Разнообразие задач в Julia: от множеств до простых чисел

Операции над множествами

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

[11]: (1, 2.0, "tmp")

[12]: # произвести какую-либо операцию (сложение)
      # с вторым и третьим элементами кортежа x1:
      c = x1[2] + x1[3]

[12]: 5

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

[13]: (2, 3, 3)

[14]: # проверка вхождения элементов tmp и 0 в кортеж x2
      # (два способа обращения к методу in()):
      in("tmp", x2), 0 in x2

[14]: (true, false)

[16]: # Словари:

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

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

[17]: # вывести ключи словаря:
      keys(phonebook)

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

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

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

[19]: # вывести заданные в словаре пары "ключ - значение":
```

Массивы и их манипуляции

[80]:
```
# Задание №3.

# Пункт 3.1. Массив от 1 до N.
N = 25
arr1 = collect(1:N)
println(arr1)
# Пункт 3.2. Массив от N до 1.
arr2 = collect(N:-1:1)
println(arr2)
# Пункт 3.3. Массив от 1 до N, затем обратно от N до 1.
arr3 = [collect(1:N); collect(N:-1:1)]
println(arr3)
# Пункт 3.4. Массив tmp = (4, 6, 3).
tmp = [4, 6, 3]
println(tmp)
# Пункт 3.5. Массив, где первый элемент tmp повторяется 10 раз.
arr5 = fill(tmp[1], 10)
println(arr5)
# Пункт 3.6. Массив, где все элементы tmp повторяются 10 раз.
arr6 = repeat(tmp, 10)
println(arr6)
```

```
[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]
[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]
[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, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 1
[4, 6, 3]
[4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
[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, 4, 6, 3]
```

[86]:
```
# Пункт 3.7. Массив, где первый элемент tmp встречается 11 раз, остальные – 10 раз.
arr7 = [fill(tmp[1], 11); fill(tmp[2], 10); fill(tmp[3], 10)]
println(arr7)
# Пункт 3.8. Массив, где элементы tmp встречаются 10, 20 и 30 раз подряд соответственно.
arr8 = [fill(tmp[1], 10); fill(tmp[2], 20); fill(tmp[3], 30)]
println(arr8)
# Пункт 3.9. Массив из элементов 2tmp[i], i = 1, 2, 3, элемент 2tmp[3] встречается 4 раза; количество 6.
arr9 = [2*tmp[1], 2*tmp[2], fill(2*tmp[3], 4)...]
count_6s = count(x -> x == 6, arr9)
println("Number of 6's: ", count_6s)
# Пункт 3.10. Вектор значений y = e^(x)*cos(x) в точках x = 3,3.1,3.2,…,6; среднее значение y.
using Statistics
x_values = 3:0.1:6
y_values = [exp(x) * cos(x) for x in x_values]
mean_y = mean(y_values)
println(mean_y)
```

Использование пакета Primes

[116]:
```
# Пункт 4. Создание массива squares.
squares = [i^2 for i in 1:100]
println(squares)
# Пункт 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, 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]
[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]
461
[461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]
```

Вычислительные задачи

[117]:
```
# 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)
```

```
26852735
61.44787453326052
12.290893162283911
```

[ ]:

Проблемы и их решения

[111]:
```
# Пункт 4. Создание массива squares.
squares = [i^2 for i in 1:100]
println(squares)
# Пункт 5. Работа с пакетом Primes.

using Primes
myprimes = primes(168)
89th_prime = myprimes[89]
slice_89_to_99 = myprimes[89:99]
println(myprimes)
println(89th_prime)
println(slice_89_to_99)
# 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)
```

```
[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, 1158
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
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, 1000
syntax: "89" is not a valid function argument name around In[111]:8

Stacktrace:
 [1] top-level scope
   @ In[111]:8
```

Заключение

“Спасибо за внимание!”