РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра теории вероятностей и кибербезопасности

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

дисциплина: Компьютерный практикум по статистическому анализу данных

Студент: Быстров Глеб

Группа: НПИбд-01-20

МОСКВА

2023 г.

Цель работы

В данной лабораторной работе мне будет необходимо освоить применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.

Описание процесса выполнения работы

3.3.1. Циклы while и for

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

   Синтаксис while

   while <условие>

   <тело цикла>

   end (рис. 3.1).

Рис. 3.1. Циклы while и for

3.3.2. Условные выражения

2. 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 (рис. 3.2).

```
# операция % вычисляет остаток от деления
N = 3

if (N % 3 == 0) && (N % 5 == 0)
    println("FizzBuzz")
elseif N % 3 == 0
    println("Fizz")
elseif N % 5 == 0
    println("Buzz")
else
    println(N)
end
```

```
Fizz
```

```
x = 5
y = 10
(x > y) ? x : y
```

```
10
```

Рис. 3.2. Условные выражения

3.3.3. Функции

3. 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) (рис. 3.3).

```
Ввод [24]: function sayhi(name)
               println("Hi $name, it's great to see you!")
           end
           sayhi("Gleb")

           Hi Gleb, it's great to see you!

Ввод [25]: function f(x)
               x^2
           end
           f(3)

Out[25]: 9

Ввод [28]: sayhi2(name) = println("Hi $name, it's great to see you!")
           sayhi2("Glebushka")

           Hi Glebushka, it's great to see you!

Ввод [29]: f2(x) = x^2
           f2(4)

Out[29]: 16
```

Рис. 3.3. Функции

3.3.4. Сторонние библиотеки (пакеты) в Julia

4. 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:

Pkg.add("Colors")

using Colors

Затем создадим палитру из 100 разных цветов:

palette = distinguishable_colors(100) (рис. 3.4)

Рис. 3.4. Работа с библиотекой Colors

5. Определим матрицу 3 × 3 с элементами в форме случайного цвета из палитры, используя функцию rand:

rand(palette, 3, 3) (рис. 3.5).

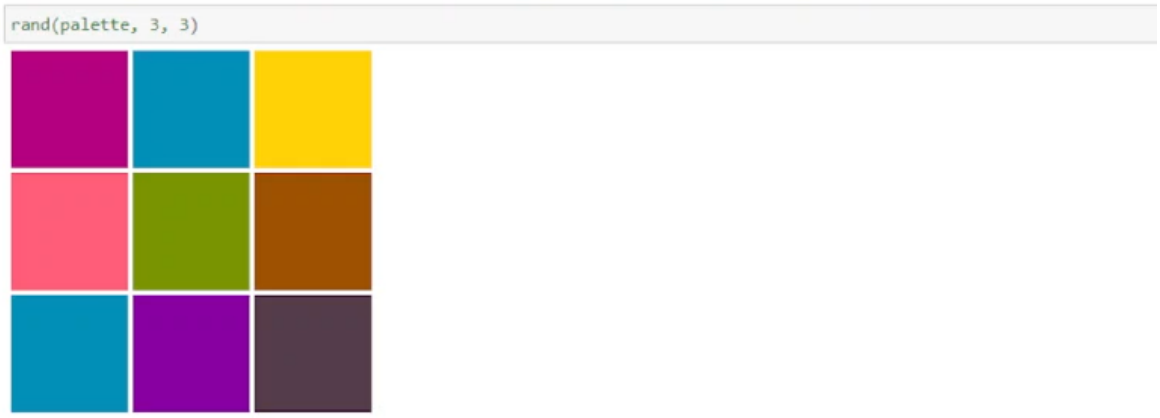

Рис. 3.5. Пример с матрицей и цветами

3.3.5. Задания для самостоятельного выполнения

6. Используя циклы while и for:

– выведите на экран целые числа от 1 до 100 и напечатайте их квадраты;

– создайте словарь squares, который будет содержать целые числа в качестве ключей и квадраты в качестве их пар-значений;

– создайте массив squares_arr, содержащий квадраты всех чисел от 1 до 100. (рис. 3.6-3.9).

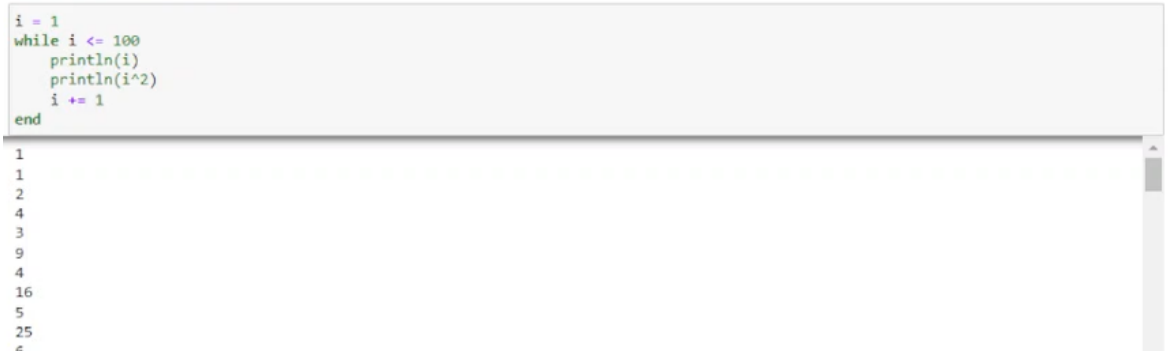

Рис. 3.6. Числа от 1 до 100 и их квадраты

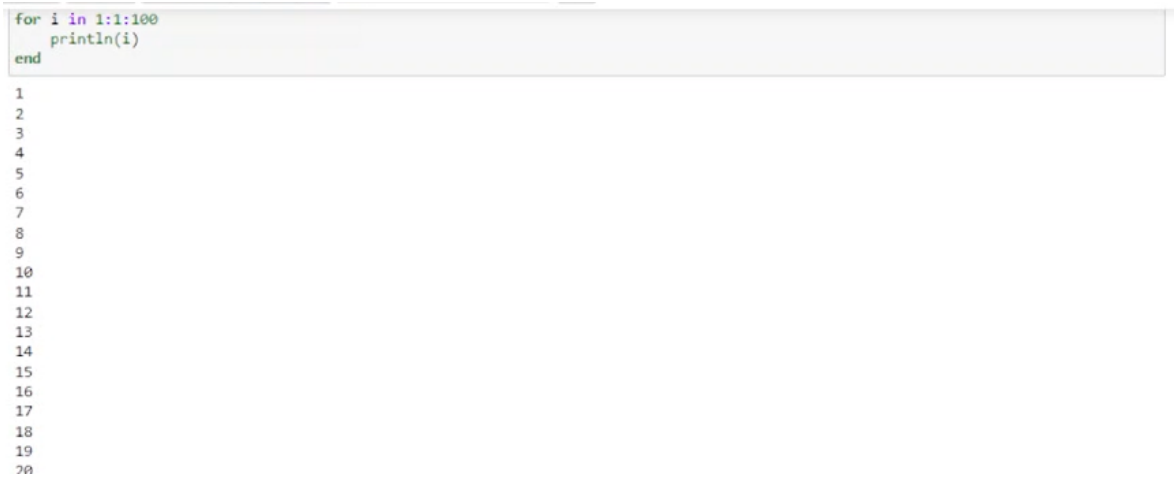

Рис. 3.7. Числа от 1 до 100

```
squares = Dict{Int64, Int64}()
for i in 1:1:100
    push!(squares, i => i^2)
end
pairs(squares)
```

```
Dict{Int64, Int64} with 100 entries:
  5  => 25
  56 => 3136
  35 => 1225
  55 => 3025
  60 => 3600
  30 => 900
  32 => 1024
  6  => 36
  67 => 4489
  45 => 2025
  73 => 5329
  64 => 4096
  90 => 8100
  4  => 16
  13 => 169
```

