

Lenguajes de Programación

Introducción a Racket

Tipos básicos y definición de funciones

Manuel Soto Romero

Universidad Nacional Autónoma de México
Facultad de Ciencias

9 de agosto de 2017

¿Qué se hace aquí?

El laboratorio de lenguajes de programación es un complemento a las clases teóricas. Tiene dos objetivos principales:

- ▶ Aprender a programar en el dialecto `plai` de Racket.
- ▶ Implementar y comprender el funcionamiento de distintos intérpretes.

Funcionamiento

- ▶ El laboratorio se evalúa mediante prácticas que cubren el **35 %** de la calificación final del curso.
- ▶ Cada práctica tendrá un periodo de dos semanas para resolverse y habrá dos sesiones de laboratorio para cubrir el material correspondiente. Habrá notas de cada tema.
- ▶ En cada sesión de laboratorio se realizarán actividades que contarán hasta **un punto extra** sobre la calificación de la práctica correspondiente.
- ▶ En cada sesión de laboratorio se pasará lista, **sólo se contarán los puntos extra a los alumnos que tengan las asistencias correspondientes**.
- ▶ Las prácticas y actividades se entregarán en equipos de **máximo tres integrantes**.

Evaluación

La evaluación de las prácticas considera:

- ▶ El *funcionamiento* de la misma, es decir, que resuelva correctamente lo que se pide.
- ▶ El *estilo* del código, que siga los lineamientos de entrega y reglas de estilo de Racket.

La fecha de entrega de las prácticas es **inamovible**, sin embargo, cada equipo tendrá cinco días de prórroga para tareas y prácticas que podrá usar como mejor le convenga.

Dudas

La atención de dudas se realizará de la siguiente manera:

- ▶ En clase, miércoles de 14 a 16 hrs.
- ▶ Por correo electrónico en la dirección `manu+ldp@ciencias.unam.mx` o mediante la lista del curso `lenguajes@googlegroups.com`.
- ▶ Los viernes de 14 a 16 hrs. en la Sala de Profesores ubicada en la planta baja del edificio Tlahuizcalpan (Previa cita).
- ▶ Mediante la sección de Preguntas Frecuentes de la práctica correspondiente.

Racket

- ▶ Es un lenguaje de propósito general de la familia de lenguajes Lisp.
- ▶ Fue diseñado como una plataforma para la creación, diseño e implementación de lenguajes de programación.
- ▶ Incluye muchos dialectos, en el curso usaremos la variante `plai` que acompaña al libro de texto que usamos.
- ▶ Para escribir y ejecutar programas, usaremos el IDE DrRacket.

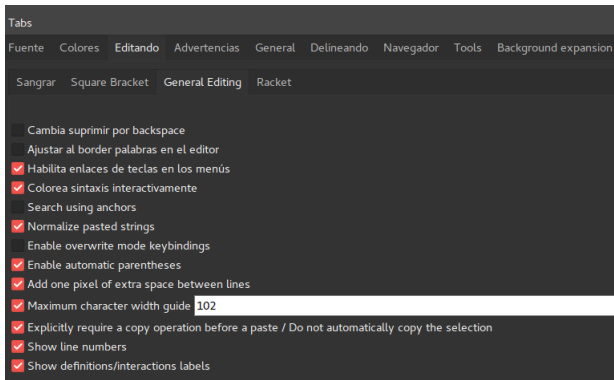
DrRacket

```
1 | #lang racket
2 |
```

Bienvenido a [DrRacket](#), versión 6.8 [3m].
Lenguaje: racket, with debugging; memory limit: 128 MB.
>

Configuraciones

CTRL + ;



Lenguaje por defecto

CTRL + I

```
#()
Automatic #lang line
#lang plai
```

Programming Languages Application and Interpretation

Tipos básicos

BOOLEANOS

Constantes lógicas para verdadero y falso

CADENAS

Secuencias de caracteres, se delimitan por comillas dobles

NÚMEROS

Exactos e Inexactos

SÍMBOLOS

Valores atómicos, usan el mecanismo de *citado* (quote)

