Lenguajes de Programación

Práctica 2 - Tipos de datos abstractos

Semestre 2017-2

Fecha de inicio: 17 de febrero de 2017

Fecha de término: 3 de marzo de 2017

Instrucciones

- Resolver de manera clara y ordenada los ejercicios en un archivo `practica2.rkt`.
- Para tener derecho a calificación, la práctica debe ejecutarse sin errores ni advertencias. No está permitido utilizar primitivas de Racket que resuelvan directamente los ejercicios.
- La entrega es en equipos de máximo 3 integrantes. Seguir los lineamientos especificados en: `http://lenguajesfc.com/lineamientos.html`.
- Los puntos extra son individuales y sólo se puede escoger uno. El punto extra teórico debe de realizarse **a computadora y entregarse impreso en hojas recicladas a más tardar el 3 de marzo durante la sesión de laboratorio**. El punto extra de programación se envía por correo electrónico en un archivo `<no.cuenta>_P02.rkt` enviar con el asunto `[LDP-EXP2]`. Incluir el nombre completo del autor en el cuerpo del correo.

Ejercicios

1. **Figuras tridimensionales**

   Definir un tipo de dato abstracto `Figura3D` que sea utilizado para trabajar con figuras tridimensionales.

   El tipo de dato abstracto debe contener:

   - Un constructor `(tetraedro a)` donde `a` es un número real y representa la arista del poliedro.
   - Un constructor `(prisma a P B h)` donde `a, P, B, h` son números reales y representa el arista lateral, el perímetro de la sección recta, el área de la base y la altura de un prisma cualquiera respectivamente.
   - Un constructor `(paralelepipedo a b c)` donde `a, b, c` son números reales y representan el largo, el ancho y la altura de un paralelepípedo rectángulo respectivamente.
   - Un constructor `(piramide B h)` donde `B, h` son números reales y representan el área de la base y la altura de una pirámide cualquiera respectivamente cualquiera.
   - Un constructor `(cilindro g C B h)` donde `g, C, B, h` son números reales y representan la generatriz, el perímetro de la sección recta, el área de la base y la altura de un cilindro circular recto respectivamente.
   - Un constructor `(esfera r)` donde `r` es un número real y representa el radio de una esfera.

- Un constructor `(cono g h r)` donde `g, h, r` son números reales y representan la generatriz, la altura y el radio de la base de un cono circular recto respectivamente.

Una vez definido el tipo de dato, se deben de definir las siguientes funciones:

a) Una función `(area-lateral fig)` que dada una figura regrese su área lateral.

b) Una función `(area-total fig)` que dada una figura regrese su área total.

c) Una función `(volumen fig)` que dada una figura regrese su volumen.

Algunos ejemplos de uso del tipo de dato abstracto:

```
> (define figura (paralelepipedo 3 2 4))
> figura
(paralelepipedo 3 2 4)
> (area-lateral figura)
40
> (area-total figura)
52
> (volumen figura)
24
```

2. **Árboles binarios ordenados**

Definir un tipo de dato abstracto `ABO` para trabajar con árboles binarios cuyos elementos están ordenados. Es decir, de acuerdo a un predicado de comparación, los elementos menores a la raíz estarán a la izquierda y los mayores o iguales estarán a la derecha.

El tipo de dato abstracto debe contener:

- Un constructor `(void)` que define al árbol vacío.
- Un constructor `(nodo a t1 t2 c)` que define al nodo con un árbol izquierdo, un parámetro `a` que representa el elemento que almacenará el nodo y puede ser de cualquier tipo, definir un predicado `any?` que permita hacer esto, un árbol derecho y una función de comparación.

Una vez definido el tipo de dato, definir las siguientes funciones:

a) Una función `(agrega-abo a e)` que agrega al elemento `e` en el árbol `a` de acuerdo al criterio de ordenamiento.

b) Una función `(numero-elems a)` que regresa el número de elementos del árbol

c) Una función `(altura a)` que regresa la altura del árbol

d) Una función `(inorder a)` que regresa una lista con la información de cada hoja del árbol en inorder.

e) Una función `(preorder a)` que regresa una lista con la información de cada hoja del árbol en preorder.

f) Una función `(postorder a)` que regresa una lista con la información de cada hoja del árbol en postorder.

Algunos ejemplos de uso del tipo de dato abstracto:

```
> (define tree (nodo 27 (void) (void) >))
tree
(nodo 27 (void) (void) >)
> (define tree2 (agrega-abo tree 2))
> tree2
(nodo 27 (nodo 2 (void) (void)) (void))
> (define tree3 (agrega-abo tree2 57) >)
> tree3
(nodo 27 (nodo 2 (void) (void) >) (nodo 57 (void) (void) >) >)
> (numero-elems tree3)
3
> (altura tree3)
1
> (inorder a)
'(2 27 57)
> (preorder a)
'(27 2 57)
> (postorder a)
'(2 57 27)
```

3. **Árboles binarios**

Definir un tipo de dato abstracto `AB` para trabajar con árboles binarios que sólo tienen información en las hojas.

