MANUAL DE RUBY

(PARTE II)

Luis José Sánchez González

1. ARRAYS

Los arrays en ruby pueden almacenar objetos de diferentes tipos. Si queremos, podemos tener un número entero en una posición, una palabra en la siguiente y un carácter en la siguiente.
Al igual que con el resto de variables, no es necesario declararla, simplemente le asignamos los valores.
Podemos acceder a los elementos del array mediante el índice. Podemos extraer varios elementos al mismo tiempo de forma análoga a como lo hacemos con las cadenas de caracteres.

```
a = [24, 5, "hola", "caracola", 7]
puts a
puts a[0]
puts a[2]
puts a[-1]
puts a[3,2]
puts a[1..3]
```

Los elementos del array se pueden modificar indicando el índice.

```
b = ["rojo", "verde", "azul", "blanco"]
puts b
b[0] = "rosa"
b[-1] = "negro"
puts b
```

Los arrays se pueden concatenar y repetir, igual que las cadenas.

```
c = [8, "coche", "moto", 6]
d = ["bicicleta", "triciclo"]
transportes = c + d
puts transportes
motorizados = transportes[1,2]
puts motorizados * 2
```

Podemos ir añadiendo elementos a un array con el doble "menor que" (<<). Se puede añadir también de forma encadenada.

Por ejemplo:

```
a << 7

a << 98 << 32 << "hola"
```

Los arrays se pueden convertir a cadenas utilizando **join**:

```
a = [8, "coche", "moto", 6]
cadena = a.join(':')
puts cadena
```

Las cadenas se pueden convertir a arrays utilizando **split**:

```
frase = "Hola, ¿qué tal?, ¿cómo te va?"
mi_array = frase.split(',')
puts mi_array
```

Algunos métodos útiles:

- **length** nos da el número de elementos de un array
- **empty?** devuelve verdadero si el array está vacío
- **reverse / reverse!** le da la vuelta al array
- **sort / sort!** ordena el array (los elementos deben ser del mismo tipo)
- **compact / compact!** elimina los elementos **nil** del array
- **delete(valor)** borra todas las coincidencias del elemento **valor**
- **delete_at(pos)** borra el elemento de la posición indicada por **pos**
- **include?(valor)** devuelve verdadero si el array contiene **valor**
- **index(valor)** devuelve el índice del elemento **valor**. Si no se encuentra, se devuelve **nil.**
- **pop** devuelve el último elemento y lo elimina del array
- **shift** devuelve el primer elemento y lo elimina del array
- **uniq / uniq!** quita elementos duplicados del array
- **unshift(valor)** introduce **valor** por delante del array

2. ARRAYS ASOCIATIVAS (TABLAS HASH)

Un array asociativo contiene elementos a los que se puede acceder, no a través de índices numéricos secuenciales, sino a través de claves que pueden tener cualquier tipo de valor. Estos arrays se conocen también como **tablas hash** o **diccionarios**.

Un array asociativo se puede crear mediante pares de elementos dentro de llaves ({ }). Se usa la clave para encontrar algo en una tabla hash de la misma forma que se utiliza el índice para encontrar algo en un array normal.

Ejemplo:

```
h = { 1 => 150, "hola" => "hello", 4 => "cuatro"}
puts h
puts h[4]
puts h["hola"]
```

3. EL ITERADOR EACH

El iterador **each** de ruby proporciona una manera fácil de recorrer un array. En otros lenguajes como C hay que usar bucles **for** y crear variables de índice, además de averiguar el tamaño del array. Con **each** no tenemos que preocuparnos de nada de eso.

Ejemplo:

```
nombres = ["Elena", "María", "Clara", "Lucía"]
nombres.each do |n|
        puts "Hola #{n}, ¡qué guapa estás hoy!"
end
```

Ejercicios

1. Escribe un programa que pida una frase por teclado y que a continuación muestre cada una de las palabras de esa frase indicando al lado si es singular o plural.
2. Realiza un programa que pida un número de teléfono y que luego muestre ese teléfono de la forma: nueve cinco dos cuatro cinco, etc.
3. Escribe un programa que averigüe de qué compañía es un número de teléfono móvil introducido por teclado (se deberán buscar los prefijos en internet).
4. Realiza un programa que mezcle dos frases introducidas por teclado. Deberá aparecer la primera palabra de la primera frase, después la primera palabra de la segunda frase, la segunda palabra de la primera frase, la segunda palabra de la segunda frase, etc. Si se termina alguna de las frases, se completa con lo que quede de la otra.

   Por ejemplo, si mezclamos "tú no tienes perro" con "el lobo se comió a Caperucita" debería salir " tú el no lobo tienes se perro comió a Caperucita"
5. Escribe un programa que diga si cinco palabras introducidas por teclado (separadas por un espacio) están ordenadas.