Компьютерный практикум по статистическому анализу данных

Лабораторная работа № 2. Julia. Структуры данных

Сунгурова Мариян

Содержание

Список иллюстраций

1 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).

2 Теоретическое введение

Julia — высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений.[1]. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков, однако имеет некоторые существенные отличия.

Для выполнения заданий была использована официальная документация Julia[2].

3 Выполнение лабораторной работы

Рассмотрим несколько структур данных, реализованных в Julia.

Несколько функций (методов), общих для всех структур данных:

– isempty() — проверяет, пуста ли структура данных;

– length() — возвращает длину структуры данных;

– in() — проверяет принадлежность элемента к структуре;

– unique() — возвращает коллекцию уникальных элементов структуры,

– reduce() — свёртывает структуру данных в соответствии с заданным бинарным оператором;

– maximum() (или minimum()) — возвращает наибольший (или наименьший) результат вызова функции для каждого элемента структуры данных.

Выполним примеры из лабораторной работы для действий над кортежами(рис. fig. 3.1 - fig. 3.2)

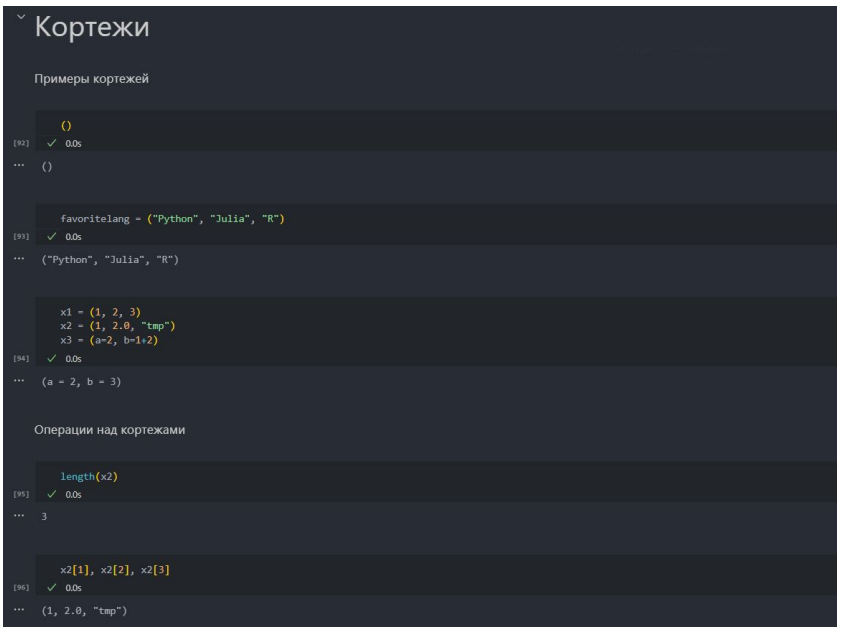

Рис. 3.1: Примеры. Кортежи

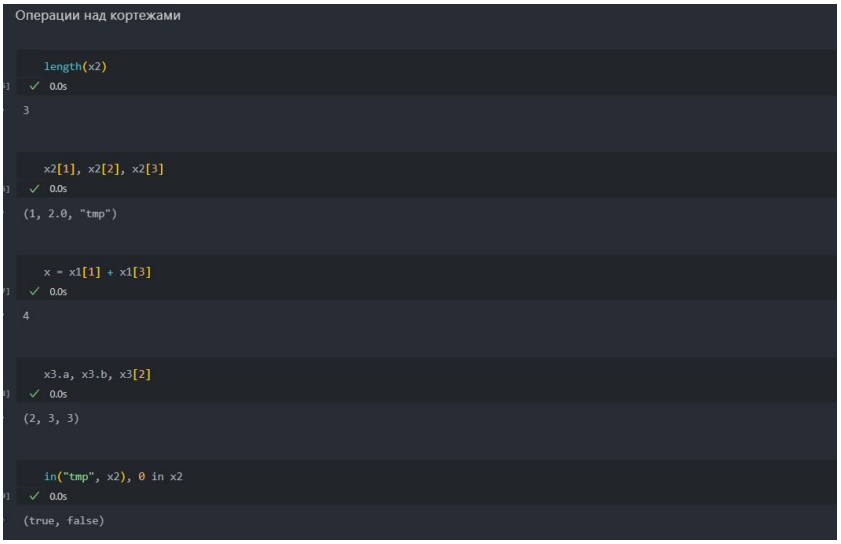

Рис. 3.2: Примеры. Кортежи

Также со словарями(рис. fig. 3.3 - fig. 3.4)

