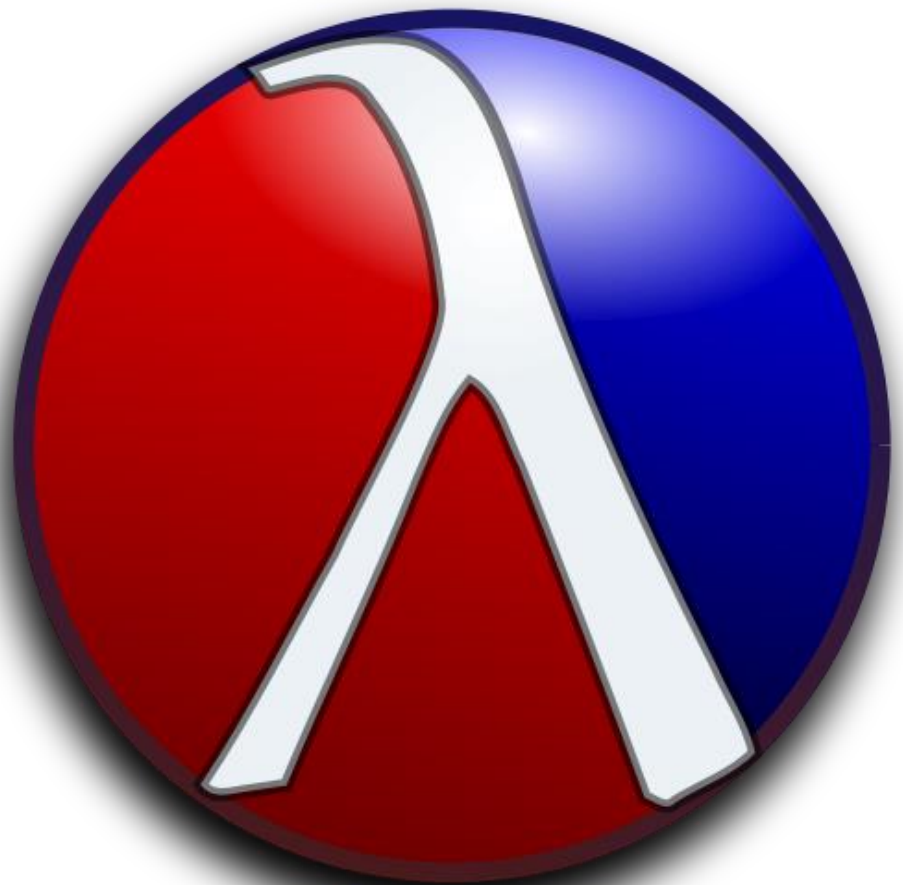


2020

MANUAL DE USUARIO

Guía de programación funcional (Racket)



((Racket))

MSC. LUIS EDUARDO MUÑOZ GUERRERO



CONTENIDO

TABLA DE IMÁGENES	3
SOBRE ESTE MANUAL.....	4
INTRODUCCIÓN.....	5
OBJETIVOS.....	5
CARACTERÍSTICAS DEL ENTORNO.....	5
Requisitos mínimos de hardware y software	5
INSTALACIÓN	6
Ejecutar el archivo llamado Guia_Racket.exe	6
Instalación del programa.....	6
Elección de tareas adicionales.....	6
Confirmación de instalación	7
Instalación de la aplicación	7
Confirmación y ejecución	8
Ejecución del programa	9
USO DEL APLICATIVO	9
Pantalla principal	9
Apartado introductorio	9
INTERIOR DE LOS MÓDULOS	13



TABLA DE IMÁGENES

IMAGEN 1	6
IMAGEN 2	6
IMAGEN 3	7
IMAGEN 4	8
IMAGEN 5	8
IMAGEN 6	9
IMAGEN 7	10
IMAGEN 8	10
IMAGEN 9	11
IMAGEN 10.....	11
IMAGEN 11.....	12
IMAGEN 12.....	12
IMAGEN 13.....	12
IMAGEN 14.....	13
IMAGEN 15.....	14
IMAGEN 16.....	15
IMAGEN 17.....	16
IMAGEN 18.....	17
IMAGEN 19.....	18
IMAGEN 20.....	19
IMAGEN 21.....	20
IMAGEN 22.....	21
IMAGEN 23.....	22
IMAGEN 24.....	23
IMAGEN 25.....	24
IMAGEN 26.....	25
IMAGEN 27.....	26
IMAGEN 28.....	27
IMAGEN 29.....	28
IMAGEN 30.....	29
IMAGEN 31.....	30
IMAGEN 32.....	31
IMAGEN 33.....	32
IMAGEN 34.....	33
IMAGEN 35.....	34
IMAGEN 36.....	35
IMAGEN 37.....	36
IMAGEN 38.....	37
IMAGEN 39.....	38
IMAGEN 40.....	39
IMAGEN 41.....	40
IMAGEN 42.....	41



SOBRE ESTE MANUAL

ENERO 2020

Este manual fue redactado para el Software. Guía de programación funcional Racket.
(v. 20200114)

INTRODUCCIÓN

Esta guía de programación funcional Racket, se crea con el fin de apoyar a los estudiantes en el aprendizaje del área de la programación funcional basado en el lenguaje de programación Racket.

El software guiará a los usuarios de forma didáctica, en los conceptos más básicos del lenguaje de programación, desde la declaración de funciones, las operaciones aritméticas y lógicas, tipos de notaciones hasta el uso de listas, estructuras de datos e interfaces gráficas en Racket. La aplicación es una herramienta de apoyo, ya que permite ejemplarizar cada uno de los temas que propone y a su vez evaluarlos para mejorar las habilidades en programación funcional.

OBJETIVOS

- Recopilar los conceptos fundamentales de la programación funcional en una guía interactiva.
- Orientar el aprendizaje de la programación en lenguaje Racket de una manera didáctica e interactiva.
- llevar al estudiante al progreso paso a paso de los conocimientos de la programación funcional, desde los conceptos fundamentales hasta fundamentos avanzados.
- Garantizar el entendimiento óptimo de la sintaxis y de componentes del lenguaje de programación Racket, utilizando ejemplos prácticos explicados a detalle.

CARACTERÍSTICAS DEL ENTORNO

Requisitos mínimos de hardware y software

- Monitor con resolución de 1024 x 768 o superior.
- Sistema operativo Windows o versiones superiores.
- Java versión 7 o superior.
- Procesador de 1.6GHz o superior
- Memoria RAM de 1Gb o superior
- 200Mb disponibles en el disco duro.
- Opcionalmente puede ejecutar el aplicativo en NetBeans IDE 8.2 o superior.



INSTALACIÓN

Para instalar el programa y poder usarlo normalmente se deben seguir los siguientes pasos.

Ejecutar el archivo llamado Guia_Racket.exe

Para este paso, se presiona doble click en el ícono presentado a continuación.



IMAGEN 1

Instalación del programa

Los siguientes procesos constituyen la instalación lógica del programa:

Elección de tareas adicionales

- ✓ Se eligen tareas adicionales como crear un acceso directo al escritorio.

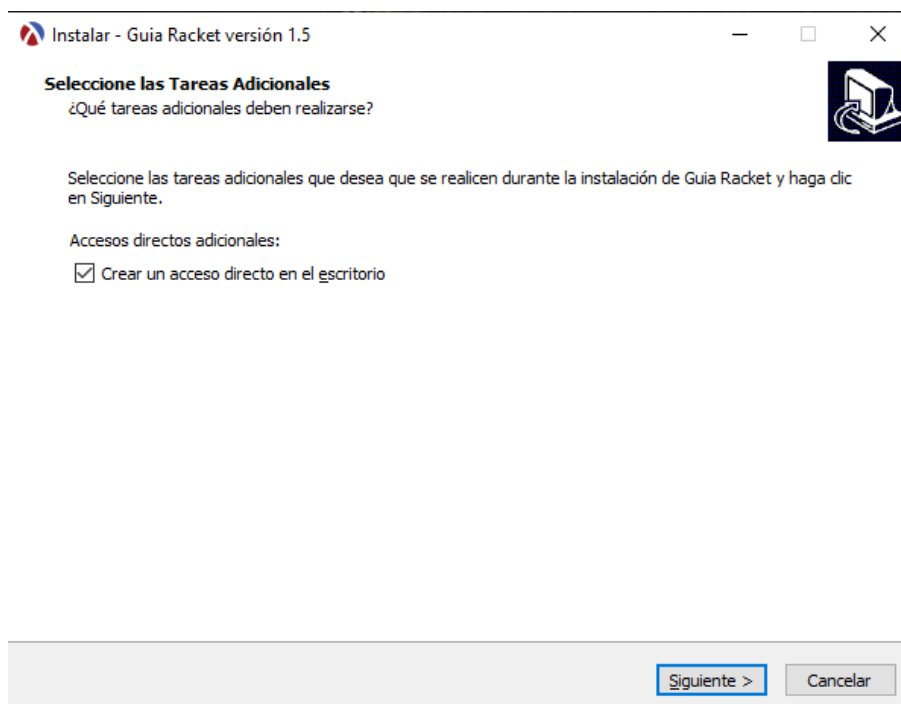


IMAGEN 2

- ✓ Se presiona click en “Siguiente” para continuar la instalación.



