Acosta Meza Alam
Sierra Casiano Vladimir

Lenguajes de Programación
Punto extra 2

Fecha de Entrega:
29/05/2020

Dos lenguajes que hacen uso de manejo de tipos distintos en las estructuras de control (condicionales e if's) son Haskell y Python.

Figura 1: logotipo python

Figura 2: logotipo haskell

1. Haskell

Como hemos visto en las clases este lenguaje es fuertemente tipificado eso quiere decir que tiene reglas estrictas de tipado en tiempo de compilación (además que se debe especificar el tipo de salida y de entrada en una función dada).

```
funcaux::Int->Int
funcaux a =
          if a `rem` 2 == 0
          then 'a'
          else 0

Resultado al compilar programa:
error:
 • Couldn't match expected type `Int' with actual type `Char'
 • In the expression: 'a'
   In the expression: if a `rem` 2 == 0 then 'a' else 0
   In an equation for `funcaux':
  funcaux a = if a `rem` 2 == 0 then 'a' else 0
```

Como podemos observar hay una contradicción en los tipos que devolvemos en la función como los que declaramos en lo que debía de regresar la firma. Esto también muestra que a pesar de ser un lenguaje de evaluación perezosa tiene cumplir que las condiciones de salida impuestas en la firma de la función.

```
funcaux::Int
funcaux  =
     if 23 `rem` 2 == 0  --Pregunta si 23 es par
     then 'n'
     else 0
Resultado a compilar:
error:
   • Couldn't match expected type `Int' with actual type `Char'
   • In the expression: 'n'
       In the expression: if 23 `rem` 2 == 0 then 'n' else 0
     In an equation for `funcaux':
      funcaux = if 23 `rem` 2 == 0 then 'n' else 0
```

En esta imagen se presencia que a pesar de solo tener que evaluar el primer caso. Manda un mensaje de error en tiempo de compilación por que no cumple con las restricciones impuestas en está.

```
describeLetter :: Char -> String
describeLetter c
   | c >= 'a' && c <= 'z' = "Lower case"
   | c >= 'A' && c <= 'Z' = "Upper case"
   | otherwise            = False

Resultado a compilar :
error:
• Couldn't match type `Bool' with `[Char]'
  Expected type: String
  Actual type: Bool
• In the expression: False
  In an equation for `describeLetter':
   describeLetter c
     | c >= 'a' && c <= 'z' = "Lower case"
     | c >= 'A' && c <= 'Z' = "Upper case"
     | otherwise = False
```

Otro caso seria la utilización de banderas (análogo a conds) e igual aplica a lo mismo que el if no puedo siquiera compilarlo por la firma que yo le declare.

2. Python

Python por el otro lado es un lenguaje de tipado debíl esto quiere decir que tiene menos restricciones que los lenguajes de tipo explicito (un claro ejemplo es que el tipo de salida y de entrada no se especifica más bien lo infiere de que tipo es y trabaja con el).

```
def sum_function(a, b):
    return a+b
```

```
sum_function (5,6)

Regresa 11
```

Esto conlleva a que se devuelva en la función cualquier ”tipo”que el programador desee. Por eso tanto en el estructura if y switch (analogo al conds) se pueda regresar un tipo distinto a otro caso de este.

```
 def theprize(option):
   if option == 1 :
       return 45000000.25 #Dollars
  else:
      return "Two tickets to Hawai"

theprize('a')

Regresa "Two tickets to Hawai"
```

Además debemos de recordar que python es un lenguaje glotón por lo que no importa si la 2da expresión es la que se regresa y tiene un tipo distinto al anterior esta acción no es errónea para el lenguaje.

```
    def switch_demo(argument):
switcher = {
    1: 1,
    2: "February",
    3: 3,
    4: "April",
    5: 5,
    6: "June",
    7: 7.23,
    8: "August",
    9: 9,
    10: "October",
    11: 11,
    12: "December"
}

return switcher.get(argument,"Invalid month of the year")

switch_demo(7)
Regresa 7.23
```