

# Лабораторная работа № 2

## Структуры данных

---

Герра Гарсия Паола Валентина

Российский университет дружбы народов, Москва, Россия

## Информация

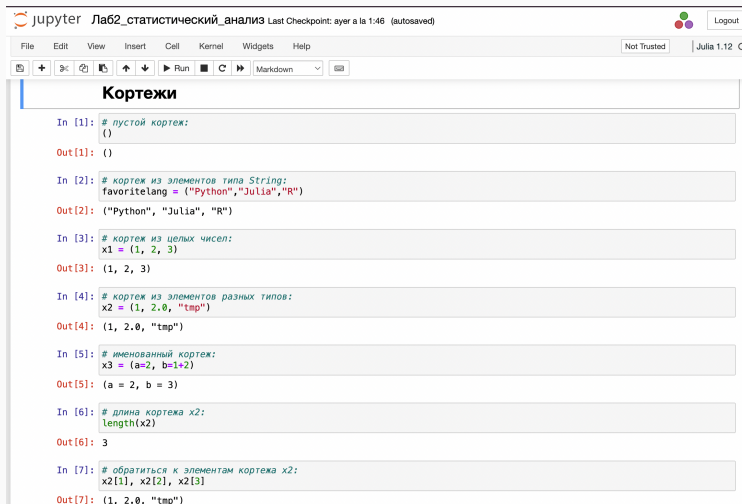
---

- Герра Гарсия Паола Валентина
- студентка
- Российский университет дружбы народов
- 1032225472@pfur.ru

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

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

# Выполнение лабораторной работы



The screenshot shows a Jupyter Notebook titled "Ла62\_статистический\_анализ" with a last checkpoint from "ayer a la 1:46 (autosaved)". The interface includes a top menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and markdown. The notebook content is titled "Кортежи" (Tuples) and contains seven code cells demonstrating tuple creation and manipulation in Julia.

```
In [1]: # пустой кортеж:
        {}

Out[1]: {}

In [2]: # кортеж из элементов типа String:
        favoritelang = ("Python", "Julia", "R")

Out[2]: ("Python", "Julia", "R")

In [3]: # кортеж из целых чисел:
        x1 = (1, 2, 3)

Out[3]: (1, 2, 3)

In [4]: # кортеж из элементов разных типов:
        x2 = (1, 2.0, "tmp")

Out[4]: (1, 2.0, "tmp")

In [5]: # именованный кортеж:
        x3 = (a=2, b=1+2)

Out[5]: (a = 2, b = 3)

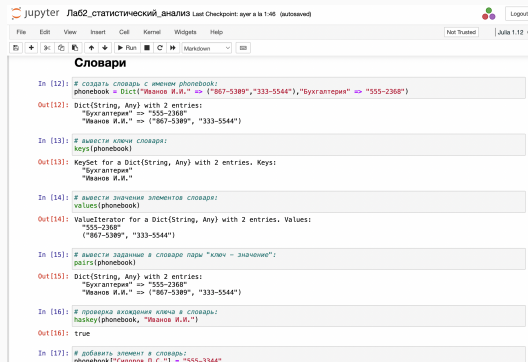
In [6]: # длина кортежа x2:
        length(x2)

Out[6]: 3

In [7]: # обратиться к элементам кортежа x2:
        x2[1], x2[2], x2[3]

Out[7]: (1, 2.0, "tmp")
```

Рис. 1: Примеры использования кортежей



The screenshot shows a Jupyter Notebook interface with the title 'Лаб2\_статистический\_анализ'. The notebook contains several code cells demonstrating dictionary operations in Python. The code is as follows:

```
In [12]: # создать словарь с именем phonebook:
phonebook = Dict{"Иванов И.И." => ("867-5389", "333-5544"), "Бухгалтерия" => "555-2368"}

Out[12]: Dict{String, Any} with 2 entries:
  "Бухгалтерия" => "555-2368"
  "Иванов И.И." => ("867-5389", "333-5544")

In [13]: # вывести ключи словаря:
keys(phonebook)

Out[13]: KeySet for a Dict{String, Any} with 2 entries. Keys:
  "Бухгалтерия"
  "Иванов И.И."

In [14]: # вывести значение элементов словаря:
values(phonebook)

Out[14]: ValueIterator for a Dict{String, Any} with 2 entries. Values:
  "555-2368"
  ("867-5389", "333-5544")

In [15]: # вывести заданные в словаре пары "ключ - значение":
pairs(phonebook)

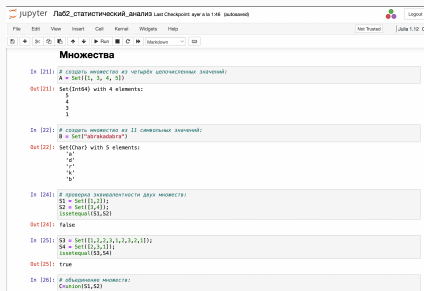
Out[15]: Dict{String, Any} with 2 entries:
  "Бухгалтерия" => "555-2368"
  "Иванов И.И." => ("867-5389", "333-5544")

In [16]: # проверка вхождения ключа в словарь:
haskey(phonebook, "Иванов И.И.")

Out[16]: true

In [17]: # добавить элемент в словарь:
phonebook["Сидоров П.С."] = "555-3344"
```

