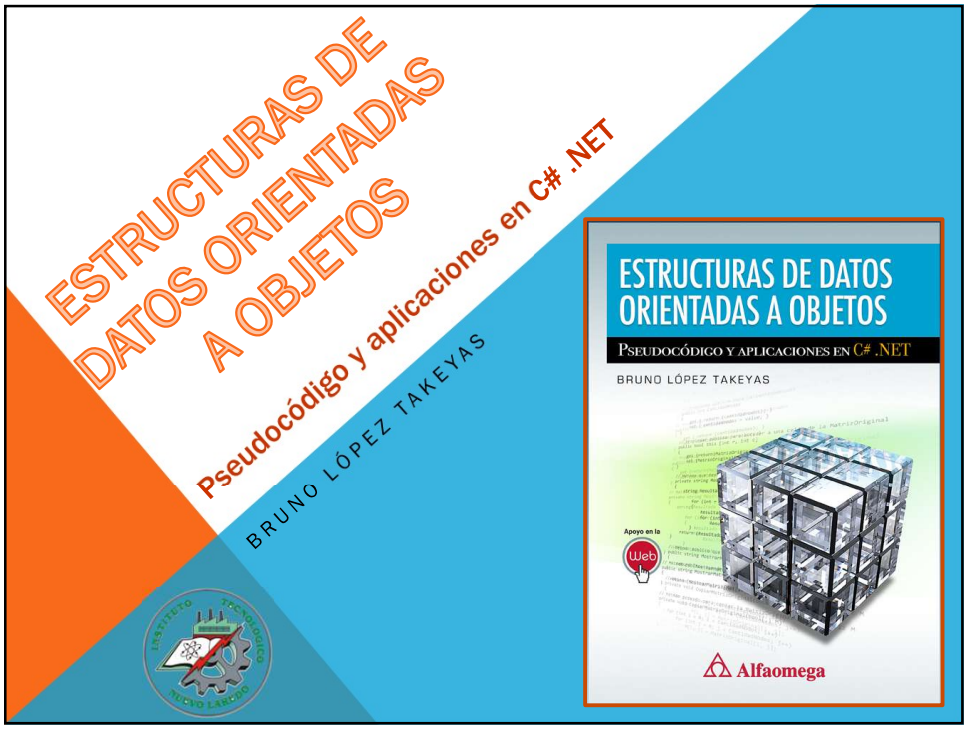
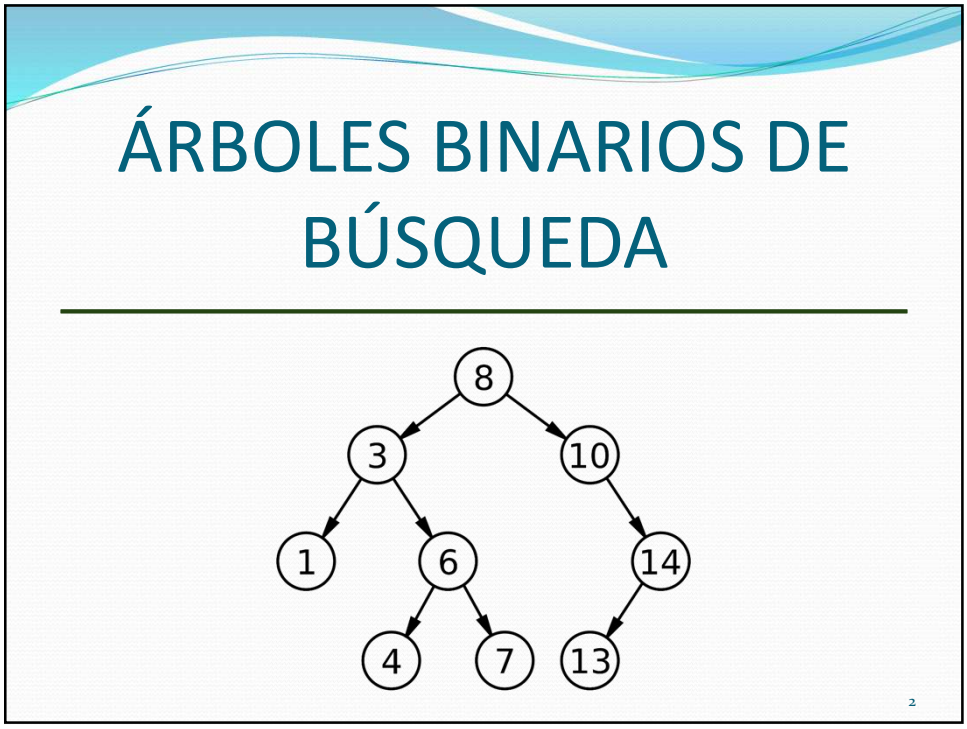

ESTRUCTURAS DE DATOS ORIENTADAS A OBJETOS
Pseudocódigo y aplicaciones en C# .NET
BRUNO LÓPEZ TAKEYAS
ESTRUCTURAS DE DATOS ORIENTADAS A OBJETOS
PSEUDOCÓDIGO Y APLICACIONES EN C# .NET
BRUNO LÓPEZ TAKEYAS
Alfaomega


ÁRBOLES BINARIOS DE BÚSQUEDA
8
3
10
1
6
14
4
7
13
2

Preguntas detonadoras

- ¿Qué es un árbol?
- ¿Cómo se clasifican los árboles?
- ¿Qué características distinguen a un árbol binario?
- ¿ … y a un árbol binario de búsqueda?
- ¿Cómo se representa gráficamente un árbol binario de búsqueda?
- ¿Qué operaciones se pueden realizar en un árbol binario de búsqueda?
- ¿Cómo se diseña un modelo orientado a objetos con un árbol binario de búsqueda?
- ¿Cómo se puede dibujar una estructura de datos?

3

Clasificaciones de ED

ED CLASIFICADAS POR TIPO DE RECORRIDO

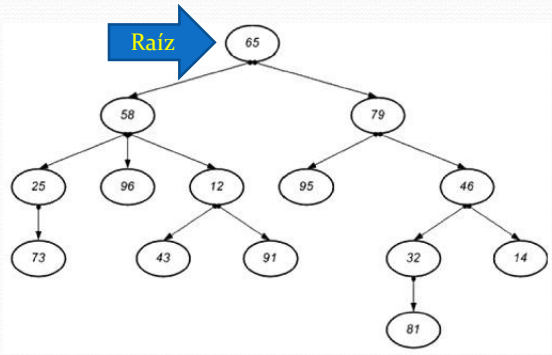
Lineales
- *Arreglos*
- *Pilas*
- *Colas*
- *Listas enlazadas simples*
- *Listas enlazadas dobles*

No lineales
- *Árboles*
- *Grafos*

4

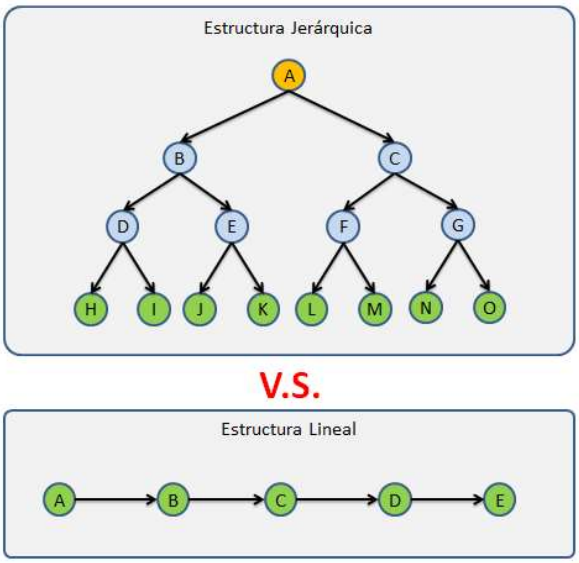
Árbol

Es una estructura de datos jerárquica, dinámica y no lineal que se compone de un conjunto de nodos y arcos, además tiene un apuntador inicial llamado "raíz"



5

Estructura jerárquica



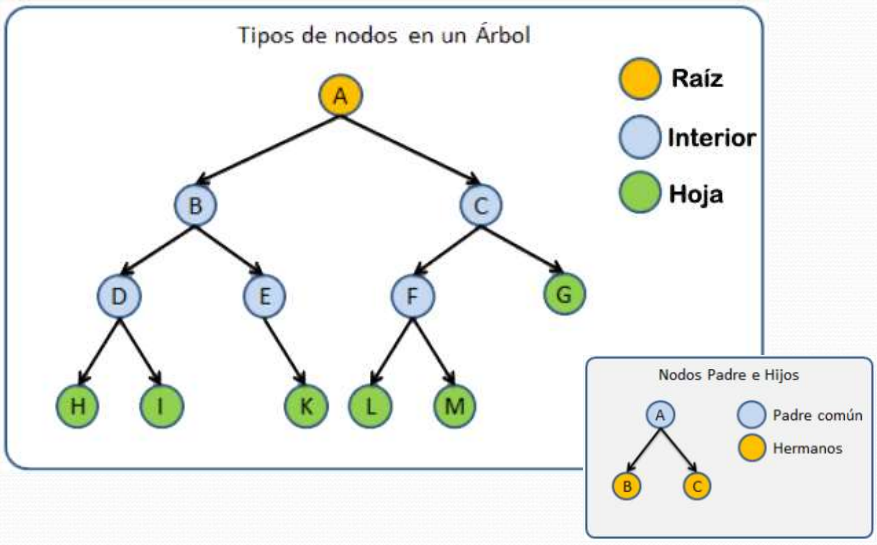
6

Conceptos

- Raíz: *Nodo inicial del recorrido del árbol*
- Padre: *Nodo antecesor de otro*
- Hijo: *Nodo sucesor o descendiente*
- Hermano: *Nodos sucesores del mismo padre*
- Hoja: *Nodo sin hijos*
- Interior: *Conjunto de nodos que no son raíz ni hojas*

7

Tipos de nodos

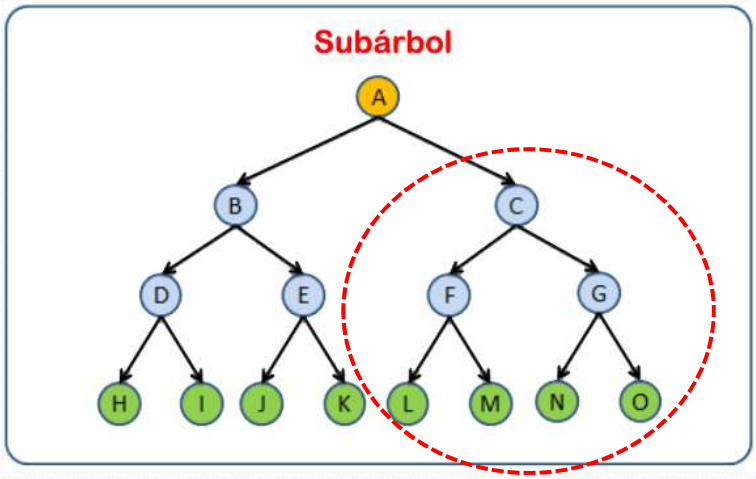


Conceptos (*cont.*)

- Subárbol: *Es una sección jerárquica de un árbol*
- Nivel de un nodo: *Cantidad de arcos recorridos+1 para llegar a un nodo. La raíz tiene nivel 1.*
- Altura del árbol: *Representa el valor más grande de los niveles de los nodos*
- Grado de un nodo: *Cantidad de hijos de un nodo*
- Grado de un árbol: *Representa el valor más grande de los grados de un nodo*

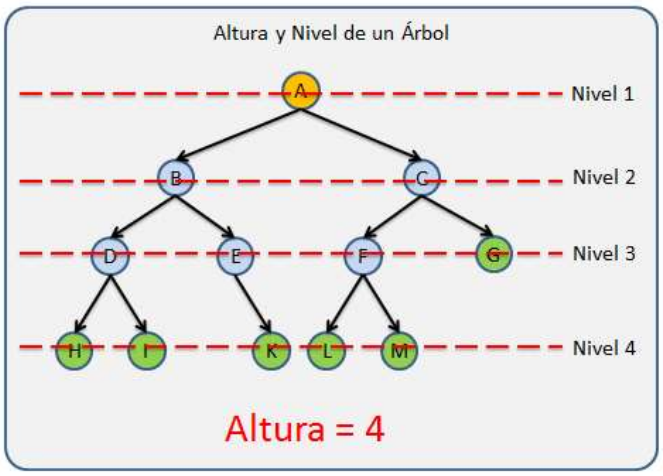
9

Subárbol



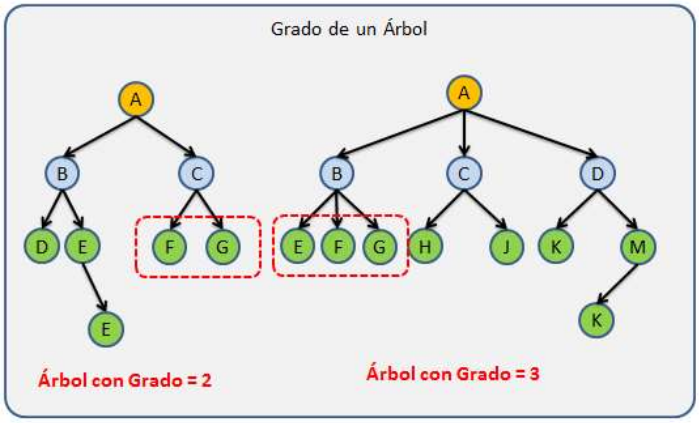
10

Altura y niveles de un árbol



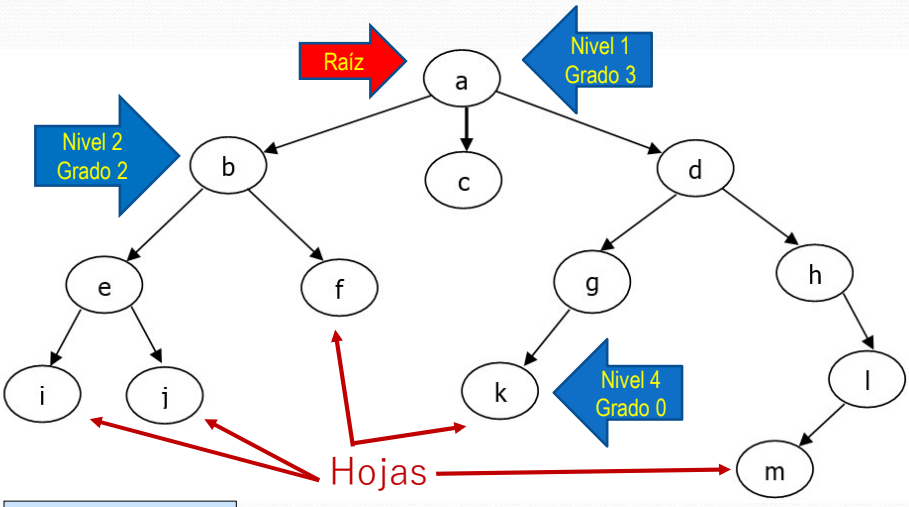
11

Grado de un árbol



12

Conceptos



Altura del árbol: 5
Grado del árbol: 3

13

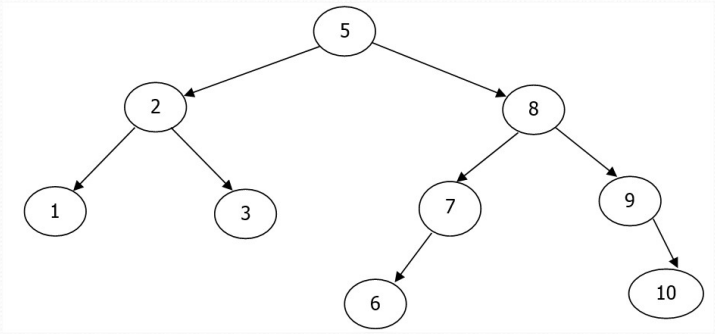
Tipos de árboles

- Generales.- *Árboles de grado > 2*
- Binarios.- *Árboles de grado 2.*
- Binarios de búsqueda.- *Árboles binarios donde los hijos izquierdos son menores y los derechos con mayores que el padre.*
- AVL.- *Árboles binarios de búsqueda autobalanceables*
- Roji-negros.- *Árbol binario de búsqueda equilibrado*
- AA.- *Es una variante del árbol roji-negro que agrega nodos rojos solamente por la derecha*
- B.- *Árboles balanceados de búsqueda de grado > 2*
- B+.- *Variante del árbol B que indexa datos*
- B*.- *Variante del árbol B que optimiza el acomodo de los nodos*

