

ARBOL BINARIO

```
import ArbolBinarioCompleto

fun main() {
    println("=== arbol binario completo ===")

    var niveles: Int? = null
    while (niveles == null || niveles <= 0) {
        println("dame los niveles del arbol: ")
        niveles = readLine()?.toIntOrNull()
        if (niveles == null || niveles <= 0) {
            println("entrada invalida ingresa un numero mayor que 0")
        }
    }

    val totalNodos = (1 shl niveles) - 1
    val valores = mutableListOf<Int>()
    println("ingresa $totalNodos valores da enter para ingresar el siguiente valor")

    while (valores.size < totalNodos) {
        print("Valor ${valores.size + 1}: ")
        val valor = readLine()?.toIntOrNull()
        if (valor != null) {
            valores.add(valor)
        } else {
            println("valor invalido intenta nuevamente")
        }
    }
}
```

```
fun main() {
    while (valores.size < totalNodos) {
        print("Valor ${valores.size + 1}: ")
        val valor = readLine()?.toIntOrNull()
        if (valor != null) {
            valores.add(valor)
        } else {
            println("valor invalido intenta nuevamente")
        }
    }

    val arbol = ArbolBinarioCompleto()
    arbol.construirDesdeLista(valores)

    println("\nRecorrido en inorden:")
    arbol.recorridoInOrden()

    println("\nRecorrido preorden:")
    arbol.recorridoPreOrden()

    println("\nRecorrido postorden:")
    arbol.recorridoPostOrden()
}
```

ARBOL BINARIO COMPLETO

```
import java.util.LinkedList
import java.util.Queue

class ArbolBinarioCompleto {
    private var raiz: NodoArbol? = null

    fun construirDesdeLista(valores: List<Int>) {
        if (valores.isEmpty()) return

        raiz = NodoArbol( valor = valores[0])
        val cola: Queue<NodoArbol> = LinkedList()
        cola.add(raiz)

        var i = 1
        while (i < valores.size) {
            val actual = cola.poll()
            if (i < valores.size) {
                actual.izquierdo = NodoArbol( valor = valores[i++])
                cola.add(actual.izquierdo!!)
            }
            if (i < valores.size) {
                actual.derecho = NodoArbol( valor = valores[i++])
                cola.add(actual.derecho!!)
            }
        }
    }

    fun recorridoInOrden() {
```

```
class ArbolBinarioCompleto {
    fun recorridoInOrden() {
        inOrden( nodo = raiz)
        println()
    }

    private fun inOrden(nodo: NodoArbol?) {
        if (nodo != null) {
            inOrden( nodo = nodo.izquierdo)
            print("${nodo.valor} ")
            inOrden( nodo = nodo.derecho)
        }
    }

    fun recorridoPreOrden() {
        preOrden( nodo = raiz)
        println()
    }

    private fun preOrden(nodo: NodoArbol?) {
        if (nodo != null) {
            print("${nodo.valor} ")
            preOrden( nodo = nodo.izquierdo)
            preOrden( nodo = nodo.derecho)
        }
    }
}
```

The screenshot shows an IDE with the file `ArbolBinarioCompleto.kt` open. The code implements a binary tree structure with the following methods:

```
class ArbolBinarioCompleto {  
    private fun preOrden(nodo: NodoArbol?) {  
        if (nodo != null) {  
            print("${nodo.valor} ")  
            preOrden(nodo = nodo.izquierdo)  
            preOrden(nodo = nodo.derecho)  
        }  
    }  
  
    fun recorridoPostOrden() {  
        postOrden(nodo = raiz)  
        println()  
    }  
  
    private fun postOrden(nodo: NodoArbol?) {  
        if (nodo != null) {  
            postOrden(nodo = nodo.izquierdo)  
            postOrden(nodo = nodo.derecho)  
            print("${nodo.valor} ")  
        }  
    }  
}
```

The IDE interface includes a sidebar with file explorer, a top bar with project settings, and a bottom status bar showing system icons and the time 10:30 AM on 7/9/2025.