Рис. 3.8. Словарь squares

```
N = []
for i in 1:1:100
    append!(N, i)
end

squares_arr = []
i = 1
while i <= length(N)
    append!(squares_arr, N[i]^2)
    i += 1
end
squares_arr
```

```
100-element Vector{Any}:
     1
     4
     9
    16
    25
    36
    49
    64
    81
```

Рис. 3.9. Массив squares_arr

7. Напишите условный оператор, который печатает число, если число чётное, и строку «нечётное», если число нечётное. Перепишите код, используя тернарный оператор (рис. 3.10).

```
N = 7

if N % 2 == 0
    println("Четное")
else
    println("Нечетное")
end
```

```
Нечетное
```

```
N = 8
(N % 2 == 0) ? println("Четное") : println("Нечетное")
```

```
Четное
```

Рис. 3.10. Условный оператор

8. Напишите функцию add_one, которая добавляет 1 к своему входу (рис. 3.11).

```
function add_one(A)
    A + 1
end
add_one(2)
```

```
3
```

Рис. 3.11. Функция add_one

9. Используйте map() или broadcast() для задания матрицы A, каждый элемент которой увеличивается на единицу по сравнению с предыдущим (рис. 3.12).

```
broadcast(x -> x + 1, A)
```

```
3×3 Matrix{Int64}:
 2  3   4
 5  6   7
 8  9  10
```

Рис. 3.12. Использование broadcast()

10. Задайте матрицу A следующего вида:

$$A = \begin{pmatrix} 1 & 1 & 3 \\ 5 & 2 & 6 \\ -2 & -1 & -3 \end{pmatrix}$$

Найдите $A3$.

Замените третий столбец матрицы A на сумму второго и третьего столбцов (рис. 3.13)

```
A = [1 1 3; 5 2 6; -2 -1 -3]
```

```
3x3 Matrix{Int64}:
  1   1   3
  5   2   6
 -2  -1  -3
```

```
A^3
```

```
3x3 Matrix{Int64}:
 0  0  0
 0  0  0
 0  0  0
```

```
for i in 7:1:9
    A[i] += A[i-3]
end
A
```

```
3x3 Matrix{Int64}:
  1   1   5
  5   2  10
 -2  -1  -5
```

Рис. 3.13. Работа с матрицей A

11. Создайте матрицу B с элементами $B_{i1} = 10$, $B_{i2} = -10$, $B_{i3} = 10$, $i = 1, 2, \dots, 15$. Вычислите матрицу $C = B^T B$.

```
B = Array{Int32, 2}(undef, 15, 3)
for i in 1:1:15
    B[i, 1] = 10
    B[i, 2] = -10
    B[i, 3] = 10
end
B
```

```
15x3 Matrix{Int32}:
 10  -10  10
 10  -10  10
 10  -10  10
```

Рис. 3.14. Создание матрицы B

12. Создайте матрицу Z размерности 6 × 6, все элементы которой равны нулю, и матрицу E, все элементы которой равны 1. Используя цикл while или for и закономерности расположения элементов, создайте следующие матрицы размерности 6 × 6 (рис. 3.15-3.20):

$$Z_1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad Z_2 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix},$$

$$Z_3 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \quad Z_4 = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

```
Z = zeros(Int64, 6, 6)
```

```
6×6 Matrix{Int64}:
 0  0  0  0  0  0
 0  0  0  0  0  0
 0  0  0  0  0  0
 0  0  0  0  0  0
 0  0  0  0  0  0
 0  0  0  0  0  0
```

```
E = ones(Int64, 6, 6)
```

```
6×6 Matrix{Int64}:
 1  1  1  1  1  1
 1  1  1  1  1  1
 1  1  1  1  1  1
 1  1  1  1  1  1
 1  1  1  1  1  1
 1  1  1  1  1  1
```

Рис. 3.16. Создание матриц Z и E

```
Z1 = zeros(Int64, 6, 6)
for i in 1:1:6
    if i != 1
        Z1[i, i - 1] = E[i, i - 1]
    end
    if i != 6
        Z1[i, i + 1] = E[i, i + 1]
    end
end
Z1
```

```
6×6 Matrix{Int64}:
 0  1  0  0  0  0
 1  0  1  0  0  0
 0  1  0  1  0  0
 0  0  1  0  1  0
 0  0  0  1  0  1
 0  0  0  0  1  0
```

Рис. 3.17. Создание матрицы Z1

```
Z2 = zeros(Int64, 6, 6)
for i in 1:1:6
    Z2[i,i] = 1
    if(i+2 <= 6) Z2[i, i + 2] = E[i, i + 2] end
    if(i-2 >= 1) Z2[i, i - 2] = E[i, i - 2] end
end
Z2
```

```
6×6 Matrix{Int64}:
 1  0  1  0  0  0
 0  1  0  1  0  0
 1  0  1  0  1  0
 0  1  0  1  0  1
 0  0  1  0  1  0
 0  0  0  1  0  1
```

Рис. 3.18. Создание матрицы Z2

```
Z3 = zeros(Int64, 6, 6)
for i in 1:1:6
    Z3[i,7-i] = 1
    if((7-i+2) <= 6) Z3[i, 9 - i] = E[i, 9 - i] end
    if((7-i-2) >= 1) Z3[i, 5 - i] = E[i, 5 - i] end
end
Z3
```

```
6×6 Matrix{Int64}:
 0  0  0  1  0  1
 0  0  1  0  1  0
 0  1  0  1  0  1
 1  0  1  0  1  0
 0  1  0  1  0  0
 1  0  1  0  0  0
```

Рис. 3.19. Создание матрицы Z3

```
Z4 = zeros(Int64, 6, 6)
for i in 1:1:6
    Z4[i,i] = 1
    if(i+2 <= 6) Z4[i, i + 2] = E[i, i + 2] end
    if(i-2 >= 1) Z4[i, i - 2] = E[i, i - 2] end
    if(i+4 <= 6) Z4[i, i + 4] = E[i, i + 4] end
    if(i-4 >= 1) Z4[i, i - 4] = E[i, i - 4] end
end
Z4
```

```
6×6 Matrix{Int64}:
 1  0  1  0  1  0
 0  1  0  1  0  1
 1  0  1  0  1  0
 0  1  0  1  0  1
 1  0  1  0  1  0
 0  1  0  1  0  1
```

Рис. 3.20. Создание матрицы Z4

13. Вычислите (рис. 3.21):

$$\sum_{i=1}^{20}\sum_{j=1}^{5}\frac{i^4}{(3+j)}$$

$$\sum_{i=1}^{20}\sum_{j=1}^{5}\frac{i^4}{(3+ij)}$$

```
function1 = 0
for i in 1:1:20
    for j in 1:1:5
        function1 += (i^4)/(3+j)
    end
end
println(function1)
```

```
639215.2833333334
```

```
function2 = 0
for i in 1:1:20
    for j in 1:1:5
        function2 += (i^4)/(3+i*j)
    end
end
println(function2)
```

```
89912.02146097136
```

