**Лабораторная работа № 1**

**Julia. Установка и настройка. Основные принципы**

**Абд эль хай мохамад**

**Содержание**

[***1 . Цель работы*** 2](#_Toc1)

[***2 . Выполнение лабораторной работы*** 2](#_Toc2)

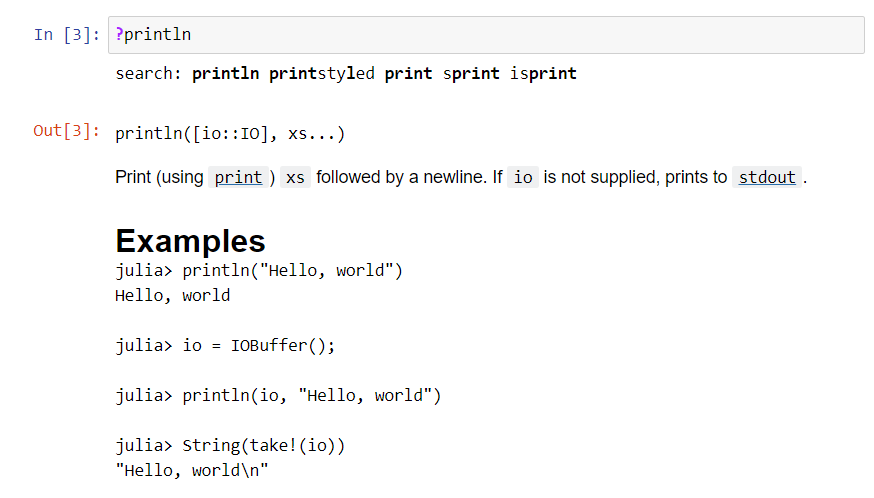
[***3. Вывод*** 12](#_Toc3)

# ***1 . Цель работы***

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

# ***2 . Выполнение лабораторной работы***

С помощью вопросительного знака получил информацию по работе функции println().



**Основы синтаксиса Julia на примерах.**

Далее я рассмотрю приведенные в задании примеры в своем ноутбуке.

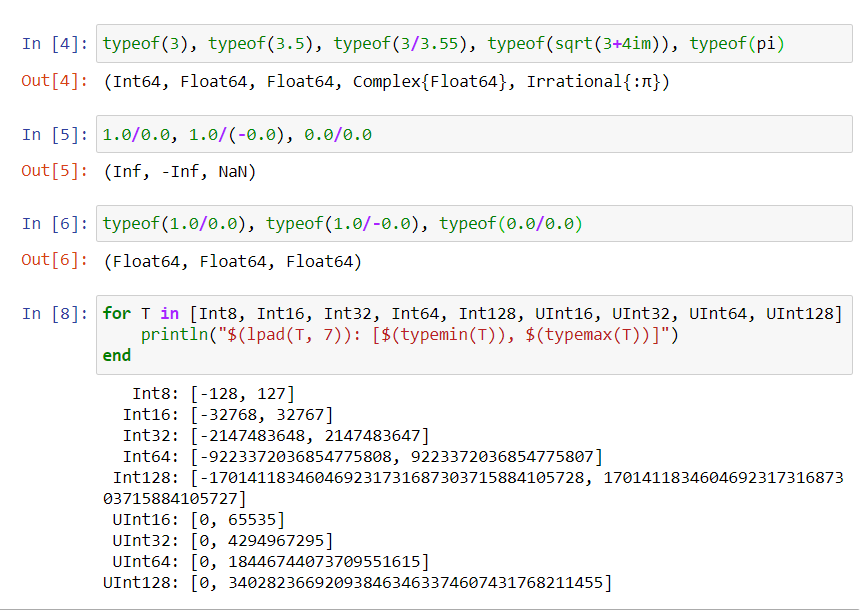


Рисунок 1. Примеры определения типа числовых величин

Ниже рассматриваются разные способы для преобразования типов данных на языке Julia.

