Árboles y su implementación en Object Pascal

Autores: MsC. Marcos Antonio León Fonseca.
MsC. Noralys Muñiz Maldonado.
MsC. Juan Antonio Fonseca Hernández

Resumen:

En este artículo el autor describe como se implementan los árboles en el lenguaje de programación Object Pascal.

Introducción:

Una de las características más importantes de Object Pascal es su capacidad para admitir diferentes tipos de datos. Entre ellos están los datos de tipo simple, los de tipo estructurado y los de tipo puntero.

Los datos de tipo simple son elementos individuales (números, caracteres, etc) que se asocian a identificadores únicos.

Existen varios tipos de datos simples que abarcan los cuatro tipos estándar (entero, real, de caracteres, boléanos) y a los tipos simples definidos por el programador (enumerados y subrango).

Los datos de tipo estructurado se componen de múltiples elementos relacionados entre sí que se asocian a un único identificador.

Los elementos individuales, dentro del grupo, pueden asociarse también a identificadores independientes. Hay cuatro tipos de datos estructurados: arrays, registros, ficheros y conjuntos.

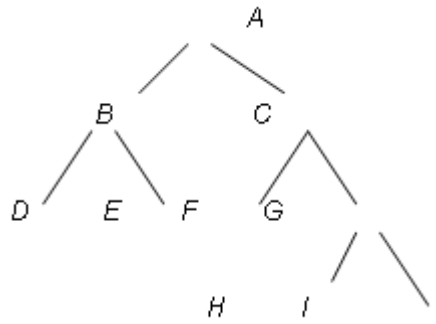
Los datos de tipo puntero se emplean para construir tipos dinámicos de datos estructurados tales como las listas enlazadas y los árboles.

Este trabajo tiene como propósito describir la implementación de estos últimos en Object Pascal.

Desarrollo:

Los datos presentan frecuentemente una relación jerárquica entre sus elementos; tal es el caso, por ejemplo, de árboles genealógicos, tablas de contenidos, etc. La estructura de datos que se emplea para reflejar esta relación recibe el nombre de grafo en árbol o simplemente árbol.

Un árbol se representa generalmente en forma de diagrama:

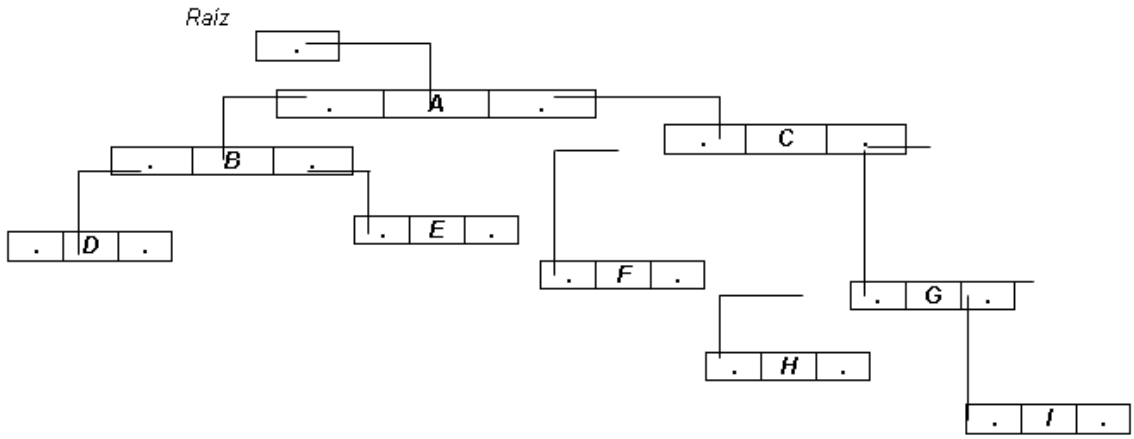


En este caso se dice que el árbol tiene nueve nodos, representados por las letras de la *A* a la *I*, la raíz del árbol es el nodo *A* que es el que está en la parte superior del diagrama; una línea hacia abajo y a la izquierda de un nodo señala un sucesor izquierdo del mismo, y una línea hacia abajo y a la derecha señala un sucesor derecho (*B* es un sucesor izquierdo de *A* y *C* un sucesor derecho), el subárbol izquierdo de la raíz *A* consiste en los nodos *B*, *D* y *E*, y el subárbol derecho en los nodos *C*, *F*, *G*, *H*, *I*.

