Лабораторная работа № 2. Структуры данных

2.1. Цель работы

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

2.2. Предварительные сведения

Рассмотрим несколько структур данных, реализованных в Julia.

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.

2.2.1. Кортежи

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

Синтаксис определения кортежа:

```
(element1, element2, ...)
```

Примеры кортежей:

```
# пустой кортеж:
()
# кортеж из элементов типа String:
favoritelang = ("Python","Julia","R")
# кортеж из целых чисел:
x1 = (1, 2, 3)
# кортеж из элементов разных типов:
x2 = (1, 2.0, "tmp")
# именованный кортеж:
x3 = (a=2, b=1+2)
```

Примеры операций над кортежами:

```
# длина кортежа x2:
length(x2)
# обратиться к элементам кортежа x2:
x2[1], x2[2], x2[3]
# произвести какую-либо операцию (сложение)
# с вторым и третьим элементами кортежа x1:
c = x1[2] + x1[3]
# обращение к элементам именованного кортежа x3:
x3.a, x3.b, x3[2]
# проверка вхождения элементов tmp и 0 в кортеж x2
# (два способа обращения к методу in()):
in("tmp", x2), 0 in x2
```

2.2.2. Словари

Словарь — неупорядоченный набор связанных между собой по ключу данных.
Синтаксис определения словаря:
```
Dict(key1 => value1, key2 => value2, ...)
```
Примеры словарей и операций над ними:
```
# создать словарь с именем phonebook:
phonebook = Dict("Иванов И.И." => ("867-5309","333-5544"),
 ↪  "Бухгалтерия" => "555-2368")
# вывести ключи словаря:
keys(phonebook)
# вывести значения элементов словаря:
values(phonebook)
# вывести заданные в словаре пары "ключ - значение":
pairs(phonebook)
# проверка вхождения ключа в словарь:
haskey(phonebook, "Иванов И.И.")
# добавить элемент в словарь:
phonebook["Сидоров П.С."] = "555-3344"
# удалить ключ и связанные с ним значения из словаря
pop!(phonebook, "Иванов И.И."):

# Объединение словарей (функция merge()):
a = Dict("foo" => 0.0, "bar" => 42.0);
b = Dict("baz" => 17, "bar" => 13.0);
merge(a, b), merge(b,a)
```

2.2.3. 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.
Синтаксис определения множества:
```
Set([itr])
```
где `itr` — набор значений, сгенерированных данным итерируемым объектом или пустое множество.

Примеры множеств и операций над ними:
```
# создать множество из четырёх целочисленных значений:
A = Set([1, 3, 4, 5])
# создать множество из 11 символьных значений:
B = Set("abrakadabra")
# проверка эквивалентности двух множеств:
S1 = Set([1,2]);
S2 = Set([3,4]);
issetequal(S1,S2)

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

# объединение множеств:
```

```
C=union(S1,S2)
# пересечение множеств:
D = intersect(S1,S3)
# разность множеств:
E = setdiff(S3,S1)
# проверка вхождения элементов одного множества в другое:
issubset(S1,S4)
# добавление элемента в множество:
push!(S4, 99)
# удаление последнего элемента множества:
pop!(S4)
```

2.2.4. Массивы

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

Общий синтаксис одномерных массивов:

```
array_name_1 = [element1, element2, ...]
array_name_2 = [element1 element2 ...]
```

Примеры массивов:

```
# создание пустого массива с абстрактным типом:
empty_array_1 = []
# создание пустого массива с конкретным типом:
empty_array_2 = (Int64)[]
empty_array_3 = (Float64)[]

# вектор-столбец:
a = [1, 2, 3]

# вектор-строка:
b = [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]]

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

# 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)
```

Примеры массивов, заданных некоторыми функциями через включение:

```
# массив из квадратных корней всех целых чисел от 1 до 10:
roots = [sqrt(i) for i in 1:10]

# 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]

# 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)]
```

Некоторые операции для работы с массивами:

- `length(A)` — число элементов массива A;
- `ndims(A)` — число размерностей массива A;
- `size(A)` — кортеж размерностей массива A;
- `size(A, n)` — размерность массива A в заданном направлении;
- `copy(A)` — создание копии массива A;
- `ones()`, `zeros()` — создать массив с единицами или нулями соответственно;
- `fill(value,array_name)` — заполнение массива заранее определенным значением;
- `sort()` — сортировка элементов;
- `collect()` — вернуть массив всех элементов в коллекции или итераторе;
- `reshape()` — изменение размера массива;
- `transpose()` — транспонирование массива;

Несколько примеров:

```
# одномерный массив из пяти единиц:
ones(5)
# двумерный массив 2x3 из единиц:
ones(2,3)
# одномерный массив из 4 нулей:
zeros(4)
# заполнить массив 3x2 цифрами 3.5
fill(3.5,(3,2))
# заполнение массива посредством функции repeat():
repeat([1,2],3,3)
repeat([1 2],3,3)

# преобразование одномерного массива из целых чисел от 1 до 12
# в двумерный массив  2x6
a = collect(1:12)
b = reshape(a,(2,6))

# транспонирование
b'
# транспонирование
c = transpose(b)

#  массив 10x5 целых чисел в диапазоне [10, 20]:
ar = rand(10:20, 10, 5)
# выбор всех значений строки в столбце 2:
ar[:, 2]
# выбор всех значений в столбцах 2 и 5:
ar[:, [2, 5]]
# все значения строк в столбцах 2, 3 и 4:
ar[:, 2:4]
# значения в строках 2, 4, 6 и в столбцах 1 и 5:
ar[[2, 4, 6], [1, 5]]
# значения в строке 1 от столбца 3 до последнего столбца:
ar[1, 3:end]
# сортировка по столбцам:
```

```
sort(ar,dims=1)
# сортировка по строкам:
sort(ar,dims=2)
# поэлементное сравнение с числом
# (результат - массив логических значений):
ar .> 14
# возврат индексов элементов массива, удовлетворяющих условию:
findall(ar .> 14)
```

