Julia. Установка и настройка. Основные принципы

Лабораторная работа № 1

Шулуужук Айраана НПИбд-02-22

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

Установим под свою операционную систему Julia, Jupyter (рис. 2.1) (рис. 2.2)

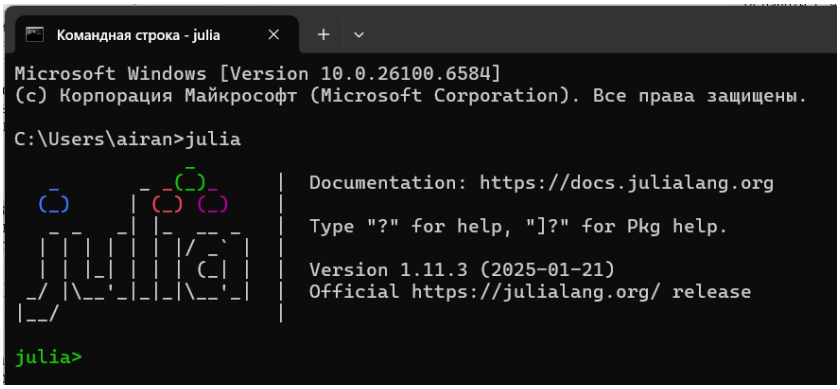

Рис. 2.1: Julia

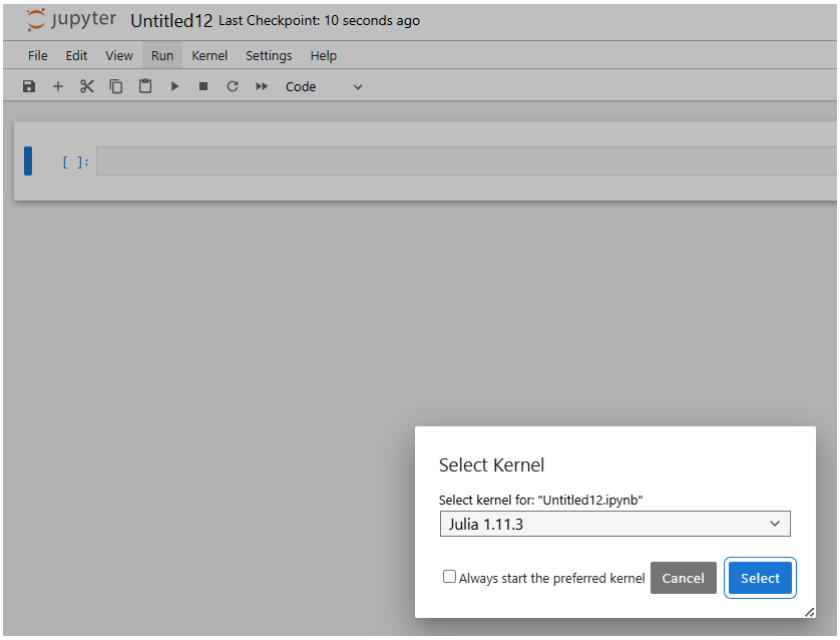

Рис. 2.2: Jupyter Lab

Используя Jupyter Lab, повторим примеры из раздела (рис. 2.3) (рис. 2.4) (рис. 2.5) (рис. 2.6)

