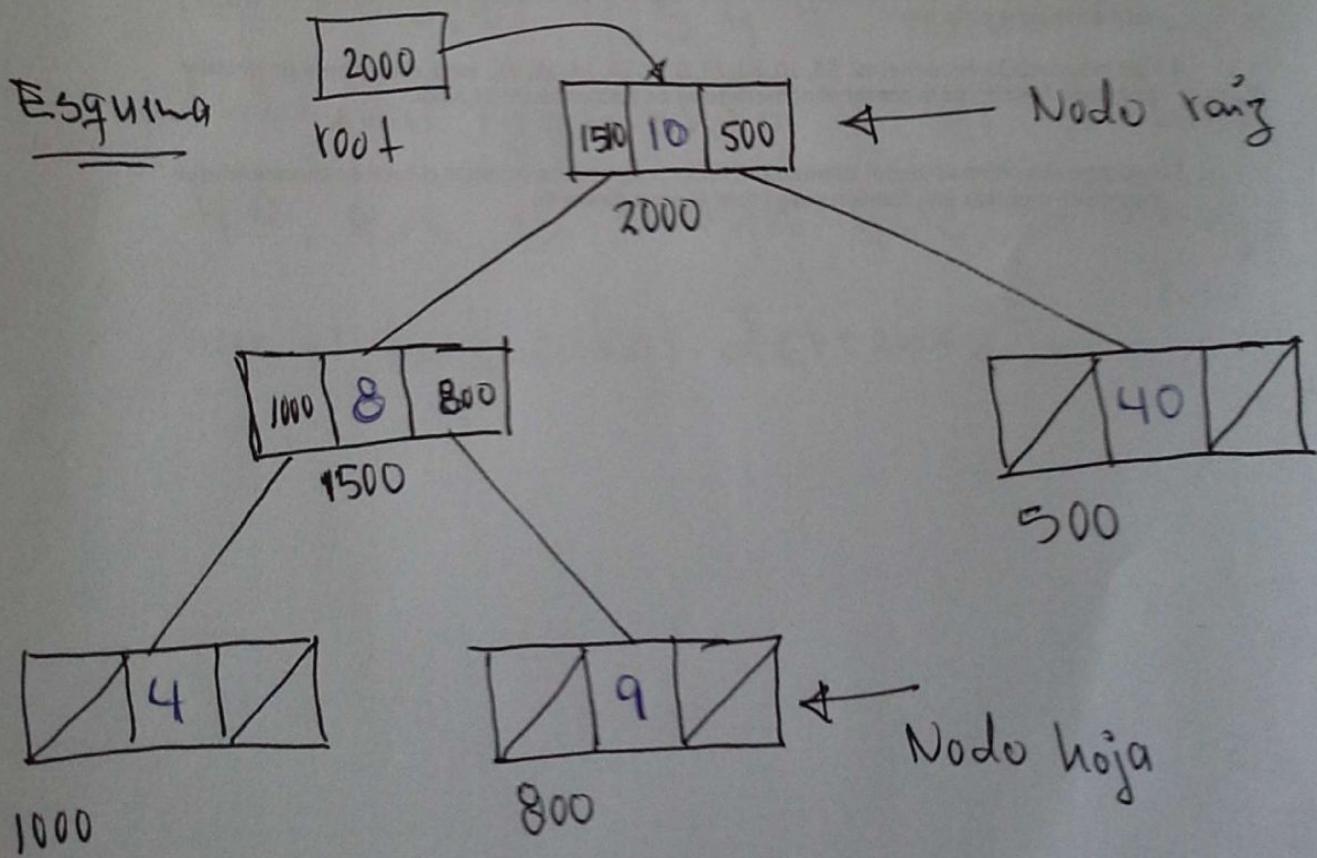


Arbol Binario

Def.

Un arbol binario es una estructura de datos no lineal, en la cual cada nodo tiene un hijo izq. y un hijo derecho; ningún nodo puede tener mas de dos hijos. Un nodo sin hijos es llamado nodo hoja y sus enlaces izq. y derecho son ambos nulos.



Arbol binario (cont...)

