РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Компьютерный практикум

по математическому моделированию

Студент: Яссин Мохамад Аламин

Группа: НКНбд – 01-20

МОСКВА

2023 г.

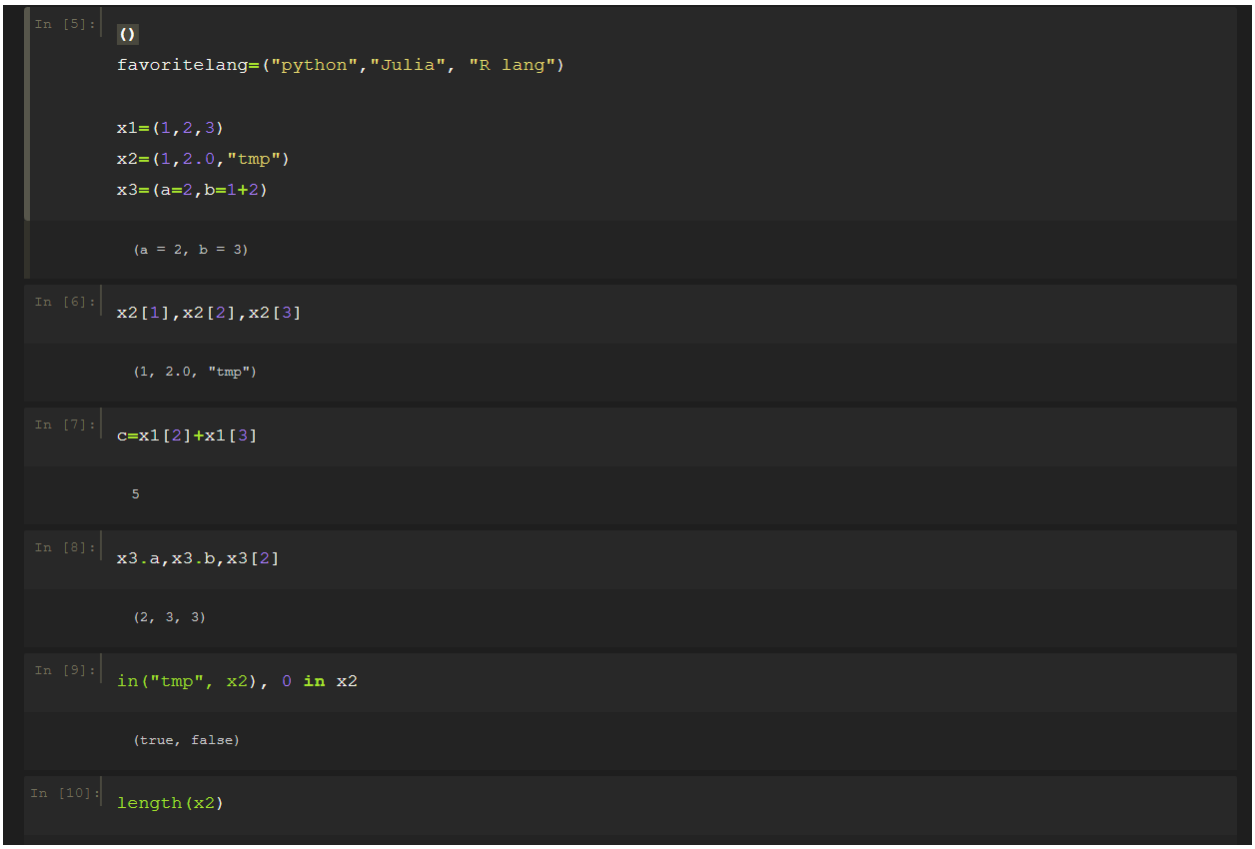
Постановка задачи

Основная цель работы — изучить несколько структур данных, реализованных в Julia, научиться применять их и операции над ними для решения задач.

1. Используя Jupyter Lab, повторите примеры
2. Выполните задания для самостоятельной работы.