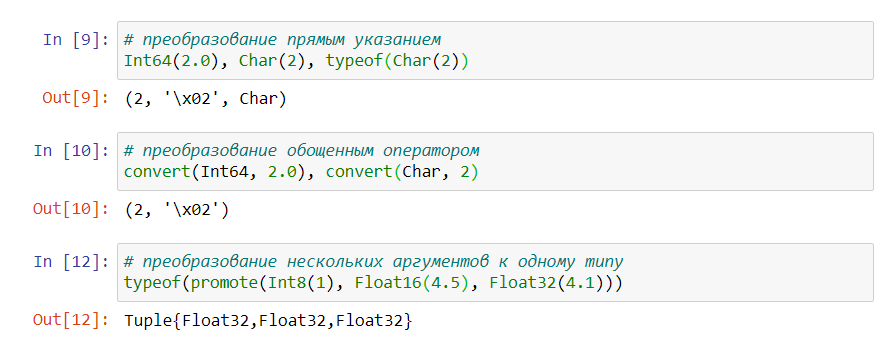


Рисунок 2. Примеры приведения аргументов к одному типу

Базовый синтаксис определения функции.

function <Имя> (<Список Параметров>)

<Действия>

End

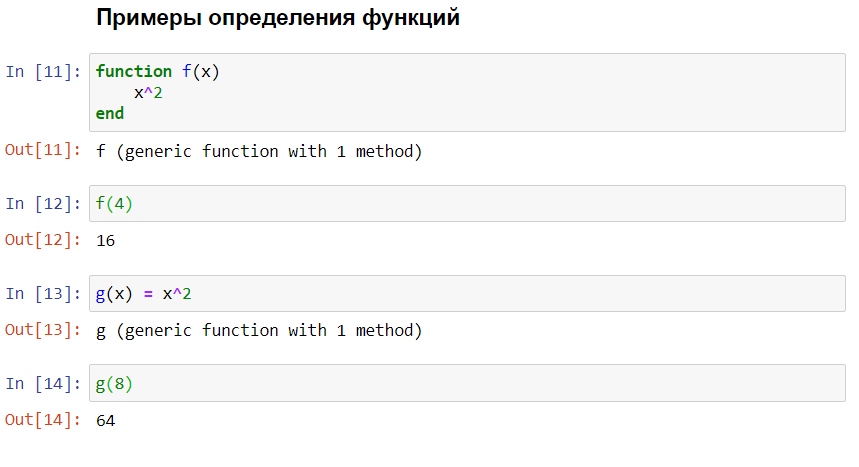


Рисунок 3. Пример определения функций.

