Diseño de programas

Cecilia Manzino

26 de marzo de 2024

- En Racket un programa es un conjunto de definiciones de constantes y funciones.
- Veremos una metodología para **diseñar** funciones.
- Seguiremos una **receta** que nos guiará en la definición una función.
- Aplicando la receta adquirirán la habilidad de transformar un problema en un programa claro, fácil de modificar y de usar, sin errores, etc.

Receta

1. Diseño de datos.
2. Signatura y declaración de propósito.
3. Ejemplos
4. Definición de la función
5. Validación de los ejemplos
6. Modificaciones en caso de error.

Ejemplo 1

Aplicaremos la receta para diseñar una función que resuelva el siguiente problema:

"Dadas las longitudes, expresadas en metros de un prisma rectangular, calcular su área"

1.Diseño de datos

- La información del problema pertenece al mundo real.
- Para que el programa pueda procesar ésta información y generar un resultado, debo representarla como un dato.
- También tengo que poder interpretar los datos como información.

1.Diseño de datos

Vamos a **documentar** cuál es la decisión que tomamos para representar la información.

```
; Representamos longitudes mediante números
; siendo 1 el equivalente a un metro
```

2.Signatura y declaración de propósito

Signatura Indica cuántos argumentos recibe la función, de qué tipo son y qué datos produce.

Declaración de propósito: descripción breve de lo que calcula la función.

Sirven como especificación o descripcipón para quien utilice la función.

Signatura

- Utilizaremos la siguiente notación para la signatura:

  nombre-función : Dominio -> Codominio

- Se utilizarán los tipos de datos vistos hasta ahora : Number, Boolean, String, Image y Any para indicar cualquier tipo.

Resultado ejemplo

```
; Representamos longitudes mediante números
; siendo 1 el equivalente a un metro

; área: Number Number Number -> Number

; Dados tres números que representan el ancho,
; largo y alto de un prisma rectangular,
; calcula el área del prisma.
; Entrada: 1 1 1   Salida: 6
; Entrada: 1 2 3   Salida: 22

(define   (area a b c)
          (+ (* 2 a b) (* 2 b c) (* 2 a c)))
```

Check-expect

El módulo (**require** test-engine/racket-tests)

provee una función check-expect para chequear funciones en Racket de manera automática. Su sintaxis es:

(check-expect exp exp-esperada)

Los resultados son recopilados y notificados por la función check-expect con un mensaje de la forma Both test pased!, si los tests pasaron o con un reporte sobre los casos que fallaron.

Ejemplo con check-expect

```
; Representamos longitudes mediante números
; siendo 1 el equivalente a un metro

; área: Number Number Number -> Number

; Dados tres números que representan el ancho,
; largo y alto de un prisma rectangular,
; calcula el área del prisma.
(check-expect (area 1 1 1) 6)
(check-expect (area 1 2 3) 22)

(define  (area a b c)
         (+ (* 2 a b) (* 2 b c) (* 2 a c)))
```