

Capítulo V Árboles Binarios.

Contenido.

1. Introducción.
2. Definición de Árbol Binario.
3. Representación Gráfica de un Árbol Binario.
4. Términos utilizados en los Árboles.
5. Conceptos relacionados con Árboles.
6. Estructura de Datos de un Árbol Binario.
7. Recorridos de Árboles Binarios.
8. Ejemplos de Recorridos
9. Implementación de un árbol binario.
10. Árboles Binarios de Búsqueda
11. Árboles Hilvanados
12. Ejercicios Propuestos.
13. Orientación para el laboratorio.

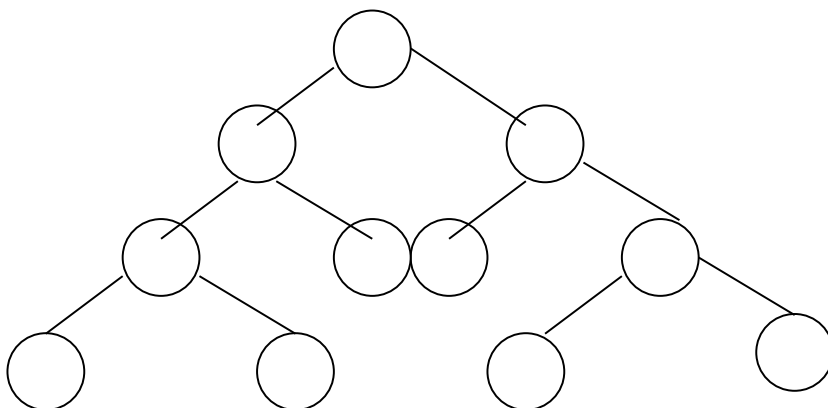
Introducción.

En este capítulo se estudian los árboles binarios; que son, como se representan, sus operaciones básicas, y algunas aplicaciones de las mismas en la informática. El contenido está organizado de lo más simple a lo más complejo y usted podrá ir haciendo ejercicios, completando en los apartados para tal efecto todo lo que falta.

Definición de Árbol Binario

Colección de elementos o nodos, donde cada nodo tiene dos subárboles un subárbol Izquierdo y un subárbol Derecho, que a su vez son Árboles Binarios. El primer nodo o elemento se denomina raíz. Si árbol tiene cero nodos se dice que está vacío.

Representación Gráfica de un Árbol Binario.



Árbol Binario

Términos utilizados en los árboles.

- Suponga el caso de un Árbol A y que el nodo N es un nodo de A con un sucesor izquierdo “S₁” y un sucesor derecho “S₂.”

Podemos decir :

- ❖ N se llama el padre de “S₁” y “S₂”
- ❖ “S₁” se llama el hijo izquierdo de N
- ❖ “S₂” se llama el hijo derecho de N.
- ❖ “S₁” y “S₂” son hermanos

Cada nodo N de un árbol binario A, excepto la raíz, tiene un único padre, llamado predecesor.

- Términos descendiente y antecesor.
 - ❖ Un nodo M se dice descendiente de un nodo N si existe una sucesión de hijos desde N hasta M.
 - ❖ Un nodo N se dice antecesor de M si existe una sucesión de hijos desde N hasta M.
 - ❖ M se dice descendiente izquierdo o derecho de N dependiendo de si pertenece al subárbol izquierdo o al derecho de N.

Conceptos relacionados con Árboles.

Grado de un nodo

Cantidad de hijos de un nodo.

Grado del Árbol

Número máximo de hijos entre todos los nodos.

Hojas

Aquellos nodos que no tienen hijos

Nivel de un nodo

Cada nodo de un árbol binario tiene asignado un número de nivel, de la siguiente forma.

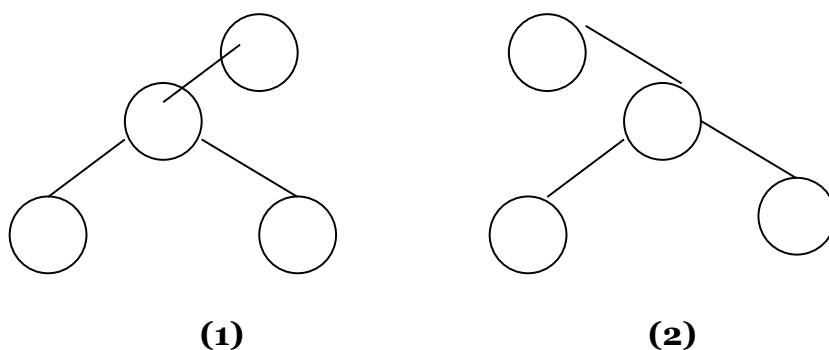
- ❖ A la raíz del árbol se le asigna el número de nivel 0
- ❖ El nivel de cualquier nodo se obtiene incrementando en uno el nivel de su padre.
- ❖ Los nodos con el mismo número de nivel se dice que pertenecen a la misma generación.