Los nodos sin sucesor se llaman nodos terminales.

Cada nodo del árbol contiene un subárbol izquierdo y uno derecho. Si el nodo es terminal, ambos subárboles están vacíos.

La forma más usual de representar un árbol en la memoria de la computadora es la llamada representación enlazada:

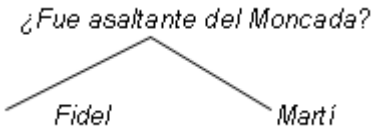


En esta representación, cada nodo consta de tres campos: uno para almacenar la información que contiene el nodo y los otros dos para apuntar a los subárboles izquierdo y derecho del mismo, respectivamente.

Para ilustrar cómo se implementan los árboles en Object Pascal, se analiza a continuación como se podría crear un programa que "aprenda" sobre personajes históricos en la medida en que se trabaje con él.

El programa constará con una base de "conocimientos", que le permitirá a partir de ciertos datos "adivinar" un personaje histórico, en el caso de que el personaje no esté registrado en su base de conocimientos, el programa lo incorporará a esta junto con una información sobre el mismo que lo distinga de los demás.

Supongamos que la base inicial de conocimientos está limitada a una pregunta con dos posibles respuestas, una afirmativa y otra negativa. La pregunta pudiera ser: ¿Fue asaltante del Moncada? Si la respuesta es afirmativa, se podría pensar en Fidel, mientras que si es negativa, se podría pensar en Martí.



Para implementar el árbol, se utilizan variables de tipo puntero y variables referenciadas.

La variable de tipo puntero se llamará *Puntero* y apuntará a la variable referenciada *Arbol* que será un registro que estará formado por tres campos: *Nodo* para almacenar la información sobre el hecho histórico, *Ramazquierda* que apuntará al personaje histórico que cumple con las exigencias de la información y *RamaDerecha* que utilizaremos para apuntar al personaje que no cumple los requisitos de la información.

```
Type

Puntero = ^ Arbol;

Arbol = Record

Nodo : String;

RamaIzquierda : Puntero;

RamaDerecha : Puntero;

End;
```

La variable referenciada se crea dinámicamente utilizando el procedimiento estándar *New*.

El árbol se construirá creando cada nodo y enlazándolo, después que se creen los subárboles, a su sucesor izquierdo y derecho respectivamente. Para ello se declaran las variables de tipo *Puntero*:

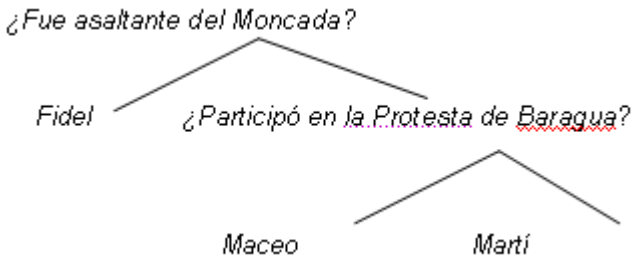
```
Var

Elemento, Predecesor : Puntero;
```

Elemento para el nodo que se crea y *Predecesor* para poder actualizar los punteros, de ese elemento que se creó, después que se creen los subárboles ya que dichos punteros, inicialmente, apuntarán a *Nil* para indicar que están vacío.

```
New (Elemento);

Elemento.Nodo := '¿Fue asaltante del Moncada?';

Elemento.RamaIzquierda := Nil;

ElementoRamaDerecha := Nil;

Predecesor := Elemento;

New (Elemento);

Elemento.Nodo := 'Fidel';

Elemento.RamaIzquierda := Nil;

ElementoRamaDerecha := Nil;

Predecesor.RamaIzquierda := Elemento;

New (Elemento);

Elemento.Nodo := 'Martí';

Elemento.RamaIzquierda := Nil;

ElementoRamaDerecha := Nil;

Predecesor.RamaDerecha := Elemento;
```

Si al correr el programa el usuario pensó por ejemplo en *Maceo*, se recorrerá el subárbol derecho al responderse que no fue asaltante al Moncada y se preguntará entonces si es *Martí*, como la respuesta es de nuevo negativa, le pedirá al usuario que introduzca el nombre del personaje y una información que lo distinga de Martí.

