Facultad:	Ingeniería
Escuela:	Computación
Asignatura:	Programación con Estructuras de Datos

Tema: Árboles Binarios de Búsqueda (ABB).

Competencia

- Desarrolla sistemas de información informáticos mediante la integración de principios matemáticos, ciencia computacional y prácticas de ingeniería, considerando estándares de calidad y mejores prácticas validadas por la industria del software.

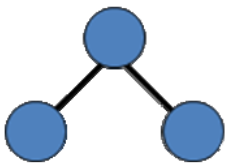
Materiales y Equipo

- Guía Número 6
- Computadora con programa Microsoft Visual C#.

Introducción Teórica

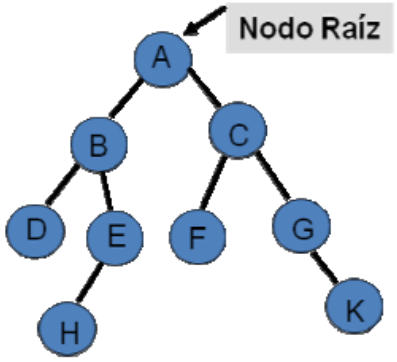
ESTRUCTURA DINÁMICA ÁRBOL BINARIO

En ciencias de la computación, un árbol binario es una estructura de datos en la cual cada nodo siempre tiene cero hijos (0), un hijo (1) o un hijo izquierdo y un hijo derecho (2). No pueden tener más de dos hijos (de ahí el nombre 'Binario'). Si algún hijo tiene como referencia a null, es decir que no almacena ningún dato, entonces este es llamado un nodo externo. En el caso contrario el hijo es llamado un nodo interno

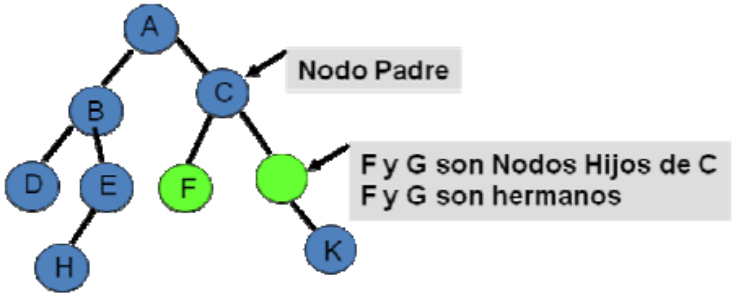


TERMINOLOGÍA:

Nodo: Cada elemento de un árbol
Nodo Raíz: Primer elemento agregado al árbol

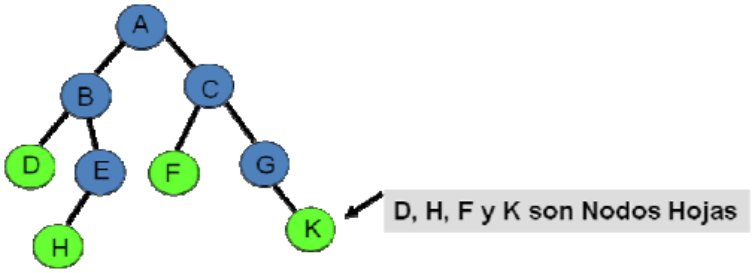


Nodo Padre: Se le llama así al nodo predecesor de un elemento.
Nodo Hijo: Es el nodo sucesor de un elemento.
Nodo Hermano: Nodos que tienen el mismo nodo padre



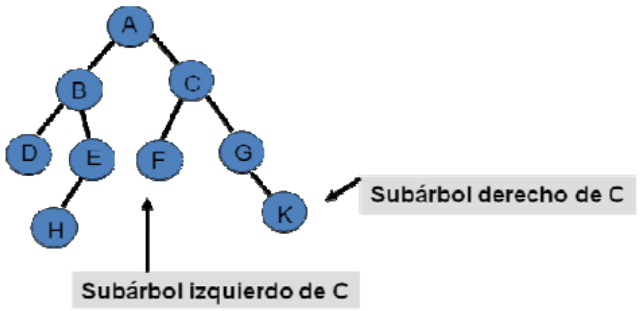
Subárbol:

