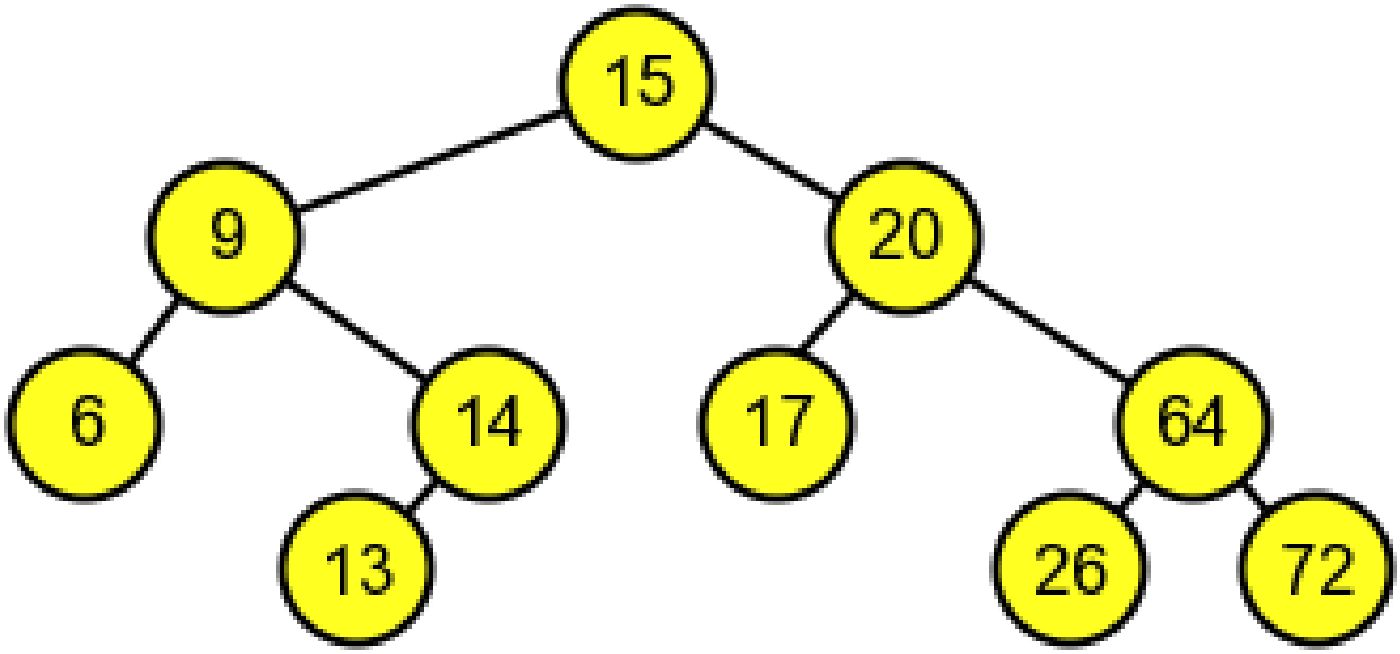


Práctica 3- Árboles

-Diego García Santos, Dimitar Vasilev

-Grupo 2

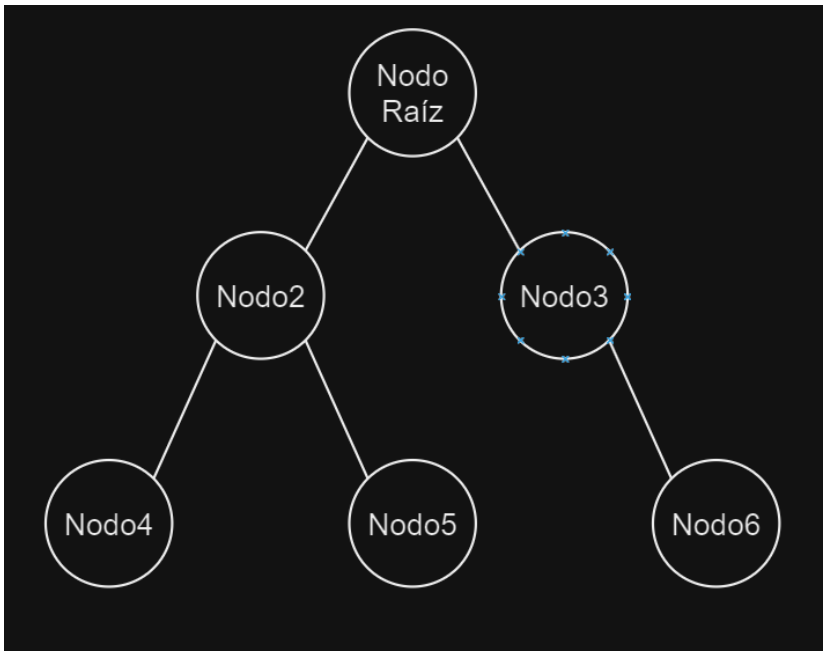
-Laboratorio 2L

22/05/2024

Especificación lógica del TAD AB

1-Definición

El TAD diseñado presenta una colección de elementos que pueden poseer dos elementos “hijos” así como un elemento “padre”, el punto de acceso al TAD es un nodo de este el cual no posee nodos predecesores y que se denomina nodo raíz.



2-Elementos

Cada elemento puede almacenar un dato cualquiera gracias a la genericidad, de esta forma se puede almacenar un tipo de dato primitivo u otro TAD. No obstante, todos los elementos almacenados en un mismo árbol deben pertenecer al mismo tipo de dato.

3-Tipo de organización

Presenta una organización jerárquica, es decir, cada elemento del TAD puede poseer uno o dos elementos hijos

4-Dominio de los elementos

Al igual que con los elementos, el dominio del TAD no está restringido pudiéndose almacenar cualquier valor siempre que pertenezca al tipo de dato que se está almacenando en la cola

5-Operaciones básicas

Nombre: listadoInOrden

Utilidad: Muestra por pantalla todos los nodos del árbol siguiendo un recorrido In Orden

Entrada: No tiene entrada

Salida: Lista de los nodos en In Orden

Pre Condición: El árbol existe

Post Condición: No tiene postcondicion

Nombre: listadoInOrdenConverso

Utilidad: Muestra por pantalla todos los nodos del árbol siguiendo un recorrido In Orden Converso

Entrada: No tiene entrada

Salida: Lista de los nodos en In Orden Converso

Pre Condición: El árbol existe

Post Condición: No tiene postcondicion

Nombre: listadoPreOrden

Utilidad: Muestra por pantalla todos los nodos del árbol siguiendo un recorrido Pre Orden

Entrada: No tiene entrada

Salida: Lista de los nodos en Pre Orden

Pre Condición: El árbol existe

Post Condición: No tiene postcondicion

Nombre: listadoPostOrden

Utilidad: Muestra por pantalla todos los nodos del árbol siguiendo un recorrido Post Orden