Suponga que la información es: *¿Participó en la Protesta de Baragua?* Con esta información se debe modificar la información de la rama derecha del árbol.



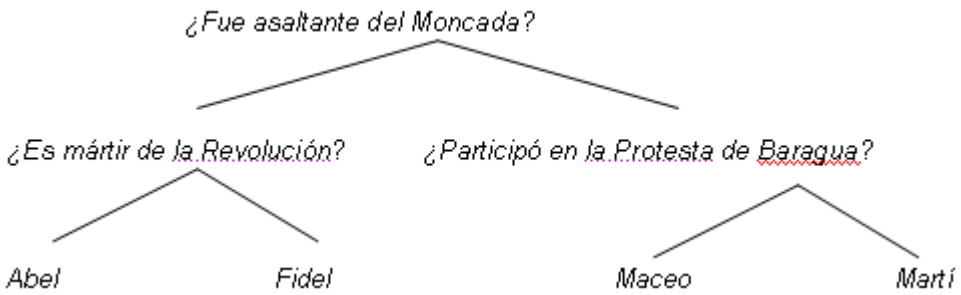
En este caso se creará primero el personaje y después el nodo que contendrá la información relacionada con el mismo por lo que se utilizará otra variable de tipo *Puntero* que se llamará *Sucesor* y que se empleará para enlazar el nuevo nodo con este personaje.

```
New (Elemento);

Elemento.Nodo := 'Maceo;

Elemento.RamaIzquierda := Nil;

ElementoRamaDerecha := Nil;

Sucesor := Elemento;

New (Elemento);

Elemento.Nodo := '¿Participó en la Protesta de Baragua?';

Elemento.RamaIzquierda := Sucesor;

Elemento.RamaDerecha:= Predecesor.RamaDerecha;

Predecesor.RamaDerecha := Elemento;
```

Note como con las instrucciones *Elemento.RamaIzquierda := Sucesor; y Elemento.RamaDerecha := Predecesor.RamaDerecha*; se enlaza el nuevo nodo con los personajes *Maceo* y *Martí*, respectivamente.

Con la instrucción *Predecesor.RamaDerecha := Elemento;* se actualiza la rama derecha del nodo que antecede al que se está insertando.

Si el usuario pensó por ejemplo en Abel y la pregunta es: ¿Es mártir de la Revolución?, el árbol sería entonces:



Por lo que se deberá modificar la información de la rama izquierda.

New (Elemento);

Elemento.Nodo := 'Abel;

Elemento.RamaIzquierda := Nil;

ElementoRamaDerecha := Nil;

Sucesor := Elemento;

New (Elemento);

Elemento.Nodo := '¿Es mártir de la Revolución?';

Elemento.RamaIzquierda := Sucesor;

Elemento.RamaDerecha := Predecesor.RamaIzquierda

Predecesor.RamaIzquierda := Elemento;

En la medida en que se corra el programa el árbol crecerá por las diferentes ramas en función de la información entrada por el usuario, se enriquecerá la base de conocimientos y el programa habrá "aprendido".

Conclusiones:

El conocimiento de la implementación de los árboles en Object Pascal, permite la representación de estructuras de datos en la memoria de la computadora cuando la relación entre los elementos que componen dicha estructura es una relación jerárquica.

Bibliografía:

De la Torre García, Ernesto. Iniciación en Logo.-- Ciudad de la Habana : Editorial Pueblo y Educación, 1988

Gottfried, Byron S. Programación en Pascal.-- Ciudad de la Habana : Editorial Pueblo y Educación, 1989

Lipschutz, Seymour. Estructura de datos.-- Ciudad de la Habana : Edición Revolucionaria, 1989