SEGUNDO AVANCE SEMANA 14

Curso: Estructura de Datos

Docente: Yesenia Concha Ramos

Integrantes:

- Jhon Gustavo Ccarita Velasquez
- Rodrigo Sevillanos Tinco
- Andre Sebastian Espinoza Zea

Tema: ABR

NRC: 59008

2025-02

Cusco - Perú

CAPÍTULO 1 - FASE DE IDEACIÓN

1. Descripción general del proyecto

Este proyecto propone el diseño e implementación de un Árbol Genealógico Mitológico, cuyo propósito es modelar la compleja jerarquía de dioses, titanes, héroes y criaturas de la mitología griega.
Para representar esta estructura se emplea un Árbol Binario de Búsqueda (ABB), tomando como clave principal el año simbólico de nacimiento de cada entidad mitológica.

El ABB permite:

- organizar la genealogía en un orden temporal coherente,
- consultar relaciones de antigüedad y descendencia,
- recorrer la estructura bajo distintos enfoques jerárquicos,
- realizar operaciones de búsqueda en tiempo eficiente,
- mantener el orden interno sin necesidad de estructuras externas.

Cada nodo representa a un ser mitológico y contiene su nombre, su rol jerárquico y su año simbólico, lo que permite reconstruir relaciones como:

- linajes divinos (Titanes → Olímpicos),
- descendencias célebres (Zeus → Atenea → héroes),
- etapas mitológicas (primigenios → olímpicos → héroes).

2. Requerimientos del sistema

Requerimientos funcionales (RF)

- RF1: Insertar dioses, titanes, héroes o criaturas en el árbol.
- RF2: Buscar miembros por año simbólico.
- RF3: Eliminar nodos conservando el orden del ABB.
- RF4: Mostrar recorridos genealógicos:
    - Inorden (línea temporal)
    - Preorden (jerarquía divina)
    - Postorden (generaciones)
- RF5: Representar líneas familiares dentro del árbol.

Requerimientos no funcionales (RNF)

- RNF1: Interfaz sencilla y compatible con evaluación académica.
- RNF2: Operaciones con eficiencia promedio O(log n).
- RNF3: Implementación en C++ bajo entorno Dev-C++.
- RNF4: Manejo robusto de entradas incorrectas.

3. Respuestas a preguntas guía

1. Información almacenada en cada nodo

Cada nodo del ABB representa un personaje mitológico.
Debe incluir:

- nombre (ej. Zeus, Hades, Cronos)
- rol mitológico (Titán, Olímpico, Semidiós, Criatura)
- año simbólico de nacimiento → clave del ABB
- puntero izquierdo → entidades más antiguas
- puntero derecho → entidades más recientes
  Esto permite mantener un árbol ordenado temporalmente y apto para consultas eficientes.

2. Inserción y eliminación sin romper el ABB

<u>Inserción</u>
Se compara el año nuevo con el nodo actual:

- menor → va al subárbol izquierdo
- mayor → va al subárbol derecho
- igual → se rechaza para evitar duplicados

El algoritmo es recursivo y garantiza la validez del ABB.

<u>Eliminación</u>

Se usa el método estándar del ABB:

1. Sin hijos: se elimina directamente
2. Un hijo: el hijo reemplaza al nodo
3. Dos hijos: se reemplaza con el sucesor inorden

(nodo más pequeño del subárbol derecho)

3. Métodos de recorrido

Tres recorridos clásicos permiten visualizar diferentes aspectos:

- Inorden (izq – raíz – der):
  - Lista cronológica de los seres mitológicos
- Preorden (raíz – izq – der):
  - Muestra jerarquía divina
- Postorden (izq – der – raíz):
  - Permite analizar generaciones completas

Cada uno responde a un tipo de consulta distinta.

4. Determinar si pertenece a una rama

Para saber si alguien pertenece a una línea ancestral:

1. buscar al ancestro (ej. Cronos, Zeus),
2. desde su nodo, recorrer preorden/inorden,
3. comprobar si aparece el miembro buscado.