CARACTERES

Codificación Unicode, se delimitan por #\

Estas primitivas pueden probarse en el área de interacción de DrRacket

Ejemplos - Tipos básicos

> #t

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

Ejemplos - Tipos básicos

> #t

#t

> #f

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

Ejemplos - Tipos básicos

```
> #t  
#t  
> #f  
#f  
> true
```

Ejemplos - Tipos básicos

```
> #t  
#t  
> #f  
#f  
> true  
#t
```


Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

```
#f
```

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

```
#f
```

```
> -1
```

Ejemplos - Tipos básicos

```
> #t
#t
> #f
#f
> true
#t
> false
#f
> -1
-1
```

Ejemplos - Tipos básicos

```
> #t
#t
> #f
#f
> true
#t
> false
#f
> -1
-1
> 7
7
```

Ejemplos - Tipos básicos

```
> #t
#t
> #f
#f
> true
#t
> false
#f
> -1
-1
> 7
7
```

Ejemplos - Tipos básicos

```
> #t
#t
> #f
#f
> true
#t
> false
#f
> -1
-1
> 7
7
> 1/7
```

Ejemplos - Tipos básicos

```
> #t
#t
> #f
#f
> true
#t
> false
#f
> -1
-1
> 7
7
> 1/7
1/7
```


Ejemplos - Tipos básicos

```
> #t
#t
> #f
#f
> true
#t
> false
#f
> -1
-1
> 7
7
> 1/7
1/7
> 1+7i
```

Ejemplos - Tipos básicos

```
> #t
#t
> #f
#f
> true
#t
> false
#f
> -1
-1
> 7
7
> 1/7
1/7
> 1+7i
1+7i
```

Ejemplos - Tipos básicos

```
> #t                                > 5.5
#t
> #f
#f
> true
#t
> false
#f
> -1
-1
> 7
7
> 1/7
1/7
> 1+7i
1+7i
```

Ejemplos - Tipos básicos

```
> #t                                > 5.5
#t                                  5.5
> #f
#f
> true
#t
> false
#f
> -1
-1
> 7
7
> 1/7
1/7
> 1+7i
1+7i
```

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

```
#f
```

```
> -1
```

```
-1
```

```
> 7
```

```
7
```

```
> 1/7
```

```
1/7
```

```
> 1+7i
```

```
1+7i
```

```
> 5.5
```

```
5.5
```

```
> 5.25e8
```

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

```
#f
```

```
> -1
```

```
-1
```

```
> 7
```

```
7
```

```
> 1/7
```

```
1/7
```

```
> 1+7i
```

```
1+7i
```

```
> 5.5
```

```
5.5
```

```
> 5.25e8
```

```
5.25e8
```

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

```
#f
```

```
> -1
```

```
-1
```

```
> 7
```

```
7
```

```
> 1/7
```

```
1/7
```

```
> 1+7i
```

```
1+7i
```

```
> 5.5
```

```
5.5
```

```
> 5.25e8
```

```
5.25e8
```

```
> 9824579845798275892750982347
```

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

```
#f
```

```
> -1
```

```
-1
```

```
> 7
```

```
7
```

```
> 1/7
```

```
1/7
```

```
> 1+7i
```

```
1+7i
```

```
> 5.5
```

```
5.5
```

```
> 5.25e8
```

```
5.25e8
```

```
> 9824579845798275892750982347
```

```
9824579845798275892750982347
```


Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

```
#f
```

```
> -1
```

```
-1
```

```
> 7
```

```
7
```

```
> 1/7
```

```
1/7
```

```
> 1+7i
```

```
1+7i
```

```
> 5.5
```

```
5.5
```

```
> 5.25e8
```

```
5.25e8
```

```
> 9824579845798275892750982347
```

```
9824579845798275892750982347
```

```
> #\a
```

