C# Y .NET 10
Parte 7. Estructuras de datos de bajo nivel

2026-01

Rafael Alberto Moreno Parra
ramsoftware@gmail.com

Contenido

Tabla de ilustraciones

Acerca del autor

Rafael Alberto Moreno Parra

ramsoftware@gmail.com o enginelife@hotmail.com

Sitio Web: http://darwin.50webs.com (dedicado a la investigación de algoritmos evolutivos y vida artificial).

Github: https://github.com/ramsoftware

Youtube: https://www.youtube.com/@RafaelMorenoP

Licencia de este libro

Licencia del software

Todo el software desarrollado aquí tiene licencia LGPL "Lesser General Public License" [1]

Marcas registradas

En este libro se hace uso de las siguientes tecnologías registradas:

Microsoft ® Windows ® Enlace: http://windows.microsoft.com/en-US/windows/home

Microsoft ® Visual Studio 2026 ® Enlace: https://visualstudio.microsoft.com/es/vs/

Dedicatoria

A mis padres, a mi hermana….

Y a mi tropa gatuna: Suini, Grisú, Milú, Arián, Frac y mis recordados Sally, Capuchina, Tinita, Tammy, Vikingo y Michu.

Lista simplemente enlazada

Se representa de esta forma:

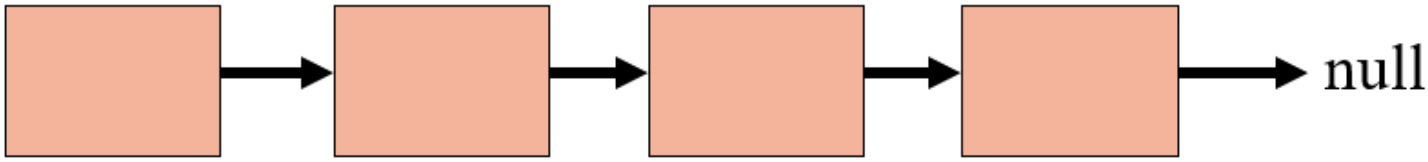

Ilustración 1:Representación gráfica de una lista simplemente enlazada

El rectángulo es el objeto con sus datos, métodos y un apuntador, se le conoce como Nodo

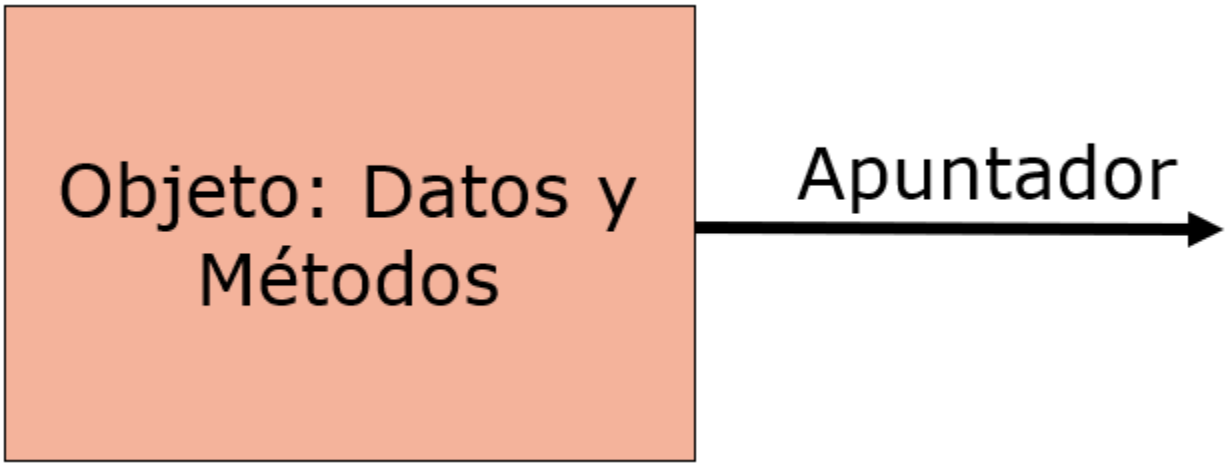

Ilustración 2: El nodo es un objeto con sus atributos, métodos y un apuntador

Esta sería un ejemplo de implementación del Nodo en C#:

G/001.cs

```
namespace Ejemplo;

class Nodo {
   //Atributos propios
   public string Cad { get; set; }
   public char Car { get; set; }
   public int Entero { get; set; }
   public double Num { get; set; }

   //Apuntador para lista simplemente enlazada
   public Nodo Apuntador;

   //Constructor
   public Nodo(string Cad, char Car, int Entero, double Num) {
      this.Cad = Cad;
      this.Car = Car;
      this.Entero = Entero;
      this.Num = Num;
   }

   //Imprime Contenido
   public void Imprime() {
      Console.Write("Cad: " + Cad + " Car: " + Car);
      Console.WriteLine(" Entero: " + Entero + " Real: " + Num);
   }
}

class Program {
   static void Main() {
      //Crea dos nodos separados
      Nodo primero = new("Rafael", 'A', 16, 8.32);
      Nodo segundo = new("Moreno", 'P', 9, 2.9);
      Nodo tercero = new("Sally", 'C', 2010, 7.18);

      //Une el primer nodo con el segundo, creando una simple lista
      primero.Apuntador = segundo;

      //Une el segundo nodo con el tercero, aumentando la lista
      segundo.Apuntador = tercero;

      //Imprime la lista
      primero.Imprime();
      primero.Apuntador.Imprime();
      primero.Apuntador.Apuntador.Imprime();
   }
```

```
}
```

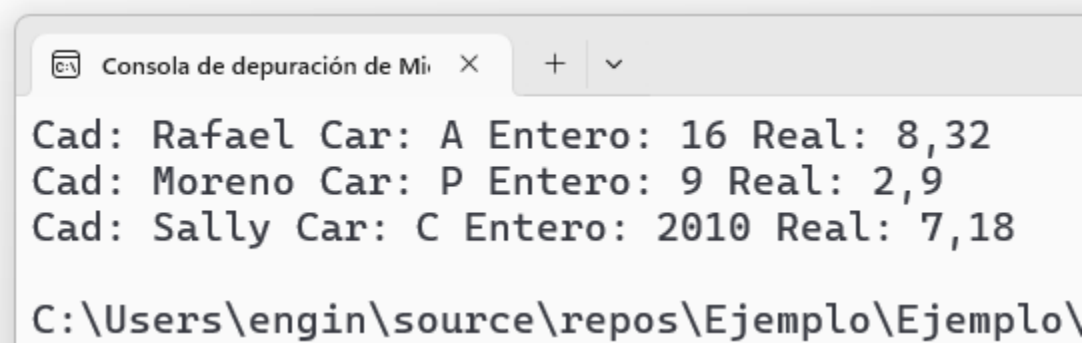

Ilustración 3: Lista simplemente enlazada

Se crean tres nodos:

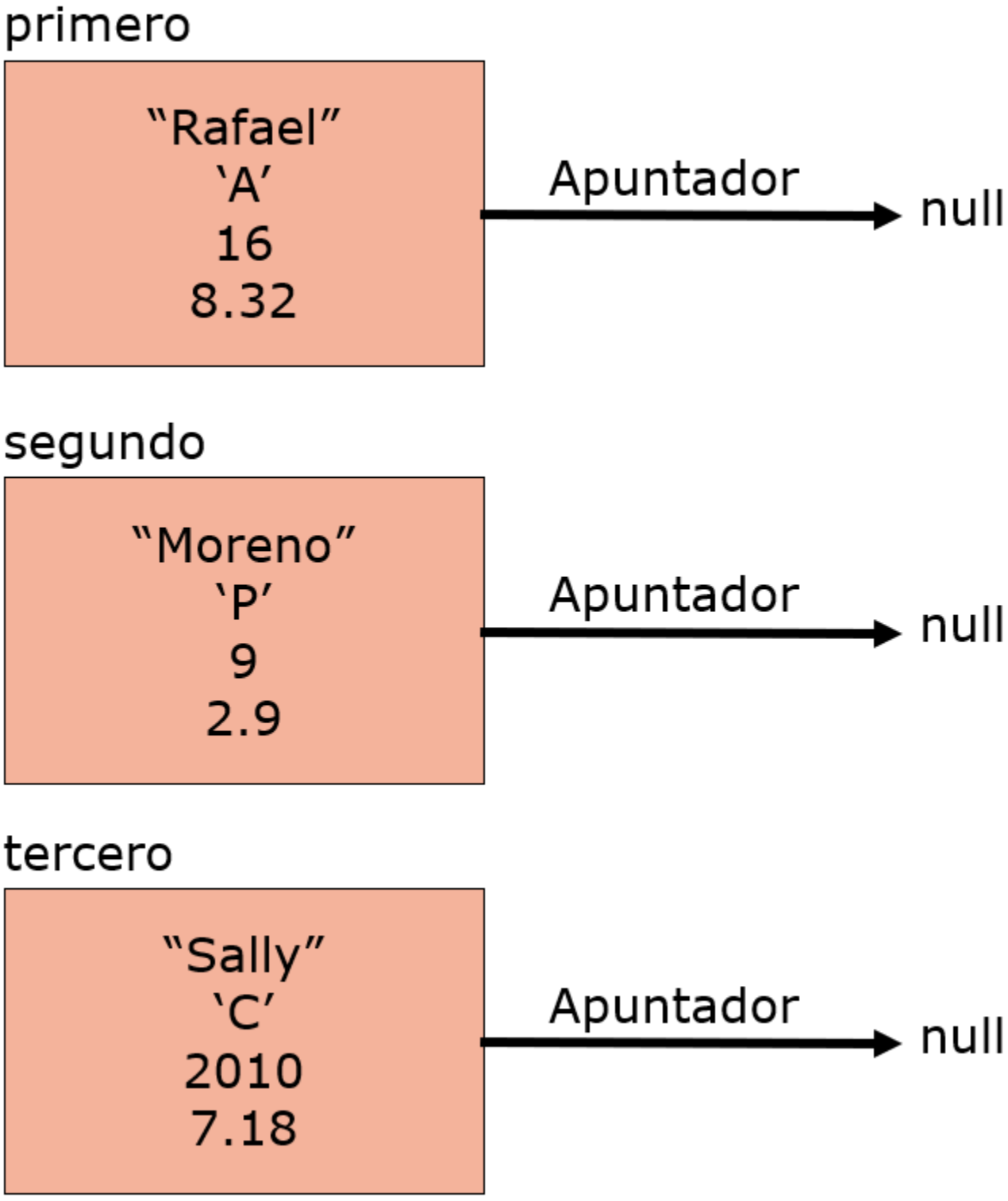

Ilustración 4: Se crean tres nodos

Este sería el código en C#:

```
        Nodo primero = new Nodo("Rafael", 'A', 16, 8.32);
        Nodo segundo = new Nodo("Moreno", 'P', 9, 2.9);
        Nodo tercero = new Nodo("Sally", 'C', 2010, 7.18);
```

Luego debe conectarse el primer nodo con el segundo nodo:

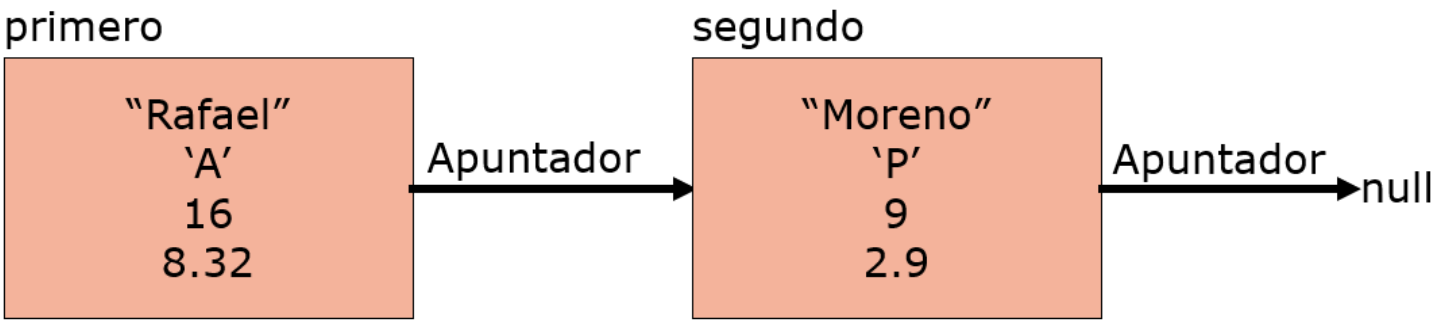

Ilustración 5: Conectar el nodo primero con el nodo segundo

Este sería el código en C#:

```
        //Une el primer nodo con el segundo, creando una simple lista
        primero.Apuntador = segundo;
```

Luego debe conectarse el segundo nodo con el tercer nodo:

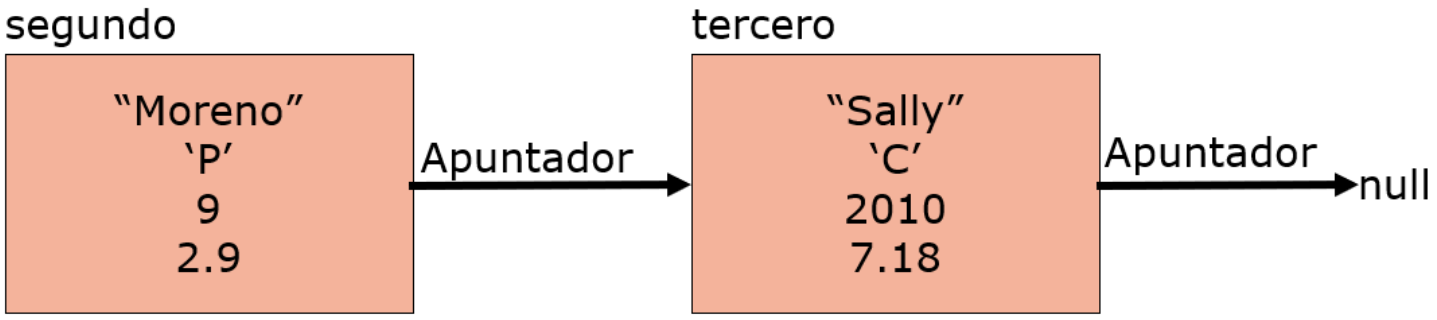

Ilustración 6: Conexión entre nodos

Este sería el código en C#:

```
        //Une el segundo nodo con el tercero, aumentando la lista
        segundo.Apuntador = tercero;
```

Enlazamiento continuo

Con una sola variable declarada, se va armando la lista. En primer lugar, se hacen cambios a la clase Nodo, para que en el constructor se pueda enviar el apuntador.

G/002.cs

```
namespace Ejemplo;

class Nodo {
   //Atributos propios
   public string Cad { get; set; }
   public char Car { get; set; }
   public int Entero { get; set; }
   public double Num { get; set; }

   //Apuntador para lista simplemente enlazada
   public Nodo Apuntador;

   //Constructor
   public Nodo(string Cad, char Car, int Entero,
      double Num, Nodo Apuntador) {
      this.Cad = Cad;
      this.Car = Car;
      this.Entero = Entero;
      this.Num = Num;
      this.Apuntador = Apuntador;
   }

   //Imprime Contenido
   public void Imprime() {
      Console.Write("Cad: " + Cad + " Car: " + Car);
      Console.WriteLine(" Entero: " + Entero + " Real: " + Num);
   }
}

class Program {
   static void Main() {
      //Crea la lista
      Nodo lista = new("aaaa", 'A', 1, 0.1, null);
      lista = new("bbbb", 'B', 2, 0.2, lista);
      lista = new("cccc", 'C', 3, 0.3, lista);

      //Imprime la lista
      lista.Imprime(); //Primer nodo
      lista.Apuntador.Imprime(); //Segundo nodo
      lista.Apuntador.Apuntador.Imprime(); //Tercer nodo
   }
```

```
}
```

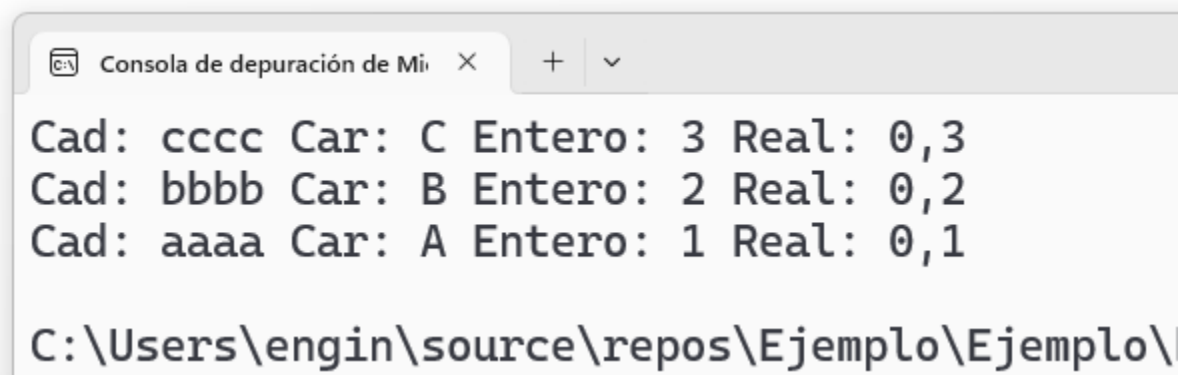

Ilustración 7: Enlazamiento continuo

Recorriendo lista simplemente enlazada

Para recorrer una lista simplemente enlazada, se debe dejar una variable que sostenga la lista y una segunda es la que la recorre. Es importante eso porque sin la variable que sostiene toda la lista, a medida que va recorriendo nodo a nodo, ¡se estará borrando!

