

# Лабораторная работа № 2. Julia. Структуры данных

Компьютерный практикум по статистическому анализу данных

---

Демидова Е. А.

16 ноября 2024

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Демидова Екатерина Алексеевна
- студентка группы НКНбд-01-21
- Российский университет дружбы народов
- <https://github.com/eademidova>



## Введение

---

## Цель работы

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

## Задачи

1. Используя Jupyter Lab, повторите примеры из раздела 2.2.
2. Выполните задания для самостоятельной работы (раздел 2.4).

## Выполнение лабораторной работы

---

```
[6] a = [1 2; 3 4]
    b = [1 1]

... 1x2 Matrix{Int64}:
    1 1

[7] a*b*2

... 2x1 Matrix{Int64}:
    6
   14

[8] x = (a = 2, b = "3")
    println(x.a, x[1])
    print(in("3", x))

... 22
    true

[9] phonebook = Dict{String{UTF8}, Tuple{String{UTF8}, String{UTF8}}} => {"867-5309", "333-5544"}, {"Бухгалтерия" => "555-2368"}

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

[10] pop!(phonebook, "Иванов И.И.")

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

[11] phonebook

... Dict{String, Any} with 1 entry:
    "Бухгалтерия" => "555-2368"

[12] a = Dict{String{UTF8}, Real} => {"foo" => 42.0};
    b = Dict{String{UTF8}, Real} => {"bar" => 13.0};
    print(merge(a, b), merge(b, a))

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

Рис. 1: Примеры. Кортежи

```
> S1 = Set([1,2]);
S2 = Set([3,4]);
println("Равенство множеств")
println(isequal(S1,S2))
S4 = Set([1,2,2,3,1,2,3,2,1]);
S3 = Set([2,3,1])
println("Объединение, пересечение и разность")
### объединение множеств:
C = union(S1,S2)
println(C)
### пересечение множеств:
D = intersect(S1,S3)
println(D)
### разность множеств:
E = setdiff(S3,S1)
println(E)
### проверка вхождения элементов одного множества в другое:
println(issubset(S1,S4))
### добавление элемента в множество:
println("Добавление и удаление элемента")
println(S4)
push!(S4, 99)
println(S4)
### удаление последнего элемента множества:
pop!(S4)
println(S4)
```

[10] ✓ 0.7s

```
--- Равенство множеств
false
Объединение, пересечение и разность
Set{4, 2, 3, 1}
Set{2, 1}
Set{3}
true
Добавление и удаление элемента
Set{2, 3, 1}
Set{2, 99, 3, 1}
Set{99, 3, 1}
```

Рис. 2: Примеры. Множества



```
D> A = rand(4,3,2)
[10]
4x3x2 Array{Float64, 3}:
[:, :, 1] =
 0.53715  0.0253307  0.557304
 0.392261  0.613429  0.321627
 0.230469  0.612413  0.746013
 0.973464  0.303498  0.595916

[:, :, 2] =
 0.122025  0.264099  0.171144
 0.963755  0.840615  0.026727
 0.00105212  0.443124  0.609205
 0.560781  0.983995  0.443232

A[1, 2, :]  
[10]
2-element Vector{Float64}:
 0.025330710121084866
 0.26409877227643766

ar_2=[1^2 for i=1:10 if (1^2%5!=0 && 1^2%4!=0)]  
[10]
4-element Vector{Int64}:
 1
 9
 49
 81

D> ar = rand(10;20, 10, 5)
[10]
10x5 Matrix{Int64}:
 17 17 16 20 14
 12 13 16 18 20
 18 10 16 17 13
 11 11 15 13 10
 16 10 16 13 15
```

Рис. 3: Примеры. Массивы

```
ar[[2, 4, 6], [1, 5]]
[17]
--- 3x2 Matrix[Int64]:
 12 20
 11 10
 18 14

ar[1, 3:end]
[18]
--- 3-element Vector{Int64}:
 10
 20
 14

D> sort(ar, dims=1)
[19]
--- 10x5 Matrix{Int64}:
 11 10 10 13 10
 11 10 10 13 10
 11 11 10 13 13
 12 11 13 14 14
 12 13 15 17 14
 16 14 16 17 15
 17 15 19 18 18
 18 17 19 19 18
 18 17 20 20 19
 20 20 20 20 20

sort(ar, dims=2)
[20]
--- 10x5 Matrix{Int64}:
 10 14 17 17 20
 10 12 13 18 20
 10 13 17 18 18
 10 11 11 13 15
 10 13 15 16 19
 13 14 14 18 20
```

Рис. 4: Примеры. Массивы

