Лабораторная работа № 1 Установка и настройка. Основные принципы

Адабор Кристофер Твум

1032225824

НКНбд-01-22

Основы работы в блокноте Jupyter

[3]:

```
2+3
```

[3]:

```
5
```

[5]:

```
3+4
1+2
```

[5]:

```
3
```

[6]:

```
3+5
4+5
```

[6]:

```
9
```

Markdown Notebook

Простейшие операции на языке Julia в Jupyter Lab ¶

[7]:

```
?println
```

```
search: println print pointer sprint printstyled
```

[7]:

```
println([io::IO], xs...)
```

Print (using `print`) `xs` to `io` followed by a newline. If `io` is not supplied, prints t
the default output stream `stdout`.

See also `printstyled` to add colors etc.

Examples

```
julia> println("Hello, world")
Hello, world

julia> io = IOBuffer();

julia> println(io, "Hello", ',', " world.")
```

Например, определим функцию $f(x)$ возведения переменной x в квадрат и возведём в квадрат число 4 (см. рис. 1.7):

[3]:

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

[3]:

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

[4]:

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

[4]:

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

[5]:

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

[5]:

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

[9]:

```
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]
```

Примеры приведения аргументов к одному типу.

Примеры определения типа числовых величин

```
[10]: Int64(2.0), Char(2), typeof(Char(2))
```

```
[10]: (2, '\x02', Char)
```

```
[11]: convert(Int64,2.0), convert(Char,2)
```

```
[11]: (2, '\x02')
```

```
[12]: typeof(promote(Int8(1), Float16(4.5), Float32(4.1)))
```

```
[12]: Tuple{Float32, Float32, Float32}
```

Примеры определения функций.

Примеры приведения аргументов к одному типу

[13]:
```
function f(x)
x^2
end
```

[13]:
```
f (generic function with 1 method)
```

[14]:
```
f(4)
```

[14]:
```
16
```

[15]:
```
g(x)=x^2
```

[15]:
```
g (generic function with 1 method)
```

[16]:
```
g(8)
```

[16]:
```
64
```

Примеры работы с массивами.

Рис. 1.7. Примеры определения функций

[18]:
```
a = [4 7 6] # вектор-строка
b = [1, 2, 3] # вектор-столбец
a[2], b[2] # вторые элементы векторов a и b соответственно
```

[18]:
```
(7, 2)
```

[19]:
```
a = 1; b = 2; c = 3; d = 4 # присвоение значений
Am = [a b; c d] # матрица 2 x 2
```

[19]:
```
2×2 Matrix{Int64}:
 1  2
 3  4
```

[20]:
```
Am[1,1], Am[1,2], Am[2,1], Am[2,2] # элементы матрицы Am
```

[20]:
```
(1, 2, 3, 4)
```

[21]:
```
aa = [1 2] # вектор-строка
AA = [1 2; 3 4] # матрица 2 x 2
aa*AA*aa' # умножение вектора-строки на матрицу и на вектор-столбец (операция транспонировани
```

[21]:
```
1×1 Matrix{Int64}:
 27
```

[23]:
```
aa, AA, aa'
```

[23]:
```
([1 2], [1 2; 3 4], [1; 2;;])
```

Задания для саомстоятельно работы

12 Лабораторная работа № 1. Julia. Установка и настройка. Основные принципы. 1.3.4. Задания для самостоятельной работы

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

[45]:
```
# Пункт 1
read("file.txt", String) # считывает содержимое файла file.txt как строку
```

[45]:
```
"Первая строка\nВторая строка\nТретья строка\nПоследняя строка(полный файл)"
```

[46]:
```
readline("file.txt") # считывает первую строку из файла file.txt
```

[46]:
```
"Первая строка"
```

[47]:
```
readlines("file.txt") # считывает все строки из файла file.txt
```

[47]:
```
4-element Vector{String}:
 "Первая строка"
 "Вторая строка"
 "Третья строка"
 "Последняя строка(полный файл)"
```

