

Лабораторная Работа №1

Julia. Установка и настройка. Основные принципы.

Козлов В.П.

Российский университет дружбы народов им. Патриса Лумумбы, Москва, Россия

- Козлов Всеволод Павлович
- НФИбд-02-22
- Российский университет дружбы народов
- [1132226428@pfur.ru]

Выполнение лабораторной работы

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

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

Установил Chocolatey

```
Creating Chocolatey CLI folders if they do not already exist.

chocolatey.nupkg file not installed in lib.
Attempting to locate it from bootstrapper.
PATH environment variable does not have C:\ProgramData\chocolatey\bin in it. Adding...
WARNING: Not setting tab completion: Profile file does not exist at
'C:\Users\vsvld\OneDrive\Документы\WindowsPowerShell\Microsoft.PowerShell_profile.ps1'.
Chocolatey CLI (choco.exe) is now ready.
You can call choco from anywhere, command line or PowerShell by typing choco.
Run choco /? for a list of functions.
You may need to shut down and restart PowerShell and/or consoles
first prior to using choco.
Ensuring Chocolatey commands are on the path
Ensuring chocolatey.nupkg is in the lib folder
PS C:\WINDOWS\system32>
```

Figure 1: Установка Chocolatey

Установил Far

```
Installing Far...
Far has been installed.
  far may be able to be automatically uninstalled.
  The install of far was successful.
  Deployed to 'C:\Program Files\Far Manager\'

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).

Enjoy using Chocolatey? Explore more amazing features to take your
experience to the next level at
https://chocolatey.org/compare
PS C:\WINDOWS\system32>
```

Figure 2: Установка Far

Установил Notepad++

```
Deployed to 'C:\ProgramData\chocolatey\extensions\chocolatey-core'
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading notepadplusplus.install 8.8.5... 100%

notepadplusplus.install v8.8.5 [Approved]
notepadplusplus.install package files install completed. Performing other installation steps.
Installing 64-bit notepadplusplus.install...
notepadplusplus.install has been installed.
WARNING: No registry key found based on 'Notepad\+\+'
notepadplusplus.install installed to 'C:\Program Files\Notepad++'
Added C:\ProgramData\chocolatey\bin\notepad++.exe shim pointed to 'c:\program files\notepad++\notepad++.exe'.
notepadplusplus.install can be automatically uninstalled.
The install of notepadplusplus.install was successful.
Software installed as 'exe', install location is likely default.
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading notepadplusplus 8.8.5... 100%

notepadplusplus v8.8.5 [Approved]
notepadplusplus package files install completed. Performing other installation steps.
The install of notepadplusplus was successful.
Software install location not explicitly set, it could be in package or
default install location of installer.

Chocolatey installed 4/4 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
```

Figure 3: Установка Notepad++

Установил Julia

```
julia v1.11.6 [Approved]
julia package files install completed. Performing other installation steps.
Installing 64-bit Julia...
Julia has been installed.
Julia installed to 'C:\Users\vsvld\AppData\Local\Programs\Julia-1.11.6\bin\julia.exe'
Added C:\ProgramData\chocolatey\bin\julia.exe shim pointed to 'c:\users\vsvld\appdata\local\julia.exe'.
  julia can be automatically uninstalled.
  The install of julia was successful.
  Deployed to 'C:\Users\vsvld\AppData\Local\Programs\Julia-1.11.6\'

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\WINDOWS\system32>
```

Figure 4: Установка Julia

Установил доп пакет IJulia

```
[83775a58] + Zlib_jll v1.2.13+1
[8e850ede] + nghttp2_jll v1.59.0+0
[3f19e933] + p7zip_jll v17.4.0+2
Info Packages marked with ⚠ have new versions available but compatibility constraints restrict them from upgrading. To see why use `status --outdated -m`
Building Conda → 'C:\Users\vsvld\.julia\scratchspaces\44cfe95a-1eb2-52ea-b672-e2afdf69b78f\b19db3927f0db4151cb86d073689f2428e524576\build.log'
Building IJulia → 'C:\Users\vsvld\.julia\scratchspaces\44cfe95a-1eb2-52ea-b672-e2afdf69b78f\00a9cbc52b819846e89ef5b221227c33b110d8e0\build.log'
Precompiling project...
11 dependencies successfully precompiled in 77 seconds. 28 already precompiled.
(@v1.11) pkg>
```

Figure 5: Установка IJulia

Установил Anaconda3

```
Using system proxy server '127.0.0.1:3067'.
Downloading anaconda3 64 bit
  from 'https://repo.anaconda.com/archive/Anaconda3-2025.06-0-Windows-x86_64.exe'
Using system proxy server '127.0.0.1:3067'.
Progress: 100% - Completed download of C:\Users\vsvld\AppData\Local\Temp\anaconda3\2025.6.0\A
x86_64.exe (914.33 MB).
Download of Anaconda3-2025.06-0-Windows-x86_64.exe (914.33 MB) completed.
Hashes match.
Installing anaconda3...
anaconda3 has been installed.
  anaconda3 can be automatically uninstalled.
Environment Vars (like PATH) have changed. Close/reopen your shell to
see the changes (or in powershell/cmd.exe just type `refreshenv`).
The install of anaconda3 was successful.
  Deployed to 'C:\tools\anaconda3'

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32>
```

Figure 6: Установка Anaconda3

Документация по println и ее использование

```
[1]: ?println
```

```
search: println print sprint pointer printstyled
```

```
[1]:
```

```
println(io::IO, xs...)
```

Print (using `print`) `xs` to `io` followed by a newline. If `io` is not supplied, prints to the default output stream `stdout`.

