

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

Tema: “Árboles Balanceados AVL”

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.

Introducción

ÁRBOLES BALANCEADOS

Se considera que un árbol binario está balanceado cuando todos sus niveles, excepto el último, están integrados a la máxima capacidad de nodos.

Existen diferentes propuestas para balancear los árboles y cada una de ellas repercute en la eficiencia de las operaciones de inserción y eliminación de los nodos.

La más común es la técnica de árboles AVL.

ARBOLES AVL

Es un árbol de búsqueda binario que trata de mantenerlo lo más balanceado posible, conforme se realizan operaciones de inserción y eliminación. Fueron propuestos en 1962 por los matemáticos rusos Adelson, Velskii y Landis, de donde surge su nombre.

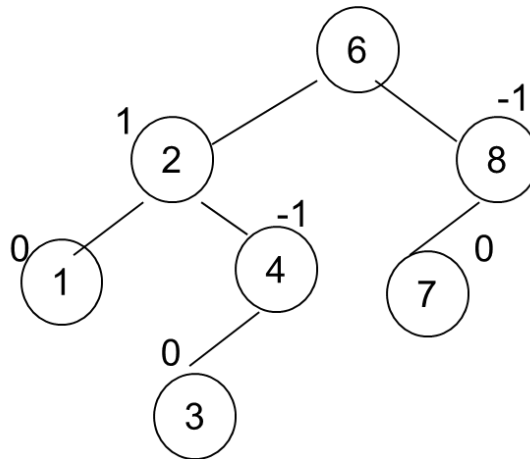
En los árboles AVL se debe cumplir el hecho de que para cualquier nodo del árbol, la diferencia entre las alturas de sus subárboles no exceda una unidad.

Factor de Balanceo

Los nodos de un árbol AVL guardan un valor -1, 0, 1, que se conoce como Factor de Balanceo (FB) y representa la altura entre las alturas de sus subárboles.

Un FB igual a cero de un nodo significa que las alturas de sus subárboles son iguales.

Un FB positivo significa que la altura de su subárbol derecho es mayor al subárbol izquierdo. Un FB negativo significa que la altura de su subárbol izquierdo es mayor al subárbol derecho.



Materiales y Equipo

- Guía de laboratorio N° 7
- Computadora con Visual Studio C#
- Dispositivo de almacenamiento (USB) (Opcional)

Procedimiento

Ejemplo 1. Implementación de un árbol balanceado AVL en C#

1. Crear un proyecto de tipo Windows Form Application, se sugiere que le dé el nombre de "Arbol_AVL".
2. Agregar una clase al proyecto, para facilitar el entendimiento nómbrelo como "ALV".
3. Dentro de la clase creada, agregue el siguiente código:

```

using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;
using System.Windows.Forms;
using System.Threading;

namespace Arbol_AVL
{
    class AVL
    {
        public int valor;
        public AVL NodoIzquierdo;
        public AVL NodoDerecho;
        public AVL NodoPadre;
        public int altura;
        public Rectangle prueba;
        private DibujaAVL arbol;
        public AVL()
        {
        }

        public DibujaAVL Arbol
        {
            get { return arbol; }
            set { arbol = value; }
        }
    }
}

```

```

}
// Constructor.
public AVL(int valorNuevo, AVL izquierdo, AVL derecho, AVL padre)
{
    valor = valorNuevo;
    NodoIzquierdo = izquierdo;
    NodoDerecho = derecho;
    NodoPadre = padre;
    altura = 0;
}

//Funcion para insertar un nuevo valor en el arbol AVL
public AVL Insertar(int valorNuevo, AVL Raiz)
{
    if (Raiz == null)
        Raiz = new AVL(valorNuevo, null, null, null);
    else if (valorNuevo < Raiz.valor)
    {
        Raiz.NodoIzquierdo = Insertar(valorNuevo, Raiz.NodoIzquierdo);
    }
    else if (valorNuevo > Raiz.valor)
    {
        Raiz.NodoDerecho = Insertar(valorNuevo, Raiz.NodoDerecho);
    }
    else
    {
        MessageBox.Show("Valor Existente en el Arbol", "Error", MessageBoxButtons.OK);
    }

    //Realiza las rotaciones simples o dobles segun el caso
    if (Alturas(Raiz.NodoIzquierdo) - Alturas(Raiz.NodoDerecho) == 2)
    {
        if (valorNuevo < Raiz.NodoIzquierdo.valor)
            Raiz = RotacionIzquierdaSimple(Raiz);
        else
            Raiz = RotacionIzquierdaDoble(Raiz);
    }

    if (Alturas(Raiz.NodoDerecho) - Alturas(Raiz.NodoIzquierdo) == 2)
    {
        if (valorNuevo > Raiz.NodoDerecho.valor)
            Raiz = RotacionDerechaSimple(Raiz);
        else
            Raiz = RotacionDerechaDoble(Raiz);
    }

    Raiz.altura = max(Alturas(Raiz.NodoIzquierdo), Alturas(Raiz.NodoDerecho)) + 1;
    return Raiz;
}

//FUNCION DE PRUEBA PARA REALIZAR LAS ROTACIONES

//Función para obtener que rama es mayor
private static int max(int lhs, int rhs)
{
    return lhs > rhs ? lhs : rhs;
}

private static int Alturas(AVL Raiz)
{
    return Raiz == null ? -1 : Raiz.altura;
}

AVL nodoE, nodoP;
public AVL Eliminar(int valorEliminar, ref AVL Raiz)
{
    if (Raiz != null)

