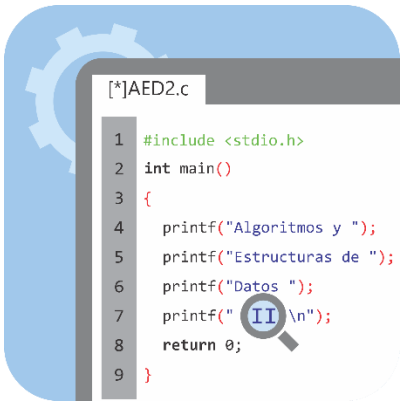


Algoritmos y Estructuras de Datos II

Trabajo Práctico 4 - Parte 1
Árboles

OBJETIVOS

- Conocer los conceptos de Árboles.
- Aprender a implementar este tipo de estructuras.
- Implementar soluciones recursivas para el tratamiento de estructuras de datos no lineales (árboles).

COMPETENCIAS

- Identificar, formular y resolver problemas mediante programación.
- Utilizar de manera efectiva técnicas y herramientas de aplicación para desarrollar software.
- Desempeñarse de manera efectiva en equipos de trabajo.
- Aprender en forma continua, autónoma y de manera colaborativa.

METODOLOGÍA

- El alumno deberá resolver individualmente los ejercicios propuestos.
- El alumno deberá codificar las soluciones en el lenguaje de programación C.
- Realizar consultas a través del canal de slack/discord correspondiente a su comisión ó del aula virtual de la asignatura.

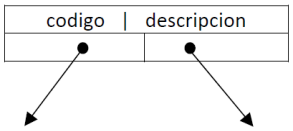
DURACIÓN

Según planificación de la asignatura se deberán utilizar para la resolución de los ejercicios de esta serie, dos (2) clases prácticas.

EJERCICIOS PROPUESTOS

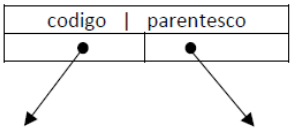
1. Diagramar el árbol de búsqueda binaria que corresponde en cada caso, si sus elementos son insertados de la siguiente manera:
   a) **54 -72 -25 -10 -15 -5**
   b) **74 -22 125 -10 -150 35**

2. Dado el siguiente nodo correspondiente a un árbol de búsqueda binaria:



Escribir en lenguaje c, la declaración del tipo de dato **tArbolTipoProductos** que corresponde al nodo dado, teniendo en cuenta que el árbol se balancea según el campo *código*.

3. Dado el siguiente nodo correspondiente a un árbol de búsqueda binaria:



Escribir en lenguaje c, la declaración del tipo de dato **tArbolTipoParientes** que corresponde al nodo dado, teniendo en cuenta que el árbol se balancea según el campo *código*.

4. Indicar el recorrido en pre-orden, in-orden y post-orden de los árboles de los ejercicios anteriores.

5. Escribir un programa que permita construir un árbol binario de búsqueda correspondiente a una lista de números enteros. Se debe considerar el nodo raíz del árbol, el primer elemento que se inserta en la lista. Los siguientes nodos serán descendientes derechos si son mayores, y descendientes izquierdos si son menores (tener en cuenta el orden de la lista dada). Además, programar funciones para el recorrido en pre-orden, in-orden y post-orden. Testear el programa, insertando los siguientes valores:

4 19 -7 49 100 0 22 12

6. Escribir un programa que permita crear un árbol binario de búsqueda, el contenido de los nodos debe ser de tipo real. Incluir los procedimientos para insertar, devolver el número de nodos y recorrer el árbol.
   a) El procedimiento *Insertar* debe emitir una leyenda *'Es un hijo izquierdo'* o *'Es un hijo derecho'*, según corresponda.
   b) Escribir la función que devuelva el número de nodos del árbol binario.
   c) El procedimiento *Recorrer* debe mostrar el contenido del nodo, siguiendo algún recorrido en profundidad.

7. Construir un árbol binario de búsqueda, donde el contenido de cada nodo sea un entero positivo, y que permita las siguientes acciones:
   a) Insertar nodos en el árbol.
   b) Saber si el árbol está vacío.
   c) Buscar un elemento en el árbol.
   d) Eliminar un nodo del árbol.

8. Construir un árbol binario de búsqueda, donde el contenido de cada nodo sea un valor de tipo **caracter**, y que permita las siguientes acciones:
   a) Insertar nodos en el árbol.
   b) Eliminar un nodo del árbol.
   c) Saber si el árbol está vacío.
   d) Una función que implemente la búsqueda binaria de un elemento del árbol y que devuelva ***true*** en caso de encontrarlo y ***false*** en caso contrario.
   e) Una función para concatenar los elementos del árbol mediante el acceso en **InOrden.**