1. Si el nodo tiene 0 relaciones recibe el nombre de hoja.

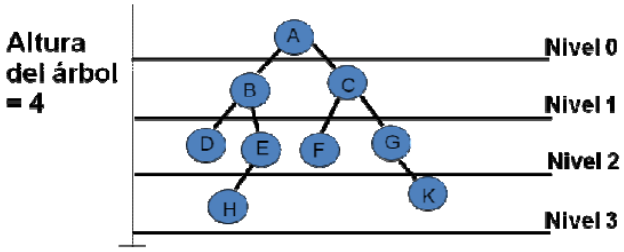


2. Si el nodo raíz tiene 1 relación a la izquierda, el segundo elemento de la relación es el subárbol izquierdo.

3. Si el nodo raíz tiene 1 relación a la derecha, el segundo elemento de la relación es el subárbol derecho.



Altura y Niveles:
La altura corresponde a la cantidad de niveles que existen, los niveles se inician desde 0.



Los árboles binarios poseen una especialización denominada **árboles binarios de búsqueda o ABB**, estos árboles se distinguen por tener un método particular para ingresar datos, respetan las siguientes reglas:

1. El primer elemento que se ingresa al árbol se convierte en la raíz
2. Todos los elementos que se ingresan posteriormente serán comparados con la raíz del árbol y con las raíces de los subárboles consecuentes.
    a. Todo valor mayor a la raíz será enviado al subárbol derecho para otra comparación o para su inserción inmediata.
    b. Todo valor menor a la raíz será enviado al subárbol izquierdo para otra comparación o para su inserción inmediata.

Procedimiento

Ejemplo 1. Implementación de un árbol binario de búsqueda en C#

1. Crear un proyecto de tipo Windows Form Application, se sugiere que le dé el nombre de **"Arbol Binario".**

2. Agregar una **clase** al proyecto, para facilitar el entendimiento nómbrelo como **"Nodo_Arbol".** Esta clase la utilizaremos para definir el elemento "nodo" del árbol binario. (Para agregar una clase al proyecto, dar click en el Explorador de soluciones, agregar y dar agregar clase y nombrarla con la sugerencia dada).
3. Dentro de la clase creada, agregue el siguiente código:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;          //librería para dibujar figuras geométricas
using System.Linq;
using System.Text;
using System.Threading;        //librería para manejo de hilos
using System.Windows.Forms;

namespace Arbol_Binario
{
    class Nodo_Arbol
    {
        public int info;                    //dato a almacenar en el nodo
        public Nodo_Arbol Izquierdo;        //nodo izquierdo del árbol
        public Nodo_Arbol Derecho;          //nodo derecho del árbol
        public Nodo_Arbol Padre;            //nodo raíz del árbol
        public int altura;
        public int nivel;
        public Rectangle nodo;            //para dibujar el nodo del árbol

//Variable que definen el tamaño de los círculos que representan los nodos del árbol
        private const int Radio = 30;
//Variable para el manejo de distancia horizontal
        private const int DistanciaH = 80;
//variable para el manejo de distancia vertical
        private const int DistanciaV = 10;
//variable para manejar posición eje X
        private int CoordenadaX;
//variable para manejar posición eje Y
        private int CoordenadaY;
        Graphics col;
        private Arbol_Binario arbol; //declarando un objeto de tipo árbol

        public Nodo_Arbol()          //constructor por defecto
        {
        }

//constructor por defecto para el objeto de tipo árbol
        public Arbol_Binario Arbol
{
            get
            { return arbol;}

            set
            { arbol = value;}
```

```
 }

//constructor con parámetros
public Nodo_Arbol (int nueva_info, Nodo_Arbol izquierdo, Nodo_Arbol  derecho, Nodo_Arbol padre)
        {
            info = nueva_info;
            Izquierdo = izquierdo;
            Derecho = derecho;
            Padre = padre;
            altura = 0;
        }

       //función para insertar un nodo en el árbol
        public Nodo_Arbol Insertar(int x, Nodo_Arbol t, int Level)
        {
            if (t == null)
            {
                t = new Nodo_Arbol(x, null, null, null);
                t.nivel = Level;
            }

            else if (x < t.info) //si el valor a insertar es menor que la raíz
            {
                Level++;
                t.Izquierdo = Insertar(x, t.Izquierdo, Level);
            }

            else if (x > t.info) //si el valor a insertar es mayor que la raíz
            {
                Level++;
                t.Derecho = Insertar(x, t.Derecho, Level);
            }
            else
            {
                MessageBox.Show("Dato existente en el Arbol", "Error de Ingreso");
            }
            return t;
        }