Entrada: No tiene entrada

Salida: Lista de los nodos en Post Orden

Pre Condición: El árbol existe

Post Condición: No tiene postcondicion

Nombre: VaciarArbolSinRecorrer

Utilidad: Vacía un árbol sin recorrer sus nodos

Entrada: No tiene entrada

Salida: No tiene salida

Pre Condición: El árbol no está vacío

Post Condición: El árbol se ha vaciado

Nombre: VaciarArbolRecorriendo

Utilidad: Vacía un árbol accediendo a todos sus nodos

Entrada: No tiene entrada

Salida: No tiene salida

Pre Condición: El árbol no está vacío

Post Condición: El árbol se ha vaciado

Nombre: esVacio

Utilidad: Comprueba si el árbol está vacío

Entrada: No tiene entrada

Salida: true si está vacío, false si no lo está

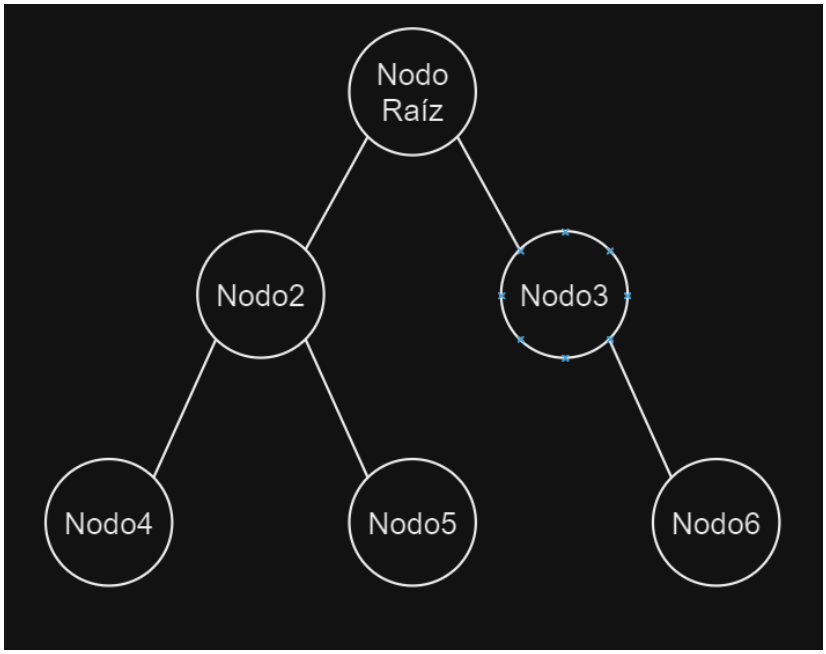
Pre Condición: El árbol existe

Post Condición: No tiene postcondición

Especificación lógica del TAD ABENTEROS

1-Definición

El TAD diseñado presenta una colección de elementos y se rige por la definición del árbol binario, pero siendo todos los elementos de este árbol de tipo entero.



2-Elementos

Cada elemento puede almacenar un dato de tipo entero, de esta forma todos los elementos son elementos de tipo entero y se puede trabajar realizando operaciones entre ellos.

3-Tipo de organización

Presenta una organización lineal, es decir, cada elemento del TAD presenta una relación 1:1

4-Dominio de los elementos

Presenta una organización jerárquica, es decir, cada elemento del TAD puede poseer uno o dos elementos hijos

5-Operaciones básicas

Nombre: comprobarSumas

Utilidad: comrueba que la suma de los elementos del subárbol izq y el subárbol der de cada uno de los nodos del árbol son iguales

Entrada: No tiene entrada

Salida: Devuelve true si se cumple la condición false si no se cumple

Pre Condición: El árbol existe (si está vacío devuelve true ya que si que se cumple la condición)

Post Condición: No tiene postcondición

Nombre: comprobarClavePequenia

Utilidad: comprueba que el valor almacenado en cada nodo es menor o igual al de cada uno de sus nodos hijo.

Entrada: No tiene entrada

Salida: Devuelve true si se cumple la condición false si no se cumple

Pre Condición: El árbol existe (si está vacío devuelve true ya que si que se cumple la condición)

Post Condición: No tiene postcondición

Nombre: sumaNodosNiveles

Utilidad: suma los nodos que hay entre dos niveles del arbol binario de enteros

Entrada: No tiene entrada

Salida: devuelve un entero con el número total de la suma de los nodos de los niveles especificados

Pre Condición: El árbol existe

Post Condición: Ninguna

Nombre: mostrarAscendientes

Utilidad: Busca un valor dado por el usuario y mostrar todos los valores de los nodos ascendentes a este