```
findall(ar .> 14)

[21]

... 26-element Vector{CartesianIndex{2}}:
 CartesianIndex{1, 1}
 CartesianIndex{3, 1}
 CartesianIndex{5, 1}
 CartesianIndex{6, 1}
 CartesianIndex{10, 1}
 CartesianIndex{1, 2}
 CartesianIndex{7, 2}
 CartesianIndex{8, 2}
 CartesianIndex{10, 2}
 CartesianIndex{3, 3}
 CartesianIndex{4, 3}
 CartesianIndex{5, 3}
 CartesianIndex{7, 3}
 CartesianIndex{8, 3}
 CartesianIndex{9, 3}
 CartesianIndex{1, 4}
 CartesianIndex{2, 4}
 CartesianIndex{3, 4}
 CartesianIndex{6, 4}
 CartesianIndex{7, 4}
 CartesianIndex{9, 4}
 CartesianIndex{2, 5}
 CartesianIndex{5, 5}
 CartesianIndex{8, 5}
 CartesianIndex{9, 5}
 CartesianIndex{10, 5}
```

Рис. 5: Примеры. Массивы

```
▷ #1, 2
A = Set([0,3,4,9]);
B = Set([1,3,4,7]);
C = Set([0,1,2,4,7,8,9]);
P = union(intersect(A, B), intersect(A, B), intersect(A, C), intersect(B, C))

[31] ✓ 0.0s

... Set{Int64} with 6 elements:
0
4
7
9
3
1

▷ setdiff(C,B)

[32] ✓ 0.0s

... Set{Int64} with 4 elements:
0
2
9
8
```

Рис. 6: Задание 1,2

[illegible]

Рис. 7: Задание 3

## Самостоятельные задания

Nº 3.10

```
function f(x)
    exp(x)*cos(x)
end
x = 3:0.1:6
y_x = [f(i) for i in x]
sum(y_x)/sizeof(y_x)
```

303

N23.11

```

p(3:3:36,1:3:34) do i, j
    [0.2^i, 0.2^j]
end

```

154

[illegible]

24

```
x1 = [i for i in 3:3:36]
x2 = [i for i in 1:3:34]
[[0.1i, 0.2j] for (i,j) in zip(x1,x2)]
```

100

[illegible]

Рис. 8: Задание 3

# Самостоятельные задания

```
NP 3.12
вектор с элементами  $\frac{2^i}{i}, i = 1, 2, \dots, M, M = 25$ 

[2^i/i for i in 1:25]