En un arbol binario, los contenidos de los nodos del subarbol izquierdo son menores que el contenido de la raiz.

Los contenidos del sub-arbol derecho son mayores que el contenido de la raiz.

Por tal motivo, en el esquema de arbol los numeros 4, 8 y 9 están en el sub-arbol izquierdo y el numero 40, que es mayor que la raiz está en el sub-arbol derecho.

formas de recorrer los elementos de un arbol binario.

recorrido en orden previo.
(preorder).

- primero se visita la raiz.
- Subarbol Izquierdo
- Subarbol Derecho

es decir;

```
void Arbol::preorder(Nodo *root){  
    if (root) = NULL) {  
        cout << root->num;  
        preorder(root->Left);  
        preorder(root->Right);  
    }  
}
```

Recorrido en orden simétrico.
(inorder).

- Subárbol Izquierdo
- Raíz
- Subárbol Derecho

es decir;

```
void Arbol::inorder(Nodo *root){  
    if (root != NULL) {  
        inorder(root->left);  
        cout << root->num;  
        inorder(root->right);  
    }  
}
```

recorrido en orden posterior.
(posOrder)

- Subarbol Izquierdo
- Subarbol Derecho
- Raíz

es decir;

```
void Arbol:: posOrder(Nodo *root){  
    if (root != NULL) {  
        posOrder(root -> Left);  
        posOrder(root -> Right);  
        cout << root -> data;  
    }  
}
```

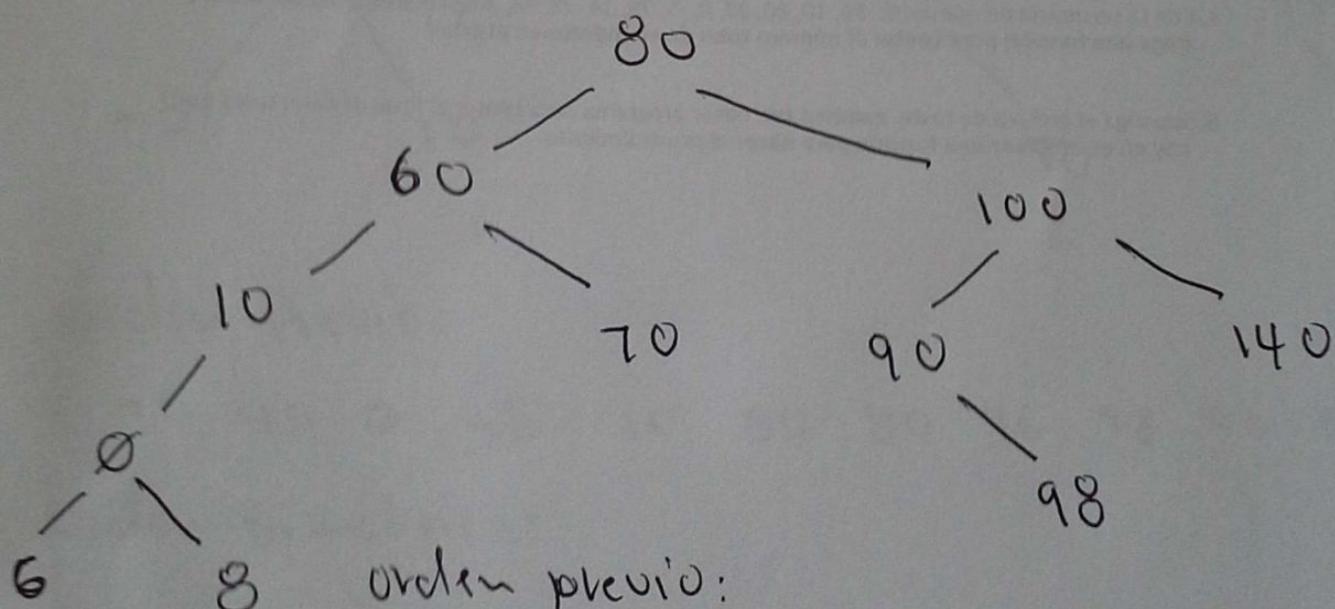
Ejercicio.