Примеры определения типа числовых данных

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

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

```
[10]: 1.0/0.0, 1.0/(-0.0), 0.0/0.0
```

```
[10]: (Inf, -Inf, NaN)
```

```
[12]: typeof(1.0/0.0,), typeof(1.0/(-0.0)), typeof(0.0/0.0 )
```

```
[12]: (Float64, Float64, Float64)
```

```
[14]: 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. 2.3: Примеры определения типа числовых величин

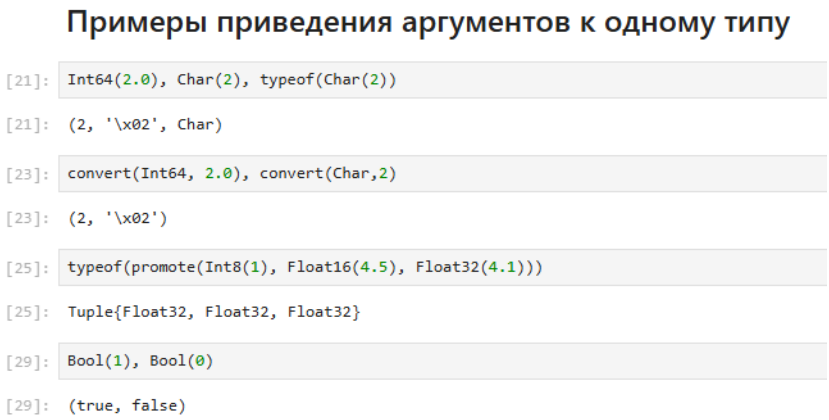

Рис. 2.4: Примеры приведения аргументов к одному типу

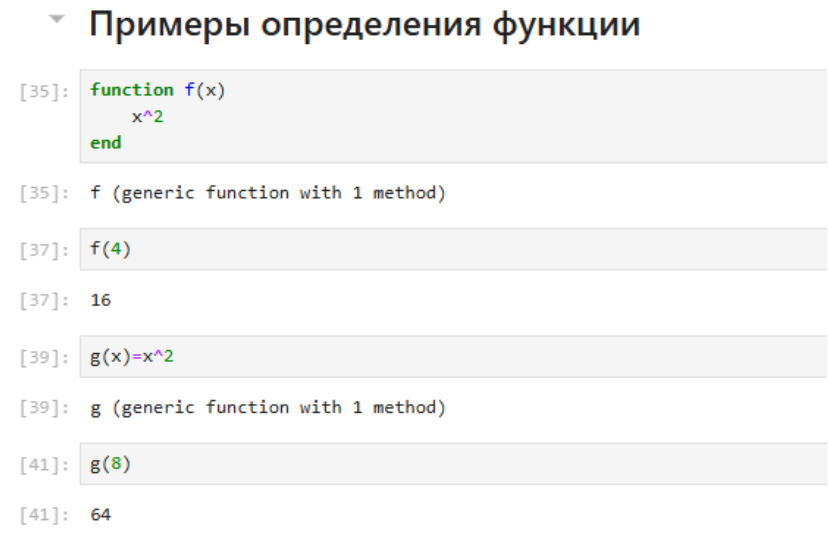

Рис. 2.5: Примеры определения функций

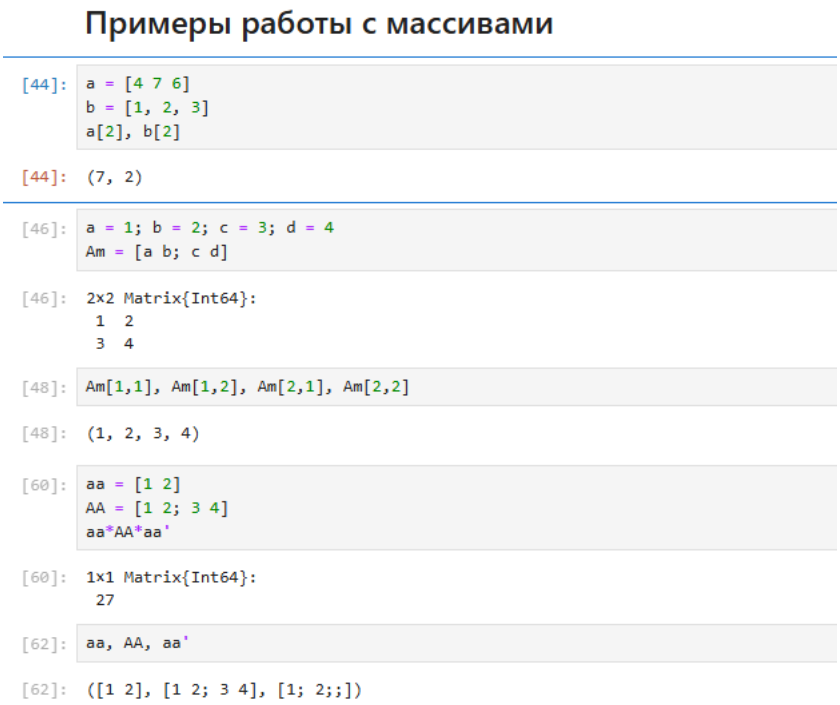

Рис. 2.6: Примеры работы с массивами

Выполним задания для самостоятельной работы

1. **Изучим 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.**

Функции print() выводит без перехода на новую строку. println(): выводит с переходом на новую строку (рис. 2.7)

```
[5]: # print() и println()
print("Hello, ")
println("Julia!")
println("Julia!")

