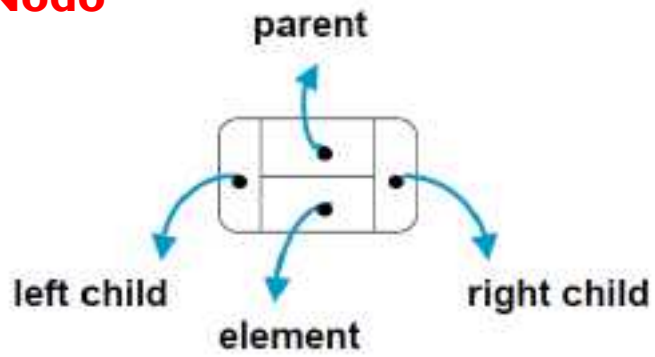
# Tema 5. Árboles
# **Implementando árboles binarios**

## Estructura de Datos y Algoritmos (EDA)

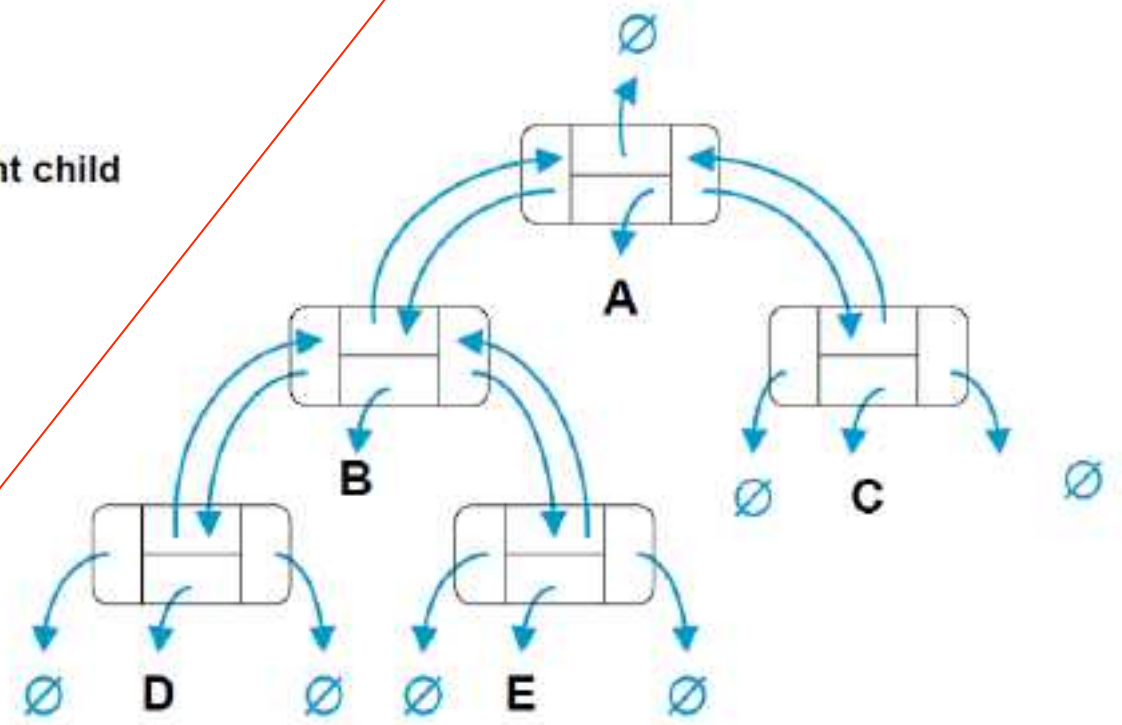
Material elaborado por los profesores de la asignatura

# Árboles binarios: Implementación

**Nodo**




**Árbol**



# Implementando árboles binarios: BinTreeNode class (parámetros y constructor)

```
public class BinTreeNode {
    Object elem;

    BinTreeNode parent;
    BinTreeNode left;
    BinTreeNode right;

    public BinTreeNode(Object element) {
        elem = element;
    }
```

# Implementando árboles binarios: BinTreeNode class (método getSize)

```
public int getSize() {
    return getSize(this);
}

static int getSize(BinTreeNode subtree) {
    if (subtree == null) {
        return 0;
    } else {
        int result = 1 + getSize(subtree.left) + getSize(subtree.right);
        return result;
    }
}
```

# Implementando árboles binarios: BinTreeNode class (método getHeight)

- El método **getHeight** que devuelve la altura de un nodo.

```
public static int getHeight(BinTreeNode node) {
    if (node == null) {
        return 0;
    } else {
        int result = 1 + Math.max(getHeight(node.left), getHeight(node.right));
        return result;
    }
}
```

# Implementando árboles binarios: BinTreeNode class (método **getDepth**)

- El método getDepth que devuelve la profundidad de un nodo.

```
public static int getDepth(BinTreeNode node) {
    if (node==null) return -1;
    else return 1 + getDepth(node.parent);
}

public static int getDepthIt(BinTreeNode node) {
    if (node==null) return -1;
    BinTreeNode nodeIt=node;
    int level=0;
    while (nodeIt.parent!=null) {
        nodeIt=nodeIt.parent;
        level++;
    }
    return level;
}
```

# Implementando árboles binarios: BinTreeNode class (método getPreorder)

```
public SList getPreorder() {
        SList list = new SList();
        getPreorder(root, list);
        return list;
    }


public static void getPreorder(BinTreeNode node,
                                        SList list) {
        if (node == null) return;
        list.addLast(node.elem);
        getPreorder(node.left, list);
        getPreorder(node.right, list);
    }
```

# Implementando árboles binarios: BinTreeNode class (método getPostOrder)

```
public SList getPostOrder() {
    SList list = new SList();
    getPostOrder(root, list);
    return list;
}
```

```
public  static void getPostOrder(BinTreeNode node, SList list) {
    if (node == null) return;

    getPostOrder(node.left, list);
    getPostOrder(node.right, list);

    list.addLast(node.elem);
    //System.out.println(node.elem)

}
```

# Implementando árboles binarios: BinTreeNode class (método getInOrder)

```
public  SList getInOrder() {
    SList list = new SList();
    getInOrder(root, list);
    return list;
}
```

```
public static void getInOrder(BinTreeNode node, SList list) {
    if (node == null) return;

    getInOrder(node.left, list);
    list.addLast(node.elem);
    //System.out.println(node.elem)

    getInOrder(node.right, list);

}
```

# Implementando árboles binarios: Interfaz IBinTree

```
public interface IBinTree {
    public int getSize();
    public int getHeight();
    public SList getPreorder();
    public SList getPostOrder();
    public SList getInOrder();
}
```