PRÁCTICA 5

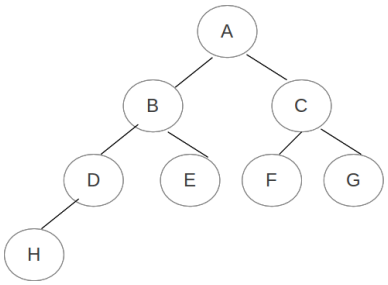
TIPOS ABSTRACTOS DE DATOS: ÁRBOLES

IMPORTANTE: para cada función pedida en los TADs analizar, diseñar y programar en todos los casos:

- **Si la operación se puede llevar a cabo, determinar qué retorna la función y el estado de las distintas variables y/o parámetros.**
- **Casos en que la operación no se puede llevar a cabo: determinar precondiciones y postcondiciones, decidir qué realiza la función en estos casos, programar y documentar. Las funciones deben funcionar correctamente para todos los casos.**
- **Analizar y justificar los parámetros de las funciones. Detectar para cada parámetro si está pasado por valor o por referencia y justificar el por qué.**

1. ÁRBOLES BINARIOS

1) Dado el siguiente árbol, desarrollar:



a. Enumerar las hojas.

b. ¿Es un árbol binario completo? Si no lo es, explicar por qué.

c. Calcular la altura del árbol (cantidad de niveles).

d. Para cada uno de los nodos, calcular la altura de su subárbol izquierdo y de su subárbol derecho.

2) Un árbol binario se puede utilizar en problemas de codificación, como la codificación y decodificación de mensajes transmitidos en Morse, un esquema en el que los caracteres se representan como secuencias de puntos y rayas, como se muestra en la siguiente tabla:

A	·–	J	·–––	S	···	2	··–––
B	–···	K	–·–	T	–	3	···–––
C	–·–·	L	·–··	U	··–	4	····–
D	–··	M	––	V	···–	5	·····
E	·	N	–·	W	·––	6	–····
F	··–·	O	–––	X	–··–	7	––···
G	––·	P	·––·	Y	–·––	8	–––··
H	····	Q	––·–	Z	––··	9	––––·
I	··	R	·–·	1	·––––	0	–––––

Representar este código utilizando árboles binarios: en este caso, los nodos del árbol binario se usan para representar los caracteres y cada arco de un nodo a sus hijos se etiqueta con un punto o una raya, dependiendo de que lleve a un hijo izquierdo o hijo derecho, respectivamente. La raíz del árbol no tiene ningún caracter, y como se dijo anteriormente, cada arco hacia un hijo izquierdo se etiqueta con el punto y cada arco hacia un hijo derecho con una raya. Represente el arbol de código Morse completo. Recorra el árbol para escribir la frase "llueve en Bariloche" en Morse.

2. ÁRBOLES BINARIOS DE EXPRESIÓN

Leer el apunte: "*Estructuras de Datos y Algoritmos: Aplicaciones de los Árboles Binarios*" (03_Arboles-de-Expresion.pdf en el Blog), de la Universidad Nacional de San Luis. Luego, resolver:

3) Dibujar la representación en árbol binario de cada una de las siguientes expresiones. Consideraciones: las expresiones serían el resultado del recorrido "enOrden". Los paréntesis definen qué abarca cada operación binaria.

1. $x+(y*(z/(a+b)))$
2. $(z+x)/(y-(3*y))$
3. $a*((5*z)+(5-z))$

4) Si se utilizan los algoritmos de recorrido, donde la visita a cada nodo es imprimir su dato: deducir (dibujar) los elementos de los árboles binarios siguientes en cada uno de los tres recorridos.

	Arbol A	Arbol B	Arbol C
PreOrden	+ * a b / c d	+ + a b + c d	/ + - a b + x y * b * c d
EnOrden	a * b + c / d	a + b + c + d	a – b + x + y / b * c * d
PostOrden	a b * c d / +	a b + c d + +	a b - x y + + b c d * * /

3. ÁRBOL BINARIO DE BÚSQUEDA

5) Construir (dibujar) un árbol binario de búsqueda con los elementos: 30, 10, 20, 50, 60, 15, 90. Asumir que los elementos se agregan en ese orden.