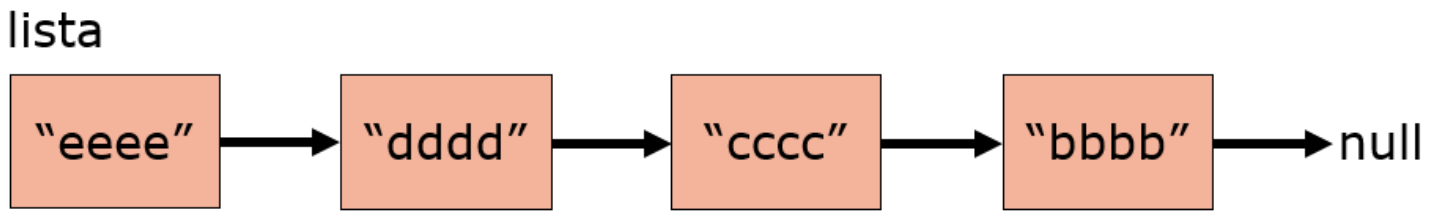

Ilustración 8: Variable "lista" sostiene la lista simplemente enlazada

G/003.cs

```
namespace Ejemplo;

class Nodo {
   //Atributos propios
   public string Cad { get; set; }
   public char Car { get; set; }
   public int Entero { get; set; }
   public double Num { get; set; }

   //Apuntador para lista simplemente enlazada
   public Nodo Apuntador;

   //Constructor
   public Nodo(string Cad, char Car, int Entero,
            double Num, Nodo Apuntador) {
      this.Cad = Cad;
      this.Car = Car;
      this.Entero = Entero;
      this.Num = Num;
      this.Apuntador = Apuntador;
   }

   //Imprime Contenido
   public void Imprime() {
      Console.Write("Cad: " + Cad + " Car: " + Car);
      Console.WriteLine(" Entero: " + Entero + " Real: " + Num);
   }
}
```

```
class Program {
   static void Main() {
      //Crea la lista
      Nodo lista = new("aaaa", 'A', 1, 0.1, null);
      lista = new("bbbb", 'B', 2, 0.2, lista);
      lista = new("cccc", 'C', 3, 0.3, lista);
      lista = new("dddd", 'D', 4, 0.4, lista);
      lista = new("eeee", 'E', 5, 0.5, lista);
      lista = new("ffff", 'F', 6, 0.6, lista);
      lista = new("gggg", 'G', 7, 0.7, lista);
      lista = new("hhhh", 'H', 8, 0.8, lista);
      lista = new("iiii", 'I', 9, 0.9, lista);

      //Pasea la lista, imprimiéndola
      Nodo pasea = lista;
      while (pasea != null) {
         pasea.Imprime();
         pasea = pasea.Apuntador;
      }
   }
}
```

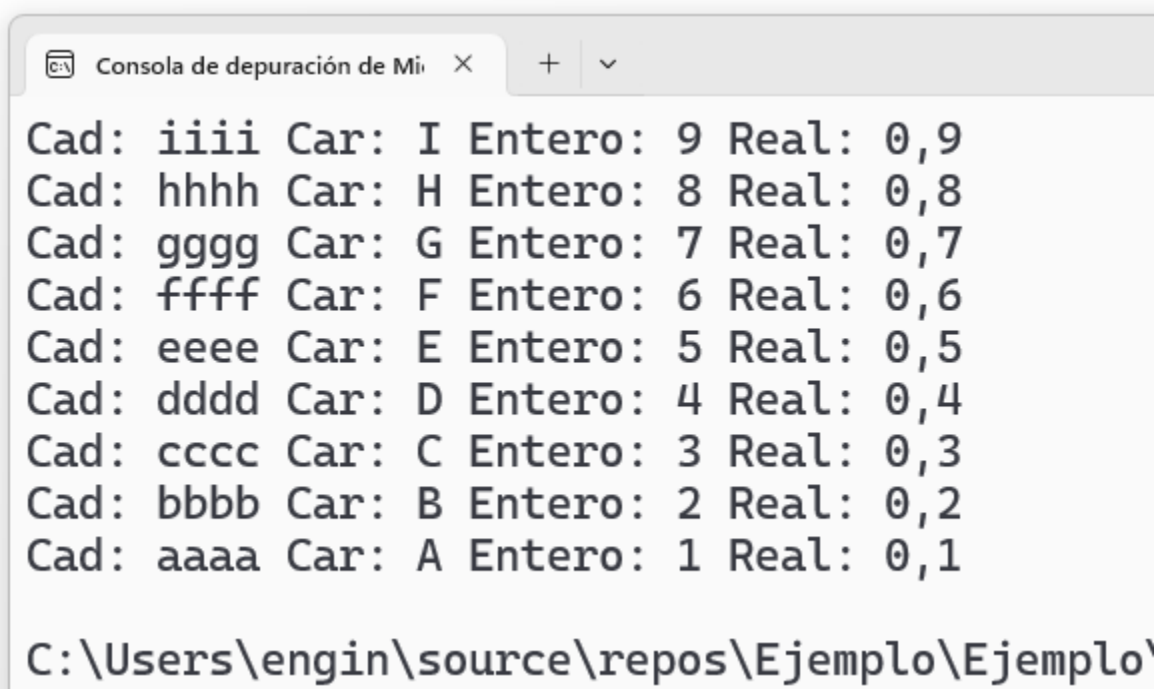

Ilustración 9: Recorriendo lista simplemente enlazada

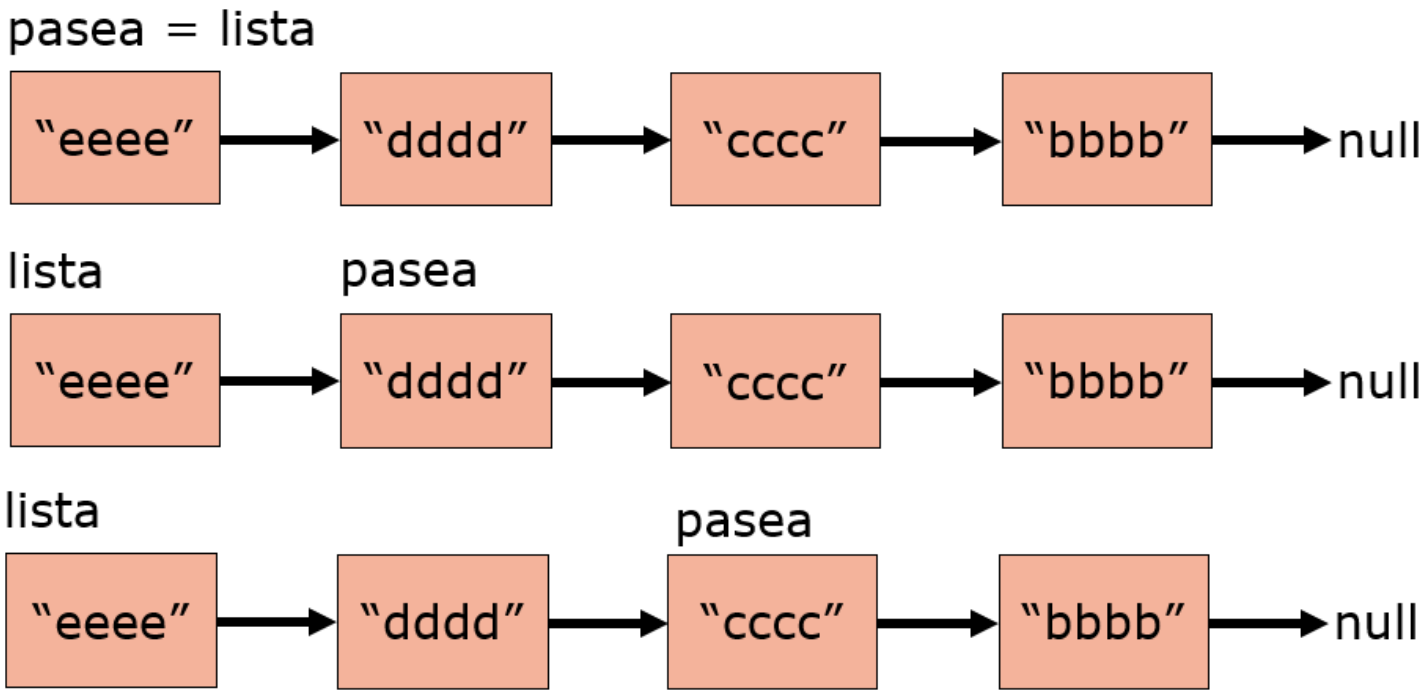

Ilustración 10: La variable "pasea" va de nodo en nodo.

Recorriendo la lista usando un método

Se puede crear un método que recorra la lista enviándole por parámetro el apuntador de la lista. En la función se genera una copia de ese apuntador, así que puede recorrerla dentro de la función sin la preocupación de estar destruyendo la lista.

G/004.cs

```
namespace Ejemplo;

class Nodo {
   //Atributos propios
   public string Cad { get; set; }
   public char Car { get; set; }
   public int Entero { get; set; }
   public double Num { get; set; }

   //Apuntador para lista simplemente enlazada
   public Nodo Apuntador;

   //Constructor
   public Nodo(string Cad, char Car, int Entero,
            double Num, Nodo Apuntador) {
      this.Cad = Cad;
      this.Car = Car;
      this.Entero = Entero;
      this.Num = Num;
      this.Apuntador = Apuntador;
   }

   //Imprime Contenido
   public void Imprime() {
      Console.Write("Cad: " + Cad + " Car: " + Car);
      Console.WriteLine(" Entero: " + Entero + " Real: " + Num);
   }
}

class Program {
   static void Main() {
      //Crea la lista
      Nodo lista = new("aaaa", 'A', 1, 0.1, null);
      lista = new("bbbb", 'B', 2, 0.2, lista);
      lista = new("cccc", 'C', 3, 0.3, lista);
      lista = new("dddd", 'D', 4, 0.4, lista);
      lista = new("eeee", 'E', 5, 0.5, lista);
      lista = new("ffff", 'F', 6, 0.6, lista);
      lista = new("gggg", 'G', 7, 0.7, lista);
```

```
      lista = new("hhhh", 'H', 8, 0.8, lista);
      lista = new("iiii", 'I', 9, 0.9, lista);

      //Pasea la lista, imprimiéndola
      Console.WriteLine("RECORRE PRIMERA VEZ");
      ImprimeLista(lista);

      Console.WriteLine("\r\nRECORRE SEGUNDA VEZ");
      ImprimeLista(lista);
   }

   static public void ImprimeLista(Nodo pasear) {
      while (pasear != null) {
         pasear.Imprime();
         pasear = pasear.Apuntador;
      }
   }
}
```

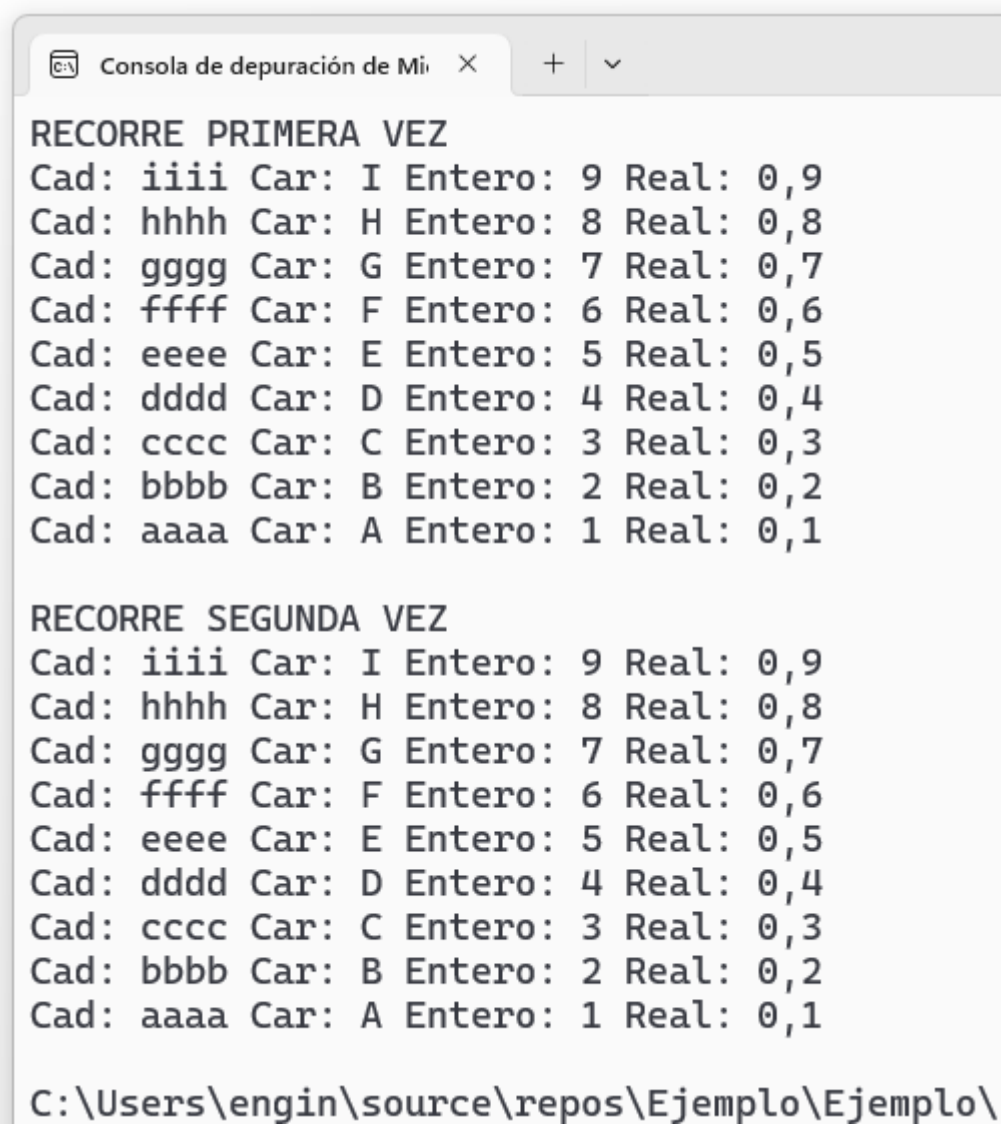

Ilustración 11: Recorriendo la lista usando un método

Tamaño de la lista

Una función puede retornar el tamaño de la lista recorriendo nodo a nodo.