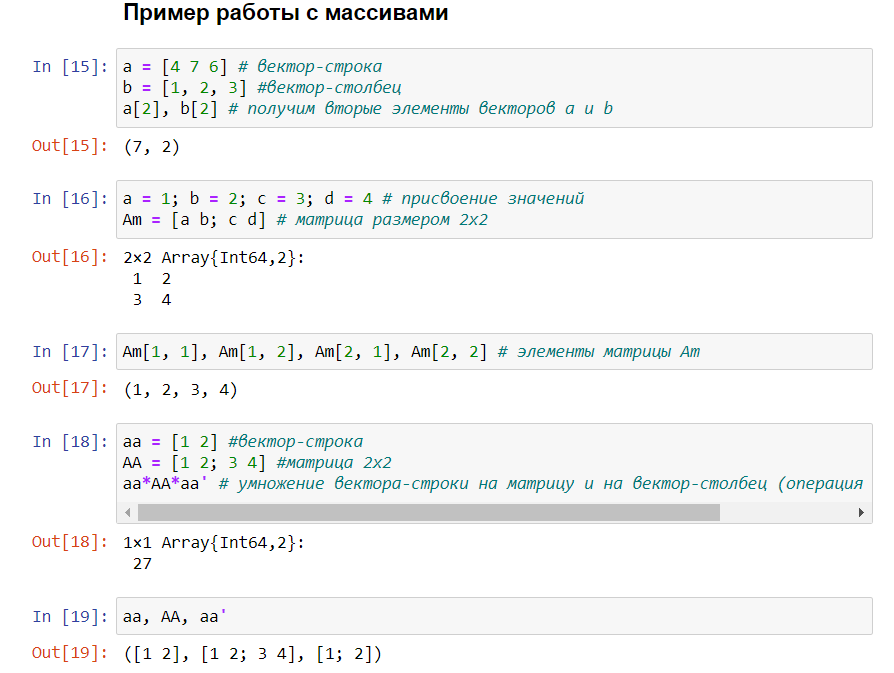


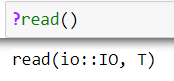
Рисунок 4. Пример работы с массивами

**Задания для самостоятельной работы.**

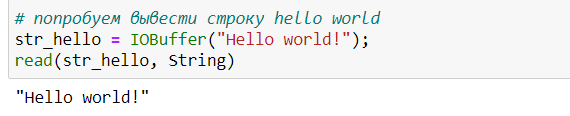
1. Изучите документацию по основным функциям Julia для чтения / записи / вывода информации на экран: read(), readline(), readlines(), readdlm(), print(), println(), show(), write(). Приведите свои примеры их использования, поясняя особенности их применения.

Для каждой функции в первой ячейке я вывожу информацию о ней (с помощью вопросительного знака). А во второй привожу пример использования.

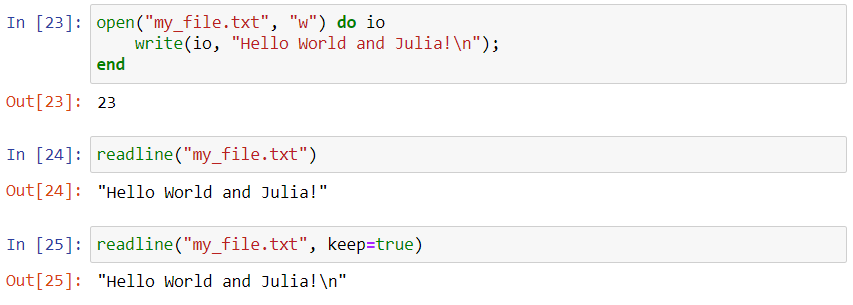
Начнем с функции read(). Данная функция считывает значение типа T из io, в общепринятом двоичном представлении.



Пример: Выведем строку “Hello world”.



Функция readline(). Считывание **одной строки** текста из данного потока ввода-вывода или файла. Строки на входе в файле заканчиваются буквой "\n " или "\r\n " или концом входного потока. Если keep имеет значение false (по умолчанию), эти конечные символы новой строки удаляются из строки до ее возврата. Когда keep имеет значение true, они возвращаются как часть строки.



Как мы видим в начале я открыл текстовый файл на запись и поместил в него строку “Hello world and Julia!” и в конце указал ‘\n’. Функция write выводит нам на количество байтов, записанных в поток.. Далее с помощью функции readline() я попробовал 2 способа вывода записи, без параметра keep и с ним. В итоге я получил следующее:

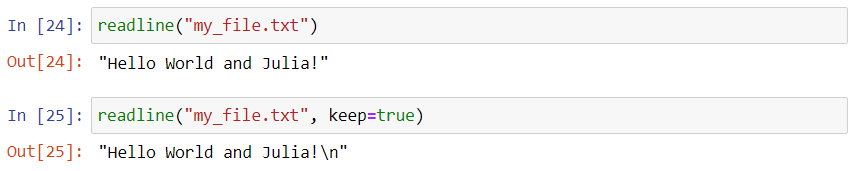


Рисунок 5. Пример использования readline()

Функция readlines(). Принципиальное отличие от предыдущей функции – считывание **всех строк** из файла или потока ввода-вывода. В данной функции также присутствует параметр keep и выполняет те же самые операции. Сохранение строк из файла происходит как сохранение вектора строк. Я воспользовалсья Lorem Ipsum для записи файла. Текст был взят с данного сайта: https://ru.lipsum.com/

