**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3**

*дисциплина: Компьютерный практикум по математическому моделированию*

Студент: Абрамян Артём

Группа: НПИбд-01-20

**МОСКВА**

2023 г.

**Постановка задачи**

Освоить применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.

**Выполнение работы**

1. **Повторите примеры из раздела 3.2**
   1. **Циклы while и for**

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

<body>

end

Примеры:



Такие же результаты можно получить при использовании цикла for.

Синтаксис for

for <переменная> in <диапазон>

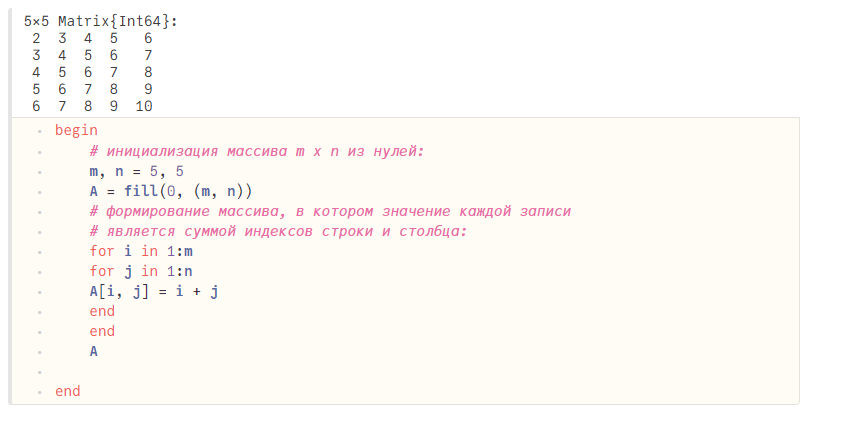
<тело цикла>

end

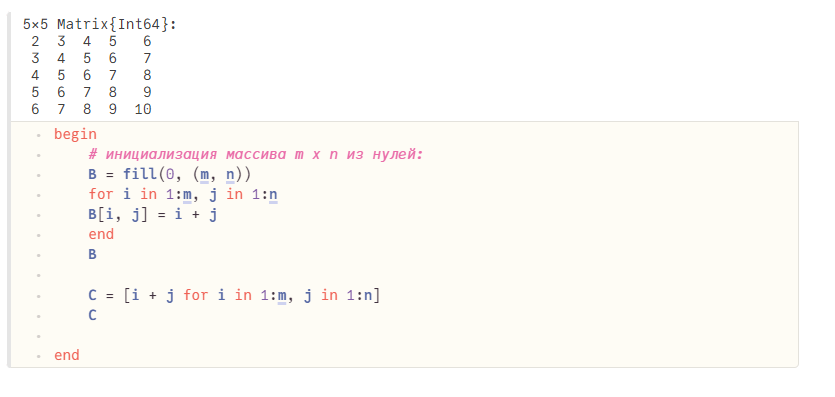
Рассмотренные выше примеры, но с использованием цикла for:



Пример использования цикла for для создания двумерного массива, в котором значение каждой записи является суммой индексов строки и столбца:



Другая реализация этого же примера:



* 1. **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>

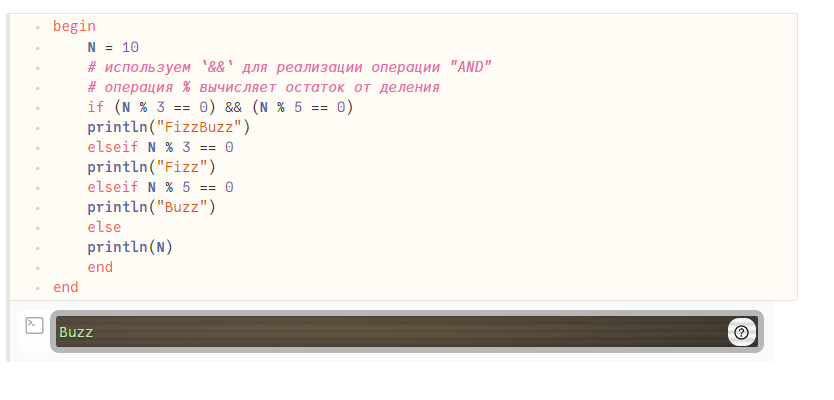
elseif <условие 2>

<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:

Синтаксис условных выражений с тернарными операторами:

a ? b : c

(если выполнено a, то выполнить b, если нет, то c).

Такая запись эквивалентна записи условного выражения с ключевым словом:

if a

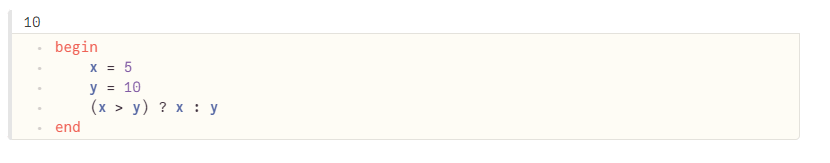
b

else

c

end

Пример использования тернарного оператора:



* 1. **Функции**

Julia дает нам несколько разных способов написать функцию. Первый требует ключевых слов function и end:

function sayhi(name)

println("Hi $name, it's great to see you!")

