

GUIA DE COMPARACIÓN DE LENGUAJE RACKET Y C MANUAL DE USUARIO



**Universidad Tecnológica
de Pereira**



**Msc Luis Eduardo Muñoz Guerrero
Universidad Tecnológica de Pereira
(UTP)**



CONTENIDO

TABLA DE IMÁGENES	3
SOBRE ESTE MANUAL	4
PROPÓSITO	5
INTRODUCCIÓN	5
CARACTERÍSTICAS DEL ENTORNO	5
USO DEL SOFTWARE	6
INGRESO AL SOFTWARE	6
USO DEL APLICATIVO	7
PANTALLA PRINCIPAL	8
MENÚ DE MÓDULOS	8
MÓDULO 1: EXPRESIONES	11
MÓDULO 2: ESTRUCTURAS DE SECUENCIA	12
MÓDULO 3: ESTRUCTURAS DE SELECCIÓN	14
MÓDULO 4: ESTRUCTURAS DE ITERACIÓN	15
MÓDULO 5: CADENAS Y CARACTERES	16
MÓDULO 6: VECTORES	18
MÓDULO 7: MATRICES	19
MÓDULO 8: ESTRUCTURAS DE DATOS	20
MÓDULO 9: FUNCIONES	22
MÓDULO 10: COMPOSICIÓN DE FUNCIONES	23
MÓDULO 11: PROCEDIMIENTOS	24
MÓDULO 12: MODO GRÁFICO	26
MÓDULOS DE PRÁCTICA: PRÁCTICA C	27
MÓDULOS DE PRÁCTICA: PRÁCTICA RACKET	28



TABLA DE IMÁGENES

Imagen 1: Barra de búsqueda	6
Imagen 2: dirección del software	6
Imagen 3: Ingreso a la pantalla principal del software	7
Imagen 4: Pantalla principal del software	8
Imagen 5: Módulos 1, 2 y 3	9
Imagen 6: Módulos 4, 5 y 6	9
Imagen 7: Módulos 7, 8 y 9	10
Imagen 8: Módulos 10, 11 y 12	10
Imagen 9: Módulos de práctica	11
Imagen 10: Módulo de expresiones	11
Imagen 11: Módulo de expresiones completo	12
Imagen 12: Módulo de estructuras de secuencias	13
Imagen 13: Módulo de estructuras de secuencia completo	14
Imagen 14: Módulo de estructuras de selección	14
Imagen 15: Módulo de estructuras de selección completo	15
Imagen 16: Módulo de estructuras de iteración	16
Imagen 17: Módulo de estructuras de iteración completo	16
Imagen 18: Módulo de cadenas y caracteres	17
Imagen 19: Módulo de cadenas y caracteres completo	18
Imagen 20: Módulo de vectores	18
Imagen 21: Módulo de vectores completo	19
Imagen 22: Módulo de matrices	20
Imagen 23: Módulo de matrices completo	20
Imagen 24: Módulo de estructuras de datos	21
Imagen 25: Módulo de estructuras de datos completo	22
Imagen 26: Módulo de funciones	22



	4
Imagen 27: Módulo de funciones completo	23
Imagen 28: Módulo de composición de funciones	24
Imagen 29: Módulo de composición de funciones completo	24
Imagen 30: Módulo de procedimientos	25
Imagen 31: Módulo de procedimientos completo	26
Imagen 32: Módulo de modo gráfico	26
Imagen 33: Módulo de modo gráfico completo	27
Imagen 34: Módulo de práctica en C	28
Imagen 35: Módulo de práctica en Racket	28

SOBRE ESTE MANUAL

Este manual de usuario es redactado para el software “guía comparativa Racket y C”. En este se busca dar un panorama al usuario de cómo está estructurado el software y la manera correcta de usarlo.



PROPÓSITO

Este software es una herramienta pedagógica, que tiene como objetivo dar continuidad y fortalecer los fundamentos básicos de la programación de estudiantes de ingeniería de sistemas, mediante la comparación entre los lenguajes Racket y C, además brinda un material estructurado y secuencial de las diferentes temáticas de programación avanzada. El sitio web se basa en el autoaprendizaje a través de fundamentos teóricos, ejemplos y prácticas que permiten alcanzar las competencias de un programador avanzado.

INTRODUCCIÓN

Esta Guía de comparación de lenguaje Racket y C se crea con el objetivo de ser un material de apoyo para los estudiantes en su proceso de transición de la programación declarativa a la imperativa.

Este software cuenta con 12 módulos los cuales permitirán al estudiante visualizar los diversos temas que se abordan en la programación imperativa desde el paradigma declarativo, como las funciones, las expresiones expresiones, las estructuras de secuencia, selección e iteración, el manejo de arreglos y las estructuras de datos.

Cada módulo cuenta con una sección de ejemplos donde se podrán visualizar problemas típicos desde las dos perspectivas, además se cuenta con una sección de ejercicios, donde se proponen clásicos enunciados de la programación desde la perspectiva declarativa para solucionarlos en la imperativa.

Cada ejemplo posee un código que se puede compilar desde la misma página para llevar seguimiento a las entradas y salidas del mismo con el fin de mejorar la experiencia del usuario.

OBJETIVOS

- Instruir los fundamentos de la programación funcional en el lenguaje de programación C según las bases adquiridas de la programación declarativa en el lenguaje de programación Racket.
- Recopilar los conocimientos fundamentales de la programación funcional en una guía web interactiva.
- Formar programadores aptos mediante la comprensión de los temas principales de la programación funcional según las analogías del paradigma declarativo.
- Mejorar el nivel de abstracción de los estudiantes mediante el uso de ejemplos y ejercicios documentados.

CARACTERÍSTICAS DEL ENTORNO

Requisitos mínimos de hardware y software

- Monitor con resolución de 1024 x 768 o superior.
- Sistema operativo: Windows xp o versiones superiores, Linux en sus diferentes distribuciones: Ubuntu, Debian, etc.

