Отчёт по лабораторной работе №1

Компьютерный практикум по статистическому анализу данных

Канева Екатерина, НФИбд-02-22

Содержание

1 Цель работы **5**

2 Задание **6**

3 Теоретическая часть **7**

4 Выполнение лабораторной работы **8**

5 Выводы **15**

Список иллюстраций

4.1 Julia. 8
4.2 Примеры определения типа числовых величин. 8
4.3 Примеры приведения аргументов к одному типу. 9
4.4 Примеры определения функций. 9
4.5 Примеры работы с массивами. 10
4.6 Изучение документации. 10
4.7 Использование read, realine и readlines. 11
4.8 Использование write и show. 11
4.9 Использование print и println. 11
4.10 Использование parse. 12
4.11 Использование арифметических операций. 13
4.12 Использование другие операции с числами. 13
4.13 Операции с матрицами. 14

Список таблиц

1 Цель работы

Подготовить рабочее пространство и инструментарий для работы с языком программирования Julia, на простейших примерах познакомиться с основами синтаксиса Julia.

2 Задание

- Установить Julia
- Познакомиться с синтаксисом языка Julia
- Выполнить задания для самостоятельной работы

3 Теоретическая часть

Julia - высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков, однако имеет некоторые существенные отличия.

Для выполнения заданий была использована официальная документация Julia.

4 Выполнение лабораторной работы

У меня уже была установлена Julia (она была установлена через chocolatey), поэтому я сразу приступила к знакомству с синтаксисом (рис. 4.1):

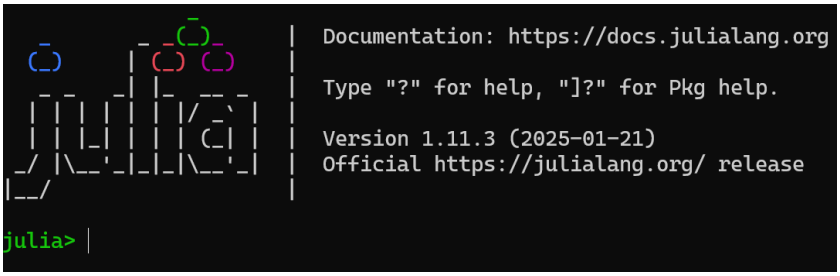

Рис. 4.1: Julia.

Сначала я выполнила примеры с определением типа числовых величин (рис. 4.2):

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

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

[12]: 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. 4.2: Примеры определения типа числовых величин.

Потом я выполнила примеры с приведением аргументов к одному типу (рис. 4.3):

```
[16]: Int64(2.0), Char(2)
[16]: (2, '\x02')
[18]: typeof(Char(2))
[18]: Char
[20]: convert(Int64, 2.0), convert(Char,2)
[20]: (2, '\x02')
[28]: typeof(promote(Int8(1), Float16(4.5), Float32(4.1)))
[28]: Tuple{Float32, Float32, Float32}
```

Рис. 4.3: Примеры приведения аргументов к одному типу.

Далее я выполнила примеры с определением функций (рис. 4.4):

```
[30]: function f(x)
          x^2
      end
[30]: f (generic function with 1 method)
[32]: f(3)
[32]: 9
[34]: g(x) = x^2
[34]: g (generic function with 1 method)
[36]: g(3)
[36]: 9
```

Рис. 4.4: Примеры определения функций.

Потом я поработала с массивами (рис. 4.5):

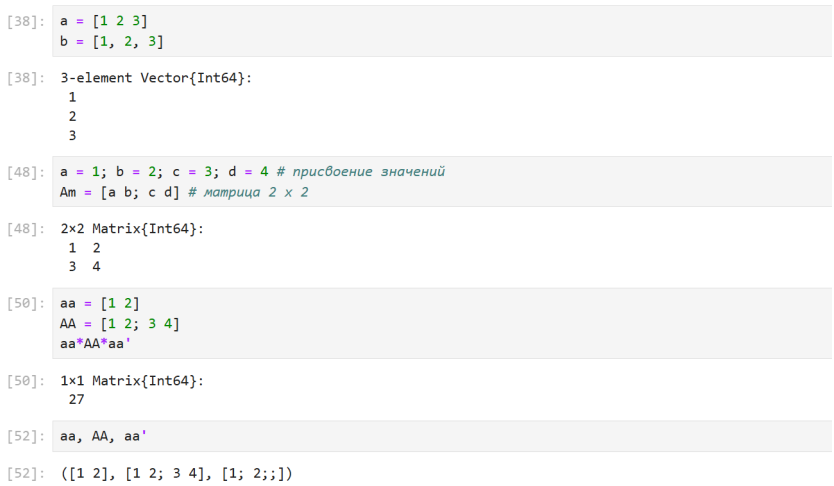

Рис. 4.5: Примеры работы с массивами.

Изучим документацию по основным функциям Julia для чтения / записи / вывода информации на экран: `read`, `readline`, `readlines`, `readdlm`, `print`, `println`, `show`, `write`. Приведем свои примеры их использования, поясняя особенности их применения.

