Лабораторная работа №2

Структуры данных

Легиньких Г.А.

Российский университет дружбы народов, Москва, Россия

Информация

Докладчик

- Легиньких Галина Андреевна
- НФИбд-02-21
- Российский университет дружбы народов
- 1032216447@pfur.ru
- https://github.com/galeginkikh

Основная информация

Цель работы

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

Задание

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

Выполнение

Кортежи

Повторила примеры из документа

[2]:
```
# пустой кортеж:
()
```

[2]:
```
()
```

[4]:
```
# кортеж из элементов типа String:
favoritelang = ("Python","Julia","R")
```

[4]:
```
("Python", "Julia", "R")
```

[6]:
```
# кортеж из целых чисел:
x1 = (1, 2, 3)
```

[6]:
```
(1, 2, 3)
```

[8]:
```
# кортеж из элементов разных типов:
x2 = (1, 2.0, "tmp")
```

[8]:
```
(1, 2.0, "tmp")
```

[10]:
```
# именованный кортеж:
x3 = (a=2, b=1+2)
```

[10]:
```
(a = 2, b = 3)
```

Кортежи

```
[12]: # длина кортежа x2:
      length(x2)

[12]: 3

[20]: # обратиться к элементам кортежа x2:
      x2[1], x2[2], x2[3]

[20]: (1, 2.0, "tmp")

[22]: # с вторым и третьим элементами кортежа x1:
      c = x1[2] + x1[3]

[22]: 5

[24]: # обращение к элементам именованного кортежа x3:
      x3.a, x3.b, x3[2]

[24]: (2, 3, 3)

[26]: in("tmp", x2)

[26]: true

[28]: 0 in x2

[28]: false
```

Рис. 2: Операции над кортежами

Словари

Далее перешла к словарям

```
[30]: # создать словарь с именем phonebook:
phonebook = Dict("Иванов И.И." => ("867-5309","333-5544"),
    "Бухгалтерия" => "555-2368")
```

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

```
[32]: # вывести ключи словаря:
keys(phonebook)
```

```
[32]: KeySet for a Dict{String, Any} with 2 entries. Keys:
  "Бухгалтерия"
  "Иванов И.И."
```

```
[34]: # вывести значения элементов словаря:
values(phonebook)
```

```
[34]: ValueIterator for a Dict{String, Any} with 2 entries. Values:
  "555-2368"
  ("867-5309", "333-5544")
```

```
[36]: # вывести заданные в словаре пары "ключ - значение":
pairs(phonebook)
```

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

```
[40]: # проверка вхождения ключа в словарь:
haskey(phonebook, "Иванов И.И.")
```

```
[40]: true
```

Множества и операции над ними

```
[52]: # создать множество из четырёх целочисленных значений:
      A = Set([1, 3, 4, 5])

[52]: Set{Int64} with 4 elements:
        5
        4
        3
        1

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

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

[56]: # проверка эквивалентности двух множеств:
      S1 = Set([1,2]);
      S2 = Set([3,4]);
      issetequal(S1,S2)

[56]: false

[58]: S3 = Set([1,2,2,3,1,2,3,2,1]);
      S4 = Set([2,3,1]);
      issetequal(S3,S4)

[58]: true
```


Рис. 4: Множества 1

Массивы

Повторила примеры из документа

```
[74]: # создание пустого массива с абстрактным типом:
      empty_array_1 = []

[74]: Any[]

[78]: # создание пустого массива с конкретным типом:
      empty_array_2 = (Int64)[]

[78]: Int64[]

[80]: empty_array_3 = (Float64)[]

[80]: Float64[]

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

[82]: 3-element Vector{Int64}:
       1
       2
       3

[84]: # вектор-строка:
      b = [1 2 3]

[84]: 1×3 Matrix{Int64}:
       1  2  3

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

[88]: 3×3 Matrix{Int64}:
       1  4  7
       2  5  8
       3  6  9

[90]: B = [[1 2 3]; [4 5 6]; [7 8 9]]

[90]: 3×3 Matrix{Int64}:
       1  2  3
       4  5  6
       7  8  9
```

Самостоятельное задание

Перешла к выполнению самостоятельных заданий. Дублировать все задания я не буду, они есть в отчете и их много. Я просто буду придерживаться их нумерации. Объясню по одному примеру на каждый вид структур.

