

Защита лабораторной работы № 2

Структуры данных.

Компьютерный практикум по статистическому анализу данных

Работу Выполнил:
Саинт-Амур Измаэль
Группа: НПИбд-01-20

Цель работы

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

Предварительные сведения

Рассмотрим 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

• `isempty()` Пример использования

```
In [1]: isempty([1, 3, 4, 7,9])
```

```
Out[1]: false
```

```
In [2]: isempty([])
```

```
Out[2]: true
```

- `length()` Пример использования

```
In [3]: length([1, 4, 5, 6, 9])
```

```
Out[3]: 5
```

```
In [13]: reduce(-, [1, 2, 3, 4, 8, 9])
```

```
Out[13]: -25
```

```
In [14]: reduce(*, [1, 2, 3, 4, 8, 9])
```

```
Out[14]: 1728
```

• `maximum()` Пример использования

```
In [16]: maximum([0, 8, 5, 4, 9, 11])
```

```
Out[16]: 11
```

• `minimum()` Пример использования

```
In [19]: minimum([0, 8, 5, 4, 9, 11])
```

```
Out[19]: 0
```

Кортеж (Tuple) — структура данных (контейнер) в виде неизменяемой индексируемой последовательности элементов какого-либо типа (элементы индексируются с единицы).

```
In [99]: # кортеж из элементов типа String:  
favoritelang = ("Python","Julia","R")
```

```
Out[99]: ("Python", "Julia", "R")
```

```
In [100]: # кортеж из целых чисел:  
x1 = (1, 2, 3)
```

```
Out[100]: (1, 2, 3)
```

```
In [101]: # кортеж из элементов разных типов:  
x2 = (1, 2.0, "tmp")
```

```
Out[101]: (1, 2.0, "tmp")
```

```
In [102]: # именованный кортеж:  
x3 = (a=2, b=1+2)
```

```
Out[102]: (a = 2, b = 3)
```

Словарь — неупорядоченный набор связанных между собой по ключу данных. Синтаксис определения словаря:

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

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

```
# Вывести ключи словаря:  
keys(phonebook)
```

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

```
# Вывести значения элементов словаря:  
values(phonebook)
```

```
ValueIterator for a Dict{String, Any} with 2 entries. Values:  
  "555-2368"
```

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.

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

Set{Int64} with 4 elements:

5
4
3
1

```
# создать множество из 11 символьных значений:  
B = Set("abracadabra")
```

Set{Char} with 5 elements:

'a'
'd'
'r'
'k'
'b'

Массив — коллекция упорядоченных элементов, размещённая в многомерной сетке. Векторы и матрицы являются частными случаями массивов.

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

Any[]

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

Int64[]

```
empty_array_3 = (Float64)[]
```

Float64[]

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

3-element Vector{Int64}:

1
2
3

параметр `outer` указывает количество повторений для всего массива `tmp`. Первый элемент повторяется 10 раз подряд, второй 20 раз подряд, и третий 30 раз подряд

используется функция `fill`, чтобы создать массив, содержащий 10 копий этого значения. `repeated` будет содержать все элементы массива `tmp`, повторенные 10 раз. параметр `inner`, где каждое число представляет количество повторений для соответствующего элемента массива `tmp`. Таким образом, первый элемент повторяется 11 раз, второй 10 раз, и третий также 10 раз.

```
In [139]: tmp = [4, 6, 3]
a = repeat(tmp, outer=[11, 20, 30])
```

```
Out[139]: 33x20x30 Array{Int64, 3}:
[:, :, 1] =
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

```
In [66]: tmp = [4, 6, 3]
```

```
Out[66]: 3-element Vector{Int64}:
 4
 6
 3
```

- 3.5) массив, в котором первый элемент массива `tmp` повторяется 1

```
In [67]: a = 4
tmp = fill(a, 10)
```

```
Out[67]: 10-element Vector{Int64}:
 4
 4
 4
 4
 4
 4
 4
 4
 4
 4
```


Установка и использование пакета Primes

установить и использовать пакет Primes.
я установил его с помощью команды:

```
using Pkg
Pkg.add("Primes")
```

После установки пакета, я использовал его для генерации массива первых 168 простых чисел. Этот код использует функцию `primes` из пакета Prime для генерации первых 168 простых чисел.

```
using Pkg
```

```
Pkg.add("Primes")
```

```
Resolving package versions...
No Changes to `C:\Users\Scorpion 1.0\.julia\environments\v1.9\Project.toml`
No Changes to `C:\Users\Scorpion 1.0\.julia\environments\v1.9\Manifest.toml`
```

```
# Подключение пакета Primes
```

```
using Primes
```

```
# Генерация массива myprimes с первыми 168 простыми числами
myprimes = primes(1000)[1:168]
```

```
# Вывод 89-го наименьшего простого числа
println("89-е наименьшее простое число: ", myprimes[89])
```

```
# Срез массива с 89-го до 99-го элемента включительно
slice_of_primes = myprimes[89:99]
```

```
# Вывод среза массива
println("Срез с 89-го до 99-го элемента включительно: ", slice_of_primes)
```

```
89-е наименьшее простое число: 461
```

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

В ходе работы по изучению структур данных в Julia и применению их для решения задач были рассмотрены следующие моменты:

- Массивы и Операции:
- Векторы и Операции:
- Установка и использование пакета Primes для работы с простыми числами.