаменатель: произведение нечетных чисел
    term = numerator / denominator
    println("Член $n: $term")
    return term
end

# Сумма первых 10 членов ряда
sum3_debug = sum(compute_term_debug(n) for n in 1:10)
println("Сумма ряда: $sum3_debug")

Член 1: 0.6666666666666666
Член 2: 0.5333333333333333
Член 3: 0.45714285714285713
Член 4: 0.40634920634920635
Член 5: 0.3694083694083694
Член 6: 0.340992340992341
Член 7: 0.31825951825951826
Член 8: 0.29953837012660545
Член 9: 0.2837731927515209
Член 10: 0.27026018357287707
Сумма ряда: 3.945724038603296
```

Вывод: Я изучил несколько структур данных, реализованных в Julia, научился применять их и операции над ними для решения задач.