Con la secuencia de numeros enteros, formar el esquema de arbol binario y efectuar los recorridos en orden previo, simétrico y posterior. No olvidar que los elementos menores que la raíz van en el subarbol izq y los elementos mayores que la raíz van del lado derecho.

80, 60, 100, 70, 10, \emptyset , -6, 8, 90, 98, 140.

Sol.

El primer elemento que llega al sistema es el 80, así que este número será la raíz del arbol.



Orden previo:

80 60 10 \emptyset -6 8 70 100 90 98 140

Orden simétrico:

-6 \emptyset 8 10 60 70 80 90 98 100 140

Orden posterior:

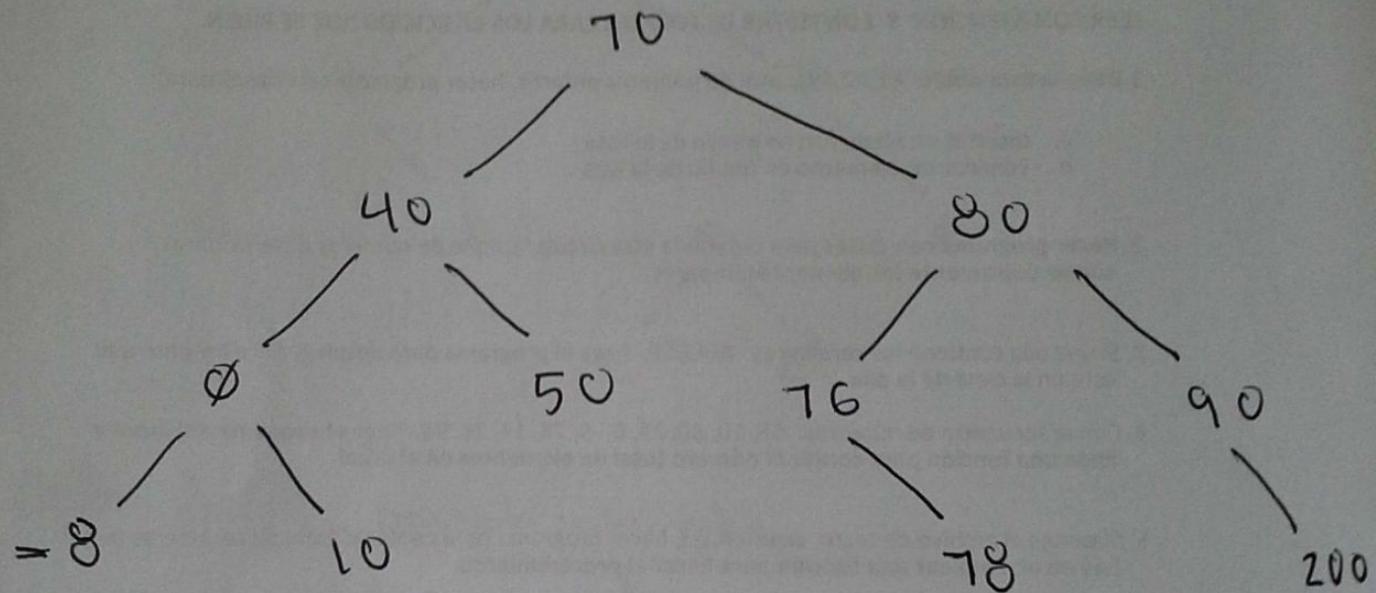
-6 8 \emptyset 10 70 60 98 90 140 100 80

Ejercicio.

Con la secuencia de numeros, hacer el esquema de arbol binario y efectuar los recorridos en orden previo, simétrico y posterior.

70, 40, 50, 0, -8, 10, 80, 90, 76, 78, 200.