Рис. 3.21. Вычисление

Вывод

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

Приложение


```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "id": "d140c1d9",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "1\n",
      "2\n",
      "3\n",
      "4\n",
      "5\n",
      "6\n",
      "7\n",
      "8\n",
      "9\n",
      "10\n"
     ]
    }
   ],
   "source": [
    "# пока n<10 прибавить к n единицу и распечатать значение:\n",
    "n = 0\n",
    "while n < 10\n",
    "    n += 1\n",
    "    println(n)\n",
    "end"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 5,
   "id": "86eff3ba",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Hi Ted, it's great to see you!\n",
      "Hi Robyn, it's great to see you!\n",
      "Hi Barney, it's great to see you!\n",
      "Hi Lily, it's great to see you!\n",
      "Hi Marshall, it's great to see you!\n"
     ]
    }
   }
```

```
 ],
 "source": [
  "myfriends = [\"Ted\", \"Robyn\", \"Barney\", \"Lily\", \"Marshall\"]\n",
  "i = 1\n",
  "while i <= length(myfriends)\n",
  "    friend = myfriends[i]\n",
  "    println(\"Hi $friend, it's great to see you!\")\n",
  "    i += 1\n",
  "end"
 ]
},
{
 "cell_type": "code",
 "execution_count": 7,
 "id": "0f8131f8",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "1\n",
    "3\n",
    "5\n",
    "7\n",
    "9\n"
   ]
  }
 ],
 "source": [
  "for n in 1:2:10\n",
  "    println(n)\n",
  "end"
 ]
},
{
 "cell_type": "code",
 "execution_count": 8,
 "id": "1f661322",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Hi Ted, it's great to see you!\n",
    "Hi Robyn, it's great to see you!\n",
    "Hi Barney, it's great to see you!\n",
    "Hi Lily, it's great to see you!\n",
    "Hi Marshall, it's great to see you!\n"
   ]
  }
 ],
```

```
 "source": [
  "myfriends = [\"Ted\", \"Robyn\", \"Barney\", \"Lily\", \"Marshall\"]\n",
  "for friend in myfriends\n",
  "    println(\"Hi $friend, it's great to see you!\")\n",
  "end"
 ]
},
{
 "cell_type": "code",
 "execution_count": 9,
 "id": "290302cf",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "5×5 Matrix{Int64}:\n",
     " 0  0  0  0  0\n",
     " 0  0  0  0  0\n",
     " 0  0  0  0  0\n",
     " 0  0  0  0  0\n",
     " 0  0  0  0  0"
    ]
   },
   "execution_count": 9,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "# инициализация массива m x n из нулей:\n",
  "m, n = 5, 5\n",
  "A = fill(0, (m, n))"
 ]
},
{
 "cell_type": "code",
 "execution_count": 10,
 "id": "91dab142",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "5×5 Matrix{Int64}:\n",
     " 2  3  4  5   6\n",
     " 3  4  5  6   7\n",
     " 4  5  6  7   8\n",
     " 5  6  7  8   9\n",
     " 6  7  8  9  10"
    ]
   },
   "execution_count": 10,
```

```
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "# формирование массива, в котором значение каждой записи\n",
 "# является суммой индексов строки и столбца:\n",
 "for i in 1:m\n",
 "    for j in 1:n\n",
 "        A[i, j] = i + j\n",
 "    end\n",
 "end\n",
 "A"
]
},
{
 "cell_type": "code",
 "execution_count": 11,
 "id": "a8dd6c3d",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "5×5 Matrix{Int64}:\n",
     " 2  3  4  5   6\n",
     " 3  4  5  6   7\n",
     " 4  5  6  7   8\n",
     " 5  6  7  8   9\n",
     " 6  7  8  9  10"
    ]
   },
   "execution_count": 11,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "# инициализация массива m x n из нулей:\n",
  "B = fill(0, (m, n))\n",
  "\n",
  "for i in 1:m, j in 1:n\n",
  "    B[i, j] = i + j\n",
  "end\n",
  "B"
 ]
},
{
 "cell_type": "code",
 "execution_count": 13,
 "id": "d4afb5ce",
 "metadata": {},
 "outputs": [
```

```
  {
   "data": {
    "text/plain": [
     "5×5 Matrix{Int64}:\n",
     " 2  3  4  5   6\n",
     " 3  4  5  6   7\n",
     " 4  5  6  7   8\n",
     " 5  6  7  8   9\n",
     " 6  7  8  9  10"
    ]
   },
   "execution_count": 13,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "# Ещё одна реализация этого же примера:\n",
  "C = [i + j for i in 1:m, j in 1:n]\n",
  "C"
 ]
},
{
 "cell_type": "code",
 "execution_count": 21,
 "id": "e8f5a800",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Fizz\n"
   ]
  }
 ],
 "source": [
  "# используем `&&` для реализации операции \"AND\"\n",
  "# операция % вычисляет остаток от деления\n",
  "N = 3\n",
  "\n",
  "if (N % 3 == 0) && (N % 5 == 0)\n",
  "    println(\"FizzBuzz\")\n",
  "elseif N % 3 == 0\n",
  "    println(\"Fizz\")\n",
  "elseif N % 5 == 0\n",
  "    println(\"Buzz\")\n",
  "else\n",
  "    println(N)\n",
  "end"
 ]
},
{
```

```
 "cell_type": "code",
 "execution_count": 22,
 "id": "8b60360f",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "10"
    ]
   },
   "execution_count": 22,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "x = 5\n",
  "y = 10\n",
  "(x > y) ? x : y"
 ]
},
{
 "cell_type": "code",
 "execution_count": 24,
 "id": "485ab684",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Hi Gleb, it's great to see you!\n"
   ]
  }
 ],
 "source": [
  "function sayhi(name)\n",
  "    println(\"Hi $name, it's great to see you!\")\n",
  "end\n",
  "sayhi(\"Gleb\")"
 ]
},
{
 "cell_type": "code",
 "execution_count": 25,
 "id": "b75ebb6d",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "9"
```

```
   ]
   },
   "execution_count": 25,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "function f(x)\n",
  "    x^2\n",
  "end\n",
  "f(3)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 28,
 "id": "0ad8a98c",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Hi Glebushka, it's great to see you!\n"
   ]
  }
 ],
 "source": [
  "sayhi2(name) = println(\"Hi $name, it's great to see you!\")\n",
  "sayhi2(\"Glebushka\")"
 ]
},
{
 "cell_type": "code",