end

# функция возведения в квадрат:

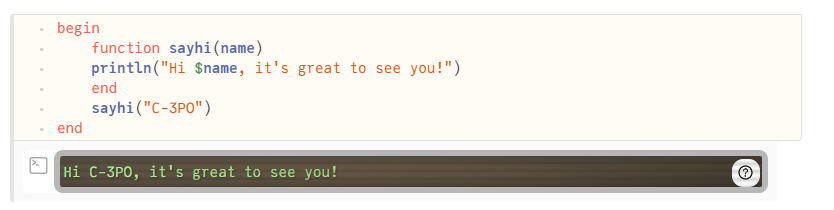
function f(x)

x^2

end

Вызов функции осуществляется по её имени с указанием аргументов, например:

sayhi("C-3PO")



f(42)

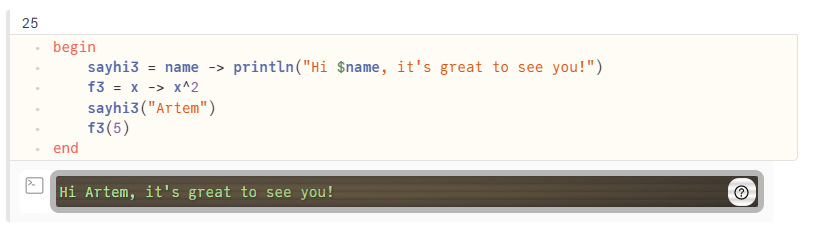
В качестве альтернативы, можно объявить любую из выше определённых функций

в одной строке:

sayhi2(name) = println("Hi $name, it's great to see you!")

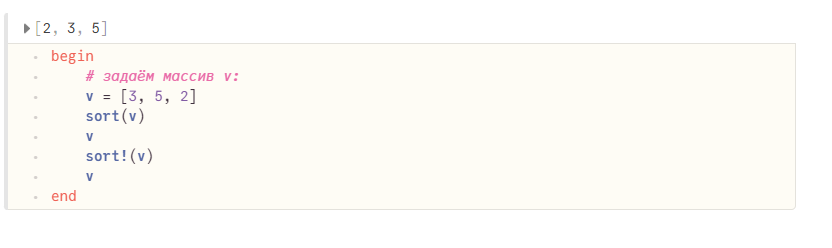
f2(x) = x^2

Наконец, можно объявить выше определённые функции как «анонимные»:



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!:



Функция sort(v) возвращает отсортированный массив, который содержит те же элементы, что и массив v, но исходный массив v остаётся без изменений. Если же использовать sort!(v), то отсортировано будет содержимое исходного массива v. В программировании под функцией высшего порядка понимается функция, принимающая в качестве аргументов другие функции или возвращающая другую функцию в качестве результата. Основная идея состоит в том, что функции имеют тот же статус, что и другие объекты данных. 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. Например, в случае выполнения выражения:

map(f, [1, 2, 3])

на выходе получим массив, в котором функция f была применена ко всем элементам

массива [1, 2, 3]:

[f(1), f(2), f(3)]

Если 𝑓(𝑥) = 𝑥2

, то получим следующий результат:

f(x) = x^2

map(f, [1, 2, 3])

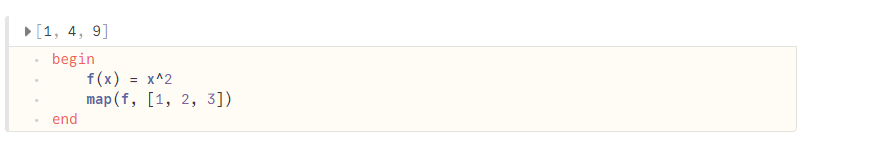
3-element Array{Int64,1}:

1

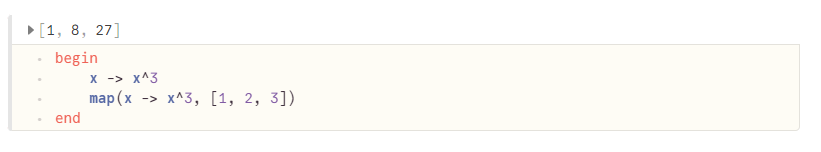
4

9

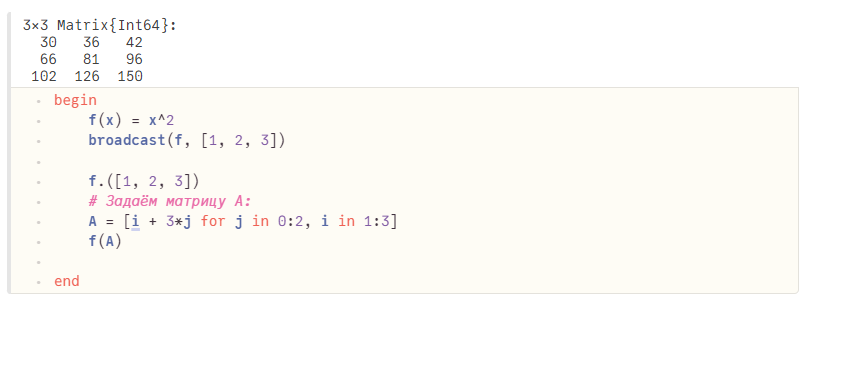
То есть в квадрат возведены все элементы массива [1, 2, 3], но не сам массив (вектор).

