Ejercicios de Árboles

Información del Proyecto

Descripción	Detalles
Profesores	Sergio Cavero y Salvador Sanchez
Asignatura	Estructuras de Datos
Universidad	Universidad Rey Juan Carlos
Curso	2024/2025

Ejercicio 1: operaciones simples sobre árboles binarios

En estos ejercicios vamos a trabajar con árboles binarios desde un punto de vista general, es decir, no vamos a centrarnos en las operaciones de inserción, eliminación o búsqueda ya que dependen de la estructura del árbol. En su lugar, nos centraremos en la los recorridos y en la creación de nuevas funciones que nos permitan trabajar con los árboles binarios.

Para realizar estos ejercicios, vamos a utilizar un árbol binario con información entera para cada nodo. Para ello, utilizaremos el siguiente código base:

- `arboles_ej1.pas`: programa principal que ejecutaremos para comprobar que hemos implementado correctamente los ejercicios.
- `uBinaryTree.pas`: unidad que contiene la definición del árbol binario. Deberemos implementar el recorrido en esta unidad.

Ejercicio 1.0: crear el árbol binario

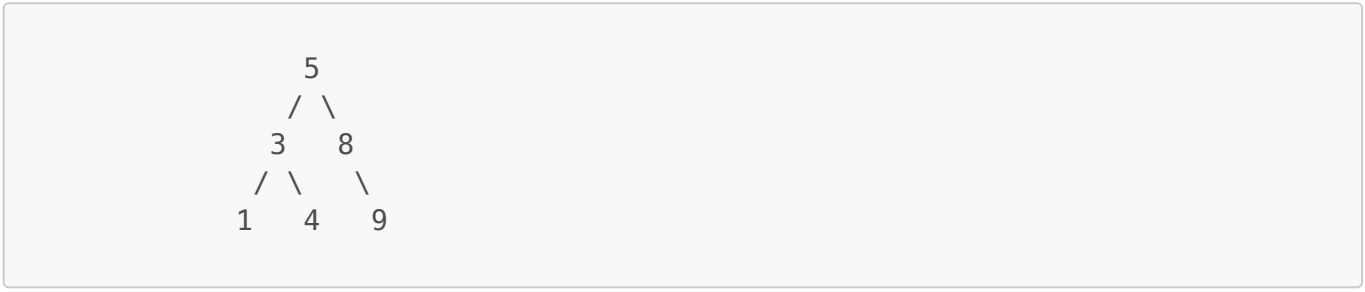
En el fichero `arboles_ej1.pas` hay dos funciones: `procedure crear_arbol(var a: tbinaryTree);` y `procedure crear_arbol2(var a: tbinaryTree);`. Debes implementar estos

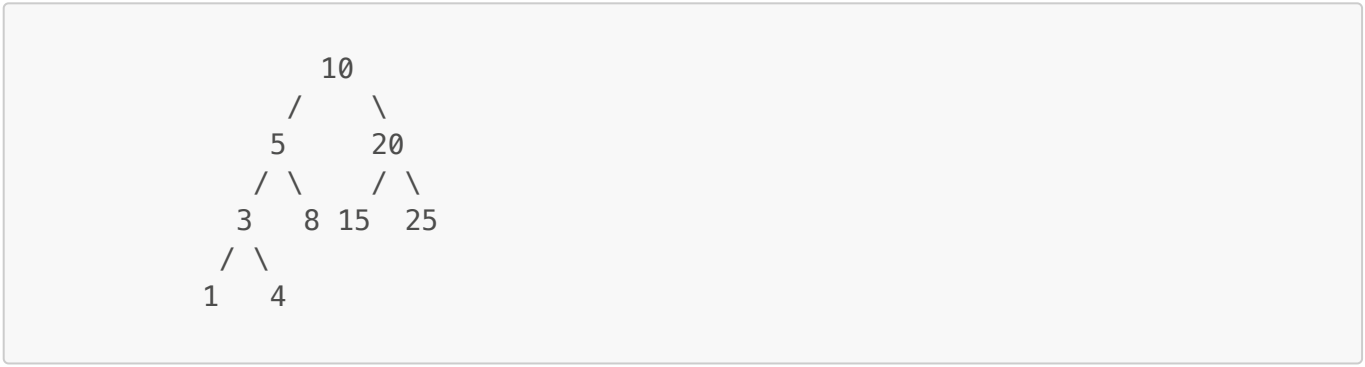
métodos suponiendo que el método `insertar` (`add`) realiza la inserción de un nodo en el árbol binario. Concretamente, la inserción se realiza como en un árbol binario de búsqueda, es decir:

- Si el árbol está vacío, se inserta el nodo como raíz.
- Si el árbol no está vacío, se compara el valor del nodo a insertar con el valor del nodo raíz. Si el valor es menor, se inserta en la subárbol izquierdo. Si el valor es mayor, se inserta en el subárbol derecho.

Suponemos para este ejercicio que el árbol binario no tiene nodos duplicados.

Los árboles a crear son los siguientes:





Ejercicio 1.1: in-order inverso

Un recorrido in-orden es un recorrido de un árbol binario en el que se visitan primero los nodos de la izquierda de un nodo, luego el propio nodo (si pensamos en el árbol completo, la raíz) y finalmente los nodos de la derecha. En este ejercicio vamos a implementar un recorrido in-orden inverso, es decir, primero se visitan los nodos de la derecha, luego el nodo raíz y finalmente los nodos de la izquierda.

Ejercicio 1.2: profundidad máxima

Implementar una función que devuelva la profundidad máxima de un árbol binario. La profundidad de un árbol binario es la longitud del camino más largo desde la raíz hasta una hoja. Como sabes, un nodo hoja es un nodo que no tiene hijos.

Ejercicio 1.3: contar nodos

Implementar una función que cuente el número de nodos de un árbol binario. La función debe devolver el número total de nodos del árbol, sean estos hojas o nodos intermedios.

Ejercicio 1.4: contar hojas

Implementar una función que cuente el número de nodos hoja de un árbol binario. Ya sabes que un nodo hoja es aquél que no tiene hijos. La función debe devolver el número total de hojas del árbol.

Ejercicio 1.5: contar nodos internos

Implementar una función que cuente el número de nodos internos de un árbol binario. Un nodo interno es un nodo que tiene al menos un hijo. La función debe devolver el número total de nodos internos del árbol.

Ejercicio 1.6: es completo

Implementar una función que determine si un árbol binario es completo. Un árbol binario es completo si todos sus niveles están completamente llenos. Es decir, si todos los nodos tienen dos hijos, excepto las hojas que no tendrían hijos. La función debe devolver verdadero si el árbol es completo y falso en caso contrario. Cabe mencionar que se espera que el árbol tenga al menos un nodo.

Ejercicio 1.7: el mayor valor de una hoja

Implementar una función que devuelva el mayor valor de una hoja de un árbol binario. La función debe devolver el valor máximo de las hojas del árbol. En caso de que el árbol no tenga ningún nodo, la función debe devolver cero.

Ejercicio 1.8: suma del valor de las hojas

Implementar una función que devuelva la suma de los valores de las hojas de un árbol binario. Si el árbol es vacío la función debe devolver cero.

Ejercicio 1.9: cuenta número de nodos pares

Implementar una función que cuente el número de nodos pares de un árbol binario. Un nodo es par si su valor es par. La función debe devolver el número total de nodos pares del árbol.

Ejercicio 1.10: contar nodos en un nivel

Implementar una función que cuente el número de nodos en un nivel determinado de un árbol binario. La raíz del árbol se considera el nivel 0. Los hijos inmediatos de la raíz nivel 1, etc.