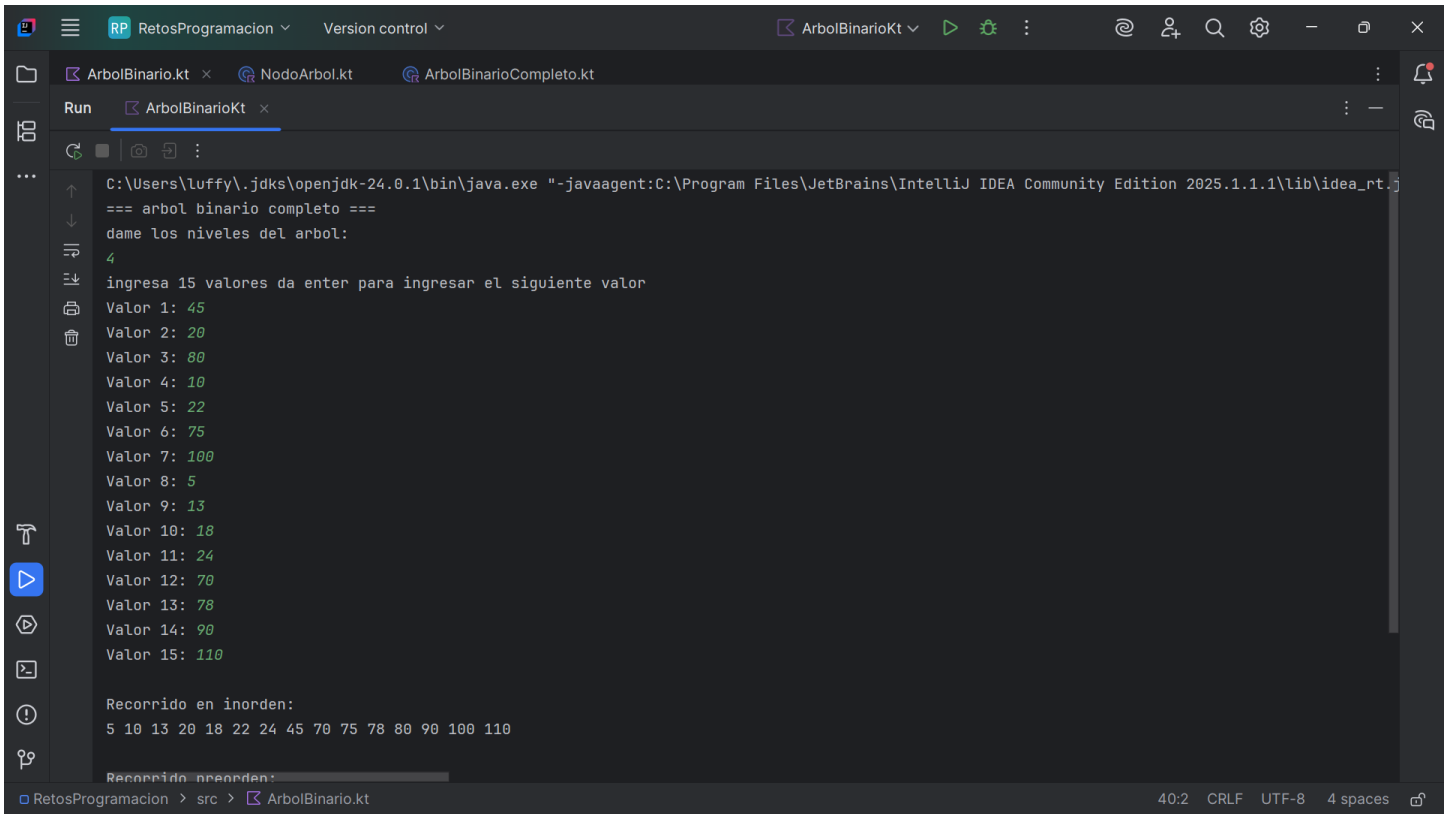
NODO ARBOL

The screenshot shows the same IDE with the file `NodoArbol.kt` open. It defines the `NodoArbol` class with the following structure:

```
class NodoArbol(val valor: Int) {  
    var izquierdo: NodoArbol? = null  
    var derecho: NodoArbol? = null  
}
```

The IDE interface is consistent with the previous screenshot, showing the same top bar and bottom status bar.