Рис. 2: Примеры использования словарей



```
jupyter Лабораторная работа 2: Статистический анализ Last Checkpoint: 2024-01-16 (Auto) Logout
```

File Edit View Insert Cell Kernel Widgets Help Not Trained Julia 1.12.0

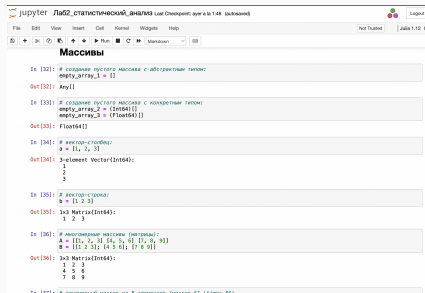
**Множества**

```
In [11]: # создать множество из четырех целочисленных значений:  
A = Set{1, 2, 4, 3}  
  
Out[11]: Set{Int64} with 4 elements:  
  5  
  4  
  3  
  1  
  
In [22]: # создать множество из 11 символьных значений:  
B = Set{"абракадабра"  
  
Out[22]: Set{Char} with 5 elements:  
  'а'  
  'б'  
  'к'  
  'д'  
  'а'
```

```
In [24]: # проверка эквивалентности двух множеств:  
S1 = Set{1, 2};  
S2 = Set{2, 1};  
isequal(S1, S2)  
  
Out[24]: false  
  
In [25]: S3 = Set{1, 2, 2, 3, 1, 2, 3, 2, 1};  
S4 = Set{2, 3, 1};  
isequal(S3, S4)  
  
Out[25]: true  
  
In [26]: # объединение множеств:  
C = union(S1, S2)
```

Рис. 3: Примеры использования множеств





The screenshot shows a Jupyter Notebook interface with the title 'Лабораторная работа 1'. The notebook contains several code cells demonstrating array operations in Julia. The first cell creates an empty array of abstract type, the second creates an empty array of concrete type, the third creates a 3-element vector, the fourth creates a 3x3 matrix, and the fifth creates a 3x3 matrix of integers. The output of each cell is displayed below the code.

```
In [32]: # создание пустого массива с абстрактным типом:
empty_array_1 = []

Out[32]: Any[]

In [33]: # создание пустого массива с конкретным типом:
empty_array_2 = {Int64}[]
empty_array_3 = {Float64}[]

Out[33]: Float64[]

In [34]: # вектор-столбец:
a = [1, 2, 3]

Out[34]: 3-element Vector{Int64}:
 1
 2
 3

In [35]: # вектор-строка:
b = [1 2 3]

Out[35]: 1x3 Matrix{Int64}:
 1 2 3

In [36]: # двумерные массивы (матрицы):
A = [11, 2, 3; 4, 5, 6; 7, 8, 9]
B = [1; 2; 3]; [4 5 6]; [7 8 9]

Out[36]: 3x3 Matrix{Int64}:
 1 2 3
 4 5 6
 7 8 9
```

Рис. 4: Примеры использования массивов

Даны множества:  $A = 0, 3, 4, 9$ ,  $B = 1, 3, 4, 7$ ,  $C = 0, 1, 2, 4, 7, 8, 9$ . Найдем  $P = A \cap B \cup A \cap B \cup A \cap C \cup B \cap C$

### Задания для самостоятельного выполнения

```
In [61]: A=Set([0, 3, 4, 9])
         B=Set([1, 3, 4, 7])
         C=Set([0, 1, 2, 4, 7, 8, 9])
         #P = A ∩ B ∪ A ∩ B ∪ A ∩ C ∪ B ∩ C
         P=union(intersect(A, B), intersect(A, B), intersect(A, C), intersect(B, C))
         println("P=$P")

P=Set([0, 4, 7, 9, 3, 1])
```

