

Лабораторная работа №1

Julia. Установка и настройка. Основные принципы

Чемоданова Ангелина Александровна

08 сентября 2025

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

- Чемоданова Ангелина Александровна
- Студентка НФИбд-02-22
- Российский университет дружбы народов имени Патриса Лумумбы
- 1132226443@pfur.ru
- <https://github.com/aachemodanova>



Цель работы

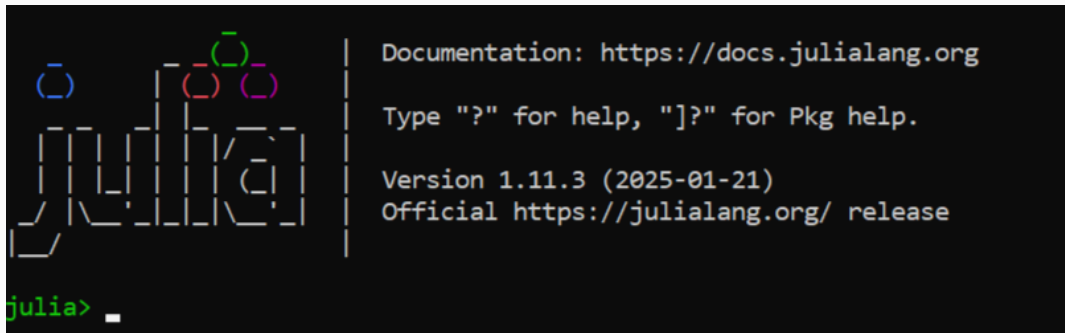
Основная цель работы — подготовить рабочее пространство и инструментарий для работы с языком программирования Julia, на простейших примерах познакомиться с основами синтаксиса Julia.

Задание

1. Установите под свою операционную систему Julia, Jupyter.
2. Используя Jupyter Lab, повторите примеры.
3. Выполните задания для самостоятельной работы.

Подготовка инструментария к работе

На моем компьютере ранее уже была установлена Julia для выполнения работ по “Математическому моделированию”:



```
julia> _
```

The screenshot shows the Julia REPL interface. On the left, the Julia logo is displayed using a grid of characters. To the right of the logo, the following text is shown:

```
Documentation: https://docs.julialang.org  
Type "?" for help, "]?" for Pkg help.  
Version 1.11.3 (2025-01-21)  
Official https://julialang.org/ release
```

Рис. 1: Julia

Для начала потренируемся с определением типов числовых величин:

```
typeof(3), typeof(3.5), typeof(3/3.5), typeof(sqrt(3+4im)), typeof(pi)
✓ 0.0s Julia
(Int64, Float64, Float64, ComplexF64, Irrational{::π})

1.0/0.0, 1.0/(-0.0), -0.0/0.0
✓ 0.0s Julia
(Inf, -Inf, NaN)

typeof(1.0/0.0), typeof(1.0/(-0.0)), typeof(0.0/0.0)
✓ 0.0s Julia
(Float64, Float64, Float64)

for T in [Int8, Int16, Int32, Int64, Int128, UInt8, UInt16, UInt32, UInt64, UInt128]
|   println("${lpad(T,7)}: [{typemin(T)}, {typemax(T)}]")
end
✓ 0.0s Julia
Int8: [-128,127]
Int16: [-32768,32767]
Int32: [-2147483648,2147483647]
Int64: [-9223372036854775808,9223372036854775807]
Int128: [-170141183460469231731687303715884105728,170141183460469231731687303715884105727]
UInt8: [0,255]
UInt16: [0,65535]
UInt32: [0,4294967295]
UInt64: [0,18446744073709551615]
UInt128: [0,340282366920938463463374607431768211455]
```

Рис. 2: Примеры определения типа числовых величин

После чего приступим к рассмотрению приведения аргументов к одному типу:

```
Int64(2.0), Char(2), typeof(Char(2))
✓ 0.0s
(2, '\x02', Char)
```

```
convert{Int64, Char}(Int64(2.0), Char(2))
✓ 0.0s
(2, '\x02')
```

```
Bool(1)
✓ 0.0s
true
```

```
typeof(promote{Int8, Float16, Float32}(Int8(1), Float16(4.5), Float32(4.1)))
✓ 0.0s
Tuple{Float32, Float32, Float32}
```

Рис. 3: Примеры приведения аргументов к одному типу

И рассмотрим примеры определения функций, а также работу с массивами:

```
function f(x)
    x^2
end
```

✓ 0.0s Julia

f (generic function with 1 method)

```
f(4)
```

✓ 0.0s Julia

16

```
g(x)=x^2
```

✓ 0.0s Julia

g (generic function with 1 method)

```
g(8)
```

✓ 0.0s Julia

64

Рис. 4: Примеры определения функций

Основы синтаксиса Julia на примерах

<pre>a = [4 7 6] # вектор-строка b = [1, 2, 3] # вектор-столбец a[2], b[2] # вторые элементы векторов a и b</pre> <p>✓ 0.0s</p>	Julia
(7, 2)	
<pre>a = 1; b = 2; c = 3; d = 4 # присвоение значений Am = [a b; c d] # матрица 2 x 2</pre> <p>✓ 0.0s</p>	Julia
<pre>2x2 Matrix{Int64}: 1 2 3 4</pre>	
<pre>Am[1,1], Am[1,2], Am[2,1], Am[2,2] # элементы матрицы</pre> <p>✓ 0.0s</p>	Julia
(1, 2, 3, 4)	
<pre>aa = [1 2] AA = [1 2; 3 4] aa*AA*aa'</pre> <p>✓ 0.0s</p>	Julia
<pre>1x1 Matrix{Int64}: 27</pre>	
<pre>aa, AA, aa'</pre> <p>✓ 0.0s</p>	Julia
([1 2], [1 2; 3 4], [1; 2;])	

Рис. 5: Примеры работы с массивами

В первом задании рассмотрим основные функции для чтения / записи / вывода информации на экран. Для этого составим свои примеры:

`write()` -- стандартная запись в файл.

```
write("myfile.txt", "Inrto_Julia!\nLab1.\n")
```

✓ 0.0s

Julia

19

`read()` -- чтение одного элемента.

```
io = open("myfile.txt", "r")  
read(io, String)
```

✓ 0.0s

Julia

"Inrto_Julia!\nLab1.\n"

`readline()` -- чтение строки.

```
readline("myfile.txt")
```

✓ 0.0s

Julia

"Inrto_Julia!"