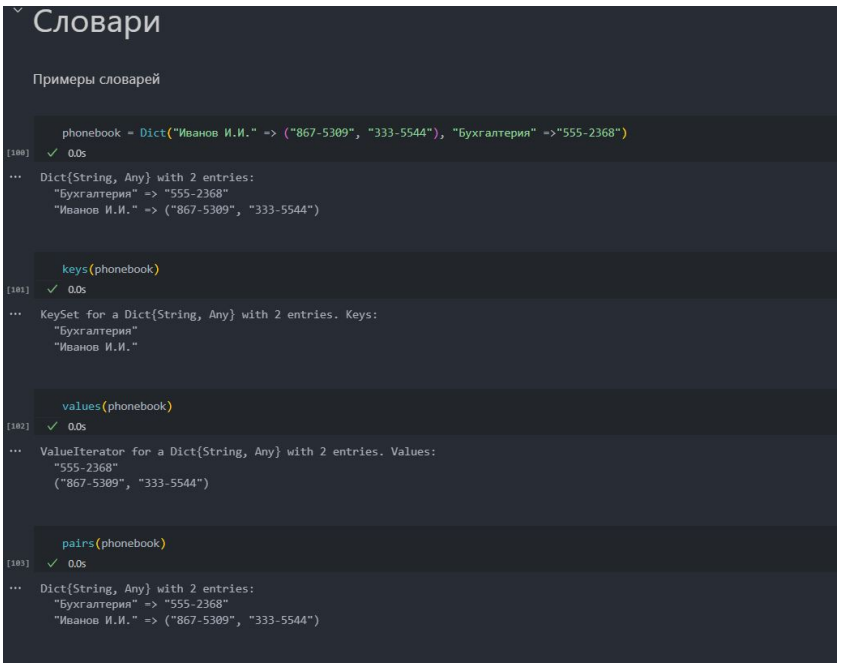

Рис. 3.3: Примеры. Словари

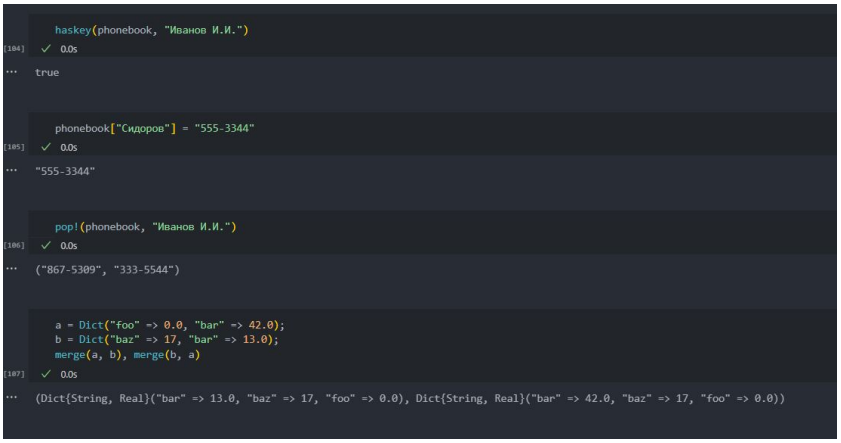

Рис. 3.4: Примеры. Словари

Рассмотрим также примеры опреций над множествами(рис. fig. 3.5 - fig. 3.6)