14

Árbol binario de búsqueda

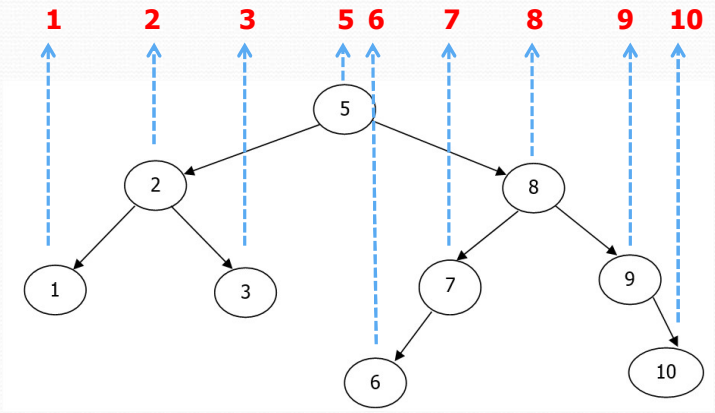
Es un árbol binario (grado 2) en el que el hijo izquierdo es menor y el hijo derecho es mayor que su padre. Esta condición se cumple para todos los nodos.



15

Ordenamiento de datos en un ABB

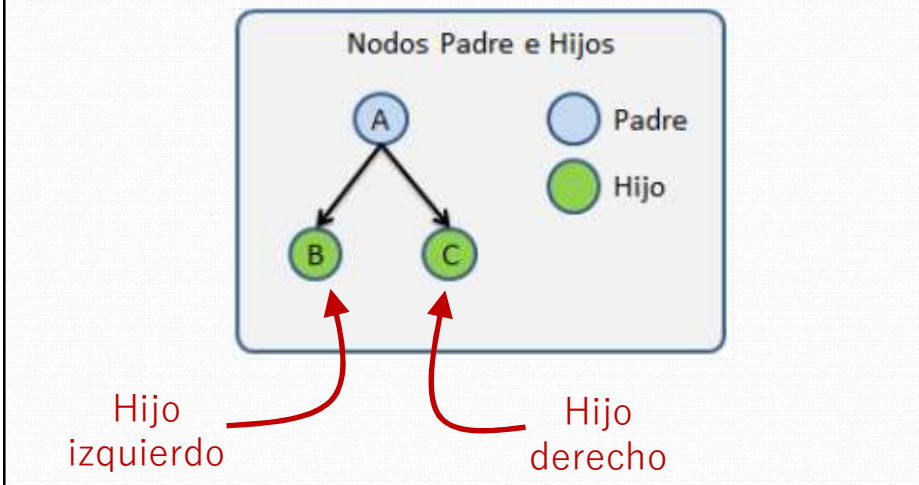
El ABB mantiene los datos ordenados, aunque no de la forma tradicional sino mediante una estructura binaria



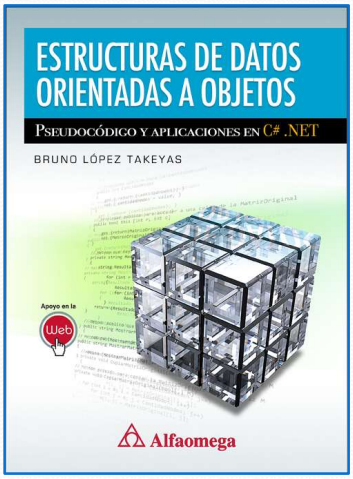
16

Ejemplo de árboles binarios en la vida cotidiana

Hijos en árboles binarios



Lectura



Para reforzar este tema se recomienda la lectura de:

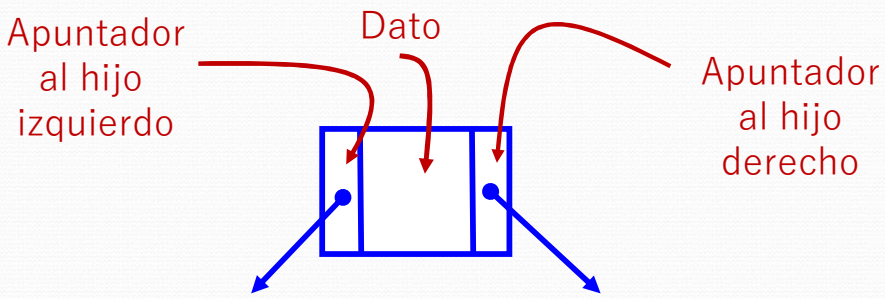
Capítulo 8.- Árboles binarios

19

Arquitectura de un nodo

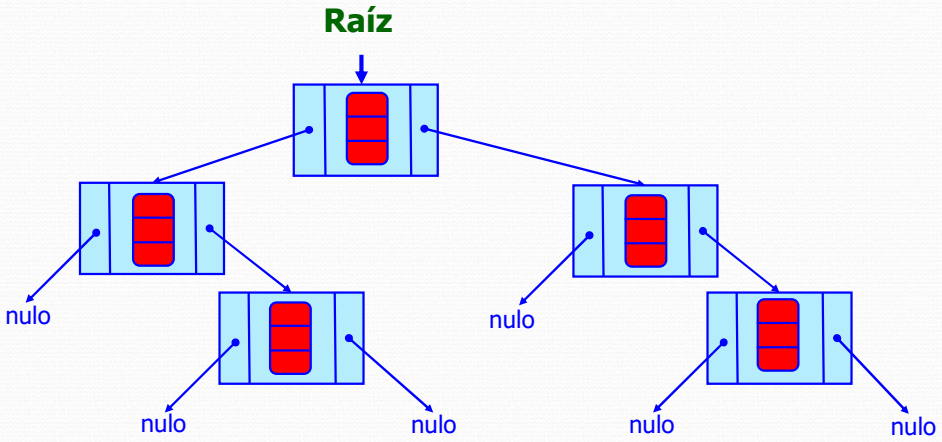
Cada nodo tiene 3 secciones:

1. *Dato (puede ser simple o compuesto)*
2. *Apuntador o referencia que enlaza al hijo izquierdo*
3. *Apuntador que enlaza al hijo derecho*



20

Representación de un ABB



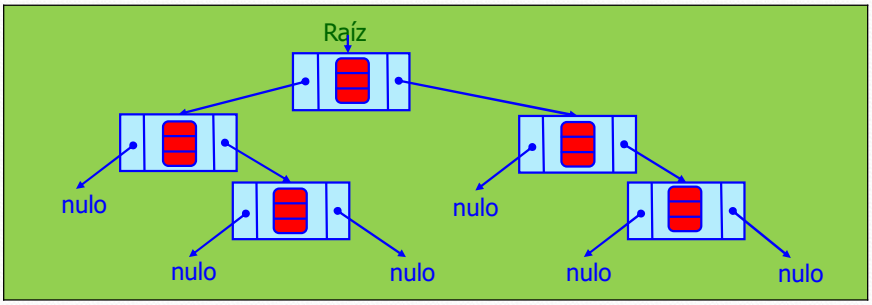
21

Diseño de un ABB orientado a objetos

- Se identifican 3 tipos de objetos en el ABB:
  1) *Los objetos con los datos que se desean almacenar y ordenar*
  2) *Los objetos de los nodos*
  3) *El objeto del ABB*

22

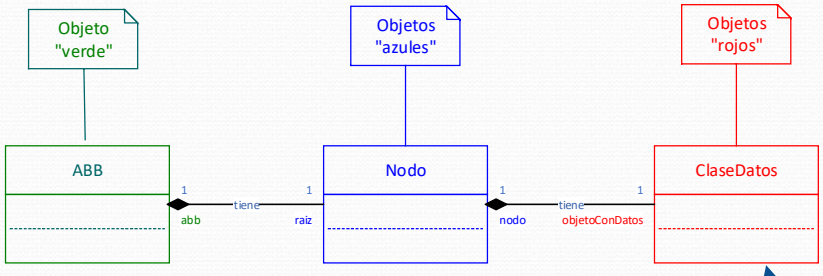
Esquema de los objetos



El **ABB (objeto verde)** dentro tiene una secuencia lógica de **nodos (objetos azules)** enlazados por apuntadores y cada uno de ellos a su vez tiene dentro un **objeto con los datos** que se desean almacenar y ordenar **(objetos rojos)**

23

Diseño genérico y didáctico del ABB



- *Esta clase puede ser...*
- `Empleado`
- `Escuela`
- `Médico`
- *etc. (según lo que se desee almacenar en el ABB)*

24

Diseño orientado a objetos de un ABB

- Se diseña un ABB con objetos creados por medio de varios tipos de clases:
  - *Clase "roja".- Sirve para crear objetos con los datos que se desean almacenar en el ABB*
  - *Clase "azul".- Clase para crear los nodos*
  - *Clase "verde".- Clase para crear el objeto del ABB*

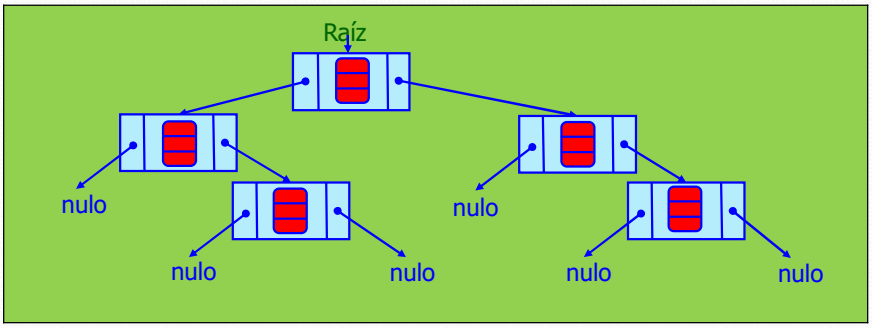
25

Notación de colores

- Los estudiantes deben poner especial atención a los colores usados en las figuras explicativas del resto del curso.
- Los objetos estarán identificados por colores
  - *Objetos "rojos".- Contienen los datos que se desean almacenar y ordenar en el ABB*
  - *Objetos "azules".- Representan los nodos del ABB*
  - *Objeto "verde".- Es el ABB*

26

Diseño de clases



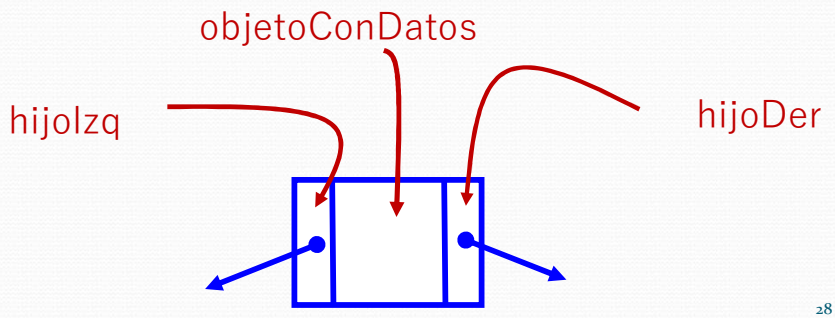
- **Clase "verde"**.- *Define la **composición** entre el ABB y la raíz. También tiene los métodos y propiedades para administrar el ABB.*
- **Clase "azul"**.- *Define los componentes de los nodos.*
- **Clase "roja"**.- *Definiciones de los datos que se desean almacenar y ordenar en el ABB*

27

Declaración del nodo en al ABB

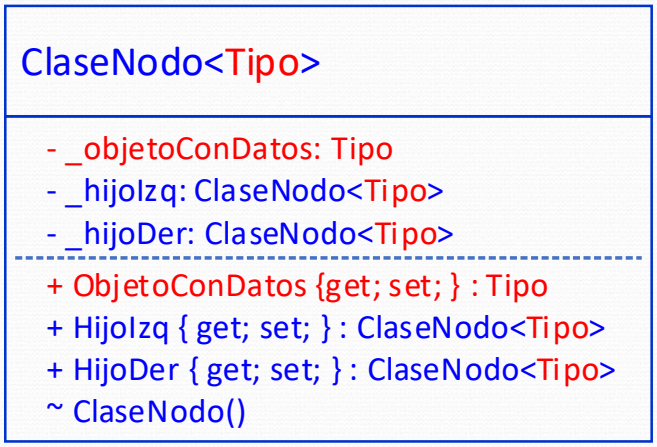
Cada nodo es un obtiene 3 atributos (con sus propiedades):

1. ***objetoConDatos***.- *El nodo recibirá un objeto con los datos que se desean almacenar en el ABB*
2. ***hijoIzq***.- *Apuntador que enlaza al hijo izquierdo*
3. ***hijoDer***.- *Apuntador que enlaza al hijo derecho*



28

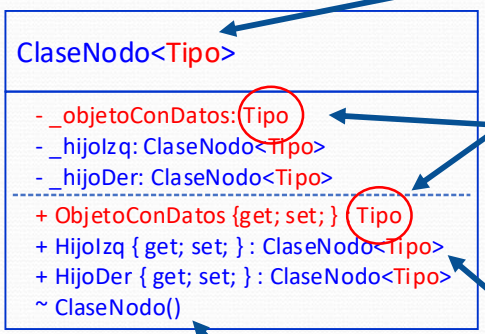
Diseño de la clase de los nodos



29

Diseño de la `ClaseNodo`

- *Clase parametrizada para recibir cualquier tipo de objeto "rojo"*
- *El parámetro `<Tipo>` define el tipo de objeto "rojo" que estará dentro de cada nodo "azul"*
- *Los apuntadores `HijoIzq` e `HijoDer` NO almacenan otro nodo "azul" sino apuntan hacia otro nodo de su mismo tipo*