Выполнение работы

1. Повторение примером



In [12]:
```
phonebook = Dict("Иванов И.И." => ("867-5309","333-5544"),"Бухгалтерия" =>"555-2368")
keys(phonebook)
```

```
KeySet for a Dict{String, Any} with 2 entries. Keys:
  "Бухгалтерия"
  "Иванов И.И."
```

In [13]:
```
values(phonebook)
```

```
ValueIterator for a Dict{String, Any} with 2 entries. Values:
  "555-2368"
  ("867-5309", "333-5544")
```

In [14]:
```
pairs(phonebook)
```

```
Dict{String, Any} with 2 entries:
  "Бухгалтерия" => "555-2368"
  "Иванов И.И." => ("867-5309", "333-5544")
```

In [15]:
```
haskey(phonebook,"Иванов И.И.")
```

```
true
```

In [16]:
```
phonebook["Сидоров П.С."]="555-3344"
```

```
"555-3344"
```

In [17]:
```
pop!(phonebook, "Иванов И.И.")
```

In [17]:
```
pop!(phonebook, "Иванов И.И.")
```

```
("867-5309", "333-5544")
```

In [18]:
```
a= Dict("foo"=>0.0,"bar"=>42.0);
b= Dict("baz"=>17, "bar"=>13.0);
merge(a,b),merge(b,a)
```

```
(Dict{String, Real}("bar" => 13.0, "baz" => 17, "foo" => 0.0), Dict{String, Real}("bar" => 42.0, "baz" => 17, "foo" => 0.0))
```

In [20]:
```
A=Set([1,3,4,5])
B=Set("abrakadabra")

S1=Set([1,2])
S2=Set([3,4])
issetequal(S1,S2)
```

```
false
```

In [23]:
```
S3=Set([1,2,2,3,1,2,3,2,1]);
S4=Set([2,3,1])
issetequal(S3,S4)
```

```
true
```

In [24]:
```
C=union(S1,S2)
```

```
Set{Int64} with 4 elements:
```

In [24]:
```
C=union(S1,S2)
```

```
Set{Int64} with 4 elements:
  4
  2
  3
  1
```

In [25]:
```
D=intersect(S1,S3)
```

```
Set{Int64} with 2 elements:
  2
  1
```

In [26]:
```
E=setdiff(S3,S1)
```

```
Set{Int64} with 1 element:
  3
```

In [27]:
```
issubset(S1,S4)
```

```
true
```

In [28]:
```
push!(S4,99)
```

```
Set{Int64} with 4 elements:
  2
  99
  3
  1
```

In [30]:
```
pop!(S4)
```

```
2
```

In [34]:
```
empty_array_1=[]

a=[1,2,3] #Вектор столбец
b=[1 2 3] #Вектор строка
A=[[1,2,3];[4,5,6];[7,8,9]]
```

```
9-element Vector{Int64}:
 1
 2
 3
 4
 5
 6
 7
 8
 9
```

In [35]:
```
B=[[1 2 3];[4 5 6];[7 8 9]]
```

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

In [36]:
```
c=rand(1,8)
```

```
1×8 Matrix{Float64}:
 0.194204  0.362973  0.695503  0.175578  …  0.19675  0.328533  0.948333
```

In [37]:
```
c=rand(2,3)
```

```
2×3 Matrix{Float64}:
 0.32506   0.989055  0.622885
 0.816279  0.175961  0.757636
```

In [39]:
```
roots=[sqrt(i) for i in 1:10]
```

```
10-element Vector{Float64}:
 1.0
 1.4142135623730951
 1.7320508075688772
 2.0
 2.23606797749979
 2.449489742783178
 2.6457513110645907
 2.8284271247461903
 3.0
 3.1622776601683795
```

In [40]:
```
ar_1=[3*i^2 for i in 1:2:9]
```

```
5-element Vector{Int64}:
   3
  27
  75
 147
 243
```

In [42]:
```
ar_2=[i^2 for i=1:10 if(i^2%5!=0&&i^2%4!=0)]
```

```
4-element Vector{Int64}:
  1
  9
 49
```

In [43]:
```
ones(5)
```

```
5-element Vector{Float64}:
 1.0
 1.0
 1.0
 1.0
 1.0
```

In [44]:
```
ones(2,3)
```

```
2×3 Matrix{Float64}:
 1.0  1.0  1.0
 1.0  1.0  1.0
```

In [45]:
```
zeros(4)
```

```
4-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
```

In [46]:
```
fill(3.5,(3,2))
```

```
3×2 Matrix{Float64}:
 3.5  3.5
 3.5  3.5
 3.5  3.5
```

In [47]:
```
repeat([1,2],3,3)
repeat([1,2],3,3)
```

In [47]:
```
repeat([1,2],3,3)
```

```
repeat([1,2],3,3)
```

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

In [48]:
```
a=collect(1:12)
```

```
b=reshape(a,(2,6))
```

```
2×6 Matrix{Int64}:
 1  3  5  7   9  11
 2  4  6  8  10  12
```

In [49]:
```
b'
```

```
6×2 adjoint(::Matrix{Int64}) with eltype Int64:
  1   2
  3   4
  5   6
  7   8
  9  10
 11  12
```

