

Лабораторная работа № 2

Покрас Илья Михайлович

2023, Москва

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

```
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))
println("P equals: ")
println(P)

P equals:
Set([0, 4, 7, 9, 3, 1])
```

Рис. 1: Пункт 1

```
Set1 = Set(["Hello", "World", 1, 2, 3])
Set2 = Set([1, 2, 3, 4])
Set_bool1 = Set([true, false, false, true])
Set_bool2 = Set([true, false, false, true])

println("Set1:")
println(Set1)
println("\nSet2:")
println(Set2)

println("\n'setdiff' Operation:")
println(setdiff(Set1, Set2))
println("\n'intersect' Operation:")
println(intersect(Set1, Set2))

println("\n'issetequal' Operation:")
println(issetequal(Set_bool1, Set_bool2))

Set1:
Set{Any["Hello", 2, "World", 3, 1]}

Set2:
Set{[4, 2, 3, 1]}

'setdiff' Operation:
Set{Any["Hello", "World"]}

'intersect' Operation:
Set{Any[2, 3, 1]}

'issetequal' Operation:
true
```

Рис. 2: Пункт 2

```
N = 30
```

```
arr1 = collect(1:N)  
arr2 = vcat(1:N-1, N)
```

```
println("Vector1:")  
println(arr1)
```

```
println("\nVector1:")  
println(arr2)
```

Vector1:

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]

Vector1:

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]

Рис. 3: Пункт 3.1

```
N = 25

arr_reverse1 = collect(N:-1:1)
arr_reverse2 = reverse(collect(1:N))

println("Reversed Vector1:")
println(arr_reverse1)

println("\nReversed Vector2:")
println(arr_reverse2)

Reversed Vector1:
[25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

Reversed Vector2:
[25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Рис. 4: Пункт 3.2

```
N = 20

arr_half1 = collect(1:N-1)
arr_half2 = collect(N-1:-1:1)
arr_combined1 = vcat(arr_half1, N, arr_half2)
arr_combined2 = [1:N; N-1:-1:1]

println("Combined Vector1:")
println(arr_combined1)

println("\nCombined Vector2:")
println(arr_combined2)
```

Combined Vector1:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

Combined Vector2:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

Рис. 5: Пункт 3.3

```
tmp1 = [4, 6, 3]
tmp2 = [x for x in [4, 6, 3]]

println("tmp1:")
println(tmp1)

println("\ntmp2:")
println(tmp2)
```

```
tmp1:
[4, 6, 3]
```

```
tmp2:
[4, 6, 3]
```



```
tmp_filled1 = []  
tmp_filled2 = tmp1[1]  
  
foreach(_ -> push!(tmp_filled1, tmp1[1]), 1:10)  
for i in 1:10 tmp_filled2 = vcat(tmp2[1], tmp_filled2) end  
  
println("Filled tmp1:")  
println(tmp_filled1)  
  
println("\nFilled tmp2:")  
println(tmp_filled2)
```

```
Filled tmp1:  
Any{4, 4, 4, 4, 4, 4, 4, 4, 4, 4}  
  
Filled tmp2:  
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
```

Рис. 7: Пункт 3.5

```
rep_tmp1 = tmp1
rep_tmp2 = repeat(tmp2, inner=10)

for i in 1:9 rep_tmp1 = vcat(tmp1, rep_tmp1) end
println("Repeated tmp1:")
println(rep_tmp1)

println("\nRepeated tmp2:")
println(rep_tmp2)
```

Repeated tmp1:

[4, 6, 3, 4, 6, 3, 4, 6, 3, 4, 6, 3, 4, 6, 3, 4, 6, 3, 4, 6, 3, 4, 6, 3, 4, 6, 3]

Repeated tmp2:

[4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3]

Рис. 8: Пункт 3.6

```
rep_tmp3 = []
rep_tmp4 = [fill(tmp1[1], 11); fill(tmp1[2], 10); fill(tmp1[3], 10)]

foreach(_ -> push!(rep_tmp3, tmp2[1]), 1:11); foreach(_ -> push!(rep_tmp3, tmp2[2]), 1:10)
foreach(_ -> push!(rep_tmp3, tmp2[3]), 1:10)

println("Repeated tmp3:")
println(rep_tmp3)

println("\nRepeated tmp4:")
println(rep_tmp4)

