TRABAJO PRACTICO N: 9.

Tema: Programación Arboles

Objetivos:

* Comprender la definición de árbol en forma general como organización jerárquica, no lineal de datos.
* Diferenciar los distintos conceptos relacionados a los árboles.
* Comprender la definición de árbol binario.
* Discriminar la terminología orientada a de los árboles binarios
* Aprender a crear árboles binarios a partir de un árbol general.
* Aprender a crear árboles binarios y recorrerlos.
* Diferenciar los recorridos pre-orden, in-orden y pos-orden.
* Crear métodos en C para generar árboles y recorrerlos

ARBOLES

El árbol es una estructura de dato no lineal, muy usado porque se adapta a la representación natural de información homogénea organizada, en la que es importante representar datos con una relación jerárquica entre sus elementos. En la figura 1 se representa el esquema general de un árbol.

Un árbol A es un conjunto finito de uno o más nodos, tales que:

1. Existe un nodo especial llamado RAIZ del árbol A.
2. Los nodos restantes A1, A2…. Se dividen en m> =0 conjuntos disjuntos denominados T1,T2,…Tm, cada uno de los cuales es a su vez un árbol. ( Subarbol de RAIZ).

La definición de árbol implica una estructura recursiva, la definición de árbol se refiere a otros árboles.

Definiciones a tener en cuenta

* Raíz del árbol (no tiene padre) no es hijo de ningún elemento.
* Nodo son los vértices o elementos.
* Nodo terminal u hoja, son los nodos que no contienen subárboles
* Cada nodo que no es hoja se asocia a uno o más subárboles llamados descendientes o hijos ( de igual modo cada uno tiene un antecesor o ascendente llamado padre)

A

D

BbBA

E

C

F

G

H

I

J

K

L

Figura 1: Esquema general de un árbol

* Los nodos de un mismo padre se llaman hermanos.
* Los nodos que no son hoja ni raíz son nodos interiores.
* Todos los nodos tienen un solo padre (excepto el de raíz)
* Se denomina camino al enlace entre dos nodos consecutivos
* Se llama rama a un camino que termina en una hoja
* Cada nodo tienen asociado un número de nivel que se determina por la longitud del camino desde el raíz al nodo específico.

En la figura 1:

Nivel 0: A

Nivel 1: B, C, D

Nivel 2: E, F, G, H, I, J

Nivel 3: K, L

* Altura o profundidad de un árbol, es el número máximo de nodos de una rama (equivale al nivel más alto de los nodos +1.
* Peso, de un árbol es el número de nodos terminales. La altura y el peso del árbol de la figura 1 son 4 y 7.

ARBOL BINARIO

Un árbol binario es una estructura de datos en la cual cada nodo tiene un hijo izquierdo y un hijo derecho. Usos comunes de los árboles binarios son: los árboles binarios de búsqueda, los montículos binarios y la Codificación de Huffman.

Un árbol general puede transformarse en un árbol binario mediante la aplicación de determinados algoritmos de conversión

Entonces, es un conjunto finito de ceros o más nodos, tales que:

* Existe un nodo llamado raíz del árbol
* Cada árbol puede tener 0,1 o 2 subárboles ( izquierdo, derecho)

En la figura 2 se muestran ejemplos de árboles binarios

A

5

+

A

C

b)

G

2

c)

K

L

Figura 2: árboles binarios

a)

/

TERMINOLOGIA DE LOS ARBOLES BINARIOS

Un árbol binario lleno, es un árbol en el que cada nodo tiene 0 o 2 hijos.

Un árbol binario perfecto, es un árbol binario lleno en el que todas sus hojas están a la misma profundidad.

Dos árboles binarios se dice que son similares si tienen la misma estructura y son equivalentes si son similares y contienen la misma información

Un árbol binario está equilibrado si las alturas de los dos subárboles de cada nodo del árbol se diferencian como máximo en una unidad.

