Лабораторная работа № 2

Структуры данных

Оразгелдиев Язгелди

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

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

3 Выполнение лабораторной работы

Для начала выполним примеры из раздела про кортежи. Кортеж (Tuple) – структура данных (контейнер) в виде неизменяемой индексируемой последовательности элементов какого-либо типа (элементы индексиру- ются с единицы).

Рисунок 3.1: Примеры использования кортежей

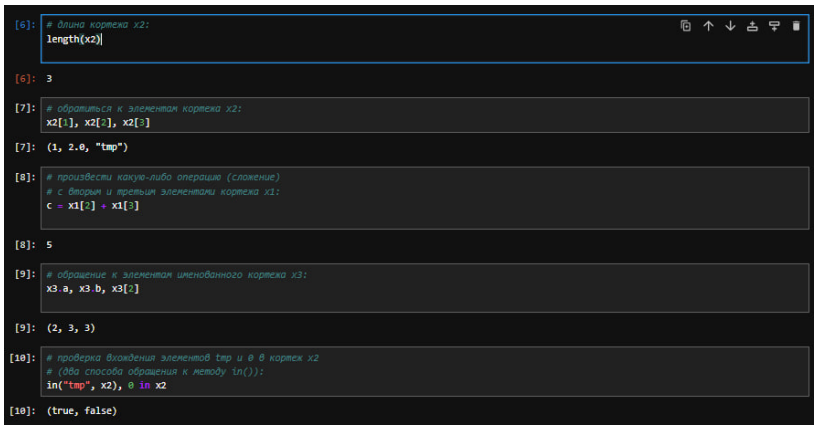

Рисунок 3.2: Примеры использования кортежей

Теперь выполним примеры из раздела про словари. Словарь – неупорядоченный набор связанных между собой по ключу данных.

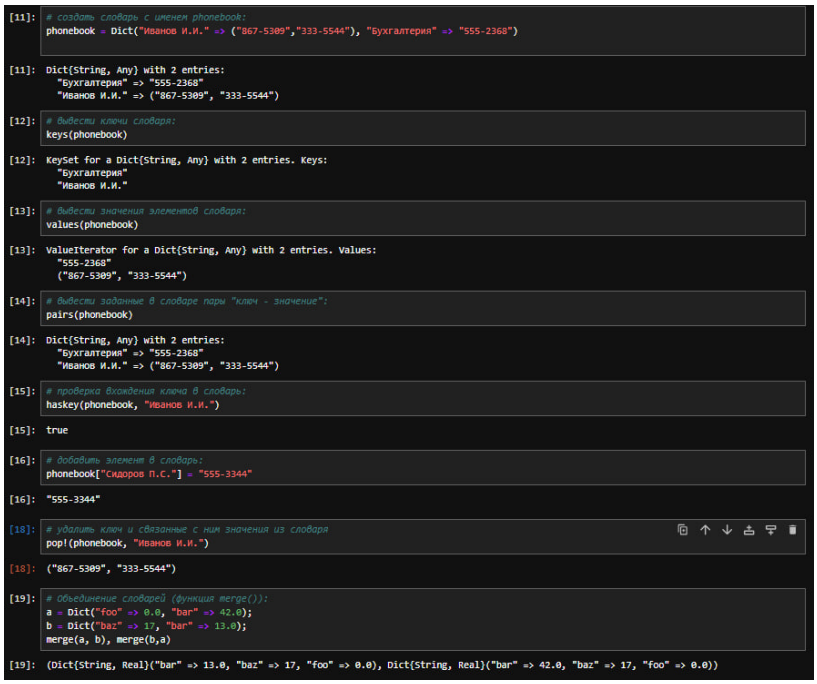

Рисунок 3.3: Примеры использования словарей

Выполним примеры из раздела про множества. Множество, как структура данных в Julia, соответствует множеству, как математическому объекту, то есть является неупорядоченной совокупностью элементов какого-либо типа.

Возможные операции над множествами: объединение, пересечение, разность; принадлежность элемента множеству.