También se pueden establecer ramas por rol:

- Titanes → primigenios
- Olímpicos → generación central
- Semidioses → descendientes
- Criaturas → derivados alternos

Si el miembro está dentro del subárbol de esa categoría, pertenece a la rama.

5. Balancear el árbol

Si se insertan años muy dispersos, el ABB puede volverse una lista.
Para evitarlo:

Métodos aplicables:
Reconstrucción balanceada (RECOMENDADO)

1. obtener nodos en inorden,
2. reconstruir tomando elemento medio como raíz,
3. subárboles con elementos anteriores/siguientes.

Complejidad: O(n).

Rotaciones (AVL)

- izquierda
- derecha
- doble

CAPÍTULO 2 – FASE DE SOLUCIÓN (VERSIÓN LISTA PARA INFORME)

1. DESCRIPCIÓN DE LA ESTRUCTURA DE DATOS Y OPERACIONES

El sistema utiliza un Árbol Binario de Búsqueda (ABB) para modelar la genealogía mitológica.
Cada nodo representa un dios, titán, semidiós o criatura, ordenados según su año simbólico de nacimiento.

1.1 Estructura de cada nodo

Cada nodo contiene:

- int nacimiento → clave para ordenar el ABB
- string nombre → nombre mitológico
- string rol → categoría (Titán, Olímpico, Criatura, etc.)
- Nodo* izq → puntero a seres más antiguos
- Nodo* der → puntero a seres más jóvenes

1.2 Operaciones implementadas

Operación	Descripción
Insertar	Inserta por año, manteniendo el orden del ABB
Buscar	Encuentra un nodo por año simbólico
Eliminar	Maneja casos sin hijos, con 1 hijo y con 2 hijos
Recorridos	Inorden, Preorden, Postorden
Mínimo	Encuentra el sucesor para eliminación

2. ALGORITMOS PRINCIPALES (PSEUDOCÓDIGO)

2.1 Pseudocódigo: Crear un Árbol Binario (Insertar nodo en ABB)

```
ALGORITMO Insertar(raiz, nacimiento, nombre, rol)
    SI raiz ES NULO ENTONCES
        // Crear un nuevo nodo con la información proporcionada
        nuevoNodo ← CrearNodo(nacimiento, nombre, rol)
        RETORNAR nuevoNodo
    FIN SI

    // Si la fecha de nacimiento es menor que la raíz, insertamos en el subárbol izquierdo
    SI nacimiento < raiz.nacimiento ENTONCES
        raiz.izq ← Insertar(raiz.izq, nacimiento, nombre, rol)

    // Si la fecha de nacimiento es mayor que la raíz, insertamos en el subárbol derecho
    SINO SI nacimiento > raiz.nacimiento ENTONCES
        raiz.der ← Insertar(raiz.der, nacimiento, nombre, rol)

    // Si la fecha de nacimiento es igual a la raíz, mostramos un mensaje de error
    SINO
        MOSTRAR "Clave duplicada. No se puede insertar."
    FIN SI

    // Retornamos la raíz (sin cambios si no se insertó nada)
    RETORNAR raiz
FIN ALGORITMO
```

La función Inserción coloca cada nuevo ser mitológico en el lugar correcto del ABB comparando su año simbólico de nacimiento con el de los nodos existentes.

2.2 Pseudocódigo: Recorridos del Árbol

A) Inorden (Izquierda – Raíz – Derecha)

```
ALGORITMO Inorden(raiz)
    SI raiz ES NULO ENTONCES
        // Si el nodo es nulo, simplemente retornamos
        RETORNAR
    FIN SI

    // Recursivamente recorremos el subárbol izquierdo
    Inorden(raiz.izq)

    // Mostramos los datos del nodo actual
    MOSTRAR raiz.nombre, raiz.nacimiento, raiz.rol

    // Recursivamente recorremos el subárbol derecho
    Inorden(raiz.der)
FIN ALGORITMO
```

El recorrido Inorden es ideal para mostrar la línea temporal de los seres mitológicos.