Sol.



orden Previo:

70 40 0 -8 10 50 80 76 78 90 200

orden simétrico:

-8 0 10 40 50 70 76 78 80 90 200

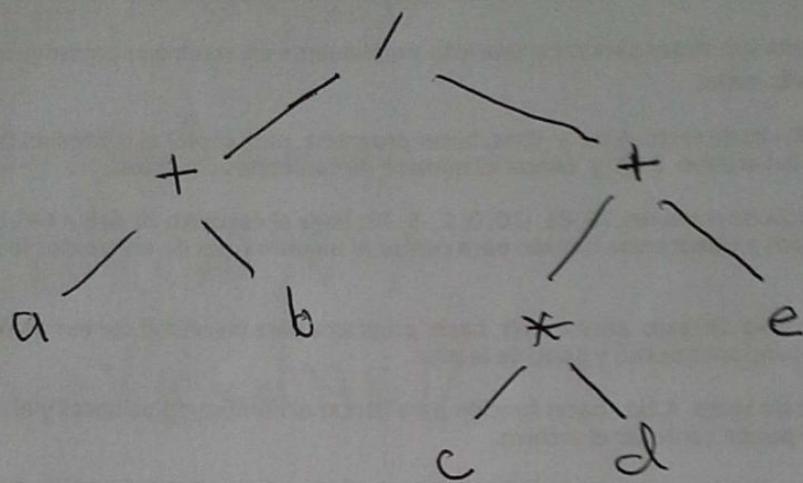
orden Posterior:

-8 10 0 50 40 78 76 200 90 80 70

Ejercicio.1

Con la expresión: $(a+b)/(c+d)+e$, hacer el esquema de árbol binario y efectuar los recorridos en orden previo y orden posterior, y en cada caso obtener la expresión original.

Sol.



Orden Previo: operador, valor, valor

/ + a b + * c d e

expresión original:

$/ (a+b)+*cde$

$/ (a+b)+ (c*d)e$

$/ (a+b) ((c+d)+e)$

$(a+b)/((c+d)+e)$

(continuación...)

orden Posterior: valor, valor, operador.

ab+cd* e + /

expresión original:

$$(a+b)cd \neq e + 1$$

$$(a+b)(c+d)e + f$$

$$(a+b)((c+d)+e)/$$

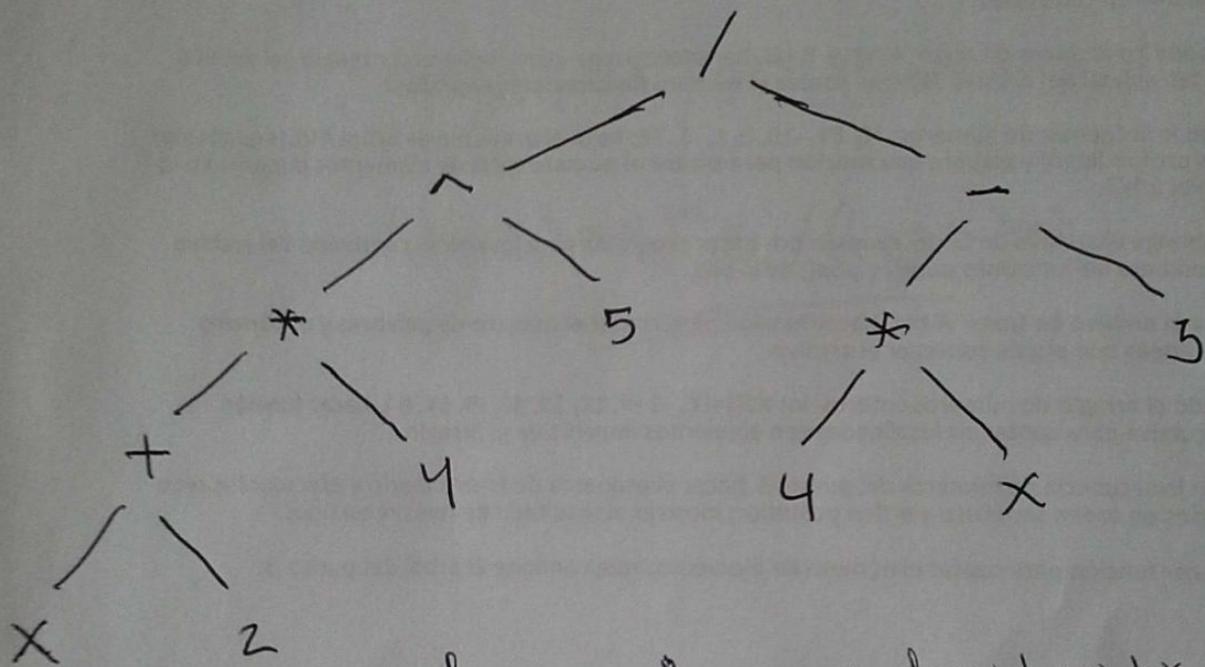
$$(a+b)/(c+d+e)$$

Ejercicio. (tomado de youtube).

