

UNIVERSIDAD TECNOLÓGICA DE PEREIRA - FACULTAD DE INGENIERÍAS
PROGRAMA DE INGENIERÍA EN SISTEMAS Y COMPUTACIÓN
ASIGNATURA: PROGRAMACIÓN – Manejo de vectores

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

FUNCIONES – PLANTILLAS

Tipo de dato a devolver	Nombre función	Parámetros
vector	make-vector	<i>cantidad</i> Construye un vector con la <i>cantidad</i> de posiciones especificadas, por defecto los llena de datos numéricos inicializándolos todos en cero(0)

Ejemplos:

(define <i>vec</i> (make-vector 6))	Si tecleo <i>vec</i> , devolverá #6(0) Indicando que es un vector de <i>seis(6)</i> posiciones, con todos sus datos en <i>cero(0)</i>
-------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------

Existe una variante a la función anterior que permite crear el vector y llenarlo con datos iguales del mismo tipo; así:

Tipo de dato a devolver	Nombre Función	Parámetros
vector	make-vector	<i>Cantidad dato</i> Construye un vector con la <i>cantidad</i> de posiciones especificadas, llenándolas todas con el dato entregado

Ejemplos:

(define <i>vec</i> (make-vector 6 #\W))	Si tecleo <i>vec</i> , devolverá #6 (#\W) Indicando que el vector <i>vec</i> es de <i>seis(6)</i> posiciones, y todos sus datos son el carácter W
-----------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------

Tipo de dato a devolver	Nombre función	Parámetros
vector	vector	dato1 dato2 dato3..... Construye un vector con los datos entregados

Ejemplos:

(define a (vector 0 "1" "Hola" "casa"))	Si tecleo <u>a</u> , devolverá #4(0 "1" "Hola" "casa") Indicando que es un vector de cuatro(4) posiciones, con los datos entregados
-----------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------

Si tengo incluido el "...PLT\teachpack\htdp\image.ss."

(define a (vector 0 "1" (rectangle 100 10 'outline 'red) "casa"))

Si tecleo a, devolverá:

#4(0 "1"  "casa")

Lo que me indica que puedo “almacenar” inclusive imágenes como elementos de un vector

(define bol (vector 0 "1" (circle 30 "solid" 'red) "c"))


Si tecleo bol, devolverá:

#4(0 "1"  "c")

(define *vec1* (vector 0 "1" "Hola" "amigos"))

(define *vec2* (vector *vec1* "UTP" (circle 30 "solid" 'red) "Pereira"))

Si tecleo vec2, devolverá:

#4(#4(0 "1" "Hola" "amigos") "UTP"  "Pereira")

Donde el primer elemento del vector *vec2*, es otro vector (*vec1*)

Tipo de dato a devolver	Nombre Función	Parámetros
Dato	vector-ref	<p><i>vector</i> <i>posición</i></p> <p>Devuelve del <i>vector</i> una copia del dato ubicado en la posición de la referencia, iniciando en cero(0)</p> <p>Parámetros</p> <p><i>vector</i>: vector de donde se sacara una copia de uno(1) de sus datos</p> <p>Posición: (integer) Posición iniciando en cero (0) de donde se sacara una copia del dato ubicado en esta posición</p>

Ejemplos:

(define *vec* (vector 12 "1" "Hola" "amigos"))

Si tecleo
(vector-ref *vec* 0)

Me devuelve el número **doce(12)** que es el dato ubicado en la posición cero(0) del vector *vec*.

Si tecleo
(vector-ref *vec* 3)

Me devuelve el string **“amigos”**, que es el dato ubicado en la posición tres(3) del vector *vec*.

Si tecleo
(substring (vector-ref *vec* 3) 1 2)

Me devuelve el string **“m”**, que es el “substring” que inicia en la posición **uno(1)** hasta la **dos(2, no inclusive)**, del string (**“amigos”**) ubicado en la posición tres(3) del vector *vec*.

Si tecleo
(+ 10 (vector-ref *vec* 0))

Me devuelve el resultado de la suma del número diez(10) con el número doce(12) que es el dato ubicado en la posición cero(0) del vector *vec*. Que serie en número **ventidos(22)**

Tipo de dato a devolver	Nombre función	Parámetros
Void	vector-set!	Vector posición nuevo-dato Coloca un nuevo-dato en la posición indicada sobre el vector Parámetros: vector Posición : Posición (integer), donde se colocará el nuevo dato dato : Dato a colocar en el vector en la posición indicada. No devuelve nada porque opera sobre el mismo vector

Ejemplos:

(define *vec* (vector 12 "1" "Hola" "amigos"))

vector <i>vec</i>	12	“1”	“Hola”	“amigos”
Posición	0	1	2	3

Si tecleo
(vector-set! *vec* 0 876)