Confirmación de instalación

- ✓ Se confirman las tareas adicionales de instalación

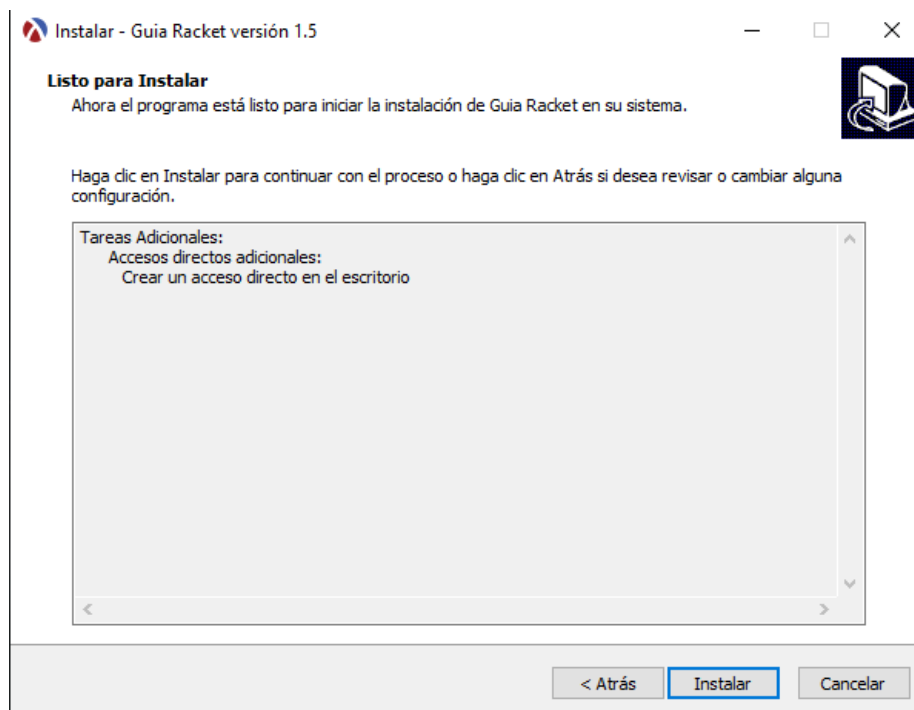


IMAGEN 3

- ✓ Se presiona click en “Instalar” para iniciar la instalación del programa.

Instalación de la aplicación

- ✓ Se cargan e instalan los archivos para instalar el programa.

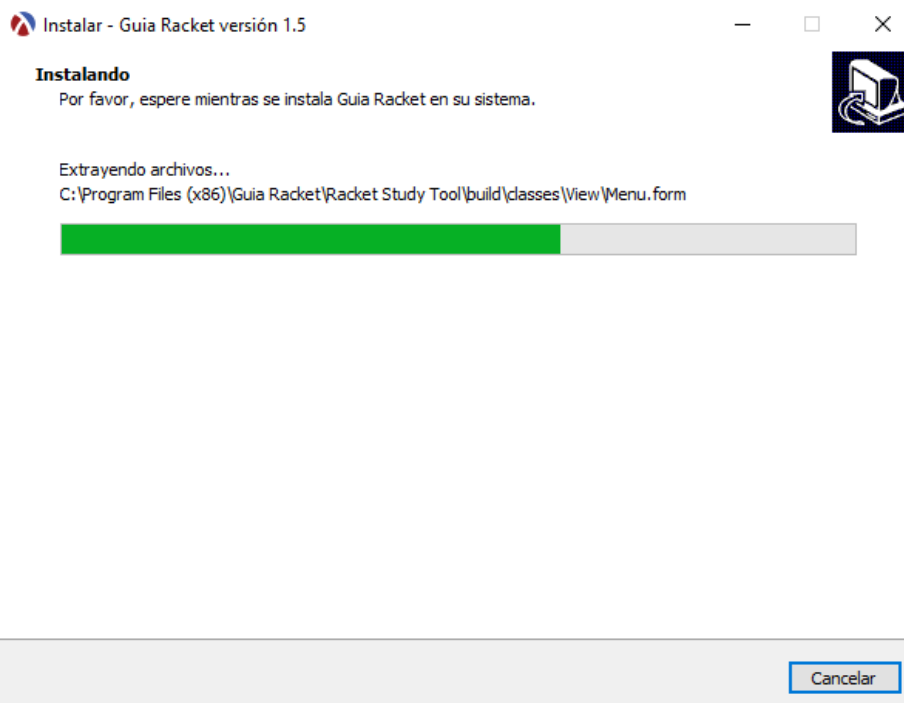


IMAGEN 4

Confirmación y ejecución

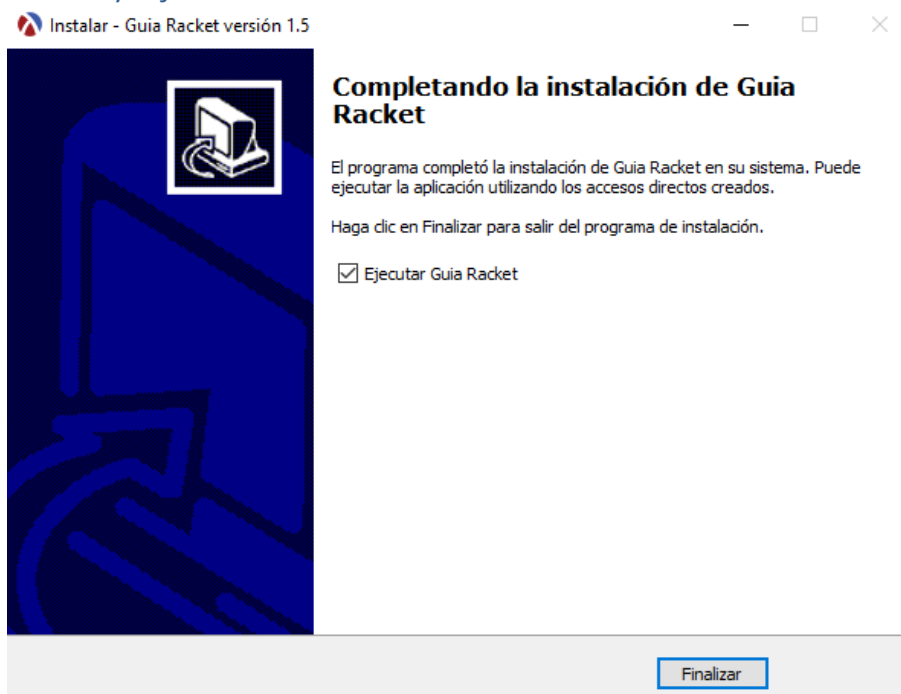


IMAGEN 5

- ✓ Se termina la instalación y se finaliza el proceso. Si se mantiene la opción de “Ejecutar Guia Racket”, el programa iniciará automáticamente.



Ejecución del programa



IMAGEN 6

- ✓ Una vez instalado el programa, debe poderse ejecutar de la siguiente manera. Si se presenta esta pantalla, la instalación fue exitosa y el programa puede usarse.

USO DEL APLICATIVO

El programa posee múltiples vistas las cuales permiten diferentes funcionalidades. Se explicarán su uso y función en los siguientes apartados:

Pantalla principal

Apartado introductorio

- ✓ Al iniciar el programa se presentará la pantalla principal titulada “Página principal Racket [1]”. En la parte superior derecha de la pantalla habrán dos botones con nombres “Acerca de [2]” y Ayuda [3]” con los cuales el usuario puede interactuar. Un poco más abajo encuentra el objetivo principal del programa [4] y a su derecha los logos de Racket y la Universidad Tecnológica de Pereira [5].



IMAGEN 7

Apartado de botones de apoyo

- ✓ Acerca de
 - Al presionar el botón “Acerca de [1]”, el programa abrirá una pestaña con información sobre el programa [2] y la opción de cerrar la pestaña con el botón “CERRAR [3]”

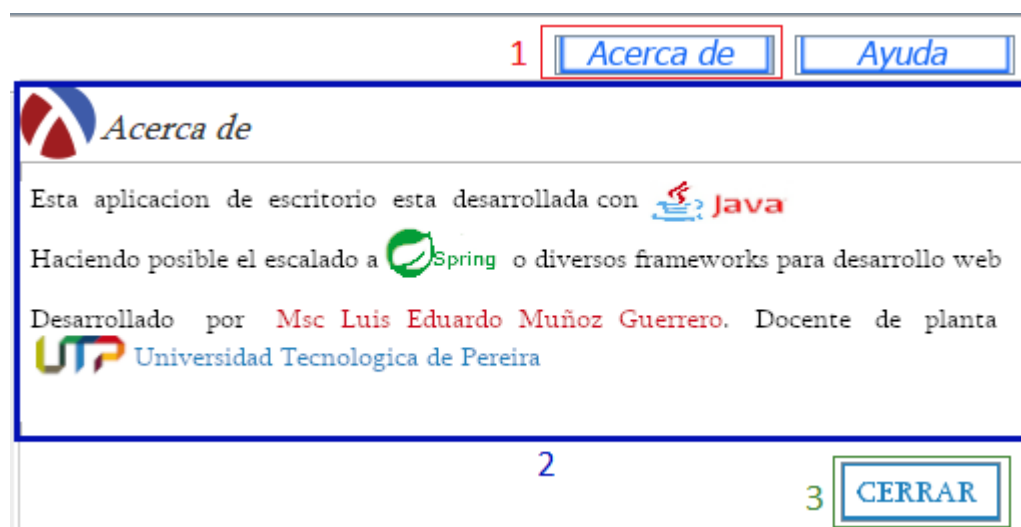


IMAGEN 8

- ✓ Ayuda
 - Al presionar el botón “Ayuda [1]” el programa abrirá el manual de usuario para que la persona se pueda instruir.



IMAGEN 9

Apartado de módulos

- ✓ Debajo del apartado introductorio, se encuentra la introducción a los módulos de estudio [1]. Debajo de esta, se encuentran los primeros tres módulos del programa los cuales se titulan “Introducción [2]”, “Funciones [3]” y “Operaciones aritméticas [4]”. Cada uno de los anteriores posee una descripción, un indicador de dificultad y un botón llamado “ENTRAR” con el cual se puede ingresar al módulo seleccionado.