 "execution_count": 29,
 "id": "ac71a506",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "16"
    ]
   },
   "execution_count": 29,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "f2(x) = x^2\n",
  "f2(4)"
```

```
 ]
},
{
 "cell_type": "code",
 "execution_count": 38,
 "id": "f586e976",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Hi Glebati, it's great to see you!\n"
   ]
  }
 ],
 "source": [
  "sayhi3 = name -> println(\"Hi $name, it's great to see you!\")\n",
  "sayhi2(\"Glebati\")"
 ]
},
{
 "cell_type": "code",
 "execution_count": 39,
 "id": "972fd5b0",
 "metadata": {},
 "outputs": [],
 "source": [
  "#f3(x) = x -> x^2\n",
  "#f3(5)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 43,
 "id": "08738397",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3-element Vector{Int64}:\n",
     " 3\n",
     " 5\n",
     " 2"
    ]
   },
   "execution_count": 43,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
```

```
  "# задаём массив v:\n",
  "v = [3, 5, 2]\n",
  "sort(v)\n",
  "v"
 ]
},
{
 "cell_type": "code",
 "execution_count": 44,
 "id": "a6994b83",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3-element Vector{Int64}:\n",
     " 2\n",
     " 3\n",
     " 5"
    ]
   },
   "execution_count": 44,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "sort!(v)\n",
  "v"
 ]
},
{
 "cell_type": "code",
 "execution_count": 46,
 "id": "195dd728",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3-element Vector{Int64}:\n",
     " 1\n",
     " 4\n",
     " 9"
    ]
   },
   "execution_count": 46,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "f(x) = x^2\n",
```

```
  "map(f, [1, 2, 3])"
 ]
},
{
 "cell_type": "code",
 "execution_count": 47,
 "id": "03db5aff",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3-element Vector{Int64}:\n",
     "  1\n",
     "  8\n",
     " 27"
    ]
   },
   "execution_count": 47,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "x -> x^3\n",
  "map(x -> x^3, [1, 2, 3])"
 ]
},
{
 "cell_type": "code",
 "execution_count": 48,
 "id": "2ee81761",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3×3 Matrix{Int64}:\n",
     " 1  2  3\n",
     " 4  5  6\n",
     " 7  8  9"
    ]
   },
   "execution_count": 48,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "# задаём матрицу A\n",
  "A = [i + 3*j for j in 0:2, i in 1:3]"
 ]
},
```

```
{
 "cell_type": "code",
 "execution_count": 49,
 "id": "81f6569b",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3×3 Matrix{Int64}:\n",
     "  30   36   42\n",
     "  66   81   96\n",
     " 102  126  150"
    ]
   },
   "execution_count": 49,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "# вызываем функцию f возведения в квадрат\n",
  "f(A)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 50,
 "id": "c57ee3af",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3×3 Matrix{Int64}:\n",
     "  1   4   9\n",
     " 16  25  36\n",
     " 49  64  81"
    ]
   },
   "execution_count": 50,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "B = f.(A)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 51,
 "id": "9d5a0a62",
```

```
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3×3 Matrix{Float64}:\n",
     "  3.0   6.0   9.0\n",
     " 12.0  15.0  18.0\n",
     " 21.0  24.0  27.0"
    ]
   },
   "execution_count": 51,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "A .+ 2 .* f.(A) ./ A"
 ]
},
{
 "cell_type": "code",
 "execution_count": 52,
 "id": "e469e79c",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3×3 Matrix{Float64}:\n",
     "  3.0   6.0   9.0\n",
     " 12.0  15.0  18.0\n",
     " 21.0  24.0  27.0"
    ]
   },
   "execution_count": 52,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "@. A + 2 * f(A) / A"
 ]
},
{
 "cell_type": "code",
 "execution_count": 53,
 "id": "cffa533c",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
```

```
    "3×3 Matrix{Float64}:\n",
    "  3.0   6.0   9.0\n",
    " 12.0  15.0  18.0\n",
    " 21.0  24.0  27.0"
   ]
  },
  "execution_count": 53,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "broadcast(x -> x + 2 * f(x) / x, A)"
]
},
{
"cell_type": "code",
"execution_count": 3,
"id": "e12fcb9f",
"metadata": {},
"outputs": [
 {
  "name": "stderr",
  "output_type": "stream",
  "text": [
   "\u001b[32m\u001b[1m   Resolving\u001b[22m\u001b[39m package versions...\n",
   "\u001b[32m\u001b[1m  No Changes\u001b[22m\u001b[39m to `C:\\Users\\GlebB\\.julia\\environments\\v1.9\\Project.toml`\n",
   "\u001b[32m\u001b[1m  No Changes\u001b[22m\u001b[39m to `C:\\Users\\GlebB\\.julia\\environments\\v1.9\\Manifest.toml`\n"
  ]
 }
],
"source": [
 "import Pkg\n",
 "Pkg.add(\"Example\")"
]
},
{
"cell_type": "code",
"execution_count": 4,
"id": "187910b4",
"metadata": {},
"outputs": [
 {
  "name": "stderr",
  "output_type": "stream",
  "text": [
   "\u001b[32m\u001b[1m   Resolving\u001b[22m\u001b[39m package versions...\n",
   "\u001b[32m\u001b[1m  No Changes\u001b[22m\u001b[39m to `C:\\Users\\GlebB\\.julia\\environments\\v1.9\\Project.toml`\n",
   "\u001b[32m\u001b[1m  No Changes\u001b[22m\u001b[39m to `C:\\Users\\GlebB\\.julia\\environments\\v1.9\\Manifest.toml`\n"
  ]
 }
],
"source": [
```

```
  "Pkg.add(\"Colors\")\n",
  "using Colors"
 ]
},
{
 "cell_type": "code",
 "execution_count": 5,
 "id": "902c9e25",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "image/svg+xml": [
     "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n",
     "<!DOCTYPE svg PUBLIC \"-//W3C//DTD SVG 1.1//EN\"\n",
     " \"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd\">\n",
     "<svg xmlns=\"http://www.w3.org/2000/svg\" version=\"1.1\"\n",
     "     width=\"180mm\" height=\"25mm\"\n",
     "     viewBox=\"0 0 100 1\" preserveAspectRatio=\"none\"\n",