In [50]:
```
c=transpose(b)
```

```
6×2 transpose(::Matrix{Int64}) with eltype Int64:
  1   2
  3   4
  5   6
```

In [51]:
```
ar=rand(10:20,10,5)
```

```
10×5 Matrix{Int64}:
 15  12  15  17  20
 16  17  11  13  14
 12  17  11  17  13
 15  12  17  11  11
 10  18  15  10  20
 10  14  20  19  11
 14  20  15  17  14
 12  11  19  15  14
 18  11  15  17  15
 12  11  13  11  14
```

In [52]:
```
ar[:,2]
```

```
10-element Vector{Int64}:
 12
 17
 17
 12
 18
 14
 20
 11
 11
 11
```

In [53]:
```
ar[:,[2,5]]
```

```
10×2 Matrix{Int64}:
 12  20
 17  14
 17  13
 12  11
 18  20
 14  11
```

In [54]:
```
ar[:,2:4]
```

```
10×3 Matrix{Int64}:
 12  15  17
 17  11  13
 17  11  17
 12  17  11
 18  15  10
 14  20  19
 20  15  17
 11  19  15
 11  15  17
 11  13  11
```

In [55]:
```
ar[[2,4,6],[1,5]]
```

```
3×2 Matrix{Int64}:
 16  14
 15  11
 10  11
```

In [56]:
```
ar[1,3:end]
```

```
3-element Vector{Int64}:
 15
 17
 20
```

In [57]:
```
sort(ar, dims=1)
```

```
10×5 Matrix{Int64}:
 10  11  11  10  11
 10  11  11  11  11
 12  11  13  11  13
 12  12  15  13  14
```

In [58]:
```
sort(ar,dims=2)
```

```
10×5 Matrix{Int64}:
 12  15  15  17  20
 11  13  14  16  17
 11  12  13  17  17
 11  11  12  15  17
 10  10  15  18  20
 10  11  14  19  20
 14  14  15  17  20
 11  12  14  15  19
 11  15  15  17  18
 11  11  12  13  14
```

In [59]:
```
ar.>14
```

```
10×5 BitMatrix:
 1  0  1  1  1
 1  1  0  0  0
 0  1  0  1  0
 1  0  1  0  0
 0  1  1  0  1
 0  0  1  1  0
 0  1  1  1  0
 0  0  1  1  0
 1  0  1  1  1
 0  0  0  0  0
```

In [60]:
```
findall(ar.>14)
```

```
24-element Vector{CartesianIndex{2}}:
 CartesianIndex(1, 1)
 CartesianIndex(2, 1)
 CartesianIndex(4, 1)
 CartesianIndex(9, 1)
 CartesianIndex(2, 2)
 CartesianIndex(3, 2)
```

2.Выполните задания для самостоятельной работы

In [62]:
```
A=Set([0,3,4,9])
B=Set([1,3,4,7])
C=Set([0,1,2,4,7,8,9])
P=intersect(A,B)
P=union(P,A)
P=intersect(P,B)
P=union(P,A)
P=intersect(P,C)
P=union(P,B)
P=intersect(P,C)
P
```

```
Set{Int64} with 5 elements:
  0
  4
  7
  9
  1
```

In [63]:
```
push!(A,4)
```

```
Set{Int64} with 4 elements:
  0
  4
  9
  3
```

In [64]:
```
pop!(A)
```

```
0
```

In [65]:
```
P1=setdiff(A,P)
```

```
Set{Int64} with 1 element:
  3
```

In [66]:
```
a=fill(0.0,9)
```

```
9-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
```

In [67]:
```
a_ran=rand(4,6)
```

```
4×6 Matrix{Float64}:
 0.0364148  0.274346  0.544914  0.670133  0.434122   0.906561
 0.330478   0.326893  0.855339  0.599666  0.266779   0.918607
 0.0739754  0.107811  0.379759  0.425155  0.0679301  0.594084
 0.333439   0.693183  0.187474  0.737073  0.392818   0.382271
```

In [68]:
```
a_col=collect(1:1:10)
```

```
10-element Vector{Int64}:
  1
  2
  3
  4
```

In [69]:
```
N = 30
A1=[i for i in 1:N]
```

```
30-element Vector{Int64}:
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
  ⋮
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
```

In [70]:
```
A2=[N-i for i in 0:N-1]
```

```
30-element Vector{Int64}:
 30
 29
 28
 27
 26
 25
 24
```

```
23
22
21
⋮
10
9
8
7
6
5
4
3
2
1
```

