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: Ли Тимофей Александрович

Группа: НФИбд-01-18

МОСКВА

2021 г.

Постановка задачи

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

Выполнение работы

Сначала выполнил все примеры к лабораторной работе №3:

Научился пользоваться циклами, условными конструкциями, функциями. Также ознакомился с использованием сторонних пакетов.

```
n=0
while n<10
    n+=1
    println(n)
end
```

```
1
2
3
4
5
6
7
8
9
10
```

```
myfriends = ["Ted", "Robyn", "Barney", "Lily", "Marshall"]
i = 1
while i <= length(myfriends)
    friend = myfriends[i]
    println("Hi $friend, it's great to see you!")
    i += 1
end
```

```
Hi Ted, it's great to see you!
Hi Robyn, it's great to see you!
Hi Barney, it's great to see you!
Hi Lily, it's great to see you!
Hi Marshall, it's great to see you!
```

```
for n in 1:2:10
    println(n)
end
```

```
1
3
5
7
9
```

```
myfriends = ["Ted", "Robyn", "Barney", "Lily", "Marshall"]
for friend in myfriends
    println("Hi $friend, it's great to see you!")
end
```

```
Hi Ted, it's great to see you!
Hi Robyn, it's great to see you!
Hi Barney, it's great to see you!
Hi Lily, it's great to see you!
Hi Marshall, it's great to see you!
```

```
m,n=5,5
A=fill(0, (m,n))
for i in 1:m
    for j in 1:n
        A[i,j]=i+j
    end
end
A
```

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

```
B=fill(0, (m,n))
for i in 1:m, j in 1:n
    B[i,j]=i+j
end
B
```

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

```
C=[i+j for i in 1:m, j in 1:n]
C
```

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

```
N=5
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
```

```
Buzz
```

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

```
10
```

```
function sayhi(name)
    println("Hi $name, it's great to see you!")
end
function f(x)
    x^2
end
```

```
f (generic function with 1 method)
```

```
sayhi("C-3PO")
```

```
Hi C-3PO, it's great to see you!
```

```
f(42)
```

```
1764
```

```
sayhi2(name) = println("Hi $name, it's great to see you!")
f2(x) = x^2
sayhi3 = name -> println("Hi $name, it's great to see you!")
f3 = x -> x^2
```

```
#5 (generic function with 1 method)
```

```
f3(2)
```

```
4
```

```
v=[3,5,2]
sort(v)
```

```
3-element Vector{Int64}:
 2
 3
 5
```

```
v
```

```
3-element Vector{Int64}:
 3
 5
 2
```

```
sort!(v)
```

```
3-element Vector{Int64}:
 2
 3
 5
```

```
v
```

```
3-element Vector{Int64}:
 2
 3
 5
```

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

```
3-element Vector{Int64}:
 1
 4
 9
```

```
map(x->x^3, [1,2,3])
```

```
3-element Vector{Int64}:
  1
  8
 27
```

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

```
3-element Vector{Int64}:
 1
 4
 9
```

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

```
3-element Vector{Int64}:
 1
 4
 9
```

```
A = [i + 3*j for j in 0:2, i in 1:3]
```

```
3×3 Matrix{Int64}:
 1  2  3
 4  5  6
 7  8  9
```

```
f(A)
```

```
3×3 Matrix{Int64}:
  30   36   42
  66   81   96
 102  126  150
```

```
f.(A)
```

```
3×3 Matrix{Int64}:
  1   4   9
 16  25  36
 49  64  81
```

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

```
3×3 Matrix{Float64}:
  3.0   6.0   9.0
 12.0  15.0  18.0
 21.0  24.0  27.0
```

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

```
3×3 Matrix{Float64}:
  3.0   6.0   9.0
 12.0  15.0  18.0
 21.0  24.0  27.0
```

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

```
3×3 Matrix{Float64}:
  3.0   6.0   9.0
 12.0  15.0  18.0
 21.0  24.0  27.0
```

```
import Pkg
```

```
Pkg.add("Example")
```