//Función para eliminar un nodo de un árbol binario
   public void Eliminar(int x, ref Nodo_Arbol t)
   {
        if (t != null)    //si la raíz es distinta de null
          {
            if (x < t.info) //si el valor a eliminar es menor que la raíz
             {
               Eliminar(x, ref t.Izquierdo);
                }
                else
                {
                    if (x > t.info) //si el valor a eliminar es mayor que la raíz
                    {
                        Eliminar(x, ref t.Derecho);
                    }

                    else
                    {
                        Nodo_Arbol NodoEliminar = t;   //se ubica el nodo a eliminar
                        //se verifica si tiene hijo derecho

                        if (NodoEliminar.Derecho == null){
```

```
            t = NodoEliminar.Izquierdo;
        }
        else
        {
            //se verifica si tiene hijo izq
            if (NodoEliminar.Izquierdo == null){
                t = NodoEliminar.Derecho;
            }

            else
            {
                if (Alturas(t.Izquierdo) - Alturas(t.Derecho) > 0)
        //Para verificar que hijo pasa a ser nueva raíz del subárbol
                {
                    Nodo_Arbol AuxiliarNodo = null;
                    Nodo_Arbol Auxiliar = t.Izquierdo;
                    bool bandera = false;

                    while (Auxiliar.Derecho != null)
                    {
                        AuxiliarNodo = Auxiliar;
                        Auxiliar = Auxiliar.Derecho;
                        bandera = true;
                    }
                     // se crea nodo temporal
                         t.info = Auxiliar.info;
                         NodoEliminar = Auxiliar;

                    if (bandera == true)
                    {
                        AuxiliarNodo.Derecho = Auxiliar.Izquierdo;

                    }

                    else
                    {
                        t.Izquierdo = Auxiliar.Izquierdo;
                    }
                }

                else
                {
                  if (Alturas(t.Derecho) - Alturas(t.Izquierdo) > 0)
                    {
                        Nodo_Arbol AuxiliarNodo = null;
                        Nodo_Arbol Auxiliar = t.Derecho;
                        bool bandera = false;

                        while (Auxiliar.Izquierdo != null)
                        {
                            AuxiliarNodo = Auxiliar;
                            Auxiliar = Auxiliar.Izquierdo;
                            bandera = true;
                        }

                        t.info = Auxiliar.info;
                        NodoEliminar = Auxiliar;

                        if (bandera == true)
                        {
                         AuxiliarNodo.Izquierdo = Auxiliar.Derecho;
```

```
                                    }

                                    else
                                    {
                                        t.Derecho = Auxiliar.Derecho;
                                    }
                                }

                            else
                             {
                              if (Alturas(t.Derecho) - Alturas(t.Izquierdo) == 0)
                                    {
                                        Nodo_Arbol AuxiliarNodo = null;
                                        Nodo_Arbol Auxiliar = t.Izquierdo;
                                        bool bandera = false;

                                        while (Auxiliar.Derecho != null)
                                        {
                                            AuxiliarNodo = Auxiliar;
                                            Auxiliar = Auxiliar.Derecho;
                                            bandera = true;
                                        }

                                        t.info = Auxiliar.info;
                                        NodoEliminar = Auxiliar;

                                        if (bandera == true)
                                        {
                                      AuxiliarNodo.Derecho = Auxiliar.Izquierdo;

                                        }

                                        else
                                        {
                                            t.Izquierdo = Auxiliar.Izquierdo;
                                        }
                                    }
                                }
                             }
                          }
                       }
                    }
                }

                else
                {
                 MessageBox.Show("Nodo NO existente el Arbol", "Error de eliminación");
                }
        }  //Final de la función eliminar

   //Función buscar un nodo
public void buscar(int x, Nodo_Arbol t)
        {
            if (t != null)
            {
                if (x == t.info)
                {
MessageBox.Show("Nodo encontrado en la posición X: "+ t.CoordenadaX +" Y:"+t.CoordenadaY);
                    encontrado(t);
                }
                else
```