```

```

{
    if (valorEliminar < Raiz.valor)
    {
        nodoE = Raiz;
        Eliminar(valorEliminar, ref Raiz.NodoIzquierdo);
    }
    else
    {
        if (valorEliminar > Raiz.valor)
        {
            nodoE = Raiz;
            Eliminar(valorEliminar, ref Raiz.NodoDerecho);
        }
        else
        {
            //Posicionado sobre el elemento a eliminar

            AVL NodoEliminar = Raiz;
            if (NodoEliminar.NodoDerecho == null)
            {
                Raiz = NodoEliminar.NodoIzquierdo;

                if (Alturas(nodoE.NodoIzquierdo) - Alturas(nodoE.NodoDerecho) == 2)
                {
                    //MessageBox.Show("nodoE" + nodoE.valor.ToString());
                    if (valorEliminar < nodoE.valor)
                        nodoP = RotacionIzquierdaSimple(nodoE);
                    else
                        nodoE = RotacionDerechaSimple(nodoE);
                }

                if (Alturas(nodoE.NodoDerecho) - Alturas(nodoE.NodoIzquierdo) == 2)
                {
                    if (valorEliminar > nodoE.NodoDerecho.valor)
                        nodoE = RotacionDerechaSimple(nodoE);
                    else
                        nodoE = RotacionDerechaDoble(nodoE);
                    nodoP = RotacionDerechaSimple(nodoE);
                }
            }
            else
            {
                if (NodoEliminar.NodoIzquierdo == null)
                {
                    Raiz = NodoEliminar.NodoDerecho;
                }
                else
                {
                    if (Alturas(Raiz.NodoIzquierdo) - Alturas(Raiz.NodoDerecho) > 0)
                    {
                        AVL AuxiliarNodo = null;
                        AVL Auxiliar = Raiz.NodoIzquierdo;
                        bool Bandera = false;
                        while (Auxiliar.NodoDerecho != null)
                        {
                            AuxiliarNodo = Auxiliar;
                            Auxiliar = Auxiliar.NodoDerecho;
                            Bandera = true;
                        }
                        Raiz.valor = Auxiliar.valor;
                        NodoEliminar = Auxiliar;
                        if (Bandera == true)
                        {
                            AuxiliarNodo.NodoDerecho = Auxiliar.NodoIzquierdo;
                        }
                    }
                }
            }
        }
    }
}