Repeated tmp3:
Any[4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3]

Repeated tmp4:
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

Рис. 9: Пункт 3.7

[illegible]

```
pov_tmp1 = [fill(2*tmp1[1]); fill(2*tmp1[2]); fill(2*tmp1[3], 4)]
pov_tmp2 = []

push!(pov_tmp2, 2*tmp1[1]); push!(pov_tmp2, 2*tmp1[2])
foreach(_-> push!(pov_tmp2, 2*tmp1[3]), 1:4)

function six_counter(array)
    amount = 0
    for num in 1:6
        if '6' in string(array[num])
            amount += 1
        end
    end
    return amount
end

println("Powered tmp1:")
println(pov_tmp1)

println("\nPowered tmp2:")
println(pov_tmp2)

println("\nSixes in powered tmp1:")
println(six_counter(pov_tmp1))

println("\nSixes in powered tmp2:")
println(six_counter(pov_tmp2))

Powered tmp1:
[16, 64, 8, 8, 8, 8]

Powered tmp2:
Any[16, 64, 8, 8, 8, 8]

Sixes in powered tmp1:
2

Sixes in powered tmp2:
2
```

Рис. 11: Пункт 3.9

```
using Statistics

x = 3:0.1:6
y1 = [exp(i) * cos(i) for i in x]
ymean1 = mean(y1)

y2 = exp.(x) .* cos.(x)
ymean2 = sum(y2) / length(y2)

println("y1 values:")
for i in 1:31
    println(y1[i])
end

println("\ny1 mean values:")
println(ymean1)

println("\ny2 values:")
for i in 1:31
    println(y2[i])
end

println("\ny2 mean values:")
println(ymean2)
```

Рис. 12: Пункт 3.10

```
x = 0.1
y = 0.2
i_values = 3:3:36
j_values = 1:3:34

res_vec1 = [(x^i, y^j) for i in 3:3:36, j in 1:3:34]
res_vec2 = [(x^i, y^j) for i in i_values, j in j_values]

println("Result vector 1:")
println(res_vec1)

println("\nResult vector 2:")
println(res_vec2)
```

Result vector 1:

[(0.0010000000000000002, 0.2) (0.0010000000000000002, 0.0016000000000000003) (0.0010000000000000002, 1.2800000000000000e-10) (0.0010000000000000002, 6.5536000000000005e-12) (0.0010000000000000002, 5.2428800000000005e-14) (0.0010000000000000002, 2.6843545600000004e-20) (0.0010000000000000002, 2.1474836480000004e-22) (0.0010000000000000002, 1.0000000000000000e-03) (1.0000000000000004e-6, 1.2800000000000006e-5) (1.0000000000000004e-6, 1.0240000000000006e-7) (1.0000000000000004e-6, 6.5536000000000005e-9) (1.0000000000000004e-6, 5.2428800000000005e-11) (1.0000000000000004e-6, 4.1943040000000004e-13) (1.0000000000000004e-6, 3.3554432000000003e-15) (1.0000000000000004e-6, 2.6843545600000004e-17) (1.0000000000000004e-6, 2.1474836480000004e-19) (1.0000000000000004e-6, 1.7179869184000003e-21) (1.0000000000000004e-6, 1.3743895347200002e-23) (1.0000000000000004e-6, 1.1075116277760002e-25) (1.0000000000000004e-6, 8.860093022208002e-28) (1.0000000000000004e-6, 7.088074417766402e-30) (1.0000000000000004e-6, 5.670459534213121e-32) (1.0000000000000004e-6, 4.536367627370497e-34) (1.0000000000000004e-6, 3.629094101896397e-36) (1.0000000000000004e-6, 2.903275281517118e-38) (1.0000000000000004e-6, 2.322620225213694e-40) (1.0000000000000004e-6, 1.858096180170955e-42) (1.0000000000000004e-6, 1.486476944136764e-44) (1.0000000000000004e-6, 1.189181555309411e-46) (1.0000000000000004e-6, 9.513452442475288e-49) (1.0000000000000004e-6, 7.61076195398023e-51) (1.0000000000000004e-6, 6.088610363184184e-53) (1.0000000000000004e-6, 4.870888290547347e-55) (1.0000000000000004e-6, 3.896710632437878e-57) (1.0000000000000004e-6, 3.117368505950302e-59) (1.0000000000000004e-6, 2.493894804760241e-61) (1.0000000000000004e-6, 1.995115843808193e-63) (1.0000000000000004e-6, 1.596092675046554e-65) (1.0000000000000004e-6, 1.276874140037243e-67) (1.0000000000000004e-6, 1.021499312029794e-69) (1.0000000000000004e-6, 8.17199449623835e-72) (1.0000000000000004e-6, 6.53759559707068e-74) (1.0000000000000004e-6, 5.230076477656544e-76) (1.0000000000000004e-6, 4.184061182125235e-78) (1.0000000000000004e-6, 3.347249745700188e-80) (1.0000000000000004e-6, 2.67779987656015e-82) (1.0000000000000004e-6, 2.14223990124812e-84) (1.0000000000000004e-6, 1.713791921006496e-86) (1.0000000000000004e-6, 1.371033536805197e-88) (1.0000000000000004e-6, 1.096826829444157e-90) (1.0000000000000004e-6, 8.774614635553256e-93) (1.0000000000000004e-6, 7.019691708442605e-95) (1.0000000000000004e-6, 5.615753366754084e-97) (1.0000000000000004e-6, 4.492602693403267e-99) (1.0000000000000004e-6, 3.594082154722614e-101) (1.0000000000000004e-6, 2.875265723778091e-103) (1.0000000000000004e-6, 2.300212579022473e-105) (1.0000000000000004e-6, 1.840170063217978e-107) (1.0000000000000004e-6, 1.472136050574382e-109) (1.0000000000000004e-6, 1.177708840459506e-111) (1.0000000000000004e-6, 9.421670723676048e-114) (1.0000000000000004e-6, 7.537336578940838e-116) (1.0000000000000004e-6, 6.030669263152671e-118) (1.0000000000000004e-6, 4.824535410522137e-120) (1.0000000000000004e-6, 3.85962832841771e-122) (1.0000000000000004e-6, 3.087702662734168e-124) (1.0000000000000004e-6, 2.470162130187334e-126) (1.0000000000000004e-6, 1.976129704150667e-128) (1.0000000000000004e-6, 1.580903763320534e-130) (1.0000000000000004e-6, 1.265523010656427e-132) (1.0000000000000004e-6, 1.012418408525141e-134) (1.0000000000000004e-6, 8.10734726820113e-137) (1.0000000000000004e-6, 6.485877814560904e-139) (1.0000000000000004e-6, 5.188702251648723e-141) (1.0000000000000004e-6, 4.150961801318978e-143) (1.0000000000000004e-6, 3.320770241055182e-145) (1.0000000000000004e-6, 2.656616192844146e-147) (1.0000000000000004e-6, 2.125293034275317e-149) (1.0000000000000004e-6, 1.696234427420253e-151) (1.0000000000000004e-6, 1.357067541936202e-153) (1.0000000000000004e-6, 1.085654033549762e-155) (1.0000000000000004e-6, 8.6852322683981e-158) (1.0000000000000004e-6, 6.94818581471848e-160) (1.0000000000000004e-6, 5.558548651774784e-162) (1.0000000000000004e-6, 4.446838921419827e-164) (1.0000000000000004e-6, 3.557471137135861e-166) (1.0000000000000004e-6, 2.845976909708689e-168) (1.0000000000000004e-6, 2.276781527766951e-170) (1.0000000000000004e-6, 1.821425222213561e-172) (1.0000000000000004e-6, 1.457140177770849e-174) (1.0000000000000004e-6, 1.165712142216679e-176) (1.0000000000000004e-6, 9.32569713773343e-179) (1.0000000000000004e-6, 7.460557710186744e-181) (1.0000000000000004e-6, 5.968446168149395e-183) (1.0000000000000004e-6, 4.774756934519516e-185) (1.0000000000000004e-6, 3.819805547615613e-187) (1.0000000000000004e-6, 3.055844438092491e-189) (1.0000000000000004e-6, 2.444675550474073e-191) (1.0000000000000004e-6, 1.955740440379258e-193) (1.0000000000000004e-6, 1.564592352303406e-195) (1.0000000000000004e-6, 1.251673881842725e-197) (1.0000000000000004e-6, 1.00133910547418e-199) (1.0000000000000004e-6, 8.01071284379344e-202) (1.0000000000000004e-6, 6.408570275034752e-204) (1.0000000000000004e-6, 5.126856220027801e-206) (1.0000000000000004e-6, 4.101485056022241e-208) (1.0000000000000004e-6, 3.281188044817793e-210) (1.0000000000000004e-6, 2.625030435854234e-212) (1.0000000000000004e-6, 2.100024348683387e-214) (1.0000000000000004e-6, 1.679979478946709e-216) (1.0000000000000004e-6, 1.343983583157367e-218) (1.0000000000000004e-6, 1.075186866525894e-220) (1.0000000000000004e-6, 8.601494932207152e-223) (1.0000000000000004e-6, 6.881195945765721e-225) (1.0000000000000004e-6, 5.505036756612577e-227) (1.0000000000000004e-6, 4.404029405290062e-229) (1.0000000000000004e-6, 3.52322352423205e-231) (1.0000000000000004e-6, 2.81857881938564e-233) (1.0000000000000004e-6, 2.254863055508512e-235) (1.0000000000000004e-6, 1.80389044440681e-237) (1.0000000000000004e-6, 1.443112355525448e-239) (1.0000000000000004e-6, 1.154490684420358e-241) (1.0000000000000004e-6, 9.235925475362864e-244) (1.0000000000000004e-6, 7.388740380290291e-246) (1.0000000000000004e-6, 5.911072304232232e-248) (1.0000000000000004e-6, 4.728857843385786e-250) (1.0000000000000004e-6, 3.783086274708629e-252) (1.0000000000000004e-6, 3.026469019766903e-254) (1.0000000000000004e-6, 2.421175215813522e-256) (1.0000000000000004e-6, 1.936940172650818e-258) (1.0000000000000004e-6, 1.549552138120654e-260) (1.0000000000000004e-6, 