Estructuras de Datos no Lineales

Práctica 3

Problemas de árboles binarios II

PASOS A SEGUIR

1. Implementar el TAD *árbol binario* con las tres representaciones estudiadas, vectorial, enlazada y vector de posiciones relativas.
2. Escribir módulos que contengan las implementaciones de los subprogramas demandados en cada problema.
3. Para cada uno de los problemas escribir un programa de prueba, independiente de la representación del TAD elegida, donde se realicen las llamadas a los subprogramas, comprobando el resultado de salida para una batería suficientemente amplia de casos de prueba.

PROBLEMAS

1. Dos árboles binarios son similares cuando tienen idéntica estructura de ramificación, es decir, ambos son vacíos, o en caso contrario, tienen subárboles izquierdo y derecho similares. Implementa un subprograma que determine si dos árboles binarios son similares.

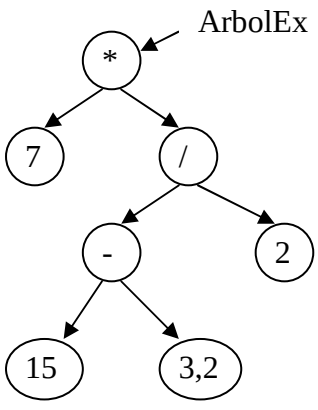
2. Para un árbol binario *B*, podemos construir el árbol binario reflejado B^R cambiando los subárboles izquierdo y derecho en cada nodo. Implementa un subprograma que devuelva el árbol binario reflejado de uno dado.

3. El TAD *árbol binario* puede albergar expresiones matemáticas mediante un árbol de expresión. Dentro del árbol binario los nodos hojas contendrán los operandos, y el resto de los nodos los operadores.

a) Define el tipo de los elementos del árbol para que los nodos puedan almacenar operadores y operandos.

b) Implementa una función que tome un árbol binario de expresión (aritmética) y devuelva el resultado de la misma. Por simplificar el problema se puede asumir que el árbol representa una expresión correcta. Los operadores binarios posibles en la expresión aritmética serán suma, resta, multiplicación y división.

Ejemplo: El siguiente árbol binario representa la expresión infija **7 * (15 – 3,2) / 2**.

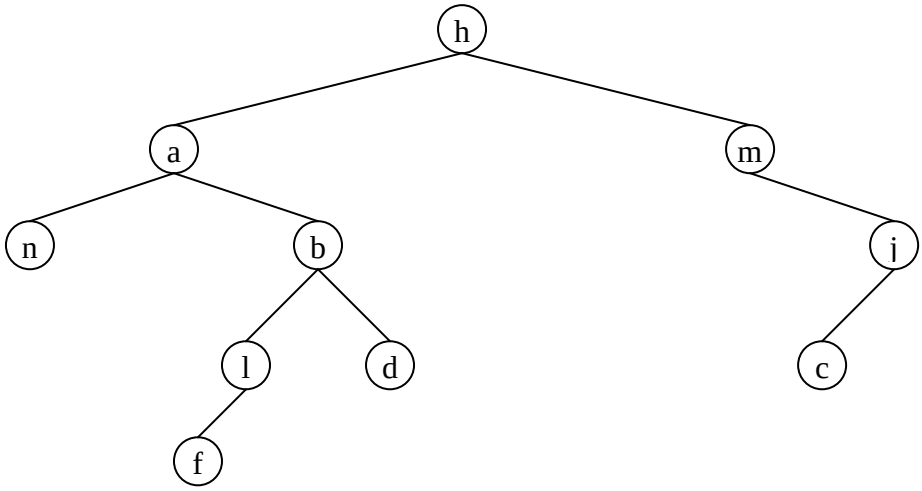


Nota: En el programa de prueba podemos usar la función *leerAbin()* de la práctica 1 para introducir por teclado el árbol de expresión a evaluar. Para ello será necesario sobrecargar los operadores, utilizados en dicha función, >>, << y != para el tipo de los elementos del árbol.

4. Una posible representación del TAD *Árbol Binario* consiste en almacenar los elementos del árbol en un vector cuyo tamaño depende de la altura máxima que pueda llegar a alcanzar el árbol. Cada nodo del árbol se corresponde con una única posición del vector, la cual viene determinada por el recorrido en inorden del árbol. Es decir, en el vector aparecen primero los nodos del subárbol izquierdo en inorden, luego la raíz y a continuación los nodos del subárbol derecho también en inorden. Por ejemplo, el árbol de la figura se representa como el vector

(-, -, -, n, -, -, -, a, f, l, -, b, -, d, -, h, -, -, -, -, -, -, -, m, -, c, -, j, -, -, -),

donde '-' representa una posición vacía.



Los hijos izquierdo y derecho de un nodo *n* corresponden, respectivamente, a las posiciones $n-(N+1)/2^{p+2}$ y $n+(N+1)/2^{p+2}$, donde *p* es la profundidad de *n* y *N* es el número máximo de nodos del árbol, es decir, el tamaño del vector. Por tanto, el padre de un nodo *n* se puede calcular de la siguiente forma:

$$Padre(n)=\begin{cases} n+(N+1)/2^{p+1} & \text{si } n \text{ es hijo izquierdo} \\ n-(N+1)/2^{p+1} & \text{si } n \text{ es hijo derecho} \end{cases}$$

Un nodo *n* es hijo izquierdo de su padre si $n \bmod \left(\frac{N+1}{2^{p-1}}\right)=\frac{N+1}{2^{p+1}}-1$

a) Define la clase genérica *Abin<T>* para esta representación.
b) Implementa una función miembro privada que calcule la profundidad de un nodo de un árbol binario representado de la forma descrita.
c) Para esta representación implementa, al menos, el constructor de árboles vacíos y las operaciones *crearRaizB()*, *insertarHijoIzqdoB()* y *padreB()*, según la especificación del TAD *Árbol Binario* vista en clase.