Con la expresión: $((x+2)*4)^5 / ((4*x)-3)$,

Hacer el esquema de árbol binario y efectuar los recorridos en orden previo y orden posterior, y en cada caso obtener la expresión original.

Sol.



orden previo: operador, valor, valor

/ ^ * + x 2 4 5 - * 4 x 3

expresión original:

$/ \wedge * (x+2) 4 5 - (4 * x) 3$

$/ \wedge ((x+2) * 4) 5 ((4 * x) - 3)$

$/ ((x+2) * 4)^5 ((4 * x) - 3)$

$((x+2) * 4)^5 / ((4 * x) - 3)$

(Continuación ...)

Orden Posterior: Valor, Valor, operador

$$x^2 + 4 \neq 5^4 \times 3 - 1$$

expression original:

$$(x+2)^4 + 5^x (4+x)^3 - 1$$

$$((x+2)*4)5^((4*x)-3) /$$

$$\frac{((x+2)*4)^5}{(4*x)-3}$$

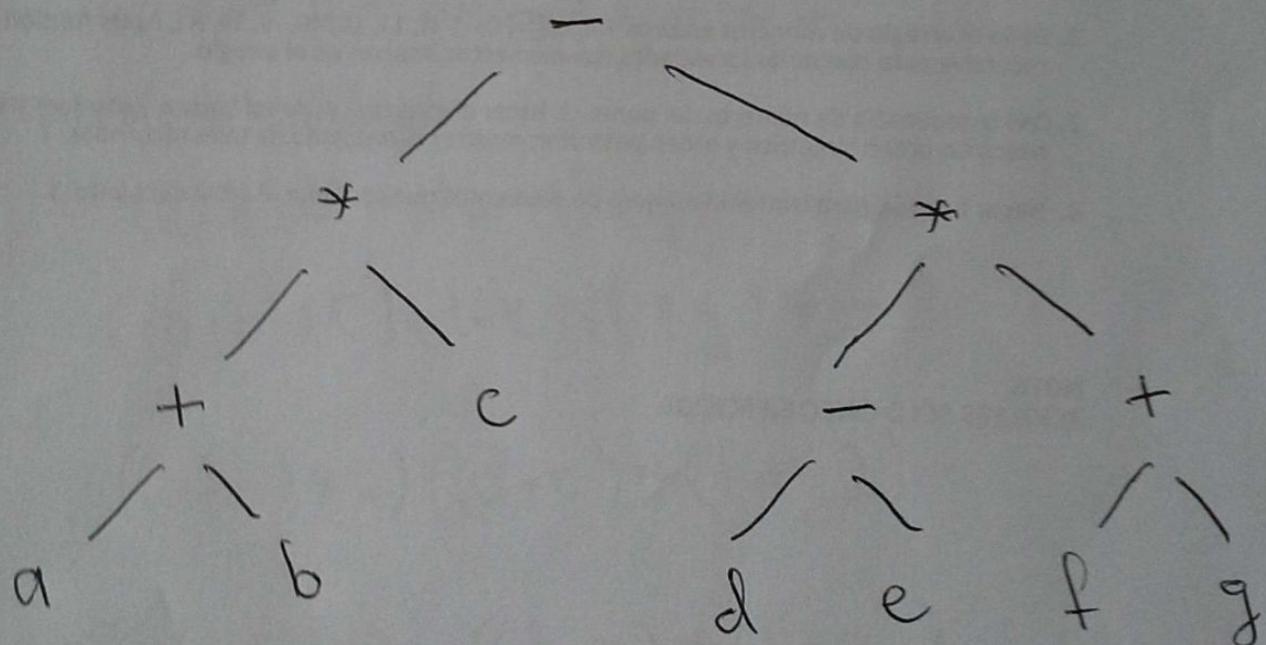
Ejercicio:

Con la expresión: $(a+b)*c - (d-e)*(f+g)$,
dibujar el esquema de árbol binario y efectuar
los recorridos en orden previo y orden poste-
rior, y en cada caso obtener la expresión
original.

