Práctica 3: programación funcional en Scala; funciones

Nuevas tecnologías de la programación

Contenido:

1 Objetivos

En esta práctica se trabajará con una representación funcional de conjuntos basada en la noción matemática de funciones características. El objetivo de la práctica es poner en juego el concepto de funciones de alto orden (**high order functions**).

2 Representación

En la práctica se trabaja, sin pérdida de generalidad, con conjuntos de enteros. Como ejemplo motivador, pensemos en la forma de representar el conjunto de todos los enteros negativos. A la hora de definir este conjunto no tiene sentido recurrir a la enumeración de todos sus elementos, por razones obvias. Sin embargo, sí que es posible definir una característica que cumplirán todos ellos. En el caso de los números negativos la función será:

```
1 (x : Int) => x < 0
```

Siguiendo esta idea, la representación del conjunto se hará definiendo un tipo asociado a la función característica (recibe argumento de valor entero y devuelve valor booleano indicando pertenencia o no):

```
1 type Conjunto = Int => Boolean
```

Usando esta representación podemos definir una función que compruebe si un elemento determinado pertenece a un conjunto de forma sencilla:

```
1 def contiene(c : Conjunto, elemento : Int) : Boolean = c(elemento)
```

Es decir, basta con comprobar si la función característica, aplicada sobre el elemento a comprobar, devuelve verdadero.

3 Funciones básicas sobre conjuntos

Interesa dotar a esta definición de conjuntos de algunas operaciones básicas, como las siguientes:

- creación de conjunto con un único elemento. Este conjunto representa a este único elemento y su declaración será

  ```
  1 def conjuntoUnElemento(elemto : Int) : Conjunto
  ```

- unión de dos conjuntos:

  ```
  1 def union(c1 : Conjunto, c2 : Conjunto) : Conjunto
  ```

- intersección:

  ```
  1 def interseccion(c1 : Conjunto, c2 : Conjunto) : Conjunto
  ```

- diferencia:

  ```
  1 def diferencia(c1 : Conjunto, c2 : Conjunto) : Conjunto
  ```

- filtrado:

  ```
  1 def filtrar(c : Conjunto, predicado : Int => Boolean) : Conjunto
  ```

4 Funciones avanzadas sobre conjuntos

Ahora estamos interesados en funciones que permitan hacer consultas sobre todos los elementos del conjunto. Como no hay forma directa de obtener todos los elementos de un conjunto, habrá que iterar sobre un determinado rango de enteros (entre -1000 y 1000 en la versión final; conviene reducirlo para pruebas) para comprobar si pertenecen o no al conjunto de interés. Las funciones que deseamos implementar en esta sección son:

- paraTodo, que comprueba si un determinado predicado se cumple para todos los elementos del conjunto. La declaración es:

  ```
  1 def paraTodo(c : Conjunto, predicado : Int => Boolean) : Boolean
  ```

Esta función debe implementarse de forma recursiva. Para ayudar a su desarrollo se ofrece un esqueleto de la misma:

```
def paraTodo(c : Conjunto, predicado : Int => Boolean) : Boolean = {
  def iter(a : Int) : Boolean = {
     if(???) ???
     else if (???) ???
     else iter(???)
  }
  iter(???)
}
```

- usando la función anterior, implementad la función existe que determine si un conjunto contiene al menos un elemento para el que se cumple un cierto predicado:

```
def existe(c : Conjunto, predicado : Int => Boolean) : Boolean
```

- implementad una función **map** que transforme un conjunto en otro aplicando una cierta función:

```
def map(c : Conjunto, funcion : Int => Int) : Conjunto
```

5 Observaciones

La mayor parte de las soluciones se pueden escribir como una línea única. En caso de no ser así seguramente debes repensar la solución. En definitiva, es una práctica en la que hay más que pensar (y dibujar en papel) que implementar.

Al igual que en prácticas anteriores el código implementado debe superar un determinado conjunto de test de prueba que garanticen su correcto funcionamiento. En este caso sois vosotros quienes tenéis que aportar los casos de prueba que consideréis adecuados.

6 Material a entregar

Al final de la realización de la práctica se entregará un archivo comprimido con el contenido completo de la práctica, tal y como se integra en el proyecto con el entorno de desarrollo que hayáis usado. Se incluirá también un pequeño documento indicando el entorno de desarrollo y una breve valoración de la práctica (si los conceptos vistos son novedosos, si os ha parecido de interés, problemas encontrados, etc) en tres o cuatro líneas.

La fecha de entrega se fijará más adelante. La entrega se hará mediante la plataforma **PRADO**.