Nivel de árbol

El nivel del árbol es el máximo nivel de los nodos del Árbol

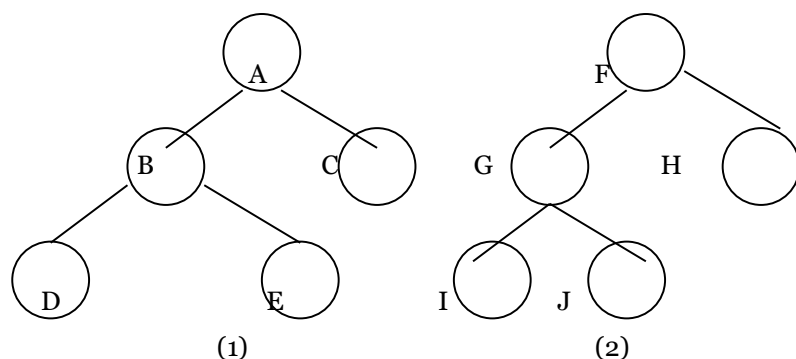
Árboles Binarios Distintos, Similares y Equivalentes

Dos árboles binarios son **DISTINTOS** cuando sus estructuras son diferentes.



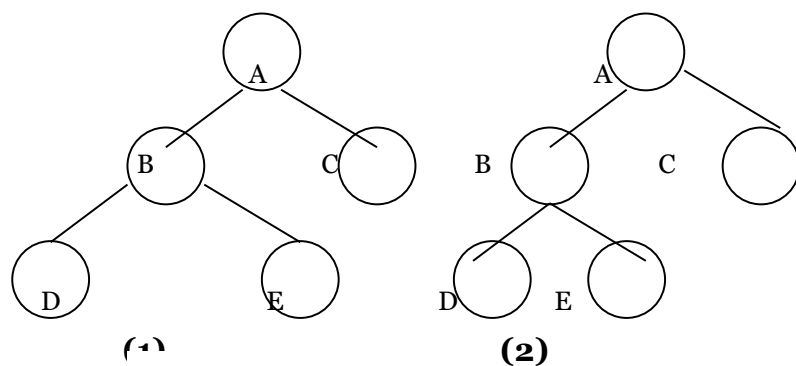
Árboles Binarios Distintos

Dos árboles binarios son **SIMILARES** cuando sus estructuras son idénticas, es decir tiene la misma forma, pero la información que contienen sus nodos difiere entre sí.



Árboles Binarios Similares

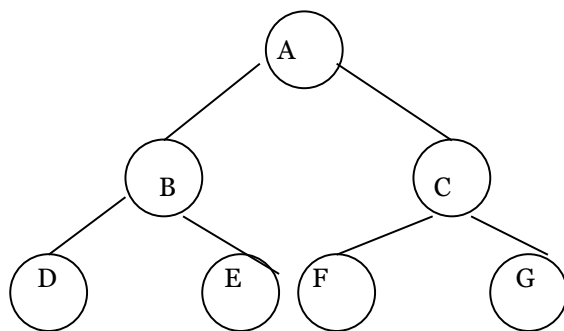
Dos árboles binarios son **EQUIVALENTES** ó copias cuando son similares y además los nodos contienen la misma información.



Árboles Binarios Equivalentes

Árboles Binarios Completos

El árbol se dice que es completo cuando todos los nodos tienen grado igual al grado del árbol



Árbol Completo

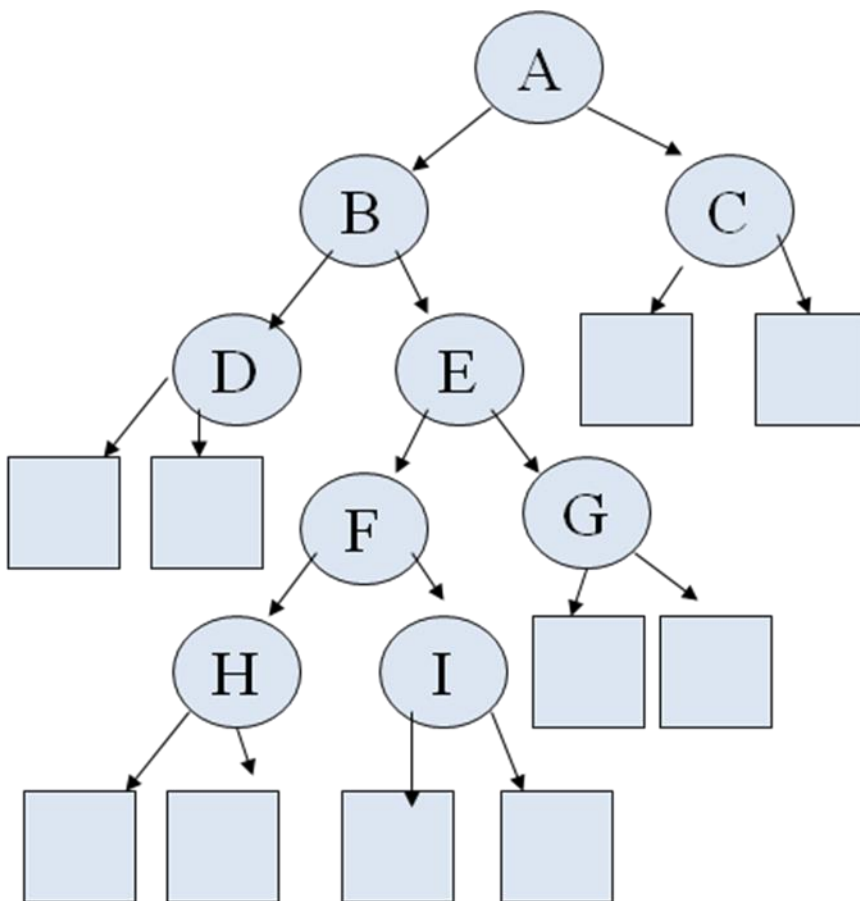
Árboles Extendidos

Un árbol extendido es aquel en que el número de hijos de cada nodo es igual al grado del árbol.