- Procesador de 1.6GHz o superior.
- Memoria RAM de 1Gb o superior.
- Navegadores web: Mozilla Firefox versión 23 o superior o Google Chrome versión 21 o superior.
- Conexión a internet

USO DEL SOFTWARE

El software Guia de comparación de lenguaje Racket y C se desarrolla para entornos web, por lo que uso debe darse con conexión a internet y para su uso se debe ingresar desde una página web. De esta manera, el método de entrar al dominio del software es el siguiente:

INGRESO AL SOFTWARE

1. Se ingresa a un navegador web desde un equipo de cómputo con conexión a internet.
2. En la barra de búsqueda, ingresar la dirección del software de la siguiente manera:

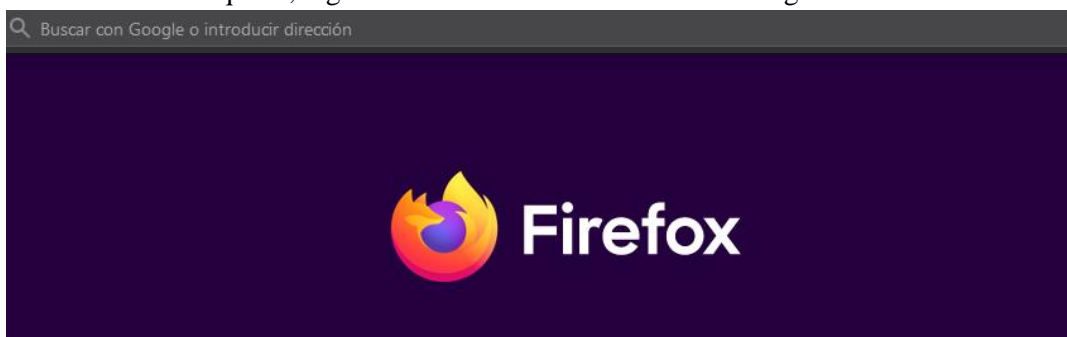


Imagen 1: Barra de búsqueda

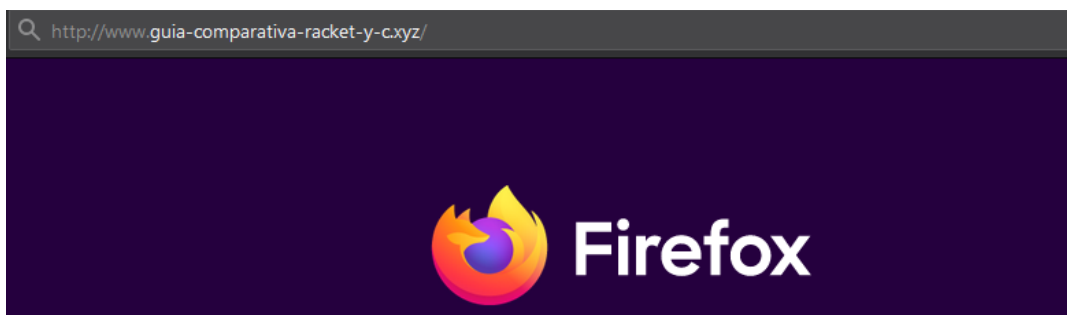


Imagen 2: dirección del software

3. Se presiona la tecla enter y se presenta la pantalla principal del software



Esta aplicación web fue desarrollada por Msc Luis Eduardo Muñoz Guerrero, Docente de planta en la Universidad Tecnológica de Pereira usando las siguientes tecnologías:



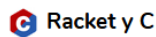
Imagen 3: Ingreso a la pantalla principal del software

USO DEL APLICATIVO

Una vez ingresado en el software mediante un navegador web, se presenta la siguiente pantalla principal con los siguientes aspectos:



PANTALLA PRINCIPAL

[Módulos](#)[Práctica](#)[Inicio](#)

Guía de comparación de lenguaje Racket y C

Este sitio web tiene como objetivo ser un instrumento que sirva de apoyo a los estudiantes de la Universidad Tecnológica de Pereira para el aprendizaje del lenguaje de programación C a través de conocimientos previos de Scheme, también conocido como Racket. En esta herramienta se podrán encontrar variedad de ejemplos con los que el estudiante podrá mejorar su nivel de abstracción y obtener el conocimiento de una manera práctica.

Información Adicional

Este sitio web cuenta con 12 módulos, cada módulo cuenta con ejemplos prácticos sobre determinados temas, el estudiante es libre de escoger a cual módulo quiere ingresar, nuestra recomendación es que antes de ingresar a un módulo, visite el módulo previo, para así poder obtener un mejor entendimiento de los temas que va a ver en el módulo al que desea entrar.

[¡Aprende más!](#)

Esta aplicación web fue desarrollada por Msc Luis Eduardo Muñoz Guerrero, Docente de planta en la Universidad Tecnológica de Pereira usando las siguientes tecnologías:



Imagen 4: Pantalla principal del software

Donde se presentan los siguientes apartados (de arriba a abajo e izquierda a derecha)

- Guía de comparación de lenguaje Racket y C e Información adicional: Textos que explican y detallan el software.
- Módulos: Botón interactivo con el que se navega a la sección de módulos
- Práctica: Botón interactivo con el que se navega a la sección de práctica
- Inicio: Botón interactivo con el que se navega a la ventana principal
- ¡Aprende más!: Botón interactivo con el que se navega a la siguiente sección, la de módulos
- Logos: Se presentan los logos de la Universidad Tecnológica de Pereira, el lenguaje de programación Scheme y el lenguaje de programación Racket. En la parte inferior, se presenta la autoría del software y las tecnologías implementadas para su desarrollo.