     "     shape-rendering=\"crispEdges\" stroke=\"none\">\n",
     "<rect width=\"1\" height=\".96\" x=\"0\" y=\"0\" fill=\"#000000\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"1\" y=\"0\" fill=\"#FFFF74\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"2\" y=\"0\" fill=\"#FF9BFF\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"3\" y=\"0\" fill=\"#00D3FF\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"4\" y=\"0\" fill=\"#E2630D\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"5\" y=\"0\" fill=\"#007E00\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"6\" y=\"0\" fill=\"#0050E6\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"7\" y=\"0\" fill=\"#AC0047\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"8\" y=\"0\" fill=\"#00FFC8\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"9\" y=\"0\" fill=\"#006468\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"10\" y=\"0\" fill=\"#FFD5C4\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"11\" y=\"0\" fill=\"#6C5200\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"12\" y=\"0\" fill=\"#7A7581\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"13\" y=\"0\" fill=\"#44005C\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"14\" y=\"0\" fill=\"#9E9E77\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"15\" y=\"0\" fill=\"#FF5C78\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"16\" y=\"0\" fill=\"#8197F1\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"17\" y=\"0\" fill=\"#003200\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"18\" y=\"0\" fill=\"#C721DD\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"19\" y=\"0\" fill=\"#FFAD07\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"20\" y=\"0\" fill=\"#611C00\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"21\" y=\"0\" fill=\"#F3FFFA\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"22\" y=\"0\" fill=\"#009E88\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"23\" y=\"0\" fill=\"#5EC700\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"24\" y=\"0\" fill=\"#002D54\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"25\" y=\"0\" fill=\"#553C4A\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"26\" y=\"0\" fill=\"#444439\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"27\" y=\"0\" fill=\"#008FB6\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"28\" y=\"0\" fill=\"#CFD4FD\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"29\" y=\"0\" fill=\"#C40000\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"30\" y=\"0\" fill=\"#A4675C\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"31\" y=\"0\" fill=\"#BB8FA8\" />\n",
     "<rect width=\"1\" height=\".96\" x=\"32\" y=\"0\" fill=\"#290001\" />\n",
```

```
"<rect width=\"1\" height=\".96\" x=\"33\" y=\"0\" fill=\"#A78600\" />\n",
"<rect width=\"1\" height=\".96\" x=\"34\" y=\"0\" fill=\"#002D30\" />\n",
"<rect width=\"1\" height=\".96\" x=\"35\" y=\"0\" fill=\"#C7DEAA\" />\n",
"<rect width=\"1\" height=\".96\" x=\"36\" y=\"0\" fill=\"#8D9FA3\" />\n",
"<rect width=\"1\" height=\".96\" x=\"37\" y=\"0\" fill=\"#6F5B95\" />\n",
"<rect width=\"1\" height=\".96\" x=\"38\" y=\"0\" fill=\"#A1FFFF\" />\n",
"<rect width=\"1\" height=\".96\" x=\"39\" y=\"0\" fill=\"#B39688\" />\n",
"<rect width=\"1\" height=\".96\" x=\"40\" y=\"0\" fill=\"#4E6D50\" />\n",
"<rect width=\"1\" height=\".96\" x=\"41\" y=\"0\" fill=\"#FF977B\" />\n",
"<rect width=\"1\" height=\".96\" x=\"42\" y=\"0\" fill=\"#FFD1EC\" />\n",
"<rect width=\"1\" height=\".96\" x=\"43\" y=\"0\" fill=\"#9E5100\" />\n",
"<rect width=\"1\" height=\".96\" x=\"44\" y=\"0\" fill=\"#AE5B8E\" />\n",
"<rect width=\"1\" height=\".96\" x=\"45\" y=\"0\" fill=\"#799400\" />\n",
"<rect width=\"1\" height=\".96\" x=\"46\" y=\"0\" fill=\"#362200\" />\n",
"<rect width=\"1\" height=\".96\" x=\"47\" y=\"0\" fill=\"#0E0026\" />\n",
"<rect width=\"1\" height=\".96\" x=\"48\" y=\"0\" fill=\"#80765F\" />\n",
"<rect width=\"1\" height=\".96\" x=\"49\" y=\"0\" fill=\"#485C00\" />\n",
"<rect width=\"1\" height=\".96\" x=\"50\" y=\"0\" fill=\"#C8C2B5\" />\n",
"<rect width=\"1\" height=\".96\" x=\"51\" y=\"0\" fill=\"#8800A1\" />\n",
"<rect width=\"1\" height=\".96\" x=\"52\" y=\"0\" fill=\"#00A853\" />\n",
"<rect width=\"1\" height=\".96\" x=\"53\" y=\"0\" fill=\"#FFE1AA\" />\n",
"<rect width=\"1\" height=\".96\" x=\"54\" y=\"0\" fill=\"#674F42\" />\n",
"<rect width=\"1\" height=\".96\" x=\"55\" y=\"0\" fill=\"#FF342D\" />\n",
"<rect width=\"1\" height=\".96\" x=\"56\" y=\"0\" fill=\"#6B0041\" />\n",
"<rect width=\"1\" height=\".96\" x=\"57\" y=\"0\" fill=\"#0806B1\" />\n",
"<rect width=\"1\" height=\".96\" x=\"58\" y=\"0\" fill=\"#986DFF\" />\n",
"<rect width=\"1\" height=\".96\" x=\"59\" y=\"0\" fill=\"#FF4EC7\" />\n",
"<rect width=\"1\" height=\".96\" x=\"60\" y=\"0\" fill=\"#8AB9A2\" />\n",
"<rect width=\"1\" height=\".96\" x=\"61\" y=\"0\" fill=\"#2EFF71\" />\n",
"<rect width=\"1\" height=\".96\" x=\"62\" y=\"0\" fill=\"#005577\" />\n",
"<rect width=\"1\" height=\".96\" x=\"63\" y=\"0\" fill=\"#0078E3\" />\n",
"<rect width=\"1\" height=\".96\" x=\"64\" y=\"0\" fill=\"#B2ADB9\" />\n",
"<rect width=\"1\" height=\".96\" x=\"65\" y=\"0\" fill=\"#00C3C6\" />\n",
"<rect width=\"1\" height=\".96\" x=\"66\" y=\"0\" fill=\"#00AEFF\" />\n",
"<rect width=\"1\" height=\".96\" x=\"67\" y=\"0\" fill=\"#4E545F\" />\n",
"<rect width=\"1\" height=\".96\" x=\"68\" y=\"0\" fill=\"#FF9BB0\" />\n",
"<rect width=\"1\" height=\".96\" x=\"69\" y=\"0\" fill=\"#FED206\" />\n",
"<rect width=\"1\" height=\".96\" x=\"70\" y=\"0\" fill=\"#687B7A\" />\n",
"<rect width=\"1\" height=\".96\" x=\"71\" y=\"0\" fill=\"#B1DCFC\" />\n",