Sol.

$$(a+b)*c - (d-e)*(f+g)$$
$$\downarrow$$
$$((a+b)*c) - ((d-e)*(f+g))$$

Así:



Continuación ...

Orden previo:

$$- * + abc * - de + fg$$

expresión original:

$$- * (a+b)c * (d-e)(f+g)$$

$$- ((a+b)*c)((d-e) * (f+g))$$

$$((a+b)*c) - ((d-e) * (f+g))$$

=====

Orden Posterior:

$$ab + c * de - fg + * -$$

expresión original:

$$(a+b)c * de - (f+g) * -$$

$$((a+b)*c)((d-e)(f+g)) * -$$

$$((a+b)*c) ((d-e) * (f+g)) -$$

$$((a+b)*c) - ((d-e) * (f+g))$$

=====

Implementación de árbol binario

```
#include<iostream.h>
class Nodo{
private: int num;
         Nodo *left, *right;
public: Nodo() {
            num = 0;
            left = right = NULL;
        }
        friend class Arbol;
    };
class Arbol{
private: Nodo *root;
public: Arbol() {
            root = NULL;
        }
        void insertaNodo(Nodo *&root, int x);
        void listar();
        Nodo *buscaElemento(int);
        Nodo *buscaMayor();
    };
}
```

```
void Arbol::insertaNodo(Nodo *&root, int x){  
    Nodo *aux;  
    if (root == NULL){  
        root = new Nodo;  
        root->Num = x;  
        root->left = root->right = NULL;  
    } else {  
        aux = new Nodo;  
        aux->Num = x;  
        aux->left = aux->right = NULL;  
        if (x < aux->Num)  
            insertaNodo(root->left, x);  
        else  
            insertaNodo(root->right, x);  
    }  
}
```

lista Arbol.

```
void Arbol:: listar() {
    Nodo *aux = root;
    if (aux == NULL) {
        cout << "arbol vacio.";
        exit(1);
    }
    if (aux != NULL) {
        while (aux->left != NULL) {
            cout << aux->num;
            aux = aux->left;
        }
        while (aux->right != NULL) {
            cout << aux->num;
            aux = aux->right;
        }
    }
}
```

listar Arbol.

```
Void Arbol:: listar() {
    Nodo *aux=root;
    if(aux==NULL){
        cout<<"arbol vacio.";
        exit(1);
    }
    if(aux!=NULL){
        while(aux->left!=NULL){
            cout<< aux->num;
            aux=aux->left;
        }
        while(aux->right!=NULL){
            cout<< aux->num;
            aux=aux->right;
        }
    }
}
```

busca elemento de forma iterativa.

```
Nodo *Arbol:: buscaElemento(int x){  
    Nodo *aux=root;  
    if(aux==NULL)  
        return NULL;  
    if(x<aux->num)  
        while(aux->left!=NULL)  
            aux=aux->left;  
    else if(x>aux->num)  
        while(aux->right!=NULL)  
            aux=aux->right;  
    return aux;  
}
```

encuentra Mayor en forma iterativa.

Node *Arbol :: buscaMayor(D){

if(root!=NULL)

while(root->Right!=NULL)

root=root->Right;

return root;

}

cuenta elementos impares

```
int Arbol:: cuentaImpar() {
    if (root == NULL)
        return 0;
    if (root != NULL)
        while (root->left != NULL) {
            if (root->num % 2 != 0)
                conta++;
            root = root->left;
        }
        while (root->right != NULL) {
            if (root->num % 2 != 0)
                conta++;
            root = root->right;
        }
    contador = conta + contaL;
}
return contador;
```