```
            {
                if (x < t.info)  //búsqueda en el subárbol izquierdo
                {
                    buscar(x, t.Izquierdo);
                }

                else
                {
                    if (x > t.info)  //búsqueda en el subárbol derecho
                    {
                        buscar(x, t.Derecho);
                    }
                }
            }
        }

        else
        {
            MessageBox.Show("Nodo NO encontrado", "Error de búsqueda");
        }
    }
```

4. A continuación se agregan las funciones que permiten dibujar el Árbol Binario en el formulario. Siempre en la misma clase **"Nodo Arbol"**, agregue el siguiente código:

```
//Función posición nodo (donde se ha creado dibujo del nodo)
        public void PosicionNodo(ref int xmin, int ymin)
        {   int aux1, aux2;
            CoordenadaY = (int)(ymin + Radio / 2);

             //obtiene la posición del sub-árbol izquierdo
            if (Izquierdo != null)
            {
                Izquierdo.PosicionNodo(ref xmin, ymin + Radio + DistanciaV);
            }

            if ((Izquierdo != null) && (Derecho != null))
            {
                xmin += DistanciaH;
            }
        //si existe nodo derecho y el nodo izquierdo deja un espacio entre ellos
            if (Derecho != null)
            {
                Derecho.PosicionNodo(ref xmin, ymin + Radio + DistanciaV);
            }

            if (Izquierdo != null && Derecho != null)
          CoordenadaX = (int) ((Izquierdo.CoordenadaX + Derecho.CoordenadaX) / 2);
            else
                if (Izquierdo != null)
            {   aux1 = Izquierdo.CoordenadaX;
                Izquierdo.CoordenadaX = CoordenadaX - 80;
                CoordenadaX = aux1;
            }
```

```
            else
                    if (Derecho != null)
            {   aux2 = Derecho.CoordenadaX;
                  //no hay nodo izquierdo, centrar en nodo derecho
                Derecho.CoordenadaX = CoordenadaX + 80;
                CoordenadaX = aux2;
            }
                    else
                    {   CoordenadaX = (int) (xmin + Radio / 2);xmin += Radio;
                    }
        }

       //Función para dibujar las ramas de los nodos izquierdo y derecho
        public void DibujarRamas(Graphics grafo,Pen Lapiz)
        {
            if
             (Izquierdo != null)
            // Dibujará rama izquierda
            {
                grafo.DrawLine(Lapiz, CoordenadaX, CoordenadaY,
Izquierdo.CoordenadaX,Izquierdo.CoordenadaY);
                Izquierdo.DibujarRamas(grafo, Lapiz);
            }
            if
             (Derecho !=null)
            // Dibujará rama derecha
            {
                grafo.DrawLine(Lapiz, CoordenadaX, CoordenadaY,
Derecho.CoordenadaX,Derecho.CoordenadaY);
                Derecho.DibujarRamas(grafo, Lapiz);
            }
        }

       //Función para dibujar el nodo en la posición especificada
        public void DibujarNodo(Graphics grafo,Font fuente,Brush Relleno,Brush
RellenoFuente,Pen Lapiz,Brush encuentro)
        {
            col = grafo;
            // Dibuja el contorno del nodo
            Rectangle rect = new Rectangle ((int)(CoordenadaX - Radio / 2), (
int)(CoordenadaY - Radio / 2), Radio, Radio);
            Rectangle prueba = new Rectangle((int)(CoordenadaX - Radio / 2),
(int)(CoordenadaY - Radio/ 2), Radio, Radio);
            grafo.FillEllipse(encuentro, rect);
            grafo.FillEllipse(Relleno, rect);
            grafo.DrawEllipse(Lapiz, rect);
            // Para dibujar el nombre del nodo, es decir el contenido
            StringFormat formato = new StringFormat();
            formato.Alignment =StringAlignment.Center;
            formato.LineAlignment =StringAlignment.Center;
            grafo.DrawString(info.ToString(), fuente, RellenoFuente, CoordenadaX,CoordenadaY,
formato);
            //Dibuja los nodos hijos derecho e izquierdo.
```