2.3 Pseudocódigo: recorrido Preorden

```
ALGORITMO Preorden(raiz)
    SI raiz ES NULO ENTONCES
        // Si el nodo es nulo, no hay nada que procesar
        RETORNAR
    FIN SI

    // 1. Procesar el nodo actual
    MOSTRAR raiz.nombre, raiz.nacimiento, raiz.rol

    // 2. Recorrer el subárbol izquierdo
    Preorden(raiz.izq)

    // 3. Recorrer el subárbol derecho
    Preorden(raiz.der)
FIN ALGORITMO
```

Permite observar la jerarquía divina empezando por la raíz.

2.4 Pseudocódigo: recorrido Postorden

```
ALGORITMO Postorden(raiz)
    SI raiz ES NULO ENTONCES
        // Si el nodo es nulo, no hay nada que procesar
        RETORNAR
    FIN SI

    // 1. Recorrer el subárbol izquierdo
    Postorden(raiz.izq)

    // 2. Recorrer el subárbol derecho
    Postorden(raiz.der)

    // 3. Procesar el nodo actual
    MOSTRAR raiz.nombre, raiz.nacimiento, raiz.rol
FIN ALGORITMO
```

Útil para mostrar generaciones completas desde los descendientes hacia el progenitor.

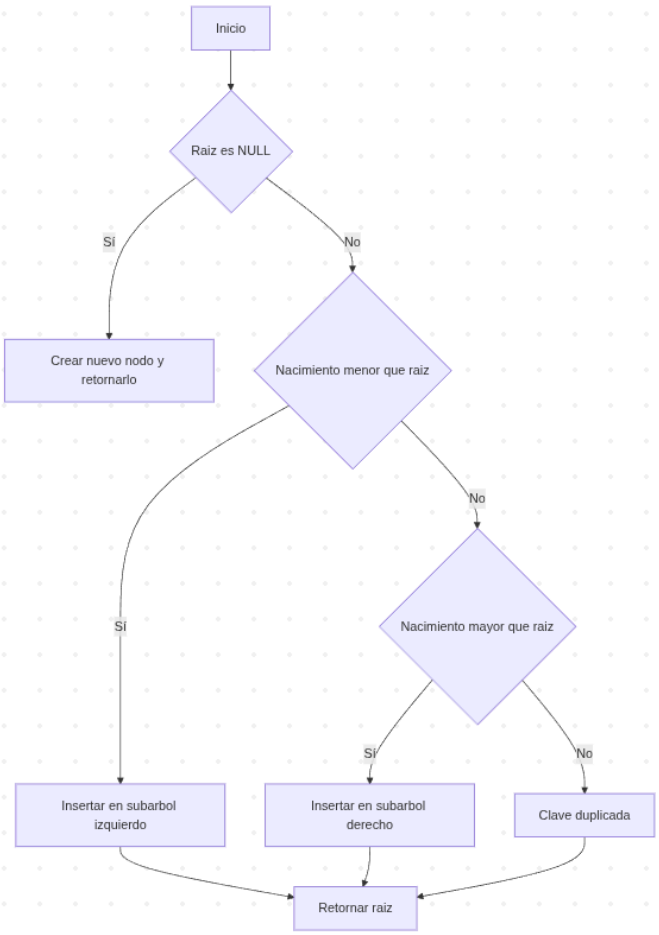
2.5 Pseudocódigo: eliminar un miembro del ABB

```
ALGORITMO Eliminar(raiz, nacimiento)
    SI raiz ES NULO ENTONCES
        // No se encontró la clave
        RETORNAR NULL
    FIN SI

    // Buscar el nodo a eliminar
    SI nacimiento < raiz.nacimiento ENTONCES
        raiz.izq ← Eliminar(raiz.izq, nacimiento)

    SINO SI nacimiento > raiz.nacimiento ENTONCES
        raiz.der ← Eliminar(raiz.der, nacimiento)

    SINO
        // Caso 1: Nodo sin hijos (hoja)
        SI raiz.izq ES NULL Y raiz.der ES NULL ENTONCES
            eliminar raiz
            RETORNAR NULL
        FIN SI

        // Caso 2: Nodo con un solo hijo (derecho)
        SI raiz.izq ES NULL ENTONCES
            temp ← raiz.der
            eliminar raiz
            RETORNAR temp
        FIN SI

        // Caso 2: Nodo con un solo hijo (izquierdo)
        SI raiz.der ES NULL ENTONCES
            temp ← raiz.izq
            eliminar raiz
            RETORNAR temp
        FIN SI

        // Caso 3: Nodo con dos hijos
        sucesor ← Minimo(raiz.der)
        copiar datos de sucesor a raiz
        raiz.der ← Eliminar(raiz.der, sucesor.nacimiento)
    FIN SI

    RETORNAR raiz
FIN ALGORITMO
```