IMAGEN 10

- ✓ Mientras se navega en el módulo, se encuentra el resto de los módulos del programa. Se ven los módulos de “Condicionales [5]”, “Recursividad [6]” y “Caracteres y documentación [7]”...

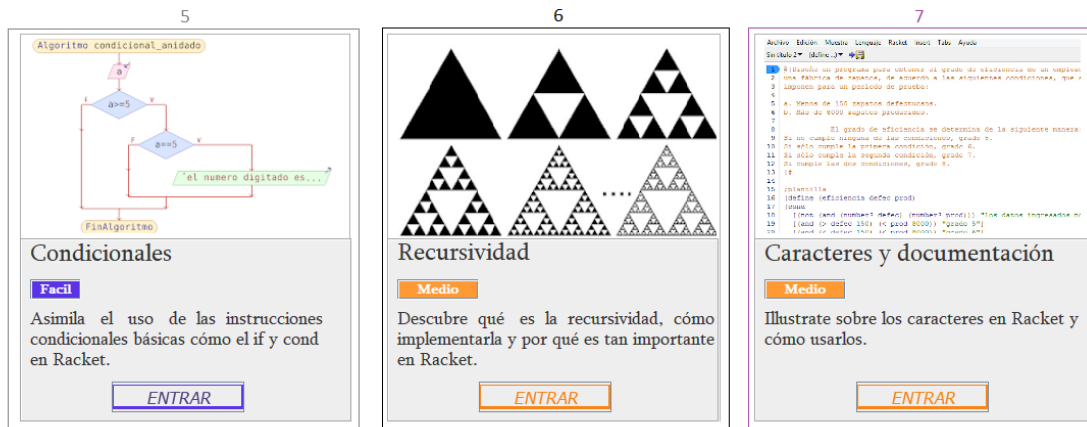


IMAGEN 11

✓ ... De “Cadenas [8]”, “Vectores [9]”, y “Pares [10]”...

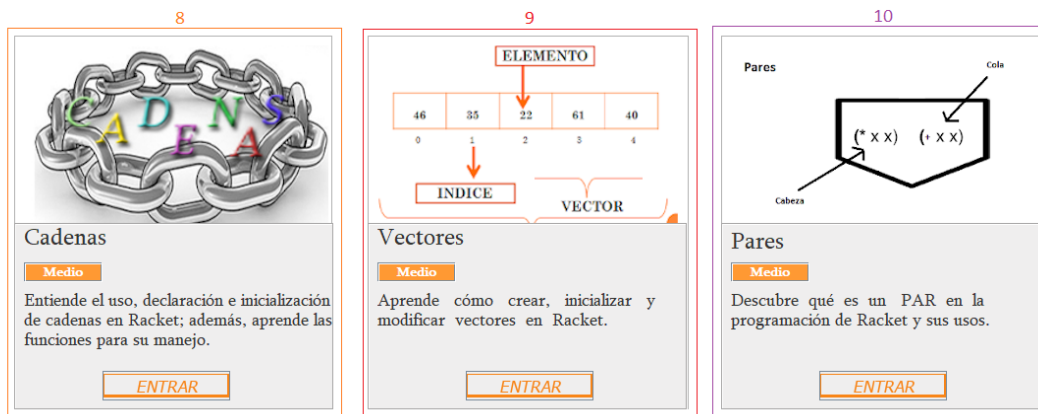


IMAGEN 12

✓ ...De “Listas [11]”, “Estructuras de datos [12]” y “Modo gráfico [13]”...

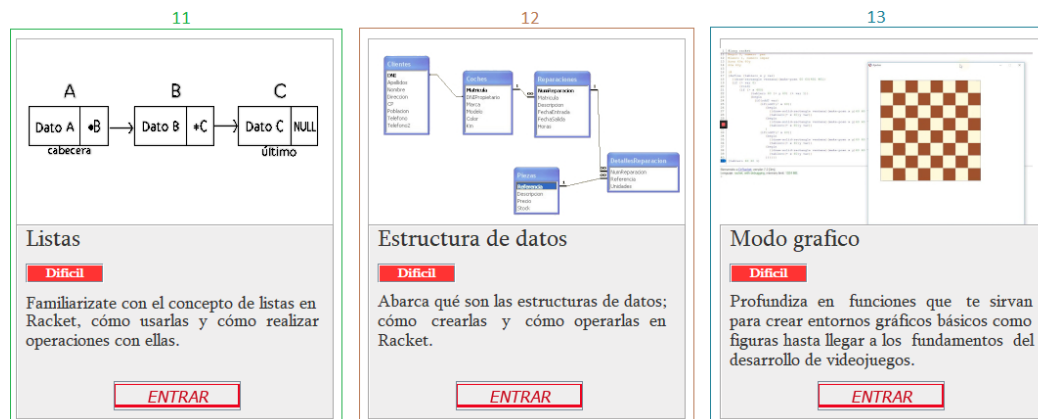


IMAGEN 13



- ✓ ... Y finalmente de “Paso a paso [14] y “¿Te atreves? [15]”

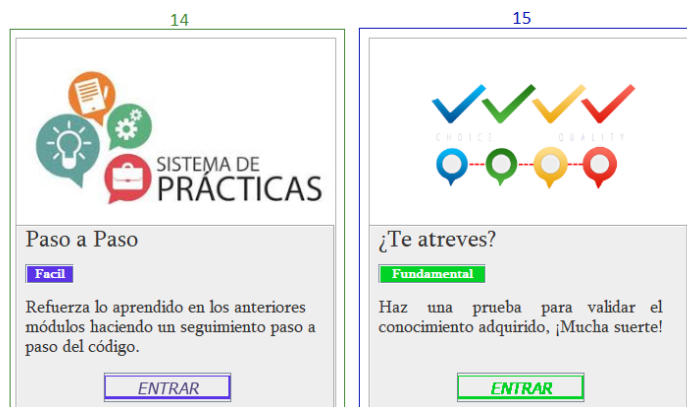


IMAGEN 14

INTERIOR DE LOS MÓDULOS

- ✓ Introducción
- Dentro del módulo de introducción del programa, se encuentran los siguientes ítems:



Introduccion

1

¿Que es la programacion funcional?

La programación es el proceso de diseñar, escribir, probar, depurar y mantener el código de programas computacionales. El propósito de la programación es crear programas que exhiban un comportamiento deseado, es decir, que cumplan con un objetivo. El proceso de escribir código requiere frecuentemente conocimientos en distintas áreas, además del dominio del lenguaje de programación a utilizar, algoritmos especializados y lógica formal.

2

¿Que es Dr.Racket?

Racket es un lenguaje de programación derivado de Scheme. Acepta la programación Funcional, Modular, Orientada a objetos y de Procedimientos, lo que quiere decir que es multi paradigma. Al momento de la definición de tipos de datos puede ser dinámica, débil, fuerte y estática, es multiplataforma y sobre todo es un lenguaje interpretado.

3

Paradigmas de la programacion

Se analiza, analiza y resuelve un problema computable. Muchos lenguajes de programación adoptan paradigmas precisos mientras que otros soportan múltiples paradigmas, esto quiere decir que, cada lenguaje se especifica para que su uso se dé, de acuerdo a determinado(s) paradigma(s), por ejemplo, se enfocó en soportar el paradigma orientado a objetos, pero sin dejar de lado el paradigma estructurado. Entre los principales paradigmas de programación se encuentran: Funcional, Orientado objetos, Imperativo, Restricciones, Lógico y Declarativo.

4

Notaciones

Existen tres tipos de notaciones al momento de escribir expresiones aritméticas, estas notaciones se refieren a la posición del operador con respecto a los operandos.

La notación de una expresión aritmética puede representarse con un árbol. Veamos cómo se representaría la suma de dos números en cada una de las notaciones:

Orden: 2 + 1

IMAGEN 15

- ✓ Donde en la parte izquierda de la pantalla se presenta teoría de la introducción a Racket como la definición de programación funcional [1], la definición de Dr. Racket [2], los paradigmas de la programación [3] y las notaciones [4]. En la parte derecha se presentan los siguientes ítems:

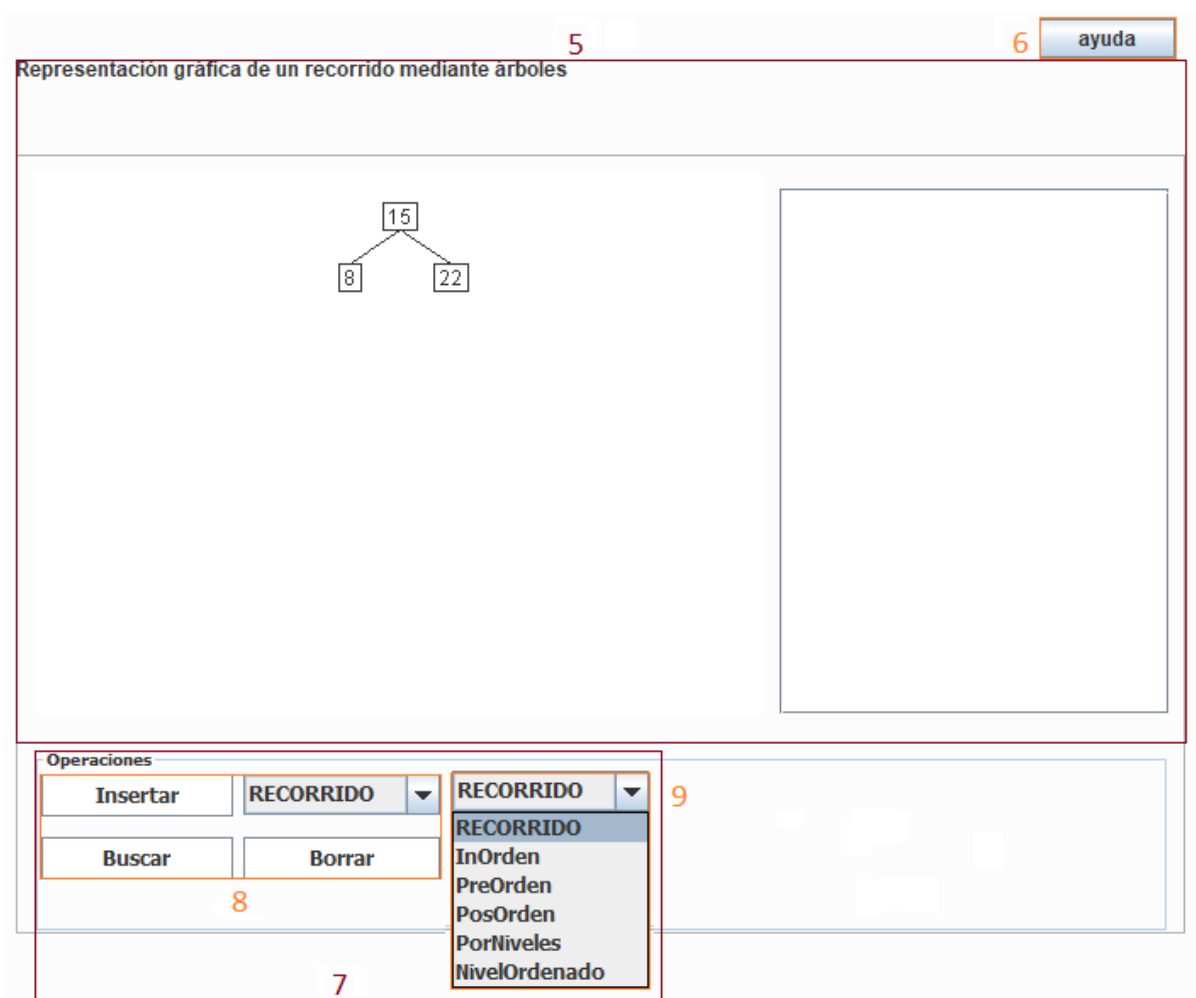


IMAGEN 16

- ✓ Donde se ve un graficador de árboles [5] con el que se puede interactuar según los siguientes botones, un botón de ayuda que redirige al manual de usuario [6], un menú de operaciones [7] el cual permite insertar, buscar y borrar datos [8]; los datos pueden ser insertados o buscados según una lista desplegable [9].
- ✓ Funciones
 - Dentro del módulo de funciones, se encuentran los siguientes ítems:

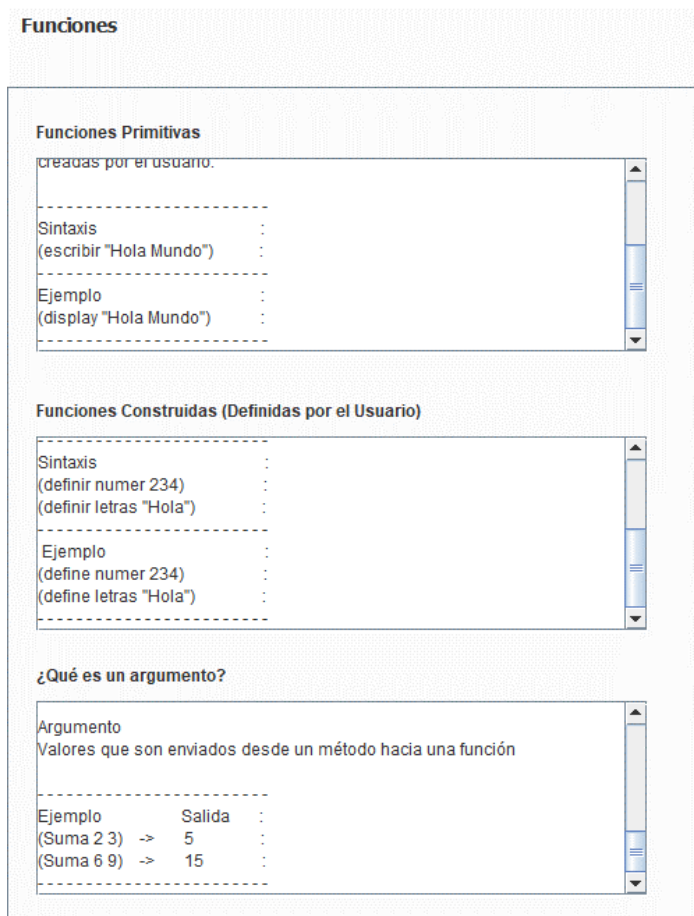


IMAGEN 17

- ✓ Donde en la parte izquierda de la pantalla se presenta la teoría de las funciones [1]. Se habla sobre las funciones primitivas [2], las funciones construidas [3] y los argumentos [4]. En la parte derecha se presentan los siguientes ítems:



Ejemplos [ayuda](#)

Funciones que operan con valores numericos

Función	Entrada-salida	Descripción
*	(* num1 num2) -> num	Multipliación
+	(+ num1 num2) -> num	Suma
-	(- num1 num2) -> num	Resta
/	(/ num1 num2) -> num	División
<	(< num1 num2) -> #t o #f	Comprar el primer valor con los ...
<=	(<= num1 num2) -> #t o #f	Comprar el primer valor con los ...
>	(> num1 num2) -> #t o #f	Comprar el primer valor con los ...
>=	(>= num1 num2) -> #t o #f	Comprar el primer valor con los ...
abs	(abs num) -> num	Valor absoluto de un numero
acos	(acos num) -> num	Arco-coseno
add1	(add1 num) -> num+1	Aumenta un número en 1

Ejemplo 1 Ejemplo 2 Ejemplo 3 Ejemplo 4 Ejemplo 5 Ejemplo 6

```
1 ;Crear un programa en Racket tal que eleve un número a una potencia, luego ese mismo
2 ;número mostrarlo por pantalla
3
4 (define (MostrarPotencia n)
5   (display (expt n 2))
6 )
7
8 (MostrarPotencia 3)
```

IMAGEN 18

- ✓ Donde se ve una pantalla de listas de funciones [5], un botón de ayuda que redirige al manual de usuario [6] y una pantalla interactiva de ejemplos de funciones [7].
- ✓ Operaciones aritméticas
 - Dentro del módulo se presentan los siguientes ítems:



Expresiones matemáticas

¿Qué es una expresión aritmética?

Se entiende por expresión aritmética a aquella donde los operandos que intervienen en dicha expresión son números, el resultado es un número y los operadores son aritméticos.

$+$, $-$, $*$, $/$

¿Qué son los operadores lógicos y de comparación?

```
(escribir b) :  
) :  
----- :  
Ejemplo :  
(if (< a b) :  
  (display a) :  
  (display b) :  
) :
```

IMAGEN 19

- ✓ Donde en la parte izquierda del módulo se habla de la teoría de las operaciones aritméticas [1]. Se habla sobre la definición de operación aritmética [2] y los operadores lógicos [3] y en la parte derecha se presentan los siguientes ítems:



Ejemplosayuda

Expresiones matemáticas básicas

Calculadora con operaciones básicas

=

+

*

-

/

Operadores logicos y comparación

Operación	Operador
menor que	<
mayor que	>
igual que	==
menor o igual que	<=
mayor o igual que	>=
distinto de	!=

Ejemplo #1Ejemplo #2Ejemplo #3Ejemplo #4Ejemplo #5Ejemplo #6

1;Calcular el área de un anillo.
2
3(define (AreaAnillo R-int R-ext)
4 (- (* pi (expt R-ext 2)) (* pi (expt R-int 2)))
5)
6
7(define (Area var1 var2)
8 (display "Area del Anillo")
9 (display (AreaAnillo var1 var2))
10)
11
12 (Area (read) (read)))

Consola

EjecutarPaso a Paso

IMAGEN 20



- ✓ Donde se ve una calculadora de operaciones matemáticas básicas [4], un botón de ayuda que redirige al manual de usuario [5], una tabla de definición de tipos de operadores básicos y lógicos [6], una serie de ejemplos ejecutables[7] y su resultado por consola [8].
- ✓ Condicionales
 - Dentro del módulo de condicionales, se encuentran los siguientes ítems:

Condicionales

Boleanos y Relaciones

Los booleanos son de la forma #t -> True o #f False

Ejemplo:

```
(= x y) ; "¿x es igual a y?"  
(< x y) ; "¿x es menor a y?"  
(> x y) ; "¿x es mayor a y?"  
(<= x y) ; "¿x es menor o igual a y?"  
(>= x y) ; "¿x es mayor o igual a y?"
```

Condicional IF

y su estructura es.