See also `printstyled` to add colors etc.

Examples

```
julia> println("Hello, world")
```

```
Hello, world
```

```
julia> io = IOBuffer();
```

```
julia> println(io, "Hello", ',', " world.")
```

```
julia> String(take!(io))
```

```
"Hello, world.\n"
```

```
[2]: println("Я буду учить Julia")
```

```
print("Молодец")
```

```
Я буду учить Julia
```

```
Молодец
```

Figure 7: Документация по println и ее использование

Документация по readline и ее использование

```
[3]: ?readline
search: readline readlines readlink readdir eachline readbytes! read @inline
```

```
[3]:      readline(io::IO=stdin; keep::Bool=false)
      readline(filename::AbstractString; keep::Bool=false)
```

Read a single line of text from the given I/O stream or file (defaults to `stdin`). When reading from a file, the text is assumed to be encoded in UTF-8. Lines in the input end with `"\n"` or `"\r"` is returned. When `keep` is true, they are returned as part of the line.

Return a `String`. See also `copyline` to instead write in-place to another stream (which can be a preallocated `IOBuffer`).

See also `readuntil` for reading until more general delimiters.

Examples

```
julia> write("my_file.txt", "JuliaLang is a GitHub organization.\nIt has many members.\n");
```

```
julia> readline("my_file.txt")
"JuliaLang is a GitHub organization."
```

```
julia> readline("my_file.txt", keep=true)
"JuliaLang is a GitHub organization.\n"
```

```
julia> rm("my_file.txt")
```

```
julia> print("Enter your name: ")
Enter your name:
```

```
julia> your_name = readline()
Logan
"Logan"
```

```
[4]: println("Как вас зовут?")
name = readline()
println("Привет, $name")
```

```
Как вас зовут?
stdin> Сена
Привет, Сена
```

Figure 8: Документация по readline и ее использование

Документация по readlines и ее использование



Figure 9: Документация по readlines и ее использование

Документация по readlm и ее использование

```
julia> using DelimitedFiles

julia> x = [1; 2; 3; 4];

julia> y = [5; 6; 7; 8];

julia> open("delim_file.txt", "w") do io
    writedlm(io, [x y])
end

julia> readlm("delim_file.txt", '\t', Int, '\n')
4x2 Matrix{Int64}:
 1  5
 2  6
 3  7
 4  8

julia> rm("delim_file.txt")
```

```
[13]: using DelimitedFiles

open("numbers.txt", "w") do f
    write(f, "1 2 3\n4 5 6\n7 8 9\n")
end

A = readlm("numbers.txt")
println(A)

[1.0 2.0 3.0; 4.0 5.0 6.0; 7.0 8.0 9.0]
```

Figure 10: Документация по readlm и ее использование

Документация по show и ее использование

```
show(io::IO, mime, x)
```

The `display` functions ultimately call `show` in order to write an object `x` as a given `mime` type to a given I/O stream `io` (usually a memory buffer), if possible. In order to provide a rich multimedia representation of a user-c type `T`, it is only necessary to define a new `show` method for `T`, via: `show(io, ::MIME"mime", x::T) = ...`, where `mime` is a MIME-type string and the function body calls `write` (or similar) to write that representation to `io`. (Note that the `MIME""` notation only supports literal strings; to construct `MIME` types in a more flexible manner use `MIME{Symbol{""}}`.)

For example, if you define a `MyImage` type and know how to write it to a PNG file, you could define a function `show(io, ::MIME"image/png", x::MyImage) = ...` to allow your images to be displayed on any PNG-capable `AbstractDisplay` (such as `Julia`). As usual, be sure to `import Base.show` in order to add new methods to the built-in Julia function `show`.

Technically, the `MIME"mime"` macro defines a singleton type for the given `mime` string, which allows us to exploit Julia's dispatch mechanisms in determining how to display objects of any given type.

The default MIME type is `MIME"text/plain"`. There is a fallback definition for `text/plain` output that calls `show` with 2 arguments, so it is not always necessary to add a method for that case. If a type benefits from custom human-readable output though, `show(io::IO, ::MIME"text/plain", ::T)` should be defined. For example, the `Day` type uses `1 day` as the output for the `text/plain` MIME type, and `Day(1)` as the output of 2-argument `show`.