Este algoritmo mantiene la estructura del árbol al eliminar un nodo, usando el sucesor inorden si es necesario.

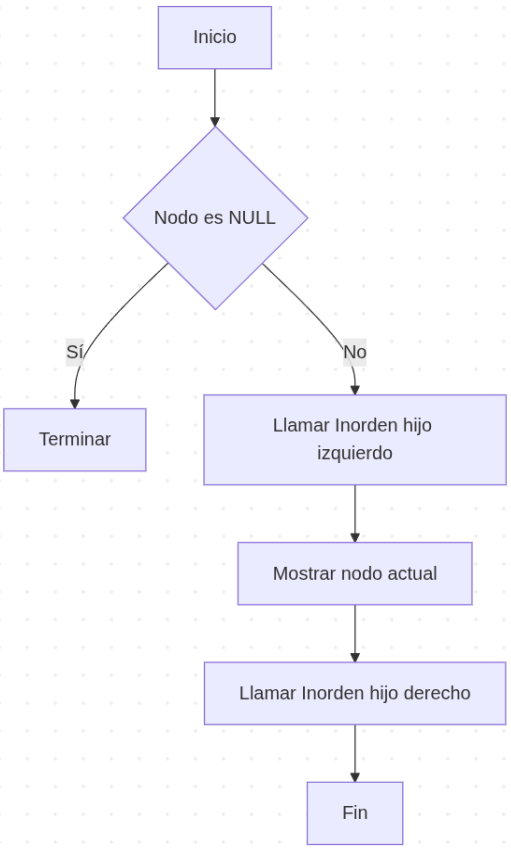
3. DIAGRAMAS DE FLUJO

DIAGRAMA 1 – Proceso de Inserción en un Árbol Binario de Búsqueda (ABB)