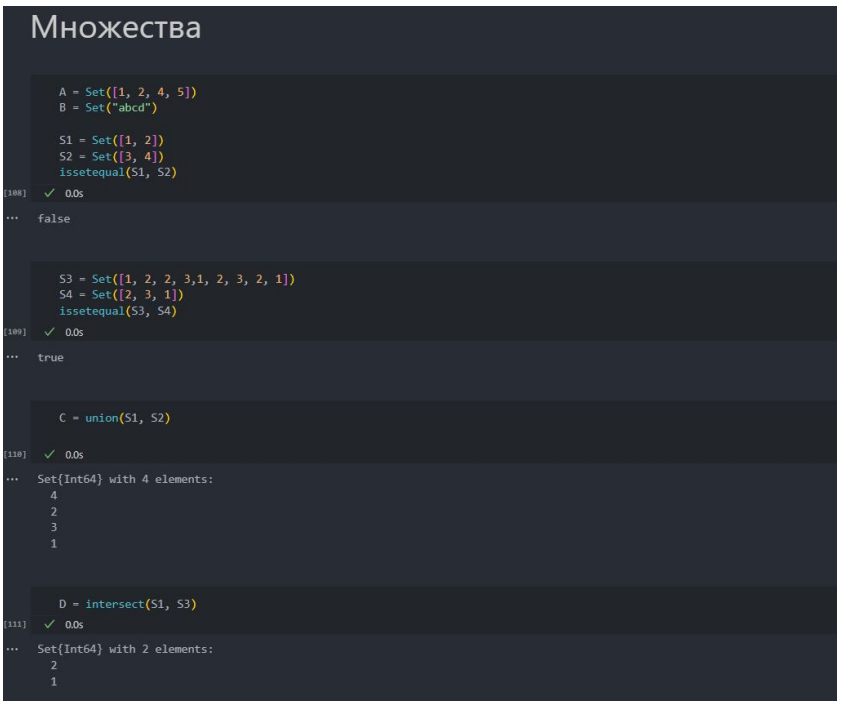

Рис. 3.5: Примеры. Множества

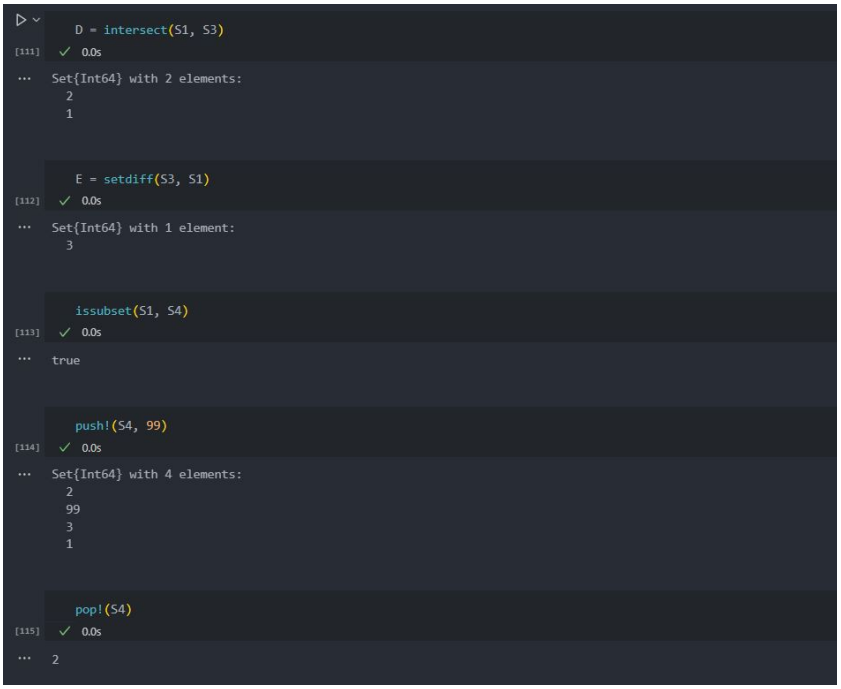

Рис. 3.6: Примеры. Множества

И с массивами(рис. fig. 3.7 - fig. 3.11)

