

# Лабораторная работа №1

Julia. Установка и настройка. Основные принципы.

---

Александр Эдуардович Аскеров

2025-09-12

## Докладчик

- Аскеров Александр Эдуардович
- Кафедра теории вероятностей и кибербезопасности
- Российский университет дружбы народов им. П. Лумумбы

## 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 из раздела 1.3.3.
3. Выполните задания для самостоятельной работы.

# Установите под свою операционную систему Julia, Jupyter.

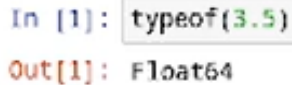
Julia и Jupyter установлены.

```
aeaskerov@vbox: $ julia --version
julia version 1.11.6
aeaskerov@vbox: $ jupyter --version
Selected Jupyter core packages...
IPython          : 8.5.0
ipykernel        : 6.17.0
ipywidgets       : 6.0.0
jupyter_client   : 7.4.9
jupyter_core     : 4.12.0
jupyter_server   : not installed
jupyterlab       : not installed
nbclient         : 0.7.2
nbconvert        : 6.5.3
nbformat         : 5.5.0
notebook         : 6.4.12
qtconsole        : not installed
traitlets        : 5.5.0
aeaskerov@vbox: $
```

Рисунок 1: Установленные Julia и Jupyter

## Используя Jupyter Lab, повторите примеры из раздела 1.3.3.

Определение типа числовой величины.



```
In [1]: typeof(3.5)  
Out[1]: Float64
```

**Рисунок 2:** Определение типа числовой величины

## Используя Jupyter Lab, повторите примеры из раздела 1.3.3.

Определение крайних значений диапазонов целочисленных числовых величин.

```
In [3]: for T in [Int8, Int16, Int32, Int64, Int128, UInt8, UInt16, UInt32, UInt64, UInt128]:
        print(f"{T}: [{T.min}, {T.max}]")
end

Int8: [-128, 127]
Int16: [-32768, 32767]
Int32: [-2147483648, 2147483647]
Int64: [-9223372036854775808, 9223372036854775807]
Int128: [-1701411834604682617713067208715864108520, 1701411834604682617713067208715864108520]
UInt8: [0, 255]
UInt16: [0, 65535]
UInt32: [0, 4294967295]
UInt64: [0, 18446744073709551615]
UInt128: [0, 340282366920938463463374607431768211455]
```

Рисунок 3: Определение крайних значений диапазонов целочисленных числовых величин

## Используя Jupyter Lab, повторите примеры из раздела 1.3.3.

### Преобразование типов.

```
In [4]: Int64{2.0}
Out[4]: 2

In [5]: Char{2}
Out[5]: '\x02': ASCII/Unicode U+0002 (category Cc: Other, control)

In [6]: convert{Int64, 2.0}
Out[6]: 2

In [8]: promote{Int8{1}, Float16{4.5}, Float32{4.1}}
Out[8]: {1.0f0, 4.5f0, 4.1f0}

In [9]: typeof(promote{Int8{1}, Float16{4.5}, Float32{4.1}})
Out[9]: Tuple{Float32, Float32, Float32}
```

Рисунок 4: Преобразование типов разными способами

## Используя Jupyter Lab, повторите примеры из раздела 1.3.3.

Определение функций.

```
In [10]: function f(x)
          x^2
        end
        f(4)

Out[10]: 16

In [11]: g(x)=x^2
        g(4)

Out[11]: 16
```

Рисунок 5: Определение функций разными способами



## Используя Jupyter Lab, повторите примеры из раздела 1.3.3.

Определение одномерных массивов (вектор-строка и вектор-столбец) и обращение к их вторым элементам.

```
In [12]: a = [4 7 6]  
         b = [1, 2, 3]  
         a[2], b[2]
```

```
Out[12]: (7, 2)
```

**Рисунок 6:** Работа с одномерными массивами

## Используя Jupyter Lab, повторите примеры из раздела 1.3.3.

Определение двумерного массива (матрицы) и операции над массивами, включая транспонирование.

```
In [13]: a = 1; b = 2; c = 3; d = 4  
        Am = [a b; c d]  
  
Out[13]: 2x2 Matrix{Int64}:  
         1  2  
         3  4  
  
In [14]: aa = [1 2]  
        AA = [1 2; 3 4]  
        aa*AA*aa'  
  
Out[14]: 1x1 Matrix{Int64}:  
         27
```

Рисунок 7: Работа с двумерными массивами

# Задания для самостоятельной работы

## Задание 1

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

Фрагмент документации по функции `read`.

```
In [15]: read{}
```

```
Out[15]: read(io::IO, T)
```

Read a single value of type `T` from `io`, in canonical binary representation.

Note that Julia does not convert the endianness for you. Use `ntoh` or `htoh` for this purpose.

```
read(io::IO, String)
```

Read the entirety of `io`, as a `String` (see also `readchomp`).