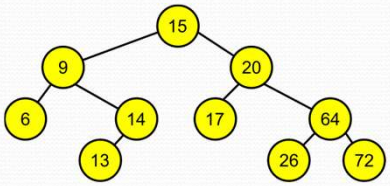


- *Al eliminar un nodo "azul", su destructor elimina el objeto "rojo" que contiene*

30

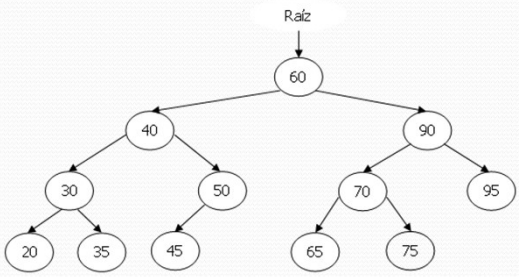
Operaciones en un ABB

- *Creación del ABB*
- *Inserción de un dato*
- *Eliminación de un dato*
- *Recorridos*
- *Búsqueda*
- *Eliminar todos los nodos (vaciar)*



31

Búsqueda de un dato en un ABB



- *Se verifica que el ABB no esté vacío*
- *El recorrido empieza en la* `Raíz`
- *Si el dato es menor entonces se avanza al próximo nodo a través del apuntador* `HijoIzq` *de lo contrario se avanza hacia el* `HijoDer`
- *El recorrido termina al llegar a una hoja (no tiene hijos) o al encontrar el dato solicitado*

32

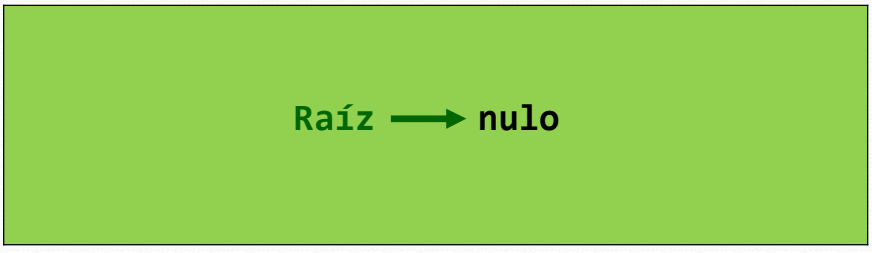
Situaciones críticas

- *Son las situaciones que se pueden presentar al realizar operaciones con estructuras de datos*
- *El programador debe prever para diseñar algoritmos eficientes*

33

Creación de un ABB

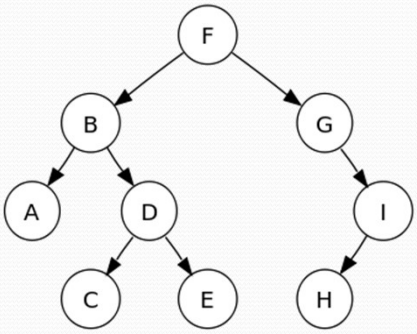
Cuando se crea un ABB la raíz apunta a nulo



34

Inserción de datos en un ABB

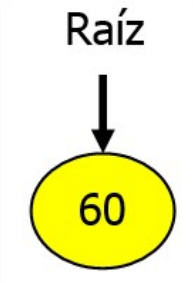
- Situaciones críticas:
  - *Alta en un ABB vacío*
  - *Alta por la izquierda*
  - *Alta por la derecha*
  - *Alta izquierda-derecha*
  - *Alta derecha-izquierda*
  - *Alta combinada*
  - ***NO** se permiten duplicados*



35

Alta en un ABB vacío

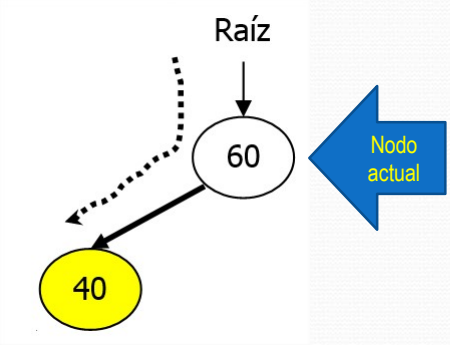
Sucede cuando se inserta el primer nodo del ABB



36

Alta por la izquierda

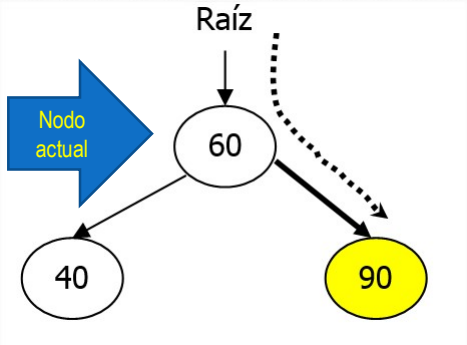
Sucede cuando se inserta un nodo menor que el nodo actual, el cual provoca avanzar hacia la izquierda del ABB



37

Alta por la derecha

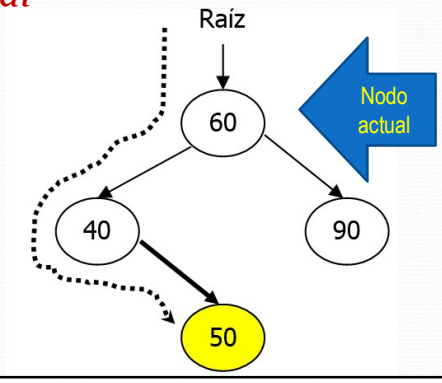
Sucede cuando se inserta un nodo mayor que el nodo actual, el cual provoca avanzar hacia la derecha del ABB



38

Alta izquierda-derecha

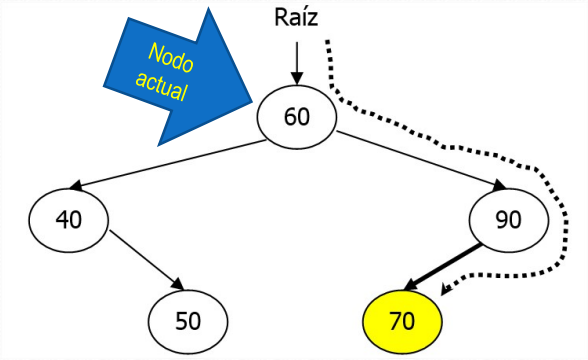
Ocurre cuando se inserta un nodo mayor que el hijo izquierdo del nodo actual, el cual provoca avanzar hacia la derecha del subárbol izquierdo del nodo actual



39

Alta derecha-izquierda

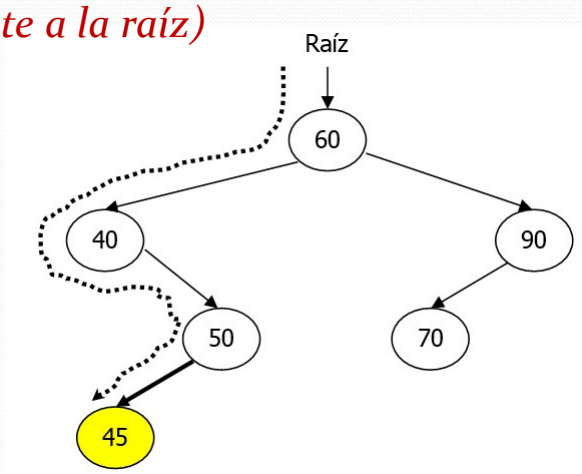
Ocurre cuando se inserta un nodo menor que el hijo derecho del nodo actual, el cual provoca avanzar hacia la izquierda del subárbol derecho del nodo actual



40

Alta combinada izquierda-derecha-izquierda

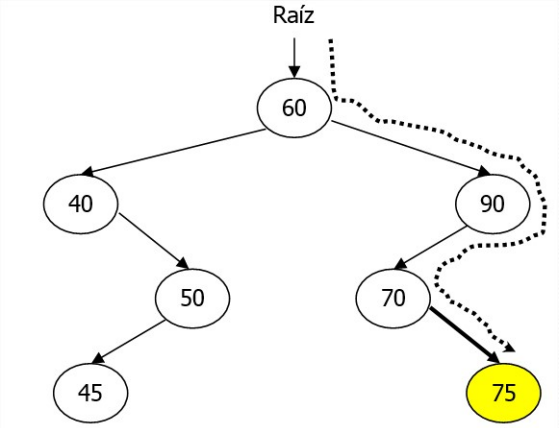
Proceso aplicable a cualquier nodo (no solamente a la raíz)



41

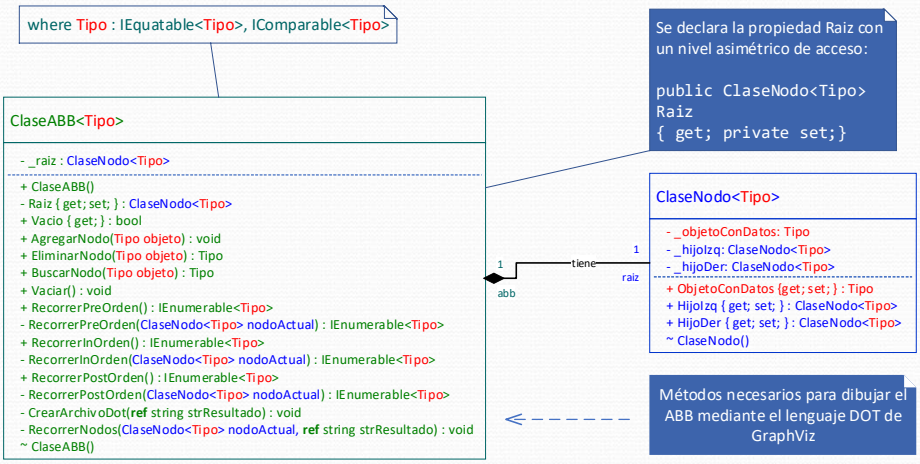
Alta combinada derecha-izquierda-derecha

Proceso aplicable a cualquier nodo (no solamente a la raíz)



42

Diseño de la clase del ABB



43

Composición ABB-Nodo

- *¿Por qué es una composición **1..1** si el ABB tiene muchos nodos dentro de él?*
  - *Porque la cardinalidad de una composición se define por la cantidad de atributos de tipo "parte" contenidos en la clase del "todo" (regla 1 de la composición)*

44

ClaseABB<Tipo>

- *Clase parametrizada para prepararla para que pueda recibir cualquier `Tipo` de objeto "rojo"*
- *Tiene una restricción de tipos para "obligar" a que la clase "roja" implemente `IEquatable` e `IComparable`*
  - *Se requiere el método `Equals()` para buscar un objeto "rojo" almacenado en el ABB*
  - *Se requiere el método `CompareTo()` para comparar objetos "rojos" y ordenarlos en el ABB*

45

Componentes de la clase

- `_raiz`.- *Atributo privado que apunta al primer nodo del ABB*
- `ClaseABB()`.- *Es el constructor que inicializa vacío el ABB*
- `Vacio`.- *Propiedad pública booleana de solo lectura para detectar si el ABB está vacío (devuelve `true` cuando el ABB está vacío).*

46

Componentes de la clase *(cont.)*

- `Raiz`.- *Propiedad con acceso asimétrico (public get y private set) que apunta al primer nodo del ABB*
- `AgregarNodo(Tipo objeto):void` .- *Método público que recibe como parámetro el objeto "rojo" que se desea almacenar en el ABB*
- `EliminarNodo(Tipo objeto):Tipo` .- *Método público que recibe como parámetro el objeto "rojo" que se desea borrar del ABB.* ***Devuelve*** *el objeto "rojo" eliminado.*

47

Componentes de la clase *(cont.)*

- `BuscarNodo(Tipo objeto):Tipo` .- *Método público que recibe como parámetro el objeto "rojo" que se desea consultar en el ABB. Devuelve el objeto "rojo" localizado.*
- `Vaciar():void` .- *Método público que recorre el ABB para eliminar todos los nodos "azules" con sus respectivos objetos "rojos"*

48

Componentes de la clase *(cont.)*

- `~ClaseABB()`.- *Destructor que invoca al método* `Vaciar()` *para eliminar todos los* `nodos` *del ABB.*

49

Componentes de la clase *(cont.)*

- La clase del ABB también contiene:
  - *Iteradores recursivos*
  - *Método recursivo para vaciar el ABB*
- Métodos requeridos para dibujar el ABB:
  - *CrearArchivoDot()*
  - *RecorrerNodos()*

50

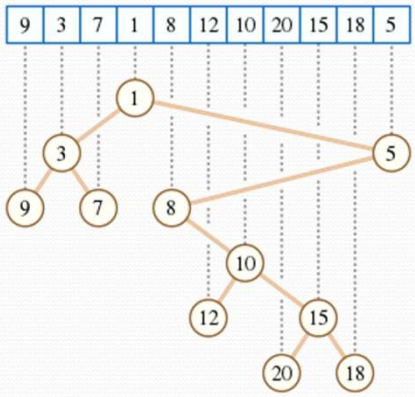
Tarea 3.02.- DF ABB (constructor y vacía)

- Subir a MS Teams el archivo con diagramas de flujo de:
  - *Constructor del ABB*
  - *Propiedad para detectar si el ABB está vacío*

51

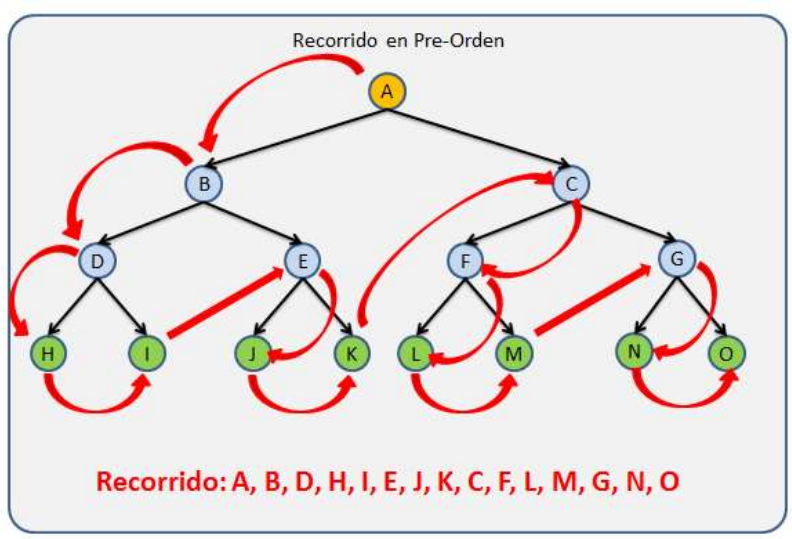
Recorridos en un ABB

- *Existen varias formas de recorrer un árbol (en general)*
- *En profundidad*
- *En anchura*
- *PreOrden*
- *InOrden*
- *PostOrden*



