Trabajo Práctico – Programación 1

Alumnos:

Elías Carulla – Carullaelias@gmail.com

Nicolás González– Nicomar94@hotmail.com

Materia: Programacion 1

Profesor: Ariel Enferrel, Prof. Cinthia Rigoni

Fecha de Entrega: 08 de junio de 2025

Índice

Introducción

En el presente trabajo se desarrolla un sistema de gestión de productos utilizando estructuras de datos en Python, específicamente árboles binarios de búsqueda (ABB). El objetivo principal es organizar un catálogo de productos de forma eficiente, permitiendo su inserción, búsqueda y visualización ordenada.

A través de un diseño modular y el uso de menús interactivos, se busca crear una solución funcional, comprensible y escalable que refleje los conceptos fundamentales de la programación estructurada y la manipulación de estructuras dinámicas. Este proyecto permite aplicar conocimientos teóricos sobre lógica de programación, estructuras condicionales, funciones, e interacción con el usuario por consola.

Marco Teórico

Árbol Binario de Búsqueda (ABB)

Un **árbol binario de búsqueda** es una estructura de datos jerárquica donde cada nodo contiene un valor único y dos subárboles: uno izquierdo y otro derecho. Cumple la propiedad de orden:

- Los valores en el subárbol izquierdo son menores que el nodo.
- Los valores en el subárbol derecho son mayores.

Esta estructura permite realizar operaciones como búsqueda, inserción y eliminación de forma eficiente, con una complejidad promedio de **O(log n)** en árboles balanceados.

Recorridos de un árbol

Los recorridos permiten visitar todos los nodos del árbol en cierto orden. Los más comunes son:

- **Preorden**: se visita el nodo actual, luego el subárbol izquierdo y finalmente el derecho.
- **Inorden**: se visita primero el subárbol izquierdo, luego el nodo y por último el derecho. Este recorrido devuelve los elementos en orden creciente.
- **Postorden**: se visitan primero los subárboles izquierdo y derecho, y luego el nodo actual.

Programación modular

La **programación modular** consiste en dividir el código en diferentes archivos y funciones para mejorar la legibilidad, reutilización y mantenimiento del software. En este trabajo se implementaron módulos separados para:

- La lógica del menú.

- La lógica del árbol.
- El flujo principal del programa.

Interacción por consola

La entrada y salida de datos se realiza por consola utilizando la función input() para recibir información del usuario, y print() para mostrar resultados. Esto permite un uso simple del sistema sin interfaz gráfica.

Caso Práctico

Este caso práctico consiste en desarrollar un programa en Python que permita gestionar productos mediante un **árbol binario de búsqueda (ABB)**. El objetivo principal es crear una estructura de datos eficiente que permita:

- Insertar productos.
- Buscar productos por código.
- Mostrar el catálogo en diferentes recorridos: Preorden, Postorden e Inorden.

Se implementó un menú interactivo para que el usuario pueda operar con el sistema de forma simple y ordenada.

Desarrollo

El proyecto está compuesto por cuatro archivos:

- main.py: Punto de entrada del programa.
- mainMenu.py: Control del flujo general y los submenús.
- menu.py: Módulo con funciones que muestran las opciones al usuario y reciben su entrada.
- operaciones.py: Contiene la lógica principal del árbol binario de búsqueda (insertar, buscar, mostrar catálogo).

Se utilizó un **árbol binario representado por listas anidadas**: [valor, subárbol_izquierdo, subárbol_derecho].

Funciones implementadas:

- insertar(arbol, valor): Inserta un nuevo producto en el ABB.
- buscar(arbol, valor): Busca un producto por su código.
- mostrarCatalogoPre, mostrarCatalogoIn, mostrarCatalogoPost: Muestran el catálogo según distintos recorridos (faltan completar las funciones Post e In en tu código).

Menús:

- Menú principal: insertar, buscar, mostrar, salir.
- Submenú de catálogo: permite elegir el tipo de recorrido.

Metodología Utilizada

- Se planificó un diseño modular usando varios archivos .py para separar funcionalidades.
- Se definió un árbol binario para organizar productos por su código de forma jerárquica.
- Se programaron las funciones para insertar, buscar y mostrar datos.
- Se implementaron menús interactivos para facilitar el uso del sistema.
- Se probó el programa con distintos valores para verificar que los recorridos y búsquedas fueran correctos.

Resultados Obtenidos

- El sistema permite **registrar productos** de forma eficiente sin duplicados.
- Se puede **buscar por código** y recibir retroalimentación inmediata.
- El catálogo puede mostrarse en distintos ordenamientos (aunque falta completar el Post e In en el código).
- La estructura con árboles permite escalar el sistema a un gran volumen de productos sin perder rendimiento.

Conclusiones

El uso de árboles binarios en Python es una forma eficiente de representar catálogos ordenados. Se logró separar correctamente la lógica del negocio de la interacción con el usuario, logrando un sistema modular y reutilizable.

Bibliografía

- **Python Docs – Data Structures**

https://docs.python.org/3/tutorial/datastructures.html

- **Programiz – Binary Search Tree (BST)**

https://www.programiz.com/dsa/binary-search-tree

- **Geeks for Geeks – Tree Traversals:**

https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/

- **Python Docs – Modules:**

https://docs.python.org/3/tutorial/modules.html

Anexos

Link del video explicativo

https://drive.google.com/file/d/1ejEPHEv-4RqGwIFr9HkE_kqD2AvcJ_Er/view