Лабораторная работа №3

Управляющие структуры

Клюкин Михаил Александрович

Содержание

# 1 Цель работы

Основная цель работы — освоить применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.

# 2 Задание

1. Используя Jupyter Lab, повторите примеры из раздела 3.2.
2. Выполните задания для самостоятельной работы (раздел 3.4).

# 3 Выполнение лабораторной работы

## 3.1 Циклы while и for

Для различных операций, связанных с перебором индексируемых элементов структур данных, традиционно используются циклы while и for.

Синтаксис while

while <условие>  
 <тело цикла>  
end

Пример использования цикла while для формирования элементов массива (рис. 1).

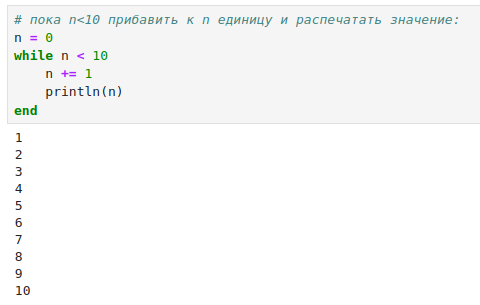


Рис. 1: Пример использования цикла while для формирования элементов массива

Другой пример демонстрирует использование while при работе со строковыми элементами массива, подставляя имя из массива в заданную строку приветствия и выводя получившуюся конструкцию на экран (рис. 2).

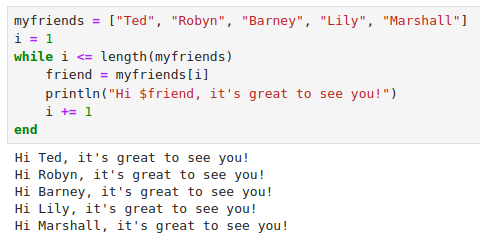


Рис. 2: Пример использования цикла while для работы со строковыми элементами массива

Такие же результаты можно получить при использовании цикла for.

Синтаксис for:

for <переменная> in <диапазон>  
 <тело цикла>  
end

Рассмотренные выше примеры, но с использованием цикла for (рис. 3, 4).

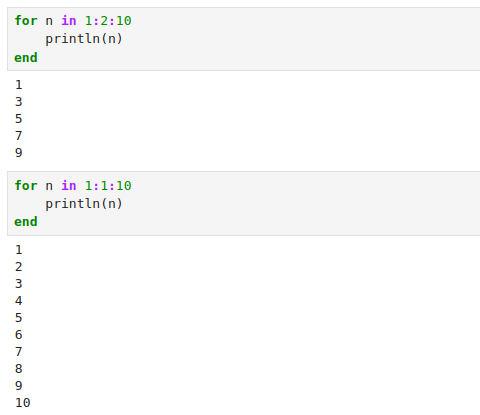


Рис. 3: Пример использования цикла for для формирования элементов массива

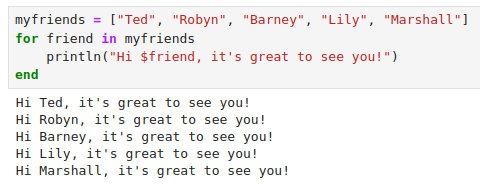


Рис. 4: Пример использования цикла for для работы со строковыми элементами массива

Пример использования цикла for для создания двумерного массива, в котором значе- ние каждой записи является суммой индексов строки и столбца (рис. 5).

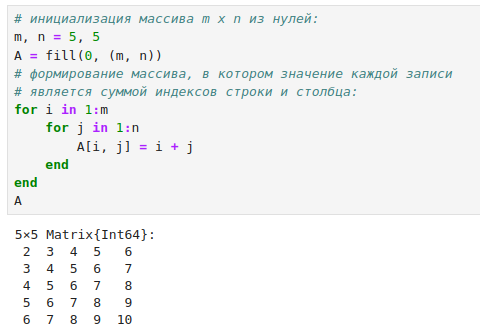


Рис. 5: Пример использования цикла for для создания двумерного массива

Создать двумерный массив также можно и другими способами (рис. 6, 7).