"<rect width=\"1\" height=\".96\" x=\"72\" y=\"0\" fill=\"#FFF6FF\" />\n",
"<rect width=\"1\" height=\".96\" x=\"73\" y=\"0\" fill=\"#620019\" />\n",
"<rect width=\"1\" height=\".96\" x=\"74\" y=\"0\" fill=\"#C79253\" />\n",
"<rect width=\"1\" height=\".96\" x=\"75\" y=\"0\" fill=\"#A891CF\" />\n",
"<rect width=\"1\" height=\".96\" x=\"76\" y=\"0\" fill=\"#EF007A\" />\n",
"<rect width=\"1\" height=\".96\" x=\"77\" y=\"0\" fill=\"#B8CE00\" />\n",
"<rect width=\"1\" height=\".96\" x=\"78\" y=\"0\" fill=\"#001700\" />\n",
"<rect width=\"1\" height=\".96\" x=\"79\" y=\"0\" fill=\"#204B39\" />\n",
"<rect width=\"1\" height=\".96\" x=\"80\" y=\"0\" fill=\"#875866\" />\n",
"<rect width=\"1\" height=\".96\" x=\"81\" y=\"0\" fill=\"#B5FF4E\" />\n",
"<rect width=\"1\" height=\".96\" x=\"82\" y=\"0\" fill=\"#B40080\" />\n",
"<rect width=\"1\" height=\".96\" x=\"83\" y=\"0\" fill=\"#853F34\" />\n",
"<rect width=\"1\" height=\".96\" x=\"84\" y=\"0\" fill=\"#69936B\" />\n",
"<rect width=\"1\" height=\".96\" x=\"85\" y=\"0\" fill=\"#FFBC80\" />\n",
```

```
   "<rect width=\"1\" height=\".96\" x=\"86\" y=\"0\" fill=\"#4C3779\" />\n",
   "<rect width=\"1\" height=\".96\" x=\"87\" y=\"0\" fill=\"#323606\" />\n",
   "<rect width=\"1\" height=\".96\" x=\"88\" y=\"0\" fill=\"#008E94\" />\n",
   "<rect width=\"1\" height=\".96\" x=\"89\" y=\"0\" fill=\"#CAAC51\" />\n",
   "<rect width=\"1\" height=\".96\" x=\"90\" y=\"0\" fill=\"#787B3B\" />\n",
   "<rect width=\"1\" height=\".96\" x=\"91\" y=\"0\" fill=\"#B6F9D9\" />\n",
   "<rect width=\"1\" height=\".96\" x=\"92\" y=\"0\" fill=\"#DA003F\" />\n",
   "<rect width=\"1\" height=\".96\" x=\"93\" y=\"0\" fill=\"#2E2124\" />\n",
   "<rect width=\"1\" height=\".96\" x=\"94\" y=\"0\" fill=\"#005815\" />\n",
   "<rect width=\"1\" height=\".96\" x=\"95\" y=\"0\" fill=\"#FF8E1D\" />\n",
   "<rect width=\"1\" height=\".96\" x=\"96\" y=\"0\" fill=\"#6674B1\" />\n",
   "<rect width=\"1\" height=\".96\" x=\"97\" y=\"0\" fill=\"#00CDAD\" />\n",
   "<rect width=\"1\" height=\".96\" x=\"98\" y=\"0\" fill=\"#007F63\" />\n",
   "<rect width=\"1\" height=\".96\" x=\"99\" y=\"0\" fill=\"#996F3D\" />\n",
   "</svg>"
  ],
  "text/plain": [
   "100-element Array{RGB{N0f8},1} with eltype RGB{FixedPointNumbers.N0f8}:\n",
   " RGB{N0f8}(0.0,0.0,0.0)\n",
   " RGB{N0f8}(1.0,1.0,0.455)\n",
   " RGB{N0f8}(1.0,0.608,1.0)\n",
   " RGB{N0f8}(0.0,0.827,1.0)\n",
   " RGB{N0f8}(0.886,0.388,0.051)\n",
   " RGB{N0f8}(0.0,0.494,0.0)\n",
   " RGB{N0f8}(0.0,0.314,0.902)\n",
   " RGB{N0f8}(0.675,0.0,0.278)\n",
   " RGB{N0f8}(0.0,1.0,0.784)\n",
   " RGB{N0f8}(0.0,0.392,0.408)\n",
   " RGB{N0f8}(1.0,0.835,0.769)\n",
   " RGB{N0f8}(0.424,0.322,0.0)\n",
   " RGB{N0f8}(0.478,0.459,0.506)\n",
   " :\n",
   " RGB{N0f8}(0.0,0.557,0.58)\n",
   " RGB{N0f8}(0.792,0.675,0.318)\n",
   " RGB{N0f8}(0.471,0.482,0.231)\n",
   " RGB{N0f8}(0.714,0.976,0.851)\n",
   " RGB{N0f8}(0.855,0.0,0.247)\n",
   " RGB{N0f8}(0.18,0.129,0.141)\n",
   " RGB{N0f8}(0.0,0.345,0.082)\n",
   " RGB{N0f8}(1.0,0.557,0.114)\n",
   " RGB{N0f8}(0.4,0.455,0.694)\n",
   " RGB{N0f8}(0.0,0.804,0.678)\n",
   " RGB{N0f8}(0.0,0.498,0.388)\n",
   " RGB{N0f8}(0.6,0.435,0.239)"
  ]
 },
 "execution_count": 5,
 "metadata": {},
 "output_type": "execute_result"
}
],
"source": [
 "palette = distinguishable_colors(100)"
```

```
 ]
},
{
 "cell_type": "code",
 "execution_count": 57,
 "id": "002705a9",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "image/svg+xml": [
     "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n",
     "<!DOCTYPE svg PUBLIC \"-//W3C//DTD SVG 1.1//EN\"\n",
     " \"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd\">\n",
     "<svg xmlns=\"http://www.w3.org/2000/svg\" version=\"1.1\"\n",
     "     width=\"75mm\" height=\"75mm\"\n",
     "     viewBox=\"0 0 3 3\" preserveAspectRatio=\"none\"\n",
     "     shape-rendering=\"crispEdges\" stroke=\"none\">\n",
     "<rect width=\".96\" height=\".96\" x=\"0\" y=\"0\" fill=\"#B40080\" />\n",
     "<rect width=\".96\" height=\".96\" x=\"1\" y=\"0\" fill=\"#008FB6\" />\n",
     "<rect width=\".96\" height=\".96\" x=\"2\" y=\"0\" fill=\"#FED206\" />\n",
     "<rect width=\".96\" height=\".96\" x=\"0\" y=\"1\" fill=\"#FF5C78\" />\n",
     "<rect width=\".96\" height=\".96\" x=\"1\" y=\"1\" fill=\"#799400\" />\n",
     "<rect width=\".96\" height=\".96\" x=\"2\" y=\"1\" fill=\"#9E5100\" />\n",
     "<rect width=\".96\" height=\".96\" x=\"0\" y=\"2\" fill=\"#008FB6\" />\n",
     "<rect width=\".96\" height=\".96\" x=\"1\" y=\"2\" fill=\"#8800A1\" />\n",
     "<rect width=\".96\" height=\".96\" x=\"2\" y=\"2\" fill=\"#553C4A\" />\n",
     "</svg>"
    ],
    "text/plain": [
     "3×3 Array{RGB{N0f8},2} with eltype RGB{FixedPointNumbers.N0f8}:\n",
     " RGB{N0f8}(0.706,0.0,0.502)  …  RGB{N0f8}(0.996,0.824,0.024)\n",
     " RGB{N0f8}(1.0,0.361,0.471)     RGB{N0f8}(0.62,0.318,0.0)\n",
     " RGB{N0f8}(0.0,0.561,0.714)     RGB{N0f8}(0.333,0.235,0.29)"
    ]
   },
   "execution_count": 57,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "rand(palette, 3, 3)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 59,
 "id": "d952a61c",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
```

```
"output_type": "stream",
"text": [
 "1\n",
 "1\n",
 "2\n",
 "4\n",
 "3\n",
 "9\n",
 "4\n",
 "16\n",
 "5\n",
 "25\n",
 "6\n",
 "36\n",
 "7\n",
 "49\n",
 "8\n",
 "64\n",
 "9\n",
 "81\n",
 "10\n",
 "100\n",
 "11\n",
 "121\n",
 "12\n",
 "144\n",
 "13\n",
 "169\n",
 "14\n",
 "196\n",
 "15\n",
 "225\n",
 "16\n",
 "256\n",
 "17\n",
 "289\n",
 "18\n",
 "324\n",
 "19\n",
 "361\n",
 "20\n",
 "400\n",
 "21\n",
 "441\n",
 "22\n",
 "484\n",
 "23\n",
 "529\n",
 "24\n",
 "576\n",
 "25\n",
 "625\n",
 "26\n",
```

