Julia. Установка и настройка. Основные принципы

Косолапов Степан [1]

11 ноября, 2023, Москва, Россия

[1] Российский Университет Дружбы Народов

Цели и задачи работы

Цель лабораторной работы

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

Процесс выполнения лабораторной работы

Функции ввода-вывода: read()

```
io = IOBuffer("JuliaLang is a GitHub organization");
read(io, String)
```

```
"JuliaLang is a GitHub organization"
```

readline()

```
input = readline()
    stdin>  Stepa

    "Stepa"
print(input)
    Stepa
```

write()

```
write("my_file.txt", "JuliaLang is a GitHub organization.\nIt has many members.
```

readlines()

```
lines = readlines("my_file.txt")
```

```
2-element Vector{String}:
"JuliaLang is a GitHub organization."
"It has many members."
```

readdlm()

```
using DelimitedFiles

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

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

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

readDLMLines = readdlm("delim_file.txt", '\t', Int, '\n')
```

```
4×2 Matrix{Int64}:
1  5
2  6
3  7
```

print() и println()

```
print("hello "); print("world"); print("!")
```

```
hello world!
```

```
println("hello "); println("world"); println("!")
```

```
hello
world
!
```

show()

```
struct Day
  n::Int
end

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

Day(1) # 1
```

Функция parse()

```
println(parse(Int, "1234")) # 1234
println(parse(Int, "1234", base = 5)) # 194
println(parse(Int, "101001", base = 2)) # 41
println(parse(Int, "afc", base = 16)) # 2812
println(parse(Float64, "1.2e-3")) # 0.0012
println(parse(Complex{Float64}, "3.2e-1 + 4.5im")) # 0.32 + 4.5im
println(parse(Bool, "0")) # false
println(parse(Bool, "false")) # false
println(parse(Bool, "1")) # true
println(parse(Bool, "true")) # true
```

Математические операции

```
println(1+1) # 2
println(1-1) # 0
println(2*2) # 4
println(2^2) # 4
println(sqrt(4)) # 2.0
println(2 > 1) # true
println(1<=3) # true
println(2==2) # true
println(2.3 + 1) # 3.3
println(sqrt(3.3) == 3.3^(1/2)) # true
println(1 | 1) # 1
println(parse(Int, "101010", base = 2) | parse(Int, "10101", base = 2) == parse(In
println(parse(Int, "101010", base = 2) & parse(Int, "10101", base = 2) == 0) # tru
println(parse(Int, "101011", base = 2) ⊻ parse(Int, "10101", base = 2) == 62) # tr
println(parse(Int, "101011", base = 2) >> 1 == parse(Int, "10101", base = 2)) # tr
```

Операции с массивами

```
using LinearAlgebra

println([1, 2, 3] + [1,2,3]) # [2, 4, 6]
println([1, 2, 3] - [1,2,3]) # [0, 0, 0]
println(dot([1,2,3], [1,2,3])) # 14
println([1, 2, 3] * 3) # [3, 6, 9]
println([1 2; 3 4] + [1 2; 3 4]) # [2 4; 6 8]
println([1 2; 3 4] - [1 2; 3 4]) # [0 0; 0 0]
println([1 2; 3 4] * [1 2; 3 4]) # [7 10; 15 22]
println(dot([1 2; 3 4], [1 2; 3 4])) # 30
println(transpose([1 2; 3 4])) # [1 3; 2 4]
println([1 2; 3 4] * 3) # [3 6; 9 12]
```

Выводы по проделанной работе

Вывод

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