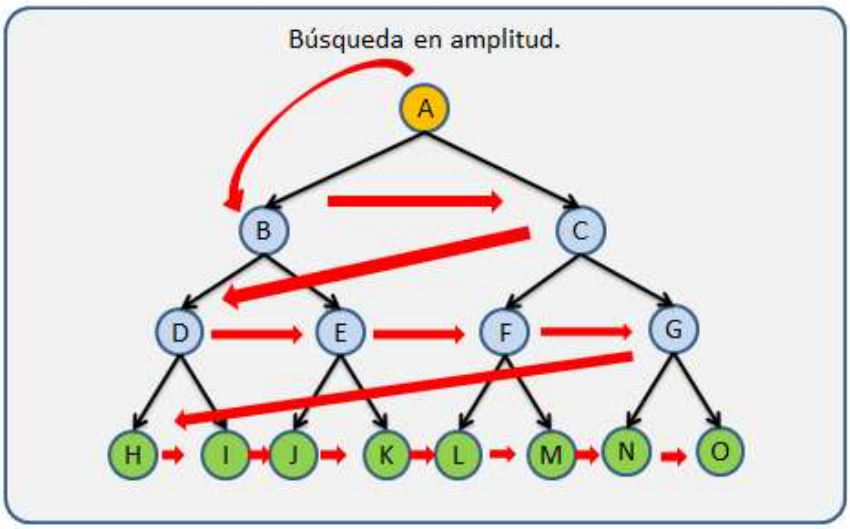
52

Recorrido en profundidad



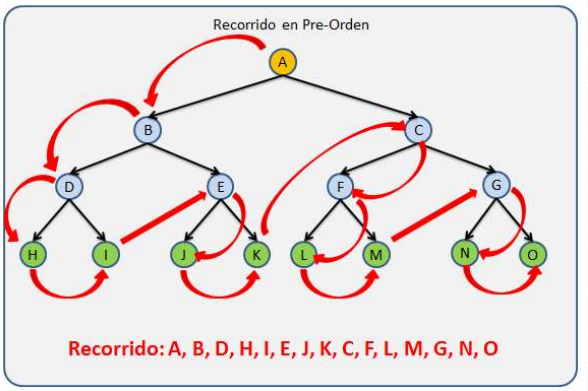
53

Recorrido en anchura o amplitud



Recorrido PreOrden

1) *Mostrar nodoActual*
2) *Recorrer hijos izquierdos*
3) *Recorrer hijos derechos*



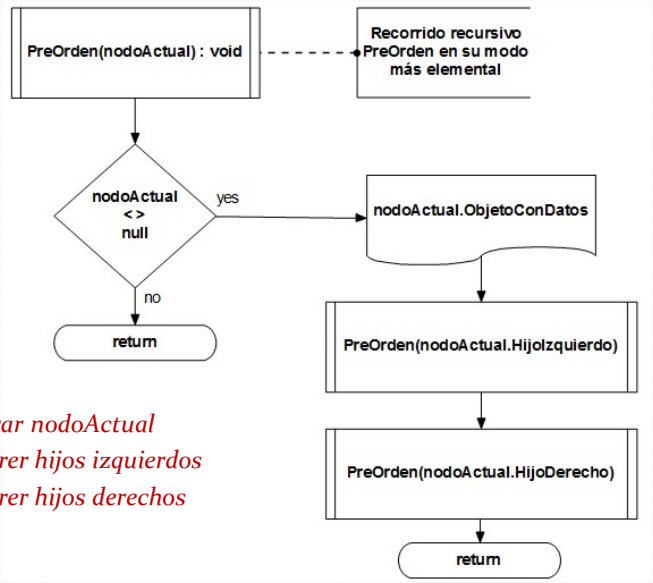
55

Pseudocódigo PreOrden

```
PreOrden(nodoActual): nulo

/*Método recursivo para recorrer el ABB en modo PreOrden*/

1.- SI nodoActual ≠ nulo ENTONCES
  1.1. Mostrar nodoActual.Dato
  1.2. PreOrden(nodoActual.HijoIzq)
  1.3. PreOrden(nodoActual.HijoDer)
2.- {FIN DE LA CONDICIONAL DEL PASO 1}
```

56

Diagrama de flujo PreOrden básico

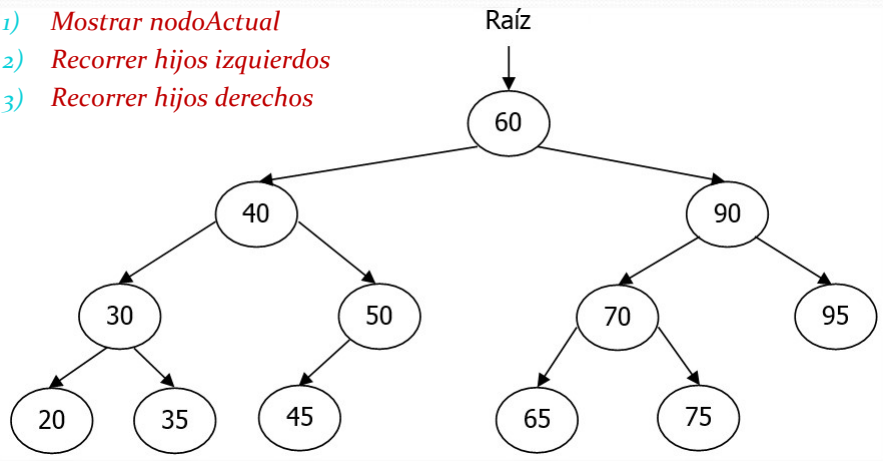


1) *Mostrar nodoActual*
2) *Recorrer hijos izquierdos*
3) *Recorrer hijos derechos*

57

Ejemplo PreOrden

1) *Mostrar nodoActual*
2) *Recorrer hijos izquierdos*
3) *Recorrer hijos derechos*

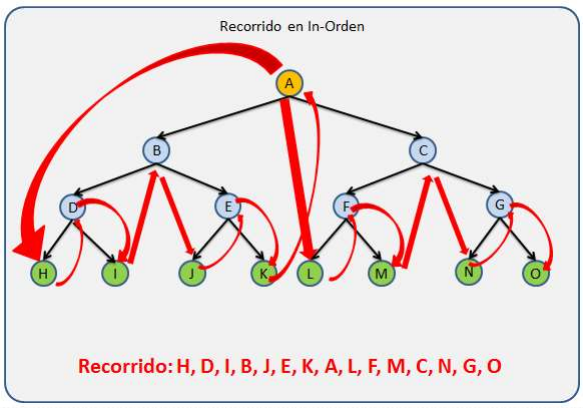


Salida: 60, 40, 30, 20, 35, 50, 45, 90, 70, 65, 75 y 95.

58

Recorrido InOrden

1) *Recorrer hijos izquierdos*
2) *Mostrar nodo actual*
3) *Recorrer hijos derechos*



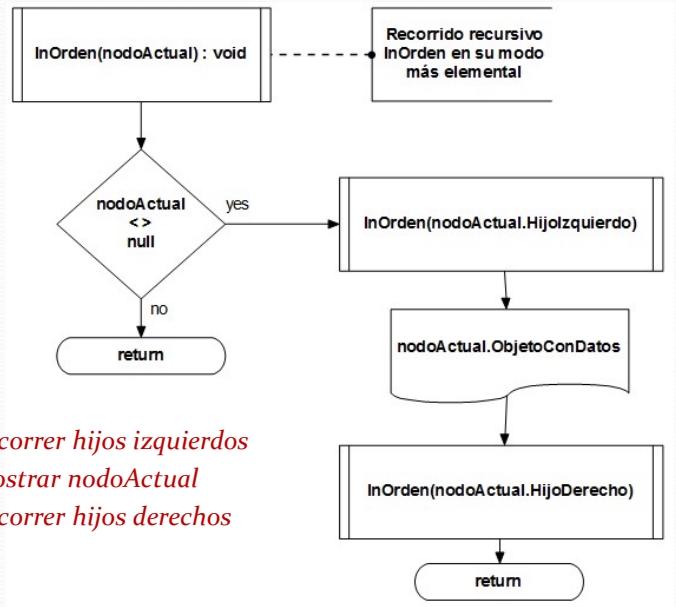
59

Pseudocódigo InOrden

```
InOrden(nodoActual): nulo

/* Método recursivo para recorrer el ABB en modo InOrden */

1.- SI nodoActual ≠ nulo ENTONCES
  1.1. InOrden(nodoActual.HijoIzq)
  1.2. Mostrar nodoActual.Dato
  1.3. InOrden(nodoActual.HijoDer)
2.- {FIN DE LA CONDICIONAL DEL PASO 1}
```

60

Diagrama de flujo InOrden básico

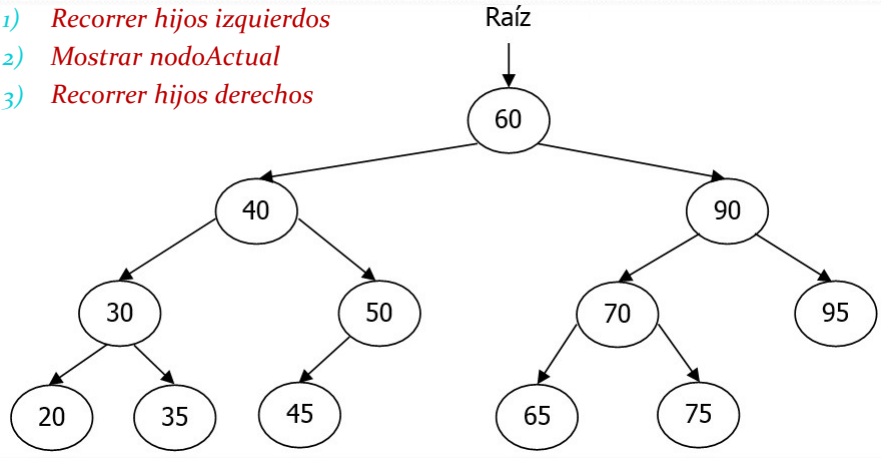


1) *Recorrer hijos izquierdos*
2) *Mostrar nodoActual*
3) *Recorrer hijos derechos*

61

Ejemplo InOrden

1) *Recorrer hijos izquierdos*
2) *Mostrar nodoActual*
3) *Recorrer hijos derechos*

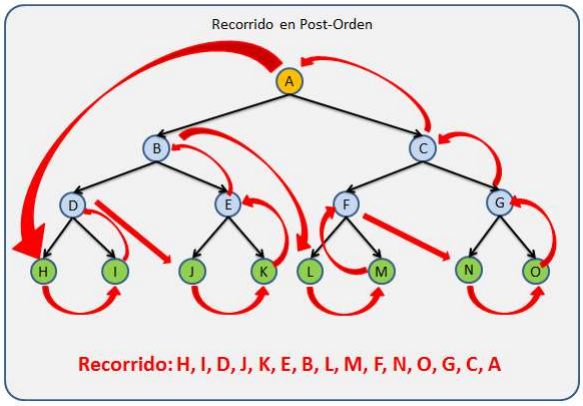


Salida: 20, 30, 35, 40, 45, 50, 60, 65, 70, 75, 90 y 95

62

Recorrido PostOrden

1) *Recorrer hijos izquierdos*
2) *Recorrer hijos derechos*
3) *Mostrar nodo actual*



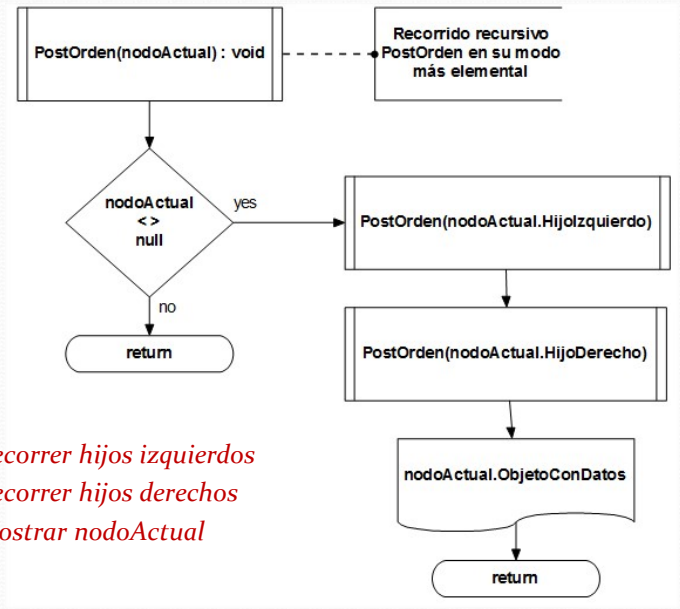
63

Pseudocódigo PostOrden

```
PostOrden(nodoActual): nulo

/*Método recursivo para recorrer el ABB en modo PostOrden*/

1.- SI nodoActual ≠ nulo ENTONCES
  1.1. PostOrden(nodoActual.HijoIzq)
  1.2. PostOrden(nodoActual.HijoDer)
  1.3. Mostrar nodoActual.Dato
2.- {FIN DE LA CONDICIONAL DEL PASO 1}
```

64

Diagrama de flujo PostOrden básico

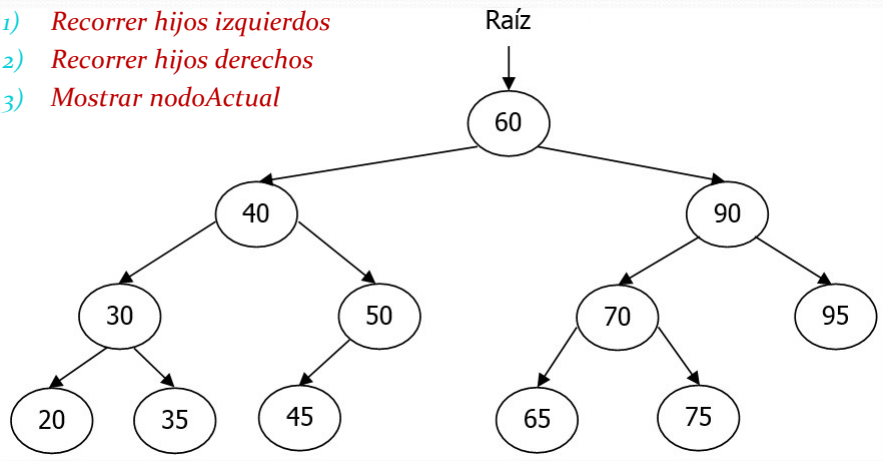


1) *Recorrer hijos izquierdos*
2) *Recorrer hijos derechos*
3) *Mostrar nodoActual*

65

Ejemplo PostOrden

1) *Recorrer hijos izquierdos*
2) *Recorrer hijos derechos*
3) *Mostrar nodoActual*



Salida: 20, 35, 30, 45, 50, 40, 65, 75, 70, 95, 90 y 60

66

Iteradores recursivos del ABB

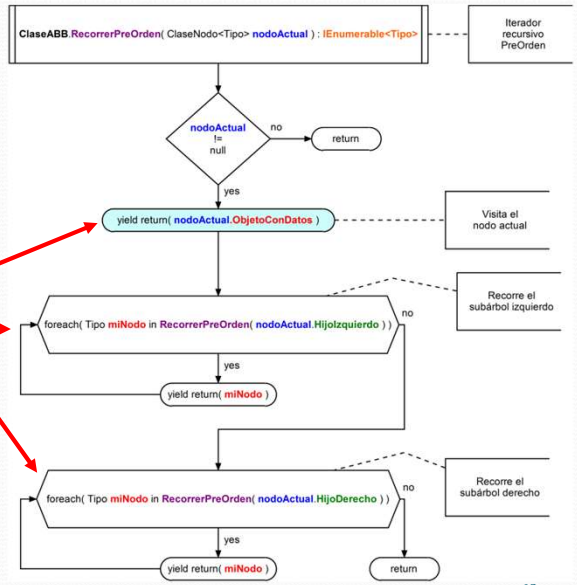
- *Debido a que el ABB se recorre de múltiples maneras entonces se requieren varios iteradores recursivos.*