1.239641710496523e-262) (1.0000000000000004e-6, 9.917133683972184e-265) (1.0000000000000004e-6, 7.933706947177747e-267) (1.0000000000000004e-6, 6.346965557742197e-269) (1.0000000000000004e-6, 5.077572446193758e-271) (1.0000000000000004e-6, 4.062057957035006e-273) (1.0000000000000004e-6, 3.250046365628005e-275) (1.0000000000000004e-6, 2.599237092502404e-277) (1.0000000000000004e-6, 2.078589674001923e-279) (1.0000000000000004e-6, 1.662871739201538e-281) (1.0000000000000004e-6, 1.33029739136123e-283) (1.0000000000000004e-6, 1.064317913089784e-285) (1.0000000000000004e-6, 8.514543304718272e-288) (1.0000000000000004e-6, 6.811634643774617e-290) (1.0000000000000004e-6, 5.449307715019694e-292) (1.0000000000000004e-6, 4.359446172015755e-294) (1.0000000000000004e-6, 3.487556937612604e-296) (1.0000000000000004e-6, 2.790045550090083e-298) (1.0000000000000004e-6, 2.232036440072066e-300) (1.0000000000000004e-6, 1.785629152057653e-302) (1.0000000000000004e-6, 1.428503321646122e-304) (1.0000000000000004e-6, 1.142802657316898e-306) (1.0000000000000004e-6, 9.142421258535184e-309) (1.0000000000000004e-6, 7.313937006828147e-311) (1.0000000000000004e-6, 5.851149605462518e-313) (1.0000000000000004e-6, 4.680919684370014e-315) (1.0000000000000004e-6, 3.744735747496011e-317) (1.0000000000000004e-6, 3.0000000000000004e-319) (1.0000000000000004e-6, 2.4000000000000004e-321) (1.0000000000000004e-6, 1.9200000000000004e-323) (1.0000000000000004e-6, 1.5360000000000004e-325) (1.0000000000000004e-6, 1.2288000000000004e-327) (1.0000000000000004e-6, 9.830400000000002e-330) (1.0000000000000004e-6, 7.864320000000002e-332) (1.0000000000000004e-6, 6.291456000000002e-334) (1.0000000000000004e-6, 5.033164800000002e-336) (1.0000000000000004e-6, 4.026531840000002e-338) (1.0000000000000004e-6, 3.221225472000002e-340) (1.0000000000000004e-6, 2.577060377600002e-342) (1.0000000000000004e-6, 2.061648302080002e-344) (1.0000000000000004e-6, 1.650118641664002e-346) (1.0000000000000004e-6, 1.320094913331201e-348) (1.0000000000000004e-6, 1.056075930664961e-350) (1.0000000000000004e-6, 8.448607445319688e-353) (1.0000000000000004e-6, 6.75888595625575e-355) (1.0000000000000004e-6, 5.407108765004601e-357) (1.0000000000000004e-6, 4.325687012003681e-359) (1.0000000000000004e-6, 3.460549609602945e-361) (1.0000000000000004e-6, 2.768439687682356e-363) (1.0000000000000004e-6, 2.214751750145885e-365) (1.0000000000000004e-6, 1.771801400116708e-367) (1.0000000000000004e-6, 1.417441120093366e-369) (1.0000000000000004e-6, 1.133952896074693e-371) (1.0000000000000004e-6, 9.071623168597544e-374) (1.0000000000000004e-6, 7.257306534878035e-376) (1.0000000000000004e-6, 5.805845227902428e-378) (1.0000000000000004e-6, 4.644676182321942e-380) (1.0000000000000004e-6, 3.715741025857554e-382) (1.0000000000000004e-6, 2.972592820686043e-384) (1.0000000000000004e-6, 2.378074256548834e-386) (1.0000000000000004e-6, 1.902459405239067e-388) (1.0000000000000004e-6, 1.521967524191254e-390) (1.0000000000000004e-6, 1.217574019353003e-392) (1.0000000000000004e-6, 9.740592154824024e-395) (1.0000000000000004e-6, 7.792473723859219e-397) (1.0000000000000004e-6, 6.233978979087375e-399) (1.0000000000000004e-6, 5.0071831832699e-401) (1.0000000000000004e-6, 4.00574654661592e-403) (1.0000000000000004e-6, 3.204597237292736e-405) (1.0000000000000004e-6, 2.563677789834189e-407) (1.0000000000000004e-6, 2.050942231867351e-409) (1.0000000000000004e-6, 1.640753785493881e-411) (1.0000000000000004e-6, 1.312603028395105e-413) (1.0000000000000004e-6, 1.050082422716084e-415) (1.0000000000000004e-6, 8.400659381728672e-418) (1.0000000000000004e-6, 6.720527505382938e-420) (1.0000000000000004e-6, 5.37642200430635e-422) (1.0000000000000004e-6, 4.30113760344508e-424) (1.0000000000000004e-6, 3.440910082756064e-426) (1.0000000000000004e-6, 2.752728066204851e-428) (1.0000000000000004e-6, 2.202182453043881e-430) (1.0000000000000004e-6, 1.761745962435105e-432) (1.0000000000000004e-6, 1.410196770748084e-434) (1.0000000000000004e-6, 1.128157416678467e-436) (1.0000000000000004e-6, 9.025259333427736e-439) (1.0000000000000004e-6, 7.220207466742189e-441) (1.0000000000000004e-6, 5.776165973393751e-443) (1.0000000000000004e-6, 4.620932778715001e-445) (1.0000000000000004e-6, 3.700746222972001e-447) (1.0000000000000004e-6, 2.960597058377601e-449) (1.0000000000000004e-6, 2.368477646702081e-451) (1.0000000000000004e-6, 1.894782117361665e-453) (1.0000000000000004e-6, 1.515825693889332e-455) (1.0000000000000004e-6, 1.212660555111466e-457) (1.0000000000000004e-6, 9.701284440891728e-460) (1.0000000000000004e-6, 7.761027552713382e-462) (1.0000000000000004e-6, 6.208822042170706e-464) (1.0000000000000004e-6, 4.967057633736565e-466) (1.0000000000000004e-6, 3.973646107069252e-468) (1.0000000000000004e-6, 3.178916885655402e-470) (1.0000000000000004e-6, 2.543133508524321e-472) (1.0000000000000004e-6, 2.034506806819457e-474) (1.0000000000000004e-6, 1.627605445455566e-476) (1.0000000000000004e-6, 1.302084356364453e-478) (1.0000000000000004e-6, 1.037667485091562e-480) (1.0000000000000004e-6, 8.3013398807325e-482) (1.0000000000000004e-6, 6.641071904586e-484) (1.0000000000000004e-6, 5.3128575236688e-486) (1.0000000000000004e-6, 4.25028601893504e-488) (1.0000000000000004e-6, 3.399428815148032e-490) (1.0000000000000004e-6, 2.719543052118426e-492) (1.0000000000000004e-6, 2.175634441694741e-494) (1.0000000000000004e-6, 1.740507553355793e-496) (1.0000000000000004e-6, 1.392406042684634e-498) (1.0000000000000004e-6, 1.113924834147707e-500) (1.0000000000000004e-6, 8.911398673181656e-503) (1.0000000000000004e-6, 7.129118938545325e-505) (1.0000000000000004e-6, 5.69529515083626e-507) (1.0000000000000004e-6, 4.556236120669008e-509) (1.0000000000000004e-6, 3.645068896535206e-511) (1.0000000000000004e-6, 2.916055117228165e-513) (1.0000000000000004e-6, 2.332844093782532e-515) (1.0000000000000004e-6, 1.866275275026026e-517) (1.0000000000000004e-6, 1.493020220020821e-519) (1.0000000000000004e-6, 1.186416176016657e-521) (1.0000000000000004e-6, 9.491329408133256e-524) (1.0000000000000004e-6, 7.593063526506605e-526) (1.0000000000000004e-6, 6.074450821205284e-528) (1.0000000000000004e-6, 4.859560656964227e-530) (1.0000000000000004e-6, 3.887648525571381e-532) (1

