

# Recorridos en Árboles Binarios

Ing. Armandina Leal Flores

# Recorridos

- ▶ Recorrer el árbol es “pasar por” o “visitar” todos los nodos del mismo.
- ▶ Recorridos típicos:
  - Preorden
  - Inorden
  - Postorden
- ▶ Otro tipo de recorridos
  - Conversos
  - Nivel por nivel

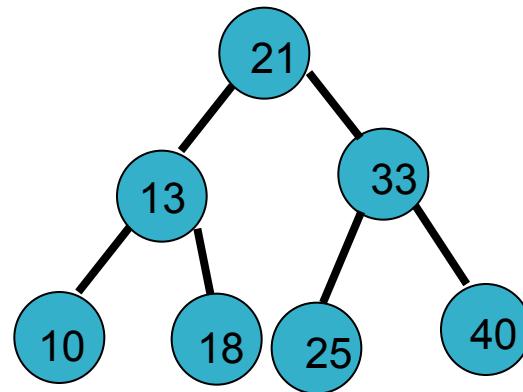
Los recorridos se aplican sobre un **Árbol Binario** NO TIENE que ser ABB

# Recorrido Preorden

## ▶ Proceso:

- Visita el nodo raíz del árbol.
- Recorre el preorden el subárbol izquierdo del nodo raíz.
- Recorre el preorden el subárbol derecho del nodo raíz.

## ▶ Aplicación: Generar una réplica del árbol.



**Recorrido en Preorden**

21, 13, 10, 18, 33, 25, 40

# Implementación del Preorden

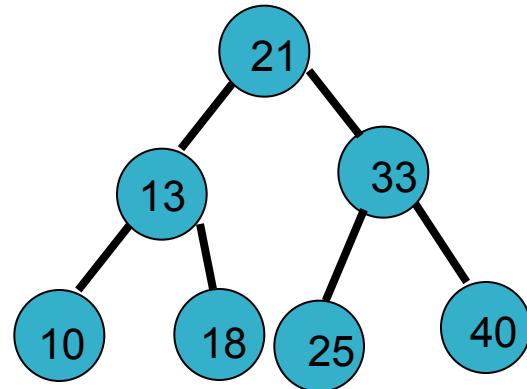
```
void Preorden()
{
    despliegaPreorden(pRaiz);
    cout << endl;
}
void despliegaPreorden ( NodoArbol *pR)
{
    if ( pR != NULL)
    {
        cout << pR -> iInfo;
        despliegaPreorden ( pR -> plzq );
        despliegaPreorden ( pR -> pDer );
    }
}
```

# Recorrido Inorden

## ▶ Proceso:

- Recorre en inorden el subárbol izquierdo.
- Visita la raíz del árbol.
- Recorre en inorden el subárbol derecho.

## ▶ Aplicación: Desplegar en orden creciente los elementos del árbol si este es un ABB.



**Recorrido en Inorden**

10, 13, 18, 21, 25, 33, 40

# Implementación del Inorden

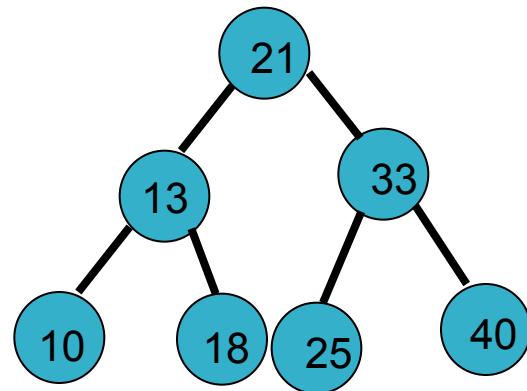
```
void Inorden()
{
    despliegalnorden(pRaiz);
    cout << endl;
}
void despliegalnorden ( NodoArbol *pR)
{
    if ( pR != NULL)
        {
            despliegalnorden ( pR -> plzq );
            cout << pR -> iInfo;
            despliegalnorden ( pR -> pDer );
        }
}
```

# Recorrido Postorden

## ▶ Proceso:

- Recorre en postorden el subárbol izquierdo.
- Recorre en postorden el subárbol derecho.
- Visita la raíz del árbol.