MENÚ DE MÓDULOS

Una vez presionado el botón de ¡Aprende más! el software redirige a la siguiente sección correspondiente al menú de módulos el cual posee un total de 12 módulos de aprendizaje divididos en 3 tipos de dificultades (principiante, medio y avanzado) y 2 módulos de práctica. Los módulos son los siguientes:

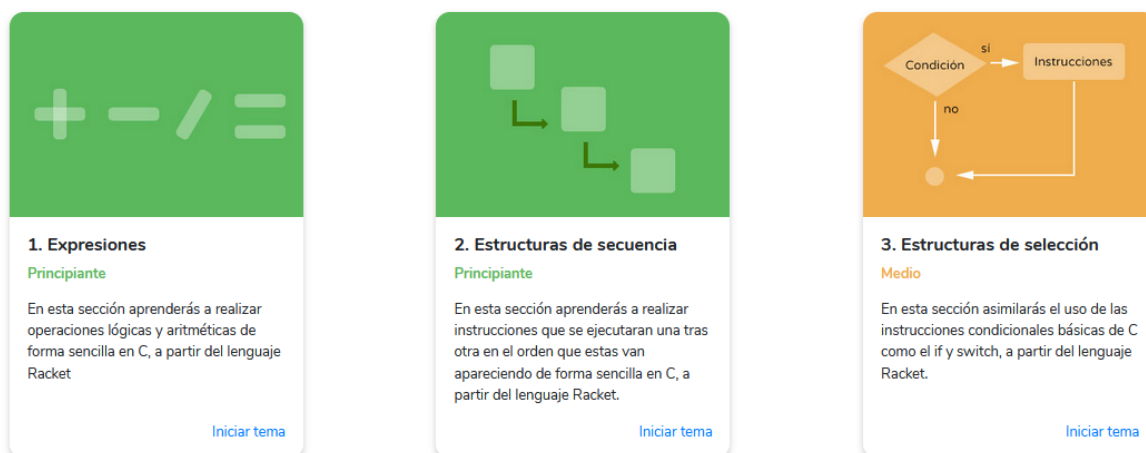


Imagen 5: Módulos 1, 2 y 3

Se presentan los módulos 1, 2 y 3 correspondientes a expresiones, estructuras de secuencias y estructuras de selección.



Imagen 6: Módulos 4, 5 y 6

Se presentan los módulos 4, 5 y 6 correspondientes a estructuras de iteración, cadenas y caracteres y vectores.



10



Imagen 7: Módulos 7, 8 y 9

Se presentan los módulos 7, 8 y 9 correspondientes a matrices, estructuras de datos y funciones.



Imagen 8: Módulos 10, 11 y 12

Se presentan los módulos 10, 11 y 12 correspondientes a composición de funciones, procedimientos y modo gráfico.

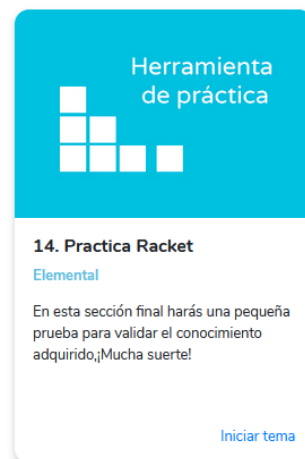
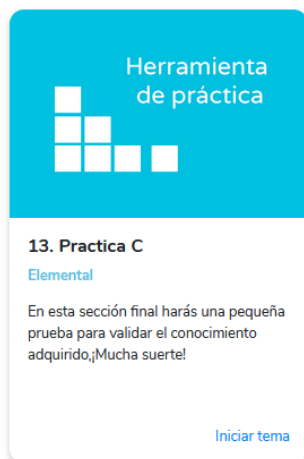


Imagen 9: Módulos de práctica

Finalmente se presenta el módulo de práctica donde se puede ejecutar código de C en la web.

MÓDULO 1: EXPRESIONES

Desde el menú de módulos se puede ingresar a cualquier módulo, presionando click en “Iniciar tema” sobre el módulo 1, se presenta la siguiente vista:



Imagen 10: Módulo de expresiones

Donde se presenta un título correspondiente, una serie de botones interactivos con diferentes ejemplos y dos cajas de texto principales con el ejemplo actual en Racket y C.



12

La pestaña mostrará inicialmente el código en el lenguaje de programación Racket el cual se puede ejecutar, pero cuando el usuario presione en el botón “Ir a C”, entonces aparece el código de la siguiente manera:

Expresiones

1. Ejemplo2. Ejemplo3. Ejemplo4. Ejemplo5. Ejemplo6. Ejemplo

```
1 ;Construir un programa tal que nos muestre el
2 ;Perímetrona circunferencia
3 (define (perimetro)
4   (define radio 0)(define pi 3.141592)
5   (printf "Ingrese el radio: ")
6   (set! radio (read))
7   (printf "El perimetro es: ~a" (* (* 2 pi) radio))
8 )
9
10 (perimetro)
```

Ir a C

```
1 //Construir un programa tal que nos muestre el
2 //Perímetrona circunferencia
3 #include <stdio.h>
4 int main(){
5   float radio, pi = 3.141592;
6   printf("Ingrese el radio: ");
7   scanf("%d", &radio);
8   printf("El perimetro es: %d", ((2 * pi)* radio));
9 }
```

El perimetro es: 21.68171068412159

Ejecutar

Comparar

El perimetro es: 132.27892039874368

Ejecutar

Imagen 11: Módulo de expresiones completo

MÓDULO 2: ESTRUCTURAS DE SECUENCIA

Desde el menú de módulos se puede ingresar a cualquier módulo, presionando click en “Iniciar tema” sobre el módulo 2, se presenta la siguiente vista:



Estructuras de Secuencia

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

```
1 ;Cree e inicialice con cualquier dato del mismo tipo:
2 ;~2 variables del tipo de dato entero.
3 ;~1 variable de tipo de dato char.
4 ;Una vez inicializados todos los datos sume los dos enteros y
5 ;almacenelos en una variable.
6 (define (main)
7   (define variable1 0)(define variable2 0)
8   (set! variable1 5)
9   (set! variable2 8)
10  (define caracter "h")
11  (define suma (+ variable1 variable2))
12 )
13
14 (main)
```

Ir a C

output

Ejecutar

Comparar

Imagen 12: Módulo de estructuras de secuencias

Donde se presenta un título correspondiente, una serie de botones interactivos con diferentes ejemplos y dos cajas de texto principales con el ejemplo actual en Racket y C.

