

# **Лабораторная работа №1**

**Julia. Установка и настройка. Основные принципы.**

Беличева Дарья Михайловна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>5</b>	<b>Выводы</b>	<b>18</b>
	<b>Список литературы</b>	<b>19</b>

# Список иллюстраций

4.1	Запуск Julia . . . . .	7
4.2	Выполнение примеров из лабораторной . . . . .	7
4.3	Выполнение примеров из лабораторной . . . . .	8
4.4	Выполнение примеров из лабораторной . . . . .	8
4.5	Чтение файла . . . . .	9
4.6	Вывод на печать . . . . .	10
4.7	Вывод на печать . . . . .	11
4.8	Команда записи . . . . .	12
4.9	Документация по функции <code>parse()</code> . . . . .	12
4.10	Примеры использования функции <code>parse()</code> . . . . .	13
4.11	Примеры базовых математических операций . . . . .	14
4.12	Примеры базовых математических операций . . . . .	15
4.13	примеры операций над матрицами . . . . .	16
4.14	примеры операций над векторами . . . . .	17

# 1 Цель работы

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

## 2 Задание

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

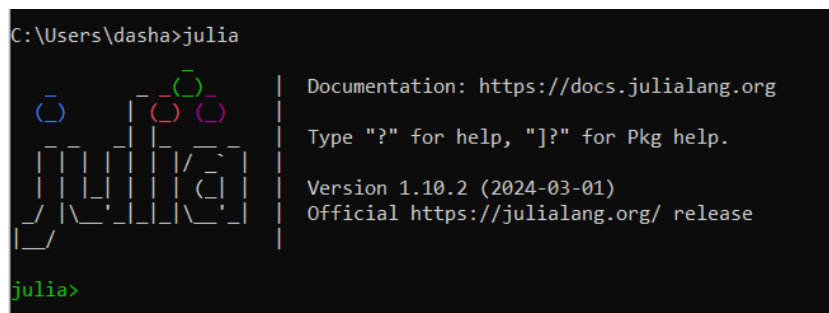
### 3 Теоретическое введение

Julia – высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений [1]. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков, однако имеет некоторые существенные отличия.

Для выполнения заданий была использована официальная документация Julia [2].

## 4 Выполнение лабораторной работы

У меня уже были установлены Julia и Jupyter (рис. 4.1).



```
C:\Users\dasha>julia

      _       _       _
     / \     / \     / \
    _/  \_   _/  \_   _/  \_
   /_    _/  /_    _/  /_    _/
  /_    _/  /_    _/  /_    _/
 /_    _/  /_    _/  /_    _/
/_    _/  /_    _/  /_    _/

Documentation: https://docs.julialang.org
Type "?" for help, "]"? for Pkg help.
Version 1.10.2 (2024-03-01)
Official https://julialang.org/ release

julia>
```

Рис. 4.1: Запуск Julia

Теперь повторим простейшие примеры для знакомства с синтаксисом Julia (рис. 4.1-4.4).



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

2+4
6

io = IOBuffer()
IOBuffer(data=UInt8{...}, readable=true, writable=true, seekable=true, append=false, size=0, maxsize=Inf, ptr=1, mark=-1)

println(io, "Hello,world!")

String(take!(io))
"Hello,world!\n"
```

Рис. 4.2: Выполнение примеров из лабораторной

```

function f(x)
x^2
end;
[23]

f(3)
[24]
... 9

g(x) = x^2;
[26]

g(4)
[27]
... 16

a = [3 4 5]
[28]
... 1x3 Matrix{Int64}:
 3  4  5

```

Рис. 4.3: Выполнение примеров из лабораторной

```

a[1], b[3]
[33]
... (3, 4)

a = 1; b = 3; c = 4; d = 5;
[3]

Am = [a b; c d]
[40]
... 2x2 Matrix{Int64}:
 1  3
 4  5

Am[1,1]
[41]
... 1

Am'
[42]
... 2x2 adjoint(::Matrix{Int64}) with eltype Int64:
 1  4
 3  5

```

Рис. 4.4: Выполнение примеров из лабораторной

Теперь перейдем к выполнению заданий.

## Задание №1



Изучим документацию по основным функциям Julia для чтения / записи / вывода информации на экран: `read()`, `readline()`, `readlines()`, `readdlm()`, `print()`, `println()`, `show()`, `write()`. Приведем свои примеры их использования, поясняя особенности их применения.

Для того, чтобы ознакомиться с документацией достаточно поставить знак ? перед интересующей функцией. Например, `?print`.

Создадим текстовый файл с любым содержанием в папке, где мы работаем. Откроем его на чтение и прочитаем с помощью команды `read()`. Текст вывелся в одну строку с разделителями `\r\n`. Также прочитаем текст используя функцию `readline()` - выведется только первая строка. Чтобы прочитать все строки в файле используем команду `readlines()` (рис. 4.5).