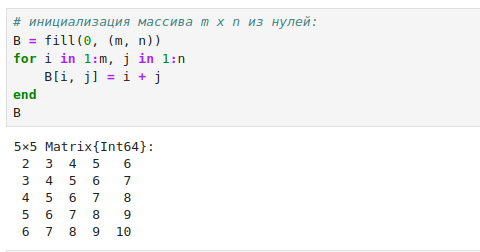


Рис. 6: Пример использования цикла for для создания двумерного массива

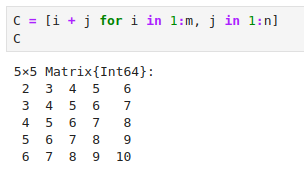


Рис. 7: Пример использования цикла for для создания двумерного массива

## 3.2 Условные выражения

Довольно часто при решении задач требуется проверить выполнение тех или иных условий. Для этого используют условные выражения.

Синтаксис условных выражений с ключевым словом:

if <условие 1>  
 <действие 1>  
elseif <условие 2>  
 <действие 2>  
else  
 <действие 3>  
end

Например, пусть для заданного числа N требуется вывести слово «Fizz», если N делится на 3, «Buzz», если N делится на 5, и «FizzBuzz», если N делится на 3 и 5 (рис. 8).

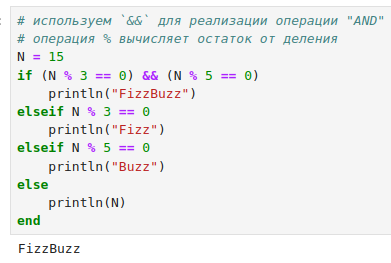


Рис. 8: Пример использования условного выражения

Синтаксис условных выражений с тернарными операторами:

a ? b : c

Такая запись эквивалентна записи условного выражения с ключевым словом:

if a  
 b  
else  
 c  
end

Пример использования тернарного оператора (рис. 9).

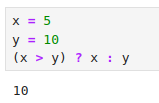


Рис. 9: Пример использования тернарного оператора

## 3.3 Функции

Julia дает нам несколько разных способов написать функцию. Первый требует ключевых слов function и end (рис. 10).

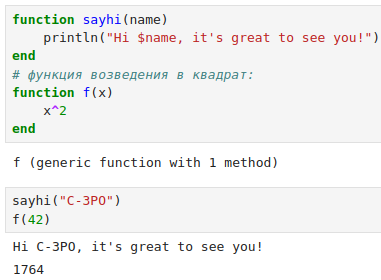


Рис. 10: Пример написания функций с помощью ключевых слов

В качестве альтернативы, можно объявить любую из выше определённых функций в одной строке либо как анонимную (рис. 11).

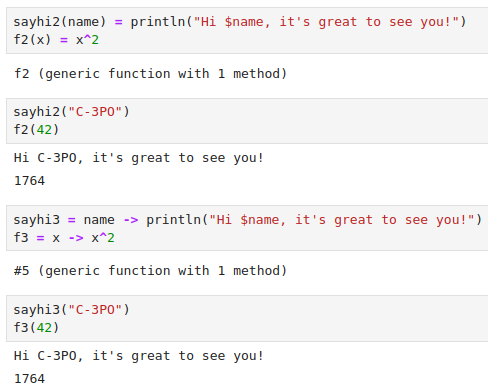


Рис. 11: Пример написания функций в одной строке

По соглашению в Julia функции, сопровождаемые восклицательным знаком, изменяют свое содержимое, а функции без восклицательного знака не делают этого. Например, сравните результат применения sort и sort! (рис. 12).