Hello, Julia!
Julia!
```

Рис. 2.7: Функции print() и println()

show(): выводит представление объекта, часто более “техническое” (рис. 2.8).

```
]: # show()
x = [5,21,3]
show(x)

[5, 21, 3]
```

Рис. 2.8: Функция show()

read(): читает байты или символы из файла или стандартного ввода (рис. 2.9).

```
# read()
data_bytes = open("test.txt") do f
    read(f)
end
```

```
53-element Vector{UInt8}:
 0xd0
 0xad
 0xd1
 0x82
 0xd0
 0xbe
 0x20
 0xd1
 0x81
 0xd1
 0x82
 0xd1
 0x80
    ⋮
 0x20
 0xd0
 0xb2
 0x20
 0xd1
 0x84
 0xd0
 0xb0
 0xd0
 0xb9
 0xd0
 0xbb
```

Рис. 2.9: Функция read()

readline(): читает одну строку из файла или стандартного ввода. readlines(): читает все строки из файла или ввода в массив строк (рис. 2.10).

```
# readline() - чтение одной строки
line = open("test.txt") do f
    readline(f)
end
```

```
"Это строка, записанная в файл"
```

```
# readlines() - чтение всех строк
lines = open("test.txt") do f
    readlines(f)
end
```

```
1-element Vector{String}:
 "Это строка, записанная в файл"
```

Рис. 2.10: Функции readline(), readlines()

2. **Изучим документацию по функции parse(). Приведем свои примеры её использования, поясняя особенности её применения. Используется для преобразования строки в число или другой тип данных (рис. 2.11).**

```
: parse(Float64, "3.14")
```

```
: 3.14
```

```
: parse(Int, "0987654432")
```

```
: 987654432
```

Рис. 2.11: Функция parse()

3. **Изучим 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 (рис. 2.12) (рис. 2.13).

```
[137]: #сложение
       5+4

[137]: 9

[139]: #вычитание
       235-12

[139]: 223

[141]: #умножение
       4*2

[141]: 8

[152]: #деление
       9/4

[152]: 2.25

[156]: #деление с остатком
       9%4

[156]: 1

[162]: #возведение в степень
       12^2

[162]: 144

[164]: #извлечение корня
       sqrt(225)

[164]: 15.0
```

Рис. 2.12: сложение, вычитание, умножение, деление, возведение в степень, извлечение корня

```
[174]: #комплексные числа
       z = 6 + 8im
       abs(z)

[174]: 10.0

[176]: angle(z)

[176]: 0.9272952180016122

[180]: #сравнения и логические операции
       10 > 8

[180]: true

[182]: 10 == 8

[182]: false

[184]: 10 != 8

[184]: true

[186]: !true

[186]: false

[188]: true&&false

[188]: false
```

Рис. 2.13: сравнение, логические операции

4. **Приведите несколько своих примеров с пояснениями с операциями над матрицами и векторами: сложение, вычитание, скалярное произведение, транспонирование, умножение на скаляр (рис. 2.14) (рис. 2.15)**

```
[200]: v1 = [1, 2, 3] #вектор столбец
       v2 = [4 5 6] #вектор сторока

[200]: 1×3 Matrix{Int64}:
        4  5  6

[204]: v1 + v1

[204]: 3-element Vector{Int64}:
        2
        4
        6

[206]: v1 - v1

[206]: 3-element Vector{Int64}:
        0
        0
        0

[208]: v1 * 2

[208]: 3-element Vector{Int64}:
        2
        4
        6

[210]: 2 * v1

[210]: 3-element Vector{Int64}:
        2
        4
        6

[220]: v1'

[220]: 1×3 adjoint(::Vector{Int64}) with eltype Int64:
        1  2  3
```

Рис. 2.14: операции над матрицами

```
[224]: A = [1 2; 3 4]
       B = [5 6; 7 8]

       A + B

[224]: 2×2 Matrix{Int64}:
         6   8
        10  12

[226]: A - B

[226]: 2×2 Matrix{Int64}:
        -4  -4
        -4  -4

[228]: A * B

[228]: 2×2 Matrix{Int64}:
        19  22
        43  50
```

Рис. 2.15: операции над матрицами

3 Выводы

В результате выполнения лабораторной работы подготовили рабочее пространство и инструментарий для работы с языком программирования Julia, на простейших примерах познакомились с основами синтаксиса Julia.