

# Отчёт по лабораторной работе №1

Компьютерный практикум по статистическому анализу данных

Ле Тиен Винь

## Содержание

I. Цель работы .....	1
II. Постановка задачи.....	1
III. Выполнение работы.....	2
Пункт I .....	2
Пункт II .....	2
Пункт III.....	5
Выводы.....	10

## 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.

## II. Постановка задачи

I. Установите под свою операционную систему Julia, Jupyter. II. Используя Jupyter Lab, повторите примеры. III. Выполните задания для самостоятельной работы.



2. Проверил специальные значения Inf, -Inf, NaN и определял их тип

```
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)
```

3. Определял крайних значений диапазонов целочисленных числовых величин

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

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

4. Преобразовал типов можно реализовать или прямым указанием

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

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

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

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

5. Преобразовал 1 в булевое true, 0 — в булевое false

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

```
(true, false)
```

6. Для приведения нескольких аргументов к одному типу, если это возможно, используется оператор `promote()` и определял их тип

```
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}
```

7. Определи функцию  $f(x)$  возведения переменной  $x$  в квадрат и возведём в квадрат число 4

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

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

```
f(4)
```

```
16
```

8.

Другой способ определения несложных функций

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

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

```
g(10)
```

```
100
```

8. Определял одномерных массивов (вектор-строка и вектор-столбец) и обращение к их 3-ым элементам

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

```
|: (7, 2)
```

9. Определял двумерного массива (матрицы) и обращение к его элементам

```
a=1;b=2;c=3;d=4
Am = [a b;c d]
```

```
2×2 Matrix{Int64}:
 1  2
 3  4
```

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

```
(1, 2, 3, 4)
```

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

10. Выполнял операций над массивами (aa' — транспонирование вектора)

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

```
: 1×1 Matrix{Int64}:
 27
```

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

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

## Пункт III

1. Изучите документацию по основным функциям Julia для чтения / записи / вывода информации на экран: read(), readline(), readlines(), readlm(), print(), println(), show(), write(). Приведите свои примеры их использования, поясняя особенности их применения.

+ Функция write():

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

11

+ Функция read():

```
k = open("file.txt")  
s = read(k, String)
```

"Hello World"

+ Функция readlines():

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

1-element Vector{String}:  
"Hello World"

+Функция readlines():

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

2-element Vector{String}:  
"Hello World"  
"My name is Vinh "

+ Функция readlm():

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

+ Функция print() и println():

```
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
```

+ Функция show():

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

```
"i am Vietnamese"
```

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

```
?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`.

---

3. 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.

```
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
```

```
# Больше, меньше
is_greate r=x < y
```

```
false
```

```
is_less=x >y
```

```
true
```

```
# Логическое НЕ (NOT)
x != y
```

```
true
```

```
z = Int64(5)
```

```
5
```

```
x | z
```

```
7
```

```
x & z
```

```
4
```

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



```
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
```

# Выводы

Познакомился с основами синтаксиса Julia. s