```

> n = open("hello.txt", "r")
[82]
... IOStream(<file hello.txt>)

data = read(n, String)
[83]
... "Hello, world\r\nHello, mir\r\nManera krutit"

m = open("hello.txt", "r")
lines = readline(m)
[92]
... "Hello, world"

m = open("hello.txt", "r")
lines = readlines(m)
[86]
... 3-element Vector{String}:
     "Hello, world"
     "Hello, mir"
     "Manera krutit"
```

Рис. 4.5: Чтение файла

Далее посмотрим, как работает команда `println()`, `print()` и `show()` (рис. 4.6-4.7).

```
using DelimitedFiles
x = [2; 3; 2; 1];

y = ["a"; "b"; "c"; "d"];

open("hello.txt", "w") do io
    for i in 1:length(x)
        println(io, string(x[i], " ", y[i]))
    end
end

readdlm("hello.txt")
```

4x2 Matrix{Any}:

2	"a"
3	"b"
2	"c"
1	"d"

Рис. 4.6: Вывод на печать

```
14] for i in 1:1:3
    | print(i, " ")
    end
· 1 2 3

15] for i in 1:1:3
    | println(i)
    end
· 1
  2
  3

13] show("Hello, world")
· "Hello, world"

14] print("Hello, world")
· Hello, world
```

Рис. 4.7: Вывод на печать

Посмотрим на работу функции `write()` (рис. 4.8).

```

[23]
io = IOBuffer()
write(io, "big boss", " is here")

... 16

[24]
String(take!(io))

... "big boss is here"

[30]
write(io, "big boss") + write(io, " is here")

... 16

[31]
String(take!(io))

... "big boss is here"

```

Рис. 4.8: Команда записи

## Задание №2

Изучим документацию по функции `parse()` (рис. 4.9). Приведем свои примеры её использования, поясняя особенности её применения (рис. 4.10).

```

?parse
[30]
... search: parse tryparse partialsortperm partialsortperm! pairs skipchars

...
ParseError: LaTeX parse error: No such environment: verbatim at position 7: \begin{verbatim} parsetype, st...

Parse a string as a number. For  $\text{Integer}$  types, a base can be specified (the default is 10). For floating-point types, the string is parsed as a decimal floating-point number.  $\text{Complex}$  types accept strings of the form  $\text{R+im}$  as a  $\text{Complex{R,Im}}$  of the requested type:  $\text{Complex{Im}}$  or  $\text{Complex{R}}$  can also be used instead of  $\text{Complex{Im}}$ , and  $\text{Complex{R}}$  or  $\text{Complex{Im}}$  are also permitted. If a valid number, an error is raised.

ParseError: LaTeX parse error: No such environment: quote at position 7: \begin{quote} \textbf{compat...

\section{Examples}\begin{verbatim} julia> parse{Int, "1234"} 1234
julia> parse{Int, "1234", base = 5} 194
julia> parse{Int, "abc", base = 16} 2812
julia> parse{Float64, "1.2e-3"} 0.0012
julia> parse{Complex{Float64}, "3.2e-1 + 4.5im"} 0.32 + 4.5im \end{verbatim} \rule{textwidth}{1pt}
ParseError: LaTeX parse error: No such environment: verbatim at position 7: \begin{verbatim} parse{TypeP...

Parses a string platform triplet back into a  $\text{Platform}$  object.

```

Рис. 4.9: Документация по функции `parse()`

```
[30] parse{Int, "134567"}
... 134567

[32] parse{Float64, "3.14"}
... 3.14

[48] parse{Int, "12"; base = 5}
... 7

[61] parse{Int, "hello", base = 30}
... 14167554

Функция parse() разбирает строки как числа. Аргумент base - это основание, показывающее систему счисления (по умолчанию десятичная)
```

Рис. 4.10: Примеры использования функции parse()

### Задание №3

Изучим синтаксис Julia для базовых математических операций с разным типом переменных: сложение, вычитание, умножение, деление, возведение в степень, извлечение корня, сравнение, логические операции. Приведем примеры с пояснениями по особенностям их применения (рис. 4.11-4.12).