Задание 1 и 2

Это работа с множествами

[160]:
```
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 = $P")
```

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

[176]:
```
A = Set([0,3,4,9,"a"])
B = Set(["a","b","c",3])
C = union(A,B)
println("C = $C")
D = intersect(A,B)
println("D = $D")
```

```
C = Set(Any[0, 4, "c", 9, "b", 3, "a"])
D = Set(Any["a", 3])
```

Рис. 6: Задания 1 и 2

Задание 3.12 - 3.13

Это работа с массивами

- Задания 3.12 - 3.13 (рис. (**fig:021?**))

Задание 3.8 - 3.10

[340]:
```
# Задание 8
count_1 = count(x_i -> x_i % 2 != 0, x)
count_2 = count(x_i -> x_i % 2 == 0, x)
println("Четных: $count_2. Нечетных: $count_2")
```

```
Четных: 147. Нечетных: 147
```

[348]:
```
# Задание 9
x_7 = count(x_i -> x_i % 7 == 0, x)
println("Количество элементов вектора x кратны 7: $x_7")
```

```
Количество элементов вектора x кратны 7: 36
```

[362]:
```
# Задание 10
s = sortperm(y)
x_by_y = x[s]
println("Элементы вектора x в порядке возрастания элементов вектора y: $x_by_y")
```

```
Элементы вектора x в порядке возрастания элементов вектора y: [902, 485, 843, 821, 228, 70
0, 82, 715, 812, 541, 504, 176, 766, 167, 939, 188, 616, 713, 692, 657, 142, 215, 273, 51
9, 182, 614, 753, 320, 983, 626, 913, 151, 699, 250, 732, 300, 192, 80, 44, 17, 103, 1, 18
5, 746, 477, 66, 804, 171, 912, 601, 566, 420, 25, 807, 100, 69, 17, 129, 411, 922, 676, 4
76, 687, 211, 756, 150, 203, 546, 510, 242, 871, 966, 54, 128, 19, 834, 388, 559, 282, 74
6, 508, 609, 238, 358, 465, 97, 826, 338, 618, 760, 703, 858, 16, 32, 858, 724, 964, 107,
666, 178, 874, 171, 580, 805, 671, 461, 778, 153, 616, 687, 979, 989, 348, 380, 210, 61, 6
45, 394, 936, 406, 408, 758, 35, 273, 938, 449, 247, 274, 794, 261, 581, 302, 661, 500, 73
7, 662, 722, 614, 954, 428, 990, 386, 328, 360, 92, 226, 184, 728, 702, 714, 470, 374, 13
6, 630, 148, 832, 357, 255, 491, 342, 628, 642, 532, 816, 284, 690, 914, 938, 96, 828, 42
0, 192, 888, 592, 574, 643, 157, 627, 750, 219, 547, 279, 272, 1, 887, 750, 858, 807, 993,
931, 988, 215, 790, 684, 425, 388, 44, 369, 211, 589, 589, 873, 418, 107, 56, 727, 724, 86
4, 655, 487, 287, 598, 79, 752, 932, 44, 859, 765, 173, 115, 348, 925, 680, 106, 597, 644,
297, 555, 981, 933, 156, 732, 544, 157, 771, 911, 262, 463, 126, 279, 706, 916, 464, 100,
734, 618, 786, 776, 409, 985]
```

Рис. 7: Задание 3.14.8-3.14.10

Задание 5

Это работа с пакетом Primes

```
[411]: using Primes

[423]: myprimes = primes(1, 10000)[1:168]
       println("89-е наименьшее простое число: $(myprimes[89])")
       println("Срез массива с 89-го до 99-го элемента: $(myprimes[89:99])")

       89-е наименьшее простое число: 461
       Срез массива с 89-го до 99-го элемента: [461, 463, 467, 479, 487, 491, 499, 503, 509, 521,
       523]
```

Рис. 8: Задание 5

Вывод

Вывод

Изучила несколько структур данных, реализованных в Julia, научилась применять их и операции над ними для решения задач.