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

```
#f
```

```
> -1
```

```
-1
```

```
> 7
```

```
7
```

```
> 1/7
```

```
1/7
```

```
> 1+7i
```

```
1+7i
```

```
> 5.5
```

```
5.5
```

```
> 5.25e8
```

```
5.25e8
```

```
> 9824579845798275892750982347
```

```
9824579845798275892750982347
```

```
> #\a
```

```
#\a
```

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

```
#f
```

```
> -1
```

```
-1
```

```
> 7
```

```
7
```

```
> 1/7
```

```
1/7
```

```
> 1+7i
```

```
1+7i
```

```
> 5.5
```

```
5.5
```

```
> 5.25e8
```

```
5.25e8
```

```
> 9824579845798275892750982347
```

```
9824579845798275892750982347
```

```
> #\a
```

```
#\a
```

```
> #\E
```

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

```
#f
```

```
> -1
```

```
-1
```

```
> 7
```

```
7
```

```
> 1/7
```

```
1/7
```

```
> 1+7i
```

```
1+7i
```

```
> 5.5
```

```
5.5
```

```
> 5.25e8
```

```
5.25e8
```

```
> 9824579845798275892750982347
```

```
9824579845798275892750982347
```

```
> #\a
```

```
#\a
```

```
> #\E
```

```
#\E
```

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

```
#f
```

```
> -1
```

```
-1
```

```
> 7
```

```
7
```

```
> 1/7
```

```
1/7
```

```
> 1+7i
```

```
1+7i
```

```
> 5.5
```

```
5.5
```

```
> 5.25e8
```

```
5.25e8
```

```
> 9824579845798275892750982347
```

```
9824579845798275892750982347
```

```
> #\a
```

```
#\a
```

```
> #\E
```

```
#\E
```

```
> "Hola Mundo"
```

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

```
#f
```

```
> -1
```

```
-1
```

```
> 7
```

```
7
```

```
> 1/7
```

```
1/7
```

```
> 1+7i
```

```
1+7i
```

```
> 5.5
```

```
5.5
```

```
> 5.25e8
```

```
5.25e8
```

```
> 9824579845798275892750982347
```

```
9824579845798275892750982347
```

```
> #\a
```

```
#\a
```

```
> #\E
```

```
#\E
```

```
> "Hola Mundo"
```

```
"Hola Mundo"
```

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

```
#f
```

```
> -1
```

```
-1
```

```
> 7
```

```
7
```

```
> 1/7
```

```
1/7
```

```
> 1+7i
```

```
1+7i
```

```
> 5.5
```

```
5.5
```

```
> 5.25e8
```

```
5.25e8
```

```
> 9824579845798275892750982347
```

```
9824579845798275892750982347
```

```
> #\a
```

```
#\a
```

```
> #\E
```

```
#\E
```

```
> "Hola Mundo"
```

```
"Hola Mundo"
```

```
> 'manzana
```

Ejemplos - Tipos básicos

```
> #t
```

```
#t
```

```
> #f
```

```
#f
```

```
> true
```

```
#t
```

```
> false
```

```
#f
```

```
> -1
```

```
-1
```

```
> 7
```

```
7
```

```
> 1/7
```

```
1/7
```

```
> 1+7i
```

```
1+7i
```

```
> 5.5
```

```
5.5
```

```
> 5.25e8
```

```
5.25e8
```

```
> 9824579845798275892750982347
```

```
9824579845798275892750982347
```

```
> #\a
```

```
#\a
```

```
> #\E
```

```
#\E
```

```
> "Hola Mundo"
```

```
"Hola Mundo"
```

```
> 'manzana
```

```
'manzana
```


Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)  
3
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

```
0
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

```
0
```

```
> (sqrt -1)
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

```
0
```

```
> (sqrt -1)
```

```
0+1i
```


Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

```
0
```

```
> (sqrt -1)
```

```
0+1i
```

```
> (or (< 5 4) (equal? 1  
(- 6 5)))
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