Ejemplo

```
(begin  
  ("a menor que b")  
  ("el número de es:" a)  
)
```

Condicional COND

(else (Respuesta_Falso))

Ejemplo

```
(cond  
  ((< a b) (display "Es mayor b"))  
  ((< b a) (display "Es mayor a"))  
)
```

IMAGEN 21



- ✓ Donde en la parte izquierda de la pantalla se presenta la teoría de los condicionales [1]. Se explica el concepto de condicional [2], los booleanos [3], el condicional IF [4] y el condicional COND [5]. En la parte derecha se presentan los siguientes ítems:

Ejemplos

ayuda

Representación gráfica de un condicional

```
graph TD; Entry(( )) --> Cond{Cond ?}; Cond -- SI --> A[A]; Cond -- NO --> B[B]; A --> Exit(( )); B --> Exit;
```

Ejemplo 1

Ejemplo 2

Ejemplo 3

Ejemplo 4

Ejemplo 5

Ejemplo 6

```
3 ;Construir un programa que dada la suma de dos numeros
4 ;imprima si dicha suma es mayo, menor o igual a 10.
5
6 ;definimos la función encargada de sumar los números.
7
8 (define (Suma N1 N2)
9   (+ N1 N2)
10  )
11
12 (define (MayorQ10? N1 N2)
13   (if (> (Suma N1 N2) 10)
14       (display "Si es mayor que 10")
15       (display "No es mayor que 10"))
16 )
17 )
18
19 (MayorQ10? (read) (read))
```

Consola

Ejecutar

Paso a Paso

n unos ejercicios para reforzar lo aprendido?!!

mos alla

Menu

IMAGEN 22



- ✓ Donde se ve el ejemplo de un diagrama de flujo con condicionales [6], un botón de ayuda que redirige al manual de usuario [7], un menú de programas con cada tipo de condicional [8] y una consola donde se ven los resultados de estos programas [9]
- ✓ Recursividad
 - Dentro del módulo de recursividad, se presentan los siguientes ítems:

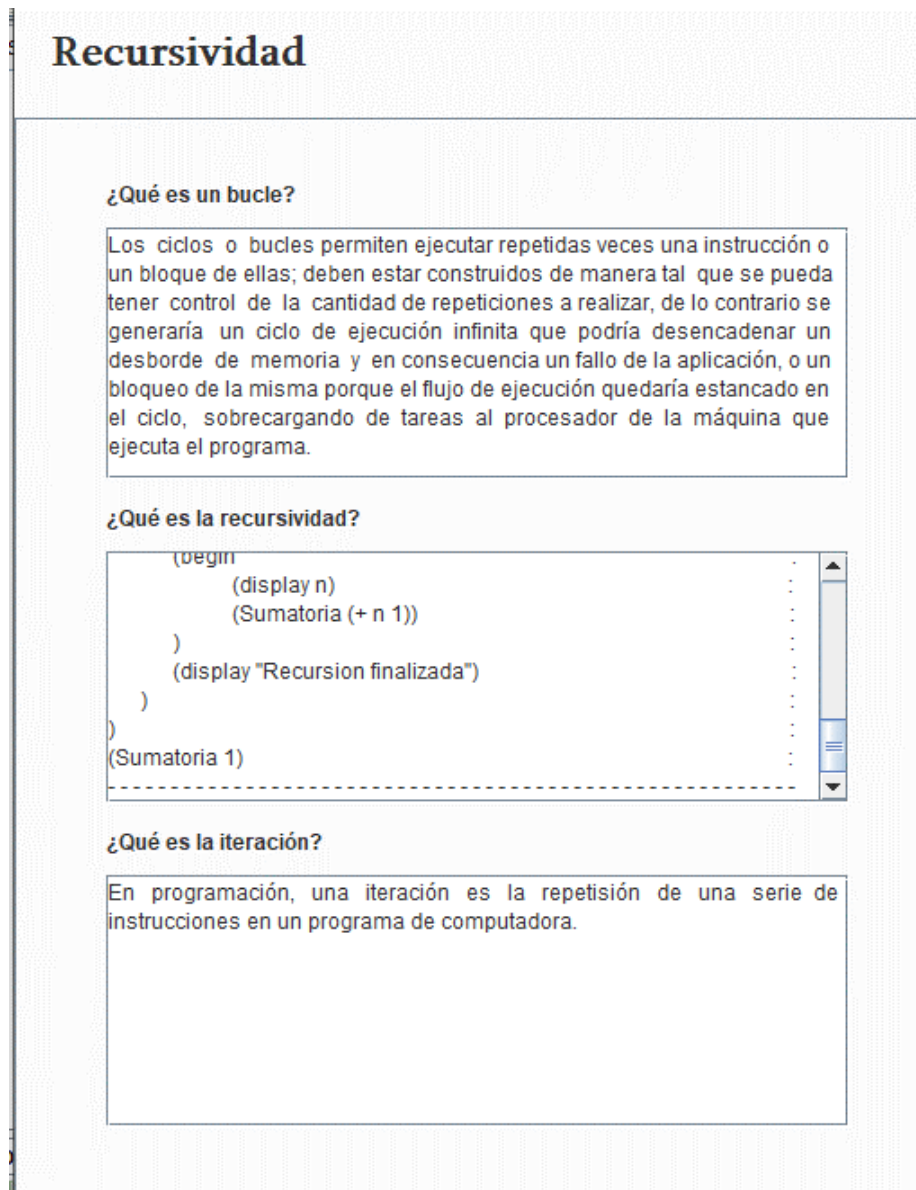


IMAGEN 23

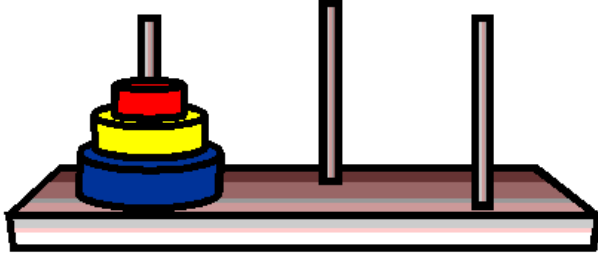
- ✓ Donde en la pantalla izquierda de la pantalla se ve la teoría relacionada con la recursividad [1]. Se ve la definición de bucle [2], la definición de recursividad [3], y la definición de iteración [4]. En la parte derecha se presentan los siguientes ítems:



Ejemplos

Ayuda

Representación de una recursión mediante la torre de Hanoi



Ejemplo #1

Ejemplo #2

Ejemplo #3

Ejemplo #4

Ejemplo #5

Ejemplo #6

```
1 ;Imprimir los números del uno al 10 recursivamente:
2
3 (define(prinNum-10 X)
4   (if (= X 10)
5     (display X)
6     (begin
7       (display X)
8       (PrinNum-10 (+ 1 X))))
9   )
10 )
11
12 (PrinNum-10 0)
```

Consola

Ejecutar

Paso a Paso

IMAGEN 24

- ✓ Donde se ve una representación de las torres de Hanoi [5], un botón de ayuda que redirige al manual de usuario [6], ejemplos codificados de recursividad [7] y una consola donde se ven los resultados [8].



✓ Caracteres

- Dentro del módulo de caracteres, se encuentran los siguientes ítems:

Caracteres

¿Qué son Caracteres?

Son un tipo de dato. Los caracteres se utilizan para mostrarlos en pantalla y así comunicarnos con el usuario.

Char?

Sintaxis	Salida
(caracter? a) ->	#t o #f

Ejemplo

Salida
(char? a) -> #t o #f

Char-ci

Se utiliza para un caso insensible, es decir, mayúsculas o minúsculas Racket las tomara como iguales.

(Racket también provee funciones para encontrar que tipo de carácter se está evaluando. Puede ser un alfabético, numérico, espacio en blanco, mayúsculas o minúscula).

Comentarios

IMAGEN 25

- ✓ En la parte izquierda del módulo se presenta la teoría de los caracteres [1]. Se define qué son los caracteres [2], la función char? [3], la función char-ci [4], los comentarios [5] y la documentación [6]. En la parte derecha se presentan los siguientes ítems:



The screenshot displays a Racket IDE window titled "Ejemplos". At the top right is a button labeled "ayuda". Below the title bar are tabs for "Ejemplo #1" through "Ejemplo #6", with "Ejemplo #1" selected. The code editor contains the following Racket code:

```
1 ;Recibir un carácter que represente una nota. Si es la letra E mayúscula o la
2 ;letra e minúscula imprimir "excelente", si es la letra B o la letra b imprimir "Bueno",
3 ;si es la letra A o la letra a imprimir "Aceptable", si es la letra D o la letra d imprimir
4 "Define", y si es la letra I o la letra i imprimir "Insuficiente".
5
6 (define (Notas X)
7   (cond
8     ((char-ci=? X #/e) (display "Excelente"))
9     ((char-ci=? X #/s) (display "Sobresaliente"))
10    ((char-ci=? X #/a) (display "Aceptable"))
11    ((char-ci=? X #/i) (display "Insuficiente"))
12    ((char-ci=? X #/d) (display "Deficiente"))
13    (else (display "No es una nota valida. "))
14  )
15 )
16 (Notas (read))
```

Below the code editor is a "Consola" area, which is currently empty. At the bottom right of the IDE window is a button labeled "Ejecutar". Below the IDE window is a button labeled "Paso a Paso".

IMAGEN 26

- ✓ Donde se ven ejemplos codificados de caracteres [7], un botón de ayuda que redirige al manual de usuario [8] y una consola para ver los resultados de los ejemplos [9].
- ✓ Cadenas
 - Dentro del módulo de cadenas, se encuentran los siguientes ítems:



Cadenas y Strings

¿Qué es una cadena?

Una cadena es la unión de dos o mas caracteres que representan una palabra en pocas palabras, un conjunto de caracteres forma una cadena, es decir, "Hola".

Representado por comillas dobles.

Creación de cadenas.

Código	Salida
(make-string 3)	" " cadena de tres posiciones
(make-string 3 #/a)	"aaa" cadena con el carácter a de 3 posiciones
(string #/H #/o #/l #/a)	"Hola" cadena con caracteres dados

Como se puede ver las funciones make-string y string sirven para crear cadenas.

La función (String?).

Código	Salida
(string? "casa")	#t
(string? 'rana)	#f ;'rana no es una cadena
(string? 8)	#f ;8 es un numero
(string? "")	#t ;es una cadena aunque este vacía

Como se puede observar se considera cadena a todo aquello que se encuentre dentro de unas comillas dobles ""

¿Una cadena tiene tamaño?