- *Se implementan a través de métodos **privados***
  - `RecorrerPreOrden(ClaseNodo<Tipo> nodoActual):IEnumerable<Tipo>`
  - `RecorrerInOrden(ClaseNodo<Tipo> nodoActual):IEnumerable<Tipo>`
  - `RecorrerPostOrden(ClaseNodo<Tipo> nodoActual):IEnumerable<Tipo>`

67

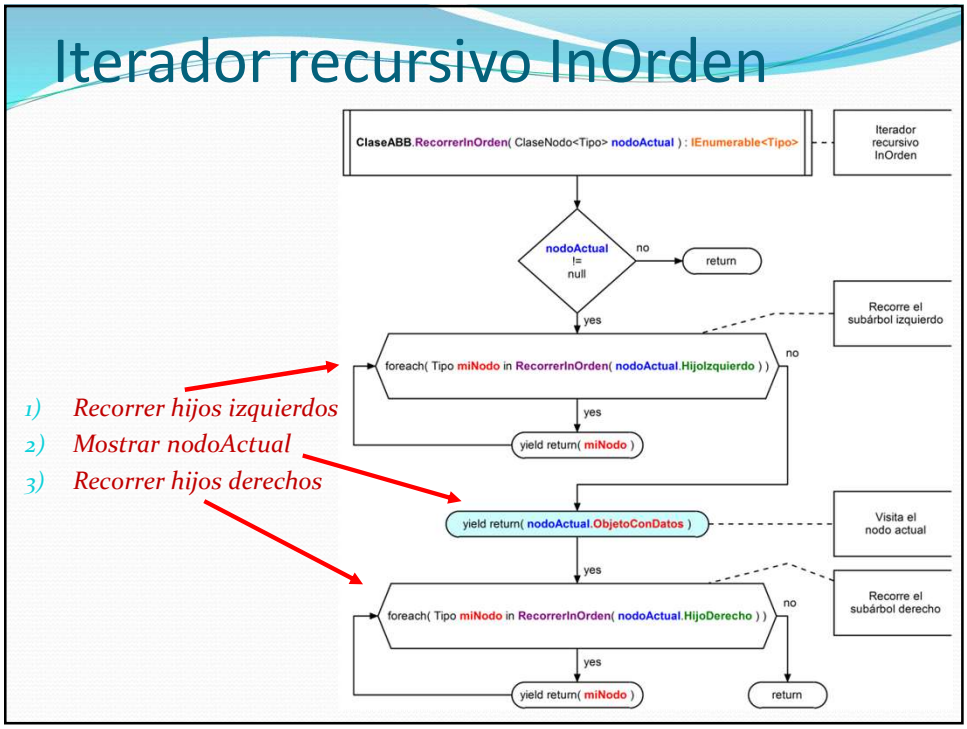
Iterador recursivo PreOrden

1) *Mostrar nodoActual*
2) *Recorrer hijos izquierdos*
3) *Recorrer hijos derechos*

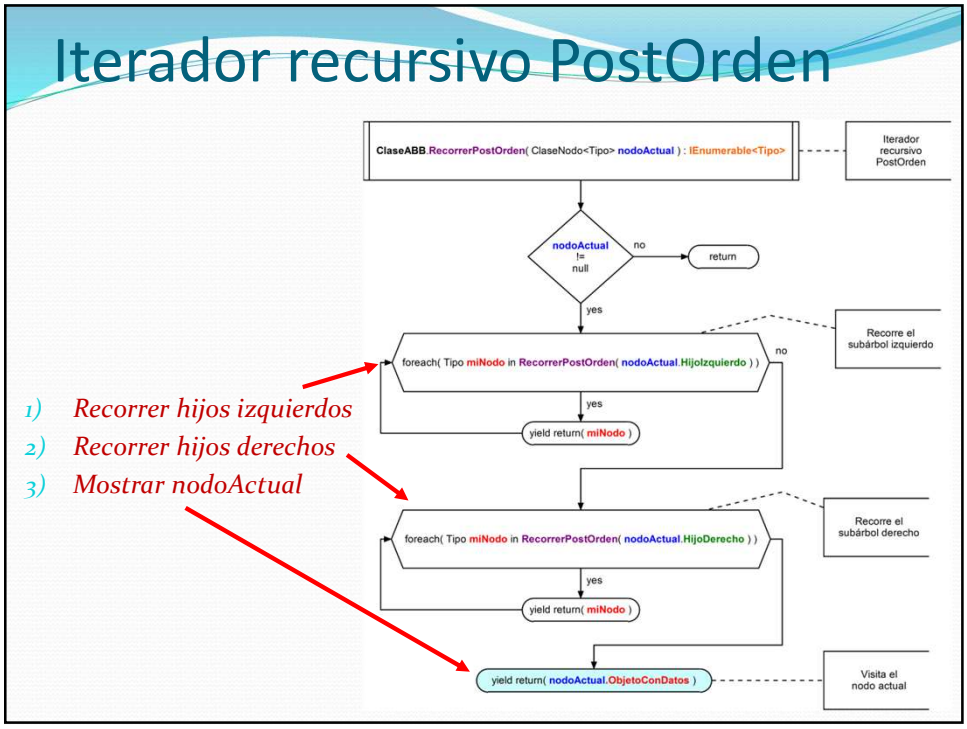


68

Iterador recursivo InOrden



Iterador recursivo PostOrden



¿Cómo invocar a los iteradores recursivos?

- *Los iteradores se invocan a través del ciclo **foreach***
- *La clase ABB tiene los siguientes iteradores públicos:*

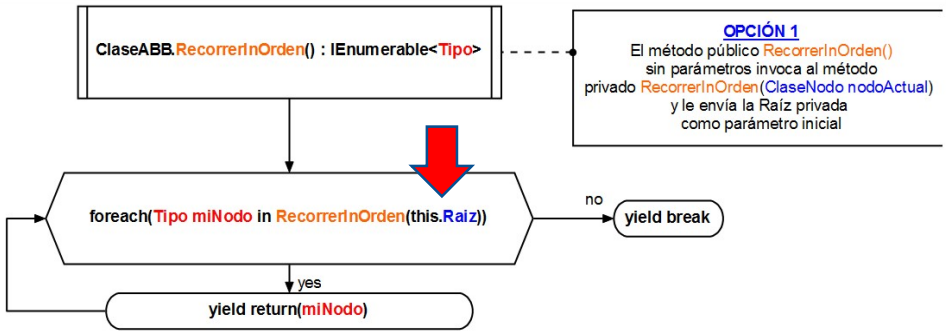
```
public IEnumerable<Tipo> RecorrerPreOrden()
public IEnumerable<Tipo> RecorrerInOrden()
public IEnumerable<Tipo> RecorrerPostOrden()
```

Y cada uno invoca a su respectivo iterador privado:

```
private IEnumerable<Tipo> RecorrerPreOrden(ClaseNodo<Tipo> nodoActual)
private IEnumerable<Tipo> RecorrerInOrden(ClaseNodo<Tipo> nodoActual)
private IEnumerable<Tipo> RecorrerPostOrden(ClaseNodo<Tipo> nodoActual)
```

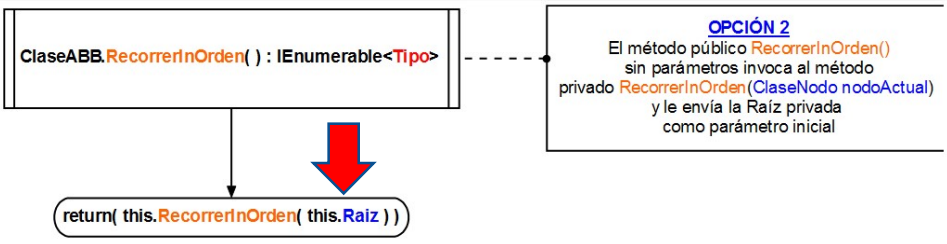
71

Opción 1



72

Opción 2



73

Implementación

```
private static btn_InOrden(object sender, EventArgs e)
{
   foreach(Empleado e in miABB.RecorrerInOrden())
   {
        . . .
        . . .
   }
}
```

74

Diseño de la clase "roja"

- *Requisitos: Debe contener al menos un dato de los siguientes tipos:*
  - *Int*
  - *Double*
  - *String*
  - *Char*
  - *DateTime*
  - *Bool*
  - *String con la ruta del archivo que contiene una fotografía del objeto*
  - ***Sobrescritura del método ToString()***

Se recomienda consultar las filminas
"El lenguaje C# y diseño de formas"
Para usar el PictureBox

75

Diseño de la clase "roja" (*cont.*)

IMPORTANTE

Considere un dato dentro de su clase "roja" que sea único e irrepetible que sirva para identificar a un objeto y que sea el criterio de comparación entre objetos

(Una clave para identificar y comparar objetos "rojos")

76

Diseño de la clase "roja" (*cont.*)

IMPORTANTE

La sobrescritura del método ToString de la clase "roja" **solamente** debe mostrar el campo único e irrepetible que sirva para identificar a un objeto (mismo campo que utiliza el método Equals) y que identificará al nodo en el dibujo del ABB

(Una clave para identificar y comparar objetos "rojos")

77

Tarea 3.03

- *Resolver la Tarea 3.03.- Diseño de la clase "roja" del ABB en MS Teams*
- *Subir el archivo con el diagrama de la clase "roja"*
- *Incluir las interfaces correspondientes*

78

Tarea 3.04.- DF ABB (Agregar)

- Subir a MS Teams el archivo con el diagrama de flujo de:
  - *Método para agregar un objeto "rojo" al ABB*

79

Diseño de la forma de la aplicación visual

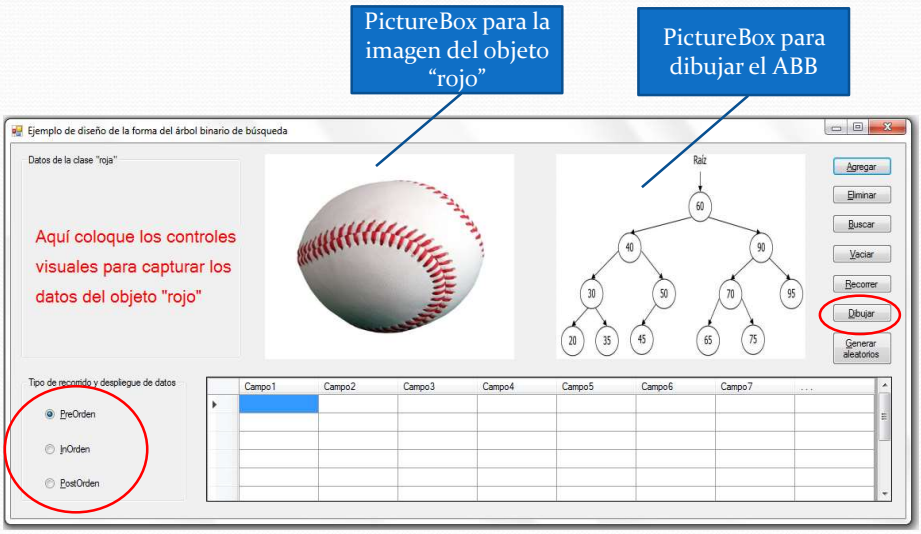
- *Requisitos: Debe contener al menos uno de estos controles visuales:*
  - *Textbox*
  - *Button*
  - *ComboBox*
  - *DateTimePicker*
  - *CheckBox*
  - *PictureBox*
  - *DataGridView*
  - *RadioButton*

Elija el control adecuado para cada dato capturado

Se recomienda consultar las filminas "Uso de los controles visuales"

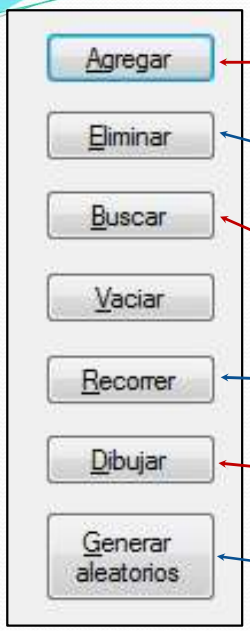
80

Sugerencia de diseño de forma



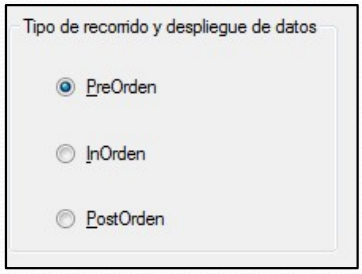
81

Menú de opciones



- **Agregar**: Agrega el objeto "rojo" al ABB con los datos capturados manualmente y actualiza el dataGridView y el dibujo del ABB
- **Eliminar**: Elimina el objeto "rojo" seleccionado del dataGridView y actualiza sus datos junto con el dibujo del ABB
- **Buscar**: Despliega un MessageBox con los datos del objeto "rojo" seleccionado del dataGridView
- **Recorrer**: Ejecuta el recorrido seleccionado en los radioButtons del "Tipo de recorrido" y despliega el strProgramaDot en un MessageBox
- **Dibujar**: Dibuja el ABB y lo muestra en el pictureBox correspondiente
- **Generar aleatorios**: Genera 10 objetos "rojos" con datos aleatorios, muestra sus datos en el dataGridView y dibuja el ABB

Tipos de recorridos

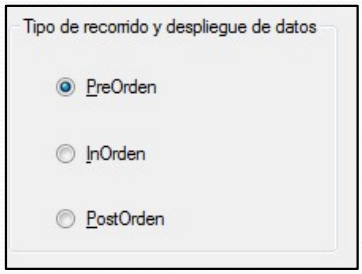


Determina la forma de recorrido del ABB para mostrar sus datos en el dataGridView

Al hacer clic en alguno de los radioButtons se debe actualizar automáticamente el dataGridView

83

Actualización de los datos del ABB



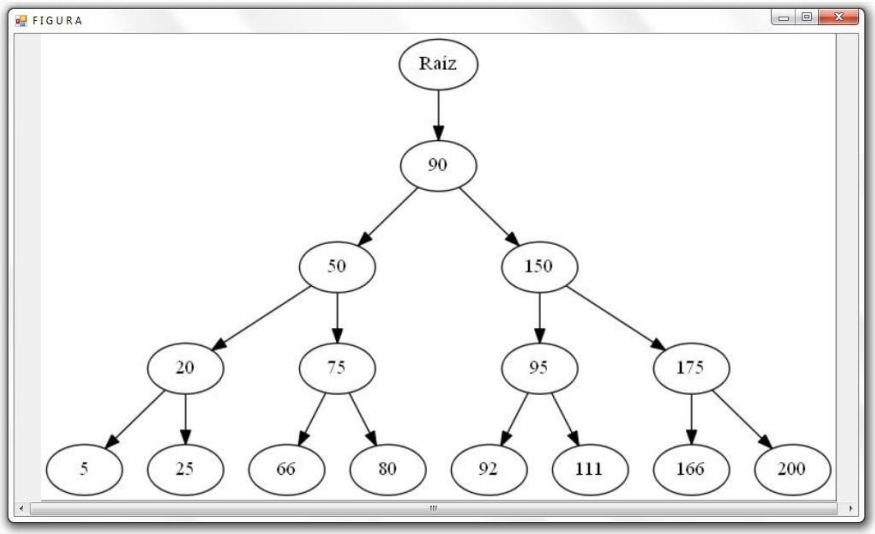
Se deben actualizar los datos mostrados en el dataGridView y en el dibujo del ABB cada vez que se ejecute:

- *Agregar nodo*
- *Eliminar nodo*
- *Vaciar*
- *Generar aleatorios*

Los datos se mostrarán en el dataGridView de acuerdo al radioButton seleccionado

84

Dibujar el ABB

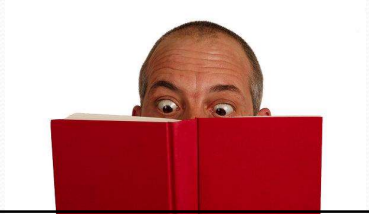


85

Lectura

¿Cómo dibujar una estructura de datos utilizando Graphviz y su lenguaje dot?

https://nlaredo.tecnm.mx/takeyas/Apuntes/Estructura%20de%20Datos/Apuntes/Graphviz.pdf



86

Demostración

Graphviz y su lenguaje dot

87

Tarea 3.05.- GraphViz

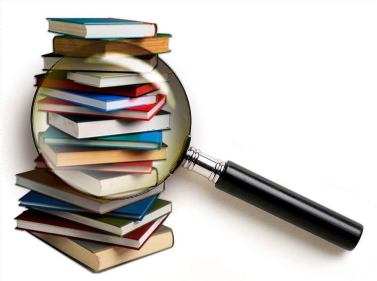
Leer ¿Cómo dibujar una estructura de datos utilizando Graphviz y su lenguaje dot?

