v.7.2

Programación 1 - Primer Parcial - Tema 2

(04-05-2017)

1 Evaluando Expresiones

(Para entregar en papel).

Ejercicio 1. Definimos la función `f` de la siguiente forma:

```
(define (f x y z) (cond [(< z (* x y)) "Blanco"]
                        [(> z (+ x y)) "Negro"]
                        [else "Chau"]))
```

1. Siguiendo todos los pasos, evalúe la expresión `(f 4 2 10)`

2. Considere ahora la función `g`:

```
(define (g x y z) (if (> z (+ x y)) "Negro"
                                    (if (< z (* x y)) "Blanco" "Chau")))
```

¿Verdadero o falso?: *Para todo x, y, z, la expresión `(f x y z)` es igual a `(g x y z)`. Si respondió Verdadero, explique por qué. Si respondió Falso, justifique con un ejemplo donde ambas funciones devuelvan valores distintos para los mismos argumentos.*

2 Diseñando Funciones Simples

Guarde la solución en el archivo ejercicio2.rkt

Ejercicio 2. Diseñe una función `mixture` que pueda ser usada con diferentes tipos de datos y dependiendo del tipo de dato que reciba calcule diferentes valores.

Queremos que se comporte de la siguiente manera:

- Si recibe un **número**, queremos que:
  - si el número es cero o positivo, devuelva el cuadrado de dicho número
  - sino, el cubo del número
- Si recibe una **cadena de caracteres**, queremos que devuelva una imagen. Dicha imagen deberá ser un círculo verde de radio 10 veces la longitud de la

cadena.

- Si recibe una **imagen**, queremos que nos devuelva la siguiente cadena de caracteres: `"La superficie de la imagen es X"`, donde `X` es el valor de la área de la imagen
- En cualquier otro caso, queremos que nos devuelva la cadena de caracteres: `"Tipo de dato incorrecto"`.

Ayuda 1: Para lograr un buen diseño se aconseja diseñar funciones auxiliares para cada uno de los casos planteados.

Ayuda 2: Puede resultarle útil la función `number->string`, así como los predicados vistos en la práctica 1, segunda parte.

---

3 Diseñando Programas Interactivos

Modifique el archivo ejercicio3.rkt

Ejercicio 3. Este programa comienza con el fondo de color negro y un triángulo azul en el centro de la escena. A medida que pasen los ticks del reloj, el triángulo irá rotando y cambiando de color.

El estado del sistema guardará el ángulo de rotación.

Defina constantes para representar el ancho y el alto de la escena. También deberá definir cualquier otra constante que considere útil.

La expresión big-bang deberá comportarse de manera tal que:

- el estado inicial del sistema sea un ángulo nulo
- la función que responde a la cláusula to-draw dibuje el triángulo en el centro de la escena, rotado la cantidad de grados que indique el estado. Cuando el ángulo de rotación sea par, el triángulo debe ser azul, mientras que al momento de ser impar debe colorearse de verde.
- Al hacer click en cualquier parte de la escena, el ángulo de rotación volverá a ser nulo
- cada 0.2 segundos, se incremente el ángulo de rotación en 5 grados

Sugerencia: La función `rotate` puede ser útil.

Ayuda: Recuerde que la expresión `(modulo x 2)` evalúa a `0` si y solo si `x` es un número par.