G/005.cs

```
namespace Ejemplo;

class Nodo {
   //Atributos propios
   public string Cad { get; set; }
   public char Car { get; set; }
   public int Entero { get; set; }
   public double Num { get; set; }

   //Apuntador para lista simplemente enlazada
   public Nodo Apuntador;

   //Constructor
   public Nodo(string Cad, char Car, int Entero,
            double Num, Nodo Apuntador) {
      this.Cad = Cad;
      this.Car = Car;
      this.Entero = Entero;
      this.Num = Num;
      this.Apuntador = Apuntador;
   }

   //Imprime Contenido
   public void Imprime() {
      Console.Write("Cad: " + Cad + " Car: " + Car);
      Console.WriteLine(" Entero: " + Entero + " Real: " + Num);
   }
}

class Program {
   static void Main() {
      //Crea la lista
      Nodo lista = new("aaaa", 'A', 1, 0.1, null);
      lista = new("bbbb", 'B', 2, 0.2, lista);
      lista = new("cccc", 'C', 3, 0.3, lista);
      lista = new("dddd", 'D', 4, 0.4, lista);
      lista = new("eeee", 'E', 5, 0.5, lista);
      lista = new("ffff", 'F', 6, 0.6, lista);
      lista = new("gggg", 'G', 7, 0.7, lista);
      lista = new("hhhh", 'H', 8, 0.8, lista);
      lista = new("iiii", 'I', 9, 0.9, lista);
```

```
        //Imprime el tamaño de la lista
        Console.WriteLine("Tamaño de la lista es: " + TamanoLista(lista));
    }

    //Imprime la lista
    static public void ImprimeLista(Nodo pasear) {
        while (pasear != null) {
            pasear.Imprime();
            pasear = pasear.Apuntador;
        }
    }

    //Retorna el tamaño de la lista
    static public int TamanoLista(Nodo pasear) {
        int tamano = 0;
        while (pasear != null) {
            tamano++;
            pasear = pasear.Apuntador;
        }
        return tamano;
    }
}
```

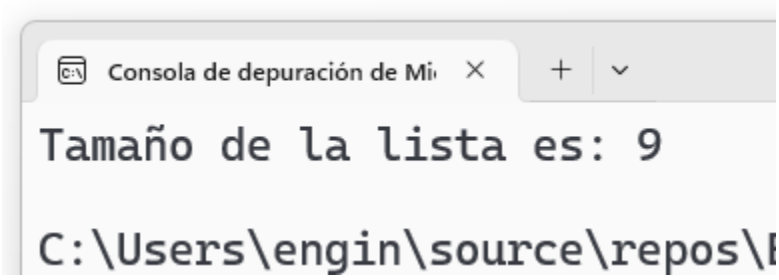

Ilustración 12: Tamaño de la lista

Traer un determinado nodo

G/006.cs

```
namespace Ejemplo;

class Nodo {
   //Atributos propios
   public string Cad { get; set; }
   public char Car { get; set; }
   public int Entero { get; set; }
   public double Num { get; set; }

   //Apuntador para lista simplemente enlazada
   public Nodo Apuntador;

   //Constructor
   public Nodo(string Cad, char Car, int Entero,
            double Num, Nodo Apuntador) {
      this.Cad = Cad;
      this.Car = Car;
      this.Entero = Entero;
      this.Num = Num;
      this.Apuntador = Apuntador;
   }

   //Imprime Contenido
   public void Imprime() {
      Console.Write("Cad: " + Cad + " Car: " + Car);
      Console.WriteLine(" Entero: " + Entero + " Real: " + Num);
   }
}

class Program {
   static void Main() {
      //Crea la lista
      Nodo lista = new("aaaa", 'A', 1, 0.1, null);
      lista = new("bbbb", 'B', 2, 0.2, lista);
      lista = new("cccc", 'C', 3, 0.3, lista);
      lista = new("dddd", 'D', 4, 0.4, lista);
      lista = new("eeee", 'E', 5, 0.5, lista);
      lista = new("ffff", 'F', 6, 0.6, lista);
      lista = new("gggg", 'G', 7, 0.7, lista);
      lista = new("hhhh", 'H', 8, 0.8, lista);
      lista = new("iiii", 'I', 9, 0.9, lista);

      //Trae un determinado nodo
      Nodo particular = TraeNodo(lista, 2);
```

```
      particular.Imprime();
   }

   //Retornar nodo de determinada posición
   static public Nodo TraeNodo(Nodo pasear, int posicion) {
      int ubicacion = 0;
      while (pasear != null) {
         if (ubicacion == posicion) return pasear;
         pasear = pasear.Apuntador;
         ubicacion++;
      }
      return null;
   }
}
```

Ilustración 13: Traer un determinado nodo

Adicionar un nodo en determinada posición

Para adicionar un nodo, se deben hacer varias operaciones:

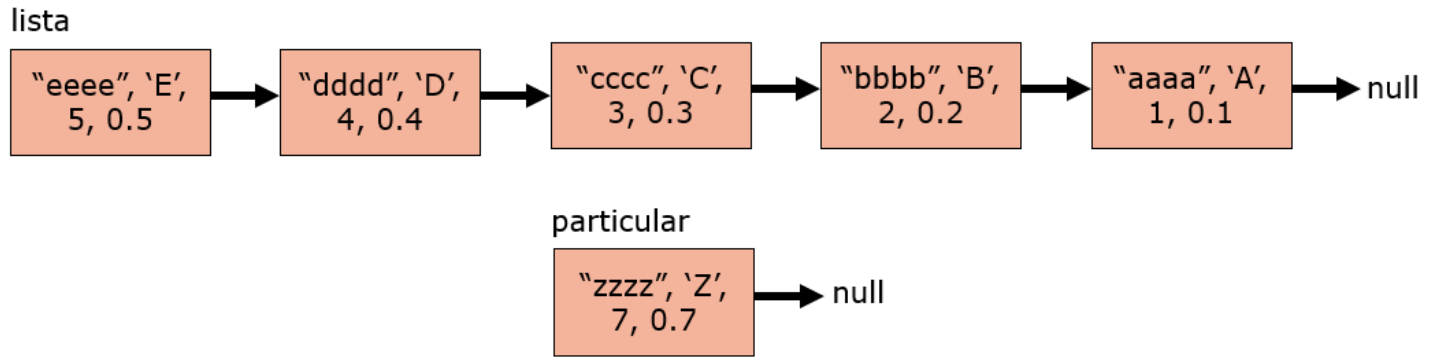

Ilustración 14: Paso 1: Lista existente y nodo nuevo a insertar

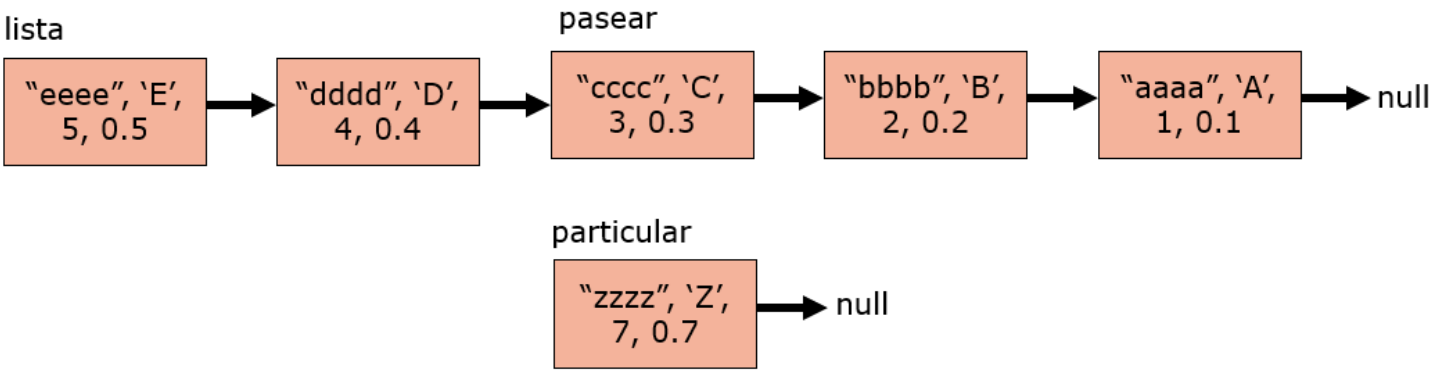

Ilustración 15: Paso 2: Ir hasta el punto de inserción

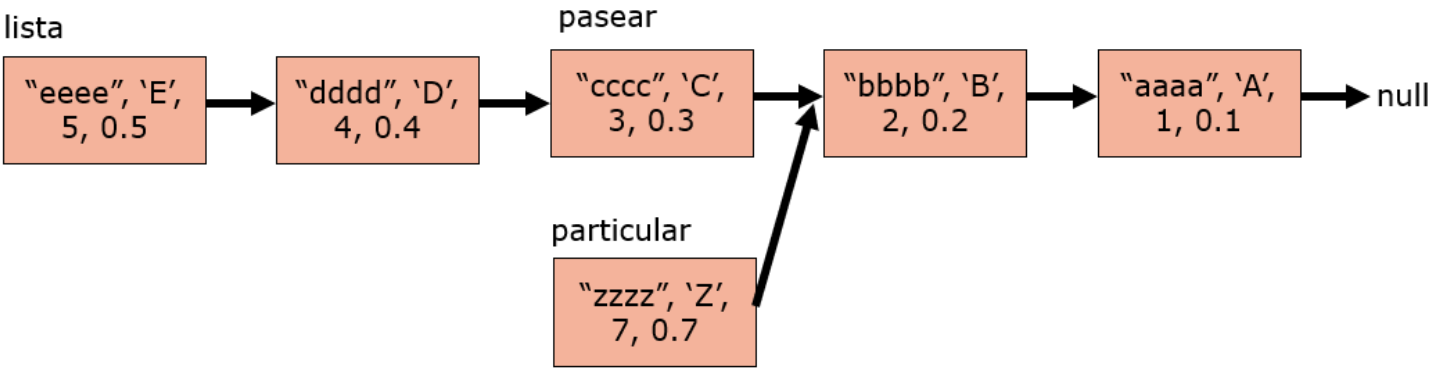

Ilustración 16: Paso 3: El nuevo nodo apunta a la posición particular de la lista

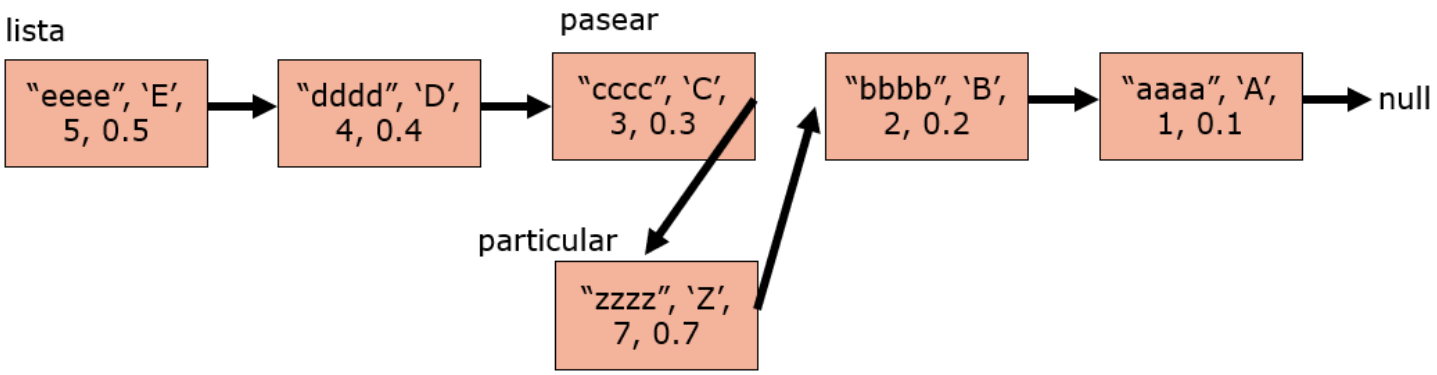

Ilustración 17: Paso 4: El nodo de la lista apunta al nuevo nodo

G/007.cs

```
namespace Ejemplo;

class Nodo {
   //Atributos propios
   public string Cad { get; set; }
   public char Car { get; set; }
   public int Entero { get; set; }
   public double Num { get; set; }

   //Apuntador para lista simplemente enlazada
   public Nodo Apuntador;

   //Constructor
   public Nodo(string Cad, char Car, int Entero,
            double Num, Nodo Apuntador) {
      this.Cad = Cad;
      this.Car = Car;
      this.Entero = Entero;
      this.Num = Num;
      this.Apuntador = Apuntador;
   }

   //Imprime Contenido
   public void Imprime() {
      Console.Write("Cad: " + Cad + " Car: " + Car);
      Console.WriteLine(" Entero: " + Entero + " Real: " + Num);
   }
}

class Program {
   static void Main() {
      //Crea la lista
```

```
      Nodo lista = new("aaaa", 'A', 1, 0.1, null);
      lista = new("bbbb", 'B', 2, 0.2, lista);
      lista = new("cccc", 'C', 3, 0.3, lista);
      lista = new("dddd", 'D', 4, 0.4, lista);
      lista = new("eeee", 'E', 5, 0.5, lista);

      //Añade un nodo en una determinada posición
      Nodo particular = new("zzzz", 'Z', 7, 0.7, null);
      lista = AdicionaNodo(particular, lista, 3);
      ImprimeLista(lista);
   }

   //Adiciona un nodo en determinada posición
   static public Nodo AdicionaNodo(Nodo nodo, Nodo lista, int pos) {
      //Si es al inicio de la lista
      if (pos == 0) {
         nodo.Apuntador = lista;
         return nodo;
      }

      //Si es en una ubicación intermedia
      int ubicacion = 0;
      Nodo pasear = lista;
      while (pasear != null) {
         if (ubicacion + 1 == pos) {
            nodo.Apuntador = pasear.Apuntador;
            pasear.Apuntador = nodo;
            return lista;
         }
         pasear = pasear.Apuntador;
         ubicacion++;
      }

      //Si es al final de la lista
      pasear = lista;
      while (pasear.Apuntador != null)
         pasear = pasear.Apuntador;

      pasear.Apuntador = nodo;
      return lista;
   }

   //Imprime la lista
   static public void ImprimeLista(Nodo pasear) {
      while (pasear != null) {
         pasear.Imprime();
         pasear = pasear.Apuntador;
      }
```

```
    }
}
```

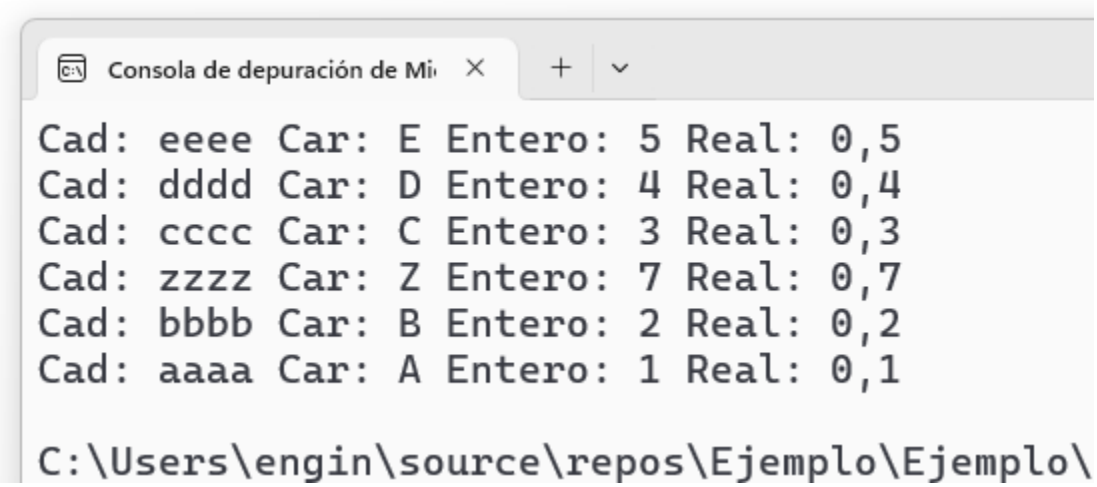

Ilustración 18: Adicionar un nodo en determinada posición

Borrar un nodo de una determinada posición

Para eliminar un nodo, se deben hacer varias operaciones:

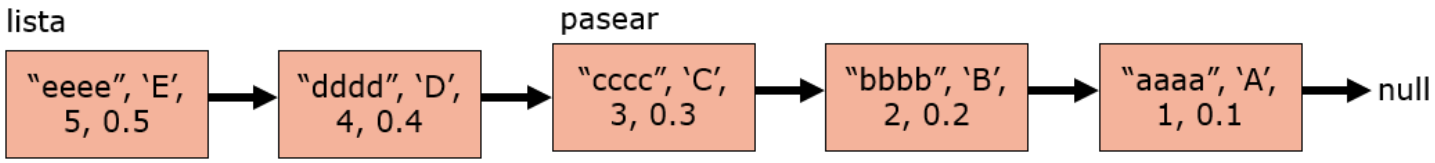

Ilustración 19: Paso 1: Lista existente e ir hasta el nodo al que se desee eliminar