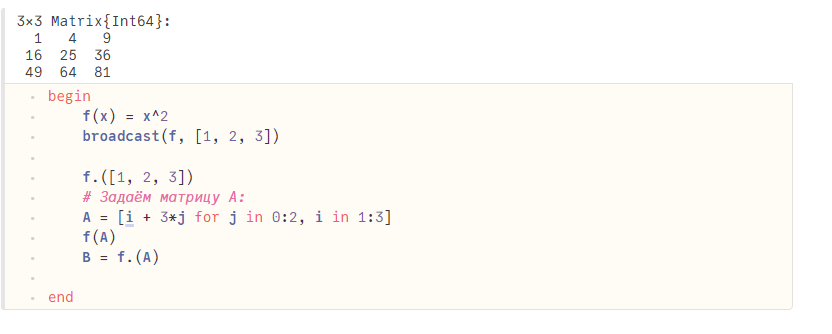


В map можно передать и анонимно заданную, а не именованную функцию:



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 вызоваmap, например применение функции f к элементам массива [1, 2, 3]:





Точечный синтаксис для broadcast() позволяет записать относительно сложные составные поэлементные выражения в форме, близкой к математической записи.

Например, запись

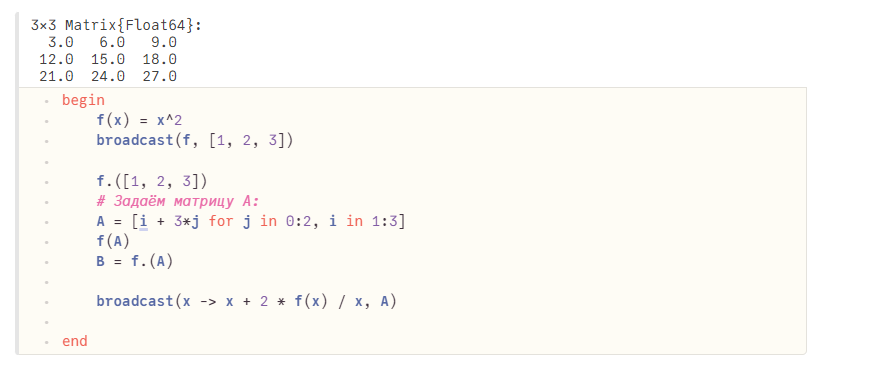
A .+ 2 .\* f.(A) ./ A

или запись

@. A + 2 \* f(A) / A

аналогичны записи

broadcast(x -> x + 2 \* f(x) / x, A)

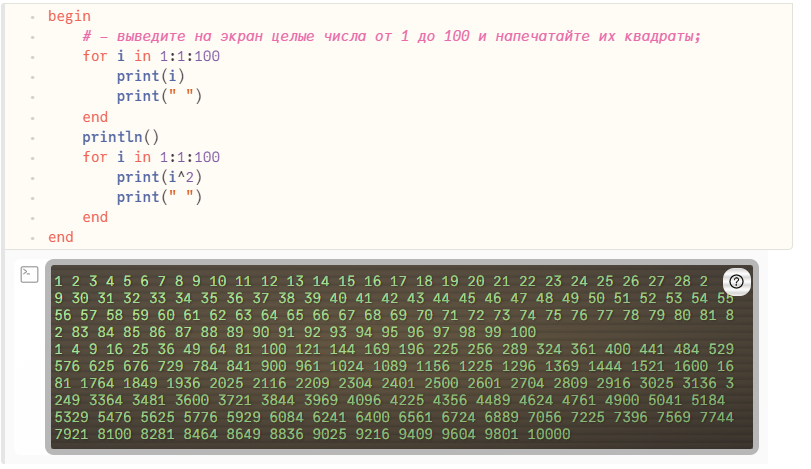


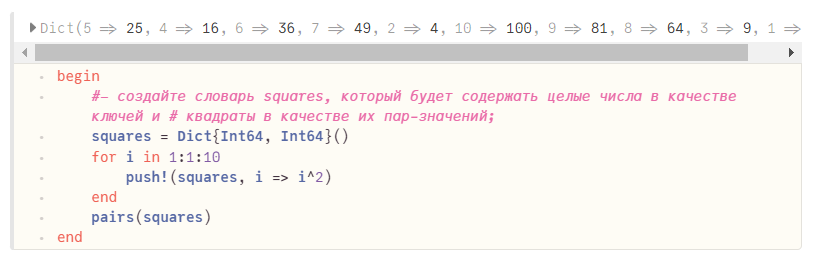
* 1. **Сторонние библиотеки (пакеты) в Julia**

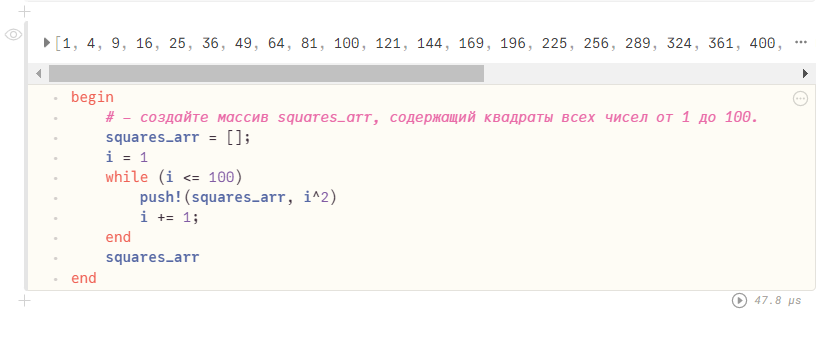
Julia имеет более 2000 зарегистрированных пакетов, что делает их огромной частью экосистемы Julia. Есть вызовы функций первого класса для других языков, обеспечивающие интерфейсы сторонних функций. Можно вызвать функции из Python или R, например, с помощью PyCall или Rcall. С перечнем доступных в Julia пакетов можно ознакомиться на страницах следующих ресурсов: – https://julialang.org/packages/ – https://juliahub.com/ui/Home – https://juliaobserver.com/ – https://github.com/svaksha/Julia.jl При первом использовании пакета в вашей текущей установке Julia вам необходимо использовать менеджер пакетов, чтобы явно его добавить: import Pkg Pkg.add("Example") 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: Например, добавим и загрузим пакет Colors:



1. **Задания для самостоятельной работы**
   1. **Используя циклы while и for:**



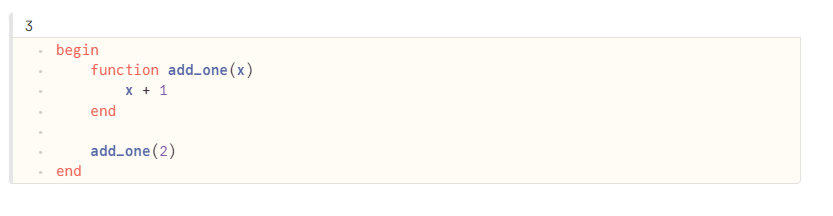




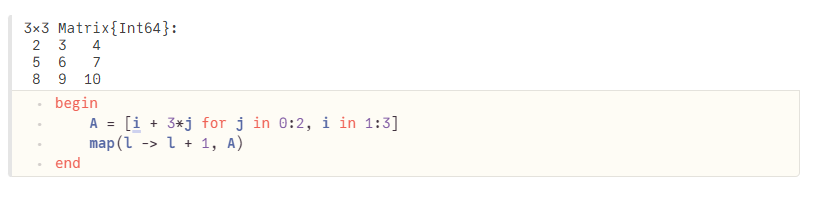
* 1. **Напишите условный оператор, который печатает число, если число чётное, и строку «нечётное», если число нечётное. Перепишите код, используя тернарный оператор.**



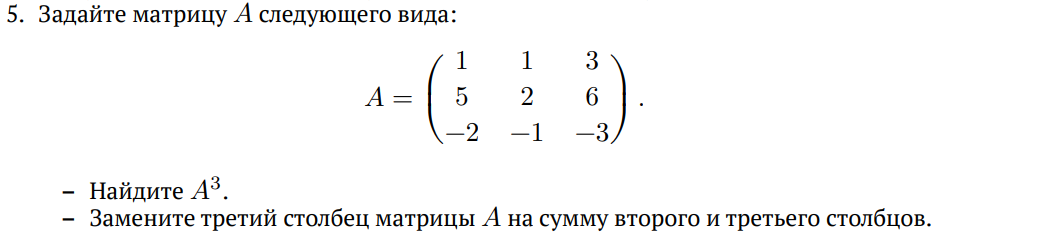
* 1. **Напишите функцию add\_one, которая добавляет 1 к своему входу.**

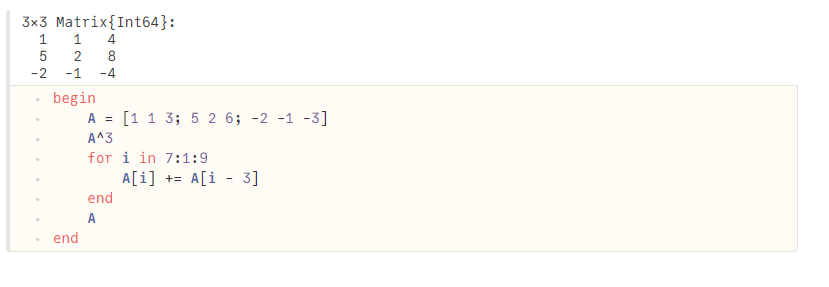


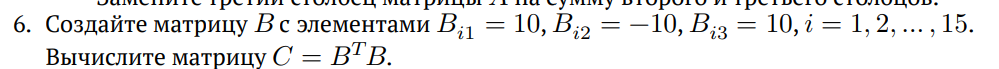
* 1. Используйте map() или broadcast() для задания матрицы 𝐴, каждый элемент которой увеличивается на единицу по сравнению с предыдущим.