Si algún nodo no cumple esta condición entonces se le incorporará tantos nodos necesarios para que cumpla la condición, llamados nodos especiales.

Los nodos especiales tienen como objetivo reemplazar las ramas vacías o nulas, no pueden tener descendientes y normalmente se representan con la forma de cuadrado.

En la figura siguiente se muestra un ejemplo y su representación gráfica.

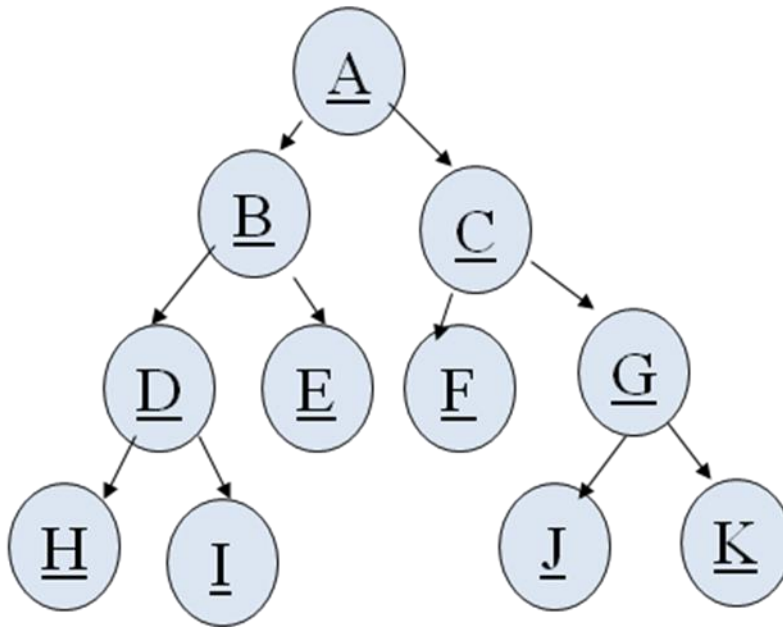


Árbol Extendido

Árboles Binario Extricto

Cada nodo tiene dos enlaces o ninguno.

En la figura siguiente se muestra un ejemplo y su representación gráfica.



Árbol Binario Extricto.

Estructura de Datos de un Árbol Binario.

Un árbol binario está compuesto por nodos los cuales los definimos como una estructura, que tiene cada uno un campo información y un campo enlace izquierdo que enlazará este nodo con su Subárbol Izquierdo y un enlace derecho que lo enlazará con su Subárbol Derecho.

```
struct nodo {  
    string info;  
    struct nodo *EI;  
    struct nodo *ED;  
};
```

Recorridos de Árboles Binarios

- ❖ Recorrido Preorden.
- ❖ Recorrido en Entreorden.
- ❖ Recorrido en Posorden.

Recorrido en Preorden

Se visita la raíz

subárbol Izquierdo Preorden

subárbol Derecho en Preorden

```
void PreOrden(nodo *r)
{
    if (r != NULL)
    {
        cout<< r->info + " ";
        PreOrden(r->EI);
        PreOrden(r->ED);
    }
    return;
}
```

Recorrido en EntreOrden

Se visita el subárbol Izquierdo en Entreorden

Se visita la Raíz

Se visita el subárbol Derecho en Entreorden

```
void EntreOrden(nodo *r)
{
    if (r != NULL)
    {
        EntreOrden(r->EI);
        cout<< r->info + " ";
        EntreOrden(r->ED);
    }
    return;
}
```


Recorrido en Posorden

subárbol Izquierdo en Posorden

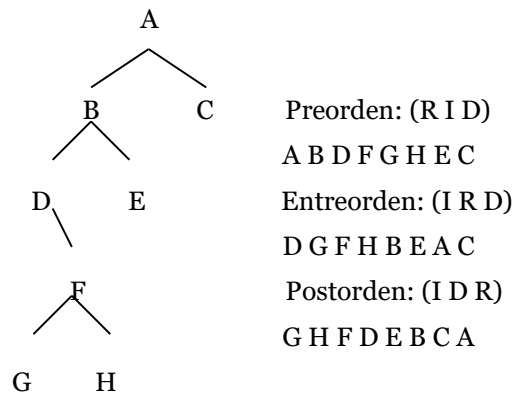
subárbol Derecho en Posorden

Raíz

```
void PosOrden(nodo *r)
{
    if (r != NULL)
    {
        PosOrden(r->EI);
        PosOrden(r->ED);
        cout<< r->info + " ";
    }
    return;
}
```