Result vector 2:

```
[(0.001000000000000002, 0.2) (0.001000000000000002, 0.001600000000000003) (0.001000000000000002, 1.280000000000006e-5) (0.001000000000000002, 1.024000000000006e-7) (0.001000000000000002, 8.192000000000005e-10) (0.001000000000000002, 6.5536000000000055e-12) (0.001000000000000002, 5.2428800000000056e-14) (0.001000000000000002, 4.194304000000005e-16) (0.001000000000000002, 3.3554432000000044e-18) (0.001000000000000002, 2.684354560000004e-20) (0.001000000000000002, 2.147483648000004e-22) (0.001000000000000002, 1.7179869184000035e-24); (1.000000000000004e-6, 0.2) (1.000000000000004e-6, 0.001600000000000003) (1.000000000000004e-6, 1.280000000000006e-5) (1.000000000000004e-6, 1.024000000000006e-7) (1.000000000000004e-6, 8.192000000000005e-10) (1.000000000000004e-6, 6.5536000000000055e-12) (1.000000000000004e-6, 5.2428800000000056e-14) (1.000000000000004e-6, 4.194304000000005e-16) (1.000000000000004e-6, 3.3554432000000044e-18) (1.000000000000004e-6, 2.684354560000004e-20) (1.000000000000004e-6, 2.147483648000004e-22) (1.000000000000004e-6, 1.7179869184000035e-24); (1.000000000000005e-9, 0.2) (1.000000000000005e-9, 0.001600000000000003) (1.000000000000005e-9, 1.280000000000006e-5) (1.000000000000005e-9, 1.024000000000006e-7) (1.000000000000005e-9, 8.192000000000005e-10) (1.000000000000005e-9, 6.5536000000000055e-12) (1.000000000000005e-9, 5.2428800000000056e-14) (1.000000000000005e-9, 4.194304000000005e-16) (1.000000000000005e-9, 3.3554432000000044e-18) (1.000000000000005e-9, 2.684354560000004e-20) (1.000000000000005e-9, 2.147483648000004e-22) (1.000000000000005e-9, 1.717
```