IMAGEN 27



Modificación de cadenas

Para modificar una cadena usamos la función (string-set)

Sintaxis :
(cadena-cambiar! cadena n carácter) :

Ejemplo :
(string-set! (string #/M #/a #/r #/i #/a) 4 #/o) :

Unión de cadenas.

Sintaxis :
(cadena-agregar "Hola" "mundo") :

Ejemplo :
Código Salida :
(string-append "Hola " "mundo") Hola mundo :
(string-append) "" cadena vacía :

Caracteres de una cadena.

Sintaxis Salida :
(subcadena "Hola" 0 2) Ho :

Ejemplo :
Código Salida :
(substring "futbol" 1 3) ut :

Comparación de cadenas.

IMAGEN 28

- ✓ Donde en la parte izquierda del programa se presenta la teoría respectiva a las cadenas [1]. Se define qué es una cadena [2], cómo crear cadenas [3], la función “string?” [4], el posible tamaño de una cadena [5], la modificación de cadenas [6], la unión de cadenas [7], los caracteres de una cadena [8] y la comparación de cadenas [9]. En la parte derecha se presentan los siguientes ítems:



Ejemplos ayuda

Representación de una cadena en Racket

make-string string? string-length string-append string-ref string-set! string=

```
1 ;Al momento de usar la función meke-string solo nos sirve para crear una cadena con los
2 ;mismos caracteres, aunque estas cadenas pueden ser modificadas
3
4 ;Crear una función que llena una cadena de 4 posiciones con el carácter J usando
5 ;make-string. Luego cree una cadena con la función string que diga "Hola que tal tu día"
6 ;y por ultimo muestra las cadenas creadas usando la función displayln
7
8 (define (llenarCadena)
9   (define cad1 (make-string 4 #j))
10  (displayln cad1)
11  (define cad2 (string #H #o #l #a #q #u #e #t #a #l #t #u #d #i #a))
12  (displayln cad2)
13 )
14
15 (llenarCadena)
```

Consola

Ejecutar

Paso a Paso

IMAGEN 29

- ✓ Donde se ve una representación gráfica de una cadena en Racket [10], un botón de ayuda que redirige al manual de usuario [11], ejemplos programados de cadenas [12] y una consola para ver los resultados de los programas anteriores [13].
- ✓ Vectores
 - Dentro del módulo de vectores, se presentan los siguientes ítems:



Vectores

¿Qué es un Vector?

matrices o (filas y columnas si tuviera dos dimensiones)

En programación, una matriz o un vector es una zona de almacenamiento continuo que contiene una serie de elementos del mismo tipo. Los elementos del vector.

El tipo de dato vector, es conocido como un tipo de dato compuesto de uno o más tipos de dato básico, primitivo o de otros tipos de datos compuestos, inclusive vectores mismos.

Creación de Vectores

Sintaxis :
(hacer-vector n char) :

Ejemplo :
(make-vector 5) :
Crea un vector de 5 posiciones vacío o con el elemento 0 :

vector-ref

(vector-referencia vect n) :
vect es el vector al cual queremos sacar uno o más datos y la "n" :
hace referencia a la posición en el vector donde queremos sacar la :
información :

Ejemplo :
(define vect (vector #\z "Pollito" 4 #t)) :
(vector-referencia vect 2) -> "Pollito" :

vector-set!

IMAGEN 30

- ✓ Donde en la parte izquierda de la pantalla se ve la teoría de los vectores [1]. Se define qué es un vector [2], cómo crear vectores [3], la función vector-ref [4], la función vector-set! [5], la función vector-fill! [6], y la función vector-length [7]. En la parte derecha se presentan los siguientes ítems.

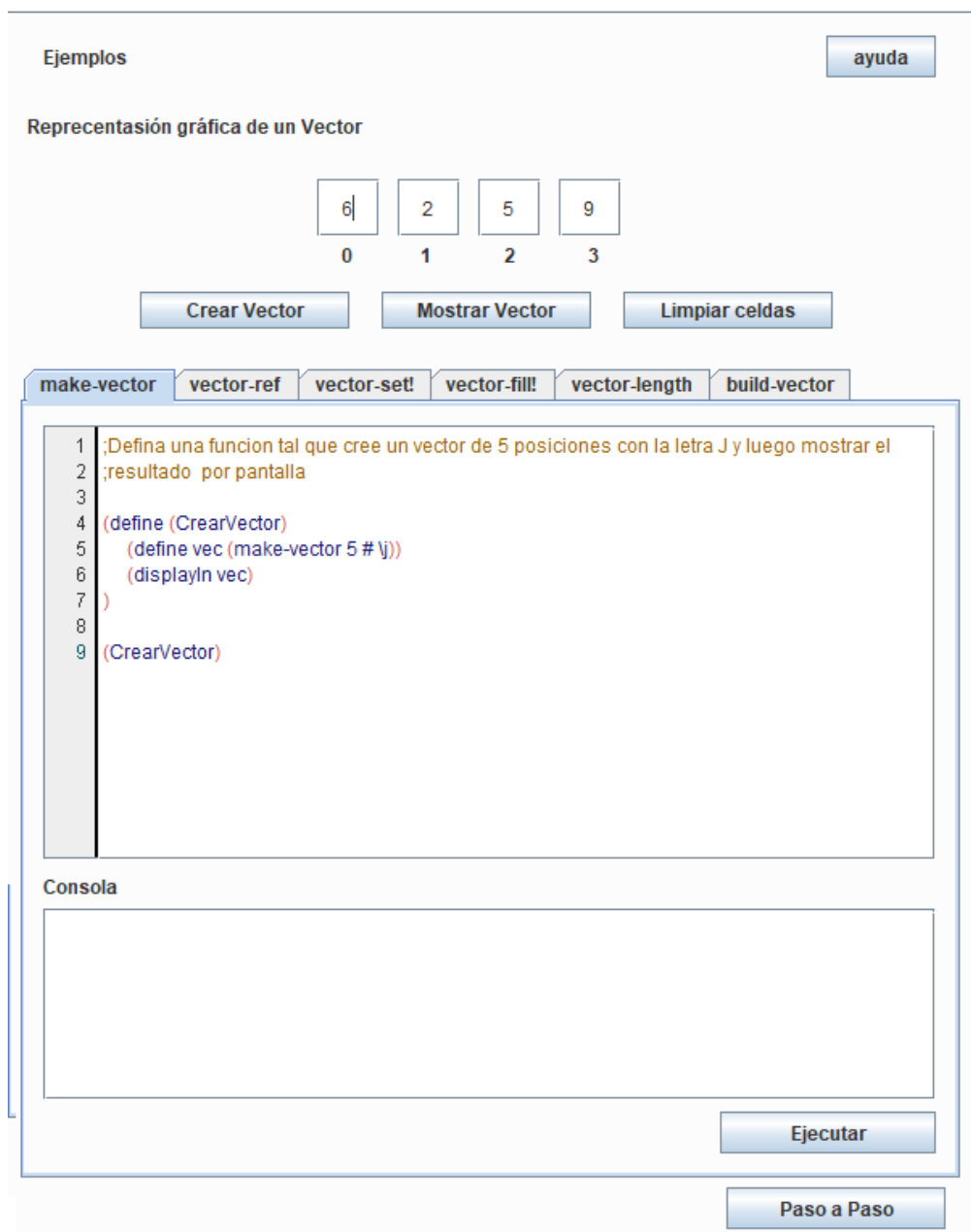


IMAGEN 31

- ✓ Donde se ejemplariza interactiva el conceptos de vectores [8], se encuentra un botón de ayuda que redirige al manual de usuario [9], se ven una serie de ejemplos codificados de vectores [10] y una consola donde saldrán los resultados de estos códigos [11].
- ✓ Pares
 - Dentro del módulo de pares, se encuentran los siguientes ítems:



Pares

¿Qué es un Par?

Un par es una estructura de datos con dos campos llamados cabeza y cola. Los pares son creados con el procedimiento cons. Se pueden agregar cualquier tipo de dato.

la Función cons

primero, la cabeza y el segundo, la cola.

Sintaxis :
(cons a b) -> la cabeza es a y la cola es b :

Ejemplo :
(define par (cons 3 "hola")) :

Car

Sintaxis :
(definir par (cons data1 data2)) :
(displayln (car par)) :

Ejemplo :
(define par (cons 3 "hola")) :
(displayln (cdr par)) -> 3 :

Cdr

IMAGEN 32

- ✓ Donde la parte izquierda de la pantalla presenta la teoría de los pares [1]. Se explica qué es un par [2], la función cons [3], la función car [4], la función cdr [5] y la función pair? [6]. En la parte derecha se presentan los siguientes ítems:

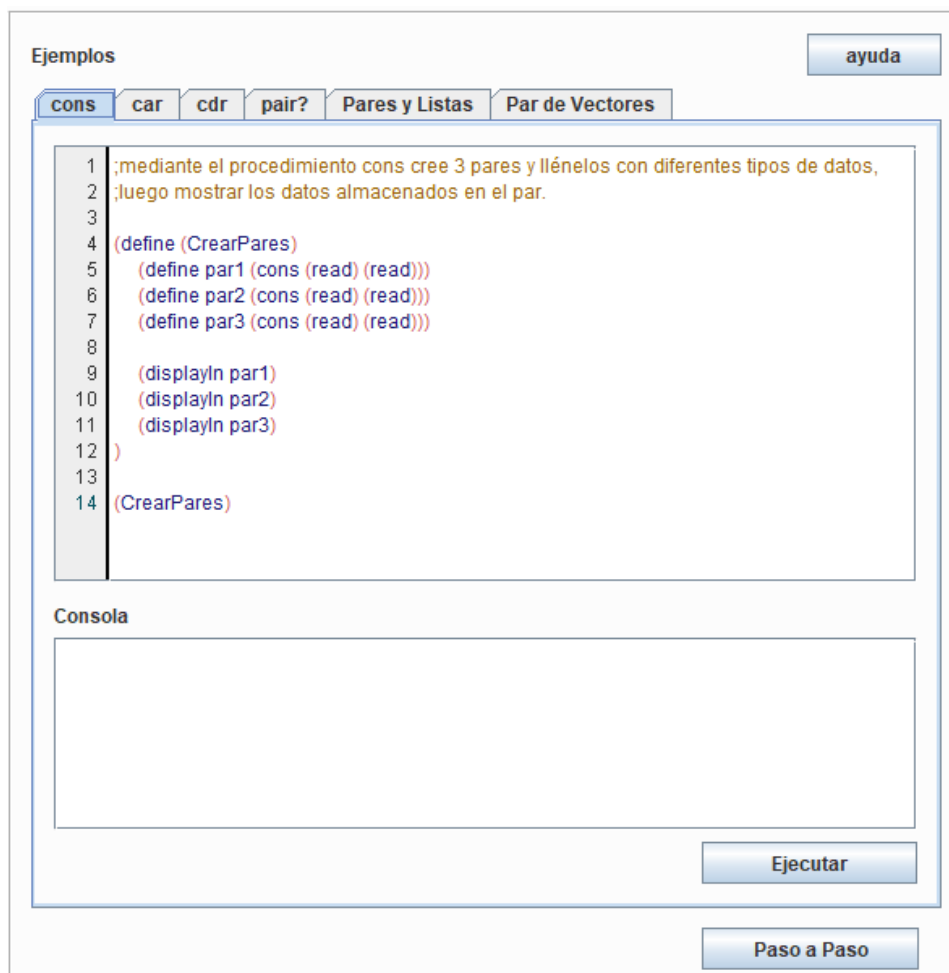


IMAGEN 33

- ✓ Donde se muestran ejemplos codificados de pares [7], un botón de ayuda que redirige al manual de usuario [8] y una consola para ver los resultados de los códigos ejecutados [9].
- ✓ Listas
 - Dentro del módulo de listas se encuentran los siguientes ítems:



Listas

¿Qué es una lista?

Una lista es un tipo de dato con cierta similitud a un vector, pero las listas se diferencian principalmente de los vectores en que estas son dinámicas es decir, se les puede añadir información y las listas siempre terminan en un espacio vacío (empty), Aunque este no se vea.

La función list

que tiene tantos como queramos incluir en la lista.

Sintaxis :
(lista data) :

Ejemplo :
(list 4 9 6 "hola" 'bl #\a (vector 8 9 4) (list 2 7 1) (+ 2 5)) :

Car

Sintaxis :
(car (list)) :

Ejemplo :
(car (list 4 5 6)) -> (4) :
(cdr (list 3 0 9)) -> (0 9) :

Null?

IMAGEN 34



Listas

Append

(definir Listas (lista datos)) :
(agregar Lista dato) :
(escribir Lista) --> datos :

Ejemplo :
(define Listas (list 1 2 3 4)) :
(append Lista 4) :
(display Lista) --> (1 2 3 4) :

List?

Sintaxis :
(definir Listas (lists datos)) :
(lista? Lista) --> #t :

Ejemplo :
(define Listas (list 1 2 3)) :
(list? Lista) --> #t :

Length

Sintaxis :
(definir Listas (lista data)) :
(tamaño Lista) :

Ejemplo :
(define Listas (list 1 2 3)) :
(length Lista) --> 3 :

Reverse

IMAGEN 35

- ✓ Donde en la parte izquierda de la pantalla se presenta la teoría sobre las listas [1]. Se define qué es una lista [2], la función list [3], la función car [4], la función null? [5], la función append [6], la función list? [7], la función length [8], la función reverse [9] y la función list-tail [10]. En la parte derecha se presentan los siguientes ítems.

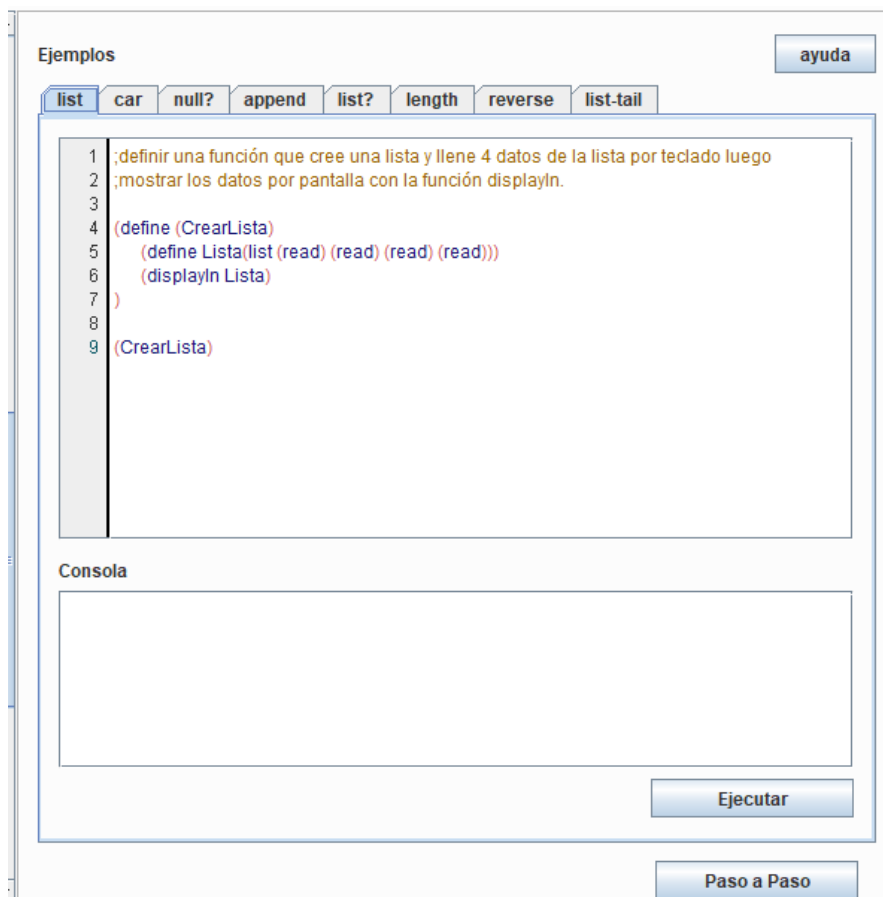


IMAGEN 36

- ✓ Donde se muestran ejemplos codificados de las funciones de listas [11], un botón de ayuda que redirige al manual de usuario [12] y una consola para mostrar los resultados de las funciones [13].
- ✓ Listas
 - Dentro del módulo de listas, se presentan los siguientes ítems:



Estructuras de datos

¿Qué es una estructura de dat...

Es una forma de organizar un conjunto de datos con el objetivo de facilitar su manipulación. Una estructura de datos define la organización e interrelación de estos y un conjunto de operaciones que se pueden realizar sobre ellos.

Define-struct

el procedimiento. (define-struct.)

Sintaxis :
(define-estructura alumnos(nombre apellido teléfono correo)) :

Ejemplo :
(define-struct alumnos(nombre apellido teléfono correo)) :

Make-struct

Veamos cómo es:
(define-struct Empleados (nombre apellido))

para la función make (make-struct) donde struct el nombre de nuestra estructura es decir:

(make-Empleados "José" "Cardona")
(make-Empleados "Luis" "Vargas")

IMAGEN 37

- ✓ Donde en la parte izquierda de la pantalla se presenta teoría sobre estructuras de datos [1]. Se define qué es una estructura de datos [2], la función define-struct [3] y la función make-struct [4]. En la parte derecha se presentan los siguientes ítems:



Ejemplos [ayuda](#)

Representación gráfica de una estructura de datos

Nombre	Apellido	Teléfono	Correo
Nicolas	Orozco	3114257895	nicolas@gmail.com
Laura	Gómez	3132452878	laura@gmail.com
Alejandro	Smith	3154856656	alejo@gmail.com
Sofia	Nixon	3058856995	nixonsofia@gmail.com

[Agregar Datos](#)

[Ejemplo 1](#) [Ejemplo 2](#) [Ejemplo 3](#) [Ejemplo 4](#) [Ejemplo 5](#)

```
1 ;Cree 5 estructuras para doctores, pacientes y Enfermeros tales que contengan los datos
2 ;nombre, apellido, teléfono y dirección
3
4 (define (CrearEstructuras)
5   (define-struct Doctores(nombre apellido telefono direccion))
6   (define-struct Enfermeros(nombre apellido telefono direccion))
7   (define-struct Clientes(nombre apellido telefono direccion))
8 )
9
10 (CrearEstructuras)
```

Consola

[Ejecutar](#)

[Paso a Paso](#)

IMAGEN 38

- ✓ Donde se propone una representación gráfica de una estructura de datos [5], un botón de ayuda que redirige al manual de usuario [6], ejemplos codificados de estructuras de datos [7] y una consola para ver los resultados de la ejecución de los programas [8].
- ✓ Modo gráfico
 - Dentro del módulo de apartado gráfico, se encuentran los siguientes ítems.



Graphics

Modo Gráfico

La gran mayoría de lenguajes de programación nos ofrece un modo gráfico o un conjunto de instrucciones y/o funciones que permiten usar ventanas, pixeles, multimedia y ratón, con el fin de desarrollar programas en modo gráfico.



Librería Graphics

Para realizar un programa en modo gráfico, primero debemos definir con que librería de gráficos vamos a programar.

En nuestro caso, será: 'Graphics'.

Para indicárselo a Racket debemos agregar la siguiente línea de código al inicio de nuestro programa.