## ▶ Aplicación: Liberar los nodos de un árbol.



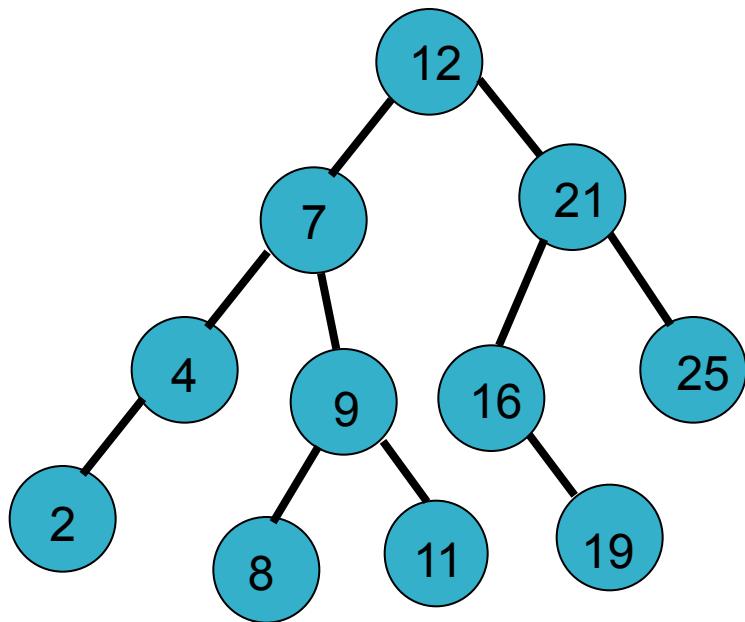
**Recorrido en Postorden**

10, 18, 13, 25, 40, 33, 21

# Implementación Postorden

```
void Postorden()
{
    despliegaPostorden(pRaiz);
    cout << endl;
}
void despliegaPostorden ( NodoArbol *pR)
{
    if ( pR != NULL)
    {
        despliegaPostorden ( pR -> plzq );
        despliegaPostorden ( pR -> pDer );
        cout << pR -> info;
    }
}
```

# Ejemplo....



**Recorrido en Preorden**

12, 7, 4, 2, 9, 8, 11, 21, 16, 19, 25

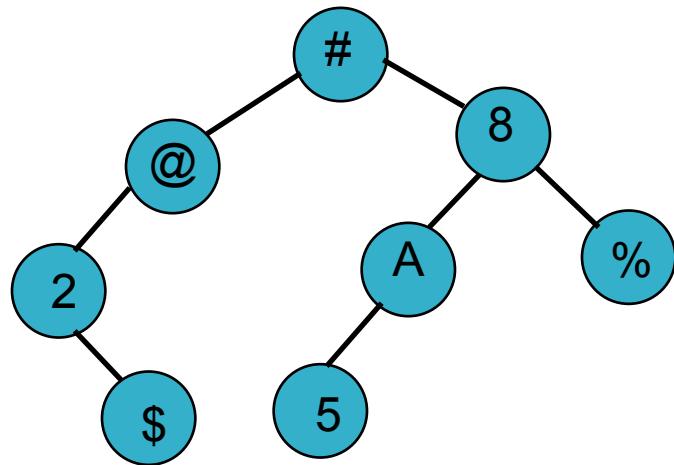
**Recorrido en Inorden**

2, 4, 7, 8, 9, 11, 12, 16, 19, 21, 25

**Recorrido en Postorden**

2, 4, 8, 11, 9, 7, 19, 16, 25, 21, 12

# Ejemplo....



**Recorrido en Preorden**

#, @, 2, \$, 8, A, 5, %

**Recorrido en Inorden**

2, \$, @, #, 5, A, 8, %

**Recorrido en Postorden**

\$, 2, @, 5, A, %, 8, #

# Dado 2 recorridos construir el árbol

Dado los siguientes recorridos:

**Recorrido en Preorden**

\$, %, A, &, #

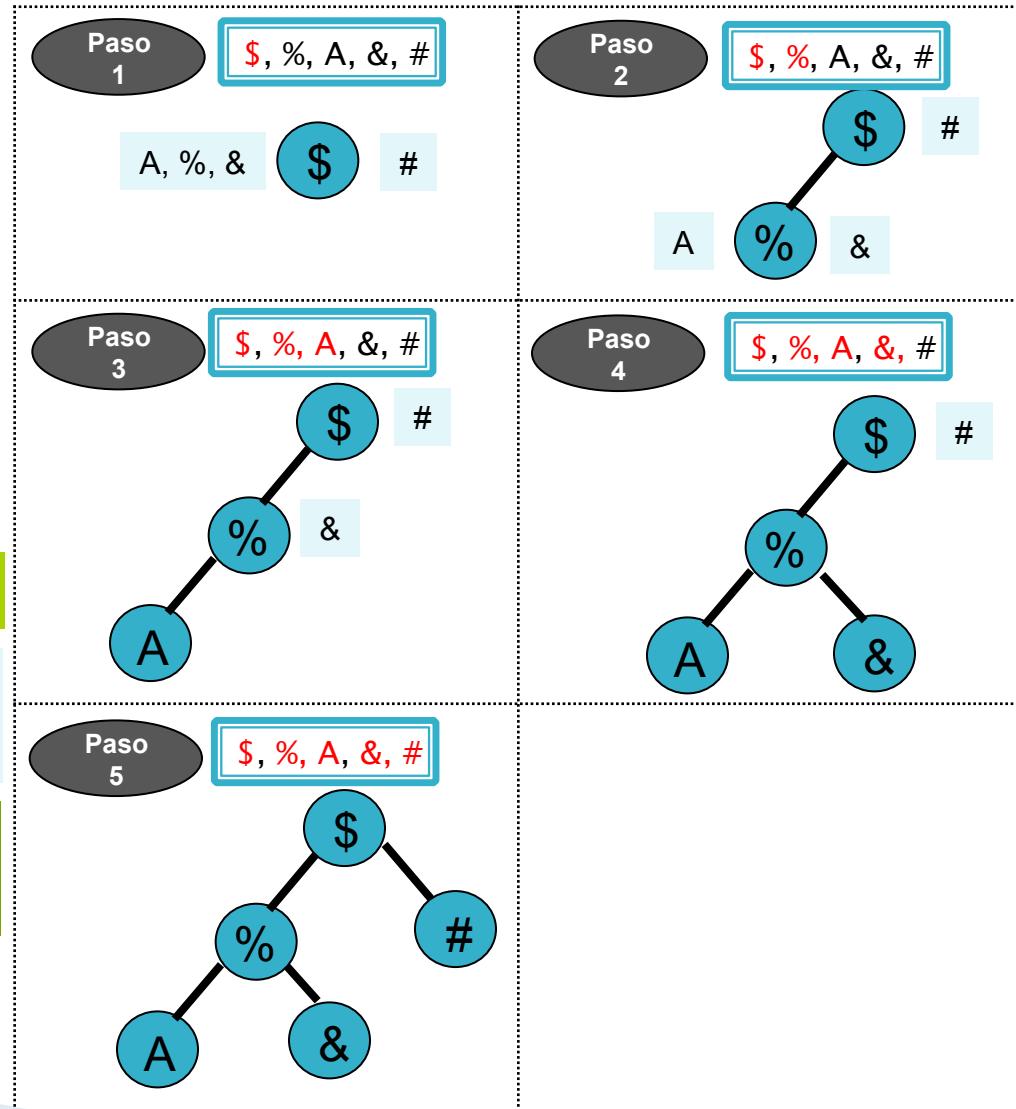
**Recorrido en Inorden**

A, %, &, \$, #

El **Preorden** indica que la raíz es: \$

El **Inorden** indica quién está a la izquierda y quién a la derecha

El **Preorden** también indica cuál es el siguiente valor a procesar



# Dado 2 recorridos construir el árbol

Dado los siguientes recorridos:

**Recorrido en Postorden**

A, &, %, #, \$

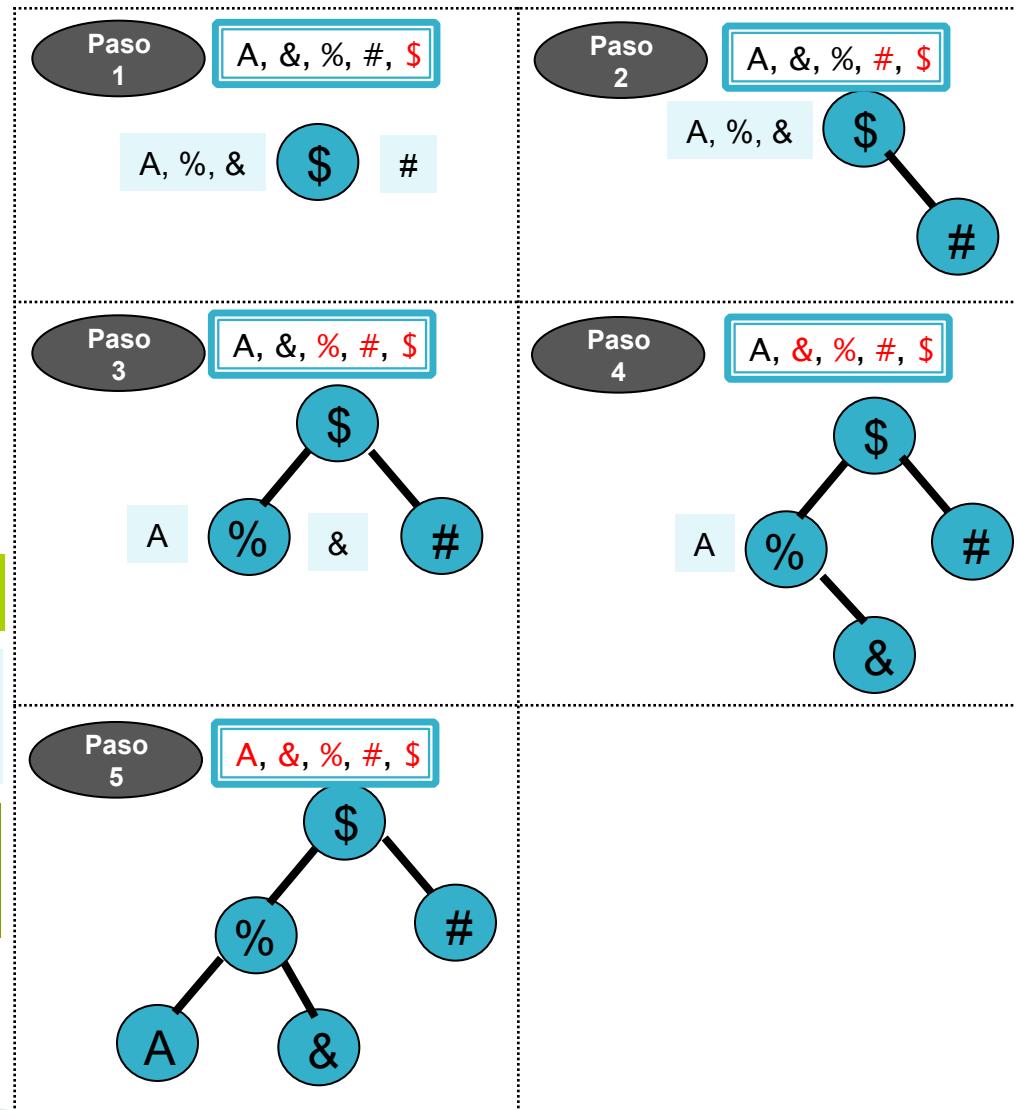
**Recorrido en Inorden**

A, %, &, \$, #

El **Postorden** indica que la raíz es: \$

El **Inorden** indica quién está a la izquierda y quién a la derecha

El **Postorden** también indica cuál es el siguiente valor a procesar

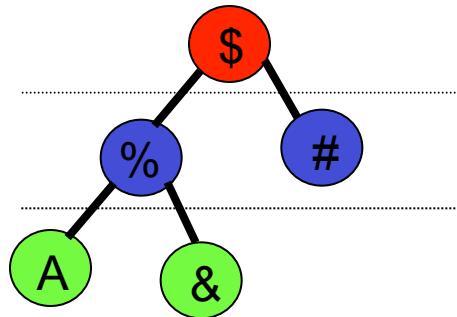


# Recorridos conversos

- ▶ Visitan los nodos en orden contrario es decir, primero a la derecha y después a la izquierda.
- ▶ Recorridos conversos:
  - INORDEN CONVERSO
  - PREORDEN CONVERSO
  - POSTORDEN CONVERSO
- ▶ No es muy común encontrar una aplicación para ellos.

# Recorridos Nivel por Nivel

- ▶ Recorre el árbol en forma horizontal.



**Recorrido Nivel por Nivel**

\$, %, #, A, &

# Implementación del Recorrido Nivel por Nivel

- ▶ Para implementar este recorrido se utiliza una Fila que almacena apuntadores a los nodos del árbol.
- ▶ Proceso:
  - Meter el apuntador al nodo raíz a una Fila.
  - Mientras la Fila no se vacíe:
    - Sacar un apuntador de la Fila y procesar el nodo apuntado.
    - Meter a la Fila los apuntadores a los hijos del nodo procesado (si éstos existen).