25-element Vector{Float64}:
 2.0
 2.0
 1.6666666666666665
 1.5
 1.4
 1.3333333333333333
 1.25
 1.176470588235294
 1.1111111111111112
 1.0588235294117647
 1.016260162601626
 0.9837430162601626
 0.9523809523809523
 0.9230769230769231
 0.8956521739130435
 0.8695652173913043
 0.8448275862068966
 0.8214285714285714
 0.79921875
 0.778125
 0.7580972365591398
 0.7391304347826087
 0.7211538461538461
 0.7041339370957895
 0.688034188034188
 0.672766016979133
 0.6582278488757223
 0.6443283582089552
 0.6310162601626016
 0.6182835820895522
 0.6061178846153846
 0.5945154545454545
 0.5834720588235294
 0.572972972972973
 0.563013698630137
 0.5535927439822426
 0.5447018518518519
 0.5363265306122449
 0.5284552845528455
 0.5210843478260869
 0.5142177286457143
 0.5078431372549019
 0.5019327731092436
 0.4964789328064516
 0.4914754098360653
 0.4869142857142857
 0.4826923076923077
 0.4788065843621445
 0.4752578125
 0.47205128205128205
 0.4691735557349129
 0.4666153846153846
 0.4643734413987234
 0.4623381343283582
 0.4605005351190476
 0.4588607595628415
 0.4574193548387097
 0.4560773109243697
 0.4548354251901961
 0.4536937095789474
 0.4526521739130435
 0.4517108432835821
 0.4508695652173913
 0.4501276595744681
 0.4494852941176471
 0.4489423076923077
 0.4484985294117647
 0.44815454545454545
 0.44791045454545455
 0.4476662601626016
 0.4475220588235294
 0.4473778488757223
 0.4472336559139872
 0.4470904761904762
 0.44694730162601626
 0.4468042307692308
 0.4466612601626016
 0.44651839370957895
 0.4463756329832437
 0.4462330820895522
 0.446090639810427
 0.4459483174603174
 0.4458061153846154
 0.4456640243902439
 0.445522043956044
 0.4453801735668784
 0.4452384126984127
 0.4450967619047619
 0.4449552205882353
 0.4448137894736842
 0.4446724791813472
 0.444531338028169
 0.4443903566433566
 0.44424954545454545
 0.4441088956217177
 0.4439684063426673
 0.4438279772627286
 0.4436877075396825
 0.443547697266758
 0.4434079464556962
 0.4432683552066116
 0.4431289235294118
 0.4429896515463918
 0.4428505402439024
 0.4427115891304348
 0.4425727982089552
 0.4424341675957895
 0.4422956962068966
 0.4421573851351351
 0.4420192242553191
 0.4418812233766234
 0.4417433829832437
 0.4416056930820896
 0.4414681536735009
 0.4413307736842105
 0.4411935432098765
 0.4410564622513812
 0.4409195308208955
 0.4407827489130435
 0.4406461176470588
 0.4405096451612903
 0.4403733243902439
 0.4402371530612244
 0.4401011316528926
 0.4399652603174603
 0.4398295390243902
 0.4396939680854166
 0.4395585476603906
 0.4394232777777778
 0.4392881481481481
 0.4391531684210526
 0.439018338028169
 0.4388836572972973
 0.4387491260162602
 0.4386147446031746
 0.4384805132098765
 0.438346429832437
 0.4382124964152344
 0.4380787130434783
 0.437945079630137
 0.4378115962278489
 0.4376782628260869
 0.4375450793253968
 0.4374120459243697
 0.4372791625938268
 0.4371464292633587
 0.4370138459328358
 0.4368814126033135
 0.4367491292727913
 0.4366169959422691
 0.436485022611746
 0.4363532092812237
 0.4362215459507015
 0.4360900326201792
 0.435958669289657
 0.4358274559591347
 0.4356963926286124
 0.4355654792980901
 0.4354347159675678
 0.4353041026370455
 0.4351736393065232
 0.4350433259760009
 0.4349131626454786
 0.4347831493149563
 0.434653285984434
 0.4345235726539117
 0.4343940093233894
 0.4342645959928671
 0.4341353326623448
 0.4340062193318225
 0.4338772559993002
 0.4337484426667779
 0.4336197793342556
 0.4334912660017333
 0.433362902669211
 0.4332346893366887
 0.4331066259991664
 0.4329788126666441
 0.4328511493341218
 0.4327236360015995
 0.4325962726690772
 0.4324690593365549
 0.4323419959990326
 0.4322150826665103
 0.432088319333988
 0.4319617060014657
 0.4318352426689434
 0.4317089293364211
 0.4315827660038988
 0.4314567526713765
 0.4313308893388542
 0.4312051760063319
 0.4310796126738096
 0.4309541993412873
 0.430828936008765
 0.4307038226762427
 0.4305788593437204
 0.4304540460111981
 0.4303293826786758
 0.4302048693461535
 0.4300804960136312
 0.4299562726811089
 0.4298321993485866
 0.4297082760160643
 0.429584502683542
 0.4294608793510197
 0.4293374060184974
 0.4292140826859751
 0.4290909093534528
 0.4289678860209305
 0.4288450126884082
 0.4287222893558859
 0.4286007160233636
 0.4284792926908413
 0.428358019358319
 0.4282368960257967
 0.4281159226932744
 0.4279950993607521
 0.4278744260282298
 0.4277539026957075
 0.4276335293631852
 0.4275133060306629
 0.4273932326981406
 0.4272733093656183
 0.427153536033096
 0.4270339127005737
 0.4269144393680514
 0.4267951160355291
 0.4266759427030068
 0.4265569193704845
 0.4264380460379622
 0.42631932270544
 0.4262007493729177
 0.4260823260403954
 0.4259640527078731