- *Contestar el cuestionario en MS Teams*
- *Se aceptará esta tarea si se obtiene calificación aprobatoria*

https://nlaredo.tecnm.mx/takeyas/Apuntes/Estructura%20de%20Datos/Apuntes/Graphviz.pdf

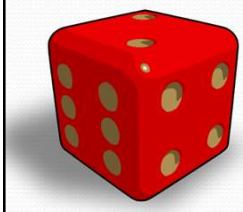
Investigación

- *Agregue un botón a su aplicación para crear `10` nodos con datos generados de manera aleatoria*
- *Muestre los datos generados en el dataGridView*



89

LECTURA



Para generar datos aleatorios, se recomienda la lectura de los apuntes

¿Cómo generar datos aleatorios en C#?

Incluye generar:

- *Nombres aleatorios*
- *Sexo*
- *Fecha*
- *Datos booleanos*
- *etc.*

https://nlaredo.tecnm.mx/takeyas/Apuntes/Fundamentos%20de%20Programacion/Apuntes/07.-Aleatorios.pdf

90

Sugerencia

- *Sugerencia para seleccionar las carpetas de:*
  - *Archivo BAT*
  - *Archivo con el programa DOT*
  - *Archivo JPG con la figura del ABB*

91

Otra sugerencia

- *Diseñar un método que verifique la existencia de:*
  - *Carpeta de trabajo para dibujar el ABB*
    - *Crear la carpeta si no existe*
  - *Batch* `Dibujar.Bat`
    - *Crear el archive* `Dibujar.Bat` *si no existe*
- *Invocar este método desde el método* `Form_Load()`

92

Código de Dibujar.bat

- *El batch `Dibujar.bat` debe ubicarse dentro de la carpeta donde dibujará la figura*
  - *P. ejem. C:\Datos*

```
@echo off
c:
cd \Datos
del Figura.jpg
\"Program Files (x86)"\Graphviz2.31\bin\DOT -Tjpg -O Figura
```

Ejemplo de ruta del programa DOT

Asegúrese escribir la ruta completa donde se ubica el programa DOT de GraphViz
(Puede variar dependiendo de la versión de GraphViz y de la carpeta de instalación)

93

Codificación del método

```
private void VerificarCarpeta()
{
   string strRuta = "\\22100123";

   // Si no existe la carpeta de la ruta
   if(!System.IO.Directory.Exists(strRuta))
   {
       // Crear la carpeta
       System.IO.Directory.CreateDirectory(strRuta);
       MessageBox.Show("Se ha creado la carpeta " + strRuta);
   }

   // Si no existe Dibujar.Bat
   if(!System.IO.File.Exists(strRuta+"\\Dibujar.bat"))
   {
      string strCodigoBat = "c:\ncd " + strRuta + "\ndel Figura.jpg\ndot -Tjpg -O Figura";

      // Se crea el archivo Dibujar.Bat
      System.IO.StreamWriter miArchivoBat = new System.IO.StreamWriter(strRuta + "\\Dibujar.Bat");
      miArchivoBat.Write(strCodigoBat);
      miArchivoBat.Close();
      MessageBox.Show("Se ha creado el archivo Dibujar.Bat\n\n"+strCodigoBat);
   }
}
```

Se recomienda variable global

94

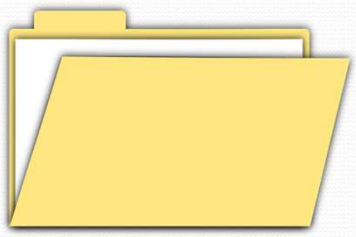
```
// Declaración de la variable global para la ruta
// de la carpeta de trabajo para dibujar el ABB
static string strRuta = "\\22100123";
private void Form1_Load(object sender, E
{
    VerificarCarpeta();
}

private void VerificarCarpeta()
{
    // Si no existe la carpeta de la ruta
    if (!System.IO.Directory.Exists(strRuta))
    {
        // Crear la carpeta
        System.IO.Directory.CreateDirectory(strRuta);
        MessageBox.Show("Se ha creado la carpeta " + strRuta);
    }

    // Si no existe Dibujar.Bat
    if (!System.IO.File.Exists(strRuta + "\\Dibujar.bat"))
    {
        string strCodigoBat = "c:\ncd " + strRuta + "\ndel Figura.jpg
          \ndot -Tjpg -O Figura";

        // Se crea el archivo Dibujar.Bat
        System.IO.StreamWriter miArchivoBat = new
          System.IO.StreamWriter(strRuta + "\\Dibujar.Bat");
        miArchivoBat.Write(strCodigoBat);
        miArchivoBat.Close();
        MessageBox.Show("Se ha creado el archivo Dibujar.Bat\n\n" +
          strCodigoBat);
    }
}
```

Variable global

95

Recomendación

- *Declarar una variable global de tipo string que almacene la ruta de la carpeta que guarde los archivos de la figura del ABB*



96

Código del botón para dibujar el ABB

```
private void btnDibujarABB_Click(object sender, EventArgs e)
{
   if (!ArbolBinarioBusqueda.EstaVacio())  // Si no está vacío ..
        DibujarFigura();  // Invoca el método para dibujar la figura
   else
   MessageBox.Show("Árbol Binario de Búsqueda vacío“);
}
```

97

Código del método DibujarFigura()

```
private void DibujarFigura()
{
// Se invoca el método para crear el archivo Figura que contiene el
programa en lenguaje DOT que crea el archivo Figura.jpg con la imagen del
ABB
    string strProgramaDot=CrearArchivoDot();

// Ejecución del proceso por lotes DIBUJAR.bat para crear la imagen
Figura.jpg
    InvocaDibujar();

    MessageBox.Show(strProgramaDot);
// Ejecución del método para crear una nueva forma y mostrar en ella la
Figura.jpg con la imagen del ABB
    MostrarImagenJPG();
}
```

98

Código del método independiente CrearArchivoDot()

```
private string CrearArchivoDot()
{
   // Recorrido del ABB para crear la línea con los comandos para el
archivo en lenguaje DOT
   string strProgramaDot = "";
   ArbolBinarioBusqueda.CrearArchivoDot(ref strProgramaDot);

  // Creación del archivo \Figura que contiene el programa en lenguaje
DOT)
  System.IO.StreamWriter miArchivoDot = new
System.IO.StreamWriter("c:\\Datos\\Figura");
  miArchivoDot.WriteLine(strProgramaDot);
  miArchivoDot.Close();

  return (strProgramaDot);
}
```

99

Código del método CrearArchivoDot() de la ClaseABB

```
class ClaseABB
{
  private void RecorrerNodos(ClaseNodoArbolBinarioBusqueda NodoActual, ref string strProgramaDot)  {
      if (NodoActual != null) {
          if (NodoActual.HijoIzq != null)
              strProgramaDot = strProgramaDot + "\n" + NodoActual.Dato.ToString() + "->" +
NodoActual.HijoIzq.Dato.ToString() + ";";

          if (NodoActual.HijoDer != null)
              strProgramaDot = strProgramaDot + "\n" + NodoActual.Dato.ToString() + "->" +
NodoActual.HijoDer.Dato.ToString() + ";";

          // Llamada recursiva para recorrer el subárbol izquierdo
          RecorrerNodos(NodoActual.HijoIzq, ref strProgramaDot);

          // Llamada recursiva para recorrer el subárbol derecho
          RecorrerNodos(NodoActual.HijoDer, ref strProgramaDot);
      }
}

public void CrearArchivoDot(ref string strProgramaDot) {
   if (!EstaVacio())  // Si no está vacío ...
   {
     strProgramaDot = strProgramaDot + "digraph Figura {";
     strProgramaDot = strProgramaDot + "\nRaíz->" + Raiz.Dato.ToString() + ";";
     RecorrerNodos(Raiz, ref strProgramaDot);
     strProgramaDot = strProgramaDot + "\n}";
   }
 }
}
```

Código del método InvocaDibujar()

```
private void InvocaDibujar()
{
   // El archivo por lotes DIBUJAR.bat contiene el siguiente código:
   //    @echo off
   //    del \Figura.jpg
   //    \"Program Files (x86)"\GraphViz2.31\bin\DOT -Tjpg -O \Figura

   // Ejecuta DIBUJAR.bat ubicado en la carpeta c:\Datos
   System.Diagnostics.Process.Start("c:\\Datos\\DIBUJAR.bat");
}
```

101

Código del método MostrarImagenJPG()

```
private void MostrarImagenJPG()    {

  picABB.Width = 990;  // Define el ancho del pictureBox
  picABB.Height = 545;  // Define la altura del pictureBox
  picABB.SizeMode = PictureBoxSizeMode.Zoom;

  // Abre el archivo Figura.jpg en modo de sólo lectura
  System.IO.FileStream miArchivoJPG; // Declaración del archivo Figura.jpg

  try { // Intenta abrir el archivo
    miArchivoJPG = new System.IO.FileStream("c:\\Datos\\Figura.jpg", System.IO.FileMode.Open,
System.IO.FileAccess.Read);
  }
  catch (Exception x) { // En caso de error ...
     MessageBox.Show("No se pudo abrir el archivo FIGURA.jpg");
     return;
  }

  // Carga la imagen del ArbolBinarioBusqueda.jpg en el pictureBox

  try {  // Intenta cargar la imagen en el pictureBox
     picABB.Image = System.Drawing.Bitmap.FromStream(miArchivoJPG);
  }
  catch (Exception x) { // En caso de error ...
     MessageBox.Show("Error al cargar la imagen del archivo FIGURA.jpg");
     miArchivoJPG.Close();  // Cierra el archivo
     return;
  }

  miArchivoJPG.Close();
  picABB.Refresh(); // Refresca la imagen
}
```

Tarea 3.06.- Diseño de la forma del ABB

- Diseñar la forma en C#:
- *Capturar los datos usando los controles visuales adecuados*
- *Agregar un dataGridView de solo lectura para visualizar los datos de los objetos "rojos"*
- *Implementar el método para agregar objetos "rojos" al ABB y visualizarlos en el dataGridView*
- *Implementar el botón "Dibujar el ABB"*
- *Subir a MS Teams un archivo comprimido con la aplicación completa (P. ejem. LopezTakeyasBruno.ZIP)*

103

Tarea 3.07.- DF ABB (Buscar)

- Subir a MS Teams el archivo con el diagrama de flujo de:
  - *Método para buscar un objeto "rojo" en el ABB (debe devolver el objeto encontrado)*

104

Eliminación de un nodo en un ABB

- *Al borrar un nodo se debe mantener el principio de que los hijos izquierdos son menores y los hijos derechos mayores que el padre*
- *En algunos casos se debe hacer una sustitución de un nodo para mantener el principio*
- *Se puede interrumpir la búsqueda por anticipado ya que el ABB almacena datos ordenados*

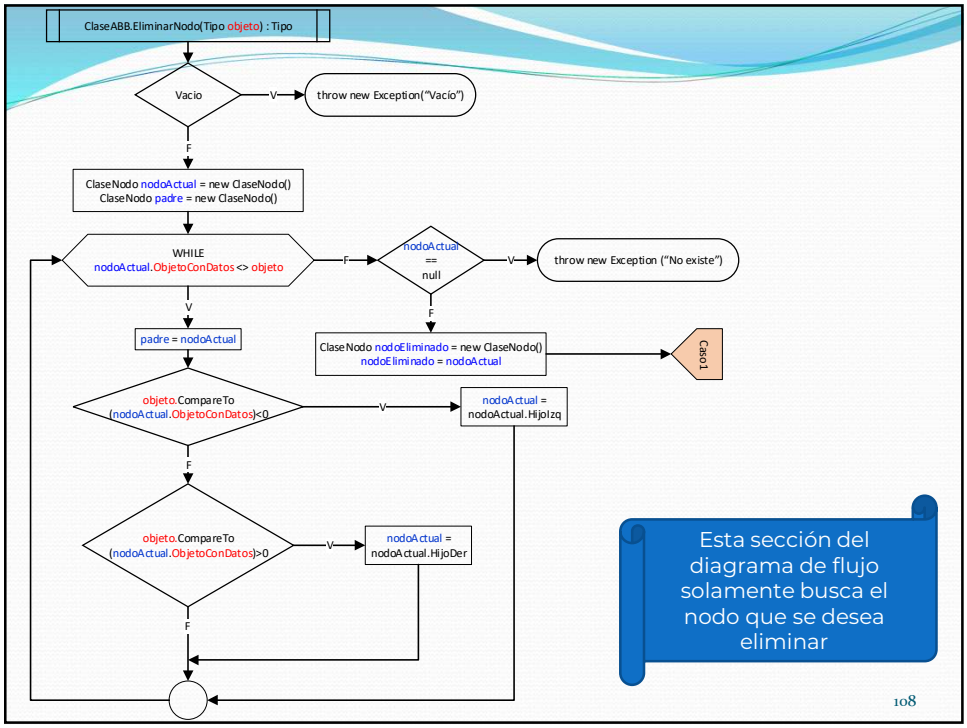
Situaciones críticas de las bajas en un ABB

- *Baja en un ABB vacío*
- *Baja de un nodo sin hijos (hoja)*
- *Baja de un nodo con hijo izquierdo solamente*
- *Baja de un nodo con hijo derecho solamente*
- *Baja de un nodo con ambos hijos*
- *Baja del único nodo*
- *Verificar la existencia del dato*