Рис. 3.7: Примеры. Массивы

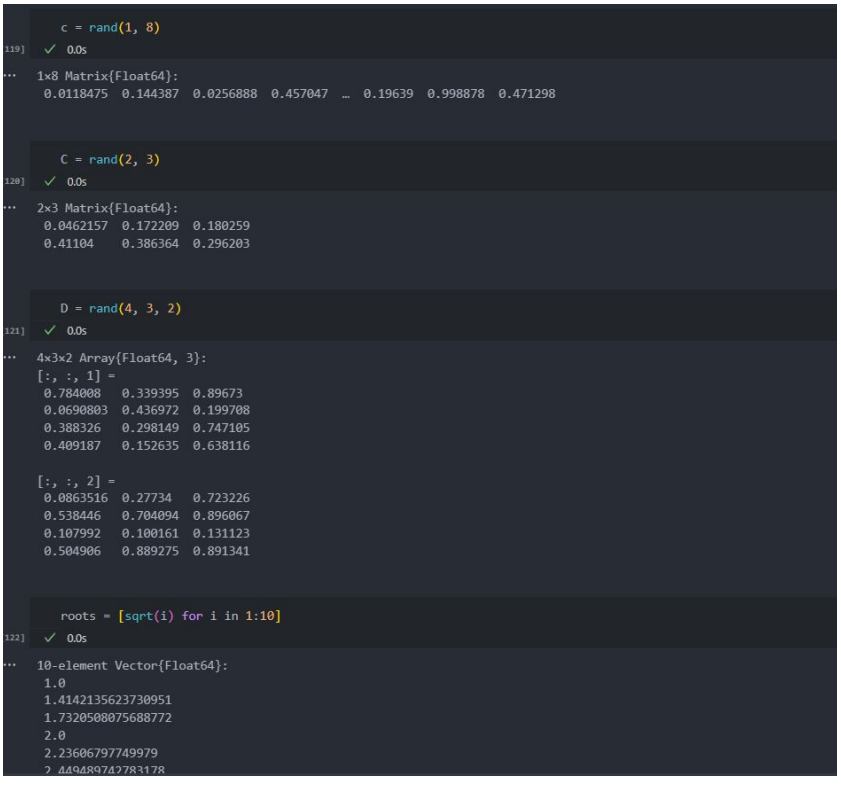

Рис. 3.8: Примеры. Массивы

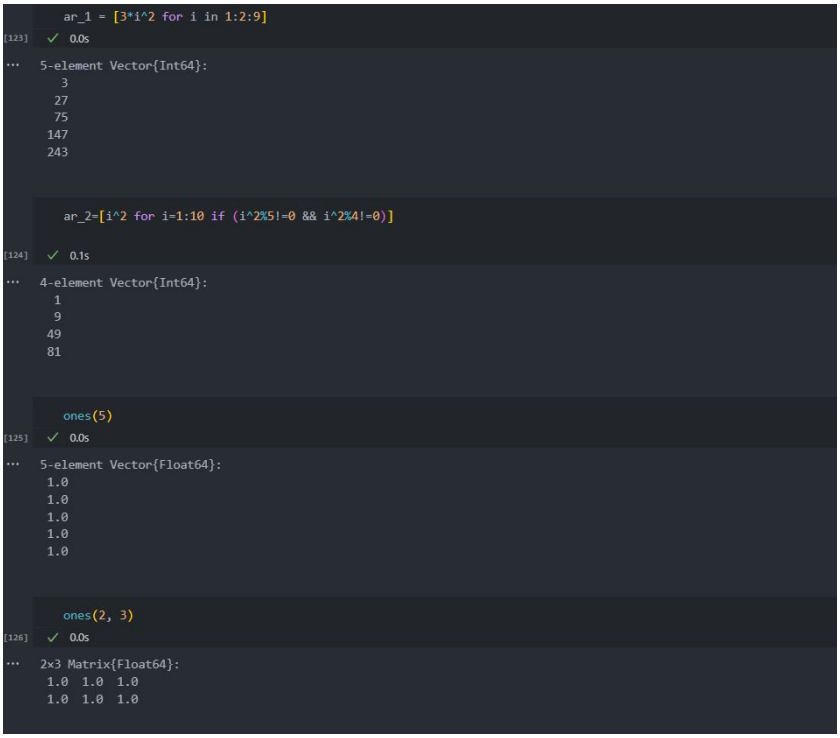

Рис. 3.9: Примеры. Массивы

Рис. 3.10: Примеры. Массивы

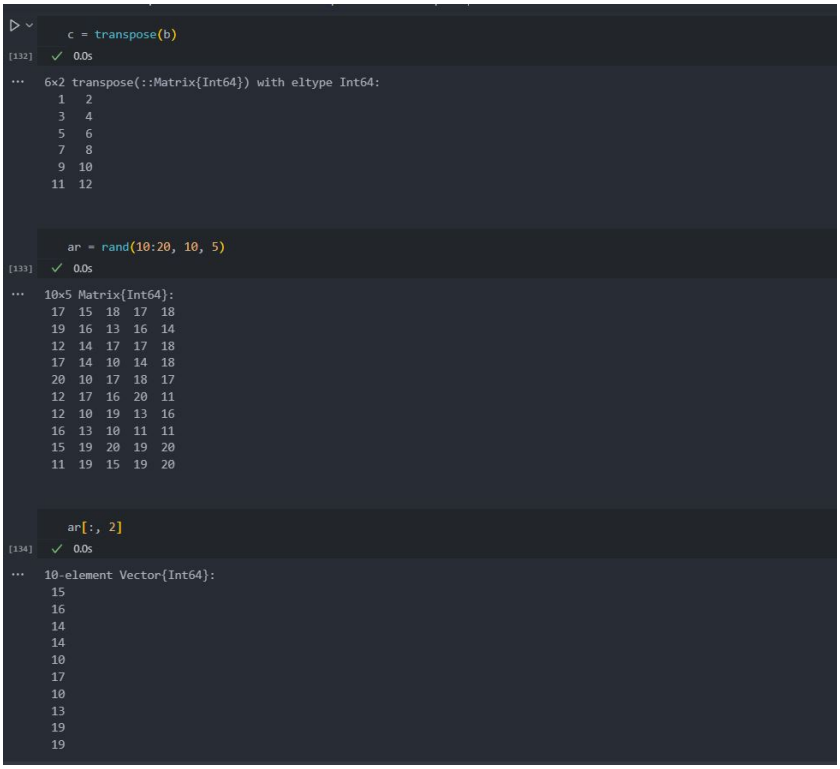

Рис. 3.11: Примеры. Массивы

Рис. 3.12: Примеры. Массивы

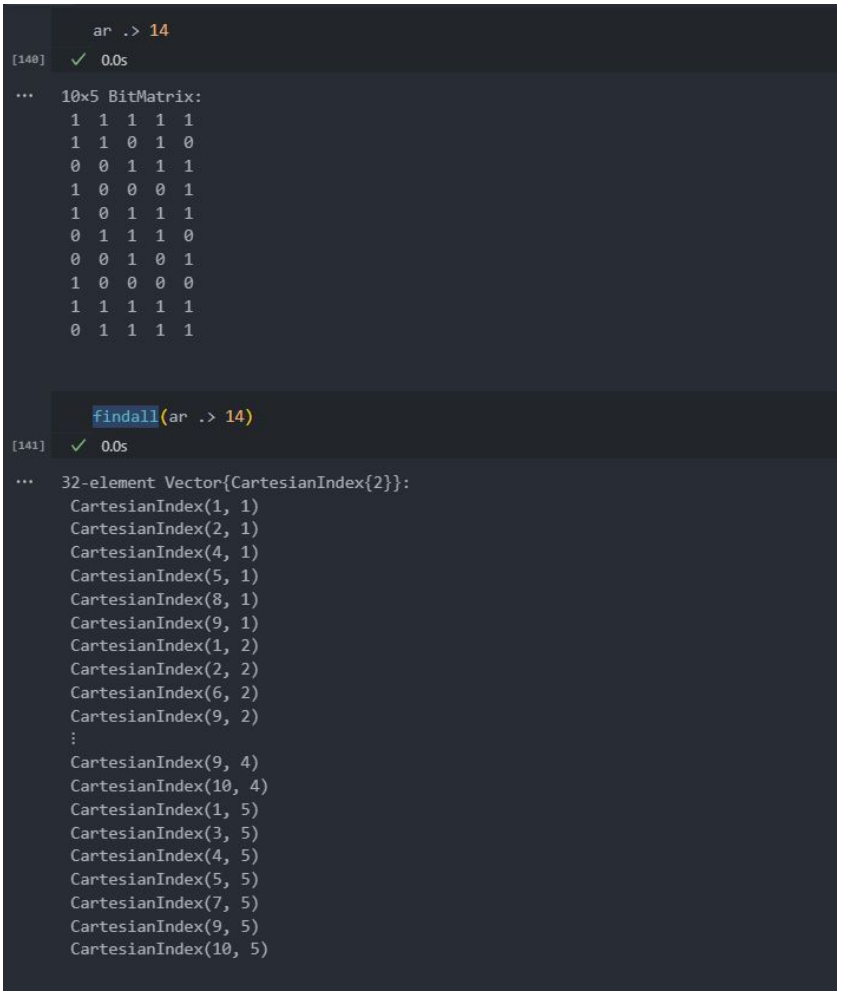

Рис. 3.13: Примеры. Массивы

Выполним задания для самостоятельной работы. С

1. Даны множества: ☒ = {0, 3, 4, 9}, ☒ = {1, 3, 4, 7}, ☒ = {0, 1, 2, 4, 7, 8, 9}. Найдем ☒ = ☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒.(рис. fig. 3.14)