```

[illegible]

```

    }

    return nodoP;
}

//Seccion de funciones de rotaciones

//Rotacion Izquierda Simple
private static AVL RotacionIzquierdaSimple(AVL k2)
{
    AVL k1 = k2.NodoIzquierdo;
    k2.NodoIzquierdo = k1.NodoDerecho;
    k1.NodoDerecho = k2;
    k2.altura = max(Alturas(k2.NodoIzquierdo), Alturas(k2.NodoDerecho)) + 1;
    k1.altura = max(Alturas(k1.NodoIzquierdo), k2.altura) + 1;
    return k1;
}

//Rotacion Derecha Simple
private static AVL RotacionDerechaSimple(AVL k1)
{
    AVL k2 = k1.NodoDerecho;
    k1.NodoDerecho = k2.NodoIzquierdo;
    k2.NodoIzquierdo = k1;
    k1.altura = max(Alturas(k1.NodoIzquierdo), Alturas(k1.NodoDerecho)) + 1;
    k2.altura = max(Alturas(k2.NodoDerecho), k1.altura) + 1;
    return k2;
}

//Doble Rotacion Izquierda
private static AVL RotacionIzquierdaDoble(AVL k3)
{
    k3.NodoIzquierdo = RotacionDerechaSimple(k3.NodoIzquierdo);
    return RotacionIzquierdaSimple(k3);
}

//Doble Rotacion Derecha
private static AVL RotacionDerechaDoble(AVL k1)
{
    k1.NodoDerecho = RotacionIzquierdaSimple(k1.NodoDerecho);
    return RotacionDerechaSimple(k1);
}

//Funcion para obtener la altura del arbol
public int getAltura(AVL nodoActual)
{
    if (nodoActual == null)
        return 0;
    else
        return 1 + Math.Max(getAltura(nodoActual.NodoIzquierdo), getAltura(nodoActual.NodoDerecho));
}

//Buscar un valor en el arbol
public void buscar(int valorBuscar, AVL Raiz)
{
    if (Raiz != null)
    {
        if (valorBuscar < Raiz.valor)
        {
            buscar(valorBuscar, Raiz.NodoIzquierdo);
        }
        else
        {
            if (valorBuscar > Raiz.valor)
            {
                buscar(valorBuscar, Raiz.NodoDerecho);
            }
        }
    }
}

```

```

    }
}
else
    MessageBox.Show("Valor no encontrado", "Error", MessageBoxButtons.OK);
}
/*+++++++FUNCIONES PARA DIBUJAR EL ÁRBOL ++++++*/

private const int Radio = 30;
private const int DistanciaH = 40;
private const int DistanciaV = 10;

private int CoordenadaX;
private int CoordenadaY;

//Encuentra la posición en donde debe crearse el 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 (NodoIzquierdo != null)
    {
        NodoIzquierdo.PosicionNodo(ref xmin, ymin + Radio + DistanciaV);
    }
    if ((NodoIzquierdo != null) && (NodoDerecho != null))
    {
        xmin += DistanciaH;
    }

    //Si existe el nodo derecho e izquierdo deja un espacio entre ellos.
    if (NodoDerecho != null)
    {
        NodoDerecho.PosicionNodo(ref xmin, ymin + Radio + DistanciaV);
    }

    // Posicion de nodos dercho e izquierdo.
    if (NodoIzquierdo != null)
    {
        if (NodoDerecho != null)
        {
            //centro entre los nodos.
            CoordenadaX = (int)((NodoIzquierdo.CoordenadaX + NodoDerecho.CoordenadaX) / 2);
        }
        else
        {
            // no hay nodo derecho. centrar al nodo izquierdo.
            aux1 = NodoIzquierdo.CoordenadaX;
            NodoIzquierdo.CoordenadaX = CoordenadaX - 40;
            CoordenadaX = aux1;
        }
    }
    else if (NodoDerecho != null)
    {
        aux2 = NodoDerecho.CoordenadaX;
        //no hay nodo izquierdo.centrar al nodo derecho.
        NodoDerecho.CoordenadaX = CoordenadaX + 40;
        CoordenadaX = aux2;
    }
    else
    {
        // Nodo hoja
        CoordenadaX = (int)(xmin + Radio / 2);
        xmin += Radio;
    }
}