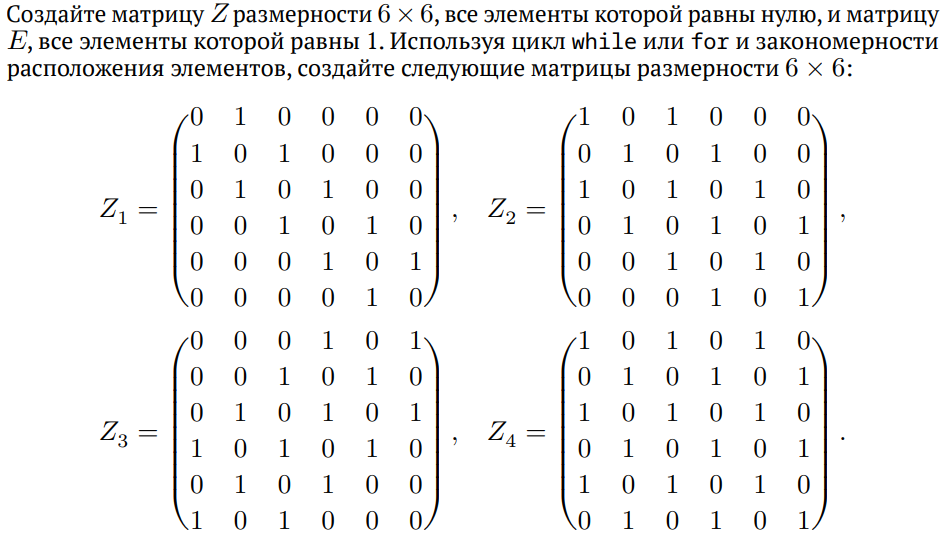
2.5

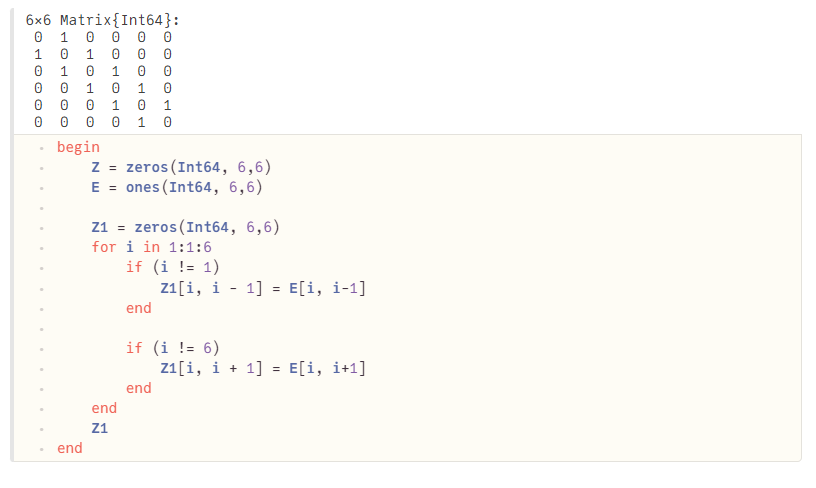


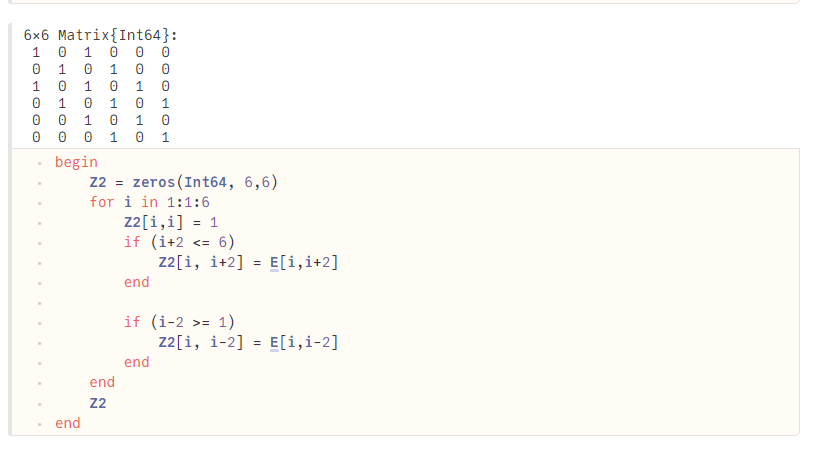


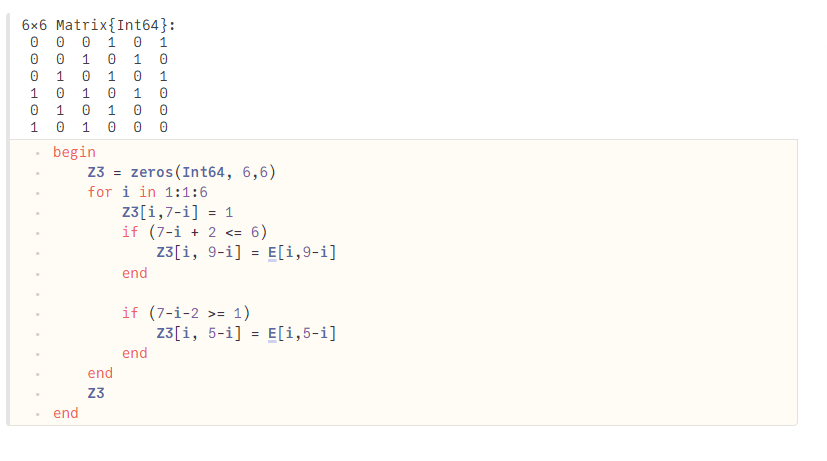
2.6 

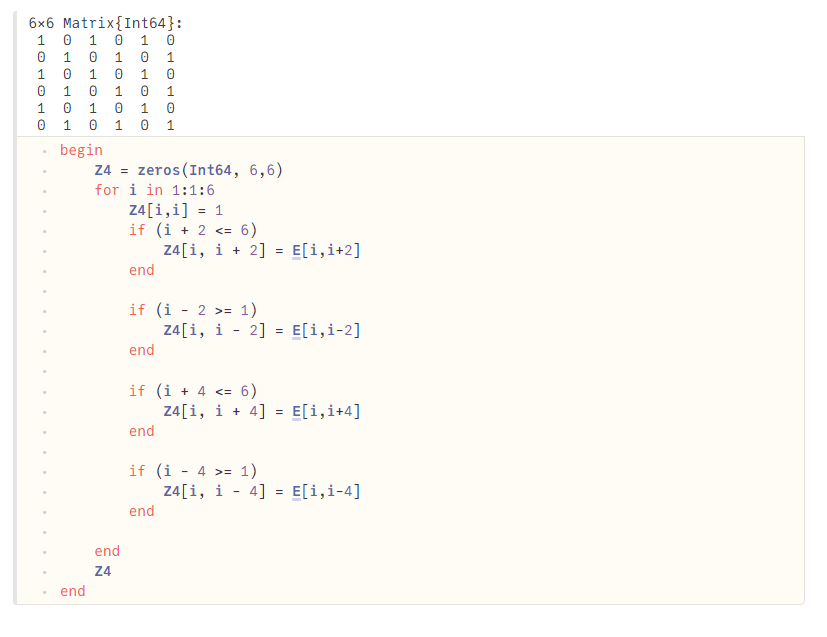