```
            if(Izquierdo != null)
            {
                Izquierdo.DibujarNodo(grafo, fuente, Relleno, RellenoFuente, Lapiz,
encuentro);
            }
            if(Derecho !=null)
            {
                Derecho.DibujarNodo(grafo, fuente, Relleno, RellenoFuente, Lapiz, encuentro);
            }
        }

        public void colorear(Graphics grafo,Font fuente,Brush Relleno,Brush RellenoFuente,Pen
Lapiz)
        {
            //Dibuja el contorno del nodo.
            Rectangle rect = new Rectangle (( int)(CoordenadaX - Radio / 2), (
int)(CoordenadaY- Radio / 2), Radio, Radio);
            Rectangle prueba =new Rectangle((int)(CoordenadaX - Radio / 2), (int)(CoordenadaY
- Radio/ 2), Radio, Radio);
            grafo.FillEllipse(Relleno, rect);
            grafo.DrawEllipse(Lapiz, rect);
            //Dibuja el nombre
            StringFormat formato = new StringFormat();
            formato.Alignment =StringAlignment.Center;
            formato.LineAlignment = StringAlignment.Center;
            grafo.DrawString(info.ToString(), fuente, RellenoFuente, CoordenadaX,CoordenadaY,
formato);
        }
        //Verificar altura del árbol
          private static int Alturas(Nodo_Arbol t)
        {
            return t == null ? -1 : t.altura;
        }

        public void encontrado(Nodo_Arbol t)
        {
            Rectangle rec = new Rectangle(t.CoordenadaX, t.CoordenadaY, 40, 40)
        }
      }//Clase
    }// Fin Namespace
```

5. Agregar una clase al proyecto, se sugiere darle el nombre de ***"Arbol Binario".*** Esta clase se utiliza para definir la estructura "Arbol". Agregue el código

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;
using System.Diagnostics;
using System.Threading;

namespace Arbol_Binario
{
```

```
    class Arbol_Binario
    {
        public Nodo_Arbol Raiz;
        public Nodo_Arbol aux;

        public Arbol_Binario()
        {
            aux =new Nodo_Arbol();
        }

        public Arbol_Binario(Nodo_Arbol nueva_raiz)
        {
            Raiz = nueva_raiz;
        }
        // Función para agregar un nuevo nodo (valor) al Árbol Binario.
        public void Insertar(int x)
        {
            if (Raiz == null)
            {
                Raiz = new Nodo_Arbol(x, null, null, null);
                Raiz.nivel = 0;
            }
            else
                Raiz = Raiz.Insertar(x, Raiz, Raiz.nivel);
        }
        // Función para eliminar un nodo (valor) del Árbol Binario.
        public void Eliminar(int x)
        {
            if (Raiz ==  null)
                Raiz =new Nodo_Arbol(x,null,null,null);
            else
                Raiz.Eliminar(x,ref Raiz);
        }

        public void Buscar(int x)
        {
            if (Raiz != null)
            {
                Raiz.buscar(x, Raiz);
            }

        }
```