```

```

    }

    // Dibuja las ramas de los nodos izquierdo y derecho
    public void DibujarRamas(Graphics grafo, Pen Lapis)
    {
        if (NodoIzquierdo != null)
        {
            grafo.DrawLine(Lapis, CoordinadaX, CoordinadaY, NodoIzquierdo.CoordenadaX,
NodoIzquierdo.CoordenadaY);
            NodoIzquierdo.DibujarRamas(grafo, Lapis);
        }
        if (NodoDerecho != null)
        {
            grafo.DrawLine(Lapis, CoordinadaX, CoordinadaY, NodoDerecho.CoordenadaX, NodoDerecho.CoordenadaY);
            NodoDerecho.DibujarRamas(grafo, Lapis);
        }
    }

    //Dibuja el nodo en la posición especificada.
    public void DibujarNodo(Graphics grafo, Font fuente, Brush Relleno, Brush RellenoFuente, Pen Lapis, int
dato, Brush encuentro)
    {
        //Dibuja el contorno del nodo.

        Rectangle rect = new Rectangle(
            (int)(CoordenadaX - Radio / 2),
            (int)(CoordenadaY - Radio / 2),
            Radio, Radio);

        if (valor == dato)
        {
            grafo.FillEllipse(encuentro, rect);
        }
        else
        {
            grafo.FillEllipse(encuentro, rect);
            grafo.FillEllipse(Relleno, rect);
        }
        grafo.DrawEllipse(Lapis, rect);

        //Dibuja el valor del nodo.
        StringFormat formato = new StringFormat();

        formato.Alignment = StringAlignment.Center;
        formato.LineAlignment = StringAlignment.Center;
        grafo.DrawString(valor.ToString(), fuente, Brushes.Black, CoordinadaX, CoordinadaY, formato);

        //Dibuja los nodos hijos derecho e izquierdo.

        if (NodoIzquierdo != null)
        {
            NodoIzquierdo.DibujarNodo(grafo, fuente, Brushes.YellowGreen, RellenoFuente, Lapis, dato,
encuentro);
        }

        if (NodoDerecho != null)
        {
            NodoDerecho.DibujarNodo(grafo, fuente, Brushes.Yellow, RellenoFuente, Lapis, dato, encuentro);
        }
    }

    public void colorear(Graphics grafo, Font fuente, Brush Relleno, Brush RellenoFuente, Pen Lapis)
    {
        //Dibuja el contorno del nodo.

        Rectangle rect = new Rectangle(
            (int)(CoordenadaX - Radio / 2),

```



```

        (int)(CoordenadaY - Radio / 2), Radio, Radio);
        prueba = new Rectangle((int)(CoordenadaX - Radio / 2), (int)(CoordenadaY - Radio / 2),
            Radio, Radio);

        //Dibuja el nombre.
        StringFormat formato = new StringFormat();

        formato.Alignment = StringAlignment.Center;
        formato.LineAlignment = StringAlignment.Center;

        grafo.DrawEllipse(Lapiz, rect);
        grafo.FillEllipse(Brushes.PaleVioletRed, rect);
        grafo.DrawString(valor.ToString(), fuente, Brushes.Black, CoordenadaX, CoordenadaY, formato);
    }
}

```

4. A continuación se agrega otra clase al proyecto, denominada DibujaAVL.cs

```

using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;
using System.Windows.Forms;
using System.Threading;

namespace Arbol_AVL
{
    class DibujaAVL
    {
        public AVL Raiz;
        public AVL aux;

        // Constructor.
        public DibujaAVL()
        {
            aux = new AVL();
        }

        public DibujaAVL(AVL RaizNueva)
        {
            Raiz = RaizNueva;
        }

        // Agrega un nuevo valor al arbol.
        public void Insertar(int dato)
        {
            if (Raiz == null)
                Raiz = new AVL(dato, null, null, null);
            else
                Raiz = Raiz.Insertar(dato, Raiz);
        }

        //Eliminar un valor del arbol
        public void Eliminar(int dato)
        {
            if (Raiz == null)
                Raiz = new AVL(dato, null, null, null);
            else
                Raiz.Eliminar(dato, ref Raiz);
        }

        private const int Radio = 30;
        private const int DistanciaH = 40;
        private const int DistanciaV = 10;

        private int CoordenadaX;
    }
}