Entrada: No tiene entrada

Salida: string con los valores de los nodos ascendentes

Pre Condición: El árbol existe y no esta vacío

Post Condición: No tiene postcondición

Nombre: numeroNodosImpares

Utilidad: muestra cuantos nodos impares hay en un nivel específico del árbol

Entrada: No tiene entrada

Salida: numero de nodos impares en un nivel

Pre Condición: El árbol existe y no esta vacío

Post Condición: No tiene postcondición

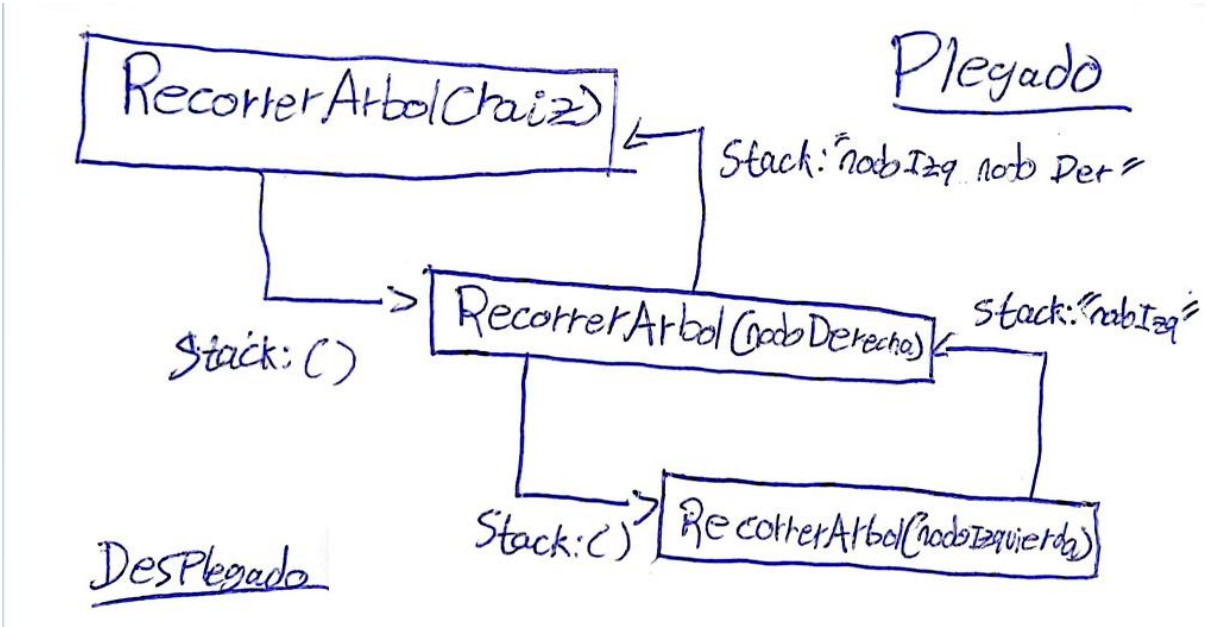
Diseño de los Métodos Recursivos

1. Recorrer árbol

Caso Base: Estar en un nodo hoja

Desplegado: Recorrer el árbol hasta acceder a un nodo hoja

Plegado: Construir la cadena con los valores

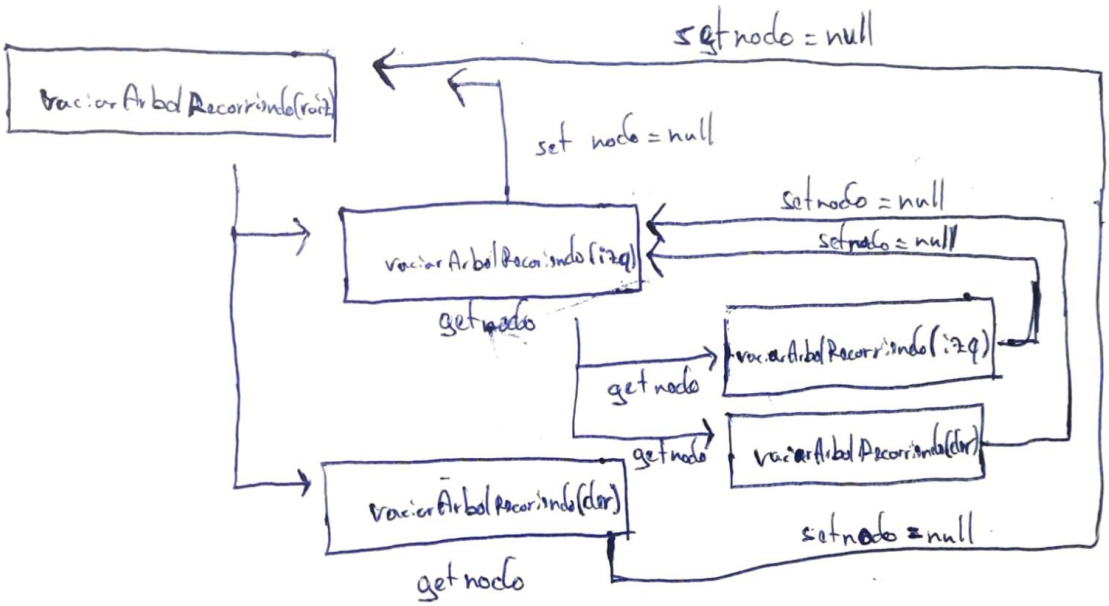


2. Vaciar árbol recorriendo

Caso Base: Estar en un nodo hoja

Desplegado: Acceder a los nodos hoja

Plegado: Eliminar referencias a los nodos

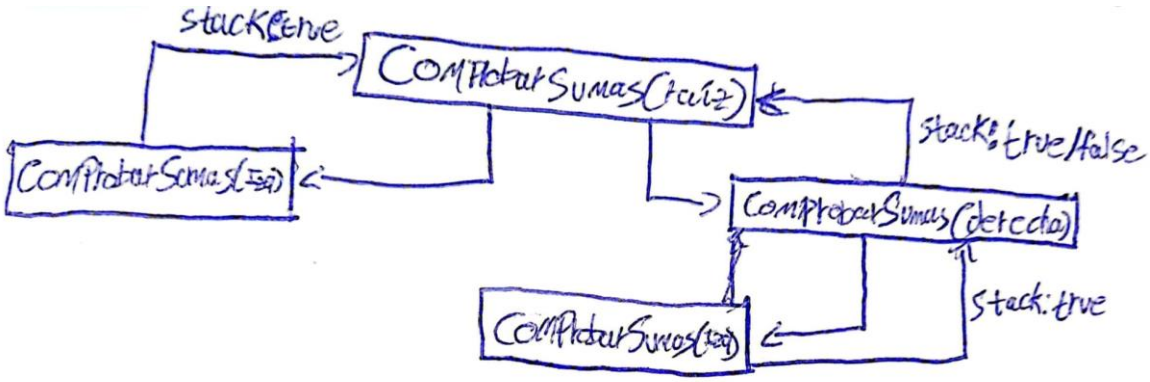


3. Comprobar Sumas

Caso Base: Estar en un nodo hoja

Desplegado: Acceder a los nodos hoja

Plegado: Sumar valores de los nodos y comprobar si las sumas de sus subarboles derecho e izquierdo es igual

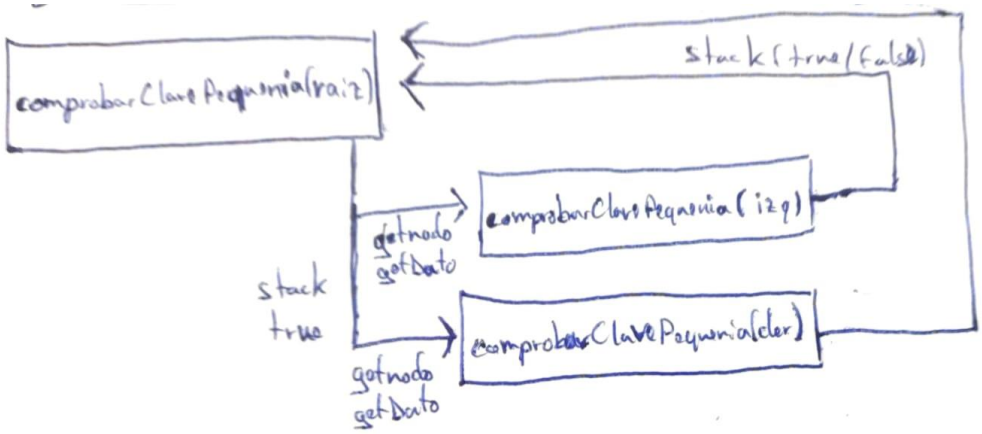


4. Comprobar clave pequeña

Caso Base: Estar en un nodo hoja

Desplegado: Acceder a los nodos hoja

Plegado: Comprobar si el valor de los nodos hijos es superior al de los nodos padre