Altura (subárbol izquierdo) – altura (subárbol derecho) ≤ 1

Lenguaje: C/C++

A

+

A

B

b)equilibrado

E

D

F

K

L

a)Altura 3

/

H

I

J

G

5

4

3

2

1

1

11

2

c)no equilibrado

Los árboles binarios se presentan en C/C++ con estructuras de datos adecuadas como tipo de dato definido por el usuario struct y con clases. La implementación dinámica del árbol requiere variables punteros y asignación dinámica de memoria. Cada nodo contienen: dato, referencia al subárbol izquierdo y referencia al subárbol derecho.

CONVERSION DE UN ARBOL GENERAL EN ÁRBOL BINARIO

Es común transformar árboles generales en binarios, porque es muy sencilla la programación de estos últimos.

Para convertir un árbol general “B” en árbol binario “A”, hay que realizar los siguientes pasos:

1. La raíz de B es la raíz de A.
2. a) enlazar el nodo raíz con el camino que conecta el nodo más a la izquierda ( su hijo).

b) Enlazar este nodo con los restantes descendentes del nodo raíz en un camino, con los que se forma el nivel 1.

c) a continuación repetir los pasos a) y b) con los nodos del nivel 2, enlazando siempre en un mismo camino todos los hermanos (descendentes del mismo nodo). Repetir estos pasos hasta llegar al nivel más alto.

3) Girar el diagrama 45° para diferenciar entre el subárbol izquierdo y el derecho.

Ejemplo1: Convertir el árbol general T de la figura 3 en un árbol binario.

A

D

BbBA

E

C

F

G

H

I

J

K

L

Figura 3: árbol T

A continuación, en la figura 4, se muestra la generación del árbol binario.

A

D

BbBA

E

C

F

G

H

I

J

K

L

Figura 4: árbol binario

RECORRIDO DE UN ARBOL

Se llama proceso de recorrido de un árbol binario, al proceso que permite acceder una sola vez a cada uno de los nodos de un árbol. Cuando un árbol binario se recorre, el conjunto completo de nodos se visitan o examinan.

Según el orden en el que se realizan las siguientes tareas se definirá un método en particular. Las tareas son:

* Visitar el nodo raíz
* Visitar el subárbol izquierdo
* Visitar el subárbol derecho

Métodos de recorrido son:

* *Pre-orden*
* *In-orden*
* *Pos-orden*

Método *pre-orden*

1. visitar el nodo raíz
2. visitar el subárbol izquierdo
3. visitar el subárbol derecho

Método *in-orden*

1. visitar el subárbol izquierdo
2. visitar el nodo raíz
3. visitar el subárbol derecho

Método *pos-orden*

1. visitar el subárbol izquierdo
2. visitar el subárbol derecho
3. visitar el nodo raíz

Para el árbol T de la figura 5, se muestran la forma en la que se visitan los nodos, según cual recorrido recursivo se elija.

M

P

EbBA

B

A

N

Z

V

D

L

Figura 5: árbol T

T

Recorridos:

Pre-Orden: M E B A D L P N V T Z

In-Orden: A B D E L M N P T V Z

Pos-orden: A D B L E N T Z V P M

PROGRAMACION DE UN ARBOL DE BUSQUEDA BINARIO

#include

#include

struct nodo { int dato;

struct nodo \*pder;

struct nodo \* piz;};

void in\_orden ( struct nodo \* p)

{if ( p ! = NULL )

{ in\_orden ( p 🡪 piz);

printf ( “%d”, p 🡪 dato);

in\_orden ( p 🡪 pder);

} }

void pre\_orden ( struct nodo \* p)

{if ( p ! = NULL )

{ printf ( “%d”, p 🡪 dato);

pre\_orden ( p 🡪 piz);

pre\_orden ( p 🡪 pder);

} }

void pos\_orden ( struct nodo \* p)

{if ( p ! = NULL )

{ pos\_orden ( p 🡪 piz);

pos\_orden ( p 🡪 pder);

printf ( “%d”, p 🡪 dato);

} }

int main ( )

struct nodo \* raiz, \*p\_actual, \*q, p\_marca;

raiz = NULL;

char op = ‘S’;

int ban = 0;