 0.4258459293753508
 0.4257279560428285
 0.4256101327103062
 0.4254924593777839
 0.4253749360452616
 0.4252575627127393
 0.425140339380217
 0.4250232660476947
 0.4249063427151724
 0.4247895693826501
 0.4246729460501278
 0.4245564727176055
 0.4244401403850832
 0.4243239580525609
 0.4242079257200386
 0.4240930433875163
 0.423978311054994
 0.4238637287224717
 0.4237492963899494
 0.4236350140574271
 0.4235208817249048
 0.4234068993923825
 0.4232930670598602
 0.4231793847273379
 0.4230658523948156
 0.4229524700622933
 0.422839237729771
 0.4227261553972487
 0.4226132230647264
 0.4225004407322041
 0.4223878084006818
 0.4222753260681595
 0.4221629937356372
 0.4220508114031149
 0.4219387790705926
 0.4218268967380703
 0.421715164405548
 0.4216035820730257
 0.4214921497405034
 0.4213808674079811
 0.4212697350754588
 0.4211587527429365
 0.4210479204104142
 0.4209372380778919
 0.4208267057453696
 0.4207163234128473
 0.420606091080325
 0.4204960087478027
 0.4203860764152804
 0.4202762940827581
 0.4201666617502358
 0.4200572794177135
 0.4199480470851912
 0.4198389647526689
 0.4197299324201466
 0.4196210500876243
 0.419512317755102
 0.4194037354225797
 0.4192953030900574
 0.4191870207575351
 0.4190788884250128
 0.4189709060924905
 0.4188630737599682
 0.4187553914274459
 0.4186478590949236
 0.4185404767624013
 0.418433244429879
 0.4183261620973567
 0.4182192297648344
 0.4181124474323121
 0.4180058150997898
 0.4178993327672675
 0.4177930004347452
 0.4176868181022229
 0.4175807857697006
 0.4174749034371783
 0.417369171104656
 0.4172635887721337
 0.4171581564396114
 0.4170528741070891
 0.4169477417745668
 0.4168427594420445
 0.4167379271095222
 0.4166332447770009
 0.4165287124444786
 0.4164243301119563
 0.416320097779434
 0.4162159154469117
 0.4161117831143894
 0.4160077907818671
 0.4159039384493448
 0.4158002261168225
 0.4156966537843002
 0.4155932214517779
 0.4154899291192556
 0.4153867767867333
 0.415283764454211
 0.4151808921216887
 0.4150781597891664
 0.4149755674566441
 0.4148731151241218
 0.4147708027915995
 0.4146686304590772
 0.4145665981265549
 0.4144647057940326
 0.4143629534615103
 0.414261341128988
 0.4141598687964657
 0.4140585364639434
 0.4139573441314211
 0.4138562917988988
 0.4137553794663765
 0.4136546071338542
 0.4135539748013319
 0.4134534824688096
 0.4133531301362873
 0.413252917803765
 0.4131528454712427
 0.4130529131387204
 0.4129531208061981
 0.4128534684736758
 0.4127539561411535
 0.4126545838086312
 0.4125553514761089
 0.4124562591435866
 0.4123573068110643
 0.412258494478542
 0.4121598221460197
 0.4120612898134974
 0.4119628974809751
 0.4118646451484528
 0.4117665328159305
 0.4116685604834082
 0.4115707281508859
 0.4114730358183636
 0.4113754834858413
 0.411278071153319
 0.4111807988207967
 0.4110836664882744
 0.4109866741557521
 0.4108898218232298
 0.4107931094907075
 0.4106965371581852
 0.4105991048256629
 0.4105018124931406
 0.4104046601606183
 0.410307647828096
 0.4102107754955737
 0.4101139431630514
 0.4100172508305291
 0.4099206984980068
 0.4098242861654845
 0.4097279138329622
 0.40963158150044
 0.4095352891679177
 0.4094390368353954
 0.4093428245028731
 0.4092466521703508
 0.4091505198378285
 0.4090544275053062
 0.4089583751727839
 0.4088623628402616
 0.4087663905077393
 0.408670458175217
 0.4085745658426947
 0.4084787135101724
 0.4083829011776501
 0.4082871288451278
 0.4081913965126055
 0.4080957041800832
 0.4080001518475609
 0.4079046395150386
 0.4078091671825163
 0.407713734850094
 0.4076183425175717
 0.4075229901850494
 0.4074276778525271
 0.4073324055200048
 0.4072371731874825
 0.4071419808549602
 0.4070468285224379
 0.4069517161899156
 0.4068566438573933
 0.406761611524871
 0.4066666191923487
 0.4065716668598264
 0.4064767545273041
 0.4063818821947818
 0.4062870498622595
 0.4061922575297372
 0.4060975051972149
 0.4060027928646926
 0.4059081205321703
 0.405813488199648
 0.4057188958671257
 0.4056243435346034
 0.4055298312020811
 0.4054353588695588
 0.4053409265370365
 0.4052465342045142
 0.4051521818719919
 0.4050578695404696
 0.4049635972079473
 0.404869364875425
 0.4047751725429027
 0.4046810202103804
 0.4045869078778581
 0.4044928355453358
 0.4043988032128135
 0.4043048108802912
 0.4042108585477689
 0.4041169462152466
 0.4040230738827243
 0.403929241550202
 0.4038354492176797
 0.4037416968851574
 0.4036479845526351
 0.4035543122201128
 0.4034606798875905
 0.4033670875550682
 0.4032735352225459
 0.4031799228900236
 0.4030863505575013
 0.402992818224979
 0.4028993258924567
 0.4028058735599344
 0.4027124612274121
 0.4026190888948898
 0.4025257565623675
 0.4024324642298452
 0.4023392118973229