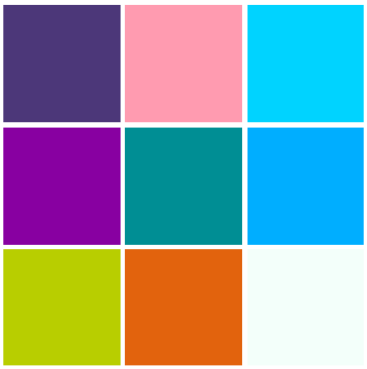
```
    Updating registry at `C:\Users\Xiaomi\.julia\registries\General`
    Updating git-repo `https://github.com/JuliaRegistries/General.gi
  Resolving package versions...
  Installed Example — v0.5.3
    Updating `C:\Users\Xiaomi\.julia\environments\v1.6\Project.toml`
```

```
Pkg.add("Colors")
using Colors
```

```
  Resolving package versions...
  Installed Reexport ──────────── v1.2.2
  Installed FixedPointNumbers ─── v0.8.4
  Installed ColorTypes ────────── v0.11.0
  Installed Colors ────────────── v0.12.8
   Updating `C:\Users\Xiaomi\.julia\environments\v1.6\Project.toml`
 [5ae59095] + Colors v0.12.8
   Updating `C:\Users\Xiaomi\.julia\environments\v1.6\Manifest.toml`
 [3da002f7] + ColorTypes v0.11.0
 [5ae59095] + Colors v0.12.8
 [53c48c17] + FixedPointNumbers v0.8.4
 [189a3867] + Reexport v1.2.2
 [37e2e46d] + LinearAlgebra
 [2f01184e] + SparseArrays
 [10745b16] + Statistics
Precompiling project...
  ✓ Reexport
  ✓ FixedPointNumbers
  ✓ ColorTypes
  ✓ Colors
  4 dependencies successfully precompiled in 18 seconds (17 already precompiled)
```

```
palette=distinguishable_colors(100)
```

```
rand(palette, 3, 3)
```



Далее выполнил поставленные задачи.

1. Используя циклы, вывел целые числа от 1 до 100 вместе с их квадратами, создал словарь squares, содержащий все эти значения, а также массив squares_arr, содержащий уже найденные квадраты.

```
for i in 1:100
    println(i, " ", i^2)
end
```

```
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
11 121
12 144
13 169
14 196
15 225
16 256
17 289
```

```
squares=Dict()
i=1
while i<101
    squares[i]=i^2
    i+=1
end
squares
```

```
Dict{Any, Any} 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
```

```
squares_arr=[]
for i in 1:100
    append!(squares_arr, i^2)
end
squares_arr
```

```
100-element Vector{Any}:
    1
    4
    9
   16
   25
   36
   49
   64
   81
  100
  121
  144
  169
    ⋮
```

2. Написал условный оператор, выводящий число, если оно честное, или слово «нечетный» в противном случае. Также переписал, используя тернарный оператор.

```
N=1
if N%2==0
    println(N)
else
    println("нечетное")
end
```

```
нечетное
```

```
N=1
(N%2==0) ? println(N) : println("нечетное")
```

```
нечетное
```

3. Написал функцию add_one, добавляющую единицу к вводному.

```
function add_one(x)
    x+1
end
add_one(1)
```

```
2
```

4. Используя broadcast, написал функцию, создающую матрицу, где каждый элемент на 1 больше предыдущего (на вход беру размеры матрицы и начальный элемент).

```
function mb(x, y, z)
    L=x*y
    A=fill(z, (L, 1))
    for i in 2:L
        A[i:L]=broadcast(+, 1, A[i:L])
    end
    A=reshape(A, (x,y))
end
mb(3, 3, 1)
```

```
3×3 Matrix{Int64}:
 1  4  7
 2  5  8
 3  6  9
```

```
mb(4,2,5)
```

```
4×2 Matrix{Int64}:
 5   9
 6  10
 7  11
 8  12
```

5. Для заданной матрицы A нашел A^3, а также заменил третий столбец на сумму двух первых

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

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

```
f(x)=x^3
f.(A)
```

```
3×3 Matrix{Int64}:
   1   1   27
 125   8  216
  -8  -1  -27
```

```
for i in 1:3
    A[i,3]=A[i,1]+A[i,2]
end
A
```

```
3×3 Matrix{Int64}:
  1   1   2
  5   2   7
 -2  -1  -3
```

6. Для заданной матрицы B посчитал $B^T * B$ (на скриншоте к 7 номеру).
7. Из матриц единиц и нулей 6*6 сделал требуемые матрицы. Для этого брал матрицу нулей и в цикле менял ее элементы на соответствующие элементы матрицы единиц, если они стоят на нужных местах.

```
B=repeat([10 -10 10], 15)
C=B' * B
```

```
3×3 Matrix{Int64}:
  1500  -1500   1500
 -1500   1500  -1500
  1500  -1500   1500
```

```
Z=fill(0, (6,6))
E=fill(1, (6,6))
Z1=Z
for i in 1:6
    for j in 1:6
        if abs(i-j)==1
            Z1[i,j]=E[i,j]
        end
    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
```

```
Z=fill(0, (6,6))
E=fill(1, (6,6))
Z2=Z
for i in 1:6
    for j in 1:6
        if abs(i-j)==2 || i==j
            Z2[i,j]=E[i,j]
        end
    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
```

```
Z3=Z2
for i in 1:3
    for j in 1:6
        tmp=Z3[i,j]
        Z3[i,j]=Z3[(7-i),j]
        Z3[(7-i),j]=tmp
    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
```

```
Z=fill(0, (6,6))
E=fill(1, (6,6))
Z4=Z
for i in 1:6
    for j in 1:6
        if abs(i-j)==2 || i==j || abs(i-j)==4
            Z4[i,j]=1
        else
            Z4[i,j]=0
        end
    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
```