Рис. 14: Пункт 3.11 - вектор 2


```
div_vec1 = []
div_vec2 = [2^i / i for i in 1:25]

M = 25
power = collect(1:M)

for p in power
    div_vec1 = push!(div_vec1, 2^p / p)
end

println("Division vector 1:")
println(div_vec1)

println("\nDivision vector 2:")
println(div_vec2)

Division vector 1:
Any{2.0, 2.0, 2.666666666666665, 4.0, 6.4, 10.666666666666666, 18.285714285714285, 32.0, 56.888888888888886, 102.4, 186.1818181818182, 341.3333333333333, 630.1538461538462, 1170.2857142857142, 2184.5333333333333, 4096.0, 7710.117647058823, 14563.555555555555, 27594.105263157893, 52428.8, 99864.38095238095, 190650.18181818182, 364722.0869565217, 699050.6666666666, 1.34217728e6}

Division vector 2:
[2.0, 2.0, 2.666666666666665, 4.0, 6.4, 10.666666666666666, 18.285714285714285, 32.0, 56.888888888888886, 102.4, 186.18181818182, 341.3333333333333, 630.1538461538462, 1170.2857142857142, 2184.5333333333333, 4096.0, 7710.117647058823, 14563.555555555555, 27594.105263157893, 52428.8, 99864.38095238095, 190650.18181818182, 364722.0869565217, 699050.6666666666, 1.34217728e6]
```