[ ]:
```
readdlm("data.csv", ',') # считывает CSV-файл с разделителем-запятой
```

```
[ ]: readdlm("data.csv", ',') # считывает CSV-файл с разделителем-запятой
```

```
[50]: print("Hello, ")
      println("world!")
      # выводят "Hello, world!" с переводом строки после "world!"
```

```
Hello, world!
```

```
[51]: show([1, 2, 3]) # отображает массив [1, 2, 3]
```

```
[1, 2, 3]
```

```
[52]: write("file.txt", "Hello World") # записывает строку "Hello World" в файл file.txt
```

```
[52]: 11
```

2. Изучите документацию по функции parse(). Приведите свои примеры её использо- вания, поясняя особенности её применения.

```
[3]: # Пункт 2
     parse(Int, "123") # преобразует строку "123" в целое число 123
```

```
[3]: 123
```

3. Изучите синтаксис Julia для базовых математических операций с разным типом пере- менных: сложение, вычитание, умножение, деление, возведение в степень, извлечение корня, сравнение, логические операции. Приведите свои примеры с пояснениями по особенностям их применения.

```
# Пункт3
# Целочисленные переменные
a = 5
b = 2

# Операции
addition = a + b       # Сложение: 7
subtraction = a - b    # Вычитание: 3
multiplication = a * b # Умножение: 10
division = a / b       # Деление: 2.5
```

```
2.5
```

```
# Возведение в степень
power = a ^ b         # 5 ^ 2 = 25
```

```
25
```

```
# Извлечение корня
sqrt_b = sqrt(b)      # Квадратный корень из 2 = 1.41421
```

```
1.4142135623730951
```

In [66]:
```
# Равенство
is_equal = (a == b)  # false

# Неравенство
is_not_equal = (a != b) # true

# Больше, меньше
is_greater = (a > b) # true
is_less = (a < b)    # false
```

Out[66]: false

In [67]:
```
# Логическое И (AND)
logical_and = (a > 1) && (b < 3)  # true

# Логическое ИЛИ (OR)
logical_or = (a < 1) || (b < 3)   # true

# Логическое НЕ (NOT)
logical_not = !(a == b)           # true
```

Out[67]: true

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

In [68]:
```
# Пункт 4
# Векторы
v1 = [1, 2, 3]
v2 = [4, 5, 6]

# Матрица
m1 = [1 2; 3 4]
```

Out[68]:
```
2×2 Matrix{Int64}:
 1  2
 3  4
```

In [69]:
```
# Сложение векторов
vector_add = v1 + v2  # [5, 7, 9]

# Сложение матриц
matrix_add = m1 + m1  # [2 4; 6 8]

# Вычитание векторов
vector_sub = v1 - v2  # [-3, -3, -3]
```

Out[69]:
```
3-element Vector{Int64}:
 -3
 -3
 -3
```

In [ ]:
```
# Скалярное произведение векторов
dot_product = dot(v1, v2)  # 32 (1*4 + 2*5 + 3*6)
```

In [72]:
```
# Транспонирование матрицы
transpose_m1 = transpose(m1)  # [1 3; 2 4]
```