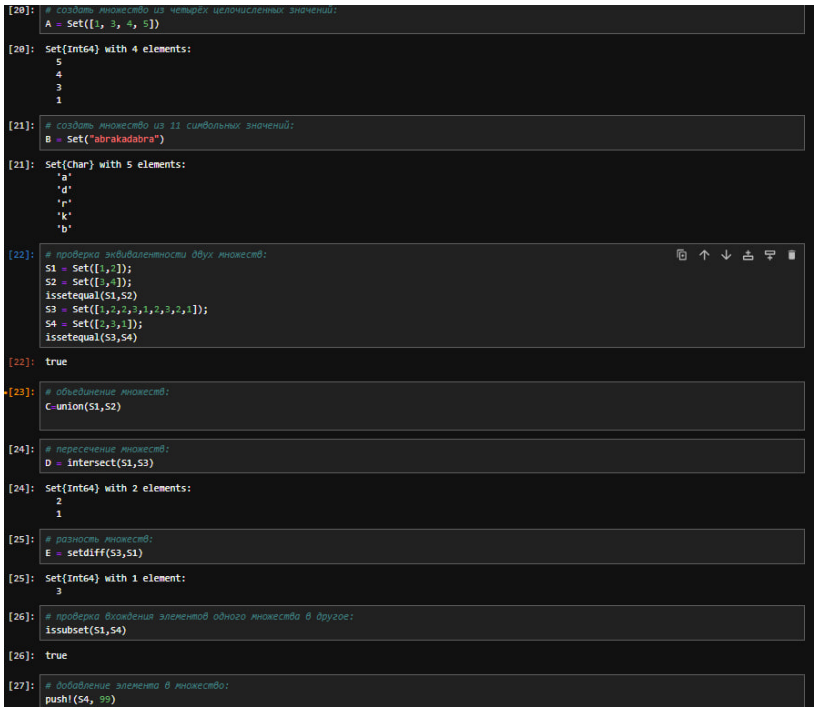

Рисунок 3.4: Примеры использования множеств

Выполним примеры из раздела про массивы. Массив — коллекция упорядоченных элементов, размещённая в многомерной сетке. Векторы и матрицы являются частными случаями массивов.

```
empty_array_1 = []
```

```
Any[]
```

```
# создание пустого массива с конкретным типом:
empty_array_2 = (Int64)[]
empty_array_3 = (Float64)[]
```

```
Float64[]
```

```
# вектор-столбец:
a = [1, 2, 3]
```

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

```
# вектор-строка:
b = [1 2 3]
```

```
1×3 Matrix{Int64}:
 1  2  3
```

```
# многомерные массивы (матрицы):
A = [[1, 2, 3] [4, 5, 6] [7, 8, 9]]
B = [[1 2 3]; [4 5 6]; [7 8 9]]
```

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

```
# одномерный массив из 8 элементов (массив $1 \times 8$)
# со значениями, случайно распределёнными на интервале [0, 1):
c = rand(1,8)
```

```
1×8 Matrix{Float64}:
 0.37157  0.868185  0.780679  0.952377  …  0.383572  0.442995  0.450085
```

```
# многомерный массив $2 \times 3$ (2 строки, 3 столбца) элементов
# со значениями, случайно распределёнными на интервале [0, 1):
C = rand(2,3);
```

```
# трёхмерный массив:
D = rand(4, 3, 2)
```

```
4×3×2 Array{Float64, 3}:
[:, :, 1] =
 0.101011  0.862076   0.188035
 0.691591  0.886666   0.505946
 0.307612  0.0336982  0.846561
 0.287509  0.399469   0.996038
```

