Universidad Tecnológica Nacional

FACULTAD REGIONAL CORDOBA

PARADIGMAS DE PROGRAMACION

Unidad IV
Paradigma Funcional
Parte IV

CONTENIDOS ABORDADOS

- Tipos definidos por el usuario.
- Tipos polimórficos

Tipos definidos por el usuario

Sinónimos de tipo

- Se utilizan para proporcionar abreviaciones para expresiones de tipo aumentando la legibilidad de los programas.
- Sintaxis:

  type Nombre a1 ... an = expresion_Tipo
- Ejemplo:

  type Nombre = String *String* ➔ *[Char]*

  type Precio = Float

  type Producto = (Nombre, Precio)
- No es un nuevo tipo de datos, solo una nueva forma de expresión.

Tipos definidos por el usuario

Definiciones de tipos de datos

- La expresión **data** permite definir nuevos tipos de datos.
- Se utiliza para construir un nuevo tipo de datos formado a partir de otros.
- La definición de nuevos tipos de datos aumenta la seguridad de los programas ya que el sistema de inferencia de tipos distingue entre los tipos definidos por el usuario y los tipos predefinidos.
- Sintaxis:

  data NombreTipo = Nombre tipo1 tipo2

  ó

  data NombreTipo = Nombre { nombreCampo1::tipo1, nombreCampo2::tipo2}

Tipos definidos por el usuario

Ejemplo: (si *type Nombre = String y type Precio = Float*

```
data Producto = Pr Nombre Precio


esCostoso:: Producto -> Bool
esCostoso (Pr _ precio) = precio > 25000

verProducto::Producto -> String
verProducto (Pr nombre precio) = "Producto: nombre " ++
show(nombre) ++ ", precio: " ++ show(precio)
```

Mediante campos con nombre:

```
data Producto = Pr { nombre::Nombre, precio::Precio}
data Producto = Pr { nombre::String, precio::Float }
```

Tipos polimórficos

- Un tipo es polimórfico (“tiene muchas formas”) si contiene una variable de tipo.
- Una función es polimórfica si su tipo es polimórfico.
- Por ejemplo,

```
id :: a -> a
id x = x
```

  Se lee “id es una función que dado un elemento de algún tipo a, retorna un elemento de ese mismo tipo”.
- Aquí ***a*** denota *variable de tipo*.
- Las variables de tipo tienen que empezar con minúsculas.

Tipos polimórficos

Ejemplos:

```
id :: a ->a
id x = x

Main> id 3
   3 :: Int
Main> id ‘x’
  ‘x’ :: Char
Main> id 3.5
  3.5 :: Float
Main> id (True, 3)
 (True, 3) :: (Bool, Int)
```

```
fst :: (a, b) -> a
fst (x, y) = x
Main> fst (1, ‘x’)
  1 :: Int
Main> fst (True, “Hoy”)
  True :: Bool

head :: [a] -> a
head (h:_) = h
Main> head [2,4,7]
  2:: Int
Main> head [‘a’, ‘b’]
 ‘a’ :: Char
```

Tipos polimórficos

Ejemplo con lista:

- La función length es polimórfica, la definición de

  length :: [a] -> Int
- Significa que para cualquier tipo a, length toma una lista de elemetos de tipo a y devuelve un entero.
- Por ejemplo:

  length [1,2,3,4] => 4

  length [”Lunes”, ”Martes”, ”Jueves”] => 3