Examples

```
julia> struct Day
    n::Int
end

julia> Base.show(io::IO, ::MIME"text/plain", d::Day) = print(io, d.n, " day")

julia> Day(1)
1 day
```

Container types generally implement 3-argument `show` by calling `show(io, MIME"text/plain"(), x)` for elements `x`, with `:compact => true` set in an `IOContext` passed as the first argument.

```
103: x = 123.456
      show(x)
      println()
      show(stdout, "text/plain", x)

123.456
123.456
```

Would you like to get notified at
Jupyter news?
[Open email](#)

Figure 11: Документация по show и ее использование

Документация по write и ее использование

44

```
julia> String(take!(io))  
"Sometimes those members write documentation."
```

User-defined plain-data types without `write` methods can be written when wrapped in a `Ref`:

```
julia> struct MyStruct; x::Float64; end  
  
julia> io = IOBuffer()  
IOBuffer{data=UInt8[], readable=true, writable=true, seekable=true, append=false, si  
  
julia> write(io, Ref{MyStruct}(42.0))  
8  
  
julia> seekstart(io); read!(io, Ref{MyStruct}(NaN))  
Base.RefValue{MyStruct}{MyStruct(42.0)}
```

```
write(filename::AbstractString, content)
```

Write the canonical binary representation of `content` to a file, which will be created if it does not exist yet or ov

Return the number of bytes written into the file.

```
[21]: open("nums.txt", "w") do f  
      write(f, "10\n20\n30\n")  
end  
  
lines = readlines("nums.txt")  
nums = parse.(Int, lines)  
println(nums)  
  
[10, 20, 30]
```

Figure 12: Документация по write и ее использование

Документация по parse и ее использование

0.0000

```
julia> parse{Complex{Float64}, "3.2e-1 + 4.5im"}  
0.32 + 4.5im
```

```
parse(::Type{Platform}, triplet::AbstractString)
```

Parses a string platform triplet back into a `Platform` object.

```
parse(::Type{SimpleColor}, rgb::String)
```

An analogue of `tryparse{SimpleColor, rgb::String}` (which see), that raises an error instead of returning `nothing`.

[23]: *# строка в целое число*

```
x = parse{Int, "42"}  
println(x)  
println(typeof(x))
```

```
42  
Int64
```

[25]: *# строка в float точку*

```
y = parse{Float64, "3.14"}  
println(y)  
println(typeof(y))
```

```
3.14  
Float64
```

Figure 13: Документация по parse и ее использование

Арифметика, степени и корни в Julia

Арифметика

```
[28]: a = 10  
      b = 3  
  
      println(a+b)  
      println(a-b)  
      println(a*b)  
      println(a/b)  
      println(div(a, b))  
      println(mod(a, b))  
  
      13  
      7  
      30  
      3.3333333333333335  
      3  
      1
```

Степени и корни

```
[29]: println(a*2)  
      println(sqrt(16))  
      println(cbrt(27))  
  
      20  
      4.0  
      3.0
```

Figure 14: Арифметика, степени и корни в Julia

▼ Сравнения

```
[31]: println(a>b)
      println(a<b)
      println(a==b)
      println(a!=b)

      true
      false
      false
      true
```

Логика

```
[32]: x=true
      y=false

      println(x && y)
      println(x || y)
      println(!x)

      false
      true
      false
```

Figure 15: Сравнения и логика в Julia

Операции с разными типами данных в Julia

Разные типы

```
[34]: i = 5  
      f = 2.5  
      println(i + f)  
  
      s1 = "Hello"  
      s2 = "julia"  
      println(s1*s2)  
  
      c = 'A'  
      println(c+1)  
  
      println(true+3)
```

7.5
Hellojulia
B
4

Figure 16: Операции с разными типами данных в Julia

Операции с матрицами в Julia

```
[36]: using LinearAlgebra

A = [1 2; 3 4]
B = [5 6; 7 8]

println("A = ")
println(A)

println("A + B = ")
println(A+B)

println("A - B = ")
println(A-B)

# скал произв
v1 = [1, 2, 3]
v2 = [4, 5, 6]
println("dot(v1, v2) = ", dot(v1, v2))
```



```
A =
[1 2; 3 4]
A + B =
[6 8; 10 12]
A - B =
[-4 -4; -4 -4]
dot(v1, v2) = 32
```

Figure 17: Операции с матрицами в Julia

Операции с матрицами в Julia

```
[37]: # транспонирование
println("A' (трансп матрица) = ")
println(A')

# умножение матрицы на скаляр
println("2 * A = ")
println(2*A)

# умножение матрицы на матрицу
println("A * B = ")
println(A*B)

A' (трансп матрица) =
[1 3; 2 4]
2 * A =
[2 4; 6 8]
A * B =
[19 22; 43 50]
```

Figure 18: Операции с матрицами в Julia

Подготовил рабочее пространство и инструментарий для работы с языком программирования Julia, на простейших примерах познакомился с основами синтаксиса Julia.