```
0
```

```
> (sqrt -1)
```

```
0+1i
```

```
> (or (< 5 4) (equal? 1  
(- 6 5)))
```

```
#t
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

```
0
```

```
> (sqrt -1)
```

```
0+1i
```

```
> (or (< 5 4) (equal? 1  
(- 6 5)))
```

```
#t
```

```
> (and (not (zero? 10))  
(+ 1 2 3))
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

```
0
```

```
> (sqrt -1)
```

```
0+1i
```

```
> (or (< 5 4) (equal? 1  
(- 6 5)))
```

```
#t
```

```
> (and (not (zero? 10))  
(+ 1 2 3))
```

```
6
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

```
0
```

```
> (sqrt -1)
```

```
0+1i
```

```
> (or (< 5 4) (equal? 1  
(- 6 5)))
```

```
#t
```

```
> (and (not (zero? 10))
```

```
(+ 1 2 3))
```

```
6
```

```
> (string-append "Ho" "la")
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

```
0
```

```
> (sqrt -1)
```

```
0+1i
```

```
> (or (< 5 4) (equal? 1  
(- 6 5)))
```

```
#t
```

```
> (and (not (zero? 10))  
(+ 1 2 3))
```

```
6
```

```
> (string-append "Ho" "la")  
"Hola"
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

```
0
```

```
> (sqrt -1)
```

```
0+1i
```

```
> (or (< 5 4) (equal? 1  
(- 6 5)))
```

```
#t
```

```
> (and (not (zero? 10))  
(+ 1 2 3))
```

```
6
```

```
> (string-append "Ho" "la")  
"Hola"
```

```
> (string-length "hola mundo")
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

```
0
```

```
> (sqrt -1)
```

```
0+1i
```

```
> (or (< 5 4) (equal? 1  
(- 6 5)))
```

```
#t
```

```
> (and (not (zero? 10))  
(+ 1 2 3))
```

```
6
```

```
> (string-append "Ho" "la")  
"Hola"
```

```
> (string-length "hola mundo")  
10
```


Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

```
0
```

```
> (sqrt -1)
```

```
0+1i
```

```
> (or (< 5 4) (equal? 1  
(- 6 5)))
```

```
#t
```

```
> (and (not (zero? 10))
```

```
(+ 1 2 3))
```

```
6
```

```
> (string-append "Ho" "la")  
"Hola"
```

```
> (string-length "hola mundo")  
10
```

```
> (substring "Apple" 3)
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

```
0
```

```
> (sqrt -1)
```

```
0+1i
```

```
> (or (< 5 4) (equal? 1  
(- 6 5)))
```

```
#t
```

```
> (and (not (zero? 10))  
(+ 1 2 3))
```

```
6
```

```
> (string-append "Ho" "la")  
"Hola"
```

```
> (string-length "hola mundo")  
10
```

```
> (substring "Apple" 3)  
"le"
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
```

```
3
```

```
> (- 1 1/4)
```

```
3/4
```

```
> (+ 1 (- 3 4))
```

```
0
```

```
> (sqrt -1)
```

```
0+1i
```

```
> (or (< 5 4) (equal? 1  
(- 6 5)))
```

```
#t
```

```
> (and (not (zero? 10))
```

```
(+ 1 2 3))
```

```
6
```

```
> (string-append "Ho" "la")  
"Hola"
```

```
> (string-length "hola mundo")  
10
```

```
> (substring "Apple" 3)  
"le"
```

```
> (string->symbol "Apple")
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
3
> (- 1 1/4)
3/4
> (+ 1 (- 3 4))
0
> (sqrt -1)
0+1i
> (or (< 5 4) (equal? 1
(- 6 5)))
#t
> (and (not (zero? 10))
(+ 1 2 3))
6
```

```
> (string-append "Ho" "la")
"Hola"
> (string-length "hola mundo")
10
> (substring "Apple" 3)
"le"
> (string->symbol "Apple")
'Apple
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
3
> (- 1 1/4)
3/4
> (+ 1 (- 3 4))
0
> (sqrt -1)
0+1i
> (or (< 5 4) (equal? 1
(- 6 5)))
#t
> (and (not (zero? 10))
(+ 1 2 3))
6
```

```
> (string-append "Ho" "la")
"Hola"
> (string-length "hola mundo")
10
> (substring "Apple" 3)
"le"
> (string->symbol "Apple")
'Apple
> (display "hola\nmundo")
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
3
> (- 1 1/4)
3/4
> (+ 1 (- 3 4))
0
> (sqrt -1)
0+1i
> (or (< 5 4) (equal? 1
(- 6 5)))
#t
> (and (not (zero? 10))
(+ 1 2 3))
6
```

```
> (string-append "Ho" "la")
"Hola"
> (string-length "hola mundo")
10
> (substring "Apple" 3)
"le"
> (string->symbol "Apple")
'Apple
> (display "hola\nmundo")
hola
mundo
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
3
> (- 1 1/4)
3/4
> (+ 1 (- 3 4))
0
> (sqrt -1)
0+1i
> (or (< 5 4) (equal? 1
(- 6 5)))
#t
> (and (not (zero? 10))
(+ 1 2 3))
6
```

```
> (string-append "Ho" "la")
"Hola"
> (string-length "hola mundo")
10
> (substring "Apple" 3)
"le"
> (string->symbol "Apple")
'Apple
> (display "hola\nmundo")
hola
mundo
> (+ 1 "Hola")
```