Рисунок 3.5: Примеры использования массивов

```
# трёхмерный массив:
D = rand(4, 3, 2)
```

```
4×3×2 Array{Float64, 3}:
[:, :, 1] =
 0.101011  0.862076   0.188035
 0.691591  0.886666   0.505946
 0.307612  0.0336982  0.846561
 0.287509  0.399469   0.996038

[:, :, 2] =
 0.84059   0.697702  0.0540585
 0.673859  0.447209  0.231141
 0.207827  0.224459  0.735297
 0.809726  0.465222  0.481809
```

```
# массив из квадратных корней всех целых чисел от 1 до 10:
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
```

```
# массив с элементами вида 3*x^2,
# где x - нечётное число от 1 до 9 (включительно)
ar_1 = [3*i^2 for i in 1:2:9]
```

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

```
# массив квадратов элементов, если квадрат не делится на 5 или 4:
ar_2=[i^2 for i=1:10 if (i^2%5!=0 && i^2%4!=0)]
```

```
4-element Vector{Int64}:
  1
  9
 49
 81
```

Рисунок 3.6: Примеры использования массивов

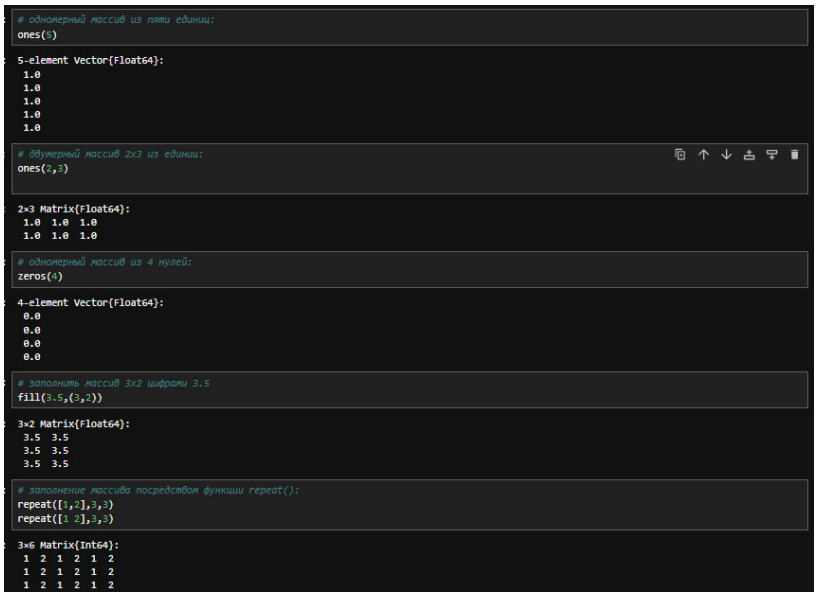

Рисунок 3.7: Примеры использования массивов

Перейдем к выполнению заданий. 1. Даны множества: $A = 0, 3, 4, 9$, $B = 1, 3, 4, 7$, $C = 0, 1, 2, 4, 7, 8, 9$. Найдем $P = A \cap B \cup A \cap C \cup B \cap C \cup A \cap B \cup A \cap C$

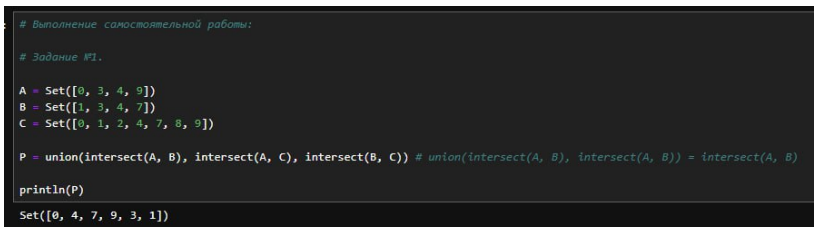

Рисунок 3.8: Задание №1. Работа с множествами

Приведем свои примеры с выполнением операций над множествами элементов разных типов