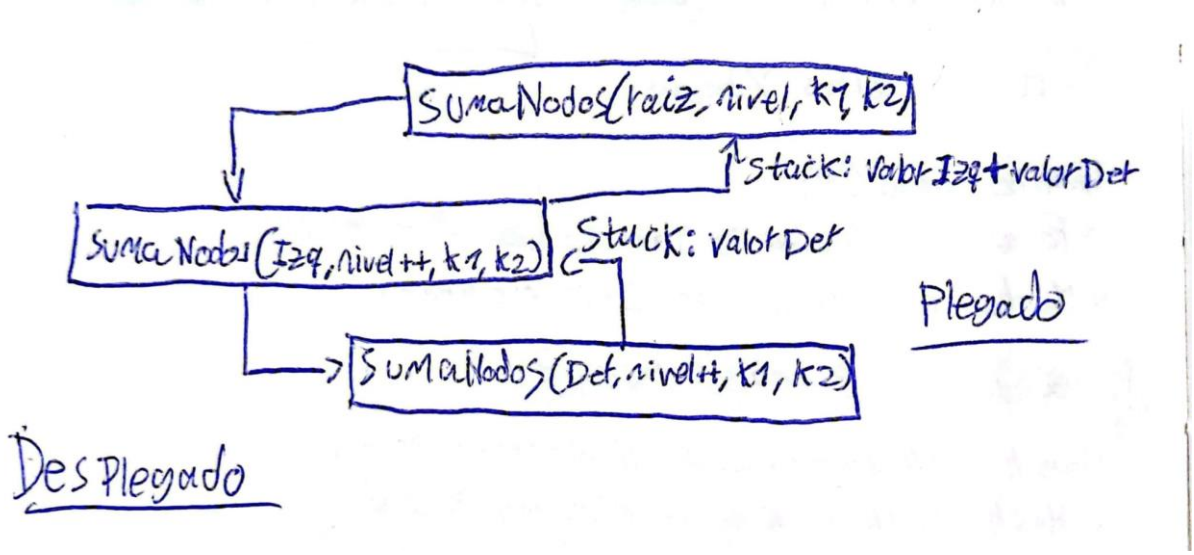


5. Suma Nodos Niveles

Caso Base: Acceder a un nodo del nivel indicado

Desplegado: Acceder a los nodos de los niveles indicados

Plegado: Sumar los valores dentro de los nodos de los niveles

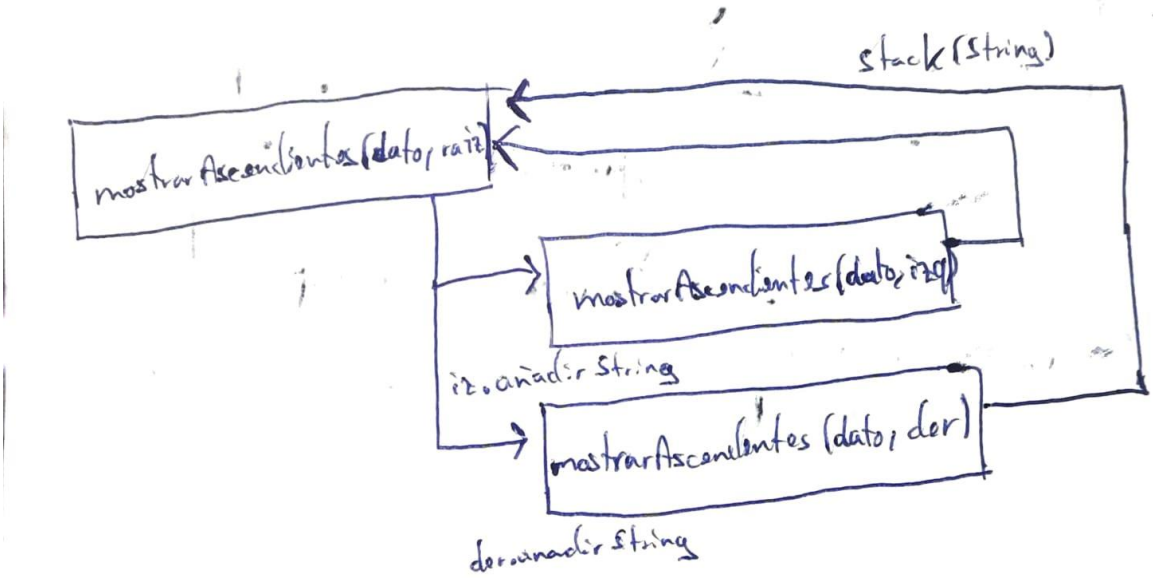


6. Mostrar Ascendientes

Caso base: Encontrarse en el nodo indicado pasado por parámetro

Desplegado: Acceder a los nodos que llevan hasta ese nodo

Plegado: Añadir al string resultado el dato de cada nodo hasta el nodo pasado por parámetro

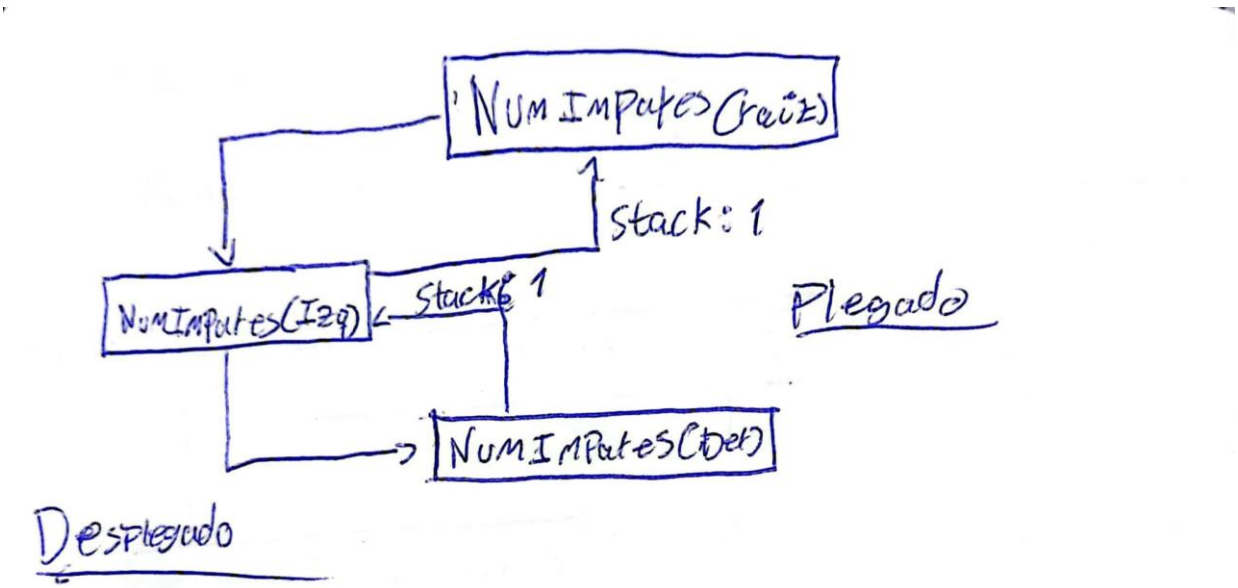


7. Numero Nodos Impares

Caso base: Estar en un nodo hoja

Desplegado: Acceder a los nodos hoja

Plegado: Contar numero de nodos con valor impar



Implementación de los métodos

1. Crear AB de enteros

Para que este método sea estático primero creamos los dos árboles por separado en el mismo método, los creamos empezando por las hojas hasta llegar a la raíz. Una vez construidos creamos un nodo que actúe como raíz de ambos árboles, establecemos que el nodo derecho sea la raíz del primer árbol y el nodo izquierdo sea la raíz del segundo árbol.

Devolvemos los dos árboles y establecemos que la raiz del primer árbol sea el nodo derecho y la raíz del segundo sea el nodo izquierdo.

2. Listado(InOrden, InordenConverso, PostOrden, PreOrden)

Recorremos el árbol y vamos accediendo de forma recursiva a todos los nodos hoja, dependiendo del tipo de orden que necesitemos para cada listado/método, lo haremos en un orden o otro y lo añadiremos a un string que será lo que devuelva el método cuando alcance el caso base

3. Comprobar sumas

Accedemos de manera recursiva a los nodos hoja y en la fase de plegado comprobamos que la suma de los subárboles izquierdo y derecho sea la misma, en caso de serlo se devuelve true, en caso contrario false. Si se recibe false no se comprueba la suma ya que, aunque se cumpliese se seguiría devolviendo false

4. Comprobar Clave Pequeña

En este método se accede de manera recursiva a los nodos hoja del árbol indicado, durante la fase de plegado se comprobará si el valor de un nodo hijo es superior al de su nodo padre, en caso de serlo se devuelve true, en caso contrario false.

Si se recibe false se deja de comprobar esta condición y se devolverá false directamente.

5. Vaciar Árbol (Sin recorrer)

En este método se vacía el árbol poniendo la referencia a la raíz del árbol como null, de esta forma el recolector automático de basura se encargará de liberar el espacio ocupado por dicho árbol

6. Vaciar Árbol (Recorriendo)

Para implementar este método se identifica como caso base acceder a un nodo hoja, una vez se tiene la referencia a dicho nodo su nodo padre la elimina dejándolo inaccesible para que el recolector automático de basura lo elimine, en la fase de plegado se busca el nodo hoja y en la de desplegado se borran sucesivamente las referencias a los nodos.

7. Mostrar ascendientes

En este método se accede recursivamente al nodo indicado por parámetro, una vez encontrado se construirá un string con el valor de sus nodos ascendientes, esto se realiza durante la fase de plegado.

8. Suma total de nodos entre 2 niveles

El caso base de este método es el acceso a un nodo del nivel que se pasa por parámetro, durante la fase de plegado se accede a estos nodos y en la fase de desplegado se suman los valores almacenados entre los nodos de los niveles indicados.

