

Лабораторная работа 1

Julia. Установка и настройка. Основные принципы.

Ланцова Яна Игоревна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Установка Julia и знакомство с синтаксисом	7
3.2	Задание 1	9
3.3	Задание 2	10
3.4	Задание 3	11
3.5	Задание 4	12
4	Выводы	13

Список иллюстраций

3.1	Установка Julia	7
3.2	Установка пакетов	7
3.3	Выполнение примеров из лабораторной	8
3.4	Выполнение примеров из лабораторной	8
3.5	Выполнение примеров из лабораторной	8
3.6	Чтение файла	9
3.7	Вывод на печать	10
3.8	Команда записи	10
3.9	Примеры использования функции <code>parse()</code>	10
3.10	Примеры базовых математических операций	11
3.11	Примеры базовых математических операций	11
3.12	Примеры операций над матрицами	12
3.13	Примеры операций над матрицами	12

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 Задание

1. Установите под свою операционную систему Julia, Jupyter.
2. Используя Jupyter Lab, повторите примеры из раздела лабораторной работы.
3. Выполните задания для самостоятельной работы.

3 Выполнение лабораторной работы

3.1 Установка Julia и знакомство с синтаксисом

Установим Julia под мою операционную систему(рис. 3.1).

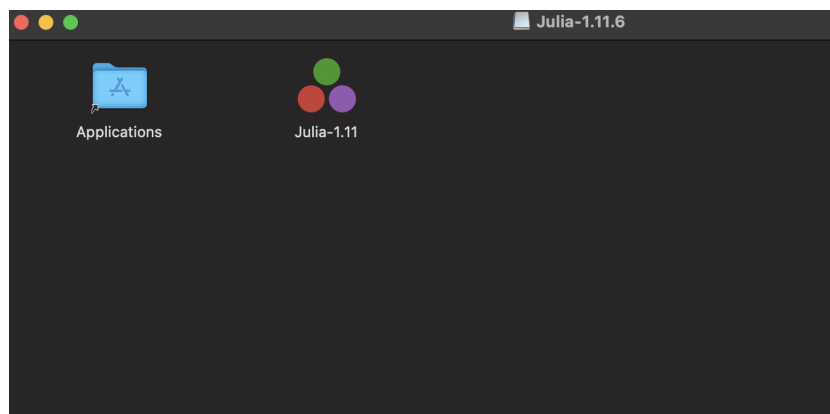


Рис. 3.1: Установка Julia

Скачаем необходимые пакеты для работы с Julia(рис. 3.2):

```
[julia> using Pkg
[julia> Pkg.add("IJulia")
  Updating registry at `~/.julia/registries/General.toml`
  Resolving package versions...
  No Changes to `~/.julia/environments/v1.11/Project.toml`
  No Changes to `~/.julia/environments/v1.11/Manifest.toml`

[julia> using IJulia

[julia> notebook()
[ Info: running setenv(`/Users/yana/.julia/conda/3/aarch64/bin/jupyter notebook`,`XPC_FLAGS=0x0`, `CE_M=`, `PATH=/Users/yana/.julia/conda/3/aarch64/bin:/Users/yana/anaconda3/condabin:/opt/homebrew/bin:/opt/homebrew/sbin:/opt/local/bin:/opt/local/sbin:/Library/Frameworks/Python.framework/Versions/3.12/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/Library/Frameworks/Python.framework/Versions/3.10/bin:/Library/Frameworks/Python.framework/Versions/3.9/bin:/usr/local/bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:/sbin:/Users/yana/Downloads/harbour/bin:/Applications/Vmware Fusion.app/Contents/Public:/Library/TeX/texbin:/usr/local/go/bin:/opt/homebrew/bin:/usr/local/share/dotnet:/usr/local/tools:/Library/Apple/usr/bin:/Applications/quarto/bin:/Applications/Postgres.app/Contents/Versions/latest/bin`, `PWD=/Users/yana`, `XPC_SERVICE_NAME=0`, `TERM_PROGRAM=Apple_Terminal`, `HOMEBREW_PREFIX=/opt/homebrew`, `SHELL=/bin/zsh`, `__CF_USER_TEXT_ENCODING=0x1F5:0x7:0x31`, `LC_CTYPE=UTF-8`, `__CFBundleIdentifier=com.apple.Terminal`, `TMPDIR=/var/folders/mj/y0wsc8sn0d5cpn85f8dg38dw0800gn/T/`, `CONDARC=/Users/yana/.julia/conda/3/aarch64/condarc-julia.yml`, `HOMEBREW_CELLAR=/opt/homebrew/Cellar`, `LOGNAME=yana`, `SHLVL=1`, `LaunchInstanceID=8E3A0041-B04A-4B0C-B4B4-5730D1D1D1C0`, `CONDA_PREFIX=/Users/yana/.julia/conda/3/aarch64`, `SSH_AUTH_SOCK=/private/tmp/com.apple.launchd.UhS791XT02/Listeners`, `TERM_SESSION_ID=61B3C30C-3032-4AAF-B3D8-7F48DF2E925B`, `OLDPWD=/Users/yana`, `INFOPATH=/opt/homebrew/share/info:/opt/homebrew/share/info:`, `=/Applications/Julia-1.11.app/Contents/Resources/julia/bin/julia`, `HOMEBREW_REPOSITORY=/opt/homebrew`, `OPENBLAS_DEFAULT_NUM_THREADS=1`, `CE_CONDA=`, `SECURITYSESSIONID=186b2`, `USER=yana`, `TERM=xterm-256color`, `HOME=/Users/yana`, `TERM_PROGRAM_VERSION=447`, `OPENBLAS_MAIN_FREE=1`, `PYTHONIOENCODING=UTF-8`)]
```