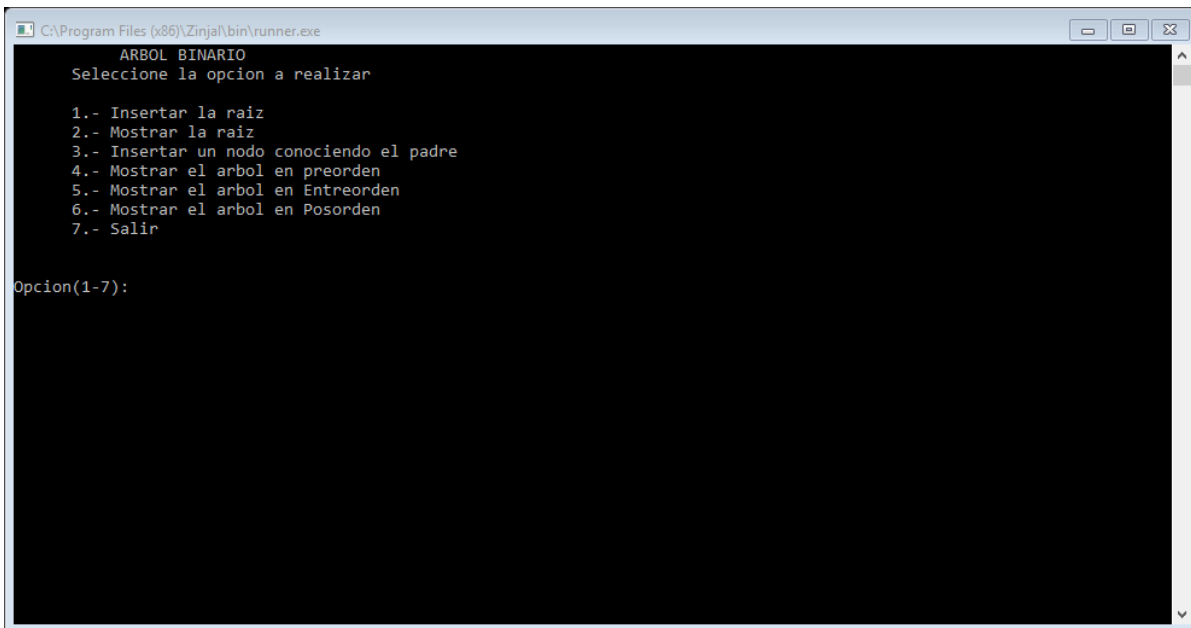
Ejemplos de recorridos

A continuación, se expone un ejemplo de cómo recorrer un árbol con los diferentes recorridos.



Implementación de un Árbol Binario.

A continuación, se muestra una interfaz en consola y la creación de un árbol binario, así como sus recorridos.



```
C:\Program Files (x86)\Zinjal\bin\runner.exe
ARBOL BINARIO
Seleccione la opcion a realizar

1.- Insertar la raiz
2.- Mostrar la raiz
3.- Insertar un nodo conociendo el padre
4.- Mostrar el arbol en preorden
5.- Mostrar el arbol en Entreorden
6.- Mostrar el arbol en Posorden
7.- Salir

Opcion(1-7):
```

Código asociado:

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
```

//Arbol Binario

using namespace std;

//Se define el nodo del árbol, con un campo info y dos enlaces

```
struct nodo {  
    string info;  
    struct nodo *EI;  
    struct nodo *ED;
```

```
};
```

```
nodo *raiz;
```

```
nodo *aux;
```

```
void iniciar()
```

```
{  
    raiz=NULL;  
    aux=NULL;  
}
```

```
int vacia()
{
    if (raiz==NULL)
        return 1;
    else
        return 0;
}

void Insertar_Raiz(string valor)
{
    if (vacía())
    {
        raiz = new nodo;
        raiz->info=valor;
        raiz->EI=NULL;
        raiz->ED=NULL;
    }
}
```

void insertar_izquierdo (nodo *padre, string valor)

```
{  
    nodo *nodonuevo;  
    nodonuevo= new nodo;  
    nodonuevo->info=valor;  
    nodonuevo->EI=NULL;  
    nodonuevo->ED=NULL;  
    padre->EI=nodonuevo;  
    return;  
}
```

void insertar_derecho (nodo *padre, string valor)

```
{  
    nodo *nodonuevo;  
    nodonuevo= new nodo;  
    nodonuevo->info=valor;  
    nodonuevo->EI=NULL;  
    nodonuevo->ED=NULL;  
    padre->ED=nodonuevo;  
    return;  
}
```

```
void PreOrden(nodo *r)  
{  
    if (r != NULL)  
    {  
        cout<< r->info + " ";  
        PreOrden(r->EI);  
        PreOrden(r->ED);  
    }  
    return;  
}
```

```
void EntreOrden(nodo *r)  
{  
    if (r != NULL)  
    {  
        EntreOrden(r->EI);  
        cout<< r->info + " ";  
        EntreOrden(r->ED);  
    }  
    return;  
}
```

```
void PosOrden(nodo *r)
```

```
{  
    if (r != NULL)  
    {  
        PosOrden(r->EI);  
        PosOrden(r->ED);  
        cout<< r->info + " ";  
    }  
    return;  
}
```

```
void buscar(nodo *r, string nombre)
```

```
{  
    if ((r != NULL))  
    {  
        if (r->info == nombre)  
        {    aux=r;        }  
        else  
        {  
            buscar(r->EI, nombre);  
            buscar(r->ED, nombre);  
        }  
    }  
    return;  
}
```