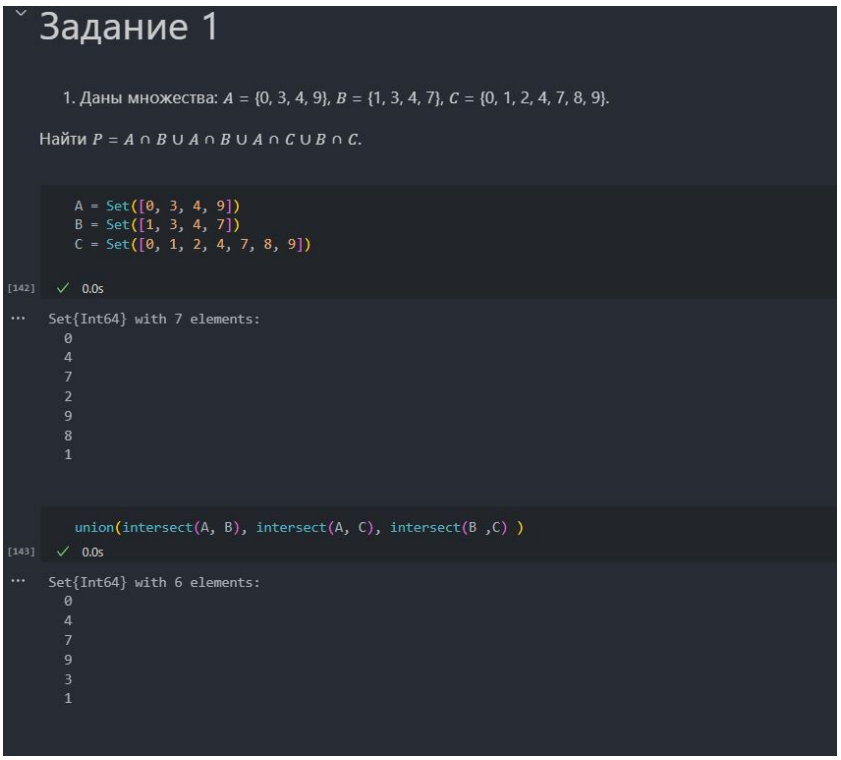

Рис. 3.14: Задание 1

2. Приведем свои примеры с выполнением операций над множествами элементов разных типов. (рис. fig. 3.16 - fig. 3.17)

Задание 2

Приведите свои примеры с выполнением операций над множествами элементов разных типов.

```
S1 = Set([10, 2.0, 3, 4])
S2 = Set(["Some", "text", "and", 10])

println("Элементы множества S1: ")
for i in S1
    println(i)
end

println("\nЭлементы множества S2: ")
for i in S2
    println(i)
end

println("\nПроверка эквивалентности:")
println(issetequal(S1, S2))

println("\nПересечение множеств:")
println(intersect(S1, S2))

println("\nОбъединение множеств:")
println(union(S1, S2))

println("\nРазность множеств:")
println(union(S1, S2))

println("\nПроверка вхождения элементов одного множества в другое:")
println(issubset(S1, S2))
```

[144] ✓ 0.0s

Рис. 3.15: Задание 2

```
Элементы множества S1:
4.0
2.0
10.0
3.0

Элементы множества S2:
10
Some
text
and

Проверка эквивалентности:
false

Пересечение множеств:
Set(Any[10])

Объединение множеств:
Set(Any[4.0, 2.0, 10, "Some", "text", 3.0, "and"])

Разность множеств:
Set(Any[4.0, 2.0, 10, "Some", "text", 3.0, "and"])

Проверка вхождения элементов одного множества в другое:
false
```