 0.4022459995648006
 0.4021528272322783
 0.402059694899756
 0.4019666025672337
 0.4018735502347114
 0.4017805379021891
 0.4016875655696668
 0.4015946332371445
 0.4015017409046222
 0.4014088885721009
 0.4013160762395786
 0.4012233039070563
 0.401130571574534
 0.4010378792420117
 0.4009452269094894
 0.4008526145769671
 0.4007600422444448
 0.4006675099119225
 0.4005750175794002
 0.4004825652468779
 0.4003901529143556
 0.4002977805818333
 0.400205448249311
 0.4001131559167887
 0.4000209035842664
 0.3999286912517441
 0.3998365189192218
 0.3997443865866995
 0.3996522942541772
 0.3995602419216549
 0.3994682295891326
 0.3993762572566103
 0.399284324924088
 0.3991924325915657
 0.3991005802590434
 0.3990087679265211
 0.3989169955940088
 0.3988252632614865
 0.3987335709289642
 0.3986419185964419
 0.3985503062639196
 0.3984587339313973
 0.398367201598875
 0.3982757092663527
 0.3981842569338304
 0.3980928446013081
 0.3980014722687858
 0.3979101399362635
 0.3978188476037412
 0.3977275952712189
 0.3976363829386966
 0.3975452106061743
 0.397454078273652
 0.3973629859411297
 0.3972719336086074
 0.3971809212760851
 0.3970899489435628
 0.3970000166110405
 0.3969091242785182
 0.3968182719459959
 0.3967274596134736
 0.3966366872809513
 0.396545954948429
 0.3964552626159067
 0.3963646102833844
 0.3962740079508621
 0.3961834456183398
 0.3960929232858175
 0.3960024409532952
 0.3959119986207729
 0.3958215962882506
 0.3957312339557283
 0.395640911623206
 0.3955506292906837
 0.3954603869581614
 0.3953701846256391
 0.3952800222931168
 0.3951898999605945
 0.3950998176280722
 0.3950097752955499
 0.3949197729630276
 0.3948298106305053
 0.394739888297983
 0.3946499059654607
 0.3945599636329384
 0.3944699613004161
 0.3943799989678938
 0.3942899766353715
 0.3941999943028492
 0.3941099519703269
 0.3940199496378046
 0.3939299873052823
 0.39383996497276
 0.3937499826402377
 0.3936599403077154
 0.3935699279751931
 0.3934799456426708
 0.3933899933101485
 0.3932999809776262
 0.3932099986451039
 0.3931199563125816
 0.3930299439800593
 0.392939951647537
 0.3928499793150147
 0.3927599269824924
 0.3926698946499701
 0.3925798823174478
 0.3924898899849255
 0.3923999176524032
 0.3923099653208809
 0.3922199329883586
 0.3921299206558363
 0.392039928323314
 0.3919499559907917
 0.3918599036582694
 0.3917698713257471
 0.3916798589932248
 0.3915898566607025
 0.3914998743281802
 0.3914099119956579
 0.3913199696631356
 0.3912299473306133
 0.391139945000091
 0.3910499626675687
 0.3909599903350464
 0.3908699279995241
 0.3907799756670018
 0.3906899333344795
 0.3905999009999572
 0.3905098786674349
 0.3904198663349126
 0.3903298640023903
 0.390239871669868
 0.3901498893373457
 0.3900599170048234
 0.3899699546723011
 0.3898799923397788
 0.3897899300072565
 0.3896999676747342
 0.3896099153422119
 0.3895198730096896
 0.3894298406771673
 0.389339818344645
 0.3892498060121227
 0.3891598036796004
 0.3890698113470781
 0.3889798290145558
 0.3888898566820335
 0.3887998943495112
 0.3887099420169889
 0.3886199996844666
 0.3885299673519443
 0.388439945019422
 0.3883499326868997
 0
```

# Самостоятельные задания

№ 3.14

векторы  $x = (x_1, x_2, \dots, x_n)$  и  $y = (y_1, y_2, \dots, y_n)$  целочисленного типа длины  $n = 250$  как случайные выборки из совокупности  $0, 1, \dots, 999$ ; на его основе:

- сформируйте вектор  $(y_2 - x_1, \dots, y_n - x_{n-1})$ ;
- сформируйте вектор  $(x_1 + 2x_2 - x_3, x_2 + 2x_3 - x_4, \dots, x_{n-2} + 2x_{n-1} - x_n)$ ;
- сформируйте вектор  $(\frac{\sin y_1}{\cos x_2}, \frac{\sin y_2}{\cos x_3}, \dots, \frac{\sin y_{n-1}}{\cos x_n})$ ;
- вычислите  $\sum_{i=1}^{n-1} \frac{e^{-x_{i+1}}}{x_i + 10}$ ;
- выберите элементы вектора  $y$ , значения которых больше 600, и выведите на экран; определите индексы этих элементов;
- определите значения вектора  $x$ , соответствующие значениям вектора  $y$ , значения которых больше 600 (под соответствием понимается расположение на аналогичных индексных позициях);
- сформируйте вектор  $(|x_1 - \bar{x}|^2, |x_2 - \bar{x}|^2, \dots, |x_n - \bar{x}|^2)$ , где  $\bar{x}$  обозначает среднее значение вектора  $x = (x_1, x_2, \dots, x_n)$ ;
- определите, сколько элементов вектора  $y$  отстоят от максимального значения не более, чем на 200;
- определите, сколько чётных и нечётных элементов вектора  $x$ ;
- определите, сколько элементов вектора  $x$  кратны 7;
- отсортируйте элементы вектора  $x$  в порядке возрастания элементов вектора  $y$ ;
- выведите элементы вектора  $x$ , которые входят в десятку наибольших ( $\text{top-10}$ )?
- сформируйте вектор, содержащий только уникальные (неповторяющиеся) элементы вектора  $x$ .