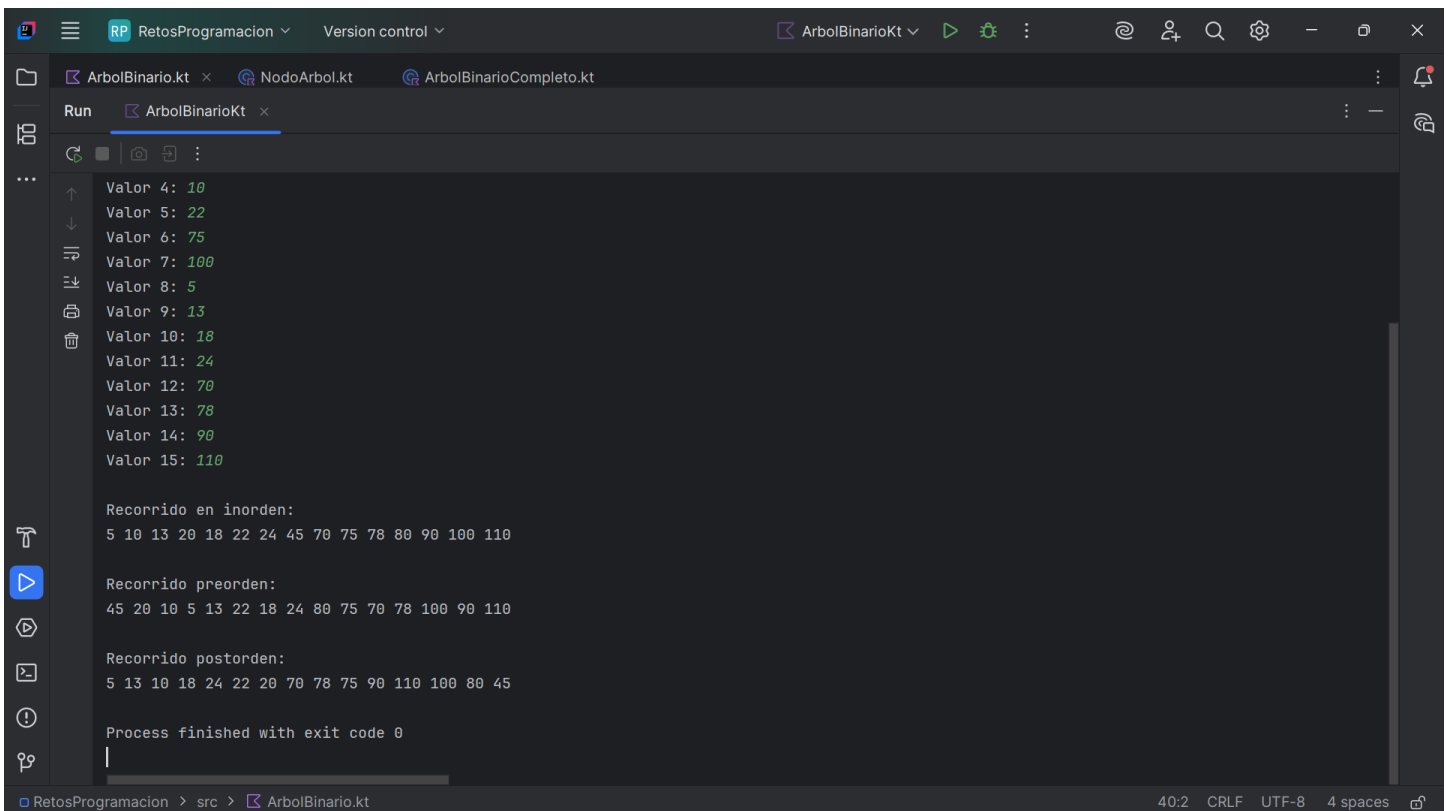
CONSOLA



```
C:\Users\luffy\.jdk\openjdk-24.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.1.1\lib\idea_rt.jar"
=== arbol binario completo ===
dame los niveles del arbol:
4
ingresa 15 valores da enter para ingresar el siguiente valor
Valor 1: 45
Valor 2: 20
Valor 3: 80
Valor 4: 10
Valor 5: 22
Valor 6: 75
Valor 7: 100
Valor 8: 5
Valor 9: 13
Valor 10: 18
Valor 11: 24
Valor 12: 70
Valor 13: 78
Valor 14: 90
Valor 15: 110

Recorrido en inorden:
5 10 13 20 18 22 24 45 70 75 78 80 90 100 110

Recorrido preorden:
```



```
Valor 4: 10
Valor 5: 22
Valor 6: 75
Valor 7: 100
Valor 8: 5
Valor 9: 13
Valor 10: 18
Valor 11: 24
Valor 12: 70
Valor 13: 78
Valor 14: 90
Valor 15: 110

Recorrido en inorden:
5 10 13 20 18 22 24 45 70 75 78 80 90 100 110

Recorrido preorden:
45 20 10 5 13 22 18 24 80 75 70 78 100 90 110

Recorrido postorden:
5 13 10 18 24 22 20 70 78 75 90 110 100 80 45

Process finished with exit code 0
```