```
int main(int argc, char *argv[])
{
    string cod, nombre, c, RPreorden;
    int opc;
    iniciar();
    do
    {
        system("cls");
        cout<<"      ARBOL BINARIO"<<endl;
        cout<<"  Seleccione la opcion a realizar\n\n";
        cout<<"  1.- Insertar la raiz\n";
        cout<<"  2.- Mostrar la raiz\n";
        cout<<"  3.- Insertar un nodo conociendo el
padre\n";
        cout<<"  4.- Mostrar el arbol en preorden\n";
        cout<<"  5.- Mostrar el arbol en Entreorden\n";
        cout<<"  6.- Mostrar el arbol en Posorden\n";
        cout<<"  7.- Salir";
        cout<<"  \n\n\nOpcion(1-7): ";
        cin>>opc;
```



```
switch(opc)
{
case 1:
    cod="";
    cout<<"Entre el valor a insertar en la raiz:\n ";
    cin>>cod;
    Insertar_Raiz(cod);
    break;
case 2:
    cout<<" "+ raiz->info + " ";
    cout<<" ";
    cout<<"Oprima una tecla para salir \n";
    getchar();
    cin>>cod;
    break;
case 3:
    cout<<"Entre el valor de otro nodo: \n";
    cin>>cod;
    cout<<"Entre el valor del padre: \n";
    cin>>nombre;
    buscar(raiz, nombre);
    if (aux==NULL)
        cout<<"No existe el padre";
    else
```

//Vamos a insertar en el enlace izquierdo y sino en el derecho

//Si ya tiene dos hijos se emitira un mensaje de error

```
{  
    if (aux->EI==NULL)  
        insertar_izquierdo(aux, cod);  
    else  
        {  
            if(aux->ED==NULL)  
                insertar_derecho(aux,cod);  
            else  
                cout<<"Ya tiene dos hijos";  
            }  
        }  
    cin>>cod;  
    getchar();  
  
    break;
```

case 4:

```
    PreOrden(raiz);  
    cout<<"Oprima una tecla para salir";  
    getchar();  
    cin>>cod;  
    break;
```

case 5:

```
    EntreOrden(raiz);
```

```
        cout<<"Oprima una tecla para salir";
        getchar();
        cin>>cod;
        break;
    case 6:
        PosOrden(raiz);
        cout<<"Oprima una tecla para salir";
        getchar();
        cin>>cod;
        break;
    case 7:
        exit(0);
        break;
    }
}
while ((opc!=7));

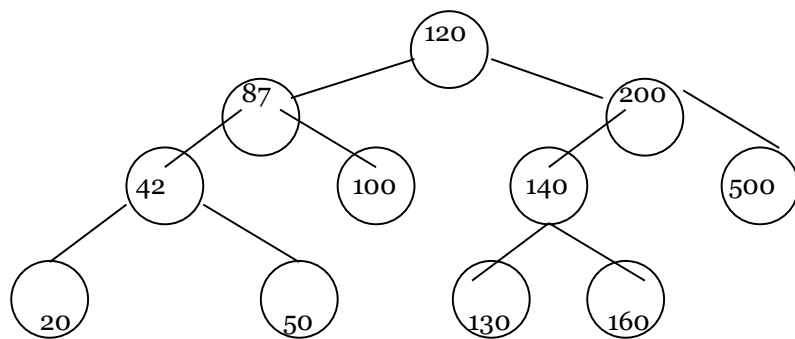
return 0;
}
```

Interfaz para trabajo con árboles binarios, utilizando un entorno de desarrollo visual



Árboles Binarios de Búsquedas.

Para cada Nodo del árbol debe cumplirse que todos los valores de los nodos del subárbol izquierdo deben ser menores al nodo raíz. De igual forma todos los valores de los nodos del subárbol derecho deben ser mayores.



Estructura de Datos para un árbol de búsqueda

```
struct nodo {  
    int info;  
    struct nodo *EI;  
    struct nodo *ED;  
  
};
```

```
nodo *raiz;
```

Crear un Árbol de Búsqueda.

```
void Insertar(int valor)  
{ nodo *nuevonodo, *p, *s;  
  
    nuevonodo = new nodo;  
    nuevonodo->info=valor;  
    nuevonodo->EI=NULL;  
    nuevonodo->ED=NULL;  
    if (vacía())  
    {  
        raiz=nuevonodo;  
    }  
    else  
    { p=raiz;  
  
        while (p!=NULL)
```

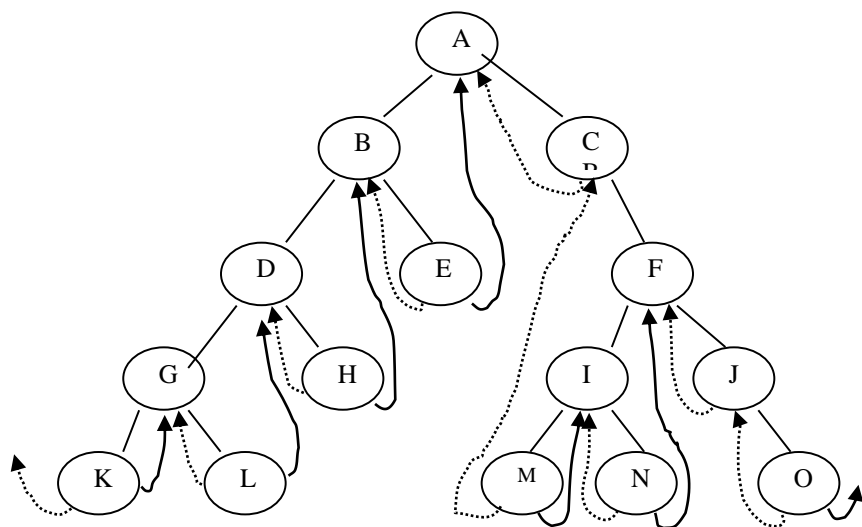
```
{
    if (valor>p->info)
    {s=p;
      p=p->ED;
    }
    else
    {
      s=p;
      p=p->EI;
    }
}