El tipo de dato abstracto debe contener:

- Un constructor `(hoja a)` que define una hoja del árbol. El parámetro `a` representa el elemento que almacenará la hoja y puede ser de cualquier tipo, definir un predicado `any?` que permita hacer esto.
- Un constructor `(mkt t1 t2)` que define al nodo sin información que tiene como hijos al árbol izquierdo `t1` y al árbol derecho `t2`.

Una vez definido el tipo de dato, definir las siguientes funciones:

a) Una función `(agrega-ab a e)` que agrega un elemento `e` al árbol `a`.

b) Una función `(numero-hojas a)` que regresa el número de hojas de un árbol.

c) Una función `(nodos-interos a)` que regresa el número de nodos internos de un árbol.

d) Una función `(hojas a)` que regresa una lista con la información de cada hoja del árbol en inorder.

e) Una función `(map-a f a)` que aplica la función `f` a cada elemento de un árbol.

f) Una función `(profunfidad a)` que regresa la profundidad de un árbol.

Algunos ejemplos de uso del tipo de dato abstracto:

```
> (define tree (hoja 'a))
tree
(hoja 'a)
> (define tree2 (agrega-ab tree 'b))
> tree2
(mkt (hoja 'b) (hoja 'a))
> (define tree3 (agrega-ab tree2 'c))
> tree3
(mkt (hoja 'c) (mkt (hoja 'b) (hoja 'a)))
> (numero-hojas tree3)
3
> (nodos-internos tree3)
2
> (hojas tree3)
'(c b a)
> (map-a symbol->string tree3)
(mkt (hoja ``c'') (mkt (hoja ``b'') (hoja ``a'')))
> (profundidad tree3)
1
```

4. **Arreglos**

   Definir un tipo de dato abstracto `Arreglo` para trabajar con arreglos.

   El tipo de dato abstracto debe contener:

   - Un constructor `(arreglo tipo dim elems)` que permite definir un arreglo. El parámetro `tipo` es un predicado para identificar el tipo de los elementos, el parámetro `dim` sirve para definir el tamaño del arreglo y `elems` es una lista de tamaño `dim` y con elementos de tipo `tipo`.
   - Un constructor `(agrega-a e a i)` para representar la operación de agregar un elemento `e` en el Arreglo `a` en la posición `i`.
   - Un constructor `(obten-a a i)` para representar la operación de obtener el elemento en la posición `i` del Arreglo `a`.

   Una vez definidos los tipos de datos, se debe de definir una función `(calc-a arreglo)` que evalúe expresiones del tipo `Arreglo`.

   Algunos ejemplos de uso del tipo de dato abstracto:

```
> (define a (arreglo number? 5 '(1 2 3 4 5)))
> (calc-a a)
(arreglo number? 5 '(1 2 3 4 5))
> (define b (arreglo boolean? 4 '(1 2 3 4)))
> (calc-a b)
error: Los elementos no son del tipo especificado.
> (define c (arreglo boolean? 3 '(#t #t #t #f)))
> (calc-a c)
error: Dimensión inválida
```

```
> (calc-a (agrega-a 2 a 4))
(arreglo number? 5 '(1 2 3 4 2))
> (calc-a (agrega-a 2 a 7))
error: Índice inválido
> (calc-a (agrega-a #t a 7))
error: Los elementos no son del tipo especificado.
> (calc-a (obten-a 2 a))
3
> (calc-a (obten-a -1 a))
error: Índice inválido
```

Puntos extra

- (1.5 pts.) Investigar quién es Bárbara Liskov e indicar sus contribuciones al diseño de lenguajes de programación, principalmente relacionados con la abstracción de datos. Escribir un ensayo de mínimo dos cuartillas donde se describa quién es Barbara Liskov, qué es la abstracción, qué utilidad tiene la abstracción en las ciencias de la computación, qué utilidad tiene la abstracción en los lenguajes de programación y ejemplificar con aplicaciones. El ensayo debe contener introducción, desarrollo, conclusiones, bibliografía y debe estar debidamente citado.

- (1.5 pts.) Invetigar qué es y definir un tipo de dato abstracto `MonticuloMinimo` de números enteros y definir las funciones: `agregar` (debe colocar el elemento en la posición correspondiente de manera que los nodos en un nivel `n` sean mayores a los del nivel `n+1`, de esta forma el elemento mínimo estará en la raíz), `eliminar` (debe regresar el elemento de la raíz y acomodar los demás elementos del árbol de manera que el elemento mínimo quede en la raíz).