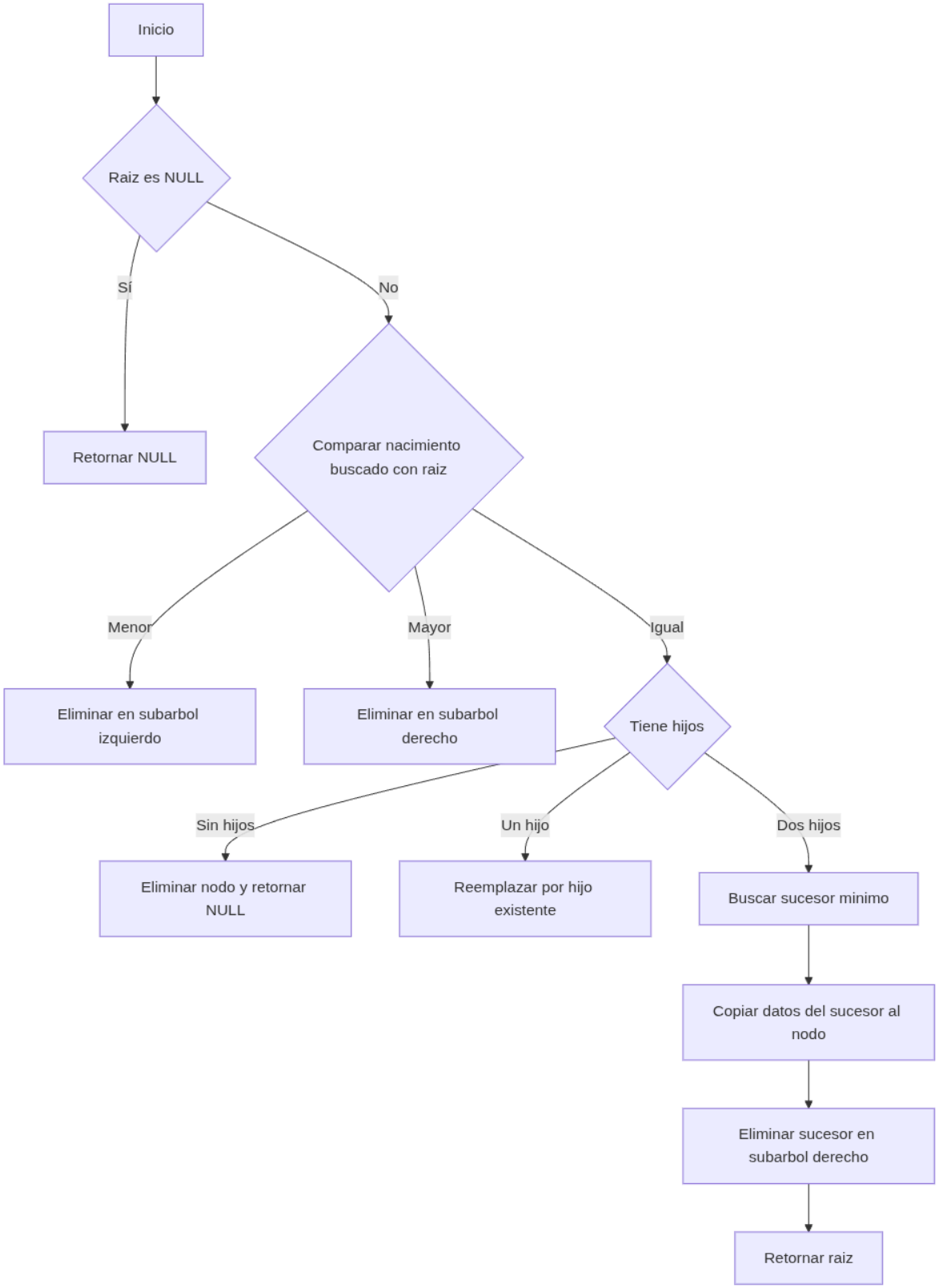
Este diagrama muestra cómo el algoritmo decide en qué parte del árbol insertar un nuevo miembro basado en su año simbólico.
Si el árbol está vacío, crea la raíz; si el año es menor avanza por el subárbol izquierdo, si es mayor avanza por el derecho; y si es igual, detecta duplicados.

DIAGRAMA 2 — Recorrido INORDEN (Izq – Raíz – Der)



