Guía Práctica N° 4
Árboles Binarios

Referencias:

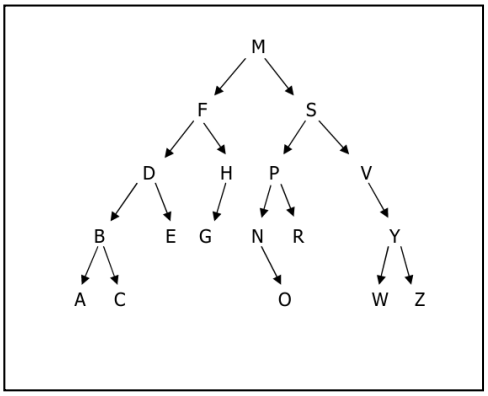


Figura 1

Recomendaciones:

- Utilizar funciones para el desarrollo.
- Probar los programas, en la máquina o con pruebas de escritorio.
- Utilizar nombres de fácil lectura para las variables.

1. Responder Verdadero o Falso.

   En un diagrama de árbol si el nodo X tiene un número de nivel mayor que el nodo Y, entonces el nodo X aparece más abajo en el diagrama que el nodo Y.
2. Si el nodo A tiene tres hermanos y B es el padre de A, ¿cuál es el grado de B?
3. Dado el árbol de la figura 1 enunciar como sería el listado de los nodos según los recorridos en pre-orden, simétrico, post-orden y por niveles.
4. Hacer un algoritmo que informe todos los nodos terminales de un árbol binario.
5. Dado un nodo en un árbol binario de búsqueda, determinar si es hoja. En caso afirmativo producir su baja.
6. Implementar una función que cuente la cantidad de nodos de un árbol binario:
   1. Usando Recursividad.
   2. Usando un método iterativo.
   3. Comparar ambos métodos.
7. Implementar una función que busque si existe un elemento determinado.
8. Hallar la altura de un árbol binario:
   1. Usando Recursividad.
   2. Usando un método interactivo.
   3. Comparar ambos métodos.
9. Realizar una función que determine la profundidad de un nodo en un árbol binario de búsqueda, si el nodo no existe debe retornar el valor simbólico -1.
10. Realizar una función que determine si un árbol binario cumple con la característica que, el valor del dato de cada uno de sus nodos representa la suma de los datos de sus hijos.
11. Realizar una función que determine el menor valor de un (ABB) usando recursión.

12. Realizar una función que determine el mayor valor de un (ABB) usando un algoritmo iterativo.

13. Imprimir la información de todos los nodos terminales que tengan la altura del árbol.

14. Realizar una función que elimine cualquier nodo de un árbol (ABB) de números enteros, según el valor de su campo dato.

15. Realizar una función que elimine todos los nodos de un árbol. Nota: se deben deletear todos sus nodos.

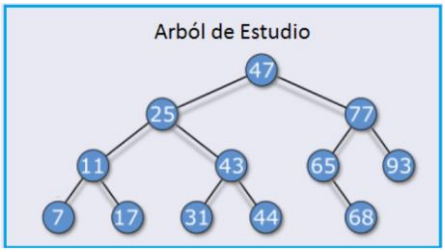
16. Hacer un algoritmo que copie un árbol A en un árbol B. (A existe, B debe ser generado).

17. Hacer un algoritmo que determine si el árbol A es subárbol de B, es decir, si A está contenido en B.

18. Dado un árbol binario, hacer un algoritmo que determine si el árbol es lleno.

19. Realizar un algoritmo que determine si un ABB es AVL.

20. Se desea realizar una implementación de una representación de árbol alternativa, en formato de árbol apaisado, donde los enlaces a la derecha van en la misma línea, y los enlaces izquierdos llevan el símbolo de enlace "\->", como se ve en el ejemplo utilizado.



Salida Obtenida

```
Árbol-->47--> 77--> 93
                \-> 65--> 68
          \-> 25--> 43--> 44
                      \-> 31
                \-> 11--> 17
                      \->  7
```

21. Realizar funciones que, usando recursión, realicen el recorrido de un árbol según los siguientes ordenes:

1. Pre-orden,
2. Simétrico,
3. Post-orden,
4. Por niveles.

22. Idem puntos 1 a 4 usando funciones iterativas. (Ver los pseudocódigos publicados en el aula virtual).

23. Evaluación de árboles de expresiones:

Supongamos que tenemos un limitado árbol que representa tanto sus operaciones como sus números con una variable de tipo char (8 bits un carácter alfanumérico).

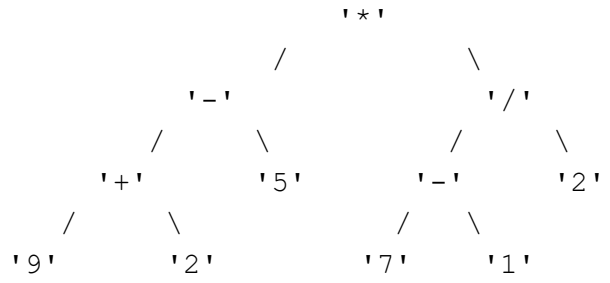
Supongamos también que sus nodos interiores toman alguno de los símbolos *, +, -, / que representan las operaciones de multiplicación, suma, resta y división.

Diseñar una función que tome como parámetro un puntero a árbol de expresión y devuelva el resultado entero de la evaluación de la expresión representada.

El árbol ejemplo siguiente representa la expresión:

```
(9 + 2 - 5) * (7 - 1) / 2
```

Representación del árbol de expresión:



Estructura:

```
struct sArbolExpresion {
    char elemento;
    struct sArbolExpresion * izq;
    struct sArbolExpresion * der;
};
```

Para convertir las hojas puede usar la función:

```
int charToNro(char elemento) {
  //si no es un digito numerico devuelve -1
  if (x >= '0' && x <= '9')
     return elemento - 48;
  else
     return -1;
}
```

24. El recorrido en preorden de un determinado árbol binario es: GEAIBMCLDFKJH y en inorden IABEGLDCFMKHJ.

1. Dibujar el árbol binario.

2. Escribir un programa que compruebe el resultado del apartado anterior.