2.7 

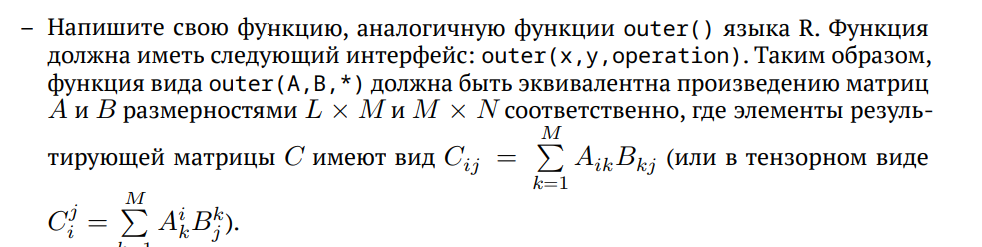


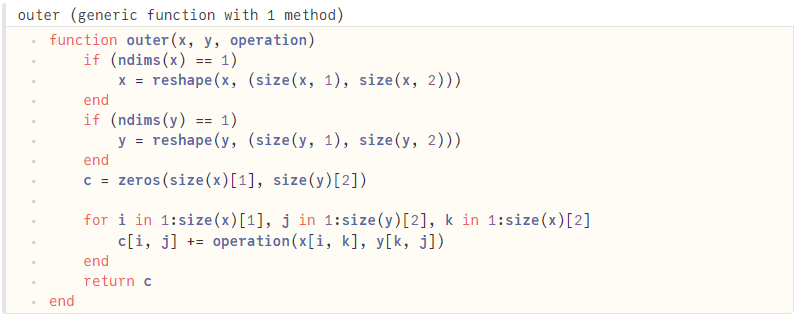


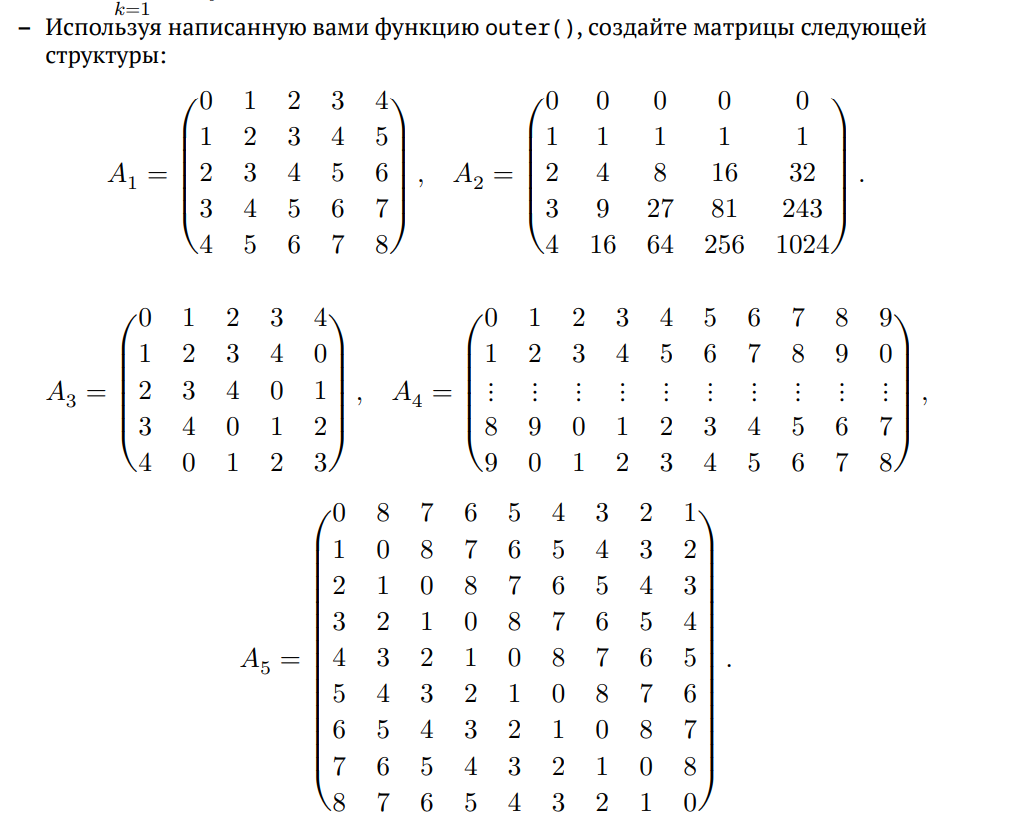


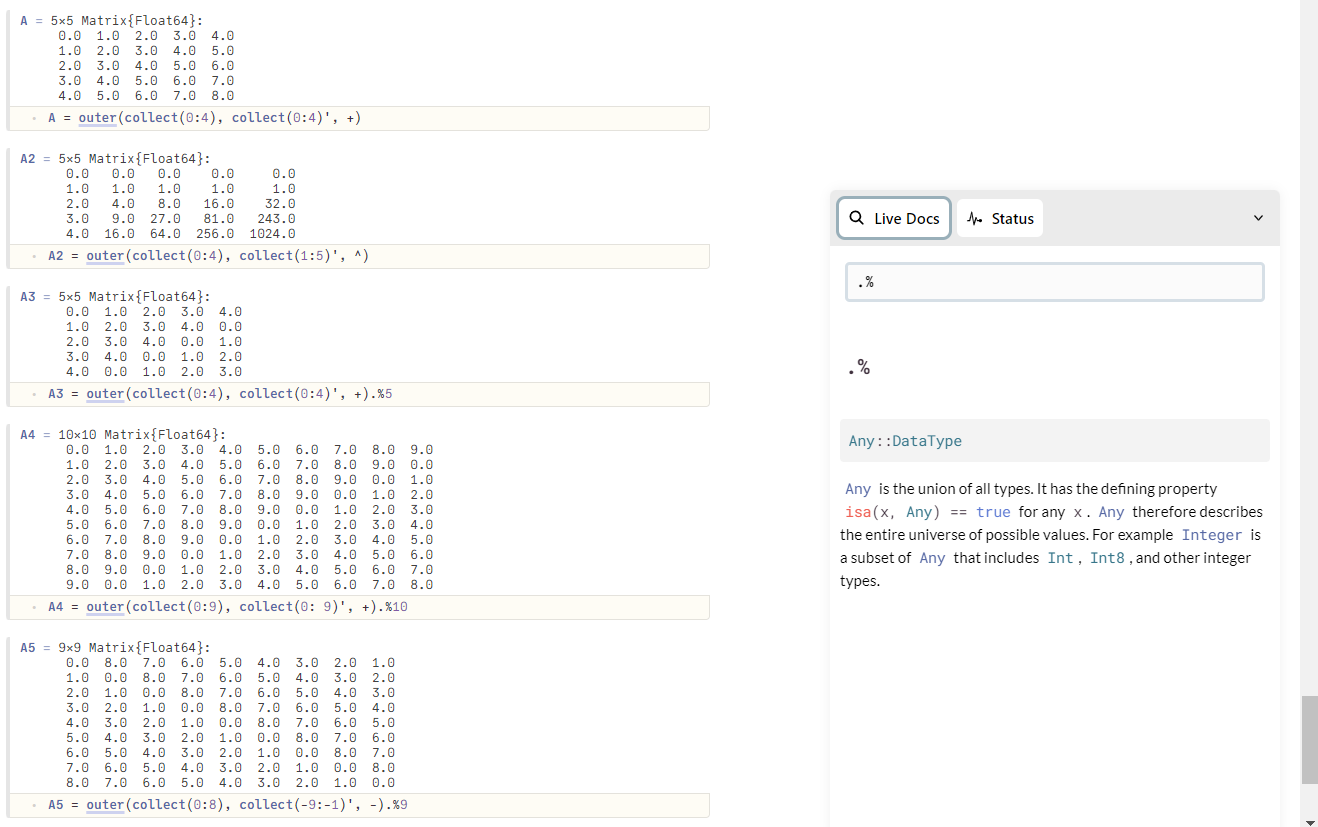


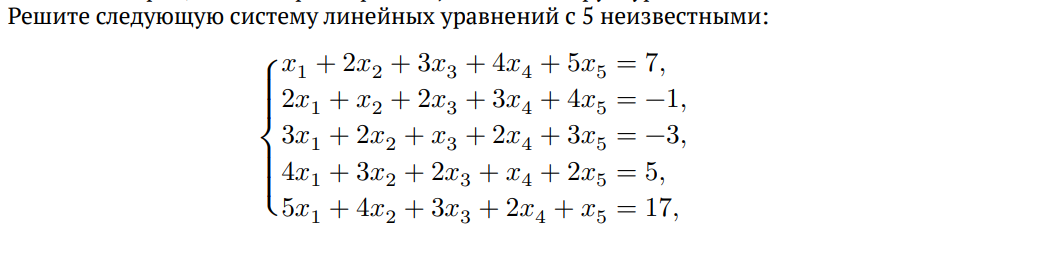
2.8 В языке R есть функция outer(). Фактически, это матричное умножение с возможностью изменить применяемую операцию (например, заменить произведение на сложение или возведение в степень).