```
n = 250
# = [rand(0:999) for i=1:n];
y = [rand(0:999) for i=1:n];
using Statistics
```

(1)

D

```
tmp1 = [y[i+1]-x[i] for i=1:n-1];
tmp2 = [x[i]+2*x[i+1]-x[i+2] for i=1:n-2];
tmp3 = [sin(y[i])/cos(x[i+1]) for i=1:n-1];
tmp4 = sum([exp(-x[i+1])/x[i+10] for i=1:n-1]);
tmp5_ind = findall(y.>600);
tmp5_el = filter(x->x.600, y);
tmp6_1 = [x[i] for i=1:250 if (y[i]>600)];
tmp6_2 = [x[i] for i in tmp5_ind];
mean_x = mean(x);
tmp7 = [abs(x[i]-mean(x)) for i=1:n];
tmp8 = sum(x.>maximum(x)-200);
tmp9_odd = sum(map(isodd, x));
tmp9_even = sum(map(iseven, x));
tmp10 = x[sortperm(y)];
tmp11 = last(sort(x), 10);
tmp12 = collect(Set(x));
```

(2)

Рис. 10: Задание 3



# Самостоятельные задания

```
n = 250
x = [rand(0.999) for i=1:n]
y = [rand(0.999) for i=1:n]
using Statistics