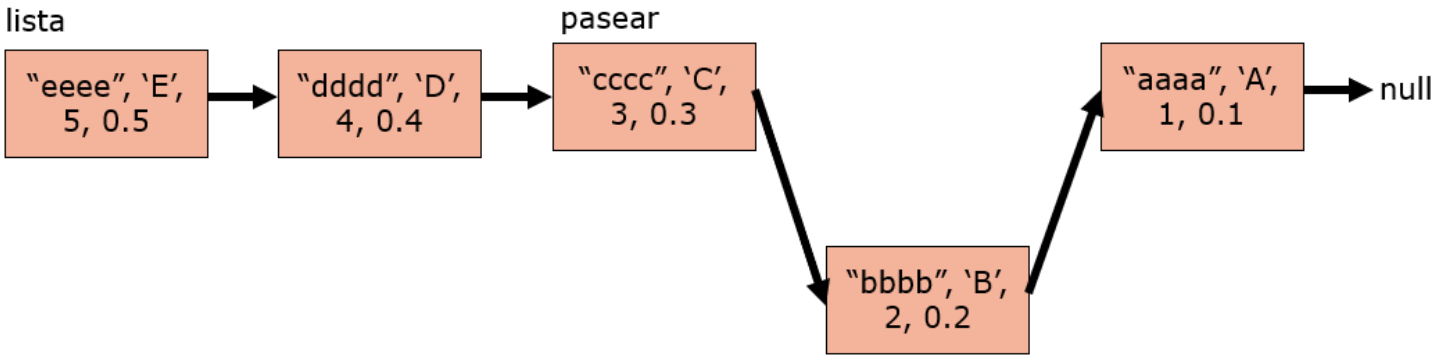

Ilustración 20: Paso 2: El nodo que se va a eliminar

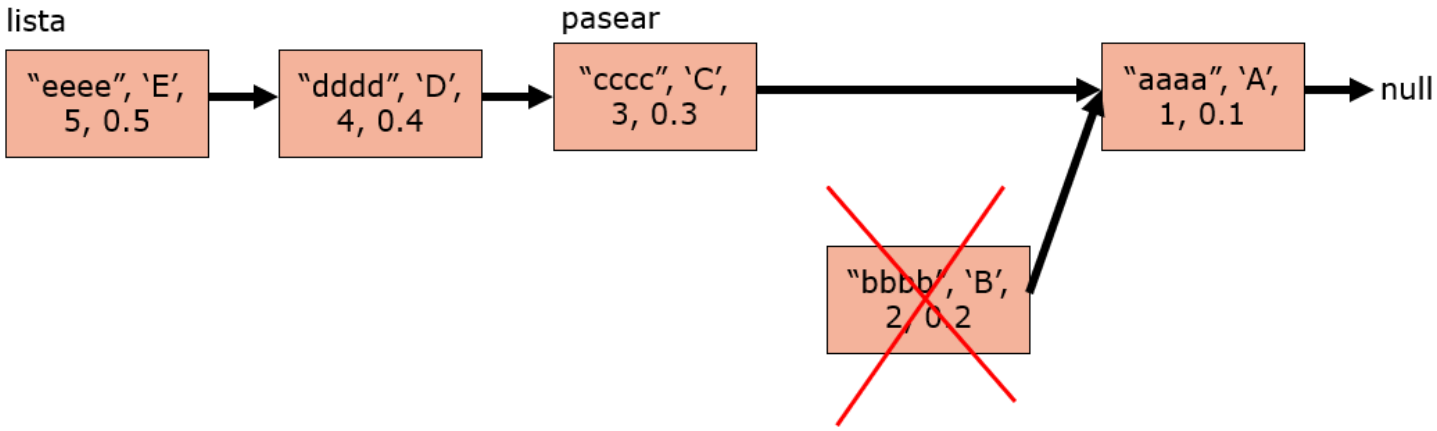

Ilustración 21: Paso 3: Se cambia el apuntador del nodo anterior al siguiente.

```
namespace Ejemplo;

class Nodo {
   //Atributos propios
   public string Cad { get; set; }
   public char Car { get; set; }
   public int Entero { get; set; }
   public double Num { get; set; }

   //Apuntador para lista simplemente enlazada
   public Nodo Apuntador;

   //Constructor
   public Nodo(string Cad, char Car, int Entero,
               double Num, Nodo Apuntador) {
      this.Cad = Cad;
      this.Car = Car;
      this.Entero = Entero;
      this.Num = Num;
      this.Apuntador = Apuntador;
   }

   //Imprime Contenido
   public void Imprime() {
      Console.Write("Cad: " + Cad + " Car: " + Car);
      Console.WriteLine(" Entero: " + Entero + " Real: " + Num);
   }
}

class Program {
   static void Main() {
      //Crea la lista
      Nodo lista = new("aaaa", 'A', 1, 0.1, null);
      lista = new("bbbb", 'B', 2, 0.2, lista);
      lista = new("cccc", 'C', 3, 0.3, lista);
      lista = new("dddd", 'D', 4, 0.4, lista);
      lista = new("eeee", 'E', 5, 0.5, lista);

      //Borra un nodo en una determinada posición
      lista = BorraNodo(lista, 3);
      ImprimeLista(lista);
   }

   //Borra nodo de una determinada posición
   static public Nodo BorraNodo(Nodo lista, int posicion) {
      //Si es al inicio de la lista
      if (posicion == 0) {
```

```
            lista = lista.Apuntador;
            return lista;
        }

        //Si es en una ubicación intermedia
        int ubicacion = 0;
        Nodo pasear = lista;
        while (pasear != null) {
            if (ubicacion + 1 == posicion) {
                pasear.Apuntador = pasear.Apuntador.Apuntador;
                return lista;
            }
            pasear = pasear.Apuntador;
            ubicacion++;
        }

        //Si es al final de la lista
        pasear = lista;
        while (pasear.Apuntador.Apuntador != null)
            pasear = pasear.Apuntador;

        pasear.Apuntador = null;
        return lista;
    }

    //Imprime la lista
    static public void ImprimeLista(Nodo pasear) {
        while (pasear != null) {
            pasear.Imprime();
            pasear = pasear.Apuntador;
        }
    }
}
```

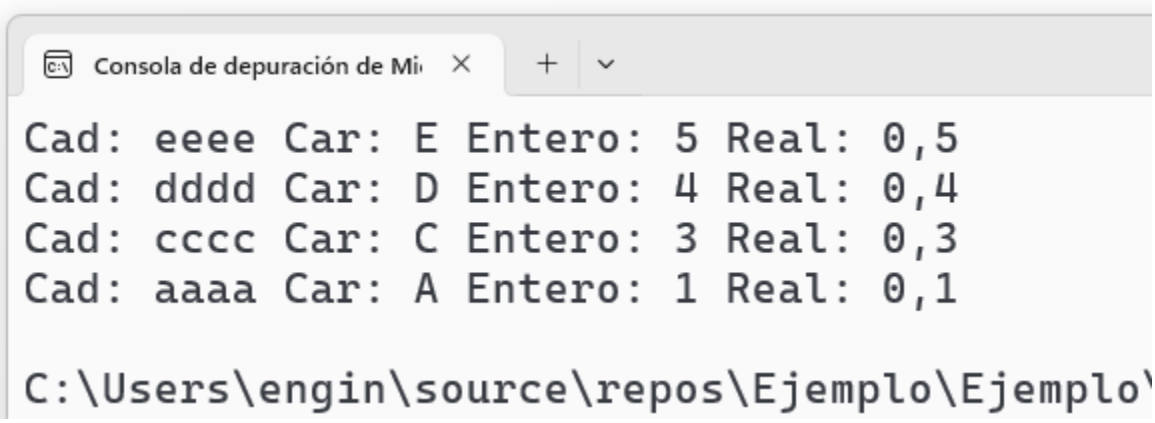

Ilustración 22: Borrar un nodo de una determinada posición

La lista doblemente enlazada

Los nodos tienen doble conexión. Eso permite poder desplazare de izquierda a derecha, o de derecha a izquierda. La variable que sostiene la lista puede estar en cualquier nodo.

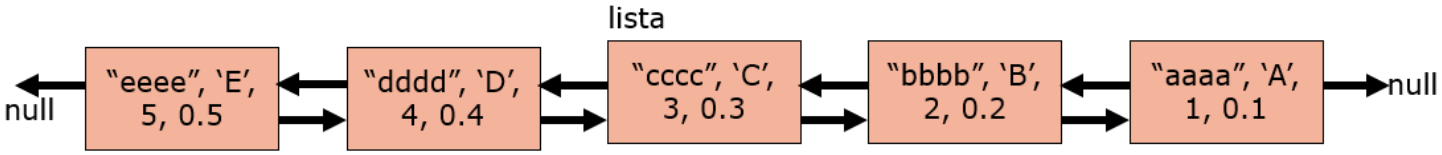

Ilustración 23: Lista doblemente enlazada

G/009.cs

```
namespace Ejemplo;

class Nodo {
   //Atributos propios
   public string Cad { get; set; }
   public char Car { get; set; }
   public int Entero { get; set; }
   public double Num { get; set; }

   //Apuntadores para listas doblemente enlazadas
   public Nodo NodoIzq;
   public Nodo NodoDer;

   //Constructor
   public Nodo(string Cad, char Car, int Entero,
            double Num, Nodo NodoDer) {
      this.Cad = Cad;
      this.Car = Car;
      this.Entero = Entero;
      this.Num = Num;
      NodoIzq = null;
      this.NodoDer = NodoDer;
      if (NodoDer != null) NodoDer.NodoIzq = this;
   }

   //Imprime Contenido
   public void Imprime() {
      Console.Write("Cad: " + Cad + " Car: " + Car);
      Console.WriteLine(" Entero: " + Entero + " Real: " + Num);
   }
}
```

```
class Program {
   static void Main() {
      //Crea la lista
      Nodo lista = new("aaaa", 'A', 1, 0.1, null);
      lista = new("bbbb", 'B', 2, 0.2, lista);
      lista = new("cccc", 'C', 3, 0.3, lista);
      lista = new("dddd", 'D', 4, 0.4, lista);
      lista = new("eeee", 'E', 5, 0.5, lista);

      //Imprime la lista en ambos sentidos
      ImprimeIzquierdaDerecha(lista);
      ImprimeDerechaIzquierda(lista);
   }

   //Imprime la lista de izquierda a derecha
   static public void ImprimeIzquierdaDerecha(Nodo pasear) {
      Console.WriteLine("\r\nDe izquierda a derecha");

      //Debe ponerse en el primer nodo de la izquierda
      while (pasear.NodoIzq != null) {
         pasear = pasear.NodoIzq;
      }

      //Una vez en el primer nodo de la izquierda, entonces va
      //de izquierda a derecha imprimiendo
      while (pasear != null) {
         pasear.Imprime();
         pasear = pasear.NodoDer;
      }
   }

   //Imprime la lista de derecha a izquierda
   static public void ImprimeDerechaIzquierda(Nodo pasear) {
      Console.WriteLine("\r\nDe derecha a izquierda");

      //Debe ponerse en el primer nodo de la derecha
      while (pasear.NodoDer != null) {
         pasear = pasear.NodoDer;
      }

      //Una vez en el primer nodo de la derecha, entonces va
      //de derecha a izquierda imprimiendo
      while (pasear != null) {
         pasear.Imprime();
         pasear = pasear.NodoIzq;
      }
   }
```

```
}
```

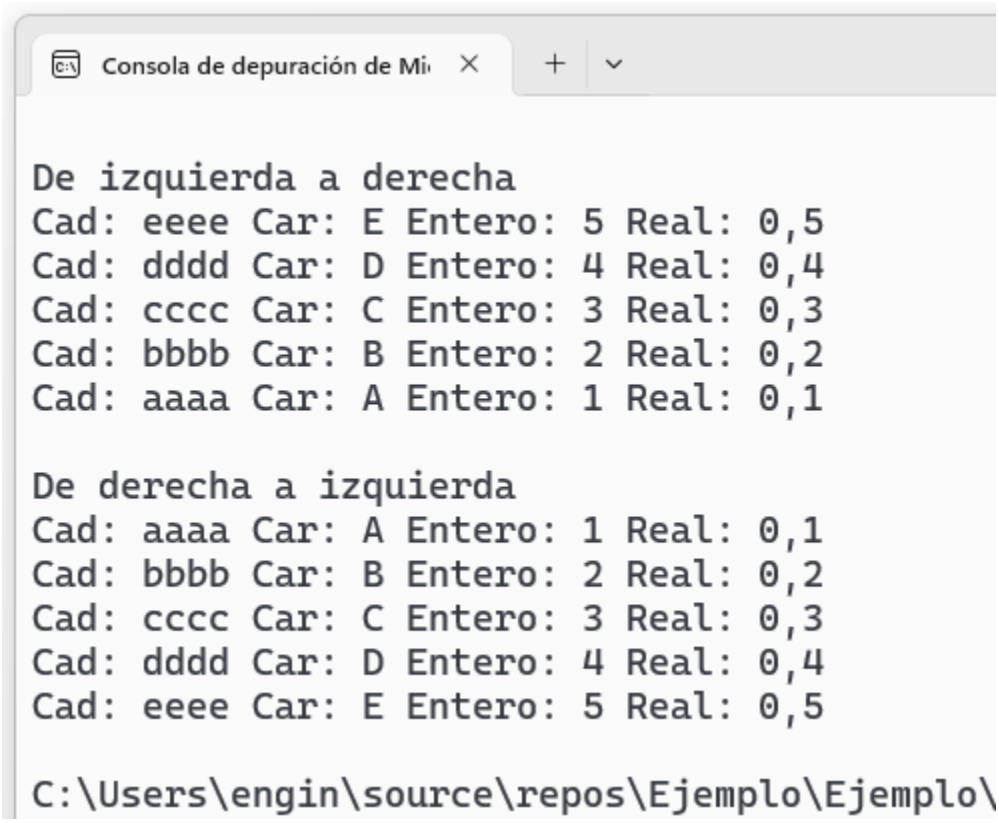

Ilustración 24: La lista doblemente enlazada

Adicionar un nodo en determinada posición

Para adicionar un nodo, se deben hacer varias operaciones:

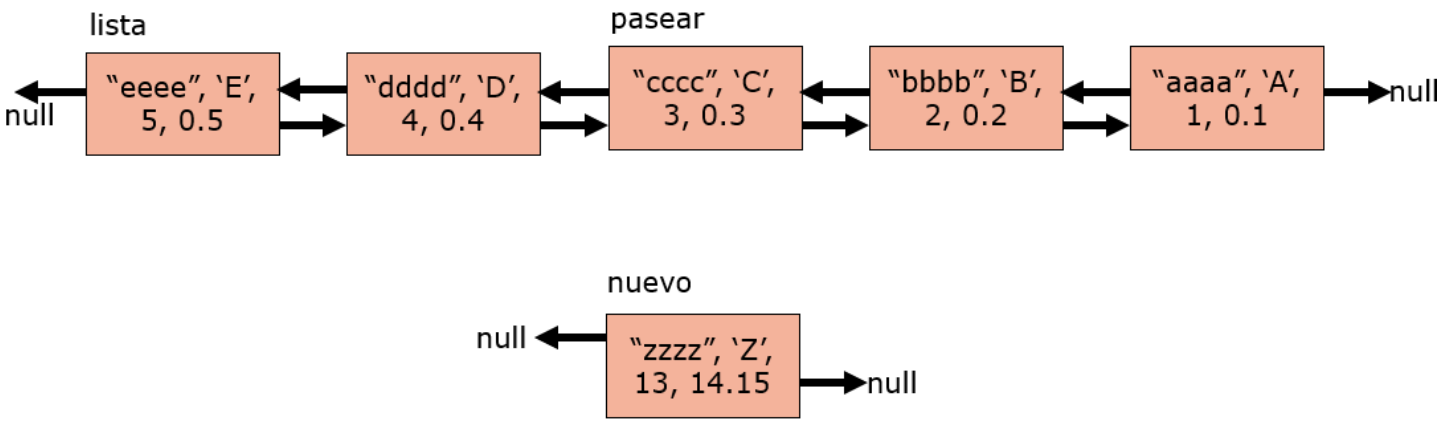

Ilustración 25: Paso 1: Lista existente y nodo nuevo a insertar. Ir al punto de inserción.