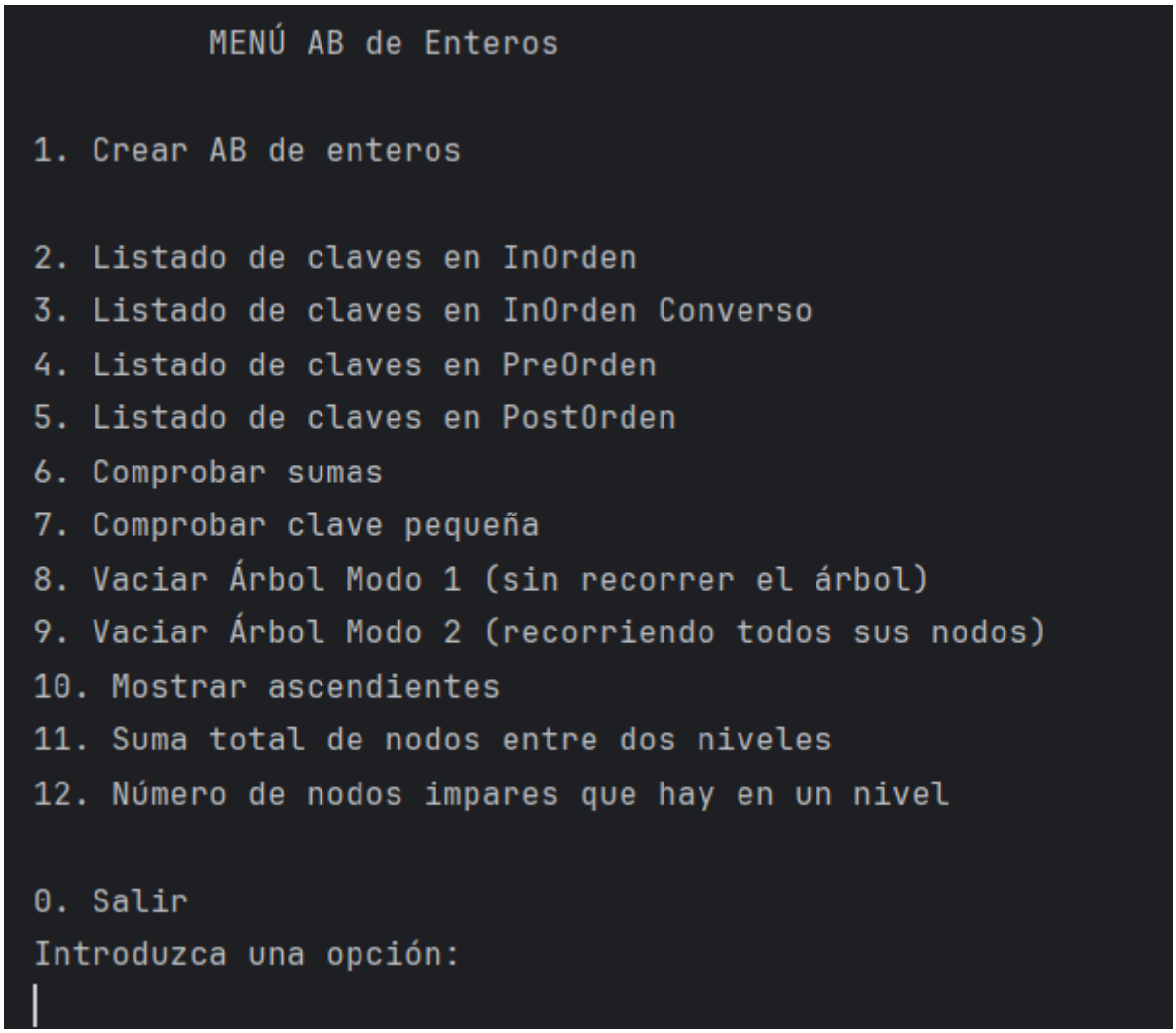
Vamos recorriendo los nodos que hay entre los 2 niveles pasado por parámetro y vamos añadiendo a un contador el valor de cada nodo para obtener el valor numérico de la suma de los nodos entre estos 2 niveles.

9. Número de nodos impares

En este método se accede de manera recursiva a todos los nodos hoja del árbol indicado, durante la fase de plegado se irá incrementando un contador en caso de que el valor almacenado en el nodo en el que nos encontremos sea impar.

Pruebas de ejecución

Menú



Opción 1

```
         MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
1
Árboles creados con éxito
Presione Enter para continuar...
```

Opción 2

```
         MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
2
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
1
9 6 7 2 5 8
Presione Enter para continuar...
```

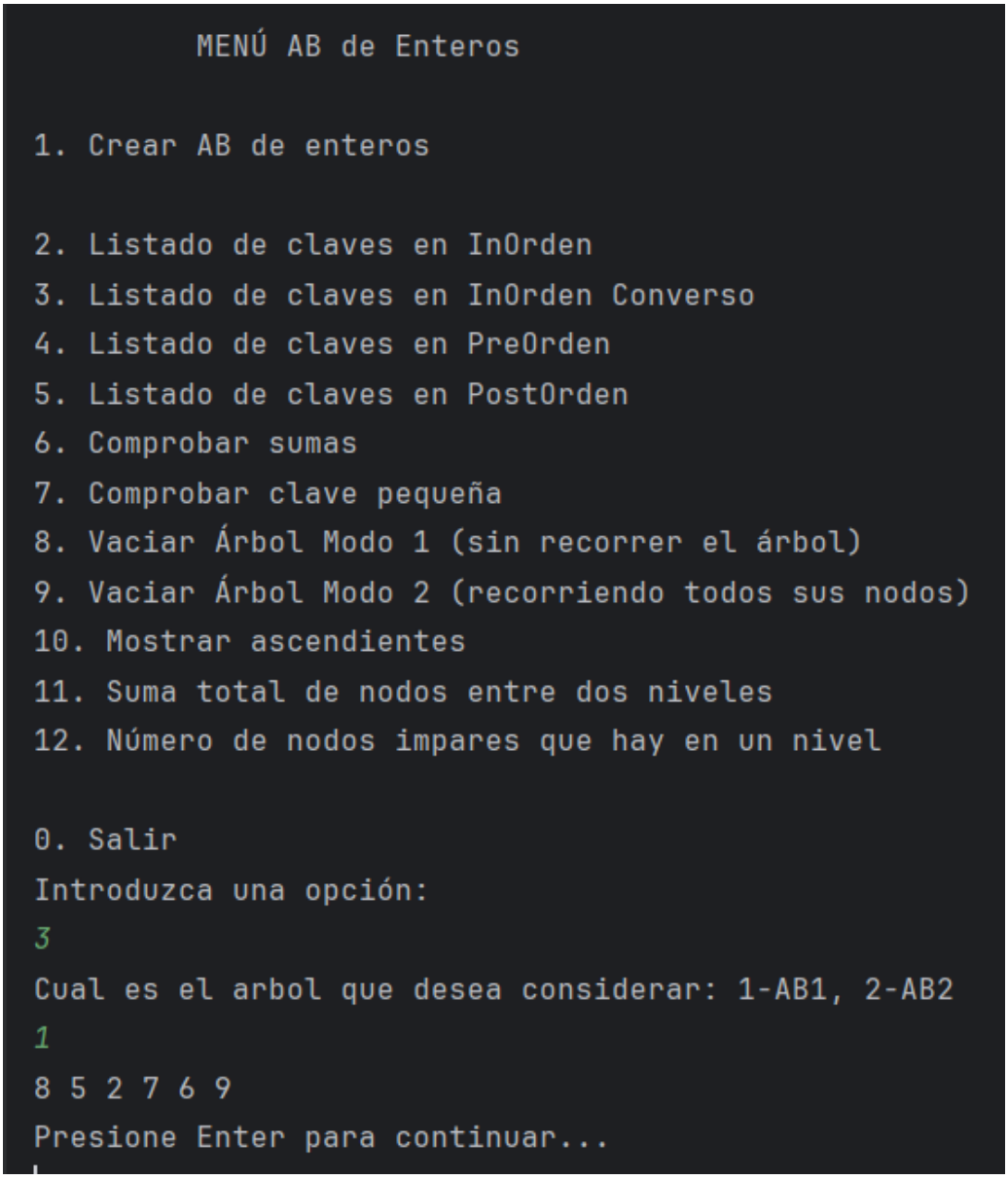
```
         MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
2
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
2
8 12 2 4 2 13 0 -3 34 -3
Presione Enter para continuar...
```