Рис. 3.16: Задание 2

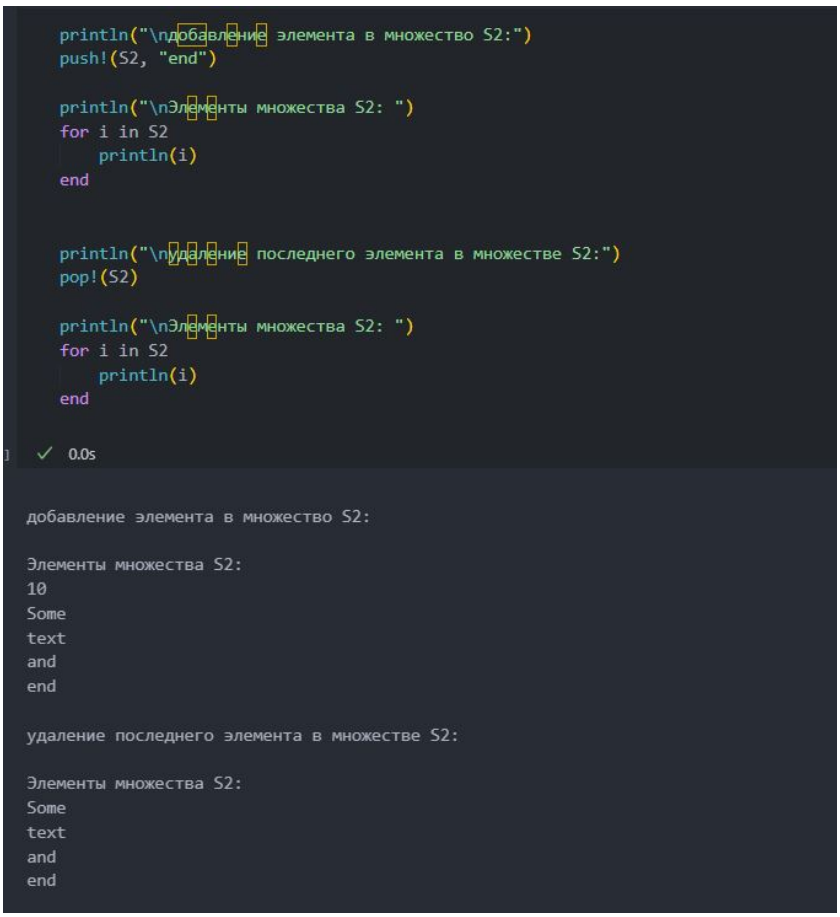

Рис. 3.17: Задание 2

1. Создадим разными способами массивы и вектора. Для создания нужных массивов, используем генераторы и циклы(рис. fig. 3.18 - fig. 3.23)

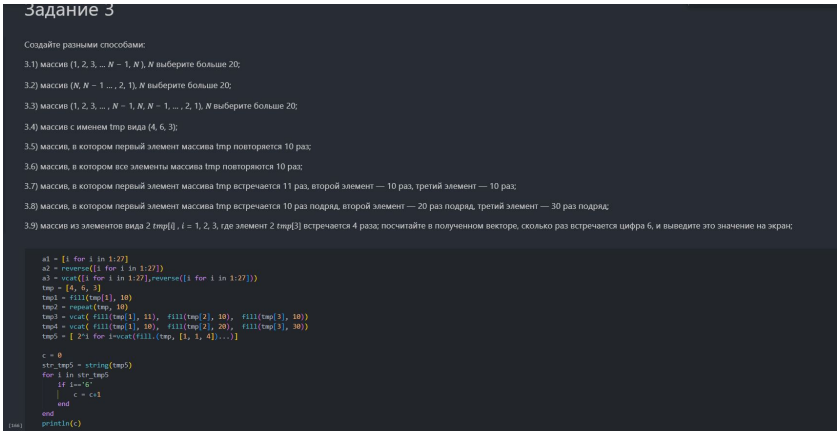

Рис. 3.18: Задание 3

вектор значений $y = ex\cos(x)$ в точках x = 3, 3.1, 3.2, ... , 6, найдите среднее значение y;

```
function Y(x)
    exp(x)*cos(x)
end
```

[168] ✓ 0.2s

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

```
X = 3:0.1:6
res = [Y(x) for x in X ]
println(sum(res)/sizeof(res))
```

[170] ✓ 0.0s

```
6.639218243303714
```

вектор вида (xi , yj), x = 0.1, i = 3, 6, 9, ... , 36, y = 0.2, j = 1, 4, 7, ... , 34;

```
is = 3:3:36
js = 1:3:34
x = 0.1
y = 0.2
v = [ [x^is[k],y^js[k]] for k in 1:length(is)]
```

[182] ✓ 0.6s

```
12-element Vector{Vector{Float64}}:
 [0.0010000000000000002, 0.2]
 [1.0000000000000004e-6, 0.0016000000000000003]
 [1.0000000000000005e-9, 1.2800000000000005e-5]
 [1.0000000000000006e-12, 1.0240000000000006e-7]
 [1.0000000000000009e-15, 8.192000000000005e-10]
 [1.000000000000001e-18, 6.553600000000055e-12]
 [1.0000000000000012e-21, 5.2428800000000056e-14]
 [1.0000000000000014e-24, 4.194304000000005e-16]
 [1.0000000000000015e-27, 3.3554432000000048e-18]
 [1.0000000000000017e-30, 2.684354560000004e-20]
 [1.0000000000000018e-33, 2.1474836480000035e-22]
 [1.000000000000002e-36, 1.717986918400003e-24]
```

Рис. 3.19: Задание 3

вектор с элементами 2^i, $i = 1, 2, \ldots, M$, $M = 25$;

```
M = 25
is = 1:M
v1 = [2^i/i for i in is]
```

184] ✓ 0.0s

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

вектор вида ("fn1", "fn2", ..., "fnN"), $N = 30$;

```
v2 = []
for i=1:30
    push!(v2, "fn"*string(i))
end
v2
```