"676\n",
"27\n",
"729\n",
"28\n",
"784\n",
"29\n",
"841\n",
"30\n",
"900\n",
"31\n",
"961\n",
"32\n",
"1024\n",
"33\n",
"1089\n",
"34\n",
"1156\n",
"35\n",
"1225\n",
"36\n",
"1296\n",
"37\n",
"1369\n",
"38\n",
"1444\n",
"39\n",
"1521\n",
"40\n",
"1600\n",
"41\n",
"1681\n",
"42\n",
"1764\n",
"43\n",
"1849\n",
"44\n",
"1936\n",
"45\n",
"2025\n",
"46\n",
"2116\n",
"47\n",
"2209\n",
"48\n",
"2304\n",
"49\n",
"2401\n",
"50\n",
"2500\n",
"51\n",
"2601\n",
"52\n",
"2704\n",

"53\n",
"2809\n",
"54\n",
"2916\n",
"55\n",
"3025\n",
"56\n",
"3136\n",
"57\n",
"3249\n",
"58\n",
"3364\n",
"59\n",
"3481\n",
"60\n",
"3600\n",
"61\n",
"3721\n",
"62\n",
"3844\n",
"63\n",
"3969\n",
"64\n",
"4096\n",
"65\n",
"4225\n",
"66\n",
"4356\n",
"67\n",
"4489\n",
"68\n",
"4624\n",
"69\n",
"4761\n",
"70\n",
"4900\n",
"71\n",
"5041\n",
"72\n",
"5184\n",
"73\n",
"5329\n",
"74\n",
"5476\n",
"75\n",
"5625\n",
"76\n",
"5776\n",
"77\n",
"5929\n",
"78\n",
"6084\n",
"79\n",

```
   "6241\n",
   "80\n",
   "6400\n",
   "81\n",
   "6561\n",
   "82\n",
   "6724\n",
   "83\n",
   "6889\n",
   "84\n",
   "7056\n",
   "85\n",
   "7225\n",
   "86\n",
   "7396\n",
   "87\n",
   "7569\n",
   "88\n",
   "7744\n",
   "89\n",
   "7921\n",
   "90\n",
   "8100\n",
   "91\n",
   "8281\n",
   "92\n",
   "8464\n",
   "93\n",
   "8649\n",
   "94\n",
   "8836\n",
   "95\n",
   "9025\n",
   "96\n",
   "9216\n",
   "97\n",
   "9409\n",
   "98\n",
   "9604\n",
   "99\n",
   "9801\n",
   "100\n",
   "10000\n"
  ]
  }
 ],
 "source": [
  "i = 1\n",
  "while i <= 100\n",
  "    println(i)\n",
  "    println(i^2)\n",
  "    i += 1\n",
  "end"
```

```
 ]
},
{
 "cell_type": "code",
 "execution_count": 60,
 "id": "5ef3816a",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "1\n",
    "2\n",
    "3\n",
    "4\n",
    "5\n",
    "6\n",
    "7\n",
    "8\n",
    "9\n",
    "10\n",
    "11\n",
    "12\n",
    "13\n",
    "14\n",
    "15\n",
    "16\n",
    "17\n",
    "18\n",
    "19\n",
    "20\n",
    "21\n",
    "22\n",
    "23\n",
    "24\n",
    "25\n",
    "26\n",
    "27\n",
    "28\n",
    "29\n",
    "30\n",
    "31\n",
    "32\n",
    "33\n",
    "34\n",
    "35\n",
    "36\n",
    "37\n",
    "38\n",
    "39\n",
    "40\n",
    "41\n",
```