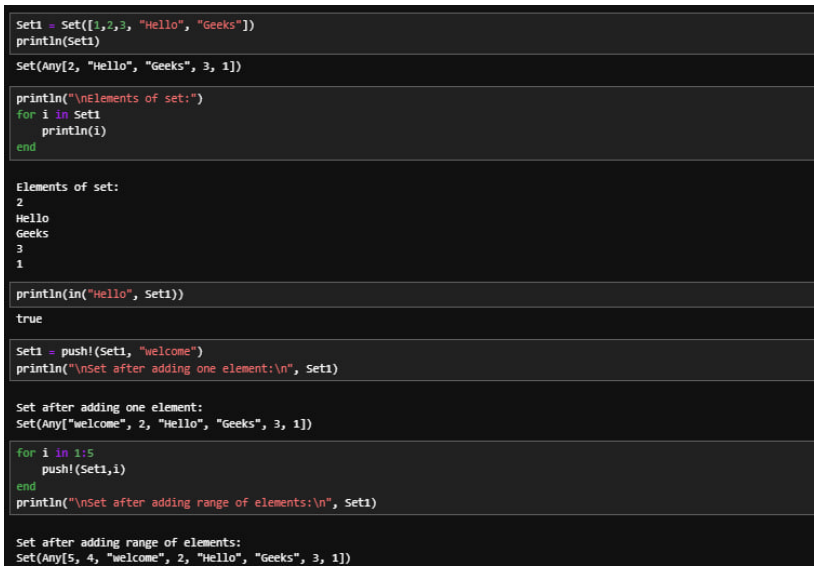

Рисунок 3.9: Задание №2. Примеры операций над множествами элементов разных типов

Создадим массивы разными способами, используя циклы

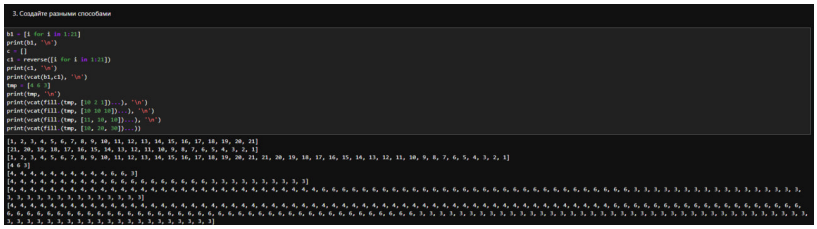

Рисунок 3.10: Задание №3. Работа с массивами

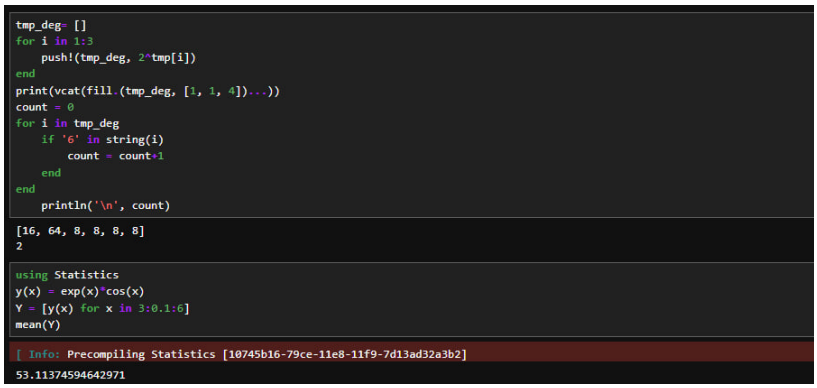