In [71]:

```
A3=vcat(A1,A2)
```

```
60-element Vector{Int64}:
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
 ⋮
10
 9
 8
 7
 6
 5
 4
 3
 2
 1
```

In [72]:
```
tmp=[4,6,3]
```

```
3-element Vector{Int64}:
 4
 6
 3
```

In [74]:
```
tmp10=repeat([tmp[1]],10)
```

```
10-element Vector{Int64}:
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
```

In [76]:
```
tmp_10=repeat(tmp,10)
```

```
30-element Vector{Int64}:
 4
 6
 3
 4
 6
 3
 4
 6
 3
 4
 ⋮
 3
 4
```

```
 6
 3
 4
 6
 3
 4
 6
 3
```

In [78]:

```
#3.7
tmp11_10_10=vcat(repeat([tmp[1]],11),repeat([tmp[2]],10), repeat([tmp[3]],10))
```

```
31-element Vector{Int64}:
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
 :
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
```

In [79]:

```
#3.8
tmp_10_20_30=vcat(repeat([tmp[1]],10),repeat([tmp[2]],20),repeat([tmp[3]],30))
```

```
60-element Vector{Int64}:
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
 ⋮
 3
 3
 3
 3
 3
 3
 3
 3
 3
 3
```

In [80]:

```
#3.9 2^tmp[i]
tmp2=repeat([2^tmp[i] for i in 1:3],4)
```

```
12-element Vector{Int64}:
 16
 64
  8
 16
 64
  8
 16
 64
  8
 16
 64
  8
```

In [82]:
```
#3.10
using Statistics
tmp6=[y=exp(x)*cos(x) for x=3.0:0.1:6]
mean(tmp6)
```

```
53.11374594642971
```

In [83]:
```
#3.11
A4=[[0.1^j,0.2^j] for i=3:3:36 for j=1:3:34]
```

```
144-element Vector{Vector{Float64}}:
 [0.1, 0.2]
 [0.00010000000000000002, 0.0016000000000000003]
 [1.0000000000000005e-7, 1.2800000000000006e-5]
 [1.0000000000000006e-10, 1.0240000000000006e-7]
 [1.0000000000000007e-13, 8.192000000000005e-10]
 [1.0000000000000008e-16, 6.553600000000055e-12]
 [1.000000000000001e-19, 5.2428800000000056e-14]
 [1.0000000000000012e-22, 4.194304000000005e-16]
 [1.0000000000000013e-25, 3.3554432000000044e-18]
 [1.0000000000000015e-28, 2.684354560000004e-20]
 ⋮
 [1.0000000000000005e-7, 1.2800000000000006e-5]
 [1.0000000000000006e-10, 1.0240000000000006e-7]
 [1.0000000000000007e-13, 8.192000000000005e-10]
 [1.0000000000000008e-16, 6.553600000000055e-12]
 [1.000000000000001e-19, 5.2428800000000056e-14]
 [1.0000000000000012e-22, 4.194304000000005e-16]
 [1.0000000000000013e-25, 3.3554432000000044e-18]
 [1.0000000000000015e-28, 2.684354560000004e-20]
 [1.0000000000000018e-31, 2.147483648000004e-22]
 [1.000000000000002e-34, 1.7179869184000035e-24]
```

In [84]:
```
A5=[(2^i)/i for i=1:1:25]
```

In [84]:
```
A5=[(2^i)/i for i=1:1:25]
```

```
25-element Vector{Float64}:
      2.0
      2.0
      2.6666666666666665
      4.0
      6.4
     10.666666666666666
     18.285714285714285
     32.0
     56.888888888888886
    102.4
      ⋮
   4096.0
   7710.117647058823
  14563.555555555555
  27594.105263157893
  52428.8
  99864.38095238095
 190650.18181818182
 364722.0869565217
 699050.6666666666
      1.34217728e6
```

In [85]:
```
num=collect(1:30)
fn=repeat("fn",30)
map(string,fn,num)
```

```
30-element Vector{String}:
 "f1"
 "n2"
 "f3"
 "n4"
 "f5"
 "n6"
```

In [86]:
```
x=rand(0:999,250)
y=rand(0:999,250)
y[2:250]-x[1:249]
```

```
249-element Vector{Int64}:
 -168
 -331
  419
 -624
 -285
 -151
   12
 -552
  154
 -232
    ⋮
 -194
  460
  503
 -915
 -544
 -332
  132
 -427
  765
 -174
```

In [87]:
```
x[1:248]+2*x[2:249]-x[3:250]
```

```
248-element Vector{Int64}:
 1711
  446
 1925
 1115
  945
  -38
 1183
 1750
```