187] ✓ 0.4s

```
30-element Vector{Any}:
 "fn1"
 "fn2"
```

Рис. 3.20: Задание 3

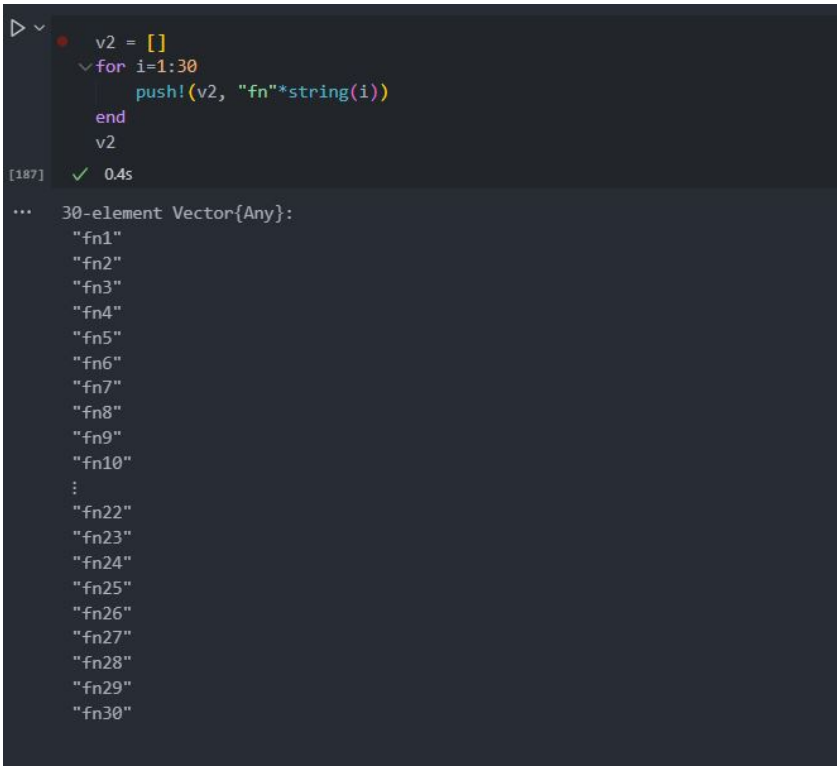

Рис. 3.21: Задание 3

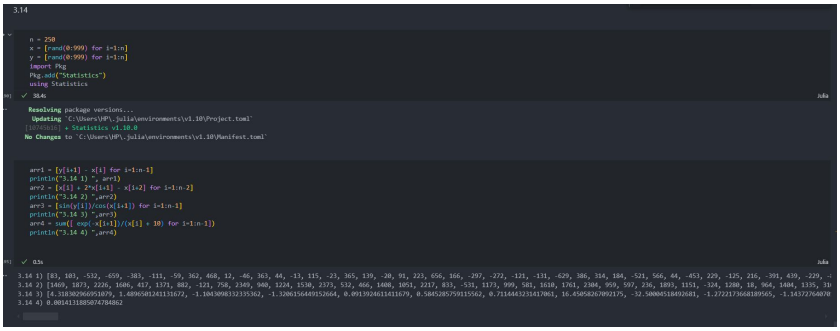

Рис. 3.22: Задание 3

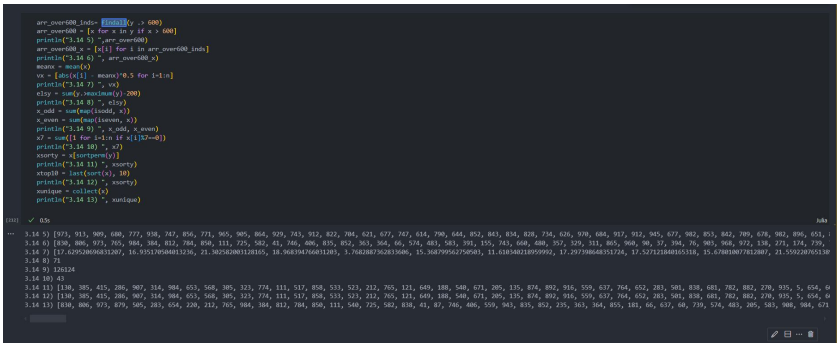

Рис. 3.23: Задание 3

4. Создадим массив squares, в котором будут храниться квадраты всех целых чисел от 1 до 100.(рис. fig. 3.24)

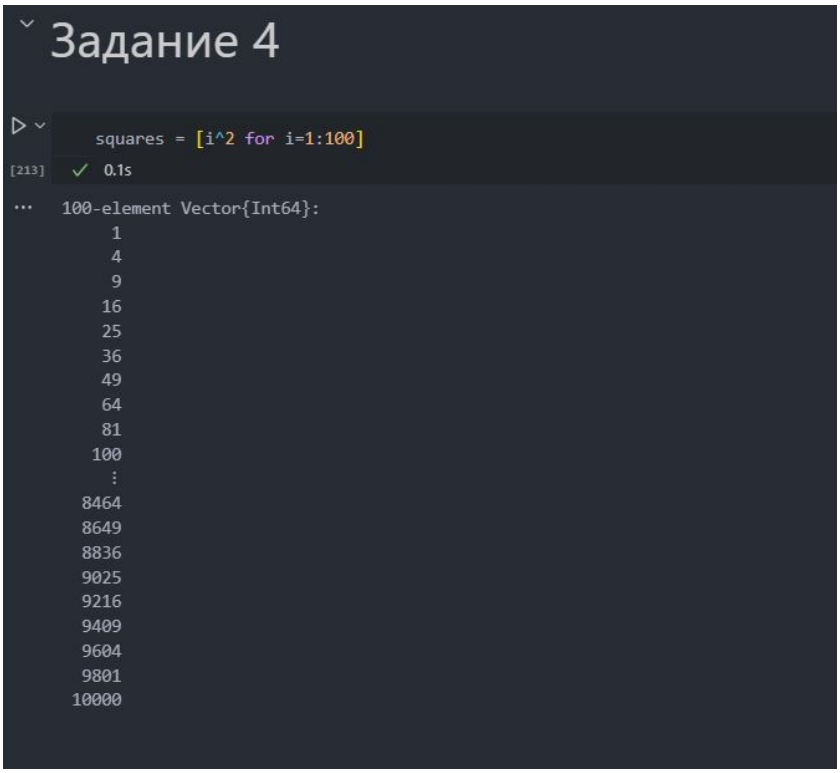

Рис. 3.24: Задание 4

5. Подключим пакет Primes (функции для вычисления простых чисел). Затем сгенерируем массив myprimes, в котором будут храниться первые 168 простых чисел. Определим также 89-е наименьшее простое число и срез