Рисунок 3.11: Задание №3. Работа с массивами

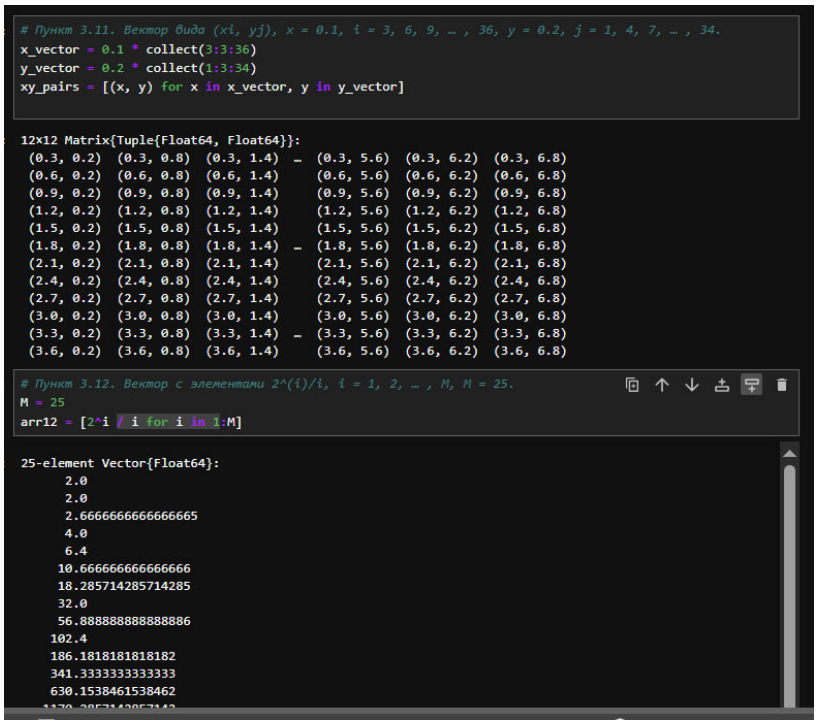

Рисунок 3.12: Задание №3. Работа с массивами

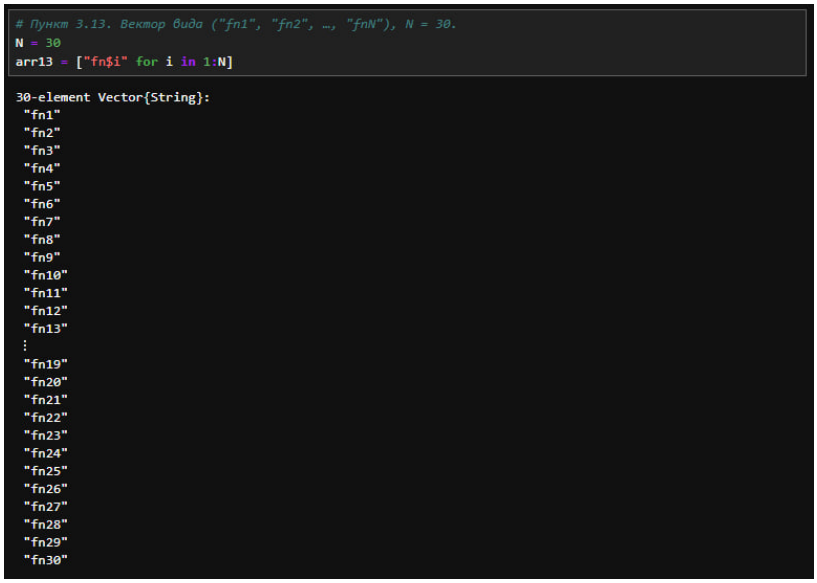

Рисунок 3.13: Задание №3. Работа с массивами

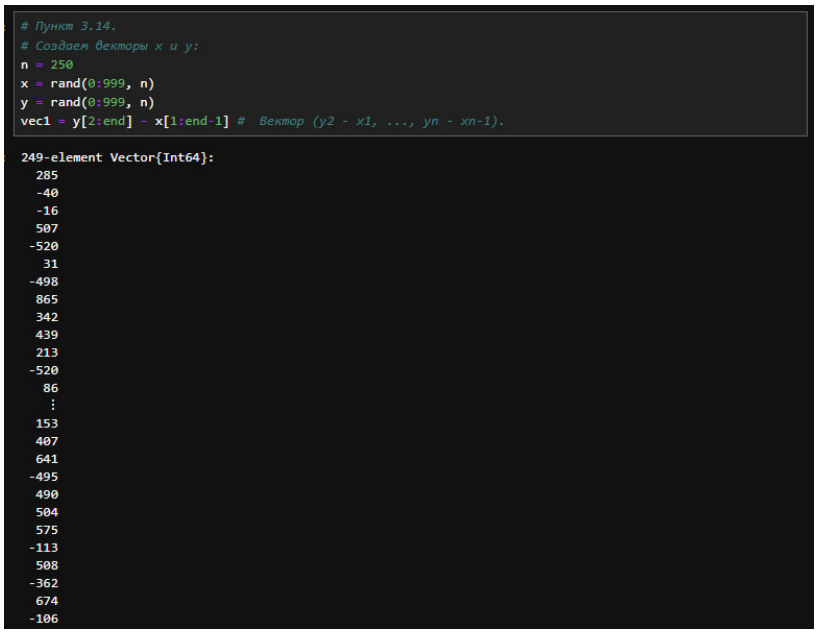

Рисунок 3.14: Задание №3. Работа с векторами