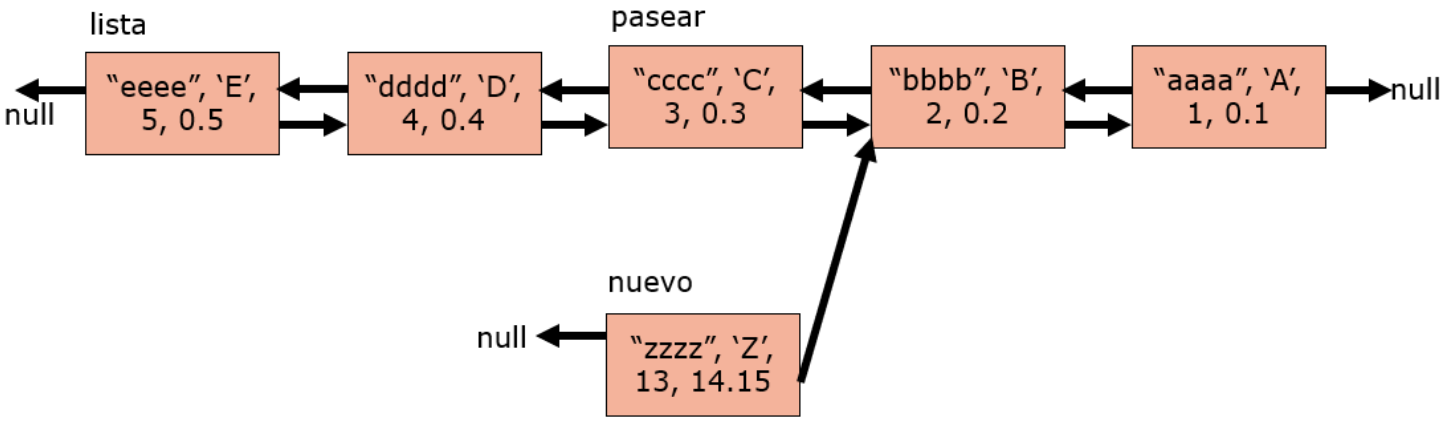

Ilustración 26: Paso 2: Poner los enlaces

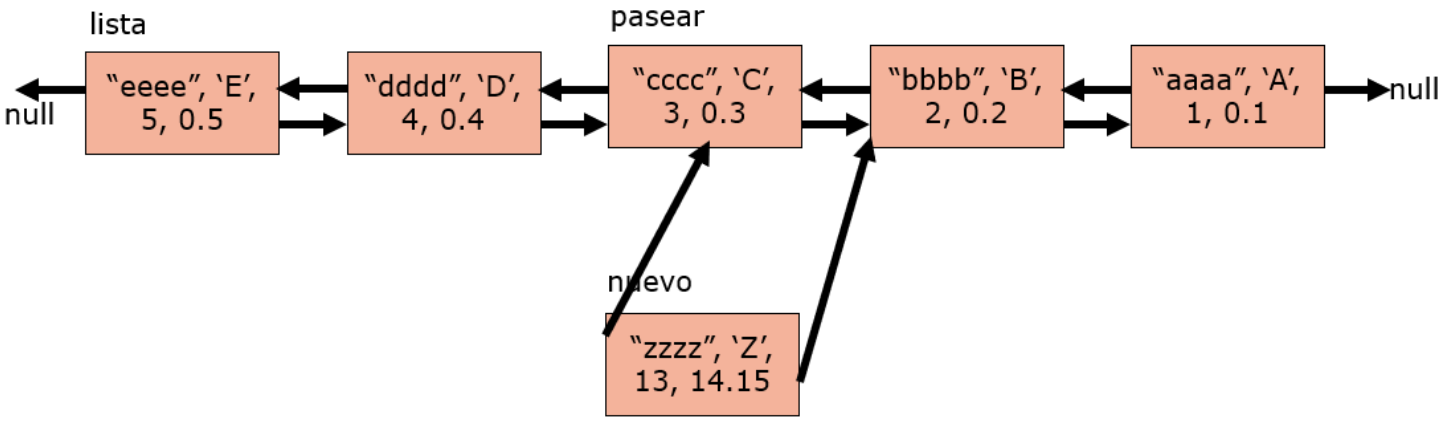

Ilustración 27: Paso 3: Poner los enlaces

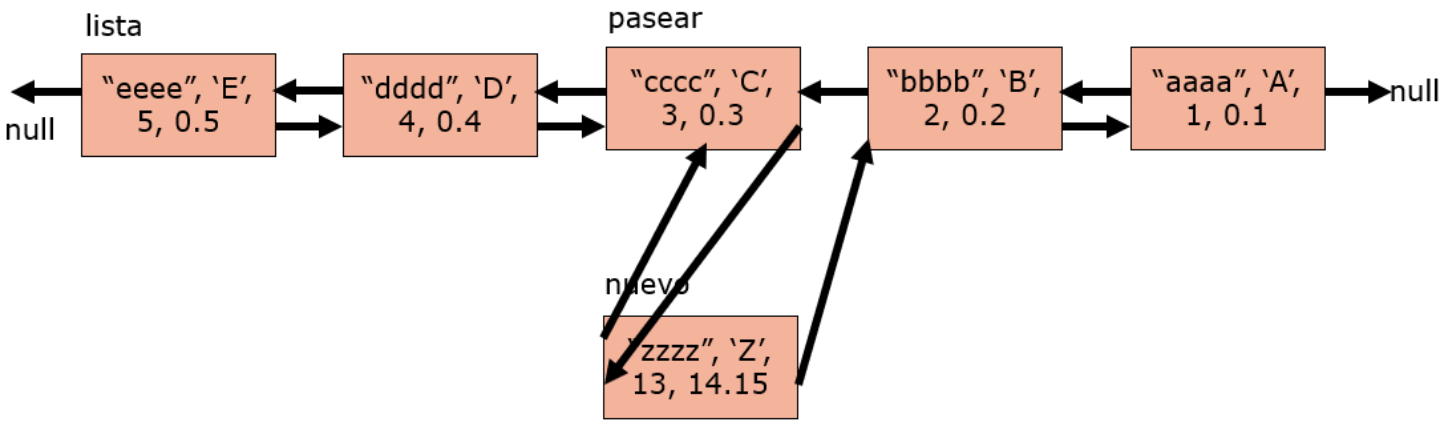

Ilustración 28: Paso 4: Poner los enlaces

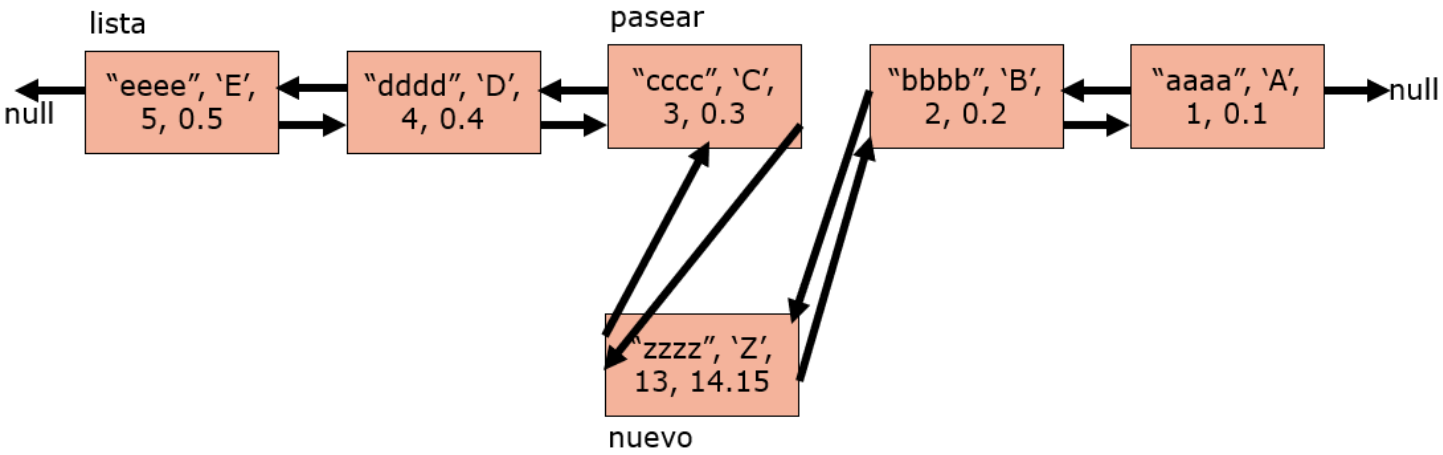

Ilustración 29: Paso 5: Poner los enlaces

G/010.cs

```
namespace Ejemplo;

class Nodo {
   //Atributos propios
   public string Cad { get; set; }
   public char Car { get; set; }
   public int Entero { get; set; }
   public double Num { get; set; }

   //Apuntadores para listas doblemente enlazadas
   public Nodo NodoIzq;
   public Nodo NodoDer;

   //Constructor
   public Nodo(string Cad, char Car, int Entero,
            double Num, Nodo NodoDer) {
```

```
      this.Cad = Cad;
      this.Car = Car;
      this.Entero = Entero;
      this.Num = Num;
      NodoIzq = null;
      this.NodoDer = NodoDer;
      if (NodoDer != null) NodoDer.NodoIzq = this;
   }

   //Imprime Contenido
   public void Imprime() {
      Console.Write("Cad: " + Cad + " Car: " + Car);
      Console.WriteLine(" Entero: " + Entero + " Real: " + Num);
   }
}


class Program {
   static void Main() {
      //Crea la lista
      Nodo lista = new("aaaa", 'A', 1, 0.1, null);
      lista = new("bbbb", 'B', 2, 0.2, lista);
      lista = new("cccc", 'C', 3, 0.3, lista);
      lista = new("dddd", 'D', 4, 0.4, lista);
      lista = new("eeee", 'E', 5, 0.5, lista);

      //Imprime la lista en ambos sentidos
      ImprimeIzquierdaDerecha(lista);
      ImprimeDerechaIzquierda(lista);

      //Agrega un nodo a la lista doblemente enlazada
      Nodo nuevo = new("zzzz", 'Z', 13, 14.15, null);
      lista = AgregaNodo(nuevo, lista, 3);

      //Imprime la lista en ambos sentidos
      ImprimeIzquierdaDerecha(lista);
      ImprimeDerechaIzquierda(lista);
   }

   //Agrega un nodo a la lista doblemente enlazada
   static public Nodo AgregaNodo(Nodo nuevo, Nodo lista, int posicion) {
      //Debe asegurarse de ponerse en el primer nodo de la izquierda
      while (lista.NodoIzq != null) {
         lista = lista.NodoIzq;
      }

      //Si es al inicio de la lista
      if (posicion == 0) {
```

```
            nuevo.NodoDer = lista;
            lista.NodoIzq = nuevo;
            return nuevo;
        }

        //Si es en una ubicación intermedia
        int ubicacion = 0;
        Nodo pasear = lista;
        while (pasear != null) {
            if (ubicacion + 1 == posicion) {
                nuevo.NodoDer = pasear.NodoDer;
                pasear.NodoDer.NodoIzq = nuevo;
                pasear.NodoDer = nuevo;
                nuevo.NodoIzq = pasear;
                return lista;
            }
            pasear = pasear.NodoDer;
            ubicacion++;
        }

        //Si es al final de la lista
        pasear = lista;
        while (pasear.NodoDer != null)
            pasear = pasear.NodoDer;

        pasear.NodoDer = nuevo;
        nuevo.NodoIzq = pasear;
        return lista;
    }


    //Imprime la lista de izquierda a derecha
    static public void ImprimeIzquierdaDerecha(Nodo pasear) {
        Console.WriteLine("\r\nDe izquierda a derecha");

        //Debe ponerse en el primer nodo de la izquierda
        while (pasear.NodoIzq != null) {
            pasear = pasear.NodoIzq;
        }

        //Una vez en el primer nodo de la izquierda, entonces va
        //de izquierda a derecha imprimiendo
        while (pasear != null) {
            pasear.Imprime();
            pasear = pasear.NodoDer;
        }
    }
```

```
   //Imprime la lista de derecha a izquierda
   static public void ImprimeDerechaIzquierda(Nodo pasear) {
      Console.WriteLine("\r\nDe derecha a izquierda");

      //Debe ponerse en el primer nodo de la derecha
      while (pasear.NodoDer != null) {
         pasear = pasear.NodoDer;
      }

      //Una vez en el primer nodo de la derecha, entonces va
      //de derecha a izquierda imprimiendo
      while (pasear != null) {
         pasear.Imprime();
         pasear = pasear.NodoIzq;
      }
   }
}
```

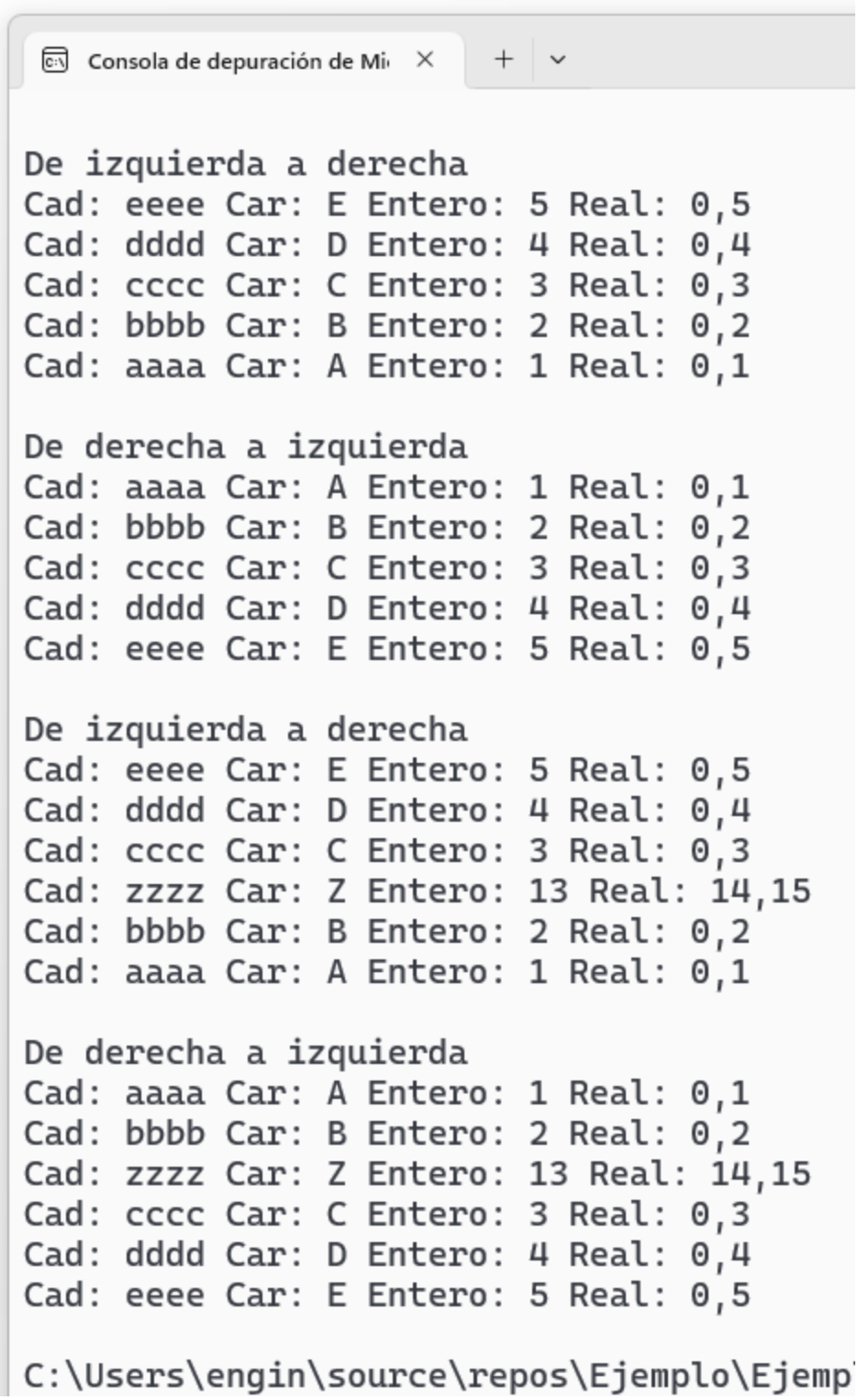

Ilustración 30: Adicionar un nodo en determinada posición

Borrar un nodo de una determinada posición

Para eliminar un nodo, se deben hacer varias operaciones:

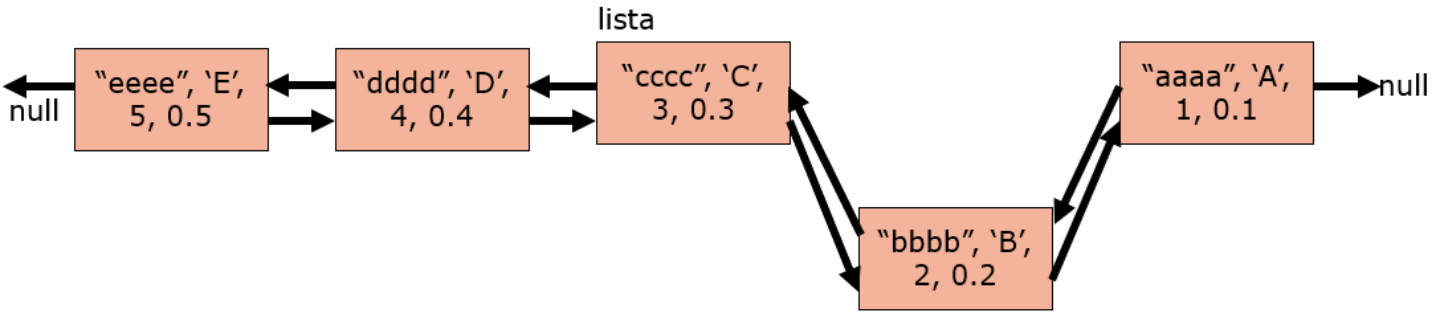

Ilustración 31: Paso 1: Lista existente e ir hasta el nodo al que se desee eliminar