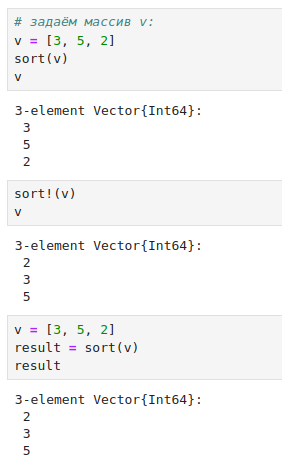


Рис. 12: Пример изменения аргумента функции

В Julia функция map является функцией высшего порядка, которая принимает функцию в качестве одного из своих входных аргументов и применяет эту функцию к каждому элементу структуры данных, которая ей передаётся также в качестве аргумента (рис. 13).

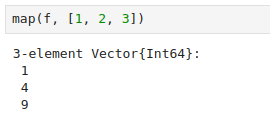


Рис. 13: Пример вызова функции map

Функция broadcast — ещё одна функция высшего порядка в Julia, представляющая собой обобщение функции map. Функция broadcast() будет пытаться привести все объекты к общему измерению, map() будет напрямую применять данную функцию поэлементно.

Синтаксис для вызова broadcast такой же, как и для вызова map (рис. 14).

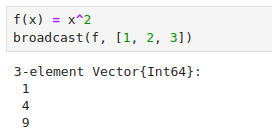


Рис. 14: Пример вызова функции broadcast

Возведение матрицы в квадрат (рис. 15).

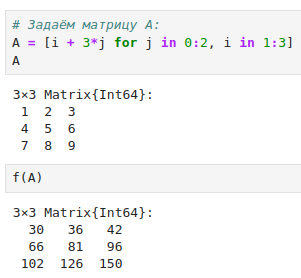


Рис. 15: Пример возведения матрицы в квадрат

Функция broadcast применяет переданную функцию ко всем элементам матрицы (рис. 16).

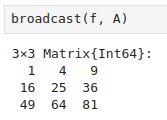


Рис. 16: Пример вызова функции broadcast

Точечный синтаксис для broadcast() позволяет записать относительно сложные со- ставные поэлементные выражения в форме, близкой к математической записи (рис. 17).

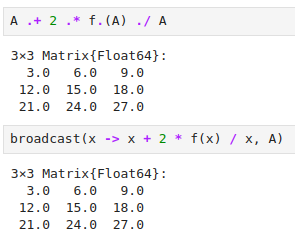


Рис. 17: Пример записи математического выражения через функцию bbroadcasr

## 3.4 Сторонние библиотеки в Julia

Julia имеет более 2000 зарегистрированных пакетов, что делает их огромной частью экосистемы Julia. Есть вызовы функций первого класса для других языков, обеспечи- вающие интерфейсы сторонних функций. Можно вызвать функции из Python или R, например, с помощью PyCall или Rcall.

При первом использовании пакета в вашей текущей установке Julia вам необходимо использовать менеджер пакетов, чтобы явно его добавить.

При каждом новом использовании Julia (например, в начале нового сеанса в REPL или открытии блокнота в первый раз) нужно загрузить пакет, используя ключевое слово using (рис. 18).

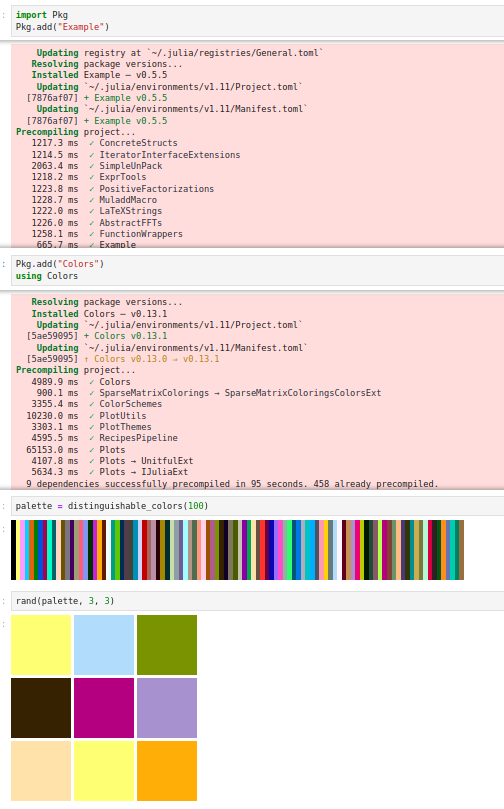


Рис. 18: Пример использования пакетов

# 4 Задания для самостоятельного выполнения

Используя циклы while и for, выведем на экран целые числа от 1 до 100 и напечатаем их квадраты (рис. 19, 20).

