Лабораторная работа № 2

Структуры данных

Оразгелдиев Язгелди

2025-09-27

Содержание I

Информация

Докладчик

- Оразгелдиев Язгелди

Докладчик

- Оразгелдиев Язгелди
- 1032225075@pfur.ru

Докладчик

- Оразгелдиев Язгелди
- 1032225075@pfur.ru
- https://github.com/YazgeldiOrazgeldiyev

Цели

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

Задачи

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

Содержание исследования

Рисунок 1: Примеры использования кортежей

Содержание исследования

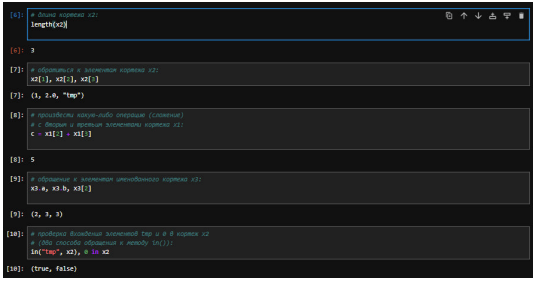

Рисунок 2: Примеры использования кортежей

Содержание исследования

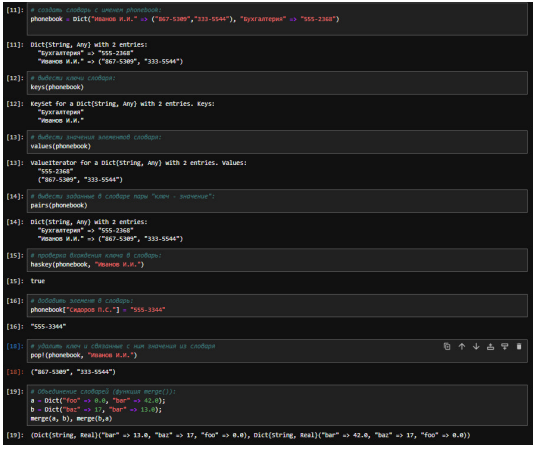

Рисунок 3: Примеры использования словарей

Содержание исследования

```
[20]: # создать множество из четырех целочисленных значений:
      A = Set([1, 2, 4, 5])

[20]: Set{Int64} with 4 elements:
        5
        4
        2
        1

[21]: # создать множество из 11 символьных значений:
      B = Set("abrakadabra")

[21]: Set{Char} with 5 elements:
        'a'
        'd'
        'r'
        'k'
        'b'

[22]: # проверка эквивалентности двух множеств:
      S1 = Set([1,2]);
      S2 = Set([1,2,1]);
      issetequal(S1,S2)
      S3 = Set([1,2,2,2,1,2,1,2,1]);
      S4 = Set([1,2,1]);
      issetequal(S3,S4)

[22]: true

[23]: # объединение множеств:
      C=union(S1,S2)

[24]: # пересечение множеств:
      D = intersect(S1,S3)

[24]: Set{Int64} with 2 elements:
        2
        1

[25]: # разность множеств:
      E = setdiff(S3,S1)

[25]: Set{Int64} with 1 element:
        3

[26]: # проверка вхождения элементов одного множества в другое:
      issubset(S1,S4)

[26]: true

[27]: # добавление элемента в множество:
      push!(S4, 99)
```

Рисунок 4: Примеры использования множеств

Содержание исследования

```
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 8\times 1)
# 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 8\times 16 (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   0.585946
 0.307612  0.0336902  0.846561
 0.287569  0.399469   0.956038
```

Рисунок 5: Примеры использования массивов

Содержание исследования

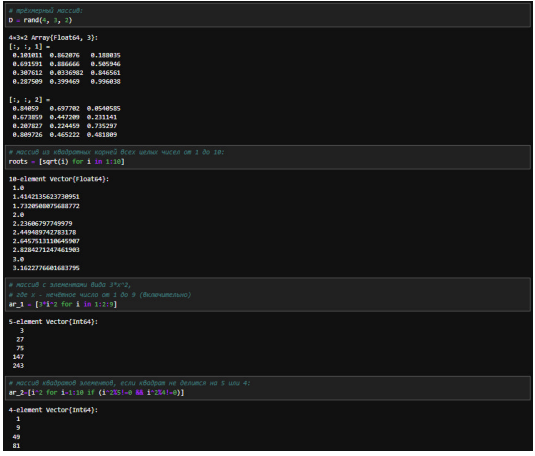

Рисунок 6: Примеры использования массивов

Содержание исследования

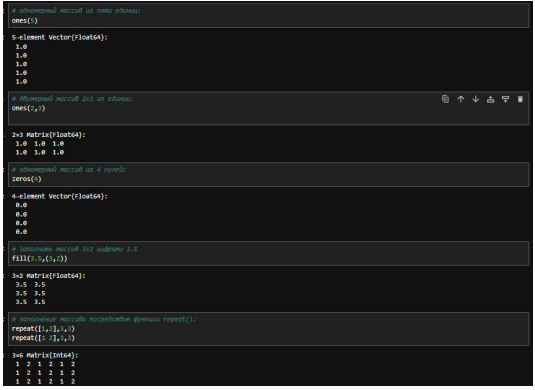

Рисунок 7: Примеры использования массивов

Содержание исследования

```
# Выполнение самостоятельной работы:

# Задание №1.

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, C), intersect(B, C)) # union(intersect(A, B), intersect(A, B)) = intersect(A, B)

println(P)
```