Рис. 3.2: Установка пакетов

Теперь повторим простейшие примеры для знакомства с синтаксисом Julia (рис. 3.3-3.5):

```
[1]: # Простейшие операции
[2]: 2+3
[2]: 5
[3]: 3+7
[3]: 11+88
[3]: 99
[4]: # Получение информации по функции println
[5]: println("Hello Julia!")
[5]: Hello Julia!
[12]: io = IOBuffer();
[12]: println(io, "Hello, world")
[13]: String(take!(io))
[13]: "Hello, world\n"
```

Рис. 3.3: Выполнение примеров из лабораторной

```
[16]: function f(x)
[16]:     x^2
[16]: end;
[18]: f(9)
[18]: 81
[17]: g(x)=x^2;
[19]: g(11)
[19]: 121
[25]: # вектор-строка
[25]: a = [4 7 6]
[25]: 1×3 Matrix{Int64}:
[25]: 4 7 6
[27]: # вектор-столбец
[27]: b = [1, 2, 3]
[27]: 3-element Vector{Int64}:
[27]: 1
[27]: 2
[27]: 3
[30]: a[1], b[1]
[30]: (4, 1)
```

Рис. 3.4: Выполнение примеров из лабораторной

```
[33]: a = 1; b = 2; c = 3; d = 4
[33]: Am = [a b; c d]
[33]: 2×2 Matrix{Int64}:
[33]: 1 2
[33]: 3 4
[34]: Am[1,1]
[34]: 1
[35]: Am'
[35]: 2×2 adjoint{::Matrix{Int64}} with eltype Int64:
[35]: 1 3
[35]: 2 4
```

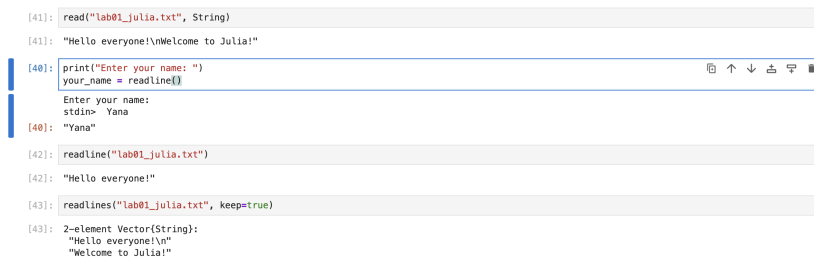
Рис. 3.5: Выполнение примеров из лабораторной

После выполнения примеров, перейдем к выполнению заданий.

3.2 Задание 1

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.

Создадим текстовый файл с любым содержанием в папке, где мы работаем. Откроем его на чтение и прочитаем с помощью команды `read()`. Данная функция в качестве первого параметра принимает файл или поток (иначе стоковый буффер). 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. 3.6).



```
[41]: read("lab01_julia.txt", String)
[41]: "Hello everyone!\nWelcome to Julia!"

[40]: print("Enter your name: ")
      your_name = readline()
Enter your name:
stdin> Yana
[40]: "Yana"

[42]: readline("lab01_julia.txt")
[42]: "Hello everyone!"

[43]: readlines("lab01_julia.txt", keep=true)
[43]: 2-element Vector{String}:
      "Hello everyone!\n"
      "Welcome to Julia!"
```

Рис. 3.6: Чтение файла

Далее посмотрим на работу `readdlm()`, `print()`, `println()` и `show()` (рис. 3.7). `readdlm()` считывает матрицу из указанного файла. `print()` может как вывести сообщение на экран, так и вывести его в поток. `println()` работает аналогично, но каждое новое сообщение выводит на следующую строчку. `show()` выводит текст на экран как строку.

```

[45]: using DelimitedFiles

[58]: readlm("lab01_julia.txt", Int)

[58]: 3x2 Matrix{Int64}:
 7  2
 4  5
 6  6

[62]: io = IOBuffer();
      print(io, "Yana the best")
      String(take!(io))

[62]: "Yana the best"

[66]: for i in 1:1:5
      print(i)
      end
      12345

[67]: for i in 1:1:5
      println(i)
      end
      1
      2
      3
      4
      5

[65]: show("I am 4 yo")
      "I am 4 yo"