Рис. 5: Задание №1. Работа с множествами

Приведем свои примеры с выполнением операций над множествами элементов разных типов

```
In [62]: #Conjunto de strings
conjunto1=Set(["a", "b", "c"])

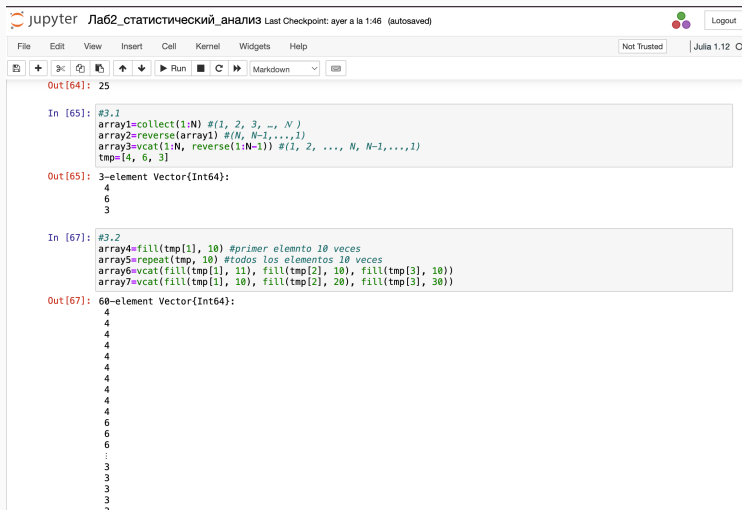
#Conjunto de floats
conjunto2=Set([1.5, 2.5, 3.5])

#Operaciones
union(conjunto1, Set(["b", "c", "d"]))
intersect(conjunto2, Set([2.5, 3.5, 4.5]))

Out[62]: Set{Float64} with 2 elements:
 3.5
 2.5
```

Рис. 6: Задание №2. Примеры операций над множествами элементов разных типов

# Выполнение лабораторной работы



The screenshot shows a Jupyter Notebook window titled "Лаб2\_статистический\_анализ". The interface includes a top bar with the Jupyter logo, the notebook title, and a "Last Checkpoint" timestamp. Below this is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar contains icons for file operations, a "Run" button, and a "Markdown" dropdown. The notebook content consists of two input cells and their corresponding outputs.

**Cell 65:**

```
#3.1
array1=collect(1:N) #(1, 2, 3, ..., N)
array2=reverse(array1) #(N, N-1,...,1)
array3=vcat(1:N, reverse(1:N-1)) #(1, 2, ..., N, N-1,...,1)
tmp=[4, 6, 3]
```

**Out[65]:** 3-element Vector{Int64}:

```
4
6
3
```

**Cell 67:**

```
#3.2
array4=fill(tmp[1], 10) #primer elemento 10 veces
array5=repeat(tmp, 10) #todos los elementos 10 veces
array6=vcat(fill(tmp[1], 11), fill(tmp[2], 10), fill(tmp[3], 10))
array7=vcat(fill(tmp[1], 10), fill(tmp[2], 20), fill(tmp[3], 30))
```

**Out[67]:** 60-element Vector{Int64}:

```
4
4
4
4
4
4
4
4
4
4
6
6
6
:
3
3
3
3
3
3
```

Рис. 7: Задание №3. Работа с массивами

```
In [68]: #3.3 arrays con potencias
array8= vcat(fill(2^tmp[1], 1), fill(2^tmp[2], 1), fill(2^tmp[3], 4))
count_6= count(x->x ==6, array8)
println("Number 6 appears $count_6 times")

Number 6 appears 0 times
```

Рис. 8: Задание №3. Работа с массивами

```
In [83]: #3.6  
M= 25  
vector3= [(2^i)/i for i in 1:M]
```

```
Out[83]: 25-element Vector{Float64}:  
 2.0  
 2.0  
 2.6666666666666665  
 4.0  
 6.4  
 10.666666666666666  
 18.285714285714285  
 32.0  
 56.888888888888886  
 102.4  
 186.1818181818182  
 341.3333333333333  
 630.1538461538462  
 1170.2857142857142  
 2184.5333333333333  
 4096.0  
 7710.117647058823  
 14563.555555555555  
 27594.105263157893  
 52428.8  
 99864.38095238095  
 190650.18181818182  
 364722.0869565217  
 699050.6666666666  
 1.34217728e6
```