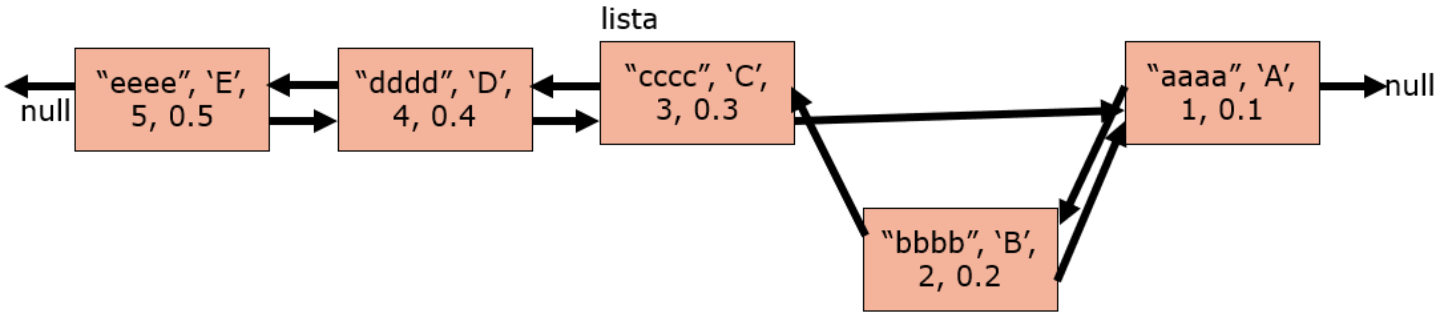

Ilustración 32: Paso 2: Modificar los apuntadores

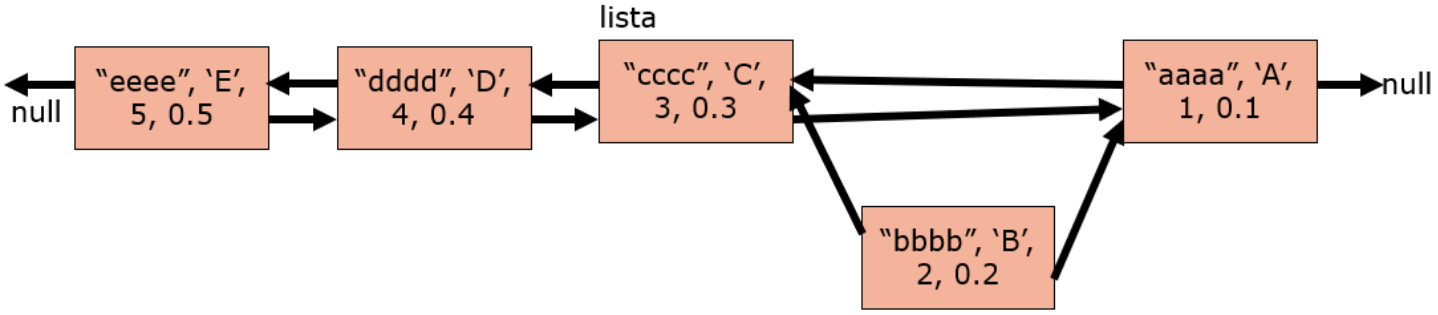

Ilustración 33: Paso 3: Modificar los apuntadores

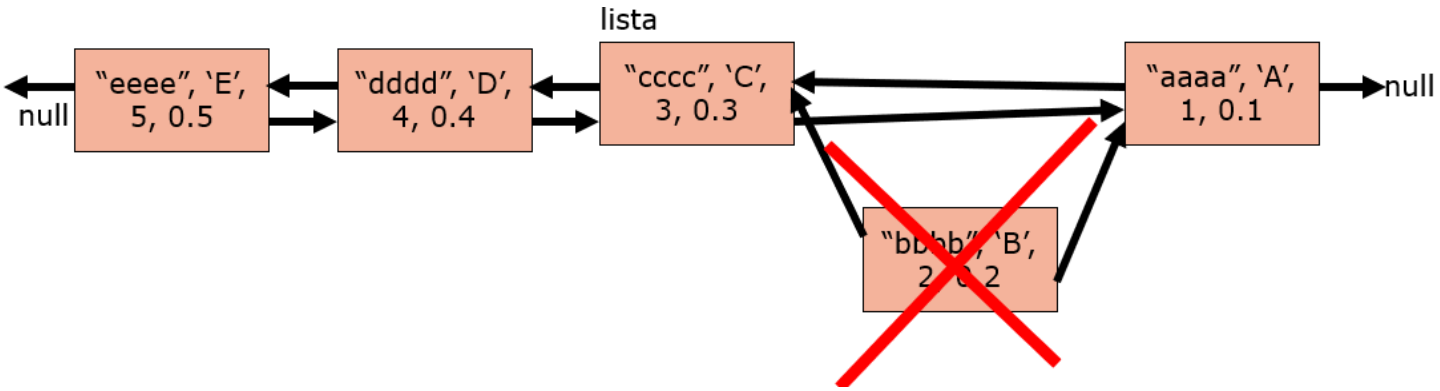

Ilustración 34: Paso 3: El nodo se destruye automáticamente al no tener quien lo sostenga

```
namespace Ejemplo;

class Nodo {
   //Atributos propios
   public string Cad { get; set; }
   public char Car { get; set; }
   public int Entero { get; set; }
   public double Num { get; set; }

   //Apuntadores para listas doblemente enlazadas
   public Nodo NodoIzq;
   public Nodo NodoDer;

   //Constructor
   public Nodo(string Cad, char Car, int Entero,
            double Num, Nodo NodoDer) {
      this.Cad = Cad;
      this.Car = Car;
      this.Entero = Entero;
      this.Num = Num;
      NodoIzq = null;
      this.NodoDer = NodoDer;
      if (NodoDer != null) NodoDer.NodoIzq = this;
   }

   //Imprime Contenido
   public void Imprime() {
      Console.Write("Cad: " + Cad + " Car: " + Car);
      Console.WriteLine(" Entero: " + Entero + " Real: " + Num);
   }
}

class Program {
   static void Main() {
      //Crea la lista
      Nodo lista = new("aaaa", 'A', 1, 0.1, null);
      lista = new("bbbb", 'B', 2, 0.2, lista);
      lista = new("cccc", 'C', 3, 0.3, lista);
      lista = new("dddd", 'D', 4, 0.4, lista);
      lista = new("eeee", 'E', 5, 0.5, lista);

      //Imprime la lista en ambos sentidos
      ImprimeIzquierdaDerecha(lista);
      ImprimeDerechaIzquierda(lista);
```

```
      //Agrega un nodo a la lista doblemente enlazada
      lista = BorraNodo(lista, 3);

      //Imprime la lista en ambos sentidos
      ImprimeIzquierdaDerecha(lista);
      ImprimeDerechaIzquierda(lista);
   }

   //Borra un nodo de la lista doblemente enlazada
   static public Nodo BorraNodo(Nodo lista, int posicion) {
      //Debe asegurarse de ponerse en el
      //primer nodo de la izquierda
      while (lista.NodoIzq != null) {
         lista = lista.NodoIzq;
      }

      //Si es al inicio de la lista
      if (posicion == 0) {
         lista = lista.NodoDer;
         lista.NodoIzq = null;
         return lista;
      }

      //Si es en una ubicación intermedia
      int ubicacion = 0;
      Nodo pasear = lista;
      while (pasear != null) {
         if (ubicacion + 1 == posicion) {
            pasear.NodoDer = pasear.NodoDer.NodoDer;
            if (pasear.NodoDer != null)
               pasear.NodoDer.NodoIzq = pasear;

            return lista;
         }
         pasear = pasear.NodoDer;
         ubicacion++;
      }

      //Si es al final de la lista
      pasear = lista;
      while (pasear.NodoDer.NodoDer != null)
         pasear = pasear.NodoDer;

      pasear.NodoDer = null;
      return lista;
   }

   //Imprime la lista de izquierda a derecha
```

```
   static public void ImprimeIzquierdaDerecha(Nodo pasear) {
      Console.WriteLine("\r\nDe izquierda a derecha");

      //Debe ponerse en el primer nodo de la izquierda
      while (pasear.NodoIzq != null) {
         pasear = pasear.NodoIzq;
      }

      //Una vez en el primer nodo de la izquierda, entonces va
      //de izquierda a derecha imprimiendo
      while (pasear != null) {
         pasear.Imprime();
         pasear = pasear.NodoDer;
      }
   }

   //Imprime la lista de derecha a izquierda
   static public void ImprimeDerechaIzquierda(Nodo pasear) {
      Console.WriteLine("\r\nDe derecha a izquierda");

      //Debe ponerse en el primer nodo de la derecha
      while (pasear.NodoDer != null) {
         pasear = pasear.NodoDer;
      }

      //Una vez en el primer nodo de la derecha, entonces va
      //de derecha a izquierda imprimiendo
      while (pasear != null) {
         pasear.Imprime();
         pasear = pasear.NodoIzq;
      }
   }
}
```

