**[ABR] ESTRUCTURA DE DATOS**

**¿Cómo se puede aplicar un árbol binario de búsqueda para representar y consultar un árbol genealógico?**

**Estudiantes:**

**-Huamán Brañez, José Antonio**

**-García Quijada, Álvaro Ever**

**Curso:**

**ESTRUCTURA DE DATOS**

**2025**

1. **Descripción del Problema:**

¿Cómo se puede aplicar un árbol binario de búsqueda para representar y consultar un árbol genealógico?

En muchas familias, especialmente aquellas con historias extensas o estructuras complejas, es común perder la claridad sobre cómo se relacionan ciertos miembros entre sí. Con el paso del tiempo, recordar quién es descendiente de quién, o cómo se conectan distintas ramas familiares, puede volverse confuso y difícil de seguir.

Por ello, surge la necesidad de una aplicación informática que permita modelar y visualizar árboles genealógicos, de manera que se pueda agregar, eliminar y consultar fácilmente la información de los miembros de una familia. La herramienta debe permitir búsquedas rápidas y mostrar relaciones familiares de forma clara.

1. **Funcionalidades:**

* Una inserción de nuevos miembros según un criterio lógico como por ejemplo (fecha de nacimiento, ID, nombre alfabético, etc.).
* Una eliminación sin romper la estructura del árbol.
* Unas Consultas eficientes sobre ancestros, descendientes y ramas genealógicas.
* La Visualización en distintos recorridos.
* El Balanceo si el árbol pierde eficiencia.

1. **Preguntas:**

### **¿Qué información se debe almacenar en cada nodo del árbol?**

En este caso, cada nodo representaría a una persona, integrante de la familia. Por lo que los datos que tendría que concentrar serían.

* ID único
* Nombre del integrante
* Fecha de nacimiento.
* Padres
* Hijos

### **¿Cómo insertar y eliminar miembros del árbol sin romper su estructura?**

#### **Inserción:**

* Si el nuevo ID es menor al nodo actual, va a la izquierda.
* Si es mayor, va a la derecha.
* Recursivamente, hasta encontrar posición nula.

#### **Eliminación:**

* Caso 1: nodo hoja → se elimina directamente.
* Caso 2: nodo con un hijo → el hijo lo reemplaza.
* Caso 3: nodo con dos hijos → se reemplaza por el menor del subárbol derecho (sucesor inorden).

### **¿Qué métodos permiten recorrer el árbol para visualizar la genealogía?**

* **Inorden (izq - raíz - der)**: muestra miembros en orden creciente (por ID o nombre).
* **Preorden (raíz - izq - der)**: útil para la reconstrucción del árbol.
* **Postorden (izq - der - raíz)**: útil para eliminar el árbol.
* **Nivel por nivel (BFS)**: para visualizar por generaciones.

### **¿Cómo determinar si un miembro pertenece a una rama específica?**

* Definir la rama como un subárbol.
* Buscar el nodo raíz de la rama, luego buscar recursivamente dentro de ese subárbol si el miembro existe.

### **¿Cómo balancear el árbol si se vuelve demasiado profundo?**

El ABB puede volverse degenerado (como una lista). Para balancear:

* Convertirlo en un árbol que mantiene balance después de cada inserción/eliminación
* O recorrerlo inorden, almacenar en lista, y reconstruir un árbol balanceado desde la lista.

1. **Descripción de estructuras de datos y operaciones:**

**Componentes del Árbol (Persona):**

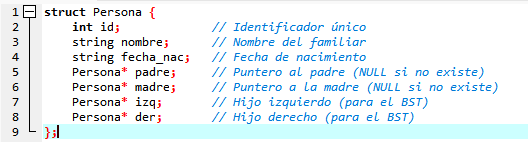
**Datos identificativos:**

* Campo numérico único (ID)
* Cadena de texto para el nombre completo
* Fecha de nacimiento en formato texto

**Vínculos ascendentes:**

* Referencia al nodo padre biológico
* Referencia al nodo madre biológica  
  (Ambas referencias pueden ser nulas)

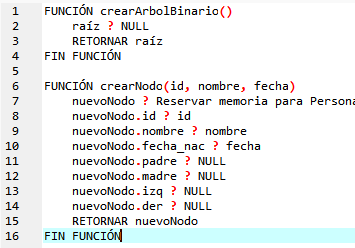
**Estructura de ordenamiento:**

* Nodo hijo menor (subárbol izquierdo)
* Nodo hijo mayor (subárbol derecho)

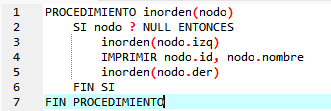
| **Operación** | **Descripción** |
| --- | --- |
| CrearPersona() | Crea un nuevo nodo con datos básicos. |
| Insertar() | Inserta un nodo en el BST según su id y establece relaciones familiares. |
| Buscar() | Busca un nodo por id usando recursividad. |
| Eliminar() | Elimina un nodo manteniendo la estructura BST (3 casos: hoja, 1 hijo, 2 hijos). |
| MostrarAncestros() | Muestra padres, abuelos, etc. (recursivo). |
| MostrarDescendientes () | Muestra hijos, nietos, etc. (simplificado). |
| BalancearArbol () | Reconstruye el árbol para optimizar búsquedas (usando recorrido inorden). |
| **Recorridos** | inorden (), preorden (), postorden (), porNiveles (). |

1. **Algoritmos Principales:**

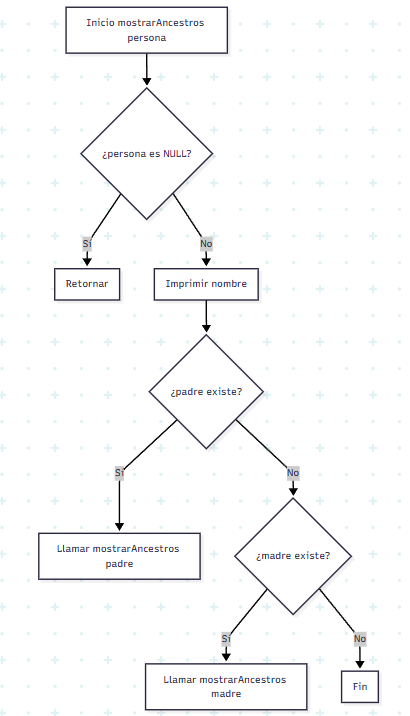
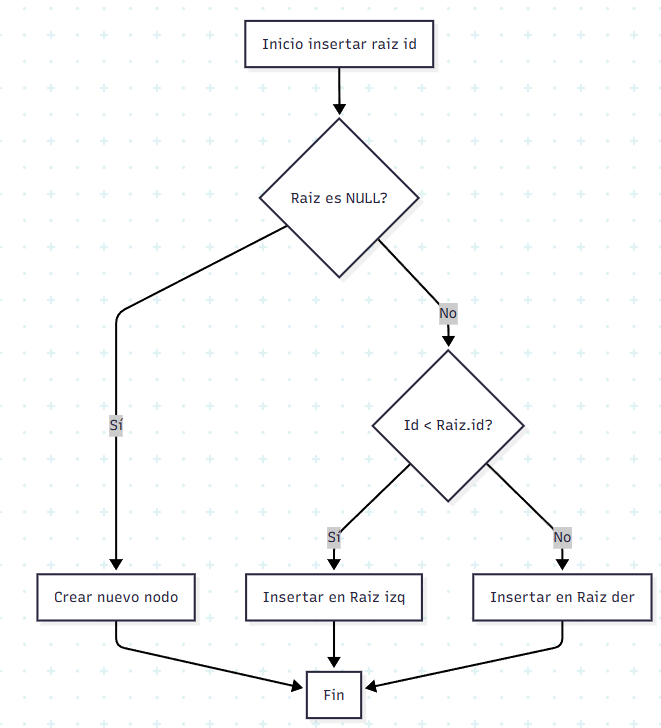
* **Pseudocodigo para la creación del Árbol:**

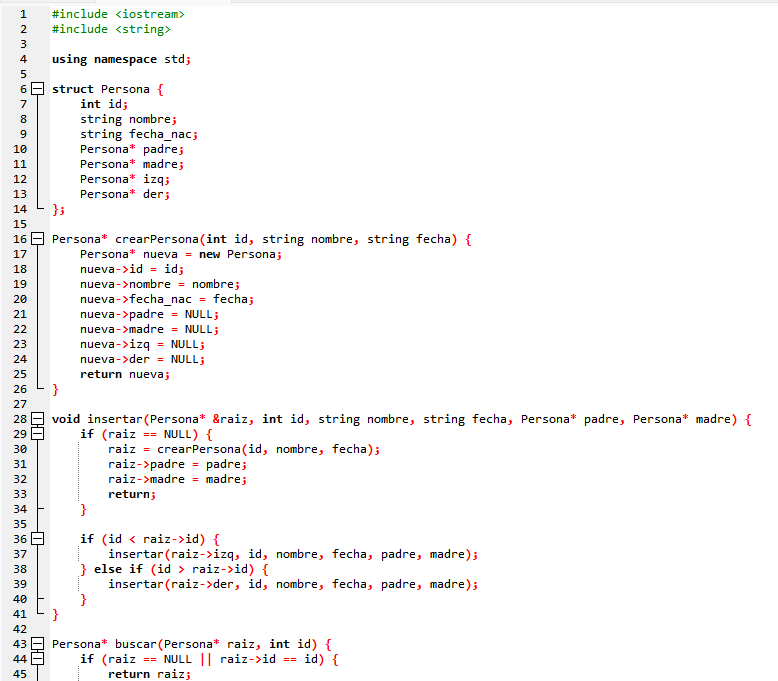
****

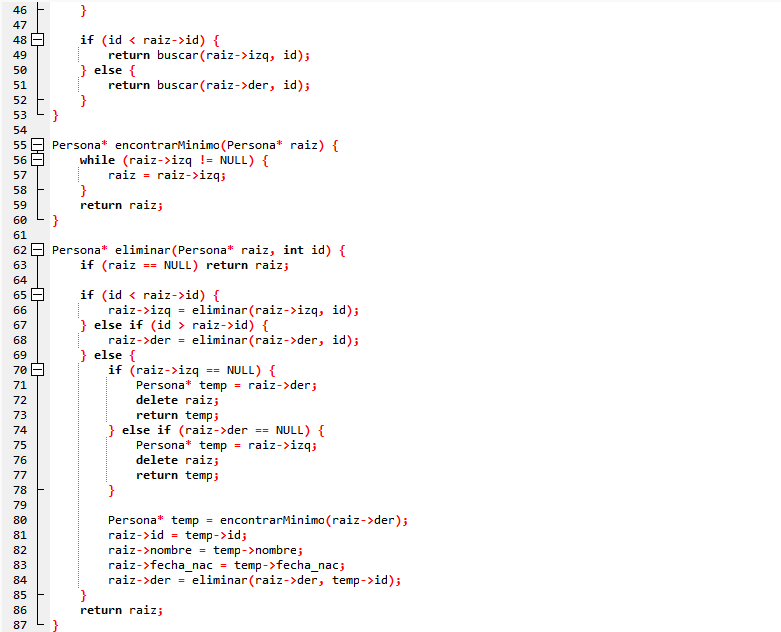
* **Pseudocódigo para recorrido inorden:**

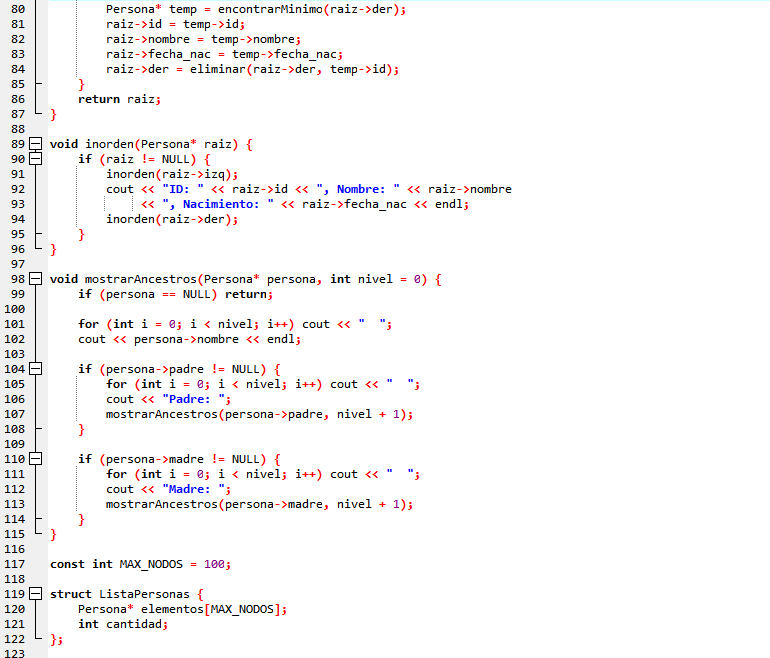
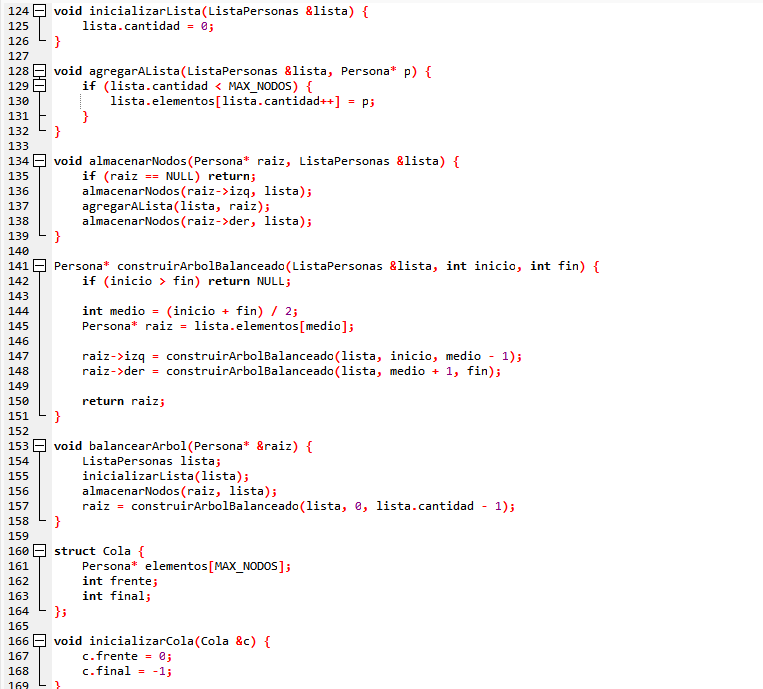
****

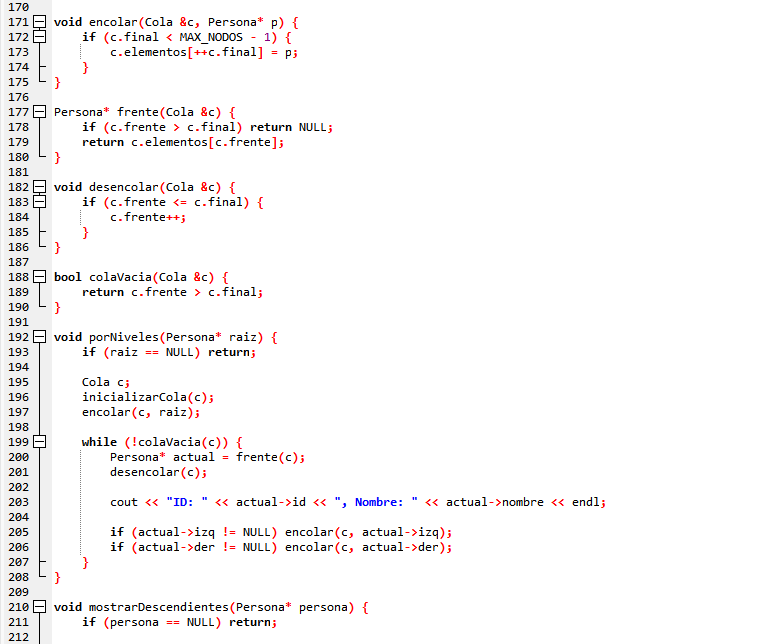
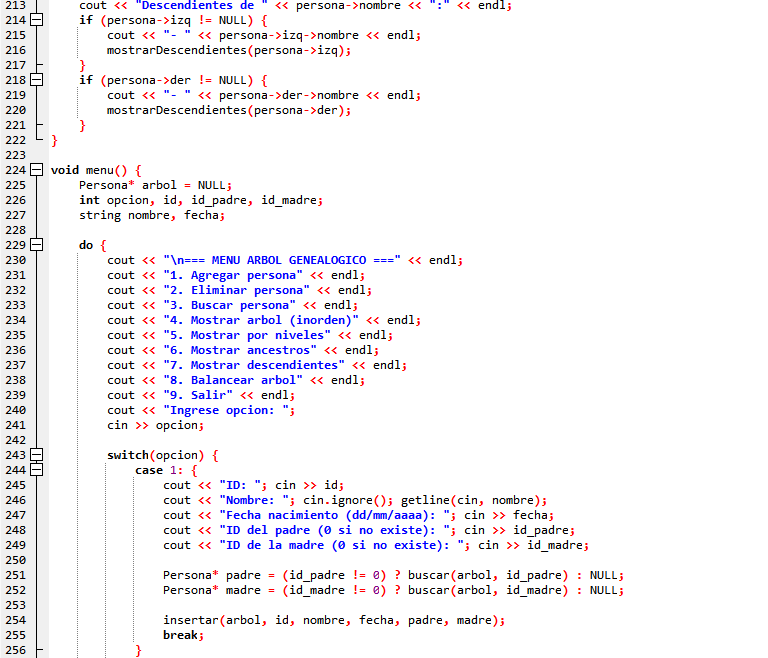
1. **Diagramas de Flujo:**

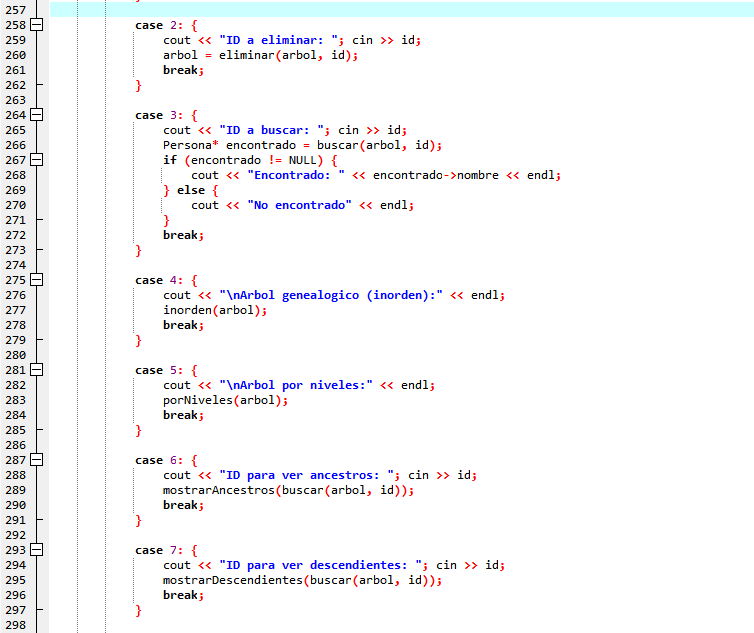
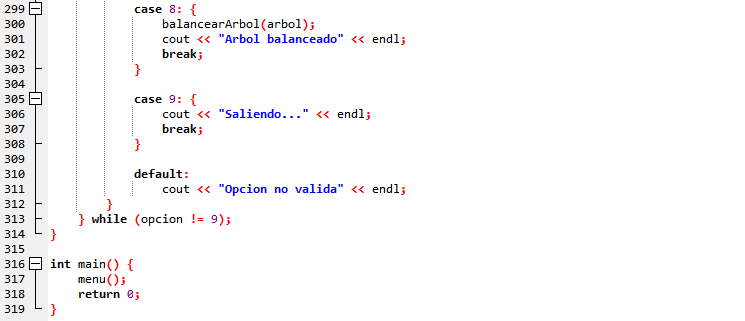
* **Diagrama Insertar Raíz y Búsqueda de ancestros:**

1. **Código Fuente:**

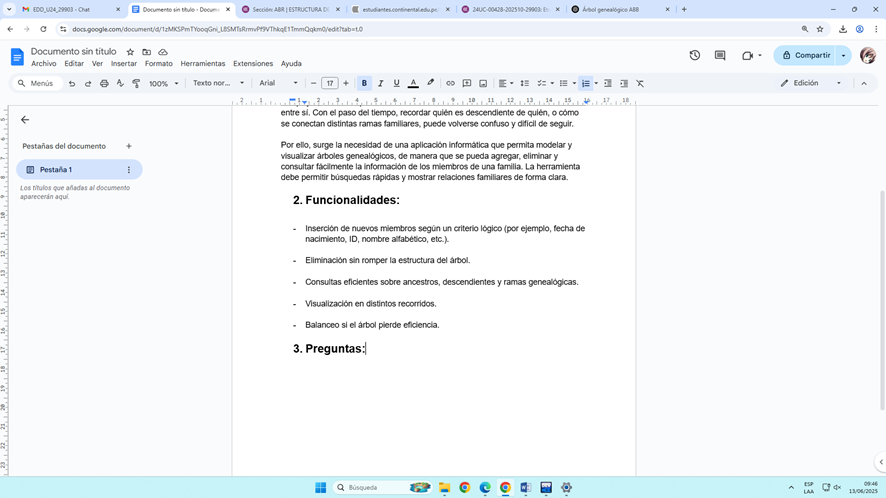








1. **Herramienta colaborativa:**

****