

Лабораторная работа №1

Хохлачева Яна, НКНбд-01-18

Установить среду для работы с Julia и ознакомиться с основами синтаксиса языка Julia.

```
typeof(3), typeof(3.5), typeof(3/3.55), typeof(sqrt(3+4im)), typeof(pi)
```

```
(Int64, Float64, Float64, ComplexF64, Irrational{::π})
```

Специальные значения в Julia

```
1.0/0.0, 1.0/(-0.0), 0.0/0.0
```

```
(Inf, -Inf, NaN)
```

```
typeof(1.0/0.0), typeof(1.0/(-0.0)), typeof(0.0/0.0)
```

```
(Float64, Float64, Float64)
```

Рис. 1: Использование функции `typeof()` и специальные величины в Julia

Выполнение примеров

Ниже представлен код для определения крайних значений диапазонов целочисленных числовых величин

```
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]
UInt64: [0,18446744073709551615]
UInt128: [0,340282366920938463463374607431768211455]
```

Рис. 2: Определение крайних диапазонов целочисленных величин

В Julia преобразование типов можно реализовать или прямым указанием, или с использованием обобщённого оператора преобразования типов `convert()`

```
Int64(2.0), Char(2), typeof(Char(2))
```

```
(2, '\x02', Char)
```

```
convert{Int64, 2.0}, convert{Char, 2}
```

```
(2, '\x02')
```

Для приведения нескольких аргументов к одному типу, если это возможно, используется оператор `promote()`

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

```
Tuple{Float32, Float32, Float32}
```

Рис. 3: Преобразование типов

Методы определения функций

```
function f(x)
    x^2
end
```

f (generic function with 1 method)

f(5)

25

```
g(x)=x^2
```

g (generic function with 1 method)

Выполнение примеров

Пример определения одномерных массивов (вектор-строка и вектор-столбец) и обращение к их вторым элементам

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

(7, 2)

Пример определения двумерного массива (матрицы) и обращение к его элементам

```
a = 1; b = 2; c = 3; d = 4 # присвоение значений
Am = [a b; c d] # матрица 2 x 2
Am[1,1], Am[1,2], Am[2,1], Am[2,2] # элементы матрицы
```

(1, 2, 3, 4)

Пример выполнения операций над массивами (aa' — транспонирование вектора)

```
aa = [1 2]
AA = [1 2; 3 4]
aa*AA*aa'
```

```
1x1 Matrix{Int64}:
 27
```

```
aa, AA, aa'
```

([1 2], [1 2; 3 4], [1; 2])

Рис. 5: Примеры работы с массивами

Основные функции Julia для чтения / записи / вывода информации на экран

```
io = IOBuffer("Words")  
text = read(io, String)
```

```
"Words"
```

```
fileread = read("text.txt", String)
```

```
"word1\nword2\n3\n4"
```

```
println(fileread)
```

```
word1
```

```
word2
```

```
3
```

```
4
```

Рис. 6: Пример применения функции `read()`

Основные функции Julia для чтения / записи / вывода информации на экран

Функция `readline()` - чтение из командной строки, а также первой строки из файла

```
println("Как вас зовут?")  
name = readline()  
print("Привет, $name")
```

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

```
readline("text.txt")
```

```
"word1"
```

Функция `readlines()` - чтение из файла

```
lines = readlines("text.txt")  
println(lines)
```

```
["word1", "word2", "3", "4"]
```

Рис. 7: Пример применения функций `readline()` и `readlines()`

Основные функции Julia для чтения / записи / вывода информации на экран

Функция `show()` - выводит строку в кавычках, также можно вывести символ по номеру

```
show_example = show("String")
```

```
"String"
```

```
example = "String"
```

```
show(example[2])
```

```
't'
```

Функция `write()` - вывод и указание числа символов, а также запись в файл

```
write(stdout, "Something")
```

```
Something
```

```
9
```

```
write("text1.txt", "Students")
```

```
println(readlines("text1.txt"))
```

```
["Students"]
```

Рис. 8: Пример применения функций `show()` и `write()`

Основные функции Julia для чтения / записи / вывода информации на экран

Функция `parse()` и пример использования

```
liniya = "312"  
typeof(liniya)
```

String

```
println("Введите новое число")  
d = Meta.parse(readline())  
println("Число - ", d)  
println("Тип - ", typeof(d))
```

```
Введите новое число  
stdin> 12  
Число - 12  
Тип - Int64
```

Рис. 9: Пример применения функции `parse()`

Синтаксис для базовых математических операций

<code>5 + 3</code>	<code>10 ^ 2</code>
8	100
<code>5 - 4</code>	<code>21 % 2</code>
1	1
<code>5 * 10</code>	<code>11 ÷ 2 # \div + tab</code>
50	5
<code>10 / 5</code>	<code>sqrt(4) # unu \sqrt + tab</code>
2.0	2.0

Рис. 10: Базовые математические операции в Julia

```
false && true # AND
```

```
false
```

```
false || true # OR
```

```
true
```

```
1.0 == 1 # Равенство значений
```

```
true
```

```
1.0 === 1 # Сравнение программного представления
```

```
false
```

Рис. 11: Операторы сравнения в Julia

Заполнение матрицы по строкам и столбцам

```
AA = [1 2 3; 4 5 6; 7 8 9]
```

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

```
1 2 3  
4 5 6  
7 8 9
```

```
BB = [[1, 2, 3] [4, 5, 6] [7, 8, 9]]
```

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

```
1 4 7  
2 5 8  
3 6 9
```

Вектор-строка и вектор-столбец

```
a1 = [1 2 3]
```

```
1x3 Matrix{Int64}:
```

```
1 2 3
```

```
a2 = [1, 2, 3]
```

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

```
1  
2  
3
```

Рис. 12: Объявление массивов и векторов

Транспонирование матрицы

CC = AA'

3×3 adjoint(::Matrix{Int64}) with eltype Int64:

```
1  4  7
2  5  8
3  6  9
```

Сложение и вычитание матриц

AA + BB

3×3 Matrix{Int64}:

```
2  6 10
6 10 14
10 14 18
```

AA - BB

3×3 Matrix{Int64}:

```
0 -2 -4
2  0 -2
4  2  0
```

Рис. 13: Транспонирование матрицы, сложение и вычитание

Инструменты для работы с матрицами

Перемножение матриц, умножение матрицы на вектор и число

AA * BB

3x3 Matrix{Int64}:

```
14  32  50
32  77 122
50 122 194
```

AA * a2

3-element Vector{Int64}:

```
14
32
50
```

AA * 2

3x3 Matrix{Int64}:

```
2  4  6
8 10 12
14 16 18
```

Умножение вектора на число и скалярное произведение векторов

a1 * 3

1x3 Matrix{Int64}:

```
3  6  9
```

a1 * a2

1-element Vector{Int64}:

```
14
```

Рис. 14: Операции с векторами и матрицами

Во время выполнения лабораторной работы я подготовила инструментарий для работы и ознакомилась с языком Julia для дальнейшей работы.