Programación I (LCC)
Programación (LM - PM)

Cecilia Manzino

7 de marzo de 2023

Conocer un lenguaje de programación

Lo primero que necesitamos conocer del lenguaje para empezar a programar es su:

Vocabulario: Palabras o símbolos del lenguaje.

+, - , *, /, define, (,), etc

Sintaxis o gramática: Reglas que indican cómo se combinan éstas palabras para armar frases válidas.

(+ 1 2) es una expresión válida en Racket

Semántica: Significado de las frases gramaticalmente correctas.

(+ 1 2) representa al nro. 3

Vocabulario

El vocabulario básico:

valores:

- **numéricos:** 1, 2, -3, 1.5, 2/5, 1+2i, 8e20, etc.
- **booleanos:** #true (o #t), #false (o #f)
- **cadenas:** "", "hola mundo", etc
- **imágenes:** archivos en formato png , jpj y otros formatos

primitivas:

- **palabras claves:** define, if, cond, etc
- **operadores matemáticos:** +, /, *, sqrt, sqr, <, >, <=, >=, =, etc
- **operadores sobre cadenas:** string-append, string-length, etc
- **operadores booleanos:** and, or, not, etc.

Comencemos a usarlos!

Expresiones

- Utilizaremos el sublenguaje Lenguaje para estudiantes
- Racket utiliza notación prefija y paréntesis para definir expresiones utilizando operadores o funciones.

matemática	Racket
1+2	(+ 1 2)
2*3	(* 2 3)
$\sqrt{9}$	(sqrt 9)
1+3*5	(+ 1 (* 3 5))
1+2+3+4	(+ 1 2 3 4)
$5 < 10$	(< 5 10)

Cadenas de caracteres

- Racket utiliza la codificación de caracteres UNICODE.
  Probar: "a", "2", "\u03BB", "\u00DF"
- Se utilizan comillas para escribir cadenas de caracteres.
- Podemos pensarlas como arreglos de caracteres de tamaño fijo.
- **Algunas funciones sobre cadenas:** string-append, string-length, substring, number->string, string->number, string-ith.

Operadores Booleanos

p	(not p)
#t	#f
#f	#t

p	q	(and p q)
#t	#t	#t
#t	#f	#f
#f	#t	#f
#f	#f	#f

p	q	(or p q)
#t	#t	#t
#t	#f	#t
#f	#t	#t
#f	#f	#f

Imágenes

- En DrRacket se pueden pegar imágenes de formatos png,jpg y otros.

- También se pueden crear imágenes con figuras geométricas con las funciones: square, circle, rectangle y otras.

  (circle 50 "outline" "red") ;radio modo color
  (rectangle 100 20 "solid" "yellow") ;ancho largo modo color

- Algunas funciones son: rotate, place-image, image-width, image-height, empty-scene

Tipos de datos

Un tipo de datos hace referencia a la clase de información que se va a manejar:

En Racket los tipos básicos son:

Tipo	**Predicado**	**Comparación**
Number	number?	=
String	string?	string=?
Boolean	boolean?	boolean=?
Image	image?	image=?

Constantes

- Se definen con la palabra clave define:
  (define <identificador> <expresión>)

- Ejemplos:

```
; LARGO : Number
(define LARGO 100)

; LARGO : Number
(define ALTO 50)

; SALUDO : String
(define SALUDO "Les damos la bienvenida a la facultad!")

; CIRCULO : Image
(define CIRCULO (circle 10 "solid" "yellow"))
```

Constantes

- Los comentarios de una línea Racket se introducen con `;` .
- Por convención usaremos MAYUSCULAS en los nombres de constantes y se escriben los tipos de éstas como comentario.
- ¿Es conveniente usar constantes cuando su definición es corta?

Definiciones de funciones

- Se definen con la palabra clave define:

```
(define   (<identificador> <arg1> ... <argN>)
          <expresión>)
```

- Ejemplos:

```
; suma10: Number -> Number
 (define   (suma10 x)
           (+ x 10))

; discriminante : Number Number Number -> Number
 (define   (discriminante a b c)
           (- (* b b) (* 4 a c)))
```

- En Racket los programas están constituidos por definiciones de constantes y de funciones.

Aplicación de funciones

Supongamos que tenemos definida una función:

(define (f $x_1 \cdots x_n$) exp)

Cuando aplicamos la función a los valores $v_1 \cdots v_n$ se aplica la siguiente regla:

$$
\begin{array}{rl}
 & (\texttt{f}\ v_1 \cdots v_n) \\
== & \texttt{<def. de f>} \\
 & \texttt{exp}[v_1/x_1 \cdots v_n/x_n]
\end{array}
$$

donde reemplazamos en e las ocurrencias de $x_1 \cdots x_n$ por $v_1 \cdots v_n$ respectivamente.

Pasos de reducción

```
     (discriminante 1 3 2)
==   <def.  de discriminante>
     (- (* 3 3) (* 4 1 2))
==   <def.  de *>
     (- 9 (* 4 1 2))
==   <def.  de *>
     (- 9 8)
==   <def.  de ->
     1
```

Proposición

Una proposición es una afirmación que puede ser verdadera o falsa.

Ejemplos:

"4 es un número primo." F
"Rosario es la capital de Santa Fe." F
"8 es múltiplo de 2." V

Oraciones que no son proposiciones:

- Oraciones interrogativas
- Oraciones exclamativas
- Órdenes

Traducción de proposiciones

- El número 5 es positivo.

  $(> 5\ 0)$

- El área de un rectángulo de lados 3 y 4 es igual a su perímetro.

  $(= (* 3\ 4)\ (+ 3\ 3\ 4\ 4))$

- La longitud de la cadena "hola" es 4 y de la cadena "mundo" es 5.

```
(and   (= (string-length "hola") 4)
       (= (string-length "mundo") 5))
```

Traducción de proposiciones

- Algún número entre 2 y 3 es par.

  (or (even? 2) (even? 3))

- No es cierto que pi está entre 1 y 2.

  (not (and (< pi 2) (> pi 1)))

- 5 es mayor o igual a su raíz cuadrada.

  (>= 5 (sqrt 5))

Predicados

- Un predicado es una función booleana.
- Sean $T_1 \cdots T_n$ tipos de datos, si f es una función con tipo:
  ; f : $T_1 \cdots T_n$ -> Boolean
  entonces f es un predicado.
- Predicados predefinidos en Racket:
  even? , odd? , integer?, string-numeric?, string-alphabetic?
- Si evaluamos un predicado obtenemos una proposición.

Condicional

Las proposiciones nos permitirán tomar desiciones en un programa.

```
( if   ⟨condición⟩   ; es una proposición
       ⟨exp1 ⟩       ; si la condición es V
       ⟨exp2 ⟩)      ; si la condición es F
```

Ejemplo1: Definir una función que reciba la medida de dos ángulos y en el caso en que sean complementarios devuelva el mensaje "Complementarios"; en caso contrario, devuelva "No complementarios".