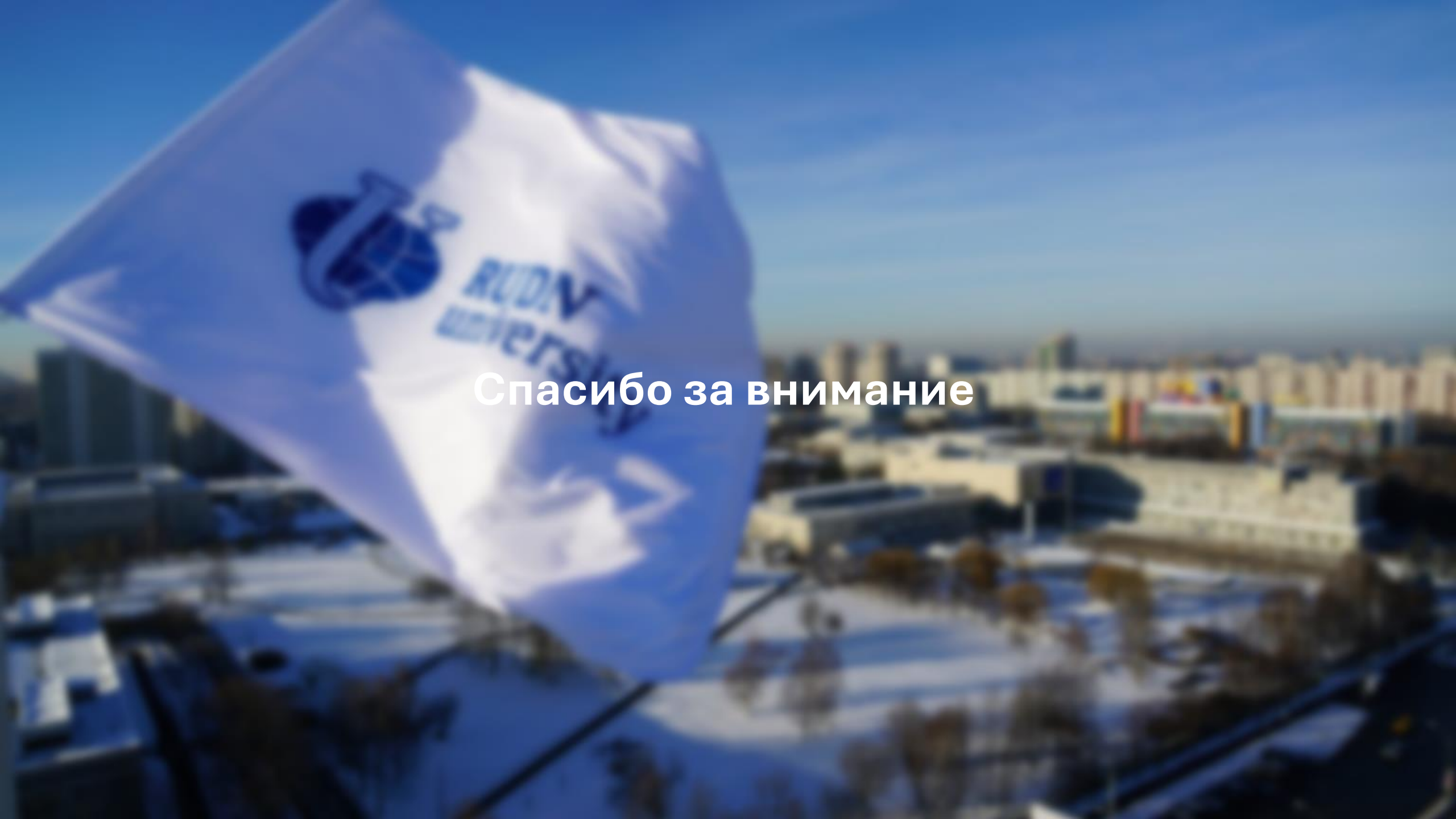
Out[72]:
```
2×2 transpose(::Matrix{Int64}) with eltype Int64:
 1  3
 2  4
```

[73]:
```
# Умножение вектора на скаляр
scalar_mult_vector = 3 * v1   # [3, 6, 9]

# Умножение матрицы на скаляр
scalar_mult_matrix = 2 * m1   # [2 4; 6 8]
```

t[73]:
```
2×2 Matrix{Int64}:
 2  4
 6  8
```

[11]:
```
# Умножение матриц
m2 = [5 6; 7 8]
matrix_mult = m1 * m2 # Результат - матрица 2x2
```


Спасибо за внимание