tmp1 = [y[i]<-x[i] for i=1:n-1]
tmp2 = [x[i]+2*x[i+1]-x[i+2] for i=1:n-2]
tmp3 = [sin(y[i])/cos(x[i+1]) for i=1:n-1]
tmp4 = sum([exp(-x[i+1])/(x[i]+10) for i=1:n-1])
tmp5_ind = findall(y.>600)
tmp5_ind = filter(x->x.>400, y)
tmp6_1 = [x[i] for i=1:250 if(y[i]>600)]
tmp6_2 = [x[i] for i in tmp5_ind]
mean_x = mean(x)
tmp7 = [abs(x[i])-mean(x) for i=1:n]
tmp8 = sum(x.>maximum(x)-200)
tmp9_odd = sum(map(isodd, x))
tmp9_even = sum(map(iseven, x))
tmp10 = x[sortperm(y)]
tmp11 = last(sort(x),10)
tmp12 = collect(Set(x))
println(tmp1, "\n", tmp2, "\n", tmp3, "\n", tmp4, "\n", tmp5_ind, "\n", tmp5_ind)
println(tmp6_1, "\n", tmp6_2, "\n", mean_x, "\n", tmp7, "\n", tmp8, "\n", tmp9_odd, "\n", tmp9_even, "\n", tmp10, "\n", tmp11, "\n", tmp12)
```

```
[1] 113, 461, -279, 744, 900, -715, -649, -12, -563, 335, -715, 396, -568, -468, -696, 443, -60, 303, -387, -178, 40, 15, -127, -521, 731, -4, -163, 135, 315, 117, -432, 142, 86, -157, -324, -244, -199, -3, -6
-472, 1540, 1875, -833, 1147, 1876, 862, 1719, 318, 1487, 1324, 1168, 921, 1795, 1196, 1649, 388, 1264, 1286, 1659, 1694, 1916, 2652, 527, 471, 1372, 1482, 236, 1423, 1487, 1834, 1623, 1520, 1914, 1752, 13
1.823404836013771, -2.6023552934829266, 1.9585194294970354, -0.7995230125982875, -2.8379130511330083, 0.491344104119382, 0.40835999501368587, -0.8427366587019793, -1.284691048742813, 1.6127858680446270, 0.
3.814899582580715e-9
[1, 4, 5, 6, 13, 17, 18, 22, 23, 24, 26, 29, 31, 33, 34, 35, 36, 39, 47, 55, 62, 64, 66, 70, 71, 82, 83, 84, 92, 93, 98, 99, 104, 105, 108, 110, 111, 112, 117, 118, 122, 125, 131, 133, 134, 138, 139, 142, 1
1824, 700, 834, 984, 889, 955, 632, 748, 939, 817, 815, 674, 908, 880, 968, 618, 626, 988, 725, 902, 876, 924, 909, 899, 932, 768, 714, 949, 683, 748, 709, 616, 717, 819, 857, 616, 923, 912, 712, 696, 631,
1363, 99, 84, 601, 627, 692, 247, 924, 944, 686, 537, 383, 586, 880, 795, 950, 781, 724, 389, 278, 525, 252, 394, 811, 760, 538, 297, 962, 924, 780, 412, 813, 872, 358, 725, 259, 399, 387, 937, 645, 251, 46
1363, 99, 84, 891, 627, 692, 247, 924, 944, 896, 537, 383, 598, 880, 795, 950, 781, 724, 383, 278, 525, 252, 394, 811, 760, 538, 297, 962, 924, 780, 412, 813, 872, 758, 725, 259, 390, 387, 937, 645, 251, 46
514.288
1151.2679999999997, 442.2679999999997, 464.79200000000003, 424.2679999999997, 430.2679999999997, 376.79200000000003, 284.79200000000003, 61.2679999999997, 248.79200000000003, 254.2679999999997, 450.79
53
186
144
1427, 390, 299, 216, 537, 680, 560, 884, 927, 245, 686, 963, 927, 74, 317, 82, 579, 609, 453, 642, 169, 970, 605, 954, 669, 243, 626, 167, 864, 487, 969, 265, 872, 38, 311, 453, 96, 945, 437, 366, 658, 322,
1370, 972, 970, 979, 882, 983, 984, 989, 991, 994]
[402, 875, 950, 609, 66, 719, 724, 501, 234, 983, 871, 687, 251, 164, 547, 394, 601, 99, 763, 525, 405, 06, 148, 991, 287, 422, 658, 316, 551, 328, 692, 177, 410, 649, 730, 264, 945, 539, 478, 487, 82, 838,
```

Рис. 11: Задание 3

# Самостоятельные задания

№ 4

Создайте массив `squares`, в котором будут храниться квадраты всех целых чисел от 1 до 100.