```

```

private int CoordenadaY;

public void PosicionNodoreocrrido(ref int xmin, ref int ymin)
{
    CoordenadaY = (int)(ymin + Radio / 2);
    CoordenadaX = (int)(xmin + Radio / 2);
    xmin += Radio;
}

public void colorear(Graphics grafo, Font fuente, Brush Relleno, Brush RellenoFuente, Pen Lapis, AVL Raiz,
bool post, bool inor, bool preor)
{
    Brush entorno = Brushes.Red;

    if (inor == true)
    {
        if (Raiz != null)
        {
            colorear(grafo, fuente, Brushes.Blue, RellenoFuente, Lapis, Raiz.NodoIzquierdo, post, inor,
preor);

            Raiz.colorear(grafo, fuente, entorno, RellenoFuente, Lapis);
            Thread.Sleep(500);
            Raiz.colorear(grafo, fuente, Relleno, RellenoFuente, Lapis);
            colorear(grafo, fuente, Relleno, RellenoFuente, Lapis, Raiz.NodoDerecho, post, inor, preor);
        }
    }
    else if (preor == true)
    {
        if (Raiz != null)
        {
            Raiz.colorear(grafo, fuente, Brushes.Yellow, Brushes.Blue, Pens.Black);
            Thread.Sleep(500);
            Raiz.colorear(grafo, fuente, Brushes.White, Brushes.Black, Pens.Black);
            colorear(grafo, fuente, Brushes.Blue, RellenoFuente, Lapis, Raiz.NodoIzquierdo, post, inor,
preor);

            colorear(grafo, fuente, Relleno, RellenoFuente, Lapis, Raiz.NodoDerecho, post, inor, preor);
        }
    }
    else if (post == true)
    {
        if (Raiz != null)
        {
            colorear(grafo, fuente, Relleno, RellenoFuente, Lapis, Raiz.NodoIzquierdo, post, inor, preor);
            colorear(grafo, fuente, Relleno, RellenoFuente, Lapis, Raiz.NodoDerecho, post, inor, preor);
            Raiz.colorear(grafo, fuente, entorno, RellenoFuente, Lapis);
            Thread.Sleep(500);
            Raiz.colorear(grafo, fuente, entorno, RellenoFuente, Lapis);
        }
    }
}

public void colorearB(Graphics grafo, Font fuente, Brush Relleno, Brush RellenoFuente, Pen Lapis, AVL Raiz,
int busqueda)
{
    Brush entorno = Brushes.Red;
    if (Raiz != null)
    {
        Raiz.colorear(grafo, fuente, entorno, RellenoFuente, Lapis);

        if (busqueda < Raiz.valor)
        {
            Thread.Sleep(500);
            Raiz.colorear(grafo, fuente, entorno, Brushes.Blue, Lapis);
            colorearB(grafo, fuente, Relleno, RellenoFuente, Lapis, Raiz.NodoIzquierdo, busqueda);
        }
    }
}

```

```

    }
    else
    {
        if (busqueda > Raiz.valor)
        {
            Thread.Sleep(500);
            Raiz.colorear(grafo, fuente, entorno, RellenoFuente, Lapis);
            colorearB(grafo, fuente, Relleno, RellenoFuente, Lapis, Raiz.NodoDerecho, busqueda);
        }
        else
        {
            Raiz.colorear(grafo, fuente, entorno, RellenoFuente, Lapis);
            Thread.Sleep(500);
        }
    }
}

//Dibuja el árbol
public void DibujarArbol(Graphics grafo, Font fuente, Brush Relleno, Brush RellenoFuente, Pen Lapis, int
dato, Brush encuentro)
{
    int x = 100;
    int y = 75;
    if (Raiz == null) return;

    //Posicion de todos los Nodos.
    Raiz.PosicionNodo(ref x, y);

    //Dibuja los Enlaces entre nodos.
    Raiz.DibujarRamas(grafo, Lapis);

    //Dibuja todos los Nodos.
    Raiz.DibujarNodo(grafo, fuente, Relleno, RellenoFuente, Lapis, dato, encuentro);
}

public int x1 = 100;
public int y2 = 75;
public void restablecer_valores()
{
    x1 = 100;
    y2 = 75;
}

public void buscar(int x)
{
    if (Raiz == null)
        MessageBox.Show("Arbol AVL Vacío", "Error", MessageBoxButtons.OK);
    else
        Raiz.buscar(x, Raiz);
}
}
}

```

- Ahora se dibujará el formulario para implementar el simulador del Árbol Balanceado AVL (AVL) que queremos realizar. El diseño del mismo dependerá de cada estudiante y su distribución, sin embargo en la figura siguiente se muestra cómo se sugiere que luzca el simulador. (Agregue también un error provider que se encuentra en el toolbox)

No necesariamente debe eliminar el borde de la ventana, eso es opcional.

6. 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.Text;
using System.Windows.Forms;
using System.Threading;

namespace Arbol_AVL
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        int cont = 0;
        int dato = 0;
        int datb = 0;
        int cont2 = 0;

        DibujaAVL arbolAVL = new DibujaAVL(null);
        DibujaAVL arbolAVL_Letra = new DibujaAVL(null);
        Graphics g;

        //No olvidar generar el evento Paint que es importante para dibujar en el form
        private void Form1_Paint(object sender, PaintEventArgs en)
        {
            en.Graphics.Clear(this.BackColor);
            en.Graphics.TextRenderingHint = System.Drawing.Text.TextRenderingHint.AntiAliasGridFit;
            en.Graphics.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
            g = en.Graphics;

            arbolAVL.DibujarArbol(g, this.Font,
                Brushes.White, Brushes.Black, Pens.White, datb, Brushes.Black);
            datb = 0;
        }
    }
}
```