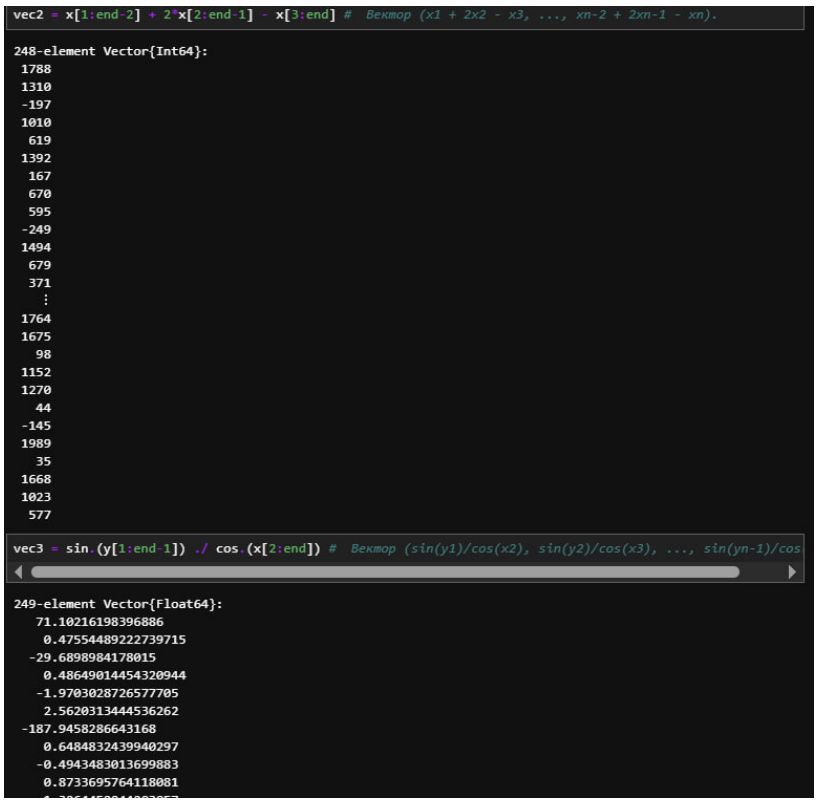

Рисунок 3.15: Задание №3. Работа с векторами

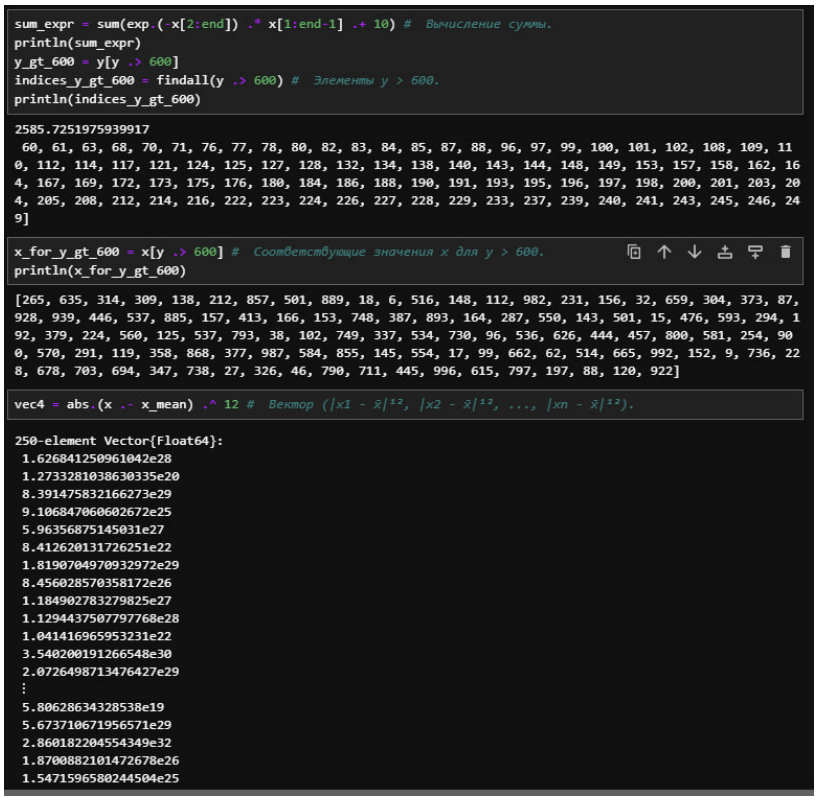

Рисунок 3.16: Задание №3. Работа с векторами

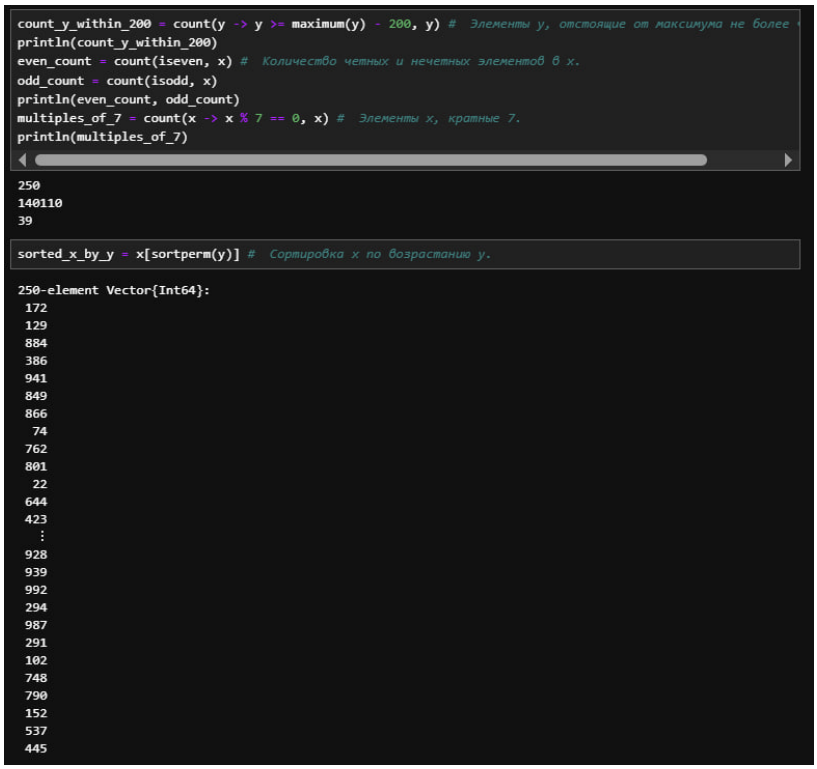

Рисунок 3.17: Задание №3. Работа с векторами

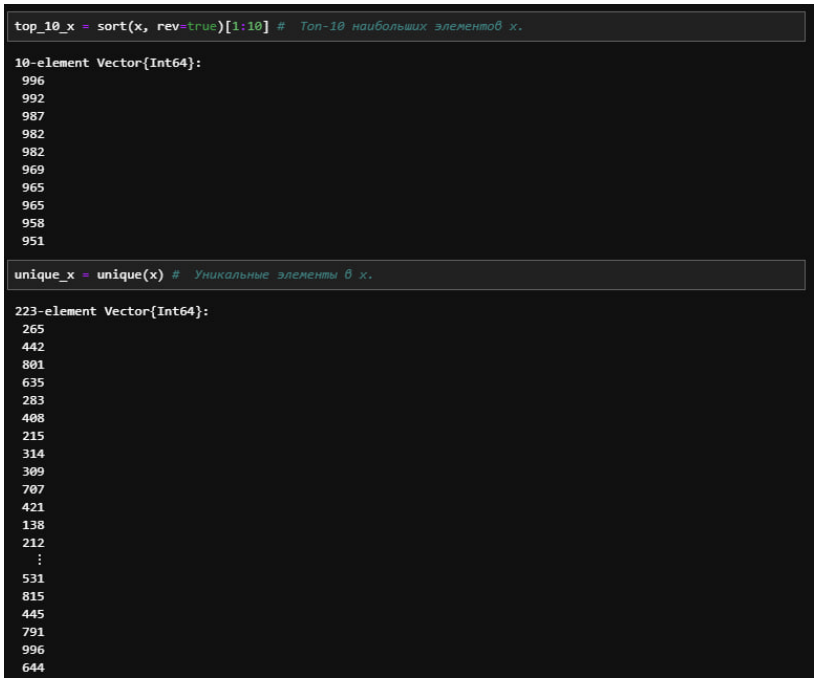