if (valor > s->info)
    s->ED=nuevonodo;
else
    s->EI=nuevonodo;
}
}
```

Árboles Hilvanados.

Un Arbol Hilvanado es un Arbol donde cada nodo tiene enlace izquierdo y derecho diferentes a NIL, con excepción de dos casos: El primer nodo que no tiene antecesor y el último nodo que no tiene sucesor.

Representación gráfica



Definición de tipos:

```

struct nodo {
    string info;
    struct nodo *EI;
    struct nodo *ED;
    int indizq,indder;

};

```

Recorrido en Entreorden en un Árbol Hilvanado.

Procedimiento recorrido Entreorden (Raíz: Puntero Arbol)

Iniciar

P = Raíz

Mientras (P ^ Enlace Izquierdo <> NIL)

P = P ^ Enlace Izquierdo Visualizar P ^ Información

Q = P

Mientras Q <> Nil hacer

Q= Sucesor (P) Visualizar (Q ^ Información)

P = Q

Fin Mientras

Terminar

Hilvanar un Arbol Binario.

Para hilvanar un Arbol se tiene que tomar en cuenta el recorrido en *Entreorden*,

Procedimiento Hilvanar (Raíz : Puntero Arbol)

Iniciar

Arreglo : = recorrido entreorden

Arreglo [1] ^ Indicador Izquierdo = Cierto

SI Arreglo[1] ^ Enlace Derecho = Nil ENTONCES

Arreglo[1]^ . Enlace derecho = arreglo [2]

Arreglo[1]^ Indicador Derecho = Cierto

SINO Arreglo [1]^ Indicador Derecho = Falso

FIN SI

Arreglo[c]^ Indicador Derecho = cierto

Si Arreglo[c]^ Enlace Izquierdo = Nil ENTONCES

Arreglo[c]^ .Enlace Izquierdo = Arreglo[c-1]

Arreglo[c]^ .indicador Izquierdo = Cierto

SINO Arreglo[c]^ .indicador Izquierdo = Falso

FIN SI

DESDE $i=2$ HASTA $c-1$ HACER

SI $\text{Arreglo}[i]^{\wedge}.\text{Enlace Izquierdo} = \text{Nil}$ ENTONCES

$\text{Arreglo}[i]^{\wedge}.\text{Enlace Izquierdo} = \text{Arreglo}[i-1]$

$\text{Arreglo}[i]^{\wedge}.\text{Indicador Izquierdo} = \text{Cierto}$

SINO $\text{Arreglo}[i]^{\wedge}.\text{indicador Izquierdo} = \text{Falso}$

FIN SI

SI $\text{Arreglo}[i]^{\wedge}.\text{Enlace Derecho} = \text{Nil}$ ENTONCES

$\text{Arreglo}[i]^{\wedge}.\text{Enlace Derecho} = \text{Arreglo}[i+1]$

$\text{Arreglo}[i]^{\wedge}.\text{Indicador Derecho} = \text{Cierto}$

SINO $\text{Arreglo}[i]^{\wedge}.\text{inEnlace Derecho} = \text{Falso}$ FINSI

Terminar

Aplicaciones de los Árboles Binarios.

Decisiones de Dos Caminos.

Árbol Genealógico.

Ordenamiento de Números.

Almacenamiento de Expresiones Aritméticas.

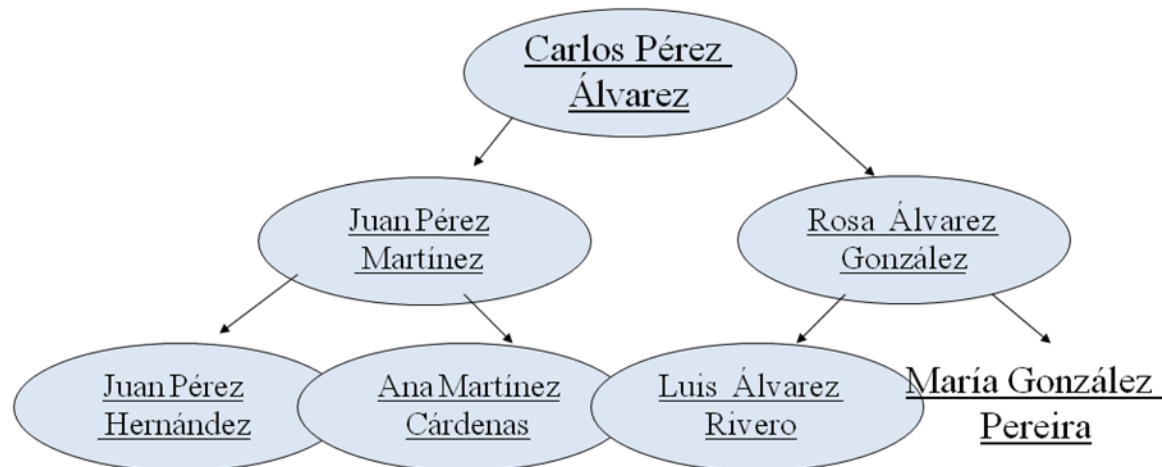
Árboles de Juegos.

Codificación de Mensajes.

Decisiones de dos caminos

- Supongamos que queremos encontrar los duplicados de una lista de números.
- Primero : Se podrían comparar cada número con todos los que le preceden.
- Otra solución: Colocar los números en un árbol, el primero en la raíz, los demás si es menor a la izquierda y si no a la derecha.
 - En este caso se utilizaría un árbol de búsqueda.
 - Si se quiere ir por un camino, se compara con la raíz y según se desee se va hacia la izquierda ó hacia la derecha.

Árbol Genealógico



Seminario.

Implementar el trabajo con árboles binarios, como hemos visto en clases, utilizando .NET, los lenguajes, C++, C# y Visual basic .NET.

Laboratorio.

Implementar un árbol binario, utilizando lo estudiado anteriormente y adicione los siguientes ejercicios:

- 1.- Cuantos hijos tiene un determinado nodo.
- 2.- Grado de un nodo determinado.

A continuación de muestra el código que debe utilizar, y de cual deberá partir para hacer su trabajo:

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>

//Arbol Binario

using namespace std;

//Se define el nodo del arbol, con un campo info y dos enlaces
struct nodo {
    string info;
    struct nodo *EI;
    struct nodo *ED;
};

nodo *raiz;
nodo *aux;

void iniciar()
{
    raiz=NULL;
    aux=NULL;
}

int vacia()
{
    if (raiz==NULL)
        return 1;
    else
```

```