La pestaña mostrará inicialmente el código en el lenguaje de programación Racket el cual se puede ejecutar, pero cuando el usuario presione en el botón “Ir a C”, entonces aparece el código de la siguiente manera:

Estructuras de Secuencia

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

```
1 ;Cree e inicialice con cualquier dato del mismo tipo:
2 ;~2 variables del tipo de dato entero.
3 ;~1 variable de tipo de dato char.
4 ;Una vez inicializados todos los datos sume los dos enteros y
5 ;almacenelos en una variable.
6 (define (main)
7   (define variable1 0)(define variable2 0)
8   (set! variable1 5)
9   (set! variable2 8)
10  (define caracter "h")
11  (define suma (+ variable1 variable2))
12 )
13
14 (main)
```

Ir a C

```
1 /*Cree e inicialice con cualquier dato del mismo tipo:
2 ~2 variables del tipo de dato entero.
3 ~1 variable de tipo de dato char.
4 Una vez inicializados todos los datos sume los dos enteros y
5 almacenelos en una variable.*/
6 int main(){
7   int variable1, variable2;
8   variable1=5;
9   variable2=8;
10  char caracter="h";
11  int suma = variable1+variable2;
12 }
```

13

Ejecutar

Comparar

Ejecutar



14

Imagen 13: Módulo de estructuras de secuencia completo

MÓDULO 3: ESTRUCTURAS DE SELECCIÓN

Desde el menú de módulos se puede ingresar a cualquier módulo, presionando click en “Iniciar tema” sobre el módulo 3, se presenta la siguiente vista:

Estructuras de Selección

[1. Ejemplo](#) [2. Ejemplo](#) [3. Ejemplo](#) [4. Ejemplo](#) [5. Ejemplo](#) [6. Ejemplo](#)

```
1 ;Escribir un programa que detecte si un número introducido
2 ;desde el teclado es positivo o negativo.
3 (define (main num)
4   (if (< num 0)
5     (printf "Numero negativo: ~a" num)
6     (printf "Numero positivo: ~a" num)
7   )
8 )
9
10 (main (read))
```

output

[Ejecutar](#)

[Ir a C](#)

```
output
```

[Comparar](#)

Imagen 14: Módulo de estructuras de selección

Donde se presenta un título correspondiente, una serie de botones interactivos con diferentes ejemplos y dos cajas de texto principales con el ejemplo actual en Racket y C.

La pestaña mostrará inicialmente el código en el lenguaje de programación Racket el cual se puede ejecutar, pero cuando el usuario presione en el botón “Ir a C”, entonces aparece el código de la siguiente manera:



15

Estructuras de Selección

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

```
1 //Escribir un programa que detecte si un número introducido
2 //desde el teclado es positivo o negativo.
3 (define (main num)
4   (if (< num 0)
5     (printf "Numero negativo: ~a" num)
6     (printf "Numero positivo: ~a" num)
7   )
8 )
9
10 (main (read))
```

Numero positivo: 16

Ejecutar

Ir a C

```
1 //Escribir un programa que detecte si un número introducido
2 //desde el teclado es positivo o negativo.
3 int main (){
4   int num;
5   scanf("%d", &num);
6   if (num < 0){
7     printf ("Numero negativo: %d", num);
8     printf ("Numero positivo: %d", num);
9   }
10 }
11
```

Numero negativo: -46

Ejecutar

Comparar

Imagen 15: Módulo de estructuras de selección completo

MÓDULO 4: ESTRUCTURAS DE ITERACIÓN

Desde el menú de módulos se puede ingresar a cualquier módulo, presionando click en “Iniciar tema” sobre el módulo 4, se presenta la siguiente vista:

Estructuras de iteración

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

```
1 //Construir un programa que muestre los números del 1 hasta
2 //el 10 utilizando la estructura de iteracion for
3 (define (numeros)
4   (for ([contador (in-range 1 11)])
5     (printf "~n~a" contador)
6   )
7 )
8 (numeros)
9
```

output

Ejecutar

Ir a C

```
output
```

Comparar



16

Imagen 16: Módulo de estructuras de iteración

Donde se presenta un título correspondiente, una serie de botones interactivos con diferentes ejemplos y dos cajas de texto principales con el ejemplo actual en Racket y C.

La pestaña mostrará inicialmente el código en el lenguaje de programación Racket el cual se puede ejecutar, pero cuando el usuario presione en el botón “Ir a C”, entonces aparece el código de la siguiente manera:

Estructuras de iteración

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

```
1 ;Construir un programa que muestre los números del 1 hasta
2 ;el 10 utilizando la estructura de iteracion for
3 (define (numeros)
4   (for ([contador (in-range 1 11 1)])
5     (printf "~n~a" contador)
6   )
7 )
8 (numeros)
9
```

1 2 3 4 5 6 7 8 9 10

Ejecutar

Ir a C

Comparar

```
1 /*Construir un programa que muestre los números del 1 hasta
2 el 10 utilizando la estructura de iteracion for*/
3 #include <stdio.h>
4 int main()
5 {
6   int contador=0;
7   for( contador=1; contador<11 ; contador++ ){
8     printf("~n%d",contador);
9   }
10  return 0;
11 }
```

1 2 3 4 5 6 7 8 9 10

Ejecutar

Imagen 17: Módulo de estructuras de iteración completo

MÓDULO 5: CADENAS Y CARACTERES

Desde el menú de módulos se puede ingresar a cualquier módulo, presionando click en “Iniciar tema” sobre el módulo 5, se presenta la siguiente vista:



Cadenas y Caracteres

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

```
1 ;Cree en programa que capture una cadena y luego
2 ;convierta cada uno de los caracteres a letras mayúsculas.
3 (define (cadena)
4   (define cad "")
5   (define cadMayus "")
6   (printf "Ingrese una cadena: ")
7   (set! cad (string-copy (read)))
8   (set! cadMayus (string-upcase cad))
9   (printf "\nLa cadena en mayusculas es: ~a" cadMayus)
10 )
11
12 (cadena)
```

Ir a C

Ejecutar

Comparar

Imagen 18: Módulo de cadenas y caracteres

Donde se presenta un título correspondiente, una serie de botones interactivos con diferentes ejemplos y dos cajas de texto principales con el ejemplo actual en Racket y C.