In [88]:
```
[sin(y[i])/cos(x[i+1]) for i in 1:249]
```

```
249-element Vector{Float64}:
  -0.6098979960447871
   1.2712944703937885
   0.3357962226863886
   2.1787603033568312
  -1.242220598338491
   0.9115819237549394
  -0.9296168651376182
   0.6079362367430735
  -0.7446920544479695
  -0.7942856275348047
   ⋮
 -19.41844662254851
  -0.29134180082907657
  -0.4069195422061545
   1.008470185548576
  -1.7215255000600915
  -3.063879681168841
  -0.039718965519796076
  -1.0451687029683614
  -0.7508705992882289
  -1.1291452155950994
```

In [89]:
```
sum([exp(-x[i+1])/(x[i]+10) for i in 1:249])
```

```
0.00010016486699746194
```

In [90]:
```
yf=findall(y->y>600,y)
yr=[]
for i in 1:length(yf)
    push!(yr,y[yf[i]])
end
```

In [91]:
```
yr
```

```
116-element Vector{Any}:
 715
 761
 947
 883
 883
 695
 887
 696
 663
 941
   ⋮
 654
 760
 816
 813
 946
 616
 896
 892
 834
 891
```

In [92]:
```
xr=[]
for i in 1:length(yf)
    push!(xr,x[yf[i]])
end
xr
```

```
116-element Vector{Any}:
 958
 271
 425
 110
 465
```

In [93]:
```
av_x=mean(x)
[abs(x[i]-av_x)^0.5 for i in 1:250]
```

```
250-element Vector{Float64}:
 11.680068492949859
 17.104853112494126
 14.94737435137021
 20.94220618750565
 15.24545834010903
 10.120474297185877
 18.53170256614324
 16.95806592745765
  9.358204956079987
 15.761472012473961
  ⋮
  9.350080213559666
 13.72676218195682
 21.876379956473603
 15.217621364720573
 15.986744509123803
 15.634065370210013
 17.075596622080297
 19.834918704143963
  4.407266726668579
 14.779174537165463
```

In [94]:
```
ymax=maximum(y)
ym200=findall(y->(abs(y-ymax)<=200),y)
length(ym200)
```

```
60
```

In [95]:
```
xchet=findall(x->(x%2==0),x)
```

```
122-element Vector{Int64}:
```

```
In [95]: xchet=findall(x->(x%2==0),x)
```

```
122-element Vector{Int64}:
   2
   3
   4
   7
  15
  16
  17
  20
  23
  24
   ⋮
 228
 230
 232
 233
 235
 237
 241
 243
 248
 249
```

```
In [96]: xnechet=findall(x->(x%2!=0),x)
```

```
128-element Vector{Int64}:
   1
   5
   6
   8
   9
  10
  11
  12
  13
  14
   ⋮
```

In [97]:
```
x_7=findall(x->(x%7==0),x)
length(x_7)
```

```
39
```

In [98]:
```
mp=sortperm(y)
x_sort=[]
for i in mp
    push!(x_sort,x[i])
end
x_sort
```

```
250-element Vector{Any}:
 417
 314
 912
 414
 367
 539
 400
 432
 793
 906
   ⋮
 877
 377
 426
 661
 999
 661
 966
 204
 392
  51
```

In [99]:
```
x_top=sort(x,rev=true)
for i in 1:10
    println(x_top[i])
end
```

```
999
998
992
990
978
977
976
970
968
966
```

In [100]:
```
unique(x)
```

```
217-element Vector{Int64}:
 383
 812
 296
 958
 287
 417
 176
 807
 607
 271
   ⋮
 686
 293
 432
 998
 751
 275
 811
```

In [101]:
```
squares=[i^2 for i in 1:100]
```

```
100-element Vector{Int64}:
     1
     4
     9
    16
    25
    36
    49
    64
    81
   100
     ⋮
  8281
  8464
  8649
  8836
  9025
  9216
  9409
  9604
  9801
 10000
```

In [104]:
```
using Primes
myprimes=primes(999)
println(myprimes[89])
println(myprimes[89:99])
```

```
461
[461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]
```

In [105]:
```
su=[i^3+4*i^2 for i in 10:100]
sum(su)
```

In [106]:
```
su2=[(2^i)/i+(3^i)/(i*i) for i in 1:25]
sum(su2)
```

```
2.1291704368143802e9
```

In [107]:
```
x=collect(2:2:38)
sum(cumprod(x./(x.+1)))
```

```
5.97634613789762
```

Выводы

Получены навыки работы со структурами данных в Julia