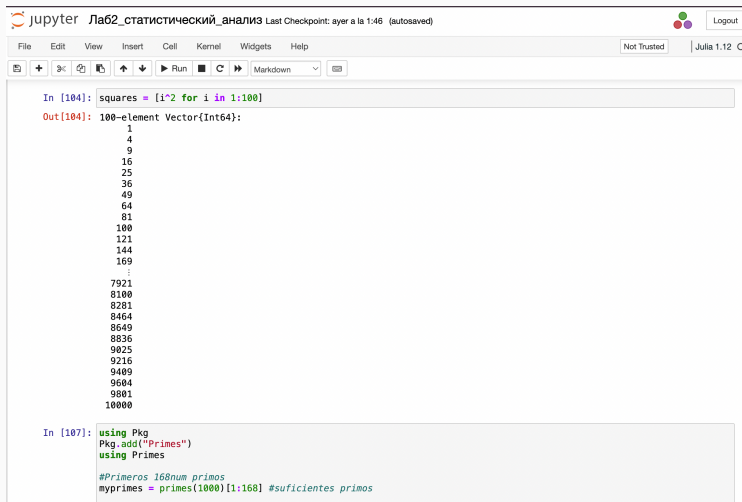
Рис. 9: Задание №3. Работа с векторами

```
In [84]: #3.7
vector4 = ["fn$i" for i in 1:30]

Out[84]: 30-element Vector{String}:
 "fn1"
 "fn2"
 "fn3"
 "fn4"
 "fn5"
 "fn6"
 "fn7"
 "fn8"
 "fn9"
 "fn10"
 "fn11"
 "fn12"
 "fn13"
 ⋮
 "fn19"
 "fn20"
 "fn21"
 "fn22"
 "fn23"
 "fn24"
 "fn25"
 "fn26"
 "fn27"
 "fn28"
 "fn29"
 "fn30"
```

Рис. 10: Задание №3. Работа с векторами

# Выполнение лабораторной работы



The screenshot shows a JupyterLab environment with the title bar "jupyter Лаб2\_статистический\_анализ" and a "Last Checkpoint: ayer a la 1:46 (autosaved)" message. The top menu bar includes "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". The top right corner has a "Logout" button and a "Julia 1.12" version indicator. The left sidebar contains icons for file operations. The main area displays two code cells. The first cell, labeled "In [104]:", contains the Julia code `squares = [i^2 for i in 1:100]`. The second cell, labeled "Out[104]:", shows the output of the first cell: "100-element Vector{Int64}:" followed by a list of squares from 1 to 10000. The third cell, labeled "In [107]:", contains the Julia code `using Pkg; Pkg.add("Primes"); using Primes; #Primeros 168num primos; myprimes = primes(1000)[1:168] #suficientes primos`.

```
jupyter Лаб2_статистический_анализ Last Checkpoint: ayer a la 1:46 (autosaved) Logout
```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.12

In [104]: `squares = [i^2 for i in 1:100]`

Out[104]: 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
```

In [107]: `using Pkg
Pkg.add("Primes")
using Primes

#Primeros 168num primos
myprimes = primes(1000)[1:168] #suficientes primos`

Рис. 11: Задание №3. Работа с векторами



```
In [108]: #89-esimo primo
primo_89= myprimes[89]
println("89-esimo primo: $primo_89")

89-esimo primo: 461

In [109]: #6.1
suma1 = sum([i^3+4i^2 for i in 10:100])
println("Suma 6.1: $suma1")

Suma 6.1: 26852735
```

Рис. 12: Задание №4

```
In [110]: #6.2
          M = 25
          suma2 = sum([(2^i)/i + (3^i)/(i^2) for i in 1:M])
          println("Suma 6.2: $suma2")

Suma 6.2: 2.1291704368143802e9
```

Рис. 13: Задание №5. Работа с пакетом Primes

```
suma 6.3 (primeros 5 terminos): 3.0634920634920633

In [111]: #6.3
function calcular_serie(terminos)
    resultado = 1.0
    numerador = 2
    denominador = 3

    for i in 1:terminos-1
        termino = numerador/denominador
        resultado += termino
        numerador *= (2*i + 2)
        denominador *= (2*i + 3)
    end
    return resultado
end

suma3 = calcular_serie(5)
println("Suma 6.3 (primeros 5 terminos): $suma3")

Suma 6.3 (primeros 5 terminos): 3.0634920634920633
```

Рис. 14: Задание №6

В результате выполнения данной лабораторной работы я изучила несколько структур данных, реализованных в Julia, научилась применять их и операции над ними для решения задач.

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).