La pestaña mostrará inicialmente el código en el lenguaje de programación Racket el cual se puede ejecutar, pero cuando el usuario presione en el botón “Ir a C”, entonces aparece el código de la siguiente manera:

Cadenas y Caracteres

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

```
1 ;Cree en programa que capture una cadena y luego
2 ;convierta cada uno de los caracteres a letras mayúsculas.
3 (define (cadena)
4   (define cad "")
5   (define cadMayus "")
6   (printf "Ingrese una cadena: ")
7   (set! cad (string-copy (read)))
8   (set! cadMayus (string-upcase cad))
9   (printf "\nLa cadena en mayusculas es: ~a" cadMayus)
10 )
11
12 (cadena)
```

Ir a C

```
1 /*Cree en programa que capture una cadena y luego
2 convierta cada uno de los caracteres a letras mayusculas.*/
3 #include <stdio.h>
4 #include <string.h>
5 #include <ctype.h>
6 int main(){
7   int i, j;
8   char cadena[20];
9   char cadenaMayus[20];
10  printf("Ingrese una cadena: ");
11  scanf("%s", cadena);
12  for(i=0; i<strlen(cadena); i++){
13    cadenaMayus[i] =toupper(cadena[i]);
14  }
15  printf("\nLa cadena en mayusculas es: %s", cadenaMayus);
16  }
17
```

Ingrese una cadena:
Hola Samuel
La cadena en mayusculas es: HOLA SAMUEL

1 2 3 4 5 6 7 8 9 10

Ejecutar

Comparar

Ejecutar



18

Imagen 19: Módulo de cadenas y caracteres completo

MÓDULO 6: VECTORES

Desde el menú de módulos se puede ingresar a cualquier módulo, presionando click en “Iniciar tema” sobre el módulo 6, se presenta la siguiente vista:

Vectores

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

```
1 ;Crear un vector de enteros que tenga tamaño igual a 3,  
2 ;inicialicelo con valores al azar, luego use un ciclo para  
3 ;imprimirlo. Seguidamente modificar la segunda posición del  
4 ;vector e imprimir los cambios  
5 (define (setVector)  
6   (define vect (vector 1 2 3))  
7   (printf "El vector es: \n")  
8   (for ([i] (in-range 0 3 1)))  
9     (printf "~a " (vector-ref vect i))  
10  )  
11  (vector-set! vect 1 5)  
12  (printf "El vector modificado es: \n")  
13  (for ([i] (in-range 0 3 1)))  
14    (printf "~a " (vector-ref vect i))  
15  )  
16  )  
17  
18  (setVector)  
19
```

Ir a C

Ejecutar

Comparar

Imagen 20: Módulo de vectores

Donde se presenta un título correspondiente, una serie de botones interactivos con diferentes ejemplos y dos cajas de texto principales con el ejemplo actual en Racket y C.

La pestaña mostrará inicialmente el código en el lenguaje de programación Racket el cual se puede ejecutar, pero cuando el usuario presione en el botón “Ir a C”, entonces aparece el código de la siguiente manera:



Vectores

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

```
1 ;Crear un vector de enteros que tenga tamaño igual a 3,
2 ;inicialcelo con valores al azar, luego use un ciclo para
3 ;imprimirlo. Seguidamente modificar la segunda posición del
4 ;vector e imprimir los cambios
5 (define (setVector)
6   (define vect (vector 1 2 3))
7   (printf "El vector es: \n")
8   (for ([i (in-range 0 3 1)])
9     (printf "~a " (vector-ref vect i))
10    )
11   (vector-set! vect 1 5)
12   (printf "El vector modificado es: \n")
13   (for ([i (in-range 0 3 1)])
14     (printf "~a " (vector-ref vect i))
15    )
16   )
17 )
18 (setVector)
19
```

El vector es:
1 2 3

El vector modificado es:
1 5 3

Ejecutar

Ir a C

```
1 /*Crear un vector de enteros que tenga tamaño igual a 3,
2 inicialcelo con valores al azar, luego use un ciclo para
3 imprimirlo. Seguidamente modificar la segunda posición del
4 vector e imprimir los cambios*/
5 #include <stdio.h>
6 int main(){
7   int i=0;
8   int vector[3]={1,2,3};
9   printf("El vector es: \n");
10  for(i=0; i<3; i++){
11    printf("%d ", vector[i]);
12  }
13  vector[1]=5;
14  printf("\nEl vector modificado es: \n");
15  for(i=0; i<3; i++){
16    printf("%d ", vector[i]);
17  }
18 }
19
```

El vector es:
1 2 3

El vector modificado es:
1 5 3

Ejecutar

Comparar

Imagen 21: Módulo de vectores completo

MÓDULO 7: MATRICES

Desde el menú de módulos se puede ingresar a cualquier módulo, presionando click en “Iniciar tema” sobre el módulo 7, se presenta la siguiente vista:

Matrices

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

```
1 ;Crear una matriz de enteros de orden 3x4, inicializarla
2 ;con los numeros en orden ascendente del 1 al 12, finalmente
3 ;imprimir la matriz.
4 (require math/matrix)
5 (define (ascendente)
6   (define valor 1)
7   (define mat (make-matrix 3 4 0))
8   (for ([i (in-range 0 3 1)])
9     (for ([j (in-range 0 4 1)])
10      (matrix-set mat i (set! valor (+ valor 1)))
11    )
12    )
13   (printf "\n***La matriz digitada es: ***\n")
14   (for ([i (in-range 0 3 1)])
15     (for ([j (in-range 0 4 1)])
16       (printf "~a " (matrix-ref mat i j))
17     )
18   )
19 )
20 )
21 (ascendente)
```

Ejecutar

Ir a C

Comparar



20

Imagen 22: Módulo de matrices

Donde se presenta un título correspondiente, una serie de botones interactivos con diferentes ejemplos y dos cajas de texto principales con el ejemplo actual en Racket y C.