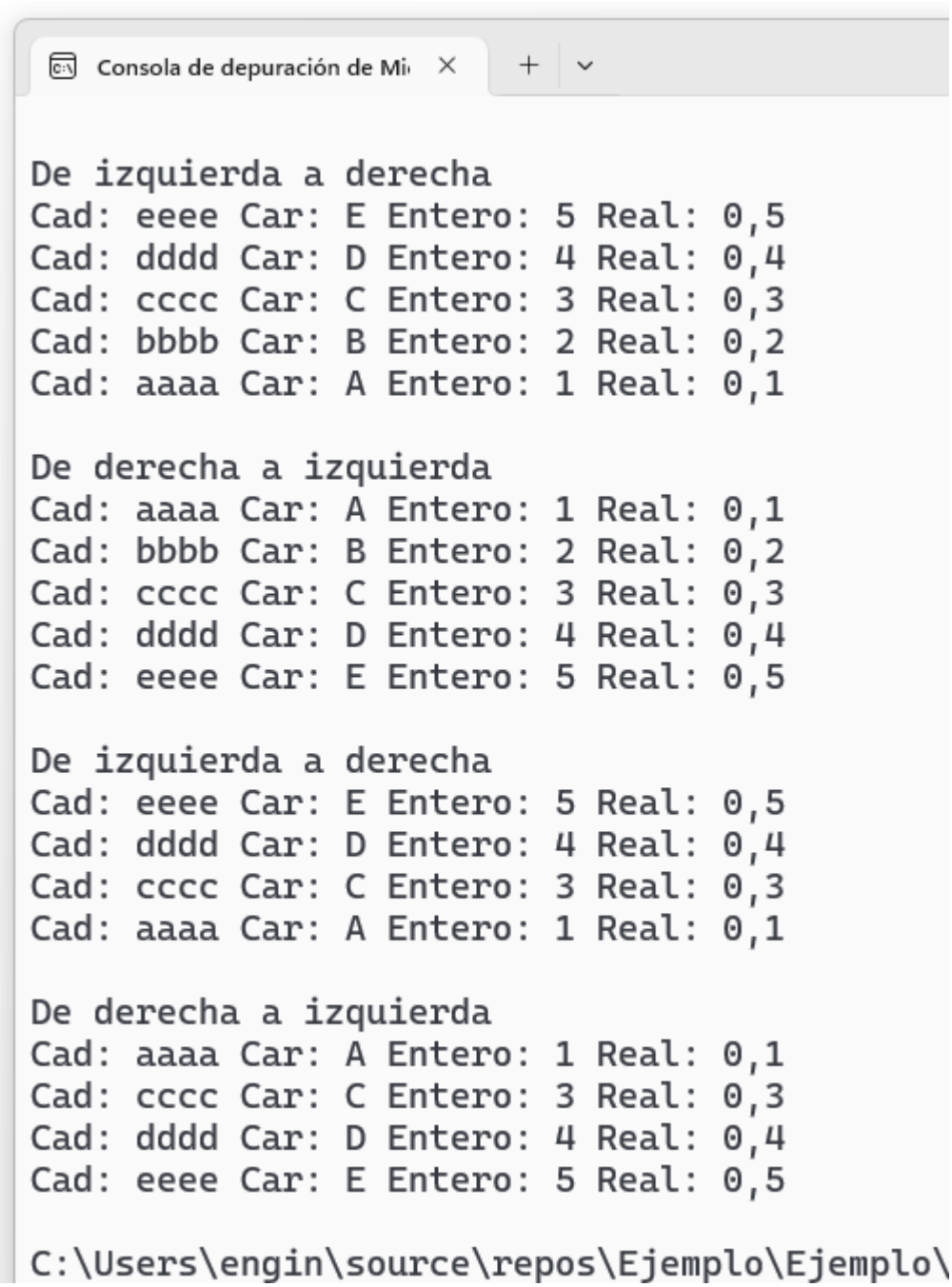

Ilustración 35: Borrar un nodo de una determinada posición

Árbol binario

Primer ejemplo

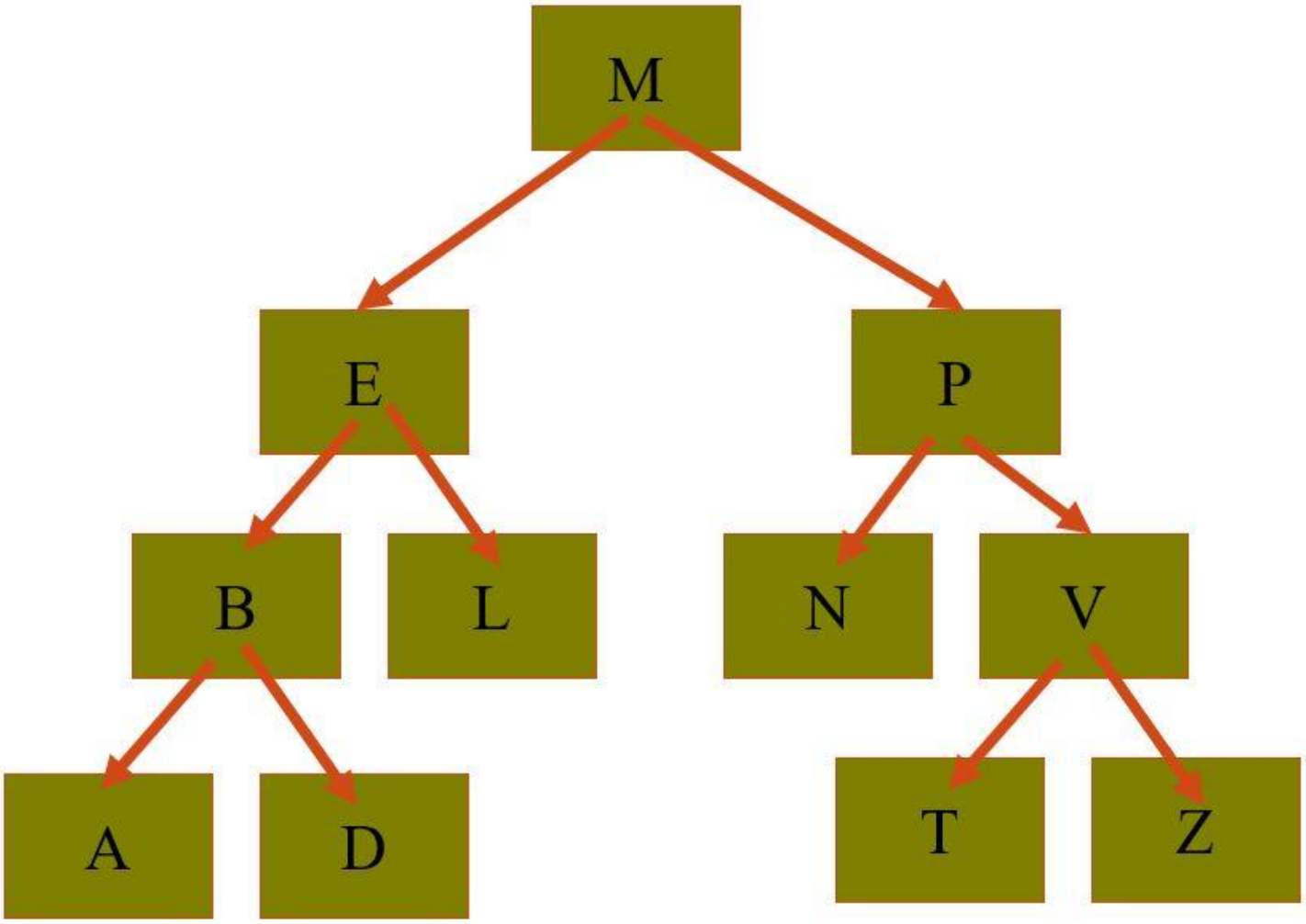

Ilustración 36: Ejemplo de un árbol binario

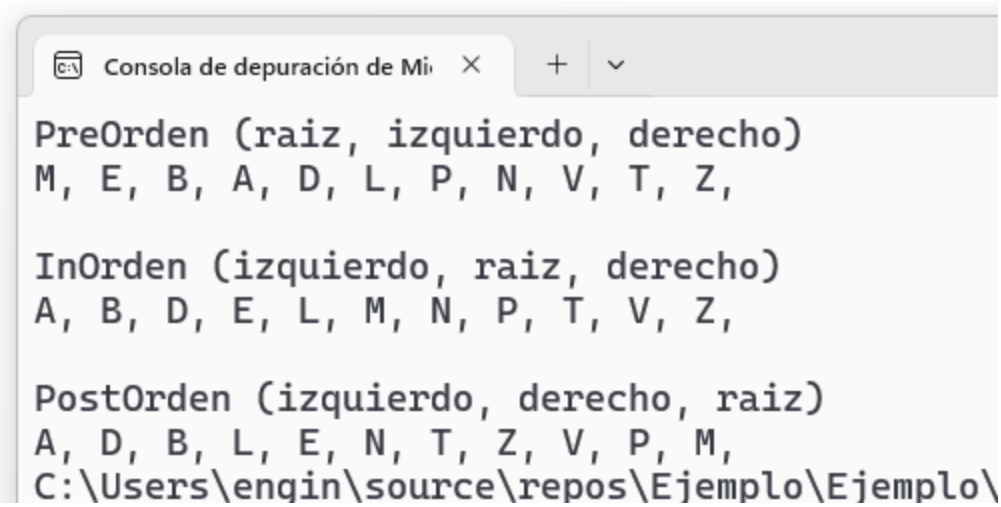

Ilustración 37: Recorrido de un árbol binario

G/012.cs

```
namespace Ejemplo;

//Recorrido de un árbol binario
class Nodo {
   public char Letra { get; set; }
   public Nodo Izquierda; //Apuntador
   public Nodo Derecha; //Apuntador

   //Constructor
   public Nodo(char Letra) {
      this.Letra = Letra;
   }
}

class Program {
   public static void Main() {
      //Crea el árbol
      Nodo Arbol = new('M');
      Arbol.Izquierda = new('E');
      Arbol.Derecha = new('P');
      Arbol.Izquierda.Izquierda = new('B');
      Arbol.Izquierda.Derecha = new('L');
      Arbol.Izquierda.Izquierda.Izquierda = new('A');
      Arbol.Izquierda.Izquierda.Derecha = new('D');
      Arbol.Derecha.Izquierda = new('N');
      Arbol.Derecha.Derecha = new('V');
      Arbol.Derecha.Derecha.Izquierda = new('T');
      Arbol.Derecha.Derecha.Derecha = new('Z');

      //Recorridos
```

```
      Console.WriteLine("PreOrden (raiz, izquierdo, derecho)");
      PreOrden(Arbol);

      Console.WriteLine("\n\nInOrden (izquierdo, raiz, derecho)");
      InOrden(Arbol);

      Console.WriteLine("\n\nPostOrden (izquierdo, derecho, raiz)");
      PostOrden(Arbol);
   }

   static void PreOrden(Nodo Arbol) {
      if (Arbol != null) {
         Console.Write(Arbol.Letra + ", ");
         PreOrden(Arbol.Izquierda);
         PreOrden(Arbol.Derecha);
      }
   }
   static void InOrden(Nodo Arbol) {
      if (Arbol != null) {
         InOrden(Arbol.Izquierda);
         Console.Write(Arbol.Letra + ", ");
         InOrden(Arbol.Derecha);
      }
   }

   static void PostOrden(Nodo Arbol) {
      if (Arbol != null) {
         PostOrden(Arbol.Izquierda);
         PostOrden(Arbol.Derecha);
         Console.Write(Arbol.Letra + ", ");
      }
   }
}
```

Segundo ejemplo

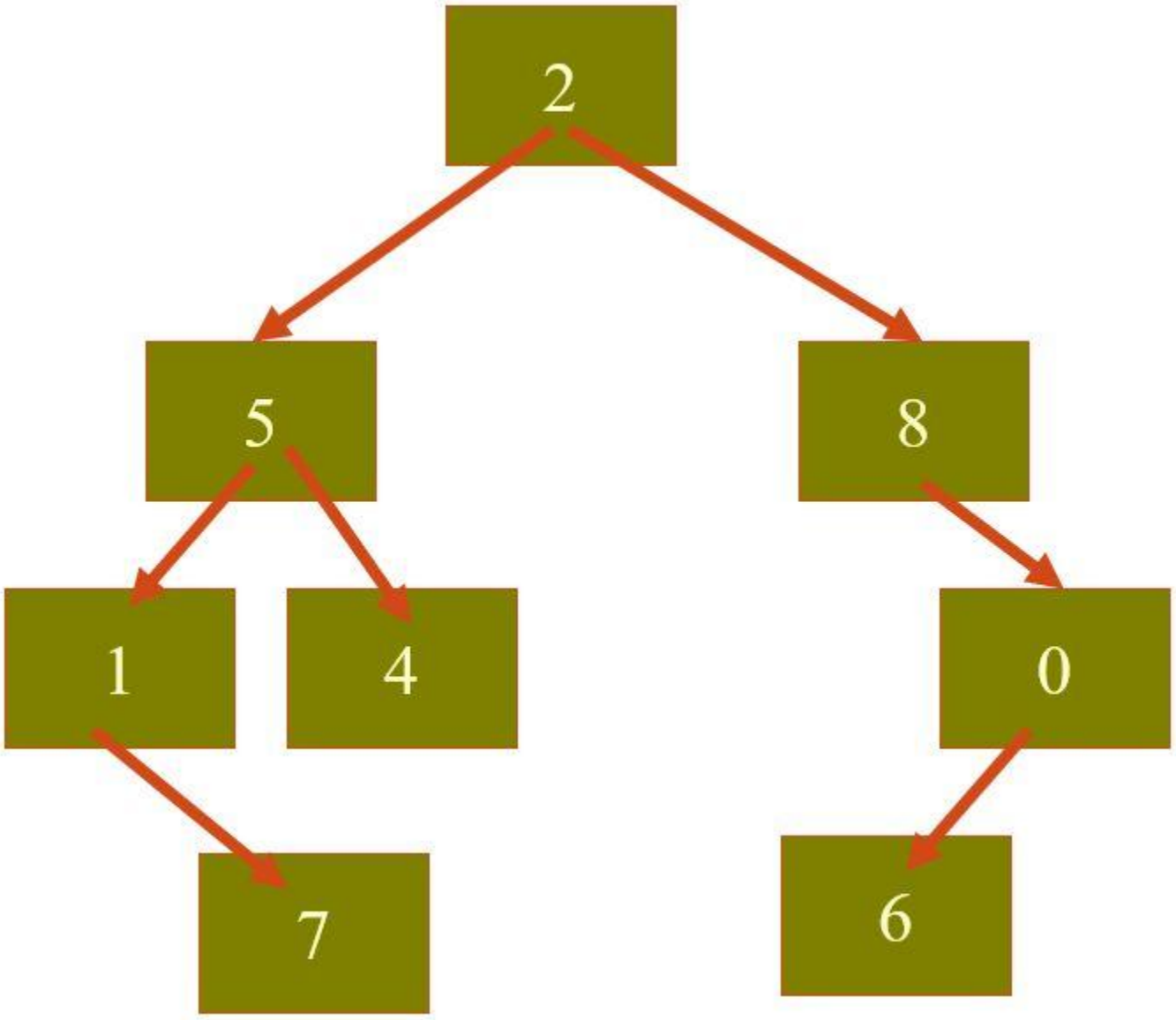

Ilustración 38: Ejemplo de árbol binario

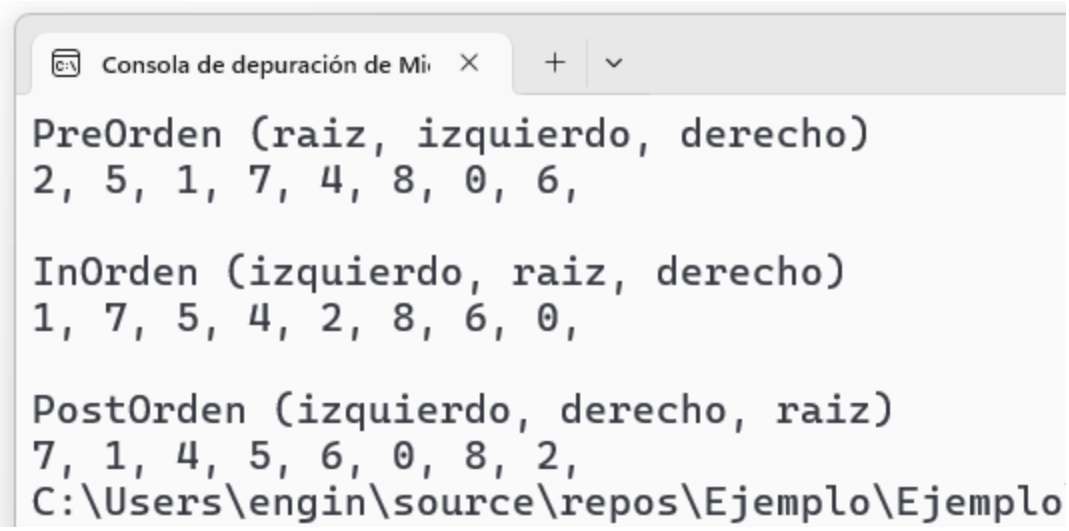

Ilustración 39: Recorrido de un árbol binario

G/013.cs

```
namespace Ejemplo;

//Recorridos de un árbol binario
class Nodo {
   public char Letra { get; set; }
   public Nodo Izquierda; //Apuntador
   public Nodo Derecha; //Apuntador

   //Constructor
   public Nodo(char Letra) {
      this.Letra = Letra;
   }
}

class Program {
   static void Main(string[] args) {
      //Crea el árbol
      Nodo Arbol = new('2');
      Arbol.Izquierda = new('5');
      Arbol.Derecha = new('8');
      Arbol.Izquierda.Izquierda = new('1');
      Arbol.Izquierda.Derecha = new('4');
      Arbol.Izquierda.Izquierda.Derecha = new('7');
      Arbol.Derecha.Derecha = new('0');
      Arbol.Derecha.Derecha.Izquierda = new('6');

      //Recorridos
      Console.WriteLine("PreOrden (raiz, izquierdo, derecho)");
      PreOrden(Arbol);
```

```
      Console.WriteLine("\n\nInOrden (izquierdo, raiz, derecho)");
      InOrden(Arbol);

      Console.WriteLine("\n\nPostOrden (izquierdo, derecho, raiz)");
      PostOrden(Arbol);
   }

   static void PreOrden(Nodo Arbol) {
      if (Arbol != null) {
         Console.Write(Arbol.Letra + ", ");
         PreOrden(Arbol.Izquierda);
         PreOrden(Arbol.Derecha);
      }
   }

   static void InOrden(Nodo Arbol) {
      if (Arbol != null) {
         InOrden(Arbol.Izquierda);
         Console.Write(Arbol.Letra + ", ");
         InOrden(Arbol.Derecha);
      }
   }

   static void PostOrden(Nodo Arbol) {
      if (Arbol != null) {
         PostOrden(Arbol.Izquierda);
         PostOrden(Arbol.Derecha);
         Console.Write(Arbol.Letra + ", ");
      }
   }
}
```

Recorrido iterativo (no recursivo)

El siguiente árbol binario será recorrido en forma iterativa. Se requiere para ello el uso de Pilas

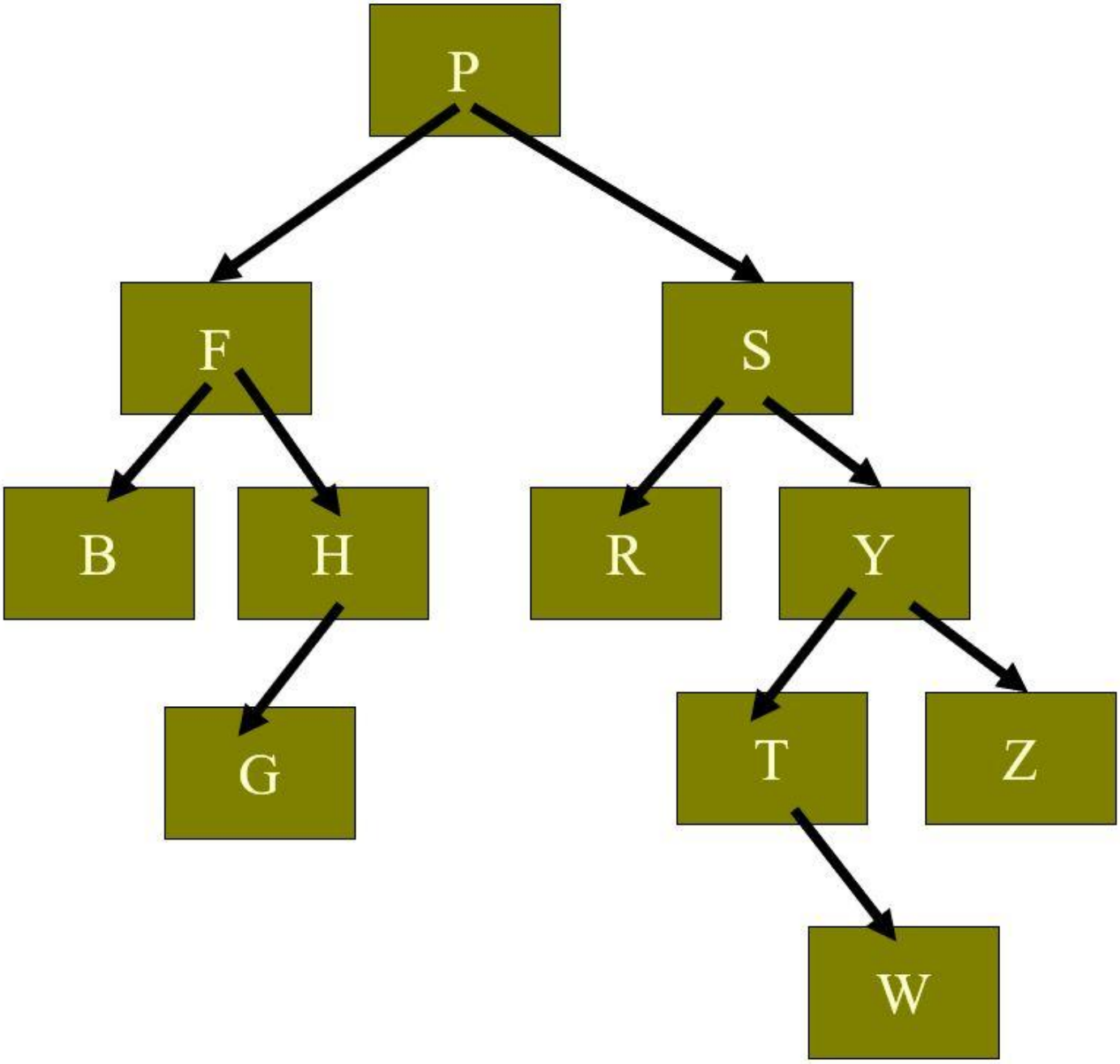

Ilustración 40: Árbol binario

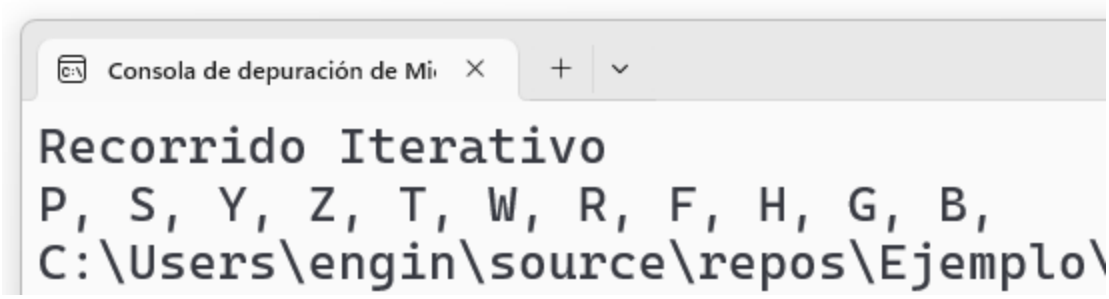

Ilustración 41: Recorrido iterativo (no recursivo)

G/014.cs

```
namespace Ejemplo;

//Recorrido (no recursivo) de un árbol binario
//Nodo de un árbol binario
class Nodo {
   public char Letra { get; set; }
   public Nodo Izquierda; //Apuntador
   public Nodo Derecha; //Apuntador

   //Constructor
   public Nodo(char Letra) {
      this.Letra = Letra;
   }
}

//Nodo de una pila para recorrer iterativamente un árbol binario
class NodoPila {
   public NodoPila Flecha;
   public Nodo Raiz;
   public NodoPila(Nodo Raiz, NodoPila Flecha) {
      this.Raiz = Raiz;
      this.Flecha = Flecha;
   }
}

class Program {
   static void Main(string[] args) {
      //Crea el árbol
      Nodo Arbol = new('P');
      Arbol.Izquierda = new('F');
      Arbol.Derecha = new('S');
      Arbol.Izquierda.Izquierda = new('B');
      Arbol.Izquierda.Derecha = new('H');
      Arbol.Izquierda.Derecha.Izquierda = new('G');
      Arbol.Derecha.Izquierda = new('R');
      Arbol.Derecha.Derecha = new('Y');
      Arbol.Derecha.Derecha.Izquierda = new('T');
```

```
      Arbol.Derecha.Derecha.Derecha = new('Z');
      Arbol.Derecha.Derecha.Izquierda.Derecha = new('W');

      //Recorrido iterativo
      Console.WriteLine("Recorrido Iterativo");
      Iterativo(Arbol);
   }

   public static void Iterativo(Nodo arbol) {
      //Usa una pila para guardar
      NodoPila inicia = new NodoPila(arbol, null);
      do {
         //Una variable tmp para ver el nodo del árbol
         Nodo tmp = inicia.Raiz;

         //Imprime el valor del nodo del árbol
         Console.Write(tmp.Letra + ", ");

         //Se quita un elemento de la pila
         inicia = inicia.Flecha;

         if (tmp.Izquierda != null)
            //Si el nodo de árbol tiene un hijo a la
            //izquierda entonces agrega este a la pila
            inicia = new NodoPila(tmp.Izquierda, inicia);

         if (tmp.Derecha != null)
            //Si el nodo de árbol tiene un hijo a la derecha
            //entonces agrega este a la pila
            inicia = new NodoPila(tmp.Derecha, inicia);

      } while (inicia != null); //Hasta que se vacíe la pila
   }
}
```

Generar árboles binarios al azar

G/015.cs

```
namespace Ejemplo;

//Crear un árbol binario al azar

//Nodo de un árbol binario
class Nodo {
   public int Numero { get; set; }
   public Nodo Izquierda; //Apuntador
   public Nodo Derecha; //Apuntador

   //Constructor
   public Nodo(int Numero) {
      this.Numero = Numero;
   }
}

class Program {
   static void Main(string[] args) {
      Random azar = new Random();
      Nodo Arbol = new(azar.Next(100));

      //Crea el árbol binario
      for (int cont = 1; cont <= 10; cont++)
         AzarNodoArbol(azar, Arbol);

      //Recorridos
      Console.WriteLine("\n\nPreOrden (raiz, izquierdo, derecho)");
      preOrden(Arbol);

      Console.WriteLine("\n\nInOrden (izquierdo, raiz, derecho)");
      inOrden(Arbol);

      Console.WriteLine("\n\nPostOrden (izquierdo, derecho, raiz)");
      postOrden(Arbol);
   }

   //Pone un nodo en una posición al azar
   static void AzarNodoArbol(Random azar, Nodo Raiz) {
      //Por debajo de 0.5 pone una rama a la izquierda
      if (azar.NextDouble() < 0.5) {
         if (Raiz.Izquierda == null)
            Raiz.Izquierda = new(azar.Next(100));
         else
            AzarNodoArbol(azar, Raiz.Izquierda);
```

```
      }
      else {
         if (Raiz.Derecha == null)
            Raiz.Derecha = new(azar.Next(100));
         else
            AzarNodoArbol(azar, Raiz.Derecha);
      }
   }

   static void inOrden(Nodo Arbol) {
      if (Arbol != null) {
         inOrden(Arbol.Izquierda);
         Console.Write(Arbol.Numero + ", ");
         inOrden(Arbol.Derecha);
      }
   }

   static void preOrden(Nodo Arbol) {
      if (Arbol != null) {
         Console.Write(Arbol.Numero + ", ");
         preOrden(Arbol.Izquierda);
         preOrden(Arbol.Derecha);
      }
   }

   static void postOrden(Nodo Arbol) {
      if (Arbol != null) {
         postOrden(Arbol.Izquierda);
         postOrden(Arbol.Derecha);
         Console.Write(Arbol.Numero + ", ");
      }
   }
}
```

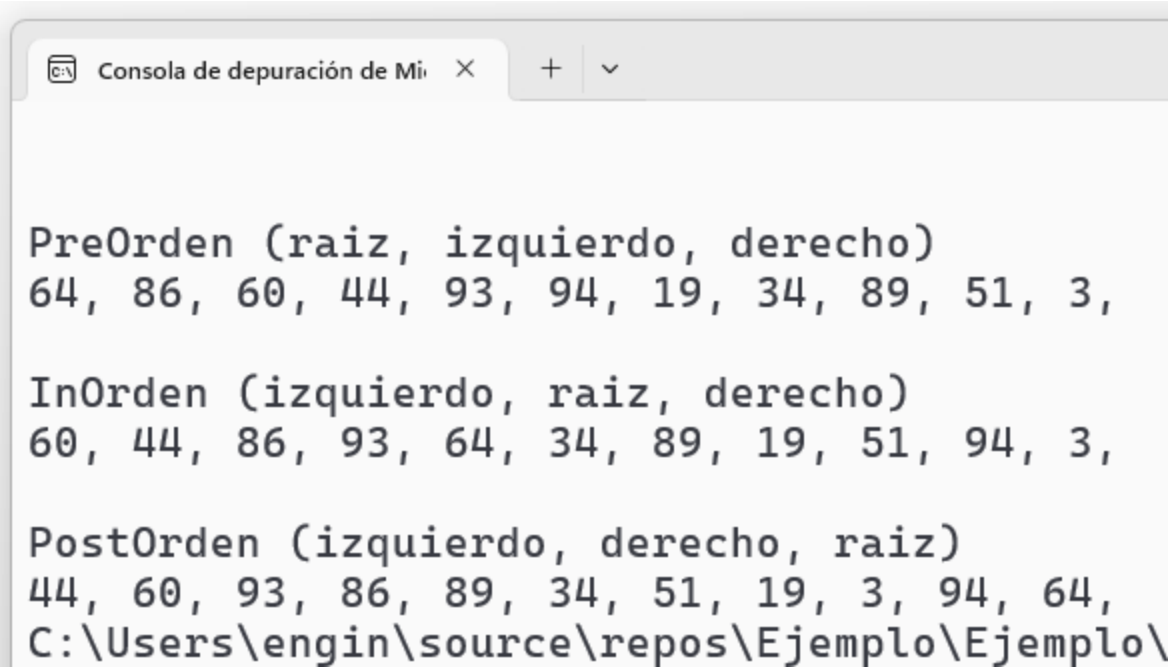

Ilustración 42: Generar árboles binarios al azar

Ordenamiento usando un árbol binario

A medida que se van agregando nodos a un árbol binario, los acomoda de tal forma que al leerlo en InOrden aparece ordenado el contenido.

G/016.cs

```
namespace Ejemplo;

//Ordenar con un árbol binario

//Nodo de un árbol binario
class Nodo {
   public int Numero { get; set; }
   public Nodo Izquierda; //Apuntador
   public Nodo Derecha; //Apuntador

   //Constructor
   public Nodo(int Numero) {
      this.Numero = Numero;
   }
}

class Program {
   static void Main(string[] args) {
      //Va agregando nodo a nodo y los va ordenando
      Nodo Arbol = new(27);
      AgregaNodo(7, Arbol);
      AgregaNodo(17, Arbol);
      AgregaNodo(2, Arbol);
      AgregaNodo(5, Arbol);
      AgregaNodo(19, Arbol);
      AgregaNodo(15, Arbol);
      AgregaNodo(9, Arbol);
      AgregaNodo(10, Arbol);
      AgregaNodo(-1, Arbol);
      AgregaNodo(18, Arbol);
      AgregaNodo(3, Arbol);

      //Al leer en inorden el arbol, los datos salen ordenados
      Console.WriteLine("\n\nInOrden (izquierdo, raiz, derecho)");
      InOrden(Arbol);
   }

   static void AgregaNodo(int Valor, Nodo Raiz) {
      if (Valor <= Raiz.Numero) {
         if (Raiz.Izquierda == null)
```

```
            Raiz.Izquierda = new(Valor);
         else
            AgregaNodo(Valor, Raiz.Izquierda);
      }
      else {
         if (Raiz.Derecha == null)
            Raiz.Derecha = new(Valor);
         else
            AgregaNodo(Valor, Raiz.Derecha);
      }
   }

   static void InOrden(Nodo Arbol) {
      if (Arbol != null) {
         InOrden(Arbol.Izquierda);
         Console.WriteLine(Arbol.Numero + ", ");
         InOrden(Arbol.Derecha);
      }
   }
}
```

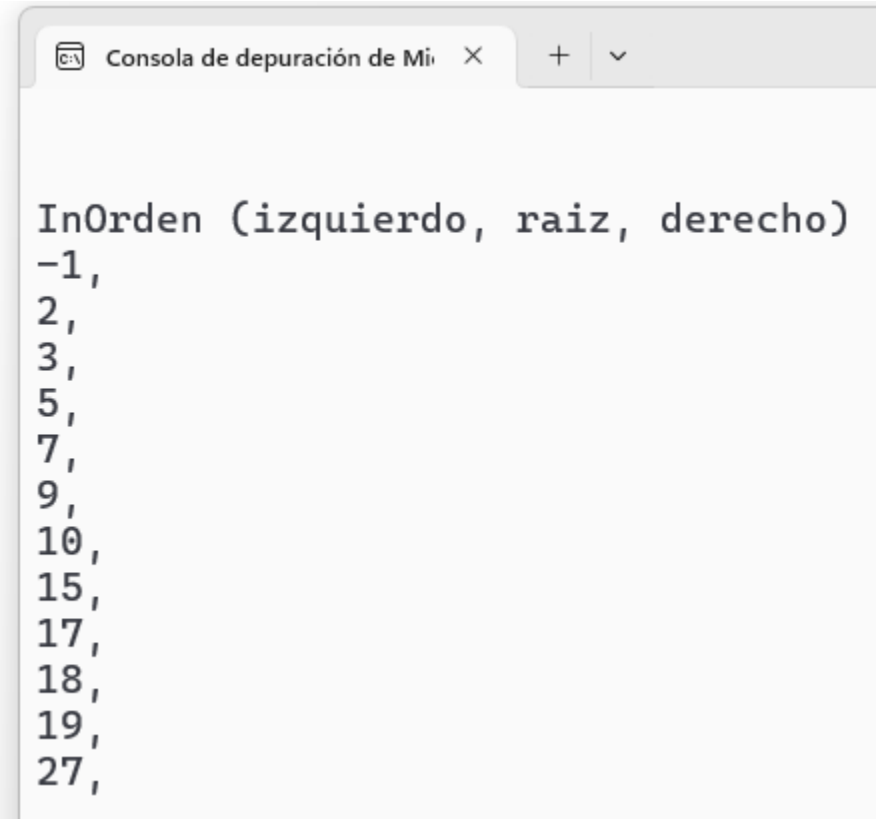

Ilustración 43: Ordenamiento usando un árbol binario

Buscar en árbol binario ordenado, número de nodos y altura del árbol

G/017.cs

```
namespace Ejemplo;

//Ordenar, buscar en árbol binario ordenado,
//número de nodos y altura del árbol
class Nodo {
   public int Numero { get; set; }
   public Nodo Izquierda;
   public Nodo Derecha;

   public Nodo(int Numero) {
      this.Numero = Numero;
      this.Izquierda = null;
      this.Derecha = null;
   }
}

class Program {

   public static void Main() {

      Nodo Arbol = new(27);
      AgregaNodo(7, Arbol);
      AgregaNodo(17, Arbol);
      AgregaNodo(2, Arbol);
      AgregaNodo(5, Arbol);
      AgregaNodo(19, Arbol);
      AgregaNodo(15, Arbol);
      AgregaNodo(9, Arbol);
      AgregaNodo(10, Arbol);
      AgregaNodo(-1, Arbol);
      AgregaNodo(18, Arbol);
      AgregaNodo(3, Arbol);

      //Al leer en inorden el arbol, los datos salen ordenados
      Console.WriteLine("Valores ordenados");
      InOrden(Arbol);

      //Ahora a buscar un determinado valor
      Console.Write("\r\n\r\nBusca el valor: 185...");
      bool encontrar = BuscaArbol(Arbol, 185);
      if (encontrar)
         Console.WriteLine(" Valor encontrado");
      else
         Console.WriteLine(" Valor NO encontrado");
```

```
      //Busca valor en el árbol
      Console.Write("\r\nBusca el valor: 19...");
      Nodo nodoEncuentra = Buscanodo(Arbol, 19);
      if (nodoEncuentra != null)
         Console.WriteLine(" Valor encontrado");
      else
         Console.WriteLine(" Valor no encontrado");

      //Contar los nodos
      Console.WriteLine("\r\nTotal nodos: " + CuentaNodosArbol(Arbol));

      //Altura del árbol
      Console.WriteLine("\nAltura del árbol: " + AlturaArbol(Arbol));
   }

   public static void AgregaNodo(int Valor, Nodo Raiz) {
      if (Valor <= Raiz.Numero) {
         if (Raiz.Izquierda == null)
            Raiz.Izquierda = new(Valor);
         else
            AgregaNodo(Valor, Raiz.Izquierda);
      }
      else {
         if (Raiz.Derecha == null)
            Raiz.Derecha = new(Valor);
         else
            AgregaNodo(Valor, Raiz.Derecha);
      }
   }

   //Recorrido dél árbol en InOrden
   static void InOrden(Nodo Arbol) {
      if (Arbol != null) {
         InOrden(Arbol.Izquierda);
         Console.Write(Arbol.Numero + ", ");
         InOrden(Arbol.Derecha);
      }
   }

   //Retorna true si encuentra el valor en el árbol binario
   static bool BuscaArbol(Nodo Arbol, int valor) {
      if (Arbol != null) {
         if (Arbol.Numero == valor) return true;
         bool encuentraI = BuscaArbol(Arbol.Izquierda, valor);
         bool encuentraD = BuscaArbol(Arbol.Derecha, valor);
         if (encuentraI || encuentraD) return true;
      }
```

```
      return false;
   }

   //Retorna el Nodo donde se encuentra el valor buscado
   static Nodo Buscanodo(Nodo Raiz, int valor) {
      if (valor == Raiz.Numero)
         return Raiz;

      if (valor < Raiz.Numero && Raiz.Izquierda != null)
         return Buscanodo(Raiz.Izquierda, valor);

      if (valor > Raiz.Numero && Raiz.Derecha != null)
         return Buscanodo(Raiz.Derecha, valor);

      return null;
   }

   //Cuenta los nodos de un árbol
   static int CuentaNodosArbol(Nodo Arbol) {
      if (Arbol == null) return 0;
      int contarI = CuentaNodosArbol(Arbol.Izquierda);
      int contarD = CuentaNodosArbol(Arbol.Derecha);
      return contarI + contarD + 1;
   }

   //Calcula la altura de un árbol
   static int AlturaArbol(Nodo Arbol) {
      if (Arbol == null) return 0;
      int alturaI = AlturaArbol(Arbol.Izquierda);
      int alturaD = AlturaArbol(Arbol.Derecha);
      if (alturaI > alturaD) return alturaI + 1;
      return alturaD + 1;
   }
}
```