        return o;

    }

void Insertar_Raiz(string valor)
{
    if (vacía())
    {
        raiz = new nodo;
        raiz->info=valor;
        raiz->EI=NULL;
        raiz->ED=NULL;
    }
}

void insertar_izquierdo ( nodo *padre, string valor)
{
    nodo *nodonuevo;

    nodonuevo= new nodo;
    nodonuevo->info=valor;
    nodonuevo->EI=NULL;
    nodonuevo->ED=NULL;
    padre->EI=nodonuevo;

    return;
}

void insertar_derecho ( nodo *padre, string valor)
{
    nodo *nodonuevo;
    nodonuevo= new nodo;
    nodonuevo->info=valor;
```

```
        nodonuevo->EI=NULL;
        nodonuevo->ED=NULL;
        padre->ED=nodonuevo;

        return;

    }

void PreOrden(nodo *r)
{
    if (r != NULL)
    {
        cout<< r->info + " ";
        PreOrden(r->EI);
        PreOrden(r->ED);
    }
    return;
}

void EntreOrden(nodo *r)
{
    if (r != NULL)
    {

        EntreOrden(r->EI);
        cout<< r->info + " ";
        EntreOrden(r->ED);
    }
    return;
}

void PosOrden(nodo *r)
{
    if (r != NULL)
```

```
{

    PosOrden(r->EI);
    PosOrden(r->ED);
    cout<< r->info + " ";
}
return;
}

void buscar(nodo *r, string nombre)
{
    if ((r != NULL))
    {
        if (r->info == nombre)
        {
            aux=r;

        }
        else
        {
            buscar(r->EI, nombre);
            buscar(r->ED, nombre);
        }
    }

    return;

}

int main(int argc, char *argv[])
{

    string cod, nombre, c, RPreorden;
    int opc;
```

```
iniciar();
do
{
    system("cls");
    cout<<"    ARBOL BINARIO"<<endl;
    cout<<"  Seleccione la opcion a realizar\n\n";
    cout<<"    1.- Insertar la raiz\n";
    cout<<"    2.- Mostrar la raiz\n";
    cout<<"    3.- Insertar un nodo conociendo el padre\n";
    cout<<"    4.- Mostrar el arbol en preorden\n";
    cout<<"    5.- Mostrar el arbol en Entreorden\n";
    cout<<"    6.- Mostrar el arbol en Posorden\n";
    cout<<"    7.- Salir";
    cout<<"    \n\nOpcion(1-7): ";
    cin>>opc;

    switch(opc)
    {
        case 1:
            cod="";
            cout<<"Entre el valor a insertar en la raiz:\n ";
            cin>>cod;
            Insertar_Raiz(cod);
            break;

        case 2:
            cout<<" "+raiz->info + " ";
            cout<<" ";
            cout<<"Oprima una tecla para salir \n";
            getchar();
            cin>>cod;
            break;

        case 3:
            cout<<"Entre el valor de otro nodo: \n";
            cin>>cod;
            cout<<"Entre el valor del padre: \n";
            cin>>nombre;
```

```
        buscar(raiz, nombre);
        if (aux==NULL)
            cout<<"No existe el padre";
        else
            //Vamos a insertar en el enlace izquierdo y sino en el derecho
            //Si ya tiene dos hijos se emitira un mensaje de error
            {
                if (aux->EI==NULL)
                    insertar_izquierdo(aux, cod);
                else
                    {
                        if(aux->ED==NULL)
                            insertar_derecho(aux,cod);
                        else
                            cout<<"Ya tiene dos hijos";
                    }
            }
        cin>>cod;
        getchar();

        break;

    case 4:
        PreOrden(raiz);
        cout<<"Oprima una tecla para salir";
        getchar();
        cin>>cod;
        break;

    case 5:
        EntreOrden(raiz);
        cout<<"Oprima una tecla para salir";
        getchar();
        cin>>cod;
        break;

    case 6:
        PosOrden(raiz);
        cout<<"Oprima una tecla para salir";
```



```
        getchar();
        cin>>cod;
        break;
    case 7:
        exit(0);
        break;

    }
}
while ((opc!=7));

return 0;
}
```