Рисунок 3.18: Задание №3. Работа с векторами

Создадим массив squares, в котором будут храниться квадраты всех целых чисел от 1 до 100

Подключим пакет Primes (функции для вычисления простых чисел). Сгенерируем массив myprimes, в котором будут храниться первые 168 простых чисел. Определим 89-е наименьшее простое число. Получии срез массива с 89-го до 99-го элемента включительно, содержащий наименьшие простые числа

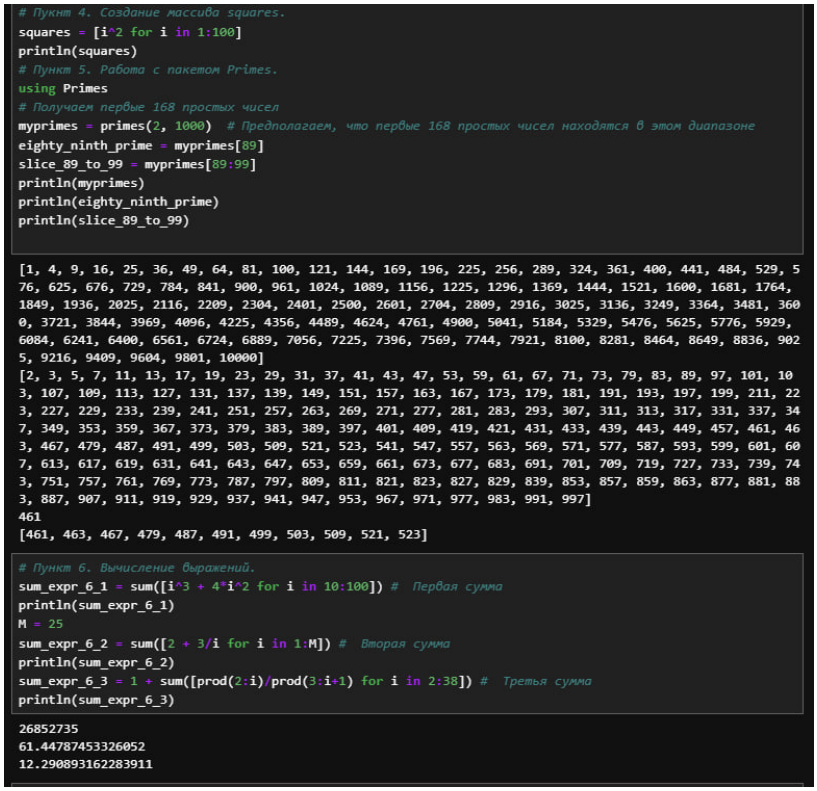

Рисунок 3.19: Задание №4, №5 и №6

4 Выводы

Мы изучили несколько структур данных, реализованных в Julia, научились применять их и операции над ними для решения задач.