

Российский университет дружбы народов
Факультет физико-математических и естественных наук

Отчёт по лабораторной работе №1

Москва 2023

1032203967
Быстров Глеб

Цель работы (задание)

- Подготовить рабочее пространство и инструментарий для работы с языком программирования Julia, на простейших примерах познакомиться с основами синтаксиса Julia.

Задачи (метод выполнения)

- Установка пакетов для работы с Jupyter

```
(jupyter) pkg> add IJulia
  Updating registry at `C:\Users\GlebB\.julia\registries\General.toml`
  Resolving package versions...
  Installed SoftGlobalScope v1.1.0
  Installed libsodium_jll v1.0.20+0
  Installed ZeroMQ_jll v4.3.4+0
  Installed ZMQ v1.2.2
  Installed IJulia v1.24.2
  Downloaded artifact: libsodium
  Downloaded artifact: ZeroMQ
  Updating `C:\Users\GlebB\.julia\environments\v1.9\Project.toml`
 [7073ff75] + IJulia v1.24.2
  Updating `C:\Users\GlebB\.julia\environments\v1.9\Manifest.toml`
 [8f4d0f93] + Conda v1.9.1
 [7073ff75] + IJulia v1.24.2
 [b85f4697] + SoftGlobalScope v1.1.0
 [81def892] + VersionParsing v1.3.0
 [c2297ded] + ZMQ v1.2.2
 [8f1865be] + ZeroMQ_jll v4.3.4+0
 [a9144af2] + libsodium_jll v1.0.20+0
  Building IJulia → `C:\Users\GlebB\.julia\scratchspaces\44cfe95a-1eb2-52ea-b672-e2afdf69b78f\47ac8cc196b81001a711f4b2c12c97372338f00c\build.log`
  Precompiling project...
  8 dependencies successfully precompiled in 26 seconds. 216 already precompiled.

(jupyter) pkg> -
```

Задачи (метод выполнения)

- Повторил примеры из разделов 1.3.2 и 1.3.3

```
Pluto.jl / данных\Lab1\Lab1_BystrovGleb.jl

5
- 2+3

3
- begin
-   3+4
-   1+2
- end

- 4+5;

Int64
- typeof(3)

+
- for T in [Int8, Int16, Int32, Int64, Int128, UInt8, UInt16, UInt32, UInt64, UInt128]
-   println("$(lpad(T,7)): [$(typemin(T)), $(typemax(T))]" )
- end

Int8: [-128,127]
Int16: [-32768,32767]
Int32: [-2147483648,2147483647]
Int64: [-9223372036854775808,9223372036854775807]
Int128: [-170141183460469231731687303715884105728,170141183460469231731687303715884105727]
UInt8: [0,255]
UInt16: [0,65535]
UInt32: [0,4294967295]
```

```
> (2, '\x02')
- convert{Int64, 2.0}, convert{Char, 2}

> (true, false)
- Bool{1}, Bool{0}

> (1.0, 4.5, 4.1)
- promote{Int8{1}, Float16{4.5}, Float32{4.1}}

Tuple{Float32, Float32, Float32}
- typeof(promote{Int8{1}, Float16{4.5}, Float32{4.1}})

f (generic function with 1 method)
- function f(x)
-   x^2
- end

16
- f(4)

g (generic function with 1 method)
- g(x)=x^2
```

1.6 ms

Задачи (метод выполнения)

- Задание для самостоятельной работы №1

```
- write("my_file.txt", "Hello\n");  
  
"Hello"  
- readline("my_file.txt")  
  
"Hello\n"  
- readline("my_file.txt", keep=true)  
  
- 7readlines  
163 μs  
  
- write("my_file.txt", "JuliaLang is a GitHub organization.\nIt has many members.\n");  
  
▶ ["JuliaLang is a GitHub organization.", "It has many members."]  
- readlines("my_file.txt")  
  
▶ ["JuliaLang is a GitHub organization.\n", "It has many members.\n"]  
- readlines("my_file.txt", keep=true)  
  
- 7readdlm  
215 μs  
  
+  
- a = [1; 2; 3];  
- b = [4; 5; 6];  
- open("file.txt")  
155 μs
```

Задачи (метод выполнения)

- Задание для самостоятельной работы №2

7parse

210 μs

13579

```
· parse(Int, "13579")
```

123456789

```
· parse(Int, "123456789", base = 10)
```

703710

```
· parse(Int, "abcde", base = 16)
```

Задачи (метод выполнения)

- Задание для самостоятельной работы №3



The screenshot shows a Java IDE with a series of `println` statements and their corresponding outputs. The statements are: `println(k ^ p)`, `println(sqrt(k))`, `println(k == p)`, `println(k != p)`, `println(k > p)`, and `println(k < p)`. The outputs are: `1419857`, `4.123105625617661`, `false`, `true`, `true`, and `false`. The IDE also shows execution times for the last two statements: `187 μs` for `println(k > p)` and `177 μs` for `println(k < p)`.

```
println(k ^ p)
1419857

println(sqrt(k))
4.123105625617661

println(k == p)
false

println(k != p)
true

println(k > p)
true
187 μs

println(k < p)
false
177 μs
```

Задачи (метод выполнения)

- Задание для самостоятельной работы №4

```
M = 3x3 Matrix{Int64}:
```

```
1 2 3  
4 5 6  
1 2 3
```

```
• M = [1 2 3; 4 5 6; 1 2 3]
```

```
N = 3x3 Matrix{Int64}:
```

```
7 8 9  
3 1 5  
1 2 3
```

```
• N = [7 8 9; 3 1 5; 1 2 3]
```

```
3x3 Matrix{Int64}:
```

```
8 10 12  
7 6 11  
2 4 6
```

```
• M + N
```

```
3x3 Matrix{Int64}:
```

```
6 6 6  
-1 -4 -1  
0 0 0
```

```
• N - M
```

```
3x3 Matrix{Int64}:
```

```
16 16 28  
49 49 79  
16 16 28
```

```
• M * N
```

```
3x3 adjoint(::Matrix{Int64}) with eltype Int64:
```

```
1 4 1  
2 5 2  
3 6 3
```

```
• M'
```


Результаты и их анализ

- Успешно удалось подготовить рабочее пространство и инструментарий для работы с языком программирования Julia и на простейших примерах познакомиться с основами синтаксиса Julia.



Благодарю за внимание