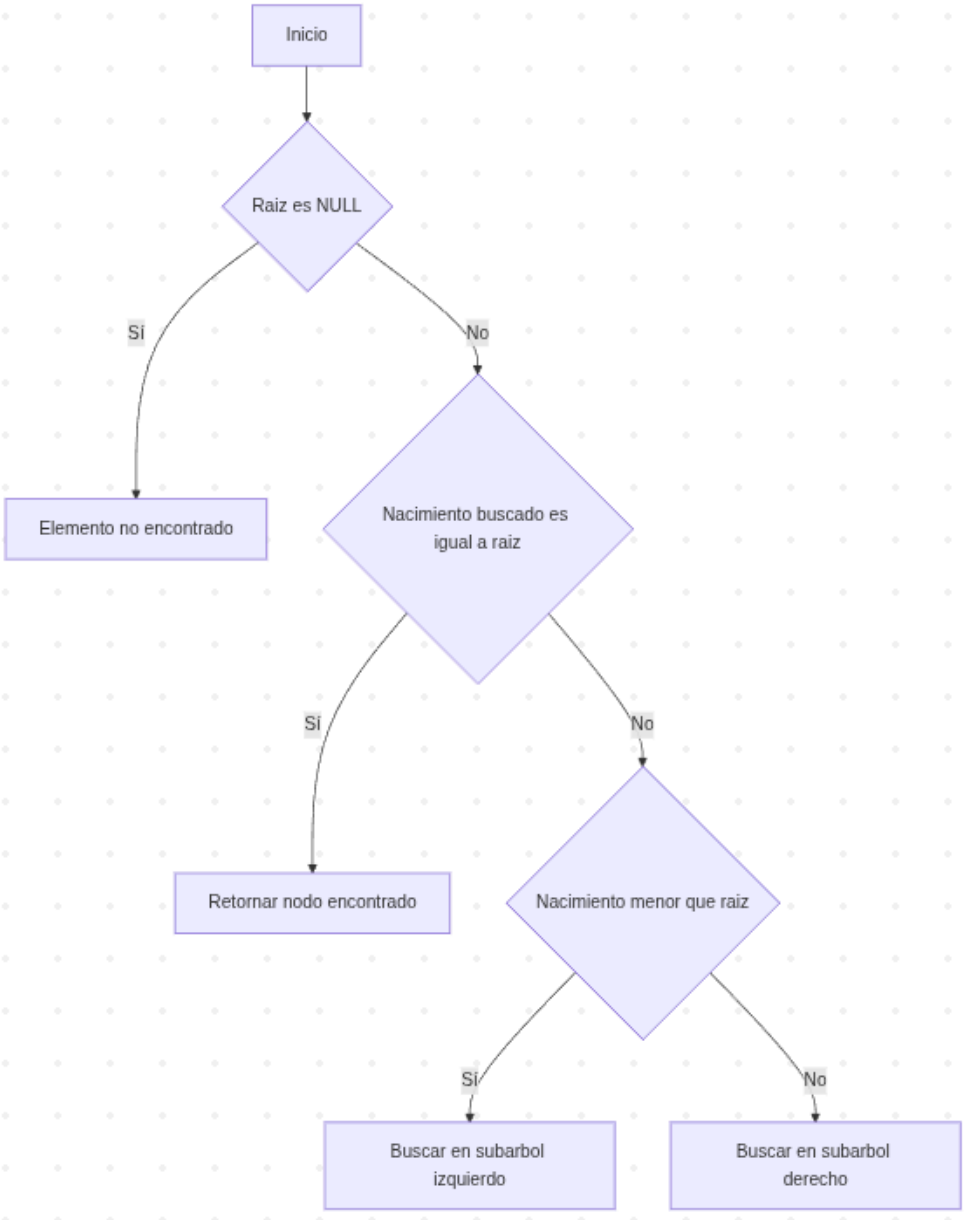
Este recorrido muestra los elementos del árbol en orden ascendente según su año simbólico.
Primero explora el subárbol izquierdo (miembros más antiguos), luego muestra el nodo actual y finalmente el subárbol derecho (miembros más recientes).