Ejemplos - Funciones predefinidas

```
> (+ 1 2)
3
> (- 1 1/4)
3/4
> (+ 1 (- 3 4))
0
> (sqrt -1)
0+1i
> (or (< 5 4) (equal? 1
(- 6 5)))
#t
> (and (not (zero? 10))
(+ 1 2 3))
6
```

```
> (string-append "Ho" "la")
"Hola"
> (string-length "hola mundo")
10
> (substring "Apple" 3)
"le"
> (string->symbol "Apple")
'Apple
> (display "hola\nmundo")
hola
mundo
> (+ 1 "Hola")
+: contract violation
```


Definición de funciones

Al definir funciones se recomienda seguir los siguientes pasos:

1. Entender qué tiene que hacer la función.
2. Escribir la descripción de la función.
3. Escribir su contrato, o sea su tipo (cuantos parámetros, de qué tipo, qué regresa).
4. Escribir las pruebas asociadas a esta función sobre los posibles datos de entrada (casos significativos).
5. Implementar el cuerpo de la función.

Ejemplo - Definición de funciones

```
;; Función que calcula el área de un círculo dado su  
;; diámetro  
;; area: number -> number  
(define(area diametro)  
  (* pi (/ diametro 2) (/ diametro 2)))  
  
(test(area 10) 78.53)  
(test(area 4) 12.56)  
(test(area 16) 201.06)
```

Asignaciones locales

Permiten guardar cálculos u otras funciones en variables para usarlas en una determinada región del programa.

```
;; Función que calcula el área de un círculo dado su  
;; diámetro  
;; area: number -> number  
(define(area diametro)  
  (let([r (/ diametro 2)])  
    (* pi r r)))
```

Asignaciones locales anidadas

```
(let ([a 2] [b (+ a a)]))  
b)
```

Este tipo de expresiones no funcionan. Necesitamos anidar:

```
(let ([a 2])  
  (let ([b (+ a a)]))  
  b))
```

O usar la versión con azúcar sintáctica, `let*`:

```
(let* ([a 2] [b (+ a a)]))  
b)
```

Existe otra variante para expresiones recursivas, llamada `letrec`.

Condicionales

;; Función que calcula el valor absoluto de un número

;; absoluto: number -> number

```
(define(absoluto n)
```

```
  (if(< n 0)
```

```
    (* n -1)
```

```
    n))
```

;; Función que regresa la representación en cadena de un mes

;; pasado como número

;; mes: number -> string

```
(define(mes n)
```

```
  (cond
```

```
    [(= n 1) 'Enero']
```

```
    [(= n 2) 'Febrero']
```

```
    ...
```

```
    [else (error 'mes "Número inválido")]))
```