```

        if (pintaR == 1)
        {
            arbolAVL.colorear(g, this.Font, Brushes.Black, Brushes.Yellow, Pens.Blue, arbolAVL.Raiz, post.Checked,
            ino.Checked, pre.Checked);
            pintaR = 0;
        }
        if (pintaR == 2)
        {
            arbolAVL.colorearB(g, this.Font, Brushes.White, Brushes.Red, Pens.White, arbolAVL.Raiz,
            int.Parse(valor.Text));
            pintaR = 0;
        }
    }

    private void btnIngresar_Click(object sender, EventArgs e)
    {
        errores.Clear();
        if (valor.Text == "")
        {
            errores.SetError(valor, "Valor obligatorio");
        }
        else
        {
            try{
                dato = int.Parse(valor.Text);

                arbolAVL.Insertar(dato);
                valor.Clear();
                valor.Focus();
                lblaltura.Text = arbolAVL.Raiz.getAltura(arbolAVL.Raiz).ToString();
                cont++;
                Refresh();
                Refresh();
            }catch(Exception ex){
                errores.SetError(valor, "Debe ser numérico");
            }
        }
    }

    int pintaR = 0;
    private void btnBuscar_Click(object sender, EventArgs e)
    {
        errores.Clear();
        if (valor.Text == "")
        {
            errores.SetError(valor, "Valor obligatorio");
        }
        else
        {
            try{
                datb = int.Parse(valor.Text);
                arbolAVL.buscar(datb);
                pintaR = 2;
                Refresh();
                valor.Clear();
            }catch(Exception ex){
                errores.SetError(valor, "Debe ser numérico");
            }
        }
    }

    private void btneliminar_Click(object sender, EventArgs e)
    {
        errores.Clear();
        if (valor.Text == "")
        {
            errores.SetError(valor, "Valor obligatorio");
        }
    }

```

```

else
{
    try{
        dato = int.Parse(valor.Text);
        valor.Clear();
        arbolAVL.Eliminar(dato);
        lblaltura.Text = arbolAVL.Raiz.getAltura(arbolAVL.Raiz).ToString();
        Refresh();
        Refresh();
        cont2++;
    }catch(Exception ex){

        errores.SetError(valor, "Debe ser numérico");
    }
}
Refresh(); Refresh(); Refresh();
}

//Al presionar enter que realice las operaciones
private void valor_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == Convert.ToChar(Keys.Enter))
    {
        errores.Clear();

        if (valor.Text == "")
        {
            errores.SetError(valor, "Valor obligatorio");
        }
        else
        {
            try
            {
                dato = int.Parse(valor.Text);
                if (dato > 0)
                {
                    arbolAVL.Insertar(dato);

                    valor.Clear();
                    valor.Focus();
                    lblaltura.Text = arbolAVL.Raiz.getAltura(arbolAVL.Raiz).ToString();
                    cont++;
                    Refresh();
                    Refresh();
                }
                else
                {
                    errores.SetError(valor, "Debe ser un numero mayor que 0");
                }
            }
            catch (Exception ex)
            {
                errores.SetError(valor, "Debe ser numérico");
            }
        }
    }
}

private void bsalir_Click_1(object sender, EventArgs e)
{
    this.Close();
}
}

```

Desarrollo de habilidades

1. Realizar las modificaciones necesarias, para que el simulador de AVL realice los siguientes recorridos:
 - ✓ Recorrido en orden
 - ✓ Recorrido Pre-orden
 - ✓ Recorrido Post-orden
2. Tomando como base el ejercicio dado, modifique para:
 - ✓ Incluir en el formulario el valor del Factor de Balanceo por cada nodo, cada vez que se inserte o se elimine un dato
 - ✓ Muestre al usuario qué tipo de rotación se realizará (RSI-RSD-RDI-RDD)

Investigación Complementaria

1. Realice un proyecto similar al presentado, pero que permita al usuario elegir si crear un árbol de números (como en el ejemplo) o un árbol de caracteres. Comente aquellas secciones modificadas que considera importantes.
2. Investigue qué otros métodos o técnicas existen para realizar Balanceo de Árboles, e incluya un ejemplo visual.