6) Construir (dibujar) un árbol binario de búsqueda con las siguientes letras: l, k, z, n, o, p, t, p, n. Asumir que los elementos se agregan en ese orden.

7) Construir un árbol binario de búsqueda con las fechas de cumpleaños de sus familiares. Imprima las fechas desde el más joven hasta el más antiguo.

8) Construir un árbol binario de búsqueda que corresponda a un recorrido EnOrden imprime: 1, 3, 4, 5, 6, 7, 8, 9 y 10.

9) Implementar el TAD Arbol Binario de Búsqueda (ABB) con elementos de tipo Tipo_Dato (Tipo_Dato en un tipo donde los operadores $=, <, >$ tienen sentido). Se puede utilizar la siguiente estructura:

```
struct Nodo
{
    struct Nodo *izq;
    Tipo_Dato dato;
    struct Nodo *der;
};

typedef struct Nodo* Arbol_T;
```

El TAD debe disponer de las siguientes funciones:

a. `Arbol_T CrearArbol(Tipo_Dato dato);`
// Crea un árbol binario de búsqueda con un único nodo (raíz) cuyo elemento es el parámetro dato. Hijos izquierdo y derecho son árboles vacíos (NULL).

b. `int InsertarElemento(Arbol_T *arbol, Tipo_Dato x );`
// Inserta el elemento x al árbol binario de búsqueda pasado como parámetro. Pre condición: el parámetro arbol es un ABB. Post condición: luego de la operación arbol debe ser un ABB.

c. `void InOrder(Arbolt_T arbol);`
// Realiza el recorrido InOrder imprimiendo sus elementos.

d. `void PostOrder(Arbolt_T arbol);`
// Realiza el recorrido PostOrder imprimiendo sus elementos.

e. `void PreOrder(Arbolt_T arbol);`
// Realiza el recorrido PreOrder imprimiendo sus elementos.

f. `Arbol_T EliminarElemento(Arbol_T *arbol, Tipo_Dato dato);`
// Elimina (si existe) el elemento dato del árbol binario de búsqueda `arbol`. Pre condición: el parámetro arbol es un ABB. Post condición: luego de la operación arbol debe ser un ABB.

g. `int Existe(Arbol_T arbol, Tipo_Dato dato);`
//Retorna True si el elemento dato existe en el árbol `arbol`, False caso contrario.

10) Usando el TAD creado anteriormente, construir un ABB con las claves 50, 25, 75, 10, 40, 60, 90, 35, 45, 70, 42. Recorrerlo utilizando preorden. Verificar que los datos están ordenados de menor a mayor.

11) Agregue al TAD implementado una función recursiva que, dado un árbol binario, cuente la cantidad de nodos de dicho árbol.

12) Agregue al TAD implementado una función recursiva que, dado un árbol, retorne su profundidad.

13) Agregue al TAD implementado una función recursiva que, imprima los valores enteros pares. Implementar para Tipo_Dato = int.

14) Agregue al TAD implementado una función recursiva que, dado un árbol binario, retorne el valor mínimo.

15) Agregue al TAD implementado una función recursiva que, dado un árbol binario, retorne el valor máximo.

16) Escribir una función no recursiva (iterativa) que, dado un árbol binario, lo recorra en postorden.

17) ¿Qué sucede si en un ABB se agregan los datos con el siguiente orden: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Dibujar. Analizar si una búsqueda en este árbol es tan eficiente como en un árbol balanceado.

18) Consideremos el problema de organizar una colección de identificadores de usuario de una computadora y sus palabras clave. Cada vez que un usuario accede al sistema introduciendo su identificador y su palabra clave secreta, el sistema debe comprobar la validez de ambos para verificar que es un usuario legítimo. Puesto que dicha validación debe hacerse en numerosas ocasiones al día, es necesario estructurar esta información de forma que se pueda encontrar rápidamente. Además debe ser una estructura dinámica porque se van añadiendo regularmente usuarios al sistema. Diseñe e implemente.

19) Agregue al TAD implementado una función que reciba como parámetro un árbol binario de búsqueda y un nivel (número). La función debe imprimir todos los datos del nivel indicado. Precondición: el nivel es correcto (existe en el ABB). b) La precondición no existe: realizar una función que retorne la profundidad de un árbol.