Для того, чтобы ознакомиться с документацией достаточно поставить знак ? перед интересующей функцией. Пример с изучением документации о команде read() (рис. 4.6):

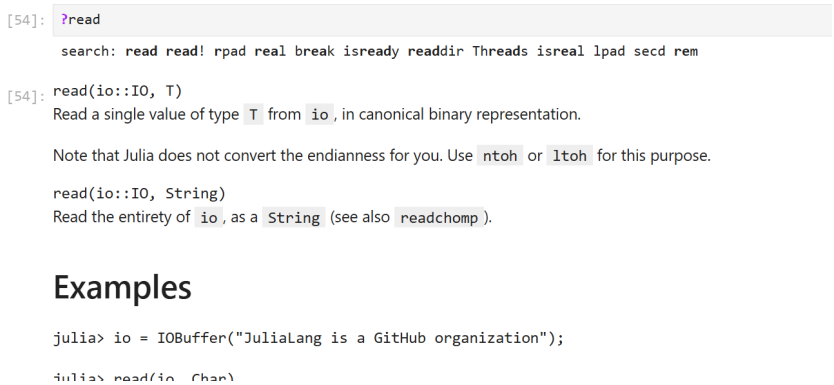

Рис. 4.6: Изучение документации.

Создадим текстовый файл с любым содержанием в папке, где мы работаем. Откроем его на чтение и прочитаем с помощью команды `read`. Текст вывелся в одну строку с разделителями `\r\n`. Также прочитаем текст используя функцию

`readline` - выведется только первая строка. Чтобы прочитать все строки в файле используем команду `readlines` (рис. 4.7):

```
[2]: file = open("example.txt", "r")
     data = read(file, String)

[2]: "Hello, world!\nSnova hellow, world!\nEscho raz hello, world!"

[4]: file = open("example.txt", "r")
     line = readline(file)

[4]: "Hello, world!"

[6]: file = open("example.txt", "r")
     lines = readlines(file)

[6]: 3-element Vector{String}:
      "Hello, world!"
      "Snova hellow, world!"
      "Escho raz hello, world!"
```

Рис. 4.7: Использование read, realine и readlines.

Далее проделаем действия с командами write и show (рис. 4.8):

```
[82]: io = IOBuffer()

[82]: IOBuffer(data=UInt8[...], readable=true, writable=true, seekable=true, append=false, size=0, maxsize=Inf, ptr=1, mark=-1)

[84]: write(io, 10)

[84]: 8

[86]: write(io, 3.14)

[86]: 8

[88]: write(io, "Hello, world!")

[88]: 13

[90]: write(io, 10) + write(io, 3.14)

[90]: 16

[92]: show("Hello, world!")
      "Hello, world!"
```

Рис. 4.8: Использование write и show.

Также выполню действия с print и println (рис. 4.9):

```
[8]: for i in 1:1:3
         print(i, " ")
     end

     1 2 3

[10]: for i in 1:1:3
          println(i, " ")
      end

      1
      2
      3
```

Рис. 4.9: Использование print и println.

После этого я изучила документацию по команде parse и выполнила несколько действий с ней (рис. 4.10):

```
[100]: parse(Int, "10", base = 2)

[100]: 2

[106]: parse(Bool, "1")

[106]: true
```

Рис. 4.10: Использование parse.

Потом я приступила к дйествиям с численными переменными: сложением, вычитанием, умножением, делением (рис. 4.11), возведение в степень, извлечение корня, сравнение, логические операции (рис. 4.12):

```
[110]: a = 1 + 2
       b = 3.5 + 4.5
       c = "Hello " * "World"
       print(a, '\n', b, '\n', c)

       3
       8.0
       Hello World

[112]: a = 1 - 2
       b = 3.5 - 4.5
       print(a, '\n', b)

       -1
       -1.0

[114]: a = 1 * 2
       b = 3.5 * 4.5
       print(a, '\n', b)

       2
       15.75

[116]: a = 1 / 2
       b = 3.5 / 4.5
       print(a, '\n', b)

       0.5
       0.7777777777777778
```

Рис. 4.11: Использование арифметических операций.

```
[122]: a = 5^2

[122]: 25

[124]: sqrt(64)

[124]: 8.0

[136]: a = 1 > 2
       b = 3.5 < 4.5
       c = 3 == 3.0
       d = 4 != 4.0
       e = true && false
       ab = true || false
       ac = !true
       print(a, '\n', b, '\n', c, '\n', d, '\n', e, '\n', ab, '\n', ac)

       false
       true
       true
       false
       false
       true
       false
```

Рис. 4.12: Использование другие операции с числами.

В заключение я проделала операции с матрицами: сложение, вычитание, скалярное произведение, транспонирование, умножение на скаляр (рис. 4.13):

```
[138]: A = [1 2; 3 4]
       B = [5 6; 7 8]
       C = [9 10]
       D = [11; 12]

[138]: 2-element Vector{Int64}:
        11
        12

[140]: AB = A + B

[140]: 2×2 Matrix{Int64}:
         6   8
        10  12

[142]: A * B

[142]: 2×2 Matrix{Int64}:
        19  22
        43  50

[150]: A'

[150]: 2×2 adjoint(::Matrix{Int64}) with eltype Int64:
        1  3
        2  4

[156]: A * 2

[156]: 2×2 Matrix{Int64}:
        2  4
        6  8
```

Рис. 4.13: Операции с матрицами.

5 Выводы

Познакомилась с синтаксисом языка Julia.