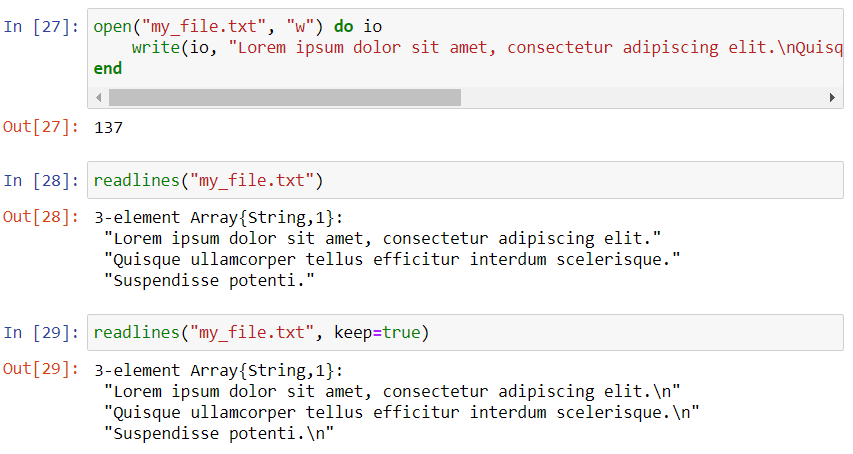


Рисунок 6. Пример использование readlines()

Функция readdlm(). Для того, чтобы воспользоваться данной функцией я воспользовалсь утилитой DelimitedFiles для чтения и записи файлов с разделителями. Предполагается, что столбцы разделены одним или несколькими пробелами. В качестве разделителя конца строки принимается \ n.

Также при вызове функции указывается тип данных в который преобразуется передаваемая строка. Ниже я создал файл с числами разделенными точкой с запятой, и далее я считал и преобразовал к Int32 и Float32.

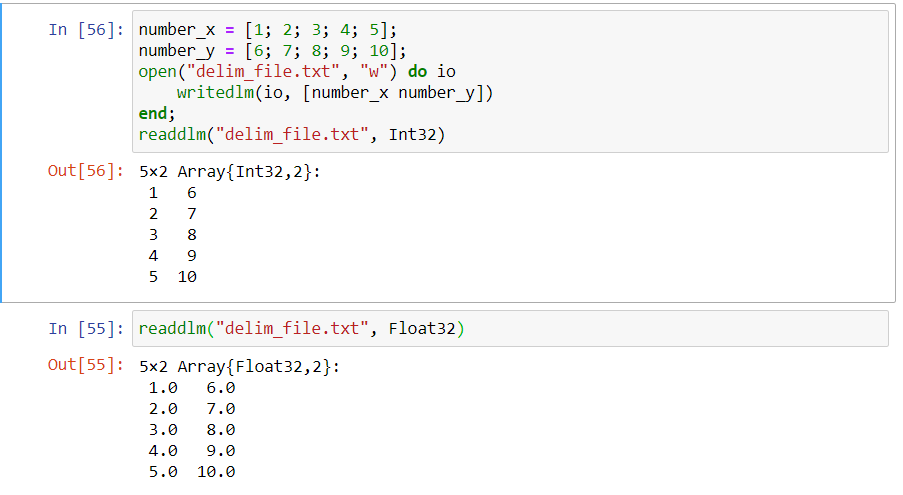


Рисунок 7. Пример использования readdlm()

Функция print(). Запись происходит в io (или в стандартный выходной поток stdout, если io не задан) общепринятое (недекорированное) текстовое представление. Представление, используемое print, включает минимальное форматирование и пытается избежать специфичных для Julia деталей.

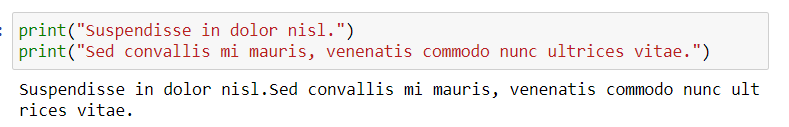


Рисунок 8. Пример использования print()

Функция println(). Эквивалентен функции print() только после вывода переходит на следующую строку

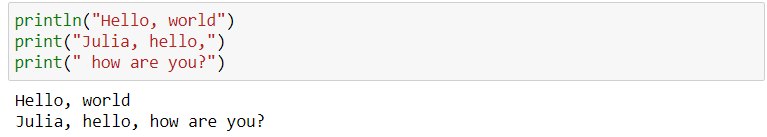


Рисунок 9. Пример использования println()

Функция show(). Записывает текстовое представление значения x в выходной поток ввода-вывода. Новые типы T должны перегружать функцию show (io:: IO, x:: T). Представление, используемое show, как правило, включает в себя специфичное для Julia форматирование и информацию о типе.