Самостоятельная работа

`readlines()` -- чтение всех строк файла и запись их в массив.

```
readlines("myfile.txt")
```

✓ 0.0s

Julia

```
2-element Vector{String}:  
"Inrto_Julia!"  
"Labi."
```

`readdlm()` -- считывает данные как матрицу с обязательно одинаковым количеством элементов в каждой строке(дописывает "", если элемента нет).

```
using DelimitedFiles  
x = [1; 2; 3; 4];  
y = ["a"; "b"; "c"; "d"];  
  
open("delim_file.txt", "w") do io  
    | | | writedlm(io, [x y]) # записываем таблицу с двумя столбцами  
    | | | end;  
    readlm("delim_file.txt") # Читаем таблицу
```

✓ 2.8s

Julia

```
4x2 Matrix{Any}:  
1 "a"  
2 "b"  
3 "c"  
4 "d"
```

Рис. 7: Примеры работы с функциями для чтения/записи/вывода информации на экран

Самостоятельная работа

`print()` -- стандартный вывод без перевода строки.

```
print("Hello, World!")
```

✓ 0.0s

Julia

Hello, World!

`println()` -- стандартный вывод с переводом строки.

```
println("Hello", ' ', " world!")
```

✓ 0.0s

Julia

Hello, world!

`show()` -- показывает внутренне представление данных(например, строковой тип с кавычками).

```
show("Hello, World!")
```

✓ 0.0s

Julia

"Hello, World!"

Рис. 8: Примеры работы с функциями для чтения/записи/вывода информации на экран

Во втором задании составим пример для функции `parse()`:

Пример работы с функцией `parse`

Функция `parse` в языке программирования Julia работает для преобразования строкового представления данных в соответствующий числовой или другой базовый тип.

```
s = "123.45"  
println(typeof(s))  
num = parse(Float64, s)  
println(num)  
println(typeof(num))
```

✓ 0.0s

Julia

```
String  
123.45  
Float64
```

Рис. 9: Пример работы с функцией `parse`

Далее изучим синтаксис Julia для базовых математических операций с разным типом переменных:

Примеры работы базовых математических операций

$2 + 3.2$ ✓ 0.0s	Julia
5.2	
$10.5 \div -2$ ✓ 0.0s	Julia
-8.5	
$2 * 3.2$ ✓ 0.0s	Julia
6.4	
$6.2 / 2$ ✓ 0.0s	Julia
3.1	
$17.0 + 2$ ✓ 0.0s	Julia
19.0	
$17.0 \% 2$ ✓ 0.0s	Julia
1.0	

Рис. 10: Примеры работы базовых математических операций

<code>2^9</code>	Julia
✓ 0.0s	
512	
<code>sqrt(64.0)</code>	Julia
✓ 0.0s	
8.0	
<code>27^(1/3)</code>	Julia
✓ 0.0s	
3.0	
<code>5 == 5</code>	Julia
✓ 0.2s	
true	
<code>5 != 5</code>	Julia
✓ 0.0s	
false	
<code>7.0 < 7</code>	Julia
✓ 0.0s	
false	

Рис. 11: Примеры работы базовых математических операций

<pre>5 >= 3.6</pre> <p>✓ 0.0s</p>	Julia
<pre>true</pre>	
<pre>3 <= 4</pre> <p>✓ 0.0s</p>	Julia
<pre>true</pre>	
<pre>x = true !x</pre> <p>✓ 0.0s</p>	Julia
<pre>false</pre>	
<pre>y = false x y</pre> <p>✓ 0.0s</p>	Julia
<pre>true</pre>	
<pre>x && y</pre> <p>✓ 0.0s</p>	Julia
<pre>false</pre>	

Рис. 12: Примеры работы базовых математических операций

В конце работы приведём несколько примеров с операциями над матрицами:

Примеры работы с операциями над матрицами

```
using LinearAlgebra
A = [5 6; 1 3]
B = [1 0; 0 1]
```

✓ 0.0s

Julia

2×2 Matrix{Int64}:

```
1 0
0 1
```

C = A + B

✓ 0.0s

Julia

2×2 Matrix{Int64}:

```
6 6
1 4
```

A - B

✓ 0.0s

Julia

2×2 Matrix{Int64}:

```
4 6
1 2
```

A * B

✓ 0.0s

Julia

2×2 Matrix{Int64}:

```
5 6
1 3
```

Рис. 13: Примеры работы с операциями над матрицами


```
q = 5
C = q * B
✓ 0.2s Julia

2×2 Matrix{Int64}:
 5  0
 0  5

A'
✓ 0.4s Julia

2×2 adjoint(::Matrix{Int64}) with eltype Int64:
 5  1
 6  3

w = [1, 2, 3]
e = [2, 2, 2]
res = dot(w, e)
✓ 0.0s Julia
```

12

Рис. 14: Примеры работы с операциями над матрицами

В результате выполнения данной лабораторной работы мы подготовили рабочее пространство и инструментарий для работы с языком программирования Julia, на простейших примерах познакомились с основами синтаксиса Julia.