// se ingresa el primer nodo

q= new ( struct nodo); // no se controla si hay memoria

scanf (“%d”,& q 🡪dato);

q 🡪pder= q🡪piz = NULL;

raíz = q; // se posiciona el padre

printf ( “desea continuar presione S mayúscula”);

while ( scanf ( “ % c” , & op) = = ‘S’ )

{ // se genera otro nodo

q = new ( struct nodo);

scanf (“%d”,& q 🡪dato);

q 🡪pder= q🡪piz = NULL;

p\_actual = raiz; // inicia búsqueda del lugar

while ( p\_actual != NULL )

{ if ( q 🡪 dato < = p\_ actual 🡪 dato) // si da verdadero va a la izquierda

p\_marca = p-actual;}

p\_actual = p\_actual🡪piz;

ban = 0;// ban avisa que es hacia la izquierda}

else { p\_marca = p\_actual;

p\_actual = p\_ actual 🡪pder;

ban = 1; // va para lado derecho

}

}// cierra el while

// encontró el lugar es una hoja

if ( ban = = 0)

p\_marca 🡪piz = q;

else

p\_marca 🡪 pder = q;

printf ( “desea continuar presione S mayúscula”);

scanf ( “ % c” , & op) ;

} // cierra while externo

// se recorre el árbol in-orden

p\_actual = raiz;

printf ( “ se muestra recorrido in-orden del árbol”);

in\_orden ( p\_actual ) ;

printf ( “ se muestra recorrido pre-orden del árbol”);

p\_actual = raiz;

pre\_orden ( p\_actual);

printf ( “ se muestra recorrido pos-orden del árbol”);

p\_actual = raiz;

pos\_orden ( p\_actual);

getch ( ) ; // detiene para ver salida en SO

return 0; }

EJERCICIOS PROPUESTOS:

1. Implementar un árbol de búsqueda binaria con 10 nodos, los nodos contienen números enteros sin repetición. Se ingresan por teclado en forma desordenada.
2. Implementar una función recursiva que permita recorrer el árbol anterior in-orden para mostrar el contenido del árbol.
3. Cargar un arreglo con N enteros tomados desde el teclado, con repetición. Posteriormente almacenarlos en un árbol binario de búsqueda y finalmente recorrerlo para mostrar por pantalla el contenido del mismo.
4. Se tienen una estructura dinámica tipo pila ya almacenada en memoria de sistema, con los datos almacenados en esta estructura generar un árbol de búsqueda binaria.
5. Se tienen una estructura árbol binario de búsqueda ya cargado, se desea implementar un método que permita tomar por parámetro
6. Se tiene un árbol con enteros con repetición, se desea generar otro sin las repeticiones. Recorrer ambos y mostrar pon pantalla.
7. Se tienen dos árboles binarios de búsqueda, se desea determinar si son equivalentes o si son similares y equivalentes.
8. Se tienen dos árboles binarios de búsqueda que almacenan números enteros, se desea generar otro con el contenido de los dos árboles dados.
9. Se tienen dos árboles binarios de búsqueda que contienen números enteros, se desea generar otro con el contenido de los dos árboles dados, cuya información sea mayor a un entero introducido por teclado.
10. Se tienen dos árboles binarios de búsqueda que contienen números enteros, se desea generar otro con el contenido de los dos árboles dados, con información sea mayor a la media de cada árbol.
11. Implementar un árbol binario de búsqueda con registros de personas (nombre, edad, sexo, ID), los nodos se organizan por campo ID, este es un número de orden, recorrerlo y mostrar por pantalla.
12. Implementar un árbol binario de búsqueda con registros de alumnos (nombre, edad, sexo, ID, promedio general, número de inasistencias), los nodos se organizan por campo ID, este es un número de orden. Recorrerlo y mostrar por pantalla, la siguiente información:
13. Los alumnos sobresalientes ( promedio mayor a 7.00)
14. Los alumnos con menos de 5 inasistencias
15. Convertir el árbol general T en un árbol binario.

A

D

BbBA

E

C

F

G

J

Arbol T

H

I

1. Realizar los recorridos del árbol binario M de la siguiente figura.

2

8

5bBA

1

7

4

12

Arbol M

10