8. Реализовал функцию outer и с ее помощью создал требуемые матрицы (на скриншоте к номеру 9)
9. Решил заданную систему уравнений

```
function outer(x,y,operation)
    return broadcast(operation,x,y)
end
```

```
outer (generic function with 1 method)
```

```
A=collect(0:4)
outer(A, A', +)
```

```
5×5 Matrix{Int64}:
 0  1  2  3  4
 1  2  3  4  5
 2  3  4  5  6
 3  4  5  6  7
 4  5  6  7  8
```

```
outer(A, collect(1:5)', ^)
```

```
5×5 Matrix{Int64}:
 0   0   0    0     0
 1   1   1    1     1
 2   4   8   16    32
 3   9  27   81   243
 4  16  64  256  1024
```

```
.%(outer(A,A',+), 5)
```

```
5×5 Matrix{Int64}:
 0  1  2  3  4
 1  2  3  4  0
 2  3  4  0  1
 3  4  0  1  2
 4  0  1  2  3
```

```
.%(outer(collect(0:9),collect(0:9)',+),10)
```

```
10×10 Matrix{Int64}:
 0  1  2  3  4  5  6  7  8  9
 1  2  3  4  5  6  7  8  9  0
 2  3  4  5  6  7  8  9  0  1
 3  4  5  6  7  8  9  0  1  2
 4  5  6  7  8  9  0  1  2  3
 5  6  7  8  9  0  1  2  3  4
 6  7  8  9  0  1  2  3  4  5
 7  8  9  0  1  2  3  4  5  6
 8  9  0  1  2  3  4  5  6  7
 9  0  1  2  3  4  5  6  7  8
```

```
.%(outer(collect(0:8),collect(9:-1:1)',+),9)
```

```
9×9 Matrix{Int64}:
 0  8  7  6  5  4  3  2  1
 1  0  8  7  6  5  4  3  2
 2  1  0  8  7  6  5  4  3
 3  2  1  0  8  7  6  5  4
 4  3  2  1  0  8  7  6  5
 5  4  3  2  1  0  8  7  6
 6  5  4  3  2  1  0  8  7
 7  6  5  4  3  2  1  0  8
 8  7  6  5  4  3  2  1  0
```

```
X=[1 2 3 4 5;2 1 2 3 4;3 2 1 2 3;4 3 2 1 2;5 4 3 2 1]
B=[7,-1,-3,5,17]
X\B
```

```
5-element Vector{Float64}:
 -2.0000000000000036
  3.0000000000000058
  4.999999999999998
  1.9999999999999991
 -3.999999999999999
```

-2
3
5
2
-4

10. Создал матрицу M 6*10 с элементами от 6 до 10. Для полученной матрицы нашел число элементов, больших N, в каждой строке, номера строк, в которых число N встречается ровно два раза, и пары столбцов, сумма элементов которых больше 70 (взял 70, а не 75, поскольку для 75 пар не было). (на скриншоте к номеру 11)

11. Посчитал данные суммы.

```
M=rand(1:10, (6,10))
```

```
6×10 Matrix{Int64}:
  3  3  1  3   8  9  6  9  7   3
  6  3  7  4   2  8  1  5  2   4
 10  8  2  7  10  1  4  3  2   5
  4  3  3  5   4  2  5  5  3  10
  6  7  6  4   3  4  8  7  9  10
  2  4  7  6   5  7  5  3  2   8
```

```
N=4
for i in 1:6
    print(i, " ")
    count=0
    for j in 1:10
        if M[i,j]>N
            count+=1
        end
    end
    println(count)
end
```

```
1 5
2 4
3 5
4 4
5 7
6 6
```

```
N=7
for i in 1:6
    count=0
    for j in 1:10
        if M[i,j]==N
            count+=1
        end
    end
    if count==2
        println(i)
    end
end
```

```
5
6
```

```
s=[]
for i in 1:10
    count=0
    for j in 1:6
        count+=M[j,i]
    end
    append!(s, count)
end
for i in 1:9
    for j in i+1:10
        if s[i]+s[j]>70
            println(i, " ", j)
        end
    end
end
```

```
1 10
5 10
6 10
8 10
```

```
sum1=0
for i in 1:20
    for j in 1:5
        sum1+=(i^4)/(3+j)
    end
end
sum1
```

```
639215.2833333334
```

```
sum2=0
for i in 1:20
    for j in 1:5
        sum2+=(i^4)/(3+i*j)
    end
end
sum2
```

```
89912.02146097136
```

Выводы

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