```
squares = [i^2 for i = j:j+9 for j = 0:10:100]
```

11-element Vector{Vector{Int64}}:  
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]  
[100, 121, 144, 169, 196, 225, 256, 289, 324, 361]  
[400, 441, 484, 529, 576, 625, 676, 729, 784, 841]  
[900, 961, 1024, 1089, 1156, 1225, 1296, 1369, 1444, 1521]  
[1800, 1881, 1964, 2049, 2136, 2225, 2316, 2409, 2504, 2601]  
[2900, 2991, 3084, 3179, 3276, 3375, 3476, 3579, 3684, 3791]  
[3800, 3921, 4044, 4169, 4296, 4425, 4556, 4689, 4824, 4961]  
[4900, 5041, 5184, 5329, 5476, 5625, 5776, 5929, 6084, 6241]  
[6400, 6561, 6724, 6889, 7056, 7225, 7396, 7569, 7744, 7921]  
[8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409, 9604, 9801]  
[10000, 10201, 10404, 10609, 10816, 11025, 11236, 11449, 11664, 11881]

№ 5

Подключите пакет `Primes` (функции для вычисления простых чисел). Сгенерируйте массив `myprimes`, в котором будут храниться первые 168 простых чисел. Определите 89-е наименьшее простое число. Получите среза массива с 89-го до 99-го элемента включительно, содержащий наименьшие простые числа.

```
import Primes as pm  
  
myprimes = [pm.prime(i) for i=1:168];  
prime_89 = pm.prime(89);  
prime_89_to_99 = myprimes[89:99];
```

Рис. 12: Задание 4

```
import Primes as pm

[30] ✓ 0.0s Julia

> myprimes = [pm.prime(i) for i=1:100];
prime_89 = pm.prime(89);
prime_89_to_99 = myprimes[89:99];

println(myprimes, "\n", prime_89, "\n", prime_89_to_99)

[30] ✓ 0.0s Julia

... [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211,
461]
[461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]

[30] ✓ 0.1s Julia

tmp1 = sum([i^2+i^1/2 for i=1:100])
tmp2 = sum(2^i/i+2^i/i^2 for i=1:25)
tmp3 = 1
tmp4 = 1
println(tmp1, "\n", tmp2)
for i=2:2:38
    tmp+=i/(i+1)
    tmp3+=tmp
end
println(tmp, "\n", tmp3)
```

Рис. 13: Задание 5,6

## Выводы

---

В результате выполнения работы изучили несколько структур данных, реализованных в Julia, научились применять их и операции над ними для решения задач.

1. JuliaLang [Электронный ресурс]. 2024 JuliaLang.org contributors. URL: <https://julialang.org/> (дата обращения: 11.10.2024).
2. Julia 1.11 Documentation [Электронный ресурс]. 2024 JuliaLang.org contributors. URL: <https://docs.julialang.org/en/v1/> (дата обращения: 11.10.2024).