Opción 3


MENÚ AB de Enteros
1. Crear AB de enteros
2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel
0. Salir
Introduzca una opción:
3
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
1
8 5 2 7 6 9
Presione Enter para continuar...

```
         MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
3
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
2
-3 34 -3 0 13 2 4 2 12 8
Presione Enter para continuar...
```

Opción 4

```
         MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
4
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
1
2 6 9 7 5 8
Presione Enter para continuar...
```

```
         MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
4
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
2
13 12 8 4 2 2 34 -3 0 -3
Presione Enter para continuar...
```

Opción 5

```
         MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
5
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
2
8  2  2  4  12  0  -3  -3  34  13
Presione Enter para continuar...
```

```
         MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
5
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
1
9  7  6  8  5  2
Presione Enter para continuar...
```

Opción 6

```
         MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
6
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
1
false
Presione Enter para continuar...
```

```
          MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
6
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
2
true
Presione Enter para continuar...
```

Opción 7

```
         MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
7
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
1
true
Presione Enter para continuar...
```

```
         MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
7
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
2
false
Presione Enter para continuar...
```

Opción 8

```
         MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
8
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
1
Árbol vaciado con éxito
Presione Enter para continuar...
```

Opción 9

```
Introduzca una opción:
9
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
2
Árbol vaciado con éxito
Presione Enter para continuar...
```

Opción 10

```
          MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
10
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
1
Introduzca la clave del nodo:
9
En el arbol AB1 los ascendientes del nodo 9 son: 9 6 2
Presione Enter para continuar...
```

```
         MENÚ AB de Enteros

1. Crear AB de enteros

2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Comprobar sumas
7. Comprobar clave pequeña
8. Vaciar Árbol Modo 1 (sin recorrer el árbol)
9. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)
10. Mostrar ascendientes
11. Suma total de nodos entre dos niveles
12. Número de nodos impares que hay en un nivel

0. Salir
Introduzca una opción:
10
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
2
Introduzca la clave del nodo:
8
En el arbol AB2 los ascendientes del nodo 8 son: 8 12 13
Presione Enter para continuar...
```

Opción 11

```
0. Salir
Introduzca una opción:
11
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
1
Introduzca K1:
0
Introduzca K2:
2
La suma de elementos enre estos niveles es:37
Presione Enter para continuar...
```

```
Introduzca una opción:
11
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
2
Introduzca K1:
1
Introduzca K2:
2
La suma de elementos enre estos niveles es:52
Presione Enter para continuar...
```

Opción 12

```
0. Salir
Introduzca una opción:
12
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
1
Introduzca el nivel:
2
El numero de nodos impares en el arbol AB1 y el nivel 2 es: 2
Presione Enter para continuar...
```

```
Introduzca una opción:
12
Cual es el arbol que desea considerar: 1-AB1, 2-AB2
2
Introduzca el nivel:
0
El numero de nodos impares en el arbol AB2 y el nivel 0 es: 1
Presione Enter para continuar...
```

Respuesta a las preguntas

Pregunta 1: Se recorre el arbol, se eliminan los punteros al nodo y se usa la operacion de liberar espacio en memoria para liberar el espacio correspondiente a cada nodo, en C es free()

INICIO

MIENTRAS NODO NO HOJA

SI NODO DERECHA ES NULL

ACCEDER NODO IZQUIERDA MEDIANTE RECURSIVIDAD

SI NO

ACCEDER NODO DERECHA MEDIANTE RECURSIVIDAD

FIN MIENTRAS

ELIMINAR PUNTERO A NODO

LIBERAR ESPACIO A NODO

FIN

Pregunta 2: En primer lugar, la construcción del árbol sería diferente ya que se tendría que adaptar a las condiciones de un ABB. Además, el método comprobarClavePequeña cambiaria de implementación ya que en un ABB sabemos que a la izquierda de un nodo se encuentran los elementos menores y a la derecha los mayores. El resto de los métodos no cambiarían.