"42\n",
"43\n",
"44\n",
"45\n",
"46\n",
"47\n",
"48\n",
"49\n",
"50\n",
"51\n",
"52\n",
"53\n",
"54\n",
"55\n",
"56\n",
"57\n",
"58\n",
"59\n",
"60\n",
"61\n",
"62\n",
"63\n",
"64\n",
"65\n",
"66\n",
"67\n",
"68\n",
"69\n",
"70\n",
"71\n",
"72\n",
"73\n",
"74\n",
"75\n",
"76\n",
"77\n",
"78\n",
"79\n",
"80\n",
"81\n",
"82\n",
"83\n",
"84\n",
"85\n",
"86\n",
"87\n",
"88\n",
"89\n",
"90\n",
"91\n",
"92\n",
"93\n",
"94\n",

```
   "95\n",
   "96\n",
   "97\n",
   "98\n",
   "99\n",
   "100\n"
  ]
 }
],
"source": [
 "for i in 1:1:100\n",
 "    println(i)\n",
 "end"
]
},
{
"cell_type": "code",
"execution_count": 61,
"id": "0062c213",
"metadata": {},
"outputs": [
 {
  "data": {
   "text/plain": [
    "Dict{Int64, Int64} with 100 entries:\n",
    "  5  => 25\n",
    "  56 => 3136\n",
    "  35 => 1225\n",
    "  55 => 3025\n",
    "  60 => 3600\n",
    "  30 => 900\n",
    "  32 => 1024\n",
    "  6  => 36\n",
    "  67 => 4489\n",
    "  45 => 2025\n",
    "  73 => 5329\n",
    "  64 => 4096\n",
    "  90 => 8100\n",
    "  4  => 16\n",
    "  13 => 169\n",
    "  54 => 2916\n",
    "  63 => 3969\n",
    "  86 => 7396\n",
    "  91 => 8281\n",
    "  62 => 3844\n",
    "  58 => 3364\n",
    "  52 => 2704\n",
    "  12 => 144\n",
    "  28 => 784\n",
    "  75 => 5625\n",
    "  ⋮  => ⋮"
   ]
  },
```

```
  "execution_count": 61,
  "metadata": {},
  "output_type": "execute_result"
  }
 ],
 "source": [
  "squares = Dict{Int64, Int64}()\n",
  "for i in 1:1:100\n",
  "    push!(squares, i => i^2)\n",
  "end\n",
  "pairs(squares)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 64,
 "id": "a16dcc33",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "100-element Vector{Any}:\n",
     "     1\n",
     "     4\n",
     "     9\n",
     "    16\n",
     "    25\n",
     "    36\n",
     "    49\n",
     "    64\n",
     "    81\n",
     "   100\n",
     "   121\n",
     "   144\n",
     "   169\n",
     "     ⋮\n",
     "  7921\n",
     "  8100\n",
     "  8281\n",
     "  8464\n",
     "  8649\n",
     "  8836\n",
     "  9025\n",
     "  9216\n",
     "  9409\n",
     "  9604\n",
     "  9801\n",
     " 10000"
    ]
   },
   "execution_count": 64,
   "metadata": {},
```

```
   "output_type": "execute_result"
  }
 ],
 "source": [
  "N = []\n",
  "for i in 1:1:100\n",
  "    append!(N, i)\n",
  "end\n",
  "\n",
  "squares_arr = []\n",
  "i = 1\n",
  "while i <= length(N)\n",
  "    append!(squares_arr, N[i]^2)\n",
  "    i += 1\n",
  "end\n",
  "squares_arr"
 ]
},
{
 "cell_type": "code",
 "execution_count": 65,
 "id": "d2a21c0c",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Нечетное\n"
   ]
  }
 ],
 "source": [
  "N = 7\n",
  "\n",
  "if N % 2 == 0\n",
  "    println(\"Четное\")\n",
  "else\n",
  "    println(\"Нечетное\")\n",
  "end"
 ]
},
{
 "cell_type": "code",
 "execution_count": 67,
 "id": "2a410644",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Четное\n"
```

```
   ]
  }
 ],
 "source": [
  "N = 8\n",
  "(N % 2 == 0) ? println(\"Четное\") : println(\"Нечетное\")"
 ]
},
{
 "cell_type": "code",
 "execution_count": 68,
 "id": "1e0b45ac",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3"
    ]
   },
   "execution_count": 68,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "function add_one(A)\n",
  "    A + 1\n",
  "end\n",
  "add_one(2)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 75,
 "id": "d2714029",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3×3 Matrix{Int64}:\n",
     " 2  3   4\n",
     " 5  6   7\n",
     " 8  9  10"
    ]
   },
   "execution_count": 75,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
```

```
  "broadcast(x -> x + 1, A)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 76,
 "id": "5e71d0f7",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3×3 Matrix{Int64}:\n",
     "  1   1   3\n",
     "  5   2   6\n",
     " -2  -1  -3"
    ]
   },
   "execution_count": 76,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "A = [1 1 3; 5 2 6; -2 -1 -3]"
 ]
},
{
 "cell_type": "code",
 "execution_count": 77,
 "id": "9f12bf96",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3×3 Matrix{Int64}:\n",
     " 0  0  0\n",
     " 0  0  0\n",
     " 0  0  0"
    ]
   },
   "execution_count": 77,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "A^3"
 ]
},
{
 "cell_type": "code",
```

```
 "execution_count": 80,
 "id": "55111aeb",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3×3 Matrix{Int64}:\n",
     "  1   1   6\n",
     "  5   2  12\n",
     " -2  -1  -6"
    ]
   },
   "execution_count": 80,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "for i in 7:1:9\n",
  "    A[i] += A[i-3]\n",
  "end\n",
  "A"
 ]
},
{
 "cell_type": "code",
 "execution_count": 6,
 "id": "0d2e6d03",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "15×3 Matrix{Int32}:\n",
     " 10  -10  10\n",
     " 10  -10  10\n",
     " 10  -10  10\n",
     " 10  -10  10\n",
     " 10  -10  10\n",
     " 10  -10  10\n",
     " 10  -10  10\n",
     " 10  -10  10\n",
     " 10  -10  10\n",
     " 10  -10  10\n",
     " 10  -10  10\n",
     " 10  -10  10\n",
     " 10  -10  10\n",
     " 10  -10  10\n",
     " 10  -10  10"
    ]
   },
   "execution_count": 6,
```

```
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "B = Array{Int32, 2}(undef, 15, 3)\n",
 "for i in 1:1:15\n",
 "    B[i, 1] = 10\n",
 "    B[i, 2] = -10\n",
 "    B[i, 3] = 10\n",
 "end\n",
 "B"
]
},
{
 "cell_type": "code",
 "execution_count": 7,
 "id": "da72e208",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "3×3 Matrix{Int32}:\n",
     "  1500  -1500   1500\n",
     " -1500   1500  -1500\n",
     "  1500  -1500   1500"
    ]
   },
   "execution_count": 7,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "C = (B')*B\n",
  "C"
 ]
},
{
 "cell_type": "code",
 "execution_count": 8,
 "id": "3fc0d1e8",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "6×6 Matrix{Int64}:\n",
     " 0  0  0  0  0  0\n",
     " 0  0  0  0  0  0\n",
     " 0  0  0  0  0  0\n",
     " 0  0  0  0  0  0\n",
```

```
    " 0  0  0  0  0  0\n",
    " 0  0  0  0  0  0"
   ]
  },
  "execution_count": 8,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "Z = zeros(Int64, 6, 6)"
]
},
{
"cell_type": "code",
"execution_count": 9,
"id": "f65c3153",
"metadata": {},
"outputs": [
 {
  "data": {
   "text/plain": [
    "6×6 Matrix{Int64}:\n",
    " 1  1  1  1  1  1\n",
    " 1  1  1  1  1  1\n",
    " 1  1  1  1  1  1\n",
    " 1  1  1  1  1  1\n",
    " 1  1  1  1  1  1\n",
    " 1  1  1  1  1  1"
   ]
  },
  "execution_count": 9,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "E = ones(Int64, 6, 6)"
]
},
{
"cell_type": "code",
"execution_count": 14,
"id": "8014b662",
"metadata": {},
"outputs": [
 {
  "data": {
   "text/plain": [
    "6×6 Matrix{Int64}:\n",
    " 0  1  0  0  0  0\n",
    " 1  0  1  0  0  0\n",
    " 0  1  0  1  0  0\n",
```

```
    " 0  0  1  0  1  0\n",
    " 0  0  0  1  0  1\n",
    " 0  0  0  0  1  0"
   ]
  },
  "execution_count": 14,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "Z1 = zeros(Int64, 6, 6)\n",
 "for i in 1:1:6\n",
 "    if i != 1\n",
 "        Z1[i, i - 1] = E[i, i - 1]\n",
 "    end\n",
 "    if i != 6\n",
 "        Z1[i, i + 1] = E[i, i + 1]\n",
 "    end\n",
 "end\n",
 "Z1"
]
},
{
 "cell_type": "code",
 "execution_count": 16,
 "id": "3061b5a9",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "6×6 Matrix{Int64}:\n",
     " 1  0  1  0  0  0\n",
     " 0  1  0  1  0  0\n",
     " 1  0  1  0  1  0\n",
     " 0  1  0  1  0  1\n",
     " 0  0  1  0  1  0\n",
     " 0  0  0  1  0  1"
    ]
   },
   "execution_count": 16,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "Z2 = zeros(Int64, 6, 6)\n",
  "for i in 1:1:6\n",
  "    Z2[i,i] = 1\n",
  "    if(i+2 <= 6) Z2[i, i + 2] = E[i, i + 2] end\n",
  "    if(i-2 >= 1) Z2[i, i - 2] = E[i, i - 2] end\n",
  "end\n",
```

```
  "Z2"
 ]
},
{
 "cell_type": "code",
 "execution_count": 17,
 "id": "0fb2e580",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "6×6 Matrix{Int64}:\n",
     " 0  0  0  1  0  1\n",
     " 0  0  1  0  1  0\n",
     " 0  1  0  1  0  1\n",
     " 1  0  1  0  1  0\n",
     " 0  1  0  1  0  0\n",
     " 1  0  1  0  0  0"
    ]
   },
   "execution_count": 17,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "Z3 = zeros(Int64, 6, 6)\n",
  "for i in 1:1:6\n",
  "    Z3[i,7-i] = 1\n",
  "    if((7-i+2) <= 6) Z3[i, 9 - i] = E[i, 9 - i] end\n",
  "    if((7-i-2) >= 1) Z3[i, 5 - i] = E[i, 5 - i] end\n",
  "end\n",
  "Z3"
 ]
},
{
 "cell_type": "code",
 "execution_count": 19,
 "id": "07e9faa0",
 "metadata": {
  "scrolled": true
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "6×6 Matrix{Int64}:\n",
     " 1  0  1  0  1  0\n",
     " 0  1  0  1  0  1\n",
     " 1  0  1  0  1  0\n",
     " 0  1  0  1  0  1\n",
     " 1  0  1  0  1  0\n",
```

```
    " 0  1  0  1  0  1"
   ]
  },
  "execution_count": 19,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "Z4 = zeros(Int64, 6, 6)\n",
 "for i in 1:1:6\n",
 "    Z4[i,i] = 1\n",
 "    if(i+2 <= 6) Z4[i, i + 2] = E[i, i + 2] end\n",
 "    if(i-2 >= 1) Z4[i, i - 2] = E[i, i - 2] end\n",
 "    if(i+4 <= 6) Z4[i, i + 4] = E[i, i + 4] end\n",
 "    if(i-4 >= 1) Z4[i, i - 4] = E[i, i - 4] end\n",
 "end\n",
 "Z4"
]
},
{
 "cell_type": "code",
 "execution_count": 20,
 "id": "eb587bdf",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "639215.2833333334\n"
   ]
  }
 ],
 "source": [
  "function1 = 0\n",
  "for i in 1:1:20\n",
  "    for j in 1:1:5\n",
  "        function1 += (i^4)/(3+j)\n",
  "    end\n",
  "end\n",
  "println(function1)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 21,
 "id": "ba6e9506",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
```

```
    "text": [
     "89912.02146097136\n"
    ]
   }
  ],
  "source": [
   "function2 = 0\n",
   "for i in 1:1:20\n",
   "    for j in 1:1:5\n",
   "        function2 += (i^4)/(3+i*j)\n",
   "    end\n",
   "end\n",
   "println(function2)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "66cb9b1e",
  "metadata": {},
  "outputs": [],
  "source": []
 }
],
"metadata": {
 "kernelspec": {
  "display_name": "Julia 1.9.3",
  "language": "julia",
  "name": "julia-1.9"
 },
 "language_info": {
  "file_extension": ".jl",
  "mimetype": "application/julia",
  "name": "julia",
  "version": "1.9.3"
 }
},
"nbformat": 4,
"nbformat_minor": 5
}
```