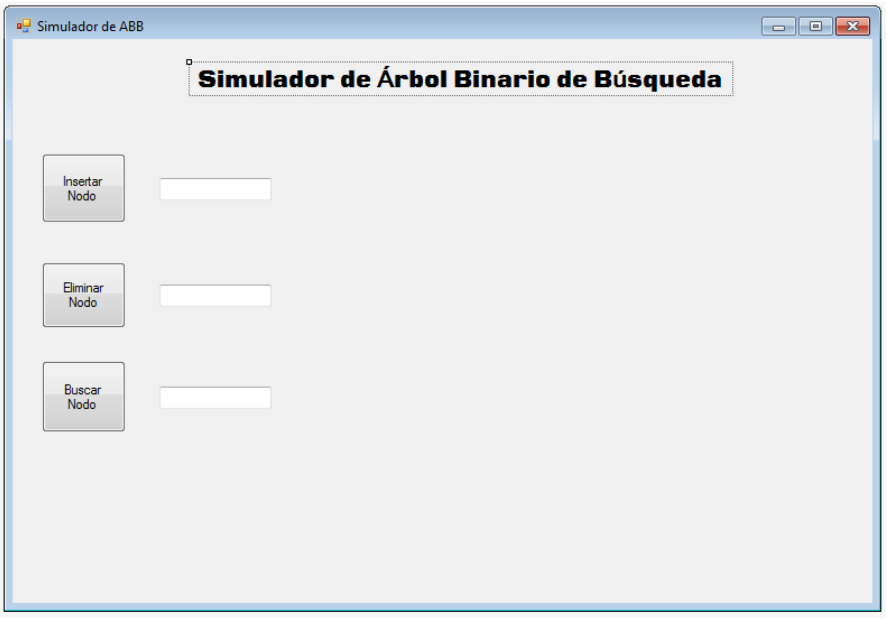
6. A continuación agregar funciones que servirán para dibujar el Árbol Binario en el formulario. Siempre dentro de la clase **"Arbol Binario",** agregue el siguiente código:

```
//*******Funciones para dibujar el árbol binario en el formulario **********
//Función para dibujar árbol binario
public void DibujarArbol(Graphics grafo,Font fuente,Brush Relleno,Brush RellenoFuente,Pen Lapiz,Brush
encuentro)
        {
            int x = 400; // Posiciones de la raíz del árbol
            int y = 75;
            if (Raiz == null )
                return;
            Raiz.PosicionNodo( ref x, y); //Posición de cada nodo
            Raiz.DibujarRamas(grafo, Lapiz); //Dibuja los Enlaces entre nodos
            //Dibuja todos los Nodos
            Raiz.DibujarNodo(grafo, fuente, Relleno, RellenoFuente, Lapiz, encuentro);
        }
```

```
        public int x1 = 400;
        // Posiciones iniciales de la raíz del árbol
        public int y2 = 75;
        // Función para Colorear los nodos
        public void colorear(Graphics grafo, Font fuente, Brush Relleno, Brush RellenoFuente, Pen
Lapiz, Nodo_Arbol Raiz, bool post, bool inor, bool preor)
        {
            Brush entorno = Brushes.Red;
            if(inor == true )
            {
                if (Raiz != null)
                {
           colorear(grafo, fuente, Relleno, RellenoFuente, Lapiz, Raiz.Izquierdo, post, inor, preor);
                    Raiz.colorear(grafo, fuente, entorno, RellenoFuente, Lapiz);
                    Thread.Sleep(1000);

                    // pausar la ejecución 1000 milisegundos
                    Raiz.colorear(grafo, fuente, Relleno, RellenoFuente, Lapiz);
             colorear(grafo, fuente, Relleno, RellenoFuente, Lapiz, Raiz.Derecho, post, inor,preor);
                }
            }
            else
                if (preor == true)
                {
                    if
                     (Raiz != null)
                    {
                        Raiz.colorear(grafo, fuente, entorno, RellenoFuente, Lapiz);
                        Thread.Sleep(1000);
                        // pausar la ejecución 1000 milisegundos
                        Raiz.colorear(grafo, fuente, Relleno, RellenoFuente, Lapiz);
                        colorear(grafo, fuente, Relleno, RellenoFuente, Lapiz, Raiz.Izquierdo, post,
                                       inor, preor);
                        colorear(grafo, fuente, Relleno, RellenoFuente, Lapiz, Raiz.Derecho, post,
                                       inor, preor);
                    }
                }
                else
                    if
                     (post ==true)
                    {
                        if
                         (Raiz !=null)
                        {
            colorear(grafo, fuente, Relleno, RellenoFuente, Lapiz, Raiz.Izquierdo, post,inor, preor);
            colorear(grafo, fuente, Relleno, RellenoFuente, Lapiz, Raiz.Derecho, post, inor, preor);
                  Raiz.colorear(grafo, fuente, entorno, RellenoFuente, Lapiz);
                  Thread.Sleep(1000); // pausar la ejecución 1000 milisegundos
                  Raiz.colorear(grafo, fuente, Relleno, RellenoFuente, Lapiz);
                        }
                    }
        }
    }
}
```

7. Ahora se dibujará el formulario para implementar el simulador del Árbol Binario de Búsqueda (ABB) que queremos realizar. El diseño del mismo dependerá de cada estudiante y su distribución, sin embargo en la figura siguiente se muestra cómo podría lucir el simulador.