массива с 89-го до 99-го элемента включительно, содержащий наименьшие простые числа.(рис. fig. 3.25)

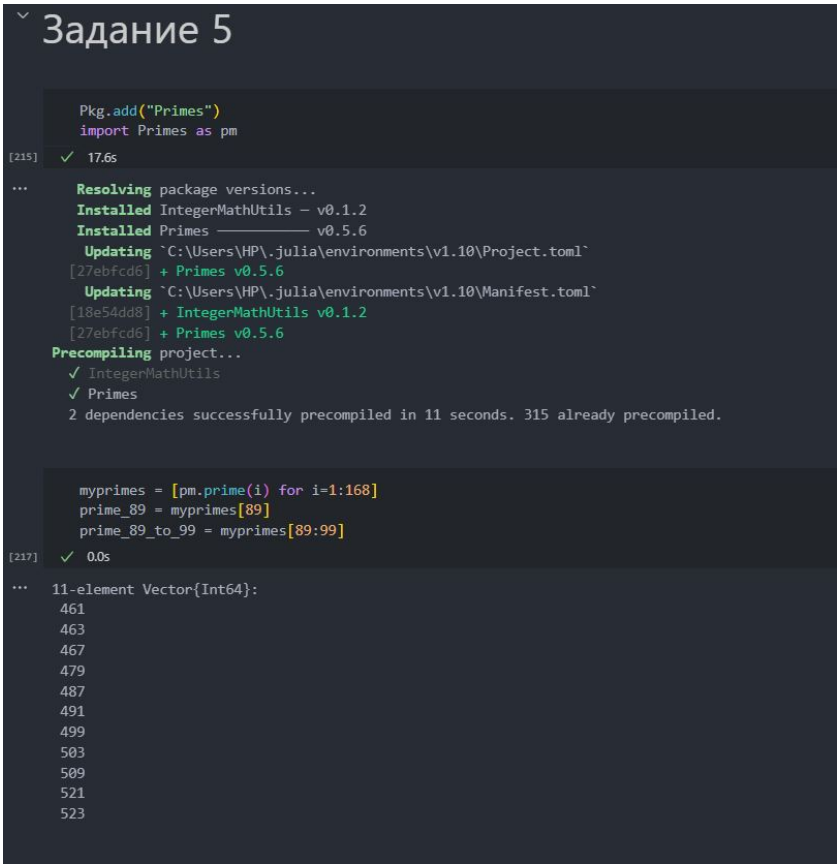

Рис. 3.25: Задание 5

6. Вычислим выражения(рис. fig. 3.26)

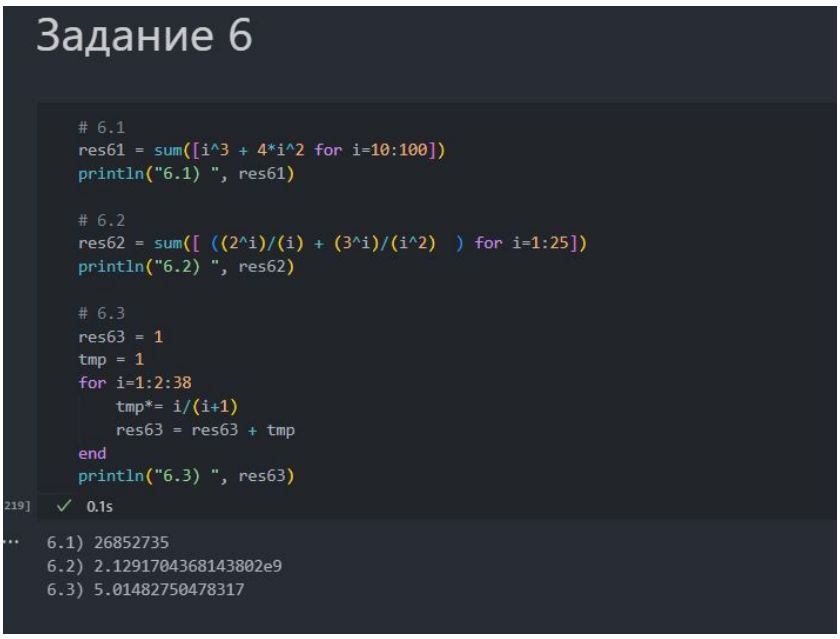

Рис. 3.26: Задание 6

4 Выводы

В результате выполнения данной лабораторной работы были изучены структуры данных, реализованных в Julia: словарь, массив, кортеж множество, также юыли получены практические навыки применения этих структур и операций над ними решения задач.

Список литературы

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).