Estudio Individual

Dado los siguientes árboles binarios:

1.- Obtenga los recorridos en

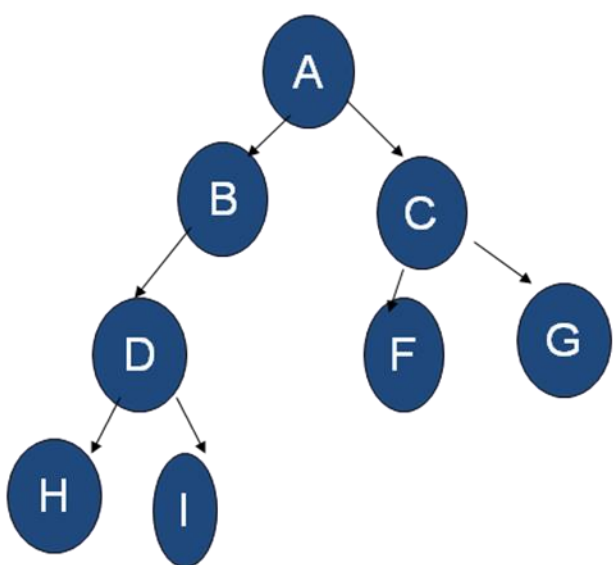
Preorden.

Entreorden.

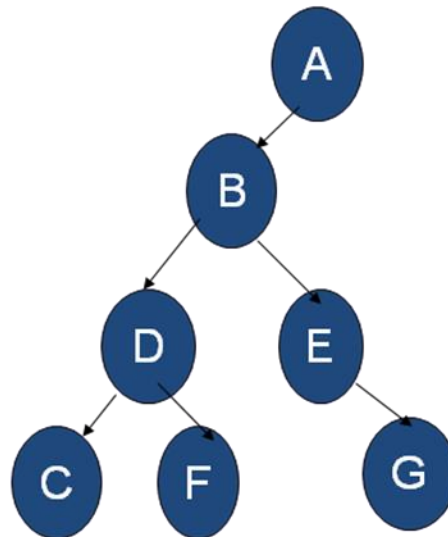
Posorden.

2.- Diga si es un árbol de Búsqueda

3.- Hilvane el Árbol.

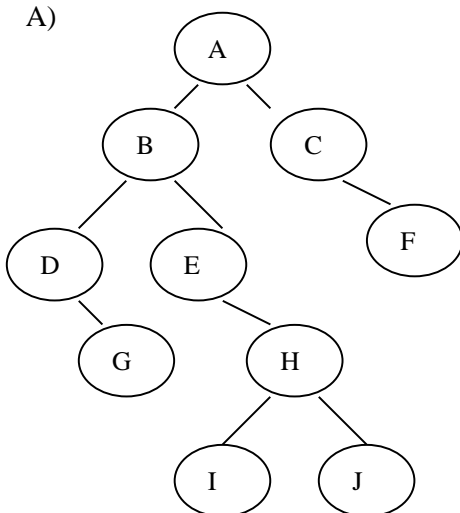


Preorden:
 Entreorden:
 Posorden:

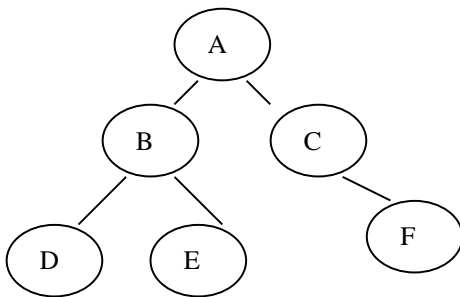
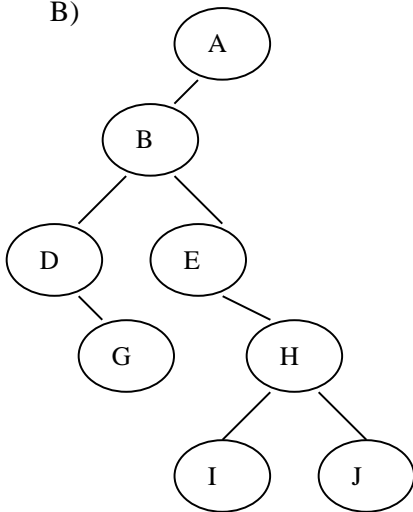


Preorden:
 Entreorden:
 Posorden

A)



B)



Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.

Este Documento es de uso personal e intransferible, no se permite la reproducción total o parcial de este documento por ningún medio.