Funciones Básicas

Operaciones con el teclado

(get-key-press Ventana) ;Espera a que el usuario presione una tecla y devuelve un descriptor

(ready-key-press Ventana) ;Esta funcion no espera a que el usuario oprima una tecla, en caso contrario devuelve un descriptor

(key-value (get-key-press Ventana)) recibe un descriptor y devuelve el carácter correspondiente a la tecla que fue presionada

IMAGEN 39

- ✓ Donde en la parte izquierda del módulo se encuentra la teoría del modo gráfico de Racket [1]. Se define el modo gráfico en Racket [2], la librería graphics [3] y las funciones básicas del modo gráfico de Racket [4]. En la parte derecha se presentan los siguientes ítems:



Ejemplos

Ayuda

Ejemplo modo gráfico y uso de botones



Ejemplo #1Ejemplo #2Ejemplo #3Ejemplo #4

```
3 (2,50), (3,50) hasta llegar a la posición (800,50). Use recursividad.
4
5 (require (lib "graphics.ss" "graphics"))
6 (open-graphics) ;Llamado de librería
7
8 (define ventanal (open-viewport "Línea de puntos" 800 600))
9 (define (línea x y) ;x, y, son coordenadas.
10   (if (= x 799) ; Caso base
11       ((draw-pixel ventanal) (make-posn 800 50) "green") ; Dibuja el ultimo pixel
12       (begin
13         ((draw-pixel ventanal) (make-posn x y) "green") ; Dibuja el pixel en la posición (x, y)
14         (sleep 0.001) ; Tiempo de espera para dibujar el siguiente pixel.
15         (línea (+ x 1) y) ; Llamado recursivo
16       )
17     )
18   )
19 (línea 0 50) ; llamado de la función
```

Consola



Ejecutar

IMAGEN 40



- ✓ Donde se presenta un ejemplo gráfico de la librería graphics de Racket [5], un botón de ayuda que redirige al manual de usuario [6], ejemplos codificados de la librería graphics [7] y una consola para ver los resultados de la ejecución de los programas [8].
- ✓ Paso a paso
 - Dentro del módulo de paso a paso se encuentran los siguientes ítems:

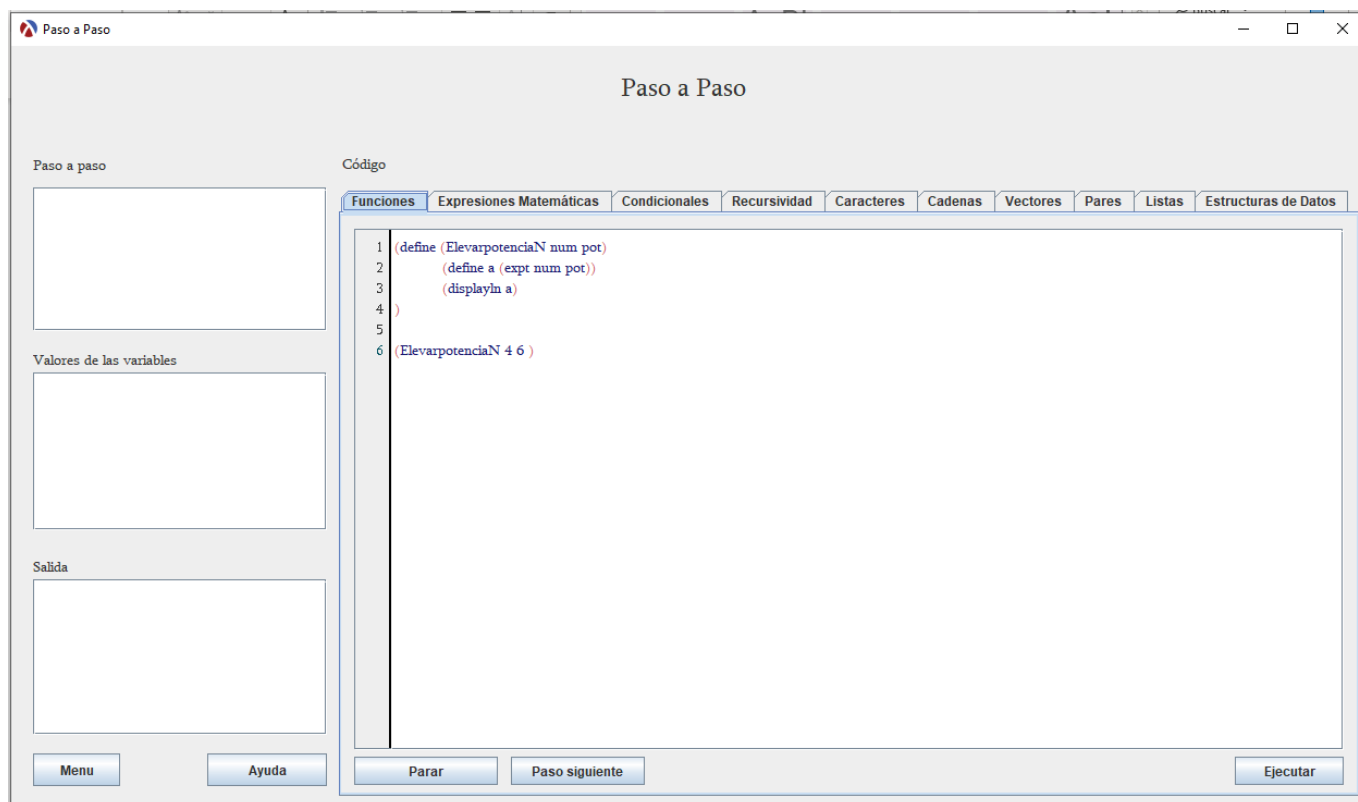


IMAGEN 41

- ✓ Donde en la parte izquierda del módulo se presenta el seguimiento de los códigos propuestos [1]. Se ve el paso a paso del código en ejecución y los valores de las variables [2], las salidas de las funciones [3], un botón de ayuda que retorna al menú al principal [4] y un botón de ayuda que redirige al manual de usuario [5]. En la parte derecha del módulo se presentan los códigos a ejecutar para llevar su seguimiento [6], un botón de parar para detener la instancia de ejecución del código [7], un botón de paso siguiente para ver el siguiente estado del código [8] y el botón de ejecutar todo el código seleccionado [9].
- ✓ ¿Te atreves?
 - En el módulo final llamado ¿te atreves? se encuentran los siguientes ítems:

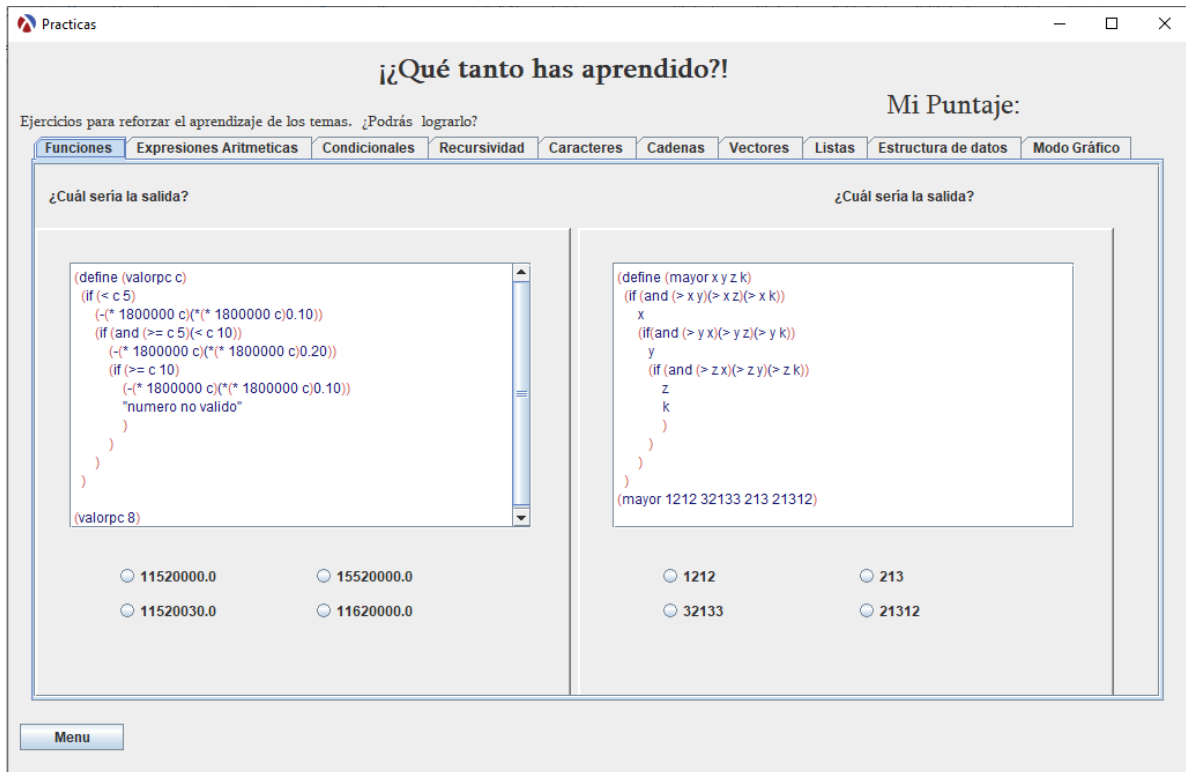


IMAGEN 42

- ✓ Donde se presentan una serie de ejercicios relacionados a los módulos anteriores [1]. Se presenta un examen donde se realiza una pregunta [2], un código a evaluar [3] y una serie de posibles respuestas a elegir [4]. Se presenta un botón de menú que redirige al menú principal [5] y un botón llamado “mi puntaje” el cual refleja el puntaje del examen [6].