La pestaña mostrará inicialmente el código en el lenguaje de programación Racket el cual se puede ejecutar, pero cuando el usuario presione en el botón “Ir a C”, entonces aparece el código de la siguiente manera:

Matrices

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

```
1 ;Crear una matriz de enteros de orden 3x4, inicializarla
2 ;con los numeros en orden ascendente del 1 al 12, finalmente
3 ;imprimir la matriz.
4 (require math/matrix)
5 (define (ascendente)
6   (define valor 1)
7   (define mat (make-matrix 3 4 0))
8   (for ([i (in-range 0 3 1)])
9     (for ([j (in-range 0 4 1)])
10      (matrix-set mat i j (set! valor (+ valor 1)))))
11   )
12   )
13   (printf "\n***La matriz digitada es: ***\n")
14   (for ([i (in-range 0 3 1)])
15     (for ([j (in-range 0 4 1)])
16      (printf "~a " (matrix-ref mat i j))))
17   )
18   )
19   )
20   )
21 (ascendente)
```

output

Ejecutar

Ir a C

Comparar

```
1 /*Crear una matriz de enteros de orden 3x4, inicializarla
2 con los numeros en orden ascendente del 1 al 12, finalmente
3 imprimir la matriz*/
4 #include <stdio.h>
5 int main(){
6   int i, j, valor=1;
7   int matriz[3][4];
8   for(i=0; i<3; i++){
9     for(j=0; j<4; j++){
10      matriz[i][j]=valor++;
11    }
12  }
13  printf("\n\n***La matriz digitada es: ***\n");
14  for(i=0; i<3; i++){
15    printf("\n");
16    for(j=0; j<4; j++){
17      printf("%d ", matriz[i][j]);
18    }
19  }
20 }
21
```

output

Ejecutar

Imagen 23: Módulo de matrices completo

MÓDULO 8: ESTRUCTURAS DE DATOS

Desde el menú de módulos se puede ingresar a cualquier módulo, presionando click en “Iniciar tema” sobre el módulo 8, se presenta la siguiente vista:



Estructuras de Datos

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

```
1 ;Crear en programa en Racket que tenga una estructura para almacenar fechas,
2 ;Leer e imprimir dos estructuras de este tipo.
3 (define-struct fecha(
4   dia
5   mes
6   anio
7 ))
8
9
10 (define (main)
11   (printf "Ingrese una fecha de nacimiento: ")
12   (define Nacimiento(make-fecha
13     (begin (printf "\nDia: ") (read))
14     (begin (printf "\nMes: ") (read))
15     (begin (printf "\nAño: ") (read))
16   ))
17 )
18 (printf "\nIngrese la fecha del día de hoy: ")
19 (define F_Actual(make-fecha
20   (begin (printf "\nDia: ") (read))
21   (begin (printf "\nMes: ") (read))
```

Ir a C

Ejecutar

Comparar

Imagen 24: Módulo de estructuras de datos

Donde se presenta un título correspondiente, una serie de botones interactivos con diferentes ejemplos y dos cajas de texto principales con el ejemplo actual en Racket y C.

La pestaña mostrará inicialmente el código en el lenguaje de programación Racket el cual se puede ejecutar, pero cuando el usuario presione en el botón “Ir a C”, entonces aparece el código de la siguiente manera:

Estructuras de Datos

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

```
1 ;Crear en programa en Racket que tenga una estructura para almacenar fechas,
2 ;Leer e imprimir dos estructuras de este tipo.
3 (define-struct fecha(
4   dia
5   mes
6   anio
7 ))
8
9
10 (define (main)
11   (printf "Ingrese una fecha de nacimiento: ")
12   (define Nacimiento(make-fecha
13     (begin (printf "\nDia: ") (read))
14     (begin (printf "\nMes: ") (read))
15     (begin (printf "\nAño: ") (read))
16   ))
17 )
18 (printf "\nIngrese la fecha del día de hoy: ")
19 (define F_Actual(make-fecha
20   (begin (printf "\nDia: ") (read))
21   (begin (printf "\nMes: ") (read))
```

Ir a C

Ejecutar

Comparar

Ejecutar



MÓDULO 9: FUNCIONES

Desde el menú de módulos se puede ingresar a cualquier módulo, presionando click en “Iniciar tema” sobre el módulo 9, se presenta la siguiente vista:

Funciones

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

```
1 :Cree una funcion que evalúe si un número es par o impar.  
2 (define (esperar n)  
3   (if (= (remainder n 2) 0)  
4     1  
5     0)  
6 )  
7 )  
8 )  
9 (define (main)  
10  (define numero 0)  
11  (printf "Ingrese un numero: ")  
12  (set! numero (read))  
13  (if (= (esperar numero) 1)  
14    (printf "\nEl numero ingresado es Par.")  
15    (printf "\nEl numero ingresado es Impar."))  
16 )  
17 )  
18 )  
19 (main)
```

output

Ejecutar

Ir a C

output

Comparar

Imagen 26: Módulo de funciones

Donde se presenta un título correspondiente, una serie de botones interactivos con diferentes ejemplos y dos cajas de texto principales con el ejemplo actual en Racket y C.

La pestaña mostrará inicialmente el código en el lenguaje de programación Racket el cual se puede ejecutar, pero cuando el usuario presione en el botón “Ir a C”, entonces aparece el código de la siguiente manera:



Funciones

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

```
1 ;Cree una funcion que evalúe si un número es par o impar.
2 (define (esperar n)
3   (if (= (remainder n 2) 0)
4     1
5     0)
6 )
7
8
9 (define (main)
10  (define numero 0)
11  (printf "Ingrese un numero: ")
12  (set! numero (read))
13  (if (= (esperar numero) 1)
14    (printf "\nEl numero ingresado es Par. ")
15    (printf "\nEl numero ingresado es Impar. "))
16  )
17 )
18
19 (main)
```

Ir a C

```
1 /*Cree una funcion que evalúe si un número es par o impar.*/
2 #include <stdio.h>
3 int esperar(int n){
4   if(n%2==0){
5     return 1;
6   }else return 0;
7 }
8
9 int main(){
10  int numero;
11  printf("Ingrese un numero: ");
12  scanf("%d", &numero);
13  if(esperar(numero) == 1){
14    printf("\nEl numero ingresado es Par. ");
15  }else printf("\nEl numero ingresado es Impar. ");
16 }
```

output

Ejecutar

output

Comparar

Ejecutar

Imagen 27: Módulo de funciones completo

MÓDULO 10: COMPOSICIÓN DE FUNCIONES

Desde el menú de módulos se puede ingresar a cualquier módulo, presionando click en “Iniciar tema” sobre el módulo 10, se presenta la siguiente vista:

Composición de Funciones

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

```
1 ;Crear un programa en Racket que capture una letra y y devuelva
2 ;la letra siguiente. Nota: Use el código ASCII, use dos funciones,
3 ;una para comprobar el valor de la letra, y otra para aumentar
4 ;dicho valor
5 (define (ASCII letra)
6   (char->integer letra)
7 )
8
9 (define (aumentar letra)
10  (define valor 0)
11  (set! valor (ASCII letra))
12  (if(= valor 90)
13    65
14    (if(= valor 122)
15      97
16      (set! valor (+ valor 1))
17    )
18  )
19 )
20
21 (define (main)
```

Ir a C

output

Ejecutar

output

Comparar



24

Imagen 28: Módulo de composición de funciones

Donde se presenta un título correspondiente, una serie de botones interactivos con diferentes ejemplos y dos cajas de texto principales con el ejemplo actual en Racket y C.

