**Ejercicio 1:** Dada la siguiente declaración correspondiente a un árbol binario:

Type

Tarbol = ^Tnodo;

Tnodo = Record

Clave: Tclave;

Izq Der: Tarbol;

end;

Implementar un algoritmo que determine si un árbol dado es simétrico respecto al eje vertical que pasa por su raíz. La siguiente figura muestra un árbol de tales características.

Imagen que contiene reloj, objeto, colgando, foto

Descripción generada automáticamente

Program Ejercicio1;

uses crt, SysUtils, uPila;

Type

Tarbol = ^Tnodo;

Tnodo = Record

Clave: Tclave;

Izq Der: Tarbol;

end;

procedure GuardarIzq (Ab: Tarbol; const Raiz: Tclave; var Pi: Tpila);

var Caso: Byte;

begin

{ Recorro hasta el Final controlando de Guardar solo la rama que me interesa }

if (Ab <> nil) and (Ab^.clave <= Raiz) then

begin

{ Compruebo los posibles casos }

If (Ab^.Izq = nil) and (Ab^.Der = nil) then Caso:= 0

Else if (Ab^.Izq = nil) and (Ab^.Der <> nil) then Caso:= 1

Else if (Ab^.Izq <> nil) and (Ab^.Der = nil) then Caso:= 2

Else if (Ab^.Izq <> nil) and (Ab^.Der <> nil) then Caso:= 3;

uPila.Push (Pi, Caso);

{ Sigo recorriendo }

GuardarIzq (Ab^.Izq);

GuardarIzq (Ab^.Der);

end;

end;

procedure GuardarDer (Ab: Tarbol; const Raiz: Tclave; var Pd: Tpila);

var Caso: Byte;

begin

{ Recorro hasta el Final controlando de Guardar solo la rama que me interesa }

if (Ab <> nil) and (Ab^.clave >= Raiz) then

begin

{ Compruebo los posibles casos }

If (Ab^.Izq = nil) and (Ab^.Der = nil) then Caso:= 0

Else if (Ab^.Izq = nil) and (Ab^.Der <> nil) then Caso:= 1

Else if (Ab^.Izq <> nil) and (Ab^.Der = nil) then Caso:= 2

Else if (Ab^.Izq <> nil) and (Ab^.Der <> nil) then Caso:= 3;

uPila.Push (Pd, Caso);

{ Sigo recorriendo }

GuardarDer (Ab^.Izq);

GuardarDer (Ab^.Der);

end;

end;

function Mirror (Pi, Pd: Tpila): boolean;

var a, b: Byte;

begin

Simetrico:= False;

{ Compruebo que no sea un chasco el que me hayan llamado }

If Not(uPila.isEmpty (Pi)) and Not(uPila.isEmpty (Pd)) then

begin

while Not(uPila.isEmpty (Pi)) and Not(uPila.isEmpty (Pd)) and (Simetrico) do

begin

uPila.Pop (Pi, a);

uPila.Pop (Pd, b);

{ Momento de la Decisión }

{ Compruebo si en cada nodo se encontró la misma situación }

if (a = b) then Simetrico:= true

else Simetrico:= False;

end;

end;

{ último control por si quedó en true, pero todavía había elementos en alguna pila }

if Not(uPila.isEmpty (Pi)) or Not(uPila.isEmpty (Pd)) then

Simetrico:= False;

end;

Var

Ab: Tarbol;

Raiz: Tclave;

Pi, Pd: Tpila;

Espejo: boolean;

Begin

//Suponemos que ya tenemos un arbol cargado.

Mirror:= false;

if (Ab <> nil) then

begin

{ Inicializamos los contenedores }

u{ Inicializamos los contenedores }

uPila.Init (Pi);

uPila.Init (Pd);

{ Guardamos la Situación de Cada nodo para compararlas luego }

Raiz:= Ab^.Clave;

GuardarIzq (Ab, Raiz, Pi);

GuardarDer (Ab, Raiz, Pd);

{ Comparamos las Situaciones de Cada nodo recorrido }

Espejo:= Mirror (Pi, Pd);

if Espejo then writeln ('Simetrico')

else writeln ('No hay simetria');

end;

End; **REGULAR**

**Ejercicio 2:** ¿Por qué es importante el estudio de las estructuras de datos?

Es importante porque permite desarrollar estrategias para un Almacenamiento Grande de Datos, de manera que se puedan mejorar las técnicas en el Espacio que se utiliza (pudiendo optar por estructuras dinámicas en lugar de estáticas, por ejemplo), y en la Velocidad de Búsqueda de los Datos. **REGULAR**

**Ejercicio 3:** Si tuviera que elegir entre un árbol binario de búsqueda o una tabla de dispersión ¿cuáles serían las características que determinaría la elección de una u otra estructura? Justifique.

Un Árbol Binario de Búsqueda es una elección correcta **cuando se quiere mantener un orden en el almacenamiento de los datos**, con la intención de que el orden de búsqueda se vaya dividiendo a medida que se avanza. Pero puede que el **tiempo de búsqueda se degrade si el árbol se sesga** hacia un lado.

Una Tabla de Dispersión es una elección correcta cuando se quiere almacenar grandes cantidades de datos, o simplemente uno, para luego **buscar de manera directa**. Pero con ésta **se pierde el orden de los datos. BIEN**

**Ejercicio 4:** ¿Bajo qué circunstancias o contexto elegiría un método de ordenamiento como el Quick Sort o el Merge Sort? Explique el funcionamiento de cada uno.

Cuando se tiene una Lista desordenada (vale también para un Vector).

Merge Sort usa espacio adicional al llamarse de manera recursiva para ir dividiéndose hasta no poder más. Separa la Lista en dos y continúa dividiendo ambas partes. Una vez divididos en su máxima posibilidad, se considera que el elemento ya está ordenado, y se vuelve a los llamados anteriores, comparando los elementos y ordenándolos a su paso.

Quick Sort se usa un pivote y dos índices, o punteros para comparar con el pivote.

El pivote se sitúa en la mitad de la Lista, y se procede a comparar el puntero del lado izquierdo con el pivote hasta hallar un valor mayor a éste, luego se procede de la misma manera con el puntero del lado derecho, hasta hallar un valor menor a éste. Cuando se obtiene uno mayor del lado izquierdo, y otro menor del lado derecho, se intercambian los contenidos, y se continúa esta iteración hasta que los punteros se crucen, luego se va sub dividiendo en ambas partes, como el método anterior. Hasta estar ordenado. **REGULAR**

**Ejercicio 5:** ¿Qué situación ocurre con los árboles binarios de búsqueda que dan lugar a los AVL? ¿Cuándo hay que equilibrar y qué operaciones debe realizar?

Los Árbol Binarios de Búsqueda pueden sesgarse hacia un lado, lo que termina desechando la ventaja que da el usarlo, por que tiende a ser una Lista ordenada.

Para salvar esto surgen los AVL, que controlan al momento de la inserción la estabilidad del árbol, impidiendo así que se sesgue hacia cualquier lado.

Se debe equilibrar al momento de la inserción para no tener que estar corroborando una vez ingresados el equilibrio, cosa que llevaría más tiempo.

Se hace con las operaciones de rotación, las cuales son: Rotación Simple a Izquierda, Rotación Simple a Derecha, Rotación Doble a Izquierda y Rotación Doble a Derecha.

Se debe controlar el equilibrio cuando se eliminan los elementos también, lógicamente.**REGULAR+**

**NOTA: el puntaje obtenido permite promocionar la materia NOTA (7).**