Código fuente

```
package modelos;

public interface I_AB<E> {
    void insertar();
    void eliminar();

}
```

```
package modelos;

public interface I_ABEnteros {
}

package jerarquicos;

public class AB <E>{
    protected NodoAB<E> raiz;
    public AB(NodoAB<E> raiz){
        this.raiz=raiz;
    }
    public boolean esVacio(){
        if(raiz==null){
            return true;
```

```
        }
        else{
            return false;
        }
    }

    public NodoAB<E> getRaiz() {
        return raiz;
    }

    public String listadoInOrden(){
        if(raiz==null){
            return "";
        }
        return listadoInOrden(raiz);
    }
    private String listadoInOrden(NodoAB<E> nodo){
        String res = "";
        if(nodo != null){
            res = listadoInOrden(nodo.getIzq());
            res += nodo.getDato().toString() + " ";
            res += listadoInOrden(nodo.getDer());
        }
        return res;
    }

    public String listadoInOrdenConverso(){
        if (raiz == null){
            return "";
        }
        return listadoInOrdenConverso(raiz);
    }

    private String listadoInOrdenConverso(NodoAB<E> nodo){
        String res = "";
        if(nodo != null){
            res = listadoInOrdenConverso(nodo.getDer());
            res += nodo.getDato().toString() + " ";
            res += listadoInOrdenConverso(nodo.getIzq());
        }
        return res;
    }
```

```java
public String listadoPreOrden(){
    String res;
    if(raiz==null){
        return "Arbol Vacío";
    }
    else {
        res = raiz.getDato().toString();
        if (raiz.getIzq() != null) {
            res += " " + listadoPreOrden(raiz.getIzq());
        }
        if (raiz.getDer() != null) {
            res += " " + listadoPreOrden(raiz.getDer());
        }
        return res;
    }
}
private String listadoPreOrden(NodoAB<E> nodo){
    String res;
    res= nodo.getDato().toString();
    if(nodo.getIzq()!=null){
        res+= " " + listadoPreOrden(nodo.getIzq());
    }
    if(nodo.getDer()!=null){
        res+= " " + listadoPreOrden(nodo.getDer());
    }
    return res;
}
public String listadoPostOrden(){
    String res="";
    if(raiz==null){
        return "Arbol Vacío";
    }
    if(raiz.getIzq()!=null){
        res+=listadoPostOrden(raiz.getIzq()) + " ";
    }
    if(raiz.getDer()!=null){
        res+=listadoPostOrden(raiz.getDer()) + " ";
    }
    res+=raiz.getDato().toString();
    return res;
}
```

```
    private String listadoPostOrden(NodoAB<E> nodo){
        String res="";
        if(nodo.getIzq()!=null){
            res+=listadoPostOrden(nodo.getIzq()) + " ";
        }
        if(nodo.getDer()!=null){
            res+=listadoPostOrden(nodo.getDer()) + " ";
        }
        res+=nodo.getDato().toString() + " ";
        return res;
    }
    public void vaciarArbolSinRecorrer(){
        raiz=null;
    }
    public void vaciarArbolRecorriendo(){
        vaciarArbolRecorriendo(raiz.getDer());
        vaciarArbolRecorriendo(raiz.getIzq());
        raiz.setDer(null);
        raiz.setIzq(null);
        raiz=null;
    }
    public void vaciarArbolRecorriendo(NodoAB<E> nodo){
        if(nodo.getDer()!=null){
            vaciarArbolRecorriendo(nodo.getDer());
        }
        if(nodo.getIzq()!=null){
            vaciarArbolRecorriendo(nodo.getIzq());
        }
        nodo.setIzq(null);
        nodo.setDer(null);
    }




}

package jerarquicos;

public class ABEnteros<Integer> extends AB{
    public ABEnteros(NodoAB raiz) {
        super(raiz);
```

```
    }
    public boolean comprobarSumas(){
        int sumaDer, sumaIzq;
        if(raiz==null){
            return true;
        }
        if(raiz.getDer()==null){
            sumaDer=0;
        }
        else{sumaDer=sumarArboles(raiz.getDer());}
        if(raiz.getIzq()==null){
            sumaIzq=0;
        }
        else{sumaIzq=sumarArboles(raiz.getIzq());}
        if(raiz.getDer()==null && raiz.getIzq()==null){
            return true;
        }
        if(sumaIzq==sumaDer && comprobarSumas(raiz.getDer()) &&
comprobarSumas(raiz.getIzq())){
            return true;
        }
        else {
            return false;
        }
    }
    private boolean comprobarSumas(NodoAB<Integer> nodo){
        int sumaDer, sumaIzq;
        if(nodo==null){
            return true;
        }
        if(nodo.getDer()==null){
            sumaDer=0;
        }
        else{sumaDer=sumarArboles(nodo.getDer());}
        if(nodo.getIzq()==null){
            sumaIzq=0;
        }
        else{sumaIzq=sumarArboles(nodo.getIzq());}
        if(nodo.getDer()==null && nodo.getIzq()==null){
            return true;
        }
        if(sumaIzq==sumaDer && comprobarSumas(nodo.getDer()) &&
```

```
comprobarSumas(nodo.getIzq())){
            return true;
        }
        else {
            return false;
        }

    }
    public int sumarArboles(NodoAB<Integer> nodo){
        int suma;
        suma= (int) nodo.getDato();
        if(nodo.getIzq()!=null){
            suma+=sumarArboles(nodo.getIzq());
        }
        if(nodo.getDer()!=null){
            suma+=sumarArboles(nodo.getDer());
        }
        return suma;
    }

    public boolean comprobarClavePequenia(){
        if(raiz == null){
            return true;
        }
        return comprobarClavePequenia(raiz);
    }
    private boolean comprobarClavePequenia(NodoAB<Integer> nodo) {
        if (nodo == null) {
            return true;
        }
        if (nodo.getIzq() != null && (int) nodo.getDato() > (int)
nodo.getIzq().getDato()) {
            return false;
        }

        if (nodo.getDer() != null && (int) nodo.getDato() > (int)
nodo.getDer().getDato()) {
            return false;
        }
        return comprobarClavePequenia(nodo.getIzq()) &&
comprobarClavePequenia(nodo.getDer());
    }
```

```java
    public int sumaNodosNiveles(int K1, int K2){
        int suma=0;
        if(esVacio()){
            return 0;
        }
        if(K1==0){
            suma+=(int)raiz.getDato();
            suma+= sumaNodosNiveles(K1, K2, raiz.getDer(), 1);
            suma+= sumaNodosNiveles(K1, K2, raiz.getIzq(), 1);
        }
        else{
            suma += sumaNodosNiveles(K1, K2, raiz.getDer(), 1);
            suma += sumaNodosNiveles(K1, K2, raiz.getIzq(), 1);
        }
        return + suma;
    }
    private int sumaNodosNiveles(int K1, int K2, NodoAB<Integer>
nodo, int K){
        int suma=0;
        if(nodo==null){
            return 0;
        }
        if(K>=K1 && K<=K2){
            K++;
            suma = (int)nodo.getDato();
            suma+=sumaNodosNiveles(K1, K2, nodo.getDer(), K) +
sumaNodosNiveles(K1, K2, nodo.getIzq(), K);
        }
        return suma;
    }

    public String mostrarAscendientes(int dato){
        String res = mostrarAscendientes(dato, raiz);
        if(res.isEmpty()){
            return "No se ha encontrado el nodo";
        }
        return res;
    }
    public String mostrarAscendientes(int dato, NodoAB<Integer>
nodo){
        if(nodo==null){
            return "";
```

```
        }
        if((int)nodo.getDato()==dato){
            return nodo.getDato().toString();
        }
        String izq = mostrarAscendientes(dato, nodo.getIzq());
        String der = mostrarAscendientes(dato, nodo.getDer());

        if(!izq.isEmpty()){
            return izq + " " + nodo.getDato().toString();
        }
        if(!der.isEmpty()){
            return der + " " + nodo.getDato().toString();
        }
        return "";
    }

    public int numeroNodosImpares(int nivel){
        if (esVacio()){
            return 0;
        }
        return numeroNodosImpares(nivel, raiz);
    }

    private int numeroNodosImpares(int nivel, NodoAB<Integer> nodo){
        if(nodo==null){
            return 0;
        }
        if(nivel==0){
            if((int)nodo.getDato()%2!=0){
                return 1;
            }
            else{
                return 0;
            }
        }
        return numeroNodosImpares(nivel-1, nodo.getIzq()) +
numeroNodosImpares(nivel-1, nodo.getDer());
    }
}
```

```java
package jerarquicos;

public class NodoAB<E> {
    private NodoAB<E> izq;
    private NodoAB<E> der;
    private E dato;
    private int tamanyo;
    public NodoAB(NodoAB<E> izq, NodoAB<E> der, E dato){
        this.dato = dato;
        this.izq = izq;
        this.der = der;
        tamanyo=1;
        if(izq!=null){
            tamanyo+= izq.tamanyo;
        }
        if(der!=null){
            tamanyo+= der.tamanyo;
        }
    }
    public NodoAB<E> getIzq() {return izq;}

    public NodoAB<E> getDer() {return der;}

    public E getDato() {return dato;}

    public int getTamanyo() {
        return tamanyo;
    }
    public void setIzq(NodoAB<E> n){
        izq=n;
    }
    public void setDer(NodoAB<E> n){