2.3. Задание

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

2.4. Задания для самостоятельного выполнения

1. Даны множества: $A = \{0, 3, 4, 9\}$, $B = \{1, 3, 4, 7\}$, $C = \{0, 1, 2, 4, 7, 8, 9\}$. Найти $P = A \cap B \cup A \cap B \cup A \cap C \cup B \cap C$.
2. Приведите свои примеры с выполнением операций над множествами элементов разных типов.
3. Создайте разными способами:
   3.1) массив $(1, 2, 3, \dots N-1, N)$, N выберите больше 20;
   3.2) массив $(N, N-1 \dots, 2, 1)$, N выберите больше 20;
   3.3) массив $(1, 2, 3, \dots, N-1, N, N-1, \dots, 2, 1)$, N выберите больше 20;
   3.4) массив с именем tmp вида $(4, 6, 3)$;
   3.5) массив, в котором первый элемент массива tmp повторяется 10 раз;
   3.6) массив, в котором все элементы массива tmp повторяются 10 раз;
   3.7) массив, в котором первый элемент массива tmp встречается 11 раз, второй элемент — 10 раз, третий элемент — 10 раз;
   3.8) массив, в котором первый элемент массива tmp встречается 10 раз подряд, второй элемент — 20 раз подряд, третий элемент — 30 раз подряд;
   3.9) массив из элементов вида $2^{tmp[i]}$, $i = 1, 2, 3$, где элемент $2^{tmp[3]}$ встречается 4 раза; посчитайте в полученном векторе, сколько раз встречается цифра 6, и выведите это значение на экран;
   3.10) вектор значений $y = e^x \cos(x)$ в точках $x = 3, 3.1, 3.2, \dots, 6$, найдите среднее значение y;
   3.11) вектор вида (x^i, y^j), $x = 0.1$, $i = 3, 6, 9, \dots, 36$, $y = 0.2$, $j = 1, 4, 7, \dots, 34$;
   3.12) вектор с элементами $\frac{2^i}{i}$, $i = 1, 2, \dots, M$, $M = 25$;
   3.13) вектор вида ("fn1", "fn2", ..., "fnN"), $N = 30$;
   3.14) векторы $x = (x_1, x_2, \dots, x_n)$ и $y = (y_1, y_2, \dots, y_n)$ целочисленного типа длины $n = 250$ как случайные выборки из совокупности $0, 1, \dots, 999$; на его основе:
   - сформируйте вектор $(y_2 - x_1, \dots, y_n - x_{n-1})$;
   - сформируйте вектор $(x_1 + 2x_2 - x_3, x_2 + 2x_3 - x_4, \dots, x_{n-2} + 2x_{n-1} - x_n)$;
   - сформируйте вектор $\left(\frac{\sin(y_1)}{\cos(x_2)}, \frac{\sin(y_2)}{\cos(x_3)}, \dots, \frac{\sin(y_{n-1})}{\cos(x_n)}\right)$;
   - вычислите $\sum_{i=1}^{n-1} \frac{e^{-x_{i+1}}}{x_i + 10}$;

- выберите элементы вектора y, значения которых больше 600, и выведите на экран; определите индексы этих элементов;
- определите значения вектора x, соответствующие значениям вектора y, значения которых больше 600 (под соответствием понимается расположение на аналогичных индексных позициях);
- сформируйте вектор $(|x_1 - \overline{x}|^{\frac{1}{2}}, |x_2 - \overline{x}|^{\frac{1}{2}}, \dots, |x_n - \overline{x}|^{\frac{1}{2}})$, где $\overline{x}$ обозначает среднее значение вектора $x = (x_1, x_2, \dots, x_n)$;
- определите, сколько элементов вектора y отстоят от максимального значения не более, чем на 200;
- определите, сколько чётных и нечётных элементов вектора x;
- определите, сколько элементов вектора x кратны 7;
- отсортируйте элементы вектора x в порядке возрастания элементов вектора y;
- выведите элементы вектора x, которые входят в десятку наибольших (top-10)?
- сформируйте вектор, содержащий только уникальные (неповторяющиеся) элементы вектора x.

4. Создайте массив `squares`, в котором будут храниться квадраты всех целых чисел от 1 до 100.
5. Подключите пакет `Primes` (функции для вычисления простых чисел). Сгенерируйте массив `myprimes`, в котором будут храниться первые 168 простых чисел. Определите 89-е наименьшее простое число. Получите срез массива с 89-го до 99-го элемента включительно, содержащий наименьшие простые числа.
6. Вычислите следующие выражения:

   6.1) $\sum\limits_{i=10}^{100} (i^3 + 4i^2)$;

   6.2) $\sum\limits_{i=1}^{M} \left(\frac{2^i}{i} + \frac{3^i}{i^2}\right), M = 25$;

   6.3) $1 + \frac{2}{3} + \left(\frac{2}{3}\frac{4}{5}\right) + \left(\frac{2}{3}\frac{4}{5}\frac{6}{7}\right) + \dots + \left(\frac{2}{3}\frac{4}{5}\dots\frac{38}{39}\right)$.

2.5. Содержание отчёта

1. Титульный лист с указанием номера лабораторной работы и ФИО студента.
2. Формулировка задания работы.
3. Описание выполнения задания:
   - подробное пояснение выполняемых в соответствии с заданием действий;
   - скриншоты (снимки экрана), фиксирующие выполнение лабораторной работы;
   - листинги (исходный код) программ и результаты его выполнения;
4. Выводы, согласованные с заданием работы.