Pseudocódigo para buscar el nodo a eliminar

```
/* Búsqueda del nodo que se desea eliminar */

1.- nodoActual = Raiz
2.- padre = nulo
3.- MIENTRAS nodoActual.ObjetoConDatos ≠ nodo REPETIR
    3.1. padre = nodoActual
    3.2. SI nodo < nodoActual.ObjetoConDatos ENTONCES
           3.2.1. nodoActual = nodoActual.HijoIzq
    SINO
      3.2.2.1. SI nodo > nodoActual.ObjetoConDatos ENTONCES
          3.2.2.1.1. nodoActual = nodoActual.HijoDer
      3.2.2.2. {FIN DE LA CONDICIONAL DEL PASO 3.1.2.1.}
    3.3. {FIN DE LA CONDICIONAL DEL PASO 3.2.}

    3.4. SI nodoActual == nulo ENTONCES
      3.4.1. DisparaExcepción("No existe")
    3.5. {FIN DE LA CONDICIONAL DEL PASO 3.4}
4.- {FIN DEL CICLO DEL PASO 3}
5.- nodoEliminado = nodoActual
6.- Ir al Caso 1
```

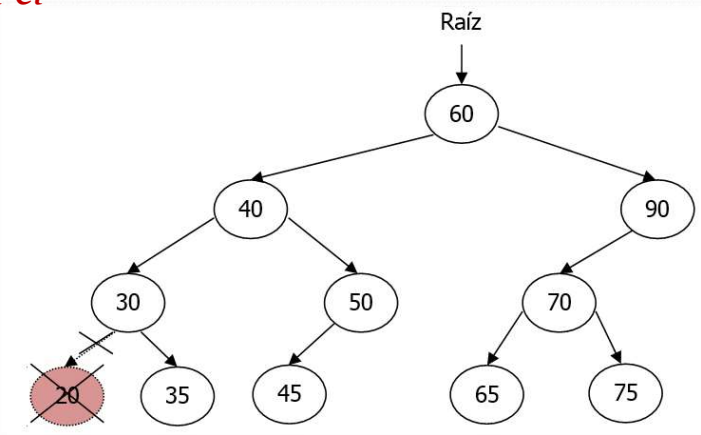
107



108

Baja de un nodo sin hijos

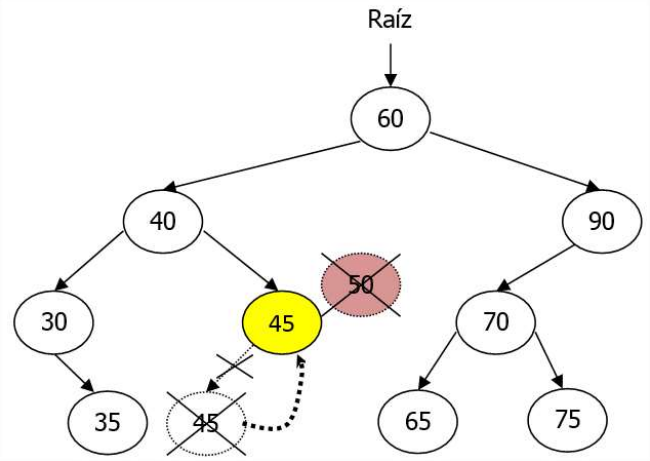
Se elimina el nodo y el apuntador de su padre hacia él



109

Baja de un nodo con hijo izquierdo solamente

El hijo izquierdo reemplaza al nodo eliminado

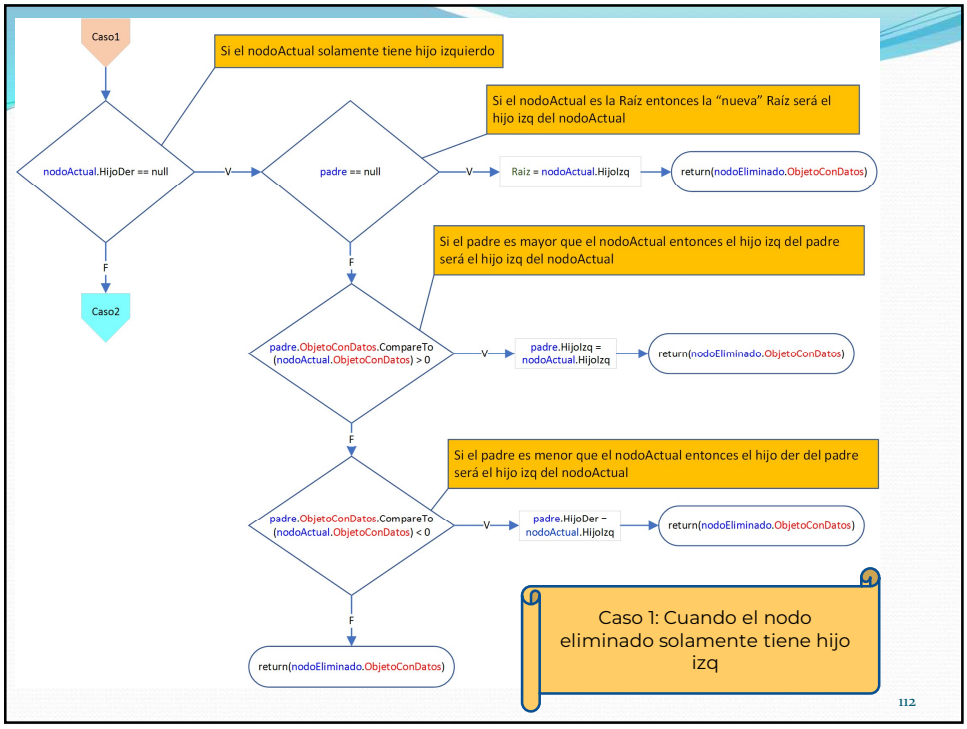


110

Pseudocódigo para eliminar nodo solamente con hijo izquierdo

```
/* Caso: Si el nodoActual no tiene hijo derecho, entonces su hijo
izquierdo se convierte en el nodo apuntado por su padre */

1.- SI nodoActual.HijoDer == nulo ENTONCES

  1.1. SI padre == nulo ENTONCES
    1.1.1. Raiz = nodoActual.HijoIzq
  SINO

    1.1.2.1. SI padre.ObjetoConDatos > nodoActual.ObjetoConDatos ENTONCES
      1.1.2.1.1. padre.HijoIzq = nodoActual.HijoIzq
    SINO
      1.1.2.1.2.1. SI padre.ObjetoConDatos < nodoActual.ObjetoConDatos
ENTONCES
        1.1.2.1.2.1.1. padre.HijoDer = nodoActual.HijoIzq
      1.1.2.1.2.2. {FIN DE LA CONDICIONAL DEL PASO 1.1.2.1.2.1.}
    1.1.2.2. {FIN DE LA CONDICIONAL DEL PASO 1.1.2.1.}

  1.2. {FIN DE LA CONDICIONAL DEL PASO 1.1.}

2.- {FIN DE LA CONDICIONAL DEL PASO 1}
```

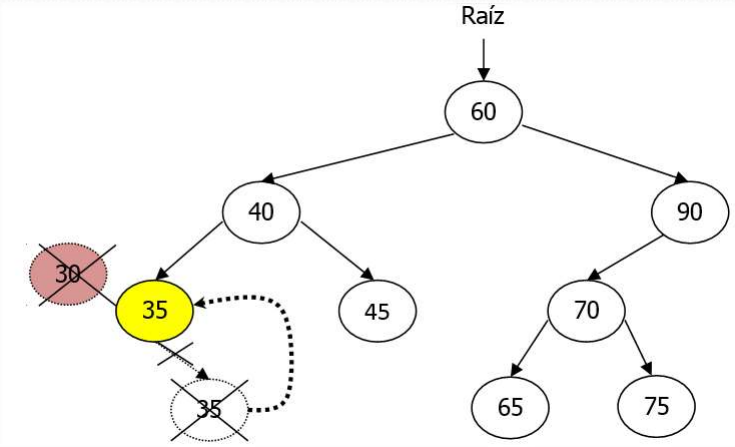
111



112

Baja de un nodo con hijo derecho solamente

El hijo derecho reemplaza al nodo eliminado

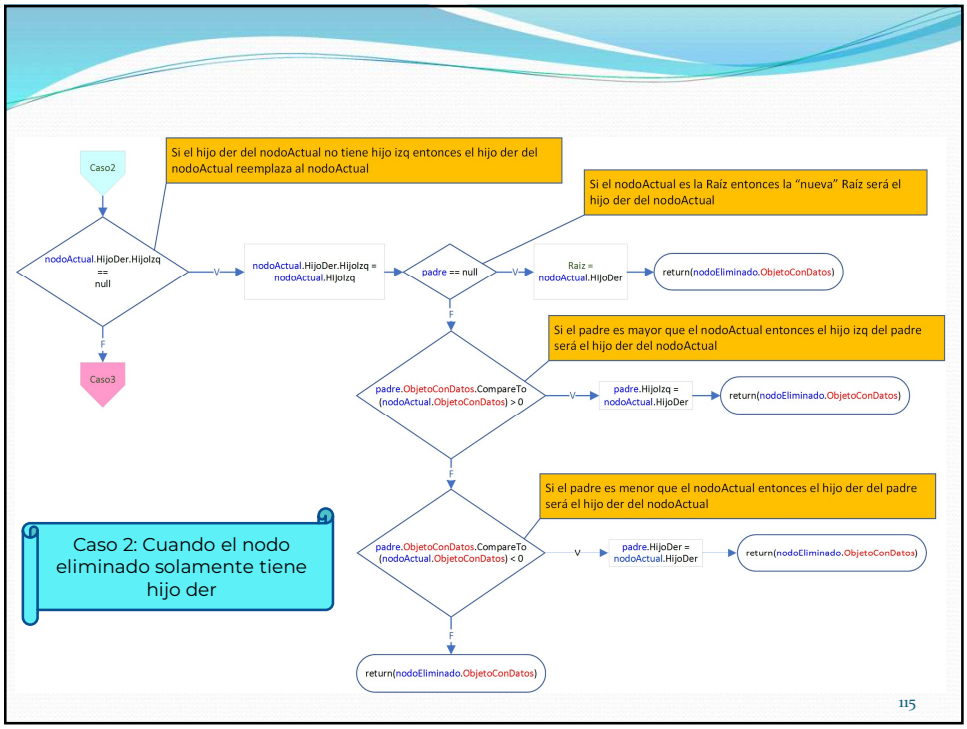


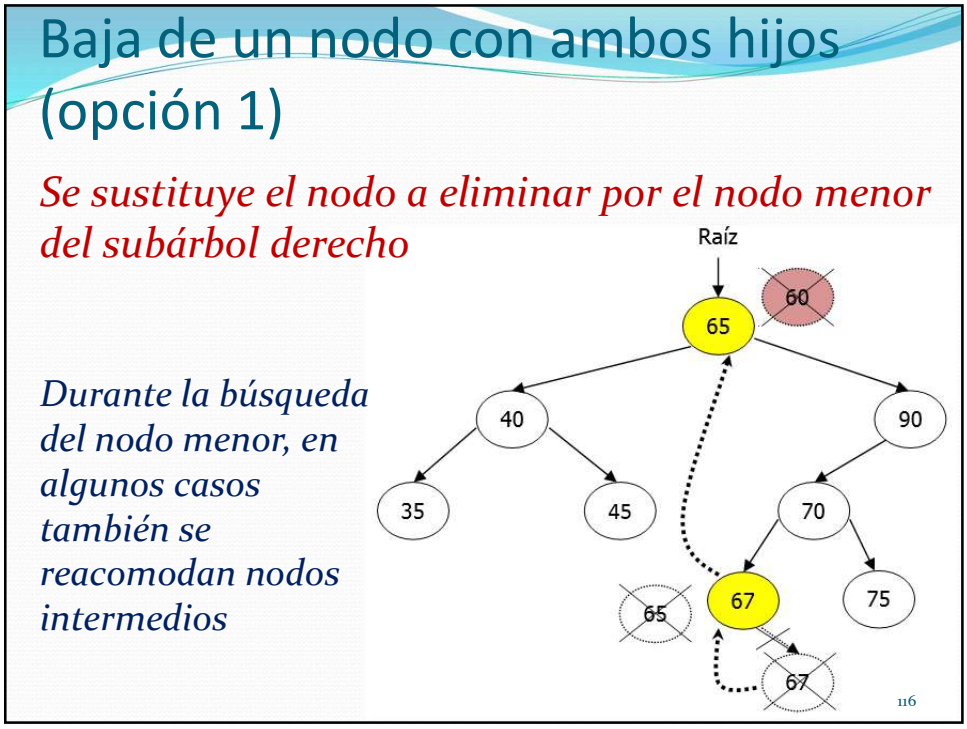
113

Pseudocódigo para eliminar nodo solamente con hijo derecho

```
/* Caso: Si el nodoActual no tiene hijo izquierdo, entonces su hijo derecho lo
reemplaza */

1.- SI nodoActual.HijoDer.HijoIzq == nulo ENTONCES
   1.1. nodoActual.HijoDer.HijoIzq = nodoActual.HijoIzq

   1.2. SI padre == nulo ENTONCES
     1.2.1. Raiz = nodoActual.HijoDer
   SINO

     1.2.2.1. SI padre.ObjetoConDatos > nodoActual.ObjetoConDatos ENTONCES
       1.2.2.1.1. padre.HijoIzq = nodoActual.HijoDer
     SINO

       1.2.2.2.1.1. SI padre.ObjetoConDatos < nodoActual.ObjetoConDatos ENTONCES
         1.2.2.2.1.1.1. padre.HijoDer = nodoActual.HijoDer
        1.2.2.2.1.2. {FIN DE LA CONDICIONAL DEL PASO 1.2.2.2.1.1.}

      1.2.2.2. {FIN DE LA CONDICIONAL DEL PASO 1.2.2.1.}
   1.3. {FIN DE LA CONDICIONAL DEL PASO 1.2.}

2.- {FIN DE LA CONDICIONAL DEL PASO 1}
```