1 label

3 buttom

3 textbox

8. Una vez definido el diseño, se le proporciona el código que debe ir en el formulario

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Arbol_Binario
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        //Declaración de variables a utilizar
        int Dato = 0;
        int cont = 0;
        Arbol_Binario mi_Arbol = new Arbol_Binario(null);   //Creación del objeto Árbol
        Graphics g;                                         //Definición del objeto gráfico

         //Evento del formulario que permitirá dibujar el Árbol Binario
        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            e.Graphics.Clear(this.BackColor);
            e.Graphics.TextRenderingHint = System.Drawing.Text.TextRenderingHint.AntiAliasGridFit;
            e.Graphics.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
            g = e.Graphics;
```

```
        mi_Arbol.DibujarArbol(g, this.Font, Brushes.Blue, Brushes.White, Pens.Black, Brushes.White);

        }

        /*Evento que permitirá insertar un nodo al árbol (código de botón "Insertar Nodo" del
        formulario mostrado en la figura) */
        private void btnInsertar_Click(object sender, EventArgs e)
        {
            if (txtDato.Text == "")
            {
                MessageBox.Show("Debe Ingresar un Valor");
            }
            else
            {
                Dato = int.Parse(txtDato.Text);
                if (Dato <= 0 || Dato >= 100)
              MessageBox.Show("Solo Recibe Valores desde 1 hasta 99", "Error de Ingreso");
                else
                {
                    mi_Arbol.Insertar(Dato);
                    txtDato.Clear();
                    txtDato.Focus();
                    cont++;
                    Refresh();
                    Refresh();
                }
            }
        }

        /*Evento que permitirá eliminar un nodo del árbol (código de botón "Eliminar Nodo" del
        formulario mostrado en la figura) */
        private void btnEliminar_Click(object sender, EventArgs e)
        {
            if (txtEliminar.Text == "")
            {
                MessageBox.Show("Debe ingresar el valor a eliminar");
            }
            else
            {
                Dato = Convert.ToInt32(txtEliminar.Text);
                if (Dato <= 0 || Dato >= 100)
                {
            MessageBox.Show("Sólo se adminten valores entre 1 y 99", "Error de Ingreso");
                }
                else
                {
                    mi_Arbol.Eliminar(Dato);
                    txtEliminar.Clear();
                    txtEliminar.Focus();
                    cont--;
                    Refresh();
                    Refresh();
                }
            }
        }

        /*Evento que permitirá buscar un nodo en el árbol (código de botón "Buscar Nodo" del
        formulario mostrado en la figura) */
        private void btnBuscar_Click(object sender, EventArgs e)
        {
            if (txtBuscar.Text == "")
```

```
            {
                MessageBox.Show("Debe ingresar el valor a buscar");
            }
            else
            {
                Dato = Convert.ToInt32(txtBuscar.Text);
                if (Dato <= 0 || Dato >= 100)
                {
                    MessageBox.Show("Sólo se admiten valores entre 1 y 99", "Error de Ingreso");
                }
                else
                {
                    mi_Arbol.Buscar(Dato);
                    txtBuscar.Clear();
                    txtBuscar.Focus();
                    Refresh();
                    Refresh();
                }
            }
        }
    }
}
```

Desarrollo de habilidades

Ejercicio 1

1. Realizar las modificaciones necesarias, para que el simulador de ABB realice los siguientes recorridos:
   - ✓ Recorrido en orden
   - ✓ Recorrido Pre-orden
   - ✓ Recorrido Post-orden

   NOTA: Muestre los recorridos en un label al pie del área de trabajo.

PREORDEN (algoritmo R-I-D)

```
void preOrden(ArbolBinario raíz)
{   if (raiz)
        {       visitar(raiz–>dato);
                preOrden(raiz–>izq);
                preOrden(raiz–>der)
        }
}
```

ENORDEN (algoritmo I-R-D)

```
void enOrden(ArbolBinario raíz)
{  if (raiz)
        {       enOrden(raiz–>izq);
                visitar(raiz–>dato);
                enOrden(raiz–>der);
        }
}
```

POSTORDEN (algoritmo I-D-R)

```
void PostOrden(ArbolBinario raíz)
{   if (raiz)
        {        PostOrden(raiz–>izq);
                 PostOrden(raiz–>der);
                 visitar(raiz–>dato);
        }
}
```

Investigación complementaria

Investigación 1.

Realizar las modificaciones al código anterior para lograr:

1. Determinar la altura del árbol
2. Mostrar la suma de elementos que conforman el árbol (valor de los nodos)
3. Determinar el número de nodos en cualquier momento
4. Determinar la profundidad de un nodo