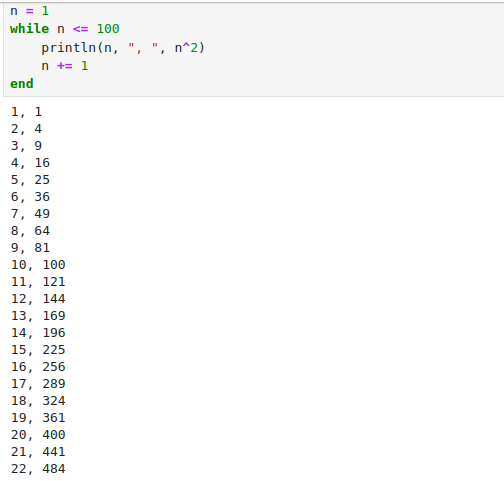


Рис. 19: Вывод чисел и их квадратов с помощью цикла while

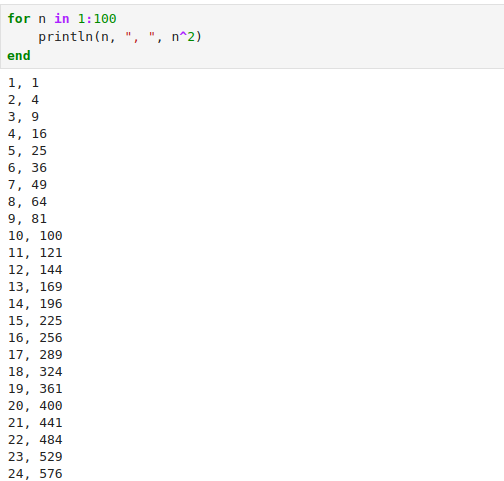


Рис. 20: Вывод чисел и их квадратов с помощью цикла for

Используя циклы while и for, создадим словарь squares, который будет содержать целые числа в качестве ключей и квадраты в качестве их пар-значений (рис. 21).

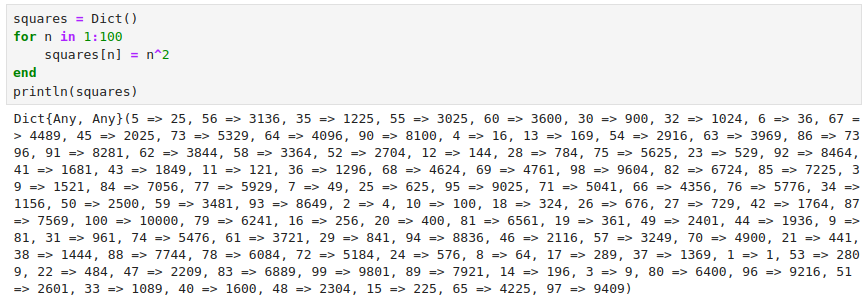


Рис. 21: Создание словаря

Используя циклы while и for, создадим массив squares\_arr, содержащий квадраты всех чисел от 1 до 100 (рис. 22).

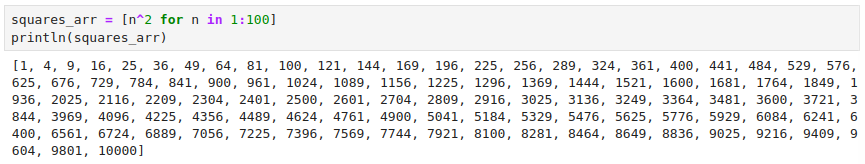


Рис. 22: Создание массива

Напишем условный оператор, который печатает число, если число чётное, и строку «нечётное», если число нечётное (рис. 23).

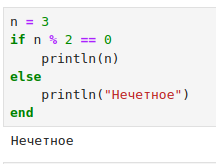


Рис. 23: Пример условного оператора

Перепишем код, используя тернарный оператор (рис. 24).