```

Рис. 3.7: Вывод на печать

Посмотрим на работу функции `write()` (рис. 3.8). Данная функция может записывать данные в стоковый буффер и файл.

```

[71]: io = IOBuffer();
      write(io, "I am 4 yo")

[71]: 9

[72]: String(take!(io))

[72]: "I am 4 yo"

[73]: write(io, "I love" * write(io, " eating food"))

[73]: 18

[74]: String(take!(io))

[74]: "I love eating food"

[ ]:

```

Рис. 3.8: Команда записи

3.3 Задание 2

Изучим документацию по функции `parse()`. Данная функция преобразует строки в другие типы данных. Приведем свои примеры её использования, поясняя особенности её применения (рис. 3.9).

```

[75]: parse{Int, "1388"}

[75]: 1388

[76]: parse{Float64, "1.388"}

[76]: 1.388

[77]: parse{Int, "a", base = 16}

[77]: 10

[ ]:

```

Рис. 3.9: Примеры использования функции `parse()`

3.4 Задание 3

Изучим синтаксис Julia для базовых математических операций с разным типом переменных: сложение, вычитание, умножение, деление, возведение в степень, извлечение корня, сравнение, логические операции. Приведем примеры с пояснениями по особенностям их применения (рис. 3.10 - 3.11):

```
[87]: ex1 = 2 + 4;
      ex2 = 2.2 + 4.4;
      println(ex1, '\n', ex2)
      6
      6.6000000000000005

[88]: ex3 = 2 - 4;
      ex4 = 2.2 - 4.4;
      println(ex3, '\n', ex4)
      -2
      -2.2

[90]: ex5 = 10 * 5;
      ex6 = 1.21 * 10;
      println(ex5, '\n', ex6)
      50
      12.1

[91]: ex7 = 10 / 5;
      ex8 = 1.21 / 10;
      println(ex7, '\n', ex8)
      2.0
      0.121

[92]: ex9 = 10 ^ 5;
      ex10 = 1.1 ^ 2;
      println(ex9, '\n', ex10)
      100000
      1.2100000000000002
```

Рис. 3.10: Примеры базовых математических операций

```
[93]: ex11 = sqrt(121)
[93]: 11.0

[96]: first = 6;
      second = 9;
      first == second
[96]: false

[99]: first != second
[99]: true

[100]: first > second
[100]: false

[101]: first < second
[101]: true

[102]: true && false
[102]: false

[103]: true && true
[103]: true

[104]: true || false
[104]: true

[105]: false || false
[105]: false
```

Рис. 3.11: Примеры базовых математических операций

3.5 Задание 4

Приведем несколько примеров с пояснениями с операциями над матрицами и векторами: сложение, вычитание, скалярное произведение, транспонирование, умножение на скаляр (рис. 3.12-3.13).

```
[108]: vec1 = [2 5];  
       vec2 = [8 1];  
       matrix1 = [2 5; 9 1];  
       matrix2 = [8 1; 0 3];  
  
[109]: vec1 + vec2  
[109]: 1×2 Matrix{Int64}:  
       10 6  
  
[110]: matrix1 + matrix2  
[110]: 2×2 Matrix{Int64}:  
       10 6  
       9 4  
  
[111]: vec1 - vec2  
[111]: 1×2 Matrix{Int64}:  
       -6 4  
  
[112]: matrix1 - matrix2  
[112]: 2×2 Matrix{Int64}:  
       -6 4  
       9 -2  
  
[113]: matrix1 * matrix2  
[113]: 2×2 Matrix{Int64}:  
       16 17  
       72 12  
  
[116]: vec1 * matrix1  
[116]: 1×2 Matrix{Int64}:  
       49 15
```

Рис. 3.12: Примеры операций над матрицами

```
[117]: matrix1''  
[117]: 2×2 Matrix{Int64}:  
       2 5  
       9 1  
  
[118]: vec1''  
[118]: 1×2 Matrix{Int64}:  
       2 5  
  
[119]: matrix1 * 2  
[119]: 2×2 Matrix{Int64}:  
       4 10  
       18 2  
  
[121]: vec1 * 2  
[121]: 1×2 Matrix{Int64}:  
       4 10  
  
[123]: using LinearAlgebra  
[124]: dot(vec1, vec2)  
[124]: 21  
  
[128]: cross([0;1;0], [0;0;1])  
[128]: 3-element Vector{Int64}:  
       1  
       0  
       0
```

Рис. 3.13: Примеры операций над матрицами

4 Выводы

В результате выполнения данной лабораторной работы я подготовила рабочее пространство и инструментарий для работы с языком программирования Julia, на простейших примерах познакомилась с основами синтаксиса Julia.