Colocará en la posición cero(0) del vector *vec*, un nuevo dato que será para este caso el número **876**.

vector <i>vec</i>	876	“1”	“Hola”	“amigos”
Posición	0	1	2	3

Si tecleo

(vector-set! **vec 0** (remainder (vector-ref *vec* 0) 10))

Colocará en la posición cero(0) del vector *vec*, un nuevo dato que será para este caso el residuo de la división entera del dato que este en este momento en la posición cero(0) del vector *vec*, siendo entonces el número *seis*(6).

vector <i>vec</i>	6	“1”	“Hola”	“amigos”
Posición	0	1	2	3

Si tecleo

(vector-set! **vec 2** (+ (random 100) 1))

Colocará en la posición dos(2) del vector *vec*, un nuevo dato que será para este caso un número aleatorio entre 1 y 100, generado por la computadora

Ejercicios resueltos:

Dado el polinomio es $Y = X^2 + 1$, con un rango de: inicio = -500 y final = +100, con intervalos de +100, construir en dos(2) vectores la tabla de valores de x y $f(x)$..:

1. ANÁLISIS: Primero debo conocer la cantidad de elementos que van a tener los dos(2) vectores pedidos, esto sale del rango o dominio y los intervalos, entregados por el usuario; así:

Rango: Inicio = -500
 Fin = +100

Intervalos: +100

El número de datos, puede salir entonces de una función que los cuente a partir de entregarle el rango inicial, el final y los intervalos. Vgr:

Valores	-500	-400	-300	-200	-100	0	100
Cantidad	1	2	3	4	5	6	7

Sabiendo el número de datos de los vectores, creo los vectores **x** y **fx**. Con la función (make-vector y/o vector)

Creados los vectores, los lleno con los valores, evaluando el polinomio para cada valor de **x**; Vgr:

x	Evaluación	$Y=f(x)$
- 500	$(-500)^2 + 1 = 250,001$	250,001
- 400	$(-400)^2 + 1 = 160,001$	160,001
-300	$(-300)^2 + 1 = 90,001$	90,001
- 200	$(-200)^2 + 1 = 40,001$	40,001
- 100	$(-100)^2 + 1 = 10,001$	10,001
0	$(0)^2 + 1 = 1$	1
100	$(100)^2 + 1 = 10,001$	10,001

2. CONTRATO: Se hará un programa en DrScheme, tal que dado el polinomio $Y = X^2 + 1$, con un rango de: inicio = -500 y final = +100, con intervalos de +100, se construyan dos(2) vectores que contengan en el vector de las x los valores calculados de rango desde el inicio hasta el final y para $f(x)$, los valores generados al evaluar el polinomio para cada valor de x . Así:

x	$Y=f(x)$
- 500	250,001
- 400	160,001
-300	90,001
- 200	40,001
- 100	10,001
0	1
100	10,001

3. EJEMPLOS: Dado lo específico de lo pedido, lo que se debe entregar es lo mostrado en el contrato

4. DESARROLLO

; Autor: Mgs Ligia Stella Bustos Rios Oct-2006

;Función c_term: Devuelve el número de términos dado el rango y el intervalo

; Invoca a la función aux_c_term a la que se le entrega el rango, el intervalo y un contador
; iniciado en cero(0)

;Plantilla

```
; +-----+-----+-----+
; !tipo de dato  ! Nombre      ! parámetros      !
; ! a devolver  ! función      !                  !
; +-----+-----+-----+
; !Tipo: integer ! c_term      !inicio    final    intervalo    !
; ;              ;              ;Tipo: real, Nombre: inicio (numero donde inicia el rango) !
; ;              ;              ;Tipo: real, Nombre: final (numero donde finaliza el rango) !
; ;              ;              ;Tipo: real, Nombre: intervalo (numero del intervalo) !
; +-----+-----+-----+
```

```
(define (c_term inicio final intervalo)
  (aux_c_term inicio final intervalo 0 )
)
```

;Función aux_c_term: Función auxiliar que devuelve el numero de términos dado
; el rango y el intervalo.

;Plantilla

```
; +-----+-----+-----+
; !tipo de dato ! Nombre ! parámetros !
; ! a devolver ! función !
; +-----+-----+-----+
; !Tipo: integer ! aux_c_term inicio final intervalo contador !
; i i iTipo: real, Nombre: inicio (numero donde inicia el rango) !
; i i iTipo: real, Nombre: final (numero donde finaliza el rango) !
; i i iTipo: real, Nombre: intervalo (numero que indica el intervalo) !
; +-----+-----+-----+
```

```
(define (aux_c_term inicio final intervalo con )
  (if ( not (= inicio final ) )
      (aux_c_term (+ inicio intervalo) final intervalo (+ con 1) )
      (+ con 1)
  )
)
```