DIAGRAMA 3 – Eliminación en un Árbol Binario de Búsqueda (ABB)



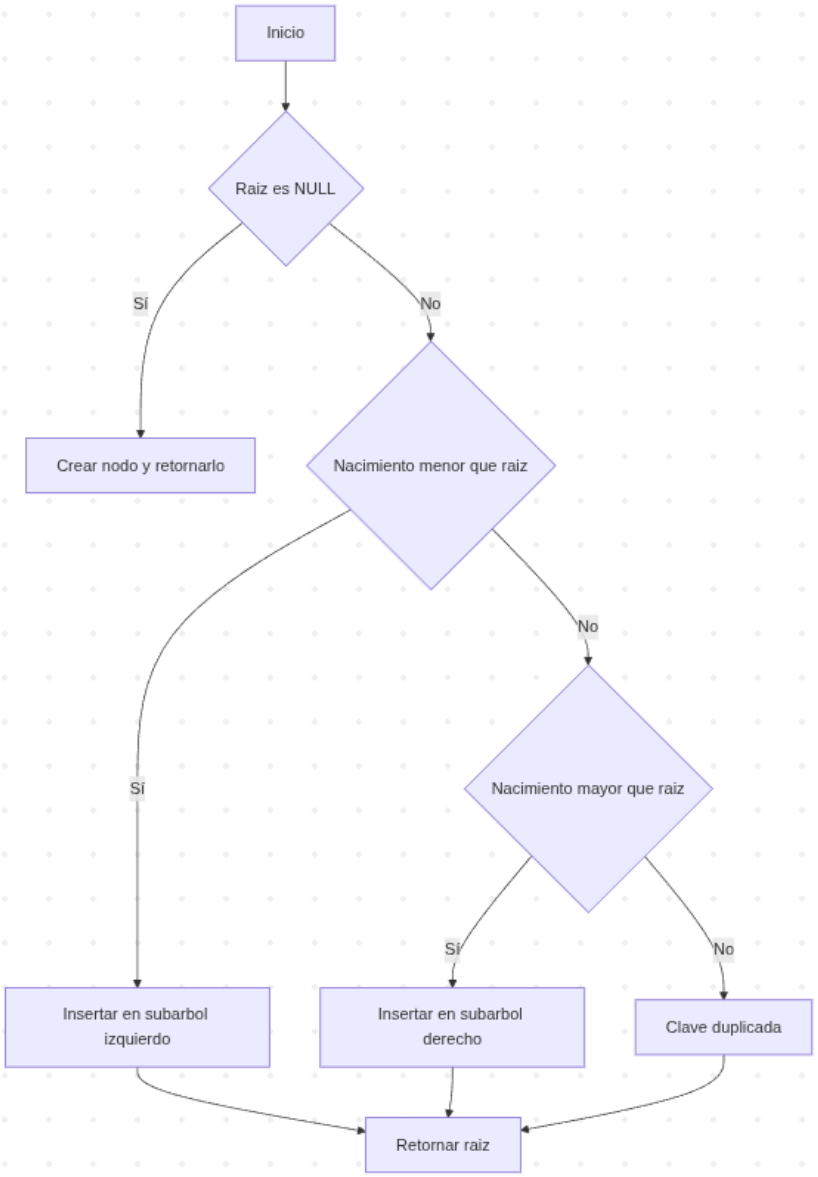
Este diagrama explica el proceso para eliminar un nodo de un ABB manteniendo el orden. El algoritmo distingue entre nodos sin hijos, con un hijo, o con dos hijos (donde es necesario reemplazar con el sucesor mínimo del subárbol derecho).

DIAGRAMA 4 – Búsqueda en ABB



El algoritmo de búsqueda compara la clave buscada con el nodo actual y decide si explorar izquierda, derecha o finalizar si encuentra el valor.

DIAGRAMA 5 – Inserción en ABB (para referencia)



El proceso inserta un nuevo nodo respetando el orden del ABB: menores a la izquierda, mayores a la derecha, evitando duplicados.

4. AVANCE DEL CÓDIGO FUENTE

1.1 AVANCE DE FUNCIONES

```
/*
  Estructura de un nodo del Árbol Binario de Búsqueda (ABB).
  Cada nodo representa un ser de la mitología griega,
  almacenado de acuerdo a su año simbólico de nacimiento.
*/
struct Nodo {
    int nacimiento;     // Clave del nodo: año simbólico
    string nombre;      // Nombre del ser mitológico
    string rol;         // Rol dentro de la mitología

    Nodo* izq;          // Hijo izquierdo
    Nodo* der;          // Hijo derecho

    // Constructor que inicializa un nodo
    Nodo(int n, string nom, string r) {
        nacimiento = n;
        nombre = nom;
        rol = r;
        izq = NULL;
        der = NULL;
    }
};

/*
  Función insertar:
  Inserta un nuevo nodo en el ABB respetando el orden.
  Si el valor ya existe, no se inserta.
*/
Nodo* insertar(Nodo* raiz, int nacimiento, string nombre, string rol) {
    if (raiz == NULL) {
        return new Nodo(nacimiento, nombre, rol);
    }

    // Inserción recursiva siguiendo la regla del ABB
    if (nacimiento < raiz->nacimiento) {
        raiz->izq = insertar(raiz->izq, nacimiento, nombre, rol);
    }
    else if (nacimiento > raiz->nacimiento) {
        raiz->der = insertar(raiz->der, nacimiento, nombre, rol);
    }
    else {
        // Valor duplicado: no se debe insertar
        cout << "Ya existe un miembro con ese año.\n";
    }

    return raiz;
}

/*
  Función buscar:
  Retorna el nodo cuya clave coincide con "nacimiento".
  Si no existe, retorna NULL.
*/
Nodo* buscar(Nodo* raiz, int nacimiento) {
    if (raiz == NULL || raiz->nacimiento == nacimiento)
        return raiz;

    if (nacimiento < raiz->nacimiento)
        return buscar(raiz->izq, nacimiento);
    else
        return buscar(raiz->der, nacimiento);
}
```

1.2 AVANCE RECORRIDO INORDEN, PREORDEN POSTORDEN

```cpp
/*
  Recorrido Inorden:
  Muestra los nodos en orden ascendente según año simbólico.
*/
void inorden(Nodo* raiz) {
    if (raiz != NULL) {
        inorden(raiz->izq);
        cout << raiz->nombre << " (" << raiz->nacimiento
             << ") - " << raiz->rol << endl;
        inorden(raiz->der);
    }
}

/*
  Recorrido Preorden:
  Muestra primero la raíz y luego los hijos.
  Útil para observar jerarquías.
*/
void preorden(Nodo* raiz) {
    if (raiz != NULL) {
        cout << raiz->nombre << " (" << raiz->nacimiento
             << ") - " << raiz->rol << endl;
        preorden(raiz->izq);
        preorden(raiz->der);
    }
}

/*
  Recorrido Postorden:
  Muestra primero los hijos y al final la raíz.
  Útil para analizar descendencia completa.
*/
void postorden(Nodo* raiz) {
    if (raiz != NULL) {
        postorden(raiz->izq);
        postorden(raiz->der);
        cout << raiz->nombre << " (" << raiz->nacimiento
             << ") - " << raiz->rol << endl;
    }
}

/*
  Función minimo:
  Encuentra el nodo más pequeño dentro de un subárbol.
  Se usa para eliminar nodos con dos hijos.
*/
Nodo* minimo(Nodo* nodo) {
    while (nodo != NULL && nodo->izq != NULL)
        nodo = nodo->izq;
    return nodo;
}
```

El programa define un nodo de árbol binario que almacena el año de nacimiento, nombre y rol de un personaje, junto con punteros a sus hijos izquierdo y derecho.
La función insertar agrega nuevos nodos respetando el orden de un árbol binario de búsqueda.
La función buscar recorre el árbol de forma recursiva hasta encontrar el año solicitado.
Los recorridos inorden, preorden y postorden muestran los nodos en distintas secuencias según el tipo de análisis deseado.
La función mínima ubica el nodo con el menor valor avanzando por la rama izquierda del árbol, y se utiliza principalmente como apoyo para la eliminación.