Рис. 15: Пункт 3.12

```
N = 30

fn_vec1 = ["fn$i" for i in 1:N]
fn_vec2 = []

for n in N fn_vec2 = push!(fn_vec1, "fn$n") end

println("Fn vector 1:")
println(fn_vec1)

println("\nFn vector 2:")
println(fn_vec2)

Fn vector 1:
["fn1", "fn2", "fn3", "fn4", "fn5", "fn6", "fn7", "fn8", "fn9", "fn10", "fn11", "fn12", "fn13", "fn14", "fn15", "fn16", "fn17", "fn18", "fn19", "fn20", "fn21", "fn22", "fn23", "fn24", "fn25", "fn26", "fn27", "fn28", "fn29", "fn30", "fn30"]

Fn vector 2:
["fn1", "fn2", "fn3", "fn4", "fn5", "fn6", "fn7", "fn8", "fn9", "fn10", "fn11", "fn12", "fn13", "fn14", "fn15", "fn16", "fn17", "fn18", "fn19", "fn20", "fn21", "fn22", "fn23", "fn24", "fn25", "fn26", "fn27", "fn28", "fn29", "fn30", "fn30"]
```

Рис. 16: Пункт 3.13

```
size = 100
squares = [i^2 for i in 1:size]
println("Squares array:")
println(squares)
```

Squares array:
[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, 1600, 1681, 1764, 1849, 1936, 2025, 2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249, 3364, 3481, 3600, 3721, 3844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 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]

Рис. 17: Пункт 4

```
using Primes

myprimes = primes(1000)[1:168]
least_number= myprimes[89]
prime_arr_cut = myprimes[89:99]

println("Primes Array: ")
println(myprimes)

println("\nLeast 89th prime number: ")
println(least_number)

println("\nSlice of 88-98 element: ")
println(prime_arr_cut)
```

Primes Array:
[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, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]

Least 89th prime number:
461

Slice of 88-98 element:
[461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]

Рис. 18: Пункт 5

```
M = 100
N = 20

sum_res1 = sum(i^3 + 4i^2 for i in 10:M)
sum_res2 = sum((2^i/i) + (3^i/i^2) for i in 1:M/4)
sum_res3 = 1.0 + sum(prod([(2 * i)/(2 * i + 1) for i in 1:n]) for n in 1:N)

println("Summary result: ", sum_res1)
println("Результат выражения: ", sum_res2)
println("Результат выражения: ", sum_res3)

Summary result: 26852735
Результат выражения: 2.1291704368143802e9
Результат выражения: 7.170891165651219
```

В ход выполнения работы я изучил несколько структур данных, реализованных в Julia, а также научился применять их и операции над ними для решения задач.