```
a = 2 + 4
b = 2.5 + 2.5
c = "Tom" * "Jerry"
println(a, '\n', b, '\n', c)
[39]
... 6
    5.0
    TomJerry

▷
a = 2 - 4
b = 2.5 - 2.5
println(a, '\n', b)
[40]
... -2
    0.0

a = 5/2
b = 2.5/10
println(a, '\n', b)
[41]
... 2.5
    0.25

a = 3^2
b = 4^3
println(a, '\n', b)
[48]
... 9
    64
```

Рис. 4.11: Примеры базовых математических операций

```
[49] a = sqrt(64)

... 8.0

▷
[52] a = 4 > 5
    b = 5 == 5
    c = 6 < 10
    d = 1 != 4
    println(a, '\n', b, '\n', c, '\n', d)

... false
    true
    true
    true

[54] a = true && true
    b = true && false
    println(a, '\n', b)

... true
    false

[55] a = true || true
    b = true || false
    println(a, '\n', b)

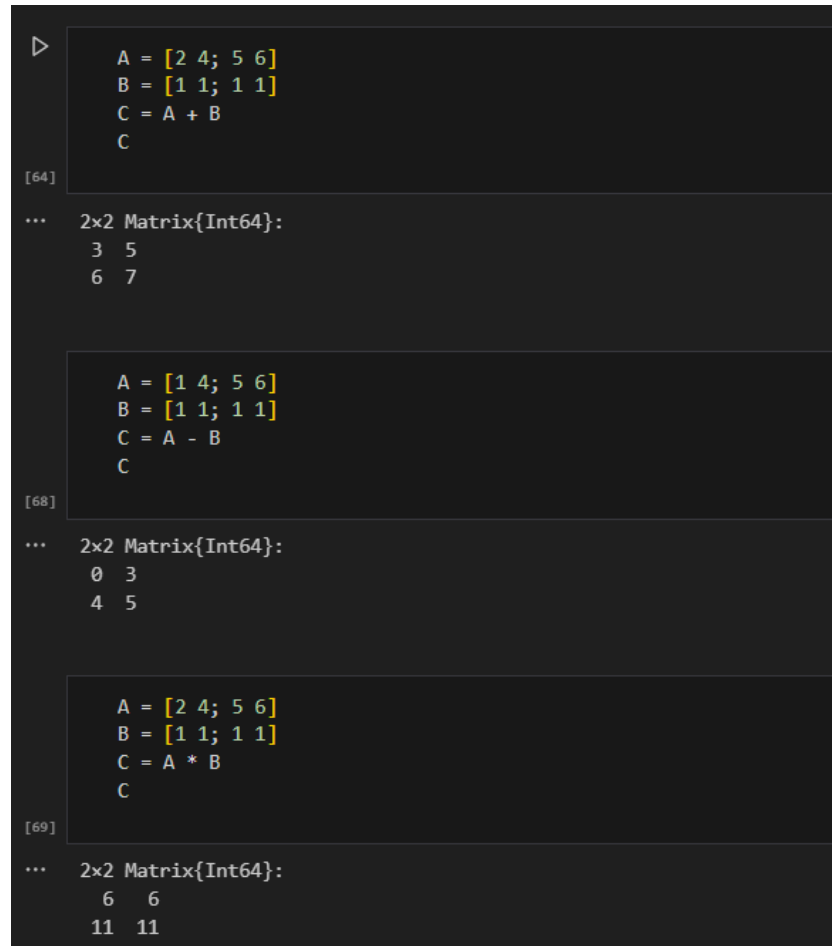
... true
    true

a = !true
b = !false
println(a, '\n', b)
```

Рис. 4.12: Примеры базовых математических операций

#### Задание №4

Приведем несколько примеров с пояснениями с операциями над матрицами и векторами: сложение, вычитание, скалярное произведение, транспонирование, умножение на скаляр (рис. 4.13-4.14).



```

A = [2 4; 5 6]
B = [1 1; 1 1]
C = A + B
C

[64]

... 2x2 Matrix{Int64}:
    3  5
    6  7

A = [1 4; 5 6]
B = [1 1; 1 1]
C = A - B
C

[68]

... 2x2 Matrix{Int64}:
    0  3
    4  5

A = [2 4; 5 6]
B = [1 1; 1 1]
C = A * B
C

[69]

... 2x2 Matrix{Int64}:
    6  6
   11 11
```

Рис. 4.13: примеры операций над матрицами



```
[72] A = [2 4; 5 6]
      k = 2
      C = k * A

... 2×2 Matrix{Int64}:
      4  8
     10 12

[74] A

... 2×2 Matrix{Int64}:
      2  4
      5  6

[75] A'

... 2×2 adjoint(::Matrix{Int64}) with eltype Int64:
      2  5
      4  6

[76] using LinearAlgebra
      u = [1, 2, 3]
      v = [1, 1, 2]
      prod = dot(u,v)

... 9

[77] u = [1, 2, 3]
      v = [1, 1, 2]
      prod = cross(u,v)

... 3-element Vector{Int64}:
      1
      1
     -1
```

Рис. 4.14: примеры операций над векторами

## 5 Выводы

В результате выполнения данной лабораторной работы я подготовила рабочее пространство и инструментарий для работы с языком программирования Julia, на простейших примерах познакомилась с основами синтаксиса Julia.

## Список литературы

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).