**Examples**

```
julia> io = IOBuffer("JuliaLang is a GitHub organization");
julia> read(io, Char)
'J': ASCII/Unicode U+004A (category Lu: Letter, uppercase)
julia> io = IOBuffer("JuliaLang is a GitHub organization");
julia> read(io, String)
"JuliaLang is a GitHub organization"
```

Рисунок 8: Фрагмент документации по функции `read`

## Задания для самостоятельной работы

Применение функций `read`, `readline`, `readlm`.

```
In [16]: read("file.txt")
Out[16]: 17-element Vector{UInt8}:
 0x61
 0x62
 0x63
 0x2c
 0x20
 0x64
 0x65
 0x66
 0x0a
 0x67
 0x68
 0x69
 0x2c
 0x20
 0x6a
 0x6b
 0x6c

In [17]: readline("file.txt")
Out[17]: "abc, def"

In [18]: using DelimitedFiles
          readlm("file.txt", ',')
Out[18]: 2x2 Matrix{Any}:
 "abc"  " def"
 "ghi"  " jkl"
```

Рисунок 9: Применение функций `read`, `readline`, `readlm`

## Задания для самостоятельной работы

Применение функций print, println, show, write.

```
In [19]: print("hello")
         print("hello")
         hellohello

In [20]: println("hello")
         println("hello")
         hello
         hello

In [21]: show("hello")
         "hello"

In [22]: write("file.txt", "hello")
Out[22]: 5
```

**Рисунок 10:** Применение функций print, println, show, write

# Задания для самостоятельной работы

## Задание 2

*Изучите документацию по функции `parse()`. Приведите свои примеры её использования, поясняя особенности её применения.*

Документация по функции `parse`.

```
In [23]: ?parse()
Out[23]: parse(type, str; base)

Parse a string as a number. For 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. Complex types
are parsed from decimal strings of the form "R±iIm" as a Complex{R,I} of the requested
type; "i" or "-j" can also be used instead of "im", and "R" or "Iim" are also
permitted. If the string does not contain a valid number, an error is raised.

!! compat "Julia 1.1" parse{Bool, STR} requires at least Julia 1.1.

Examples
julia> parse{Int, "1234"}
1234

julia> parse{Int, "1234", base = 5}
194
```

Рисунок 11: Документация по функции `parse`

# Задания для самостоятельной работы

Примеры использования.

```
In [24]: parse(Int, "4153", base = 6)
Out[24]: 933

In [25]: parse(Bool, "1")
Out[25]: true
```

Рисунок 12: Примеры использования

### Задание 3

*Изучите синтаксис Julia для базовых математических операций с разным типом переменных: сложение, вычитание, умножение, деление, возведение в степень, извлечение корня, сравнение, логические операции. Приведите свои примеры с пояснениями по особенностям их применения.*



# Задания для самостоятельной работы

## Арифметические операции.

```
In [26]: 1 + 1
```

```
Out[26]: 2
```

```
In [29]: "hello" * " goodbye"
```

```
Out[29]: "hello goodbye"
```

```
In [31]: 5 - {-9}
```

```
Out[31]: 14
```

```
In [32]: 45 * 666.0
```

```
Out[32]: 29970.0
```

```
In [33]: 6 / 2
```

```
Out[33]: 3.0
```

```
In [34]: 5^7
```

```
Out[34]: 78125
```

```
In [35]: sqrt(81)
```

```
Out[35]: 9.0
```

```
In [36]: 81^0.5
```

```
Out[36]: 9.0
```

```
In [39]: 57 > 9
```

```
Out[39]: true
```

```
In [38]: 99 == 99
```

```
Out[38]: true
```

```
In [40]: {75 == 75} && {0 != 1} || !(true == false)
```

```
Out[40]: true
```

## Задание 4

*Приведите несколько своих примеров с пояснениями с операциями над матрицами и векторами: сложение, вычитание, скалярное произведение, транспонирование, умножение на скаляр.*

# Задания для самостоятельной работы

## Операции над матрицами.

```
In [42]: v = [1, 2, 3]  
m = [1 2; 3 4]
```

```
Out[42]: 2×2 Matrix{Int64}:  
 1  2  
 3  4
```

```
In [43]: v + v
```

```
Out[43]: 3-element Vector{Int64}:  
 2  
 4  
 6
```

```
In [44]: m + m
```

```
Out[44]: 2×2 Matrix{Int64}:  
 2  4  
 6  8
```

```
In [45]: using LinearAlgebra  
dot(v, v)
```

```
Out[45]: 14
```

```
In [46]: m'
```

```
Out[46]: 2×2 adjoint{::Matrix{Int64}} with eltype Int64:  
 1  3  
 2  4
```

```
In [47]: 2 * v
```

```
Out[47]: 3-element Vector{Int64}:  
 2  
 4  
 6
```

Было подготовлено рабочее пространство и инструментарий для работы с языком программирования Julia, а также на простейших примерах было произведено ознакомление с основами синтаксиса Julia.