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

Рисунок 8: Задание №1. Работа с множествами

Содержание исследования

```
Set1 = Set([1,2,3, "Hello", "Geeks"])
println(Set1)
```

```
Set(Any[2, "Hello", "Geeks", 3, 1])
```

```
println("\nElements of set:")
for i in Set1
    println(i)
end
```

```
Elements of set:
2
Hello
Geeks
3
1
```

```
println(in("Hello", Set1))
```

```
true
```

```
Set1 = push!(Set1, "welcome")
println("\nSet after adding one element:\n", Set1)
```

```
Set after adding one element:
Set(Any["welcome", 2, "Hello", "Geeks", 3, 1])
```

```
for i in 1:5
    push!(Set1,i)
end
println("\nSet after adding range of elements:\n", Set1)
```

```
Set after adding range of elements:
Set(Any[5, 4, "welcome", 2, "Hello", "Geeks", 3, 1])
```

Рисунок 9: Задание №2. Примеры операций над множествами элементов разных типов

Содержание исследования

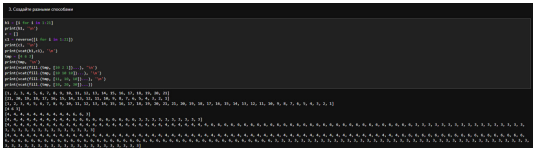

Рисунок 10: Задание №3. Работа с массивами

Содержание исследования

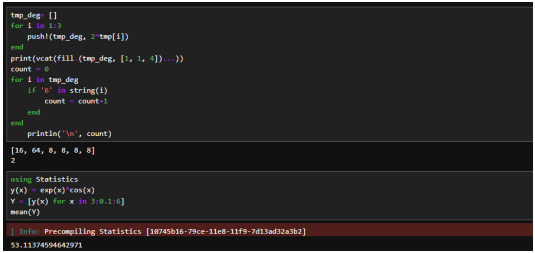

Рисунок 11: Задание №3. Работа с массивами

Содержание исследования

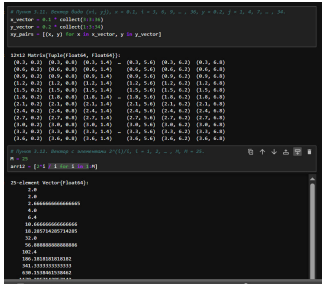

Рисунок 12: Задание №3. Работа с массивами

Содержание исследования

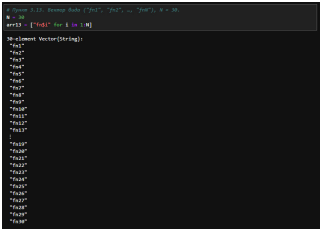

Рисунок 13: Задание №3. Работа с массивами

Содержание исследования

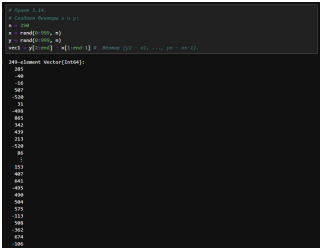

Рисунок 14: Задание №3. Работа с векторами

Содержание исследования

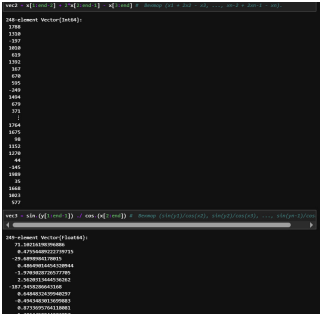

Рисунок 15: Задание №3. Работа с векторами

Содержание исследования

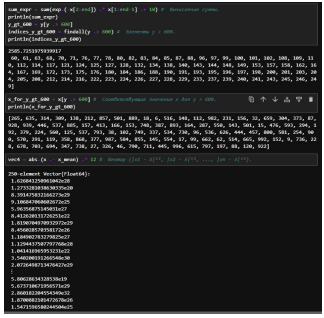

Рисунок 16: Задание №3. Работа с векторами

Содержание исследования

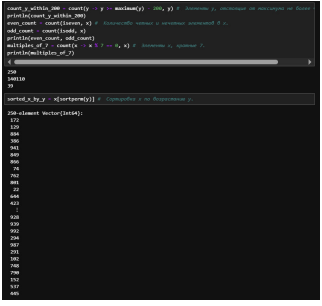

Рисунок 17: Задание №3. Работа с векторами

Содержание исследования

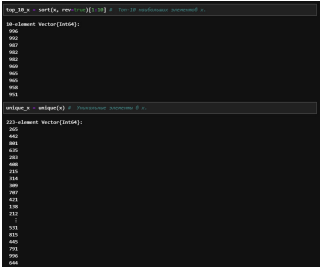

Рисунок 18: Задание №3. Работа с векторами

Содержание исследования

```
# Пукнт 4. Создание массива squares.
squares = [i^2 for i in 1:100]
println(squares)
# Пукнт 5. Работа с пакетом Primes.
using Primes
# Получаем первые 168 простых чисел
myprimes = primes(2, 1000)  # Предполагаем, что первые 168 простых чисел находятся в этом диапазоне
eighty_ninth_prime = myprimes[89]
slice_89_to_99 = myprimes[89:99]
println(myprimes)
println(eighty_ninth_prime)
println(slice_89_to_99)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441, 484, 529, 5
76, 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, 360
0, 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, 902
5, 9216, 9409, 9604, 9801, 10000]
[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, 10
3, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 22
3, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 34
7, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 46
3, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 60
7, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 74
3, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 88
3, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]
461
[461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]
```

```
# Пункт 6. Вычисление выражений.
sum_expr_6_1 = sum([i^3 + 4*i^2 for i in 10:100]) #  Первая сумма
println(sum_expr_6_1)
M = 25
sum_expr_6_2 = sum([2 + 3/i for i in 1:M]) #  Вторая сумма
println(sum_expr_6_2)
sum_expr_6_3 = 1 + sum([prod(2:i)/prod(3:i+1) for i in 2:38]) #  Третья сумма
println(sum_expr_6_3)
```

Результаты

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