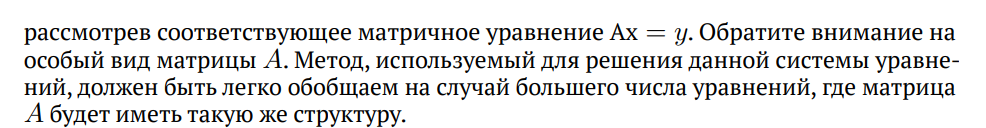


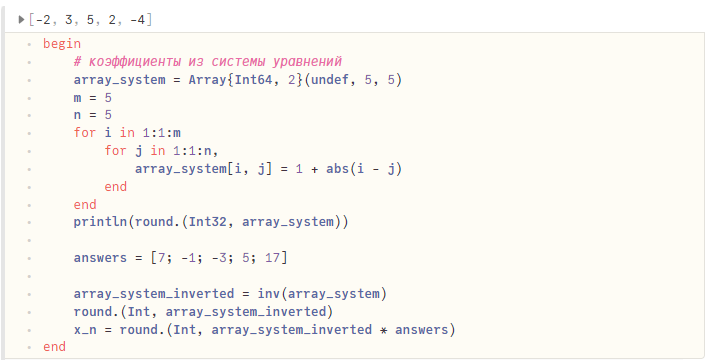


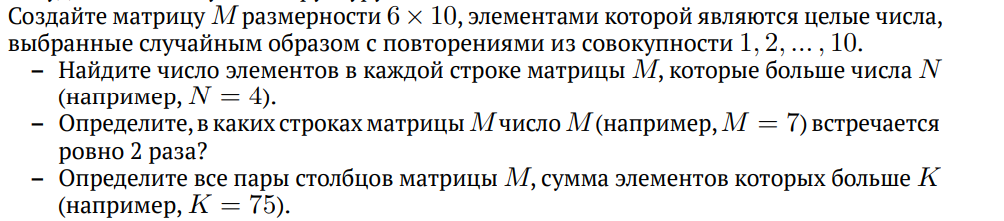


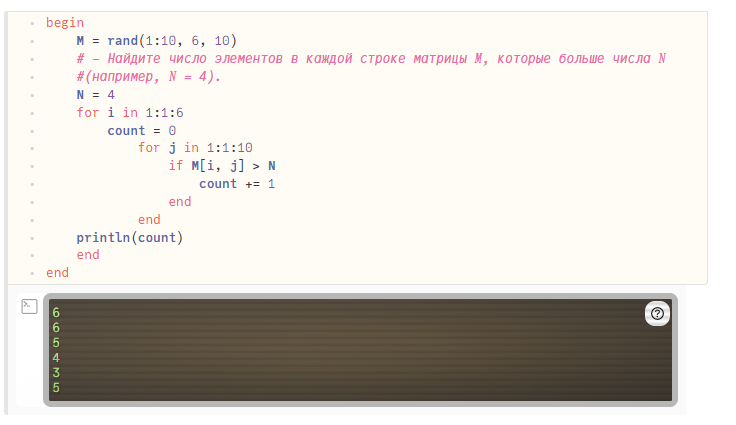


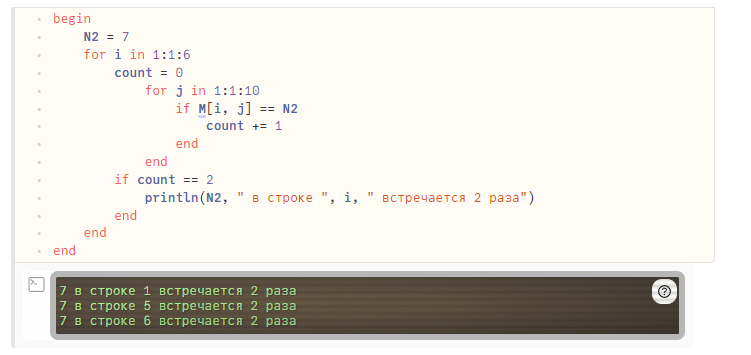
2.9 

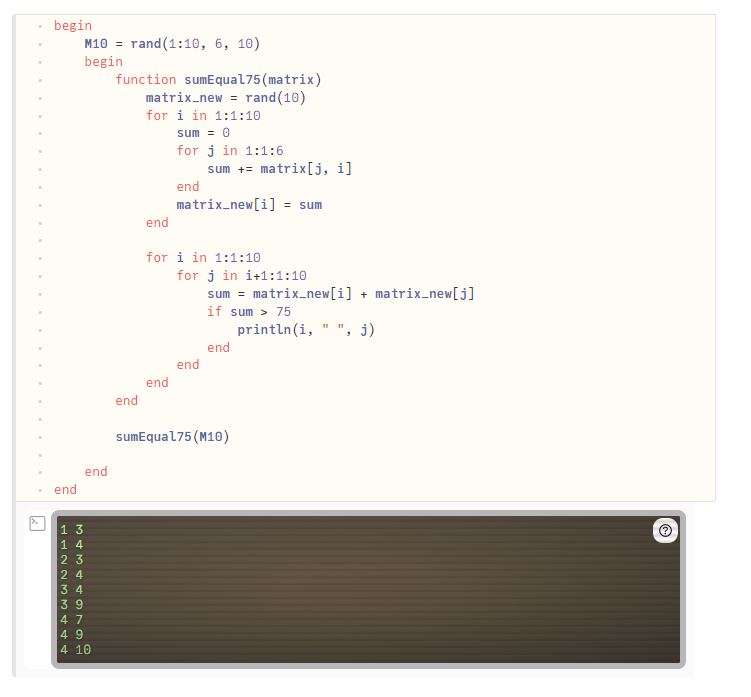




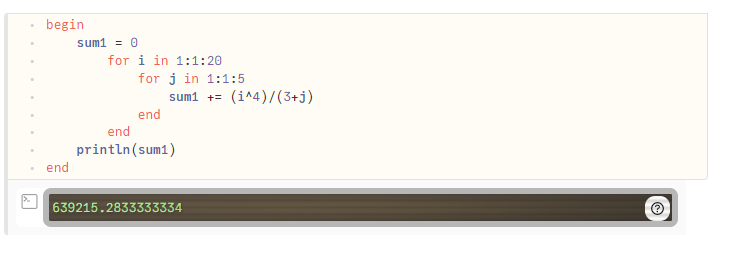
2.10 

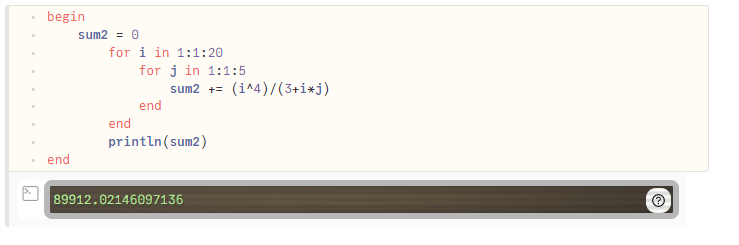






2.11 





**Выводы**

В ходе выполнения лабораторной работы успешно удалось освоить применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.