

Презентация по лабораторной работе №1.

Компьютерный практикум по статистическому
анализу данных

Ле Тиен Винь

Информация

- Ле Тиен Винь
- студент
- Российский университет дружбы народов
- 1032215241@pfur.ru
- <https://github.com/tvle2000/information>



vinh

I.Цель работы

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

Подготовка инструментария к работе

- Установим Julia (<https://julialang.org/>) и Jupyter (<https://jupyter.org/>) под вашу операционную систему.

Задание

Основы синтаксиса Julia на примерах

1. Определим тип числовой величины.
2. В Julia введены специальные значения `Inf`, `-Inf`, `NaN`, обозначающие бесконечность и отсутствие какого-либо значения. Такие значения могут получаться в результате операций типа деления на ноль, а также могут быть допустимой частью выражений, поскольку в языке имеют тип вещественного числа.
3. Для определения крайних значений диапазонов целочисленных числовых величин можно воспользоваться следующим кодом:

Выполнения работы

Last login: Wed Nov 13 21:16:08 on ttys000

[(base) letienvinh@MacBook-Air-cua-Le ~ % julia

```
  ( )      - ( )
 ( )      - ( ) ( )
  - -      - - - -
 | | | | | | | / - \
 | | | | | | | ( | |
 - / | \ - - ' - - | \ - - ' -
 | - /
```

Documentation: <https://docs.julialang.org>

Type "?" for help, "]? " for Pkg help.

Version 1.11.1 (2024-10-16)

Official <https://julialang.org/> release

julia> █


```
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]
```



```
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)
```

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

```
(Int64, Float64, Float64, ComplexF64, Irrational{: $\pi$ })
```

Выполнения работы

Основы синтаксиса Julia на примерах - В Julia преобразование типов можно реализовать или прямым указанием, например вещественное число 2.0 преобразовать в целое, а число 2 в символ или использовать обобщённый оператор преобразования типов `convert()`. Преобразование 1 в булево `true`, 0 — в булево `false`.

- Для приведения нескольких аргументов к одному типу, если это возможно, используется оператор `promote()`.
- Способы определения функций
- Пример определения одномерных массивов (вектор-строка и вектор-столбец) и обращение к их 3-ым элементам
- Пример определения двумерного массива (матрицы) и обращение к его элементам.
- Пример выполнения операций над массивами (`aa'` — транспонирование вектора).

Выполнения работы

```
Bool(1), Bool(0)
```

```
(true, false)
```

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

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

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

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

```
promote(Int8(1), Float16(4.5), Float32(4.1))
```

```
(1.0f0, 4.5f0, 4.1f0)
```

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

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

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

f (generic function with 1 method)

```
f(4)
```

16

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

```
g (generic function with 1 method)
```

```
g(10)
```

```
100
```



```
a = [5 7 9]
```

```
b = [2,4,6]
```

```
a[3], b[3]
```

(9, 6)

```
: a=2;b=4;c=6;d=8  
Am = [a b;c d]
```

```
: 2x2 Matrix{Int64}:  
  2  4  
  6  8
```

```
: Am[1,1], Am[1,2], Am[2,1], Am[2,2]
```

```
: (2, 4, 6, 8)
```

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

1x1 Matrix{Int64}:
84

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

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

Задания для самостоятельной работы

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

Задания для самостоятельной работы

```
write("file.txt", "Hello World")
```

```
11
```

```
k = open("file.txt")
```

```
s = read(k, String)
```

```
"Hello World"
```

```
readlines("file.txt")
```

```
1-element Vector{String}:
```

```
"Hello World"
```

```
readlines("file.txt")
```

```
2-element Vector{String}:
```

```
"Hello World"
```

```
"My name is Vinh "
```

```
using DelimitedFiles  
readdlm("file.txt")
```

```
println(" My name is Vinh")  
println("i am 22")  
print("and i am Vietnamese ")
```

```
My name is Vinh  
i am 22  
and i am Vietnamese
```

```
show("i am Vietnamese")
```

```
"i am Vietnamese"
```

```
parse(type, str; base)
```

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
```

```
julia> parse{Int, "afc", base = 16}
2812
```

```
julia> parse{Float64, "1.2e-3"}
0.0012
```

```
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`.

```
x=Int64(6)
y=Float64(4.0)
```

4.0

addition=x+y

10.0

subtraction=x-y

2.0

multiplication= x*y

24.0

division=x/y

1.5

power=r= x^y

1296.0

sqrt_x=sqrt(x)

2.449489742783178

sqrt_y= sqrt(y)

2.0

```
v1 = [1,2]
v2 = [5,6]
N1 = [1 2;3 4]
M1 = [5 6;2 1];
```

v1 + v2

2-element Vector{Int64}:

```
6
8
```

N1 + M1

2×2 Matrix{Int64}:

```
6 8
5 5
```

v1 - v2

2-element Vector{Int64}:

```
-4
-4
```

N1 - M1

2×2 Matrix{Int64}:

```
-4 -4
1  3
```

```
v1[1,1]*v2[1,1]+v1[2,1]*v2[2,1]
```

17

transpose(N1)

2×2 transpose(::Matrix{Int64}) with eltype Int64:

```
1 3
2 4
```

transpose(v1)

1×2 transpose(::Vector{Int64}) with eltype Int64:

```
1 2
```

M1*3

2×2 Matrix{Int64}:

```
15 18
6  3
```

v2*3

2-element Vector{Int64}:

```
15
18
```


Задания для самостоятельной работы

- 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.

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

Задания для самостоятельной работы

```
parse(type, str; base)
```

```
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 `"i"`, 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
```

```
julia> parse{Int, "afc", base = 16}
2812
```

```
julia> parse{Float64, "1.2e-3"}
0.0012
```

```
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.
```

```
x=Int64(6)
y=Float64(4.0)
```

```
4.0
```

```
addition=x+y
```

```
10.0
```

```
subtraction=x-y
```

```
2.0
```

```
multiplication= x*y
```

```
24.0
```

```
division=x/y
```

```
1.5
```

```
power=x^y
```

```
1296.0
```

```
sqrt_x=sqrt(x)
```

```
2.449489742783178
```

```
sqrt_y= sqrt(y)
```

```
2.0
```

```
v1 = [1,2]
v2 = [5,6]
N1 = [1 2;3 4]
M1 = [5 6;2 1];
```

v1 + v2

2-element Vector{Int64}:

6
8

N1 + M1

2×2 Matrix{Int64}:

6 8
5 5

v1 - v2

2-element Vector{Int64}:

-4
-4

N1 - M1

2×2 Matrix{Int64}:

-4 -4
1 3

```
v1[1,1]*v2[1,1]+v1[2,1]*v2[2,1]
```

17

transpose(N1)

2×2 transpose(::Matrix{Int64}) with eltype Int64:

1 3
2 4

transpose(v1)

1×2 transpose(::Vector{Int64}) with eltype Int64:

1 2

M1*3

2×2 Matrix{Int64}:

15 18
6 3

v2*3

2-element Vector{Int64}:

15
18

Вывод

В ходе данной лабораторной работы я подготовил рабочее пространство и инструментарий для работы с языком программирования Julia, на простейших примерах познакомился с основами синтаксиса Julia.