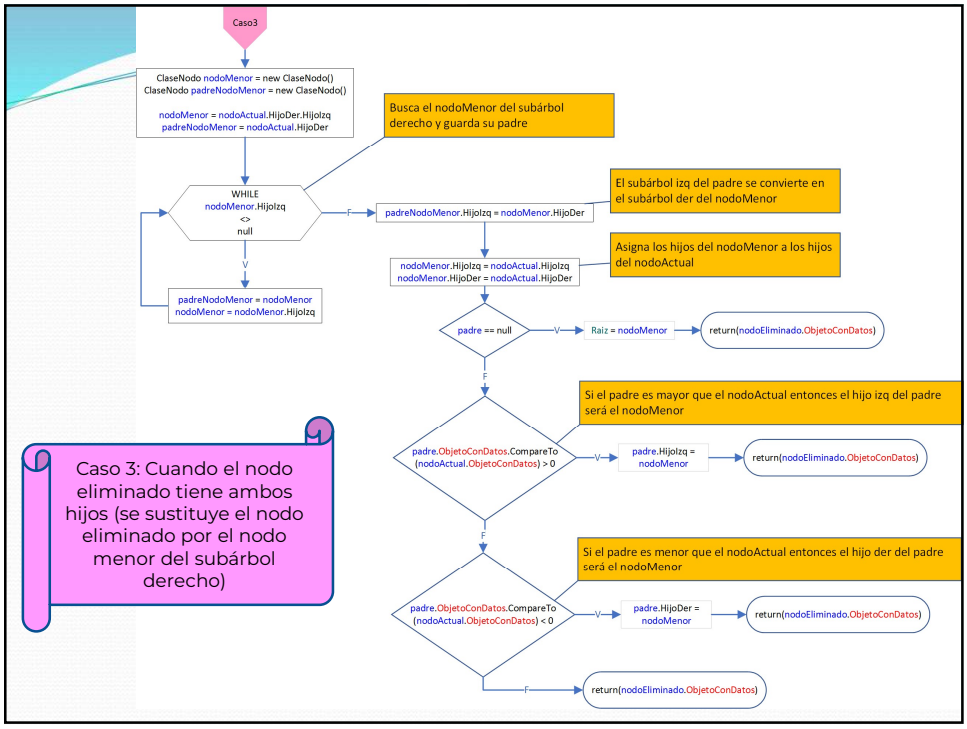
114


Caso2
Si el hijo der del nodoActual no tiene hijo izq entonces el hijo der del nodoActual reemplaza al nodoActual
Si el nodoActual es la Raíz entonces la "nueva" Raíz será el hijo der del nodoActual
nodoActual.HijoDer.HijoIzq == null
V
nodoActual.HijoDer.HijoIzq = nodoActual.HijoIzq
padre == null
V
Raiz = nodoActual.HijoDer
return(nodoEliminado.ObjetoConDatos)
F
Caso3
Si el padre es mayor que el nodoActual entonces el hijo izq del padre será el hijo der del nodoActual
padre.ObjetoConDatos.CompareTo (nodoActual.ObjetoConDatos) > 0
V
padre.HijoIzq = nodoActual.HijoDer
return(nodoEliminado.ObjetoConDatos)
F
Si el padre es menor que el nodoActual entonces el hijo der del padre será el hijo der del nodoActual
Caso 2: Cuando el nodo eliminado solamente tiene hijo der
padre.ObjetoConDatos.CompareTo (nodoActual.ObjetoConDatos) < 0
V
padre.HijoDer = nodoActual.HijoDer
return(nodoEliminado.ObjetoConDatos)
F
return(nodoEliminado.ObjetoConDatos)
115


Baja de un nodo con ambos hijos (opción 1)
Se sustituye el nodo a eliminar por el nodo menor del subárbol derecho
Durante la búsqueda del nodo menor, en algunos casos también se reacomodan nodos intermedios
Raíz
60
65
40
90
35
45
70
65
67
75
67
116

Pseudocódigo para eliminar nodo con ambos hijos (opción 1)

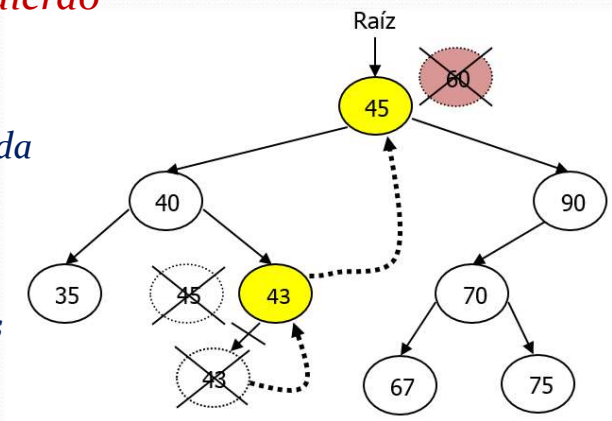
```
/* Caso: Si el hijo derecho del nodoActual tiene hijo izquierdo, entonces se reemplaza el nodoActual por el nodo
menor del subárbol derecho */

// Inicia la búsqueda del nodo ubicado más a la izquierda del subárbol derecho

1.- nodoMenor = nodoActual.HijoDer.HijoIzq
2.- padreDelNodoMenor = nodoActual.HijoDer
3.- MIENTRAS nodoMenor.HijoIzq ≠ nulo REPETIR
    3.1. padreDelNodoMenor = nodoMenor
    3.2. nodoMenor = nodoMenor.HijoIzq
4.- {FIN DEL CICLO DEL PASO 3}

// El subárbol izq. de su padre se convierte en el subárbol der. del nodoMenor

5.- padreDelNodoMenor.HijoIzq = nodoMenor.HijoDer
6.- nodoMenor.HijoIzq = nodoActual.HijoIzq
7.- nodoMenor.HijoDer = nodoActual.HijoDer

8.- SI padre == nulo ENTONCES
    8.1. Raiz = nodoMenor
SINO

    8.2.1. SI padre.ObjetoConDatos > nodoActual.ObjetoConDatos ENTONCES
      8.2.1.1. padre.HijoIzq = nodoMenor
    SINO

      8.2.1.2.1. SI padre.ObjetoConDatos < nodoActual.ObjetoConDatos ENTONCES
        8.2.1.2.1.1. padre.HijoDer = NodoMenor
      8.2.1.2.1. {FIN DE LA CONDICIONAL DEL PASO 8.2.1.2.1.}

    8.2.2. {FIN DE LA CONDICIONAL DEL PASO 8.2.1.}

9.- {FIN DE LA CONDICIONAL DEL PASO 8}
```



Baja de un nodo con ambos hijos (opción 2)

Se sustituye el nodo a eliminar por el nodo mayor del subárbol izquierdo

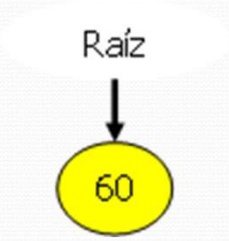
Durante la búsqueda del nodo menor, en algunos casos también se reacomodan nodos intermedios



119

Baja del único nodo del ABB

Se elimina el nodo actual y la Raíz apunta a nulo




```
Raíz ⟶ nulo
```

120

Eliminación de objetos

- *La forma natural de borrar un objeto es asignarle el valor null*

```
// Creación del objeto "azul" del nodoActual
ClaseNodo<Tipo> nodoActual = new ClaseNodo<Tipo>();
   .
   .
nodoActual = null; // Eliminación del nodoActual
```

- *Sin embargo, no todos los datos aceptan el valor null, entonces... ¿cómo se eliminarían?*

121

Destructor de la clase "azul"

- *La* `ClaseNodo<Tipo>` *es parametrizada y está preparada para recibir un objeto "rojo" de cualquier tipo.*
- *El destructor de la clase "azul" elimina el objeto con datos "rojo" que contiene.*
- *Utiliza* `default(Tipo)` *para eliminar el objeto "rojo" porque desconoce si el ObjetoConDatos acepta el valor null.*

```
~ClaseNodo() // Destructor de la clase "azul"
{
     // Elimina el ObjetoConDatos "rojo"
     ObjetoConDatos = default(Tipo);
}
```

Tarea 3.08.- DF ABB (Eliminar)

- Hacer el diagrama de flujo de:
  - Método `Eliminar(Tipo objeto) : Tipo`
    - ***Devolver el objeto "rojo" eliminado***

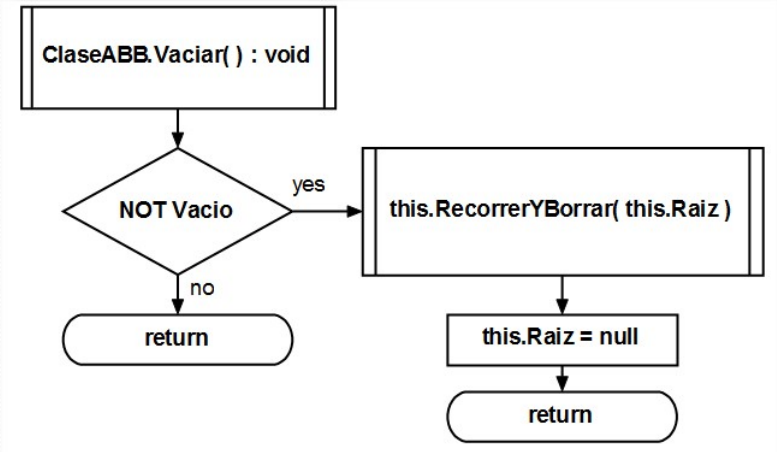
123

Vaciar el ABB

- *Recorrer el árbol y borrar cada uno de sus nodos*
- *Se recomienda apoyarse en una variante del recorrido PostOrden (recorrer primero el subárbol izquierdo, luego el subárbol derecho y por ultimo visitar el nodo a eliminar)*
- *Encontrar las hojas y enviarlas al método Eliminar()*

124

Diagrama de flujo del método para vaciar



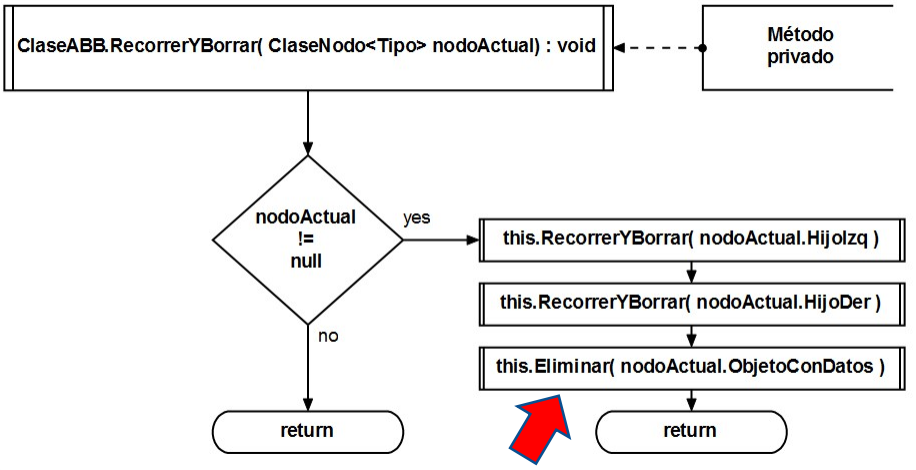
125

Método recursivo RecorrerYBorrar()

- *Utiliza una variante del recorrido PostOrden para encontrar cada hoja del árbol y enviarlas como parámetro al método Eliminar()*
- *Una hoja es el nodo más fácil de eliminar*

126

Diagrama de flujo del método recursivo RecorrerYBorrar()



127

Tarea 3.09.- DF ABB (Vaciar y Destructor)

- Subir a MS Teams el archivo con diagramas de flujo de:
  - *Método para vaciar el ABB*
  - *Destructor del ABB*

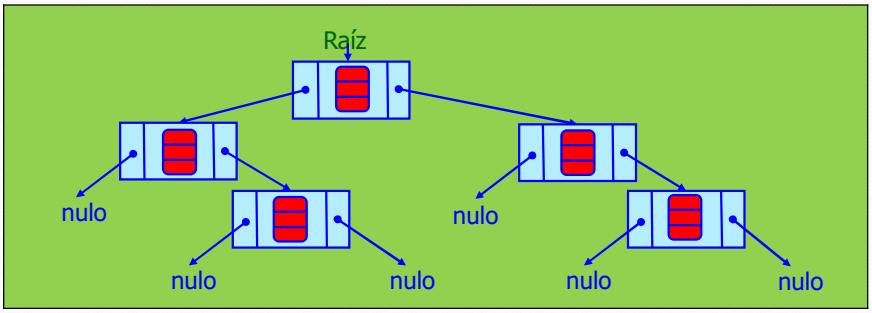
128

Tarea 3.10.- Aplicación completa del ABB

- *Agregar un botón "Vaciar"*
- *Solicitar al usuario que confirme las operaciones (insertar, eliminar, vaciar, etc.). Preguntarle si está seguro que desea realizar la operación solicitada*
- *Mostrar los mensajes adecuados*
- *Subir a MS Teams un archivo comprimido con la aplicación completa (P. ejem. LopezTakeyasBruno.ZIP)*

129

Nivel de abstracción



- *No se debe perder de vista que los objetos "rojos" son privados, por lo tanto sus componentes son inacesibles para el objeto "verde"*

130

Nivel de abstracción (*cont.*)

- *El objeto "verde" recibe un objeto "rojo" y lo compara para almacenarlo y ordenarlo*

  ¡¡¡ SIN SABER LO QUE TIENE DENTRO !!!

- *¿Cómo es posible que el objeto "verde" compare y almacene objetos "rojos" sin tener acceso a sus componentes?*

131

"Pensar en objetos …"

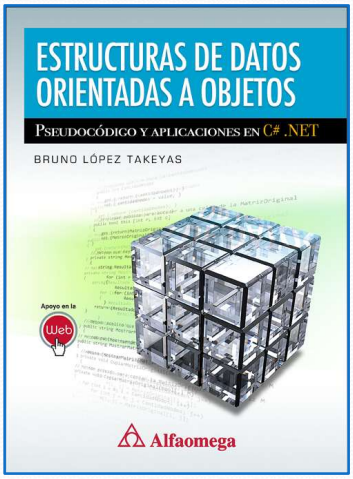
- *El objeto "verde" **NO** compara los objetos "rojos" (ellos mismos se comparan entre sí).*
- *El objeto "verde" **NO** requiere acceso a los componentes de los objetos "rojos" para manipularlos.*
- *Recibe cualquier tipo de objetos "rojos" …*

  ¡¡¡ SIN MODIFICAR NI UNA LÍNEA DE SU CÓDIGO!!!

- ***¿Cómo lo logra? …***
  - *Clases parametrizadas*
  - *Uso de interfaces*
  - *Composición*
  - *Comportamiento polimórfico*
  - *Restricción de tipos*

132

Lectura



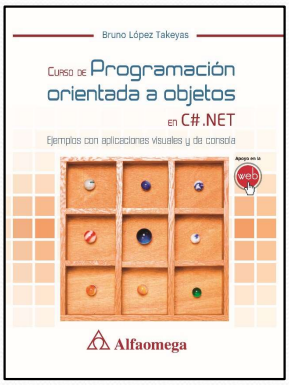
Para reforzar este tema se recomienda la lectura de:

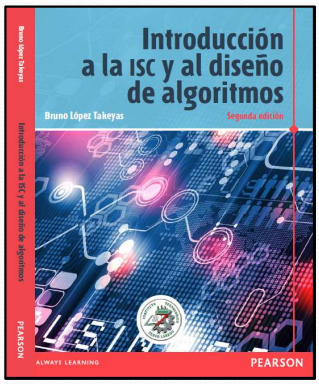
Capítulo 8.- Árboles binarios

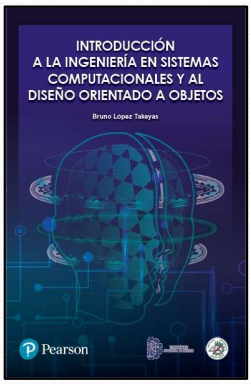
133

Otros libros del autor

https://nlaredo.tecnm.mx/takeyas/Libro







bruno.lt@nlaredo.tecnm.mx

Bruno López Takeyas