La pestaña mostrará inicialmente el código en el lenguaje de programación Racket el cual se puede ejecutar, pero cuando el usuario presione en el botón “Ir a C”, entonces aparece el código de la siguiente manera:

Composición de Funciones

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

```
1 ;Crear un programa en Racket que capture una letra y y devuelva
2 ;la letra siguiente. Nota: Use el codigo ASCII, use dos funciones,
3 ;una para comprobar el valor de la letra, y otra para aumentar
4 ;dicho valor
5 (define (ASCII letra)
6   (char->integer letra)
7 )
8
9 (define (aumentar letra)
10  (define valor 0)
11  (set! valor (ASCII letra))
12  (if(= valor 90)
13    65
14    (if(= valor 122)
15      97
16      (set! valor (+ valor 1))
17    )
18  )
19 )
20
21 (define (main)
```

output

Ejecutar

Ir a C

```
1 /*Crear un programa en c que capture una letra y y devuelva la
2 letra siguiente. Nota: Use el codigo ASCII, use dos funciones,
3 una para comprobar el valor de la letra, y otra para aumentar
4 dicho valor*/
5 #include <stdio.h>
6 int ASCII(char letra){
7   return letra;
8 }
9
10 int aumentar(char letra){
11   int valor;
12   valor=ASCII(letra);
13   if(valor==90){
14     return 65;
15   }else if(valor==122){
16     return 97;
17   }else return valor+1;
18 }
19
20 int main(){
21   char letra;
```

output

Ejecutar

Comparar

Imagen 29: Módulo de composición de funciones completo

MÓDULO 11: PROCEDIMIENTOS

Desde el menú de módulos se puede ingresar a cualquier módulo, presionando click en “Iniciar tema” sobre el módulo 11, se presenta la siguiente vista:



Procedimientos

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

```
1 ;implemente un procedimiento que reciba un numero e imprima
2 ;si el numero es positivo o negativo, en caso de ser positivo
3 ;debe analizar si es par o impar.
4 (define (analisis n)
5   (if (> n 0)
6     (begin
7       (printf "\nEl numero ingresado es positivo.")
8       (if (= (remainder n 2) 0)
9         (printf "\nAdemas el numero es par.")
10        (printf "\nAdemas el numero es impar.")
11      )
12     )
13     (printf "\nEl numero ingresado es Negativo.")
14   )
15 )
16 )
17
18 (define (main)
19   (define numero 0)
20
21   (printf "Ingrese un numero para analizar: ")
```

Ir a C

output

Ejecutar

Comparar

output

Imagen 30: Módulo de procedimientos

Donde se presenta un título correspondiente, una serie de botones interactivos con diferentes ejemplos y dos cajas de texto principales con el ejemplo actual en Racket y C.

La pestaña mostrará inicialmente el código en el lenguaje de programación Racket el cual se puede ejecutar, pero cuando el usuario presione en el botón “Ir a C”, entonces aparece el código de la siguiente manera:

Procedimientos

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

```
1 ;implemente un procedimiento que reciba un numero e imprima
2 ;si el numero es positivo o negativo, en caso de ser positivo
3 ;debe analizar si es par o impar.
4 (define (analisis n)
5   (if (> n 0)
6     (begin
7       (printf "\nEl numero ingresado es positivo.")
8       (if (= (remainder n 2) 0)
9         (printf "\nAdemas el numero es par.")
10        (printf "\nAdemas el numero es impar.")
11      )
12     )
13     (printf "\nEl numero ingresado es Negativo.")
14   )
15 )
16 )
17
18 (define (main)
19   (define numero 0)
20
21   (printf "Ingrese un numero para analizar: ")
```

Ir a C

output

Ejecutar

Comparar

```
1 /*implemente un procedimiento que reciba un numero e imprima
2 si el numero es positivo o negativo, en caso de ser positivo
3 debe analizar si es par o impar.*/
4 #include <stdio.h>
5 void analisis(int n){
6   if(n>0){
7     printf("\nEl numero ingresado es positivo.");
8     if(n%2==0){
9       printf("\nAdemas el numero es par.");
10    }else {
11      printf("\nAdemas el numero es impar.");
12    }
13   }else printf("\nEl numero ingresado es Negativo.");
14 }
15
16 int main(){
17   int numero;
18   printf("Ingrese un numero para analizar: ");
19   scanf("%d", &numero);
20   analisis(numero);
21 }
```

output

Ejecutar



MÓDULO 12: MODO GRÁFICO

Desde el menú de módulos se puede ingresar a cualquier módulo, presionando click en “Iniciar tema” sobre el módulo 12 se presenta la siguiente vista:

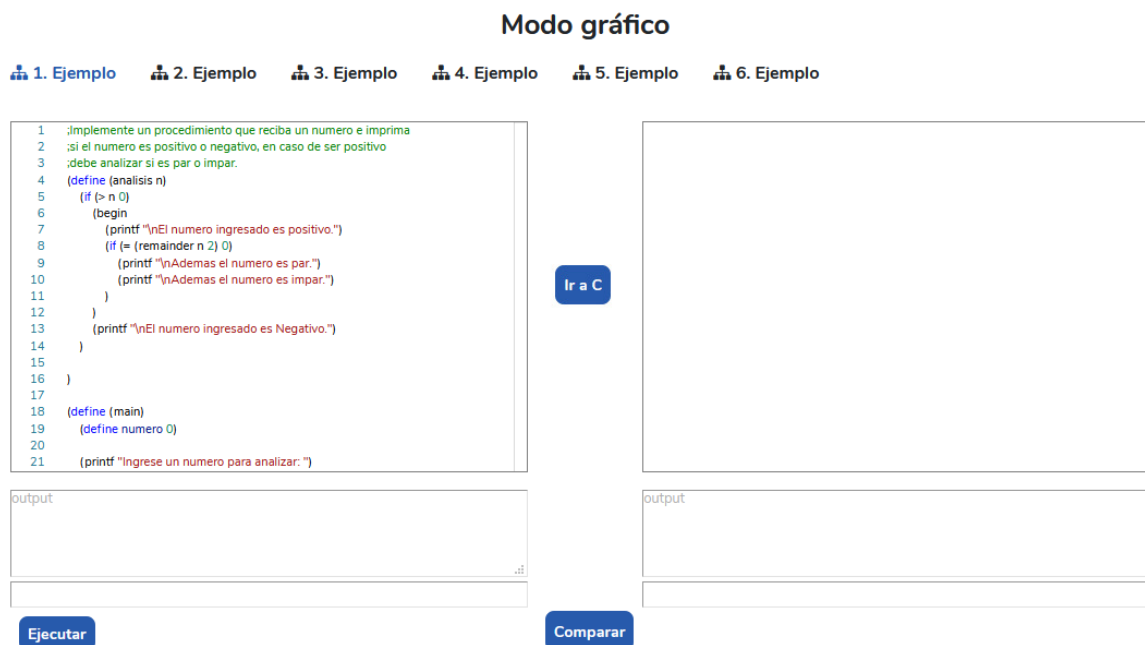


Imagen 32: Módulo de modo gráfico

Donde se presenta un título correspondiente, una serie de botones interactivos con diferentes ejemplos y dos cajas de texto principales con el ejemplo actual en Racket y C.

La pestaña mostrará inicialmente el código en el lenguaje de programación Racket el cual se puede ejecutar, pero cuando el usuario presione en el botón “Ir a C”, entonces aparece el código de la siguiente manera:



Modo gráfico

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

```
1 //Implemente un procedimiento que reciba un numero e imprima
2 //si el numero es positivo o negativo, en caso de ser positivo
3 //debe analizar si es par o impar.
4 (define (analisis n)
5   (if (> n 0)
6     (begin
7       (printf "\nEl numero ingresado es positivo.")
8       (if (= (remainder n 2) 0)
9         (printf "\nAdemas el numero es par.")
10        (printf "\nAdemas el numero es impar.")
11      )
12     )
13     (printf "\nEl numero ingresado es Negativo.")
14   )
15 )
16 )
17 (define (main)
18   (define numero 0)
19   (printf "Ingrese un numero para analizar: ")
20   (scanf "%d", &numero)
21   (analisis numero))
```

output

Ejecutar

Ir a C

```
1 /*Implemente un procedimiento que reciba un numero e imprima
2 /*si el numero es positivo o negativo, en caso de ser positivo
3 /*debe analizar si es par o impar.*/
4 #include <stdio.h>
5 void analisis(int n){
6   if(n>0){
7     printf("\nEl numero ingresado es positivo.");
8     if(n%2==0){
9       printf("\nAdemas el numero es par.");
10    }else {
11      printf("\nAdemas el numero es impar.");
12    }
13   }else printf("\nEl numero ingresado es Negativo.");
14 }
15
16 int main(){
17   int numero;
18   printf("Ingrese un numero para analizar: ");
19   scanf("%d", &numero);
20   analisis(numero);
21 }
```

output

Ejecutar

Comparar

Imagen 33: Módulo de modo gráfico completo

MÓDULOS DE PRÁCTICA: PRÁCTICA C

Desde el menú de módulos se puede ingresar a cualquier módulo, presionando click en “Iniciar tema” sobre el módulo 13 se presenta la siguiente vista:

Herramienta de práctica

1. Ejercicio 2. Ejercicio 3. Ejercicio

Un supermercado ha puesto en oferta la venta al por mayor de cierto producto, ofreciendo un descuento del 15% por la compra de más de 3 docenas y 10% en caso contrario. Además por la compra de más de 3 docenas se obsequia una unidad del producto por cada docena en exceso sobre 3. Diseñe un algoritmo que determine el monto de la compra, el monto del descuento, el monto a pagar y el número de unidades de obsequio por la compra de cierta cantidad de docenas del producto.

```
1 //type code here
```

output

Ir a Racket Ejecutar



28

Imagen 34: Módulo de práctica en C

Donde se presentan una serie de ejercicios con contexto y se pide pasar el enunciado a código. Se escribe el código y se recibe una salida en la parte inferior. En la derecha inferior se encuentran 2 botones: Ir a Racket el cual lleva al usuario al módulo de práctica de Racket o Ejecutar el cual ejecuta el código ingresado.

MÓDULOS DE PRÁCTICA: PRÁCTICA RACKET

Desde el menú de módulos se puede ingresar a cualquier módulo, presionando click en “Iniciar tema” sobre el módulo 14 se presenta la siguiente vista:

Herramienta de práctica

1. Ejercicio 2. Ejercicio 3. Ejercicio

Un supermercado ha puesto en oferta la venta al por mayor de cierto producto, ofreciendo un descuento del 15% por la compra de más de 3 docenas y 10% en caso contrario. Además por la compra de más de 3 docenas se obsequia una unidad del producto por cada docena en exceso sobre 3. Diseñe un algoritmo que determine el monto de la compra, el monto del descuento, el monto a pagar y el número de unidades de obsequio por la compra de cierta cantidad de docenas del producto.

1 .type code here

output

Ir a C

Ejecutar

Imagen 35: Módulo de práctica en Racket

Donde se presentan una serie de ejercicios con contexto y se pide pasar el enunciado a código. Se escribe el código y se recibe una salida en la parte inferior. En la derecha inferior se encuentran 2 botones: Ir a Cel el cual lleva al usuario al módulo de práctica de C o Ejecutar el cual ejecuta el código ingresado.