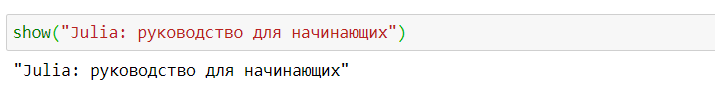
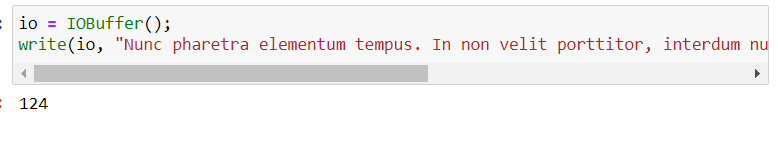


Рисунок 10. Пример использования show()

Функция write(). Записывает общепринятое двоичное представление значения в данный поток ввода-вывода или файл. Возвращает количество байтов, записанных в поток.



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

Парсит строку в число. Для целочисленных типов можно указать base (по умолчанию 10). Для типов с плавающей запятой строка анализируется как десятичное число с плавающей запятой. Комплексные числа разбираются из десятичных строк формы «R ± Iim» как Комплекс (R, I) запрошенного типа; «i» или «j» также могут использоваться вместо «im», также разрешены «R» или «Iim». Если строка не содержит допустимого числа, возникает ошибка.

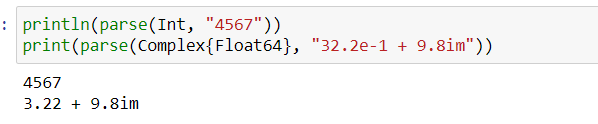


Рисунок 11. Пример использования parse()

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.

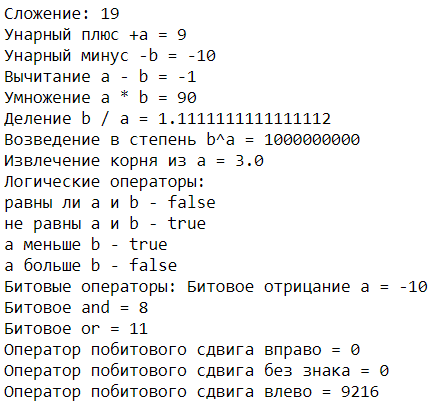
Информацию о языке Julia я нашел на данном ресурсе: http://ihed.ras.ru/~thermo/Julia/Brief%20description%20of%20Julia%20language.pdf

Объявим 2 переменные и запишем в них целые числа. Далее я сделал все перечисленные операции. К операциям сравнения можно еще добавить больше или равно (>=) и меньше или равно (<=).



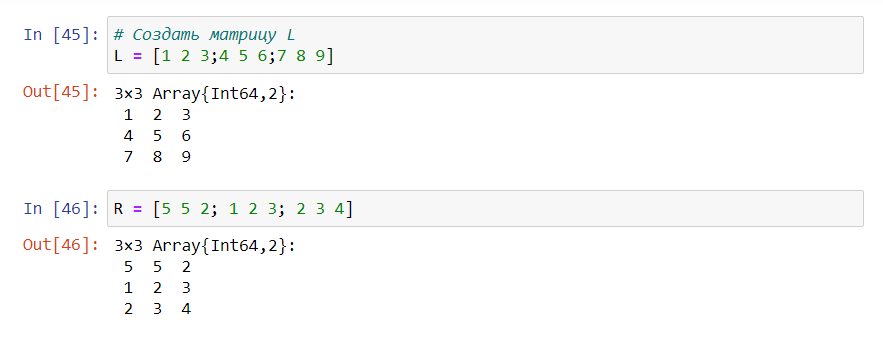
Рисунок 12. Примеры арифметических операций с данными

Вывод:



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

Создадим 2 матрицы одинакового размера 3x3 для удобства.



Далее идут привычные нам операции – сложение (+), вычитание (-), умножение (\*), и транспонирование (‘).



Рисунок 13. Основные операции с матицами

Теперь создадим скаляр. Допустим, что он равен 8.



Теперь умножим матрицу R на данный скаляр Q.



**Выводы**

Подготовил рабочее пространство и инструментарии для работы с языком программирования Julia, на простейших примерах познакомился с основами синтаксиса Julia.

# ***3. Вывод***