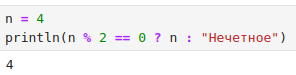


Рис. 24: Пример тернарного оператора

Напишем функцию add\_one, которая добавляет 1 к своему входу (рис. 25).

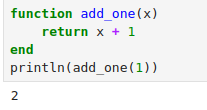


Рис. 25: Пример функции

Используем map() для задания матрицы A, каждый элемент которой увеличивается на единицу по сравнению с предыдущим (рис. 26).

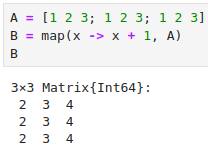


Рис. 26: Пример использования map

Зададим матрицу A и найдем ее куб (рис. 27).

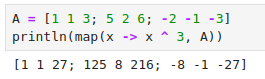


Рис. 27: Пример возведения матрицы в степень

Заменим третий столбец матрицы A на сумму второго и третьего столбцов (рис. 28).

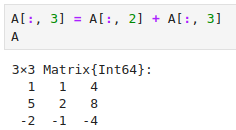


Рис. 28: Пример замены столбца в матрице

Создадим матрицу B с элементами (рис. 29).

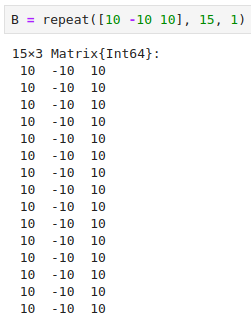


Рис. 29: Создание матрицы B

Вычислим матрицу (рис. 30).

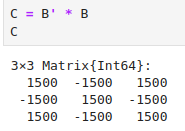


Рис. 30: Вычисление матрицы

Создайте матрицу Z размерности , все элементы которой равны нулю, и матрицу E, все элементы которой равны 1. Используя цикл for, создадим матрицу (рис. 31).

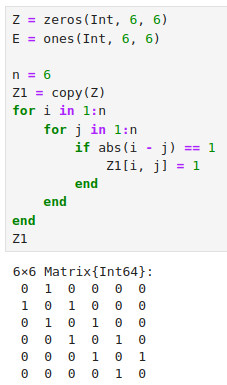


Рис. 31: Создание матриц Z, E, Z1

Используя цикл for, создадим матрицу (рис. 32).

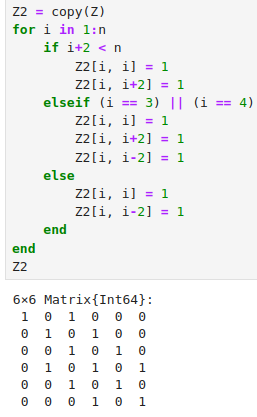


Рис. 32: Создание матриц Z2

Используя цикл for, создадим матрицу (рис. 33).

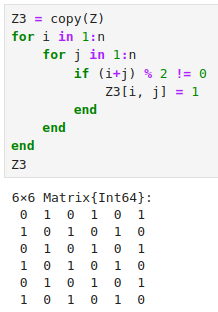


Рис. 33: Создание матриц Z3

Используя цикл for, создадим матрицу (рис. 34).

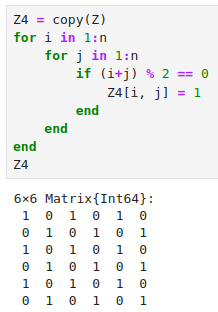


Рис. 34: Создание матриц Z4

Напишем свою функцию, аналогичную функции outer() языка R. Функция должна иметь следующий интерфейс: outer(x,y,operation) (рис. 35).

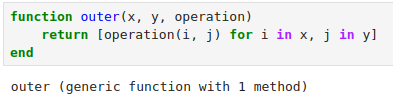


Рис. 35: Создание функции outer

Используя написанную функцию outer(), создадим матрицу (рис. 36).

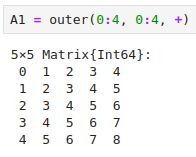


Рис. 36: Создание матрицы

Используя написанную функцию outer(), создадим матрицу (рис. 37).

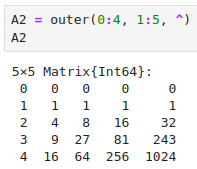


Рис. 37: Создание матрицы

Используя написанную функцию outer(), создадим матрицу (рис. 38).

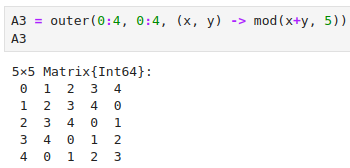


Рис. 38: Создание матрицы

Используя написанную функцию outer(), создадим матрицу (рис. 39).

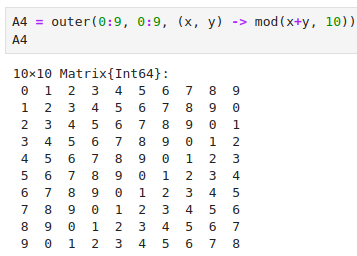


Рис. 39: Создание матрицы

Используя написанную функцию outer(), создадим матрицу (рис. 40).

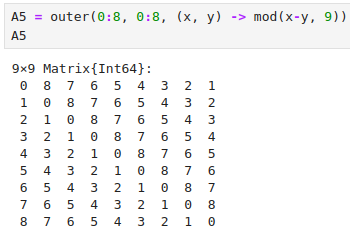


Рис. 40: Создание матрицы

Решим следующую систему линейных уравнений с 5 неизвестными (рис. 41).

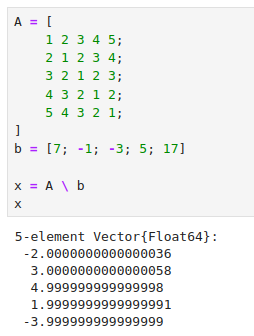


Рис. 41: Решение системы линейных уравнений

Создайте матрицу M размерности , элементами которой являются целые числа выбранные случайным образом с повторениями из совокупности (рис. 42).

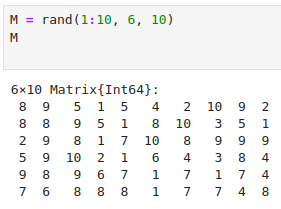


Рис. 42: Создание матрицы M

Найдите число элементов в каждой строке матрицы , которые больше числа (например, = 4) (рис. 43).

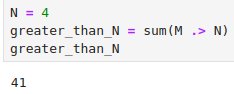


Рис. 43: Поиск числа элементов в каждой строке матрицы

Определим, в каких строках матрицы число M (например, M = 7) встречается ровно 2 раза (рис. 44).

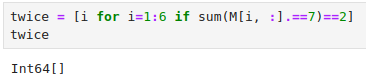


Рис. 44: Поиск строк с двумя числами M

Определим все пары столбцов матрицы , сумма элементов которых больше (например, = 75) (рис. 45).

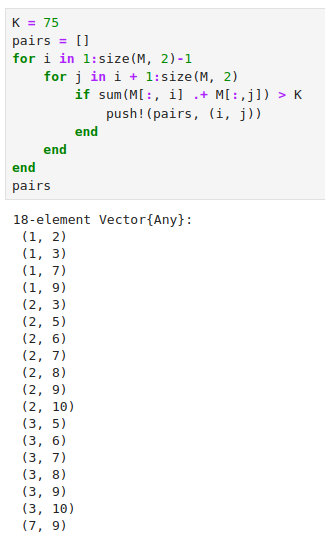


Рис. 45: Поиск столбцов матрицы

Вычислим выражения (рис. 46).

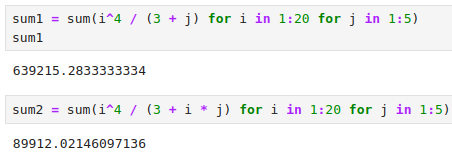


Рис. 46: Вычиление выражений

# 5 Выводы

В результате выполнения данной лабораторной работы мы освоили применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.