        der=n;
    }
}
```

```java
import jerarquicos.AB;
import jerarquicos.ABEnteros;
import jerarquicos.NodoAB;

import java.util.Scanner;
```

```
public class Menu {
    private ABEnteros<Integer> AB, AB1, AB2;
    private int opcion, opcionArbol;
    private Scanner sc;
    public Menu(){

    }
    public void ejecutar(){
        do{
            sc = new Scanner(System.in);
            System.out.println("\n\t\t MENÚ AB de Enteros\n");
            System.out.println("1. Crear AB de enteros\n");
            System.out.println("2. Listado de claves en InOrden");
            System.out.println("3. Listado de claves en InOrden
Converso");
            System.out.println("4. Listado de claves en PreOrden");
            System.out.println("5. Listado de claves en PostOrden");
            System.out.println("6. Comprobar sumas");
            System.out.println("7. Comprobar clave pequeña");
            System.out.println("8. Vaciar Árbol Modo 1 (sin recorrer
el árbol)");
            System.out.println("9. Vaciar Árbol Modo 2 (recorriendo
todos sus nodos)");
            System.out.println("10. Mostrar ascendientes");
            System.out.println("11. Suma total de nodos entre dos
niveles");
            System.out.println("12. Número de nodos impares que hay
en un nivel\n");
            System.out.println("0. Salir");
            System.out.println("Introduzca una opción:");
            opcion=sc.nextInt();
            switch (opcion){
                case 0:
                    System.out.println("Gracias por usar la
aplicación!");
                    break;
                case 1:
                    AB = construirArbol();
                    AB1 = new
ABEnteros<Integer>(AB.getRaiz().getDer());
                    AB2= new
```

```
ABEnteros<Integer>(AB.getRaiz().getIzq());
                    break;
                case 2:
                    System.out.println("Cual es el arbol que desea
considerar: 1-AB1, 2-AB2");
                    opcionArbol=sc.nextInt();
                    if(opcionArbol==1){
                        System.out.println(AB1.listadoInOrden());
                    }
                    else{
                        System.out.println(AB2.listadoInOrden());
                    }
                    break;
                case 3:
                    System.out.println("Cual es el arbol que desea
considerar: 1-AB1, 2-AB2");
                    opcionArbol=sc.nextInt();
                    if(opcionArbol==1){

System.out.println(AB1.listadoInOrdenConverso());
                    }
                    else{

System.out.println(AB2.listadoInOrdenConverso());
                    }
                    break;
                case 4:
                    System.out.println("Cual es el arbol que desea
considerar: 1-AB1, 2-AB2");
                    opcionArbol=sc.nextInt();
                    if(opcionArbol==1){
                        System.out.println(AB1.listadoPreOrden());
                    }
                    else{
                        System.out.println(AB2.listadoPreOrden());
                    }
                    break;
                case 5:
                    System.out.println("Cual es el arbol que desea
considerar: 1-AB1, 2-AB2");
                    opcionArbol=sc.nextInt();
                    if(opcionArbol==1){
```

```
                    System.out.println(AB1.listadoPostOrden());
                }
                else{
                    System.out.println(AB2.listadoPostOrden());
                }
                break;
            case 6:
                System.out.println("Cual es el arbol que desea
considerar: 1-AB1, 2-AB2");
                opcionArbol=sc.nextInt();
                if(opcionArbol==1){
                    System.out.println(AB1.comprobarSumas());
                }
                else{
                    System.out.println(AB2.comprobarSumas());
                }
                break;
            case 7:
                System.out.println("Cual es el arbol que desea
considerar: 1-AB1, 2-AB2");
                opcionArbol=sc.nextInt();
                if(opcionArbol==1){

System.out.println(AB1.comprobarClavePequenia());
                }
                else{

System.out.println(AB2.comprobarClavePequenia());
                }
                break;
            case 8:
                System.out.println("Cual es el arbol que desea
considerar: 1-AB1, 2-AB2");
                opcionArbol=sc.nextInt();
                if(opcionArbol==1){
                    AB1.vaciarArbolSinRecorrer();
                }
                else{
                    AB2.vaciarArbolSinRecorrer();
                }
                break;
            case 9:
```

```
                    System.out.println("Cual es el arbol que desea
considerar: 1-AB1, 2-AB2");
                    opcionArbol=sc.nextInt();
                    if(opcionArbol==1){
                        AB1.vaciarArbolRecorriendo();
                    }
                    else{
                        AB2.vaciarArbolRecorriendo();
                    }
                    break;
                case 10:
                    System.out.println("Cual es el arbol que desea
considerar: 1-AB1, 2-AB2");
                    opcionArbol=sc.nextInt();
                    System.out.println("Introduzca la clave del
nodo:");
                    int clave = sc.nextInt();
                    if(opcionArbol==1){
                        System.out.println("En el arbol " + "AB" +
opcionArbol + " los ascendientes del nodo " + clave + " son: " +
AB1.mostrarAscendientes(clave));
                    }
                    else{
                        System.out.println("En el arbol " + "AB" +
opcionArbol + " los ascendientes del nodo " + clave + " son: " +
AB2.mostrarAscendientes(clave));
                    }
                    break;
                case 11:
                    System.out.println("Cual es el arbol que desea
considerar: 1-AB1, 2-AB2");
                    opcionArbol=sc.nextInt();
                    System.out.println("Introduzca K1:");
                    int K1 = sc.nextInt();
                    System.out.println("Introduzca K2:");
                    int K2 = sc.nextInt();
                    if(opcionArbol==1){
                        System.out.println("La suma de elementos
enre estos niveles es:" + AB1.sumaNodosNiveles(K1, K2));
                    }
                    else{
                        System.out.println("La suma de elementos
```

```
enre estos niveles es:" + AB2.sumaNodosNiveles(K1, K2));
                    }
                    break;
                case 12:
                    System.out.println("Cual es el arbol que desea
considerar: 1-AB1, 2-AB2");
                    opcionArbol=sc.nextInt();
                    System.out.println("Introduzca el nivel:");
                    int nivel = sc.nextInt();
                    if(opcionArbol==1){
                        System.out.println("El numero de nodos
impares en el arbol " + "AB" + opcionArbol + " y el nivel " + nivel
+ " es: " + AB1.numeroNodosImpares(nivel));
                    }
                    else{
                        System.out.println("El numero de nodos
impares en el arbol " + "AB" + opcionArbol + " y el nivel " + nivel
+ " es: " + AB2.numeroNodosImpares(nivel));
                    }
                    break;
                default:
                    System.out.println("Opción no válida");
                    break;

            }
        }while(opcion!=0);

    }
    public static ABEnteros<Integer> construirArbol(){
        ABEnteros<Integer> AB;
        NodoAB<Integer> n0, n1, n2, n3, n4, n5, n6, n7, n8, n9, n10,
n11, n12, n13, n14, n15, n16;
        n1 = new NodoAB<Integer>(null, null, 2);
        n2 = new NodoAB<Integer>(null, null, 2);
        n3 = new NodoAB<Integer>(n1, n2, 4);
        n4 = new NodoAB<Integer>(null, null, 8);
        n5 = new NodoAB<Integer>(n4, n3, 12);
        n6 = new NodoAB<Integer>(null, null, 0);
        n7 = new NodoAB<Integer>(n6, null, -3);
        n8 = new NodoAB<Integer>(null, null, -3);
        n9 = new NodoAB<Integer>(n7, n8, 34);
        n10 = new NodoAB<Integer>(n5, n9, 13);
```

```
        n11= new NodoAB<Integer>(null,null, 8);
        n12 = new NodoAB<Integer>(null, n11, 5);
        n13= new NodoAB<Integer>(null, null, 9);
        n14 = new NodoAB<Integer>(null, null, 7);
        n15 = new NodoAB<Integer>(n13, n14, 6);
        n16= new NodoAB<Integer>(n15, n12, 2);
        n0 = new NodoAB<Integer>(n10, n16, 0);
        AB = new ABEnteros<Integer>(n0);
        return AB;

    }
}

import jerarquicos.AB;
import jerarquicos.ABEnteros;
import jerarquicos.NodoAB;

public class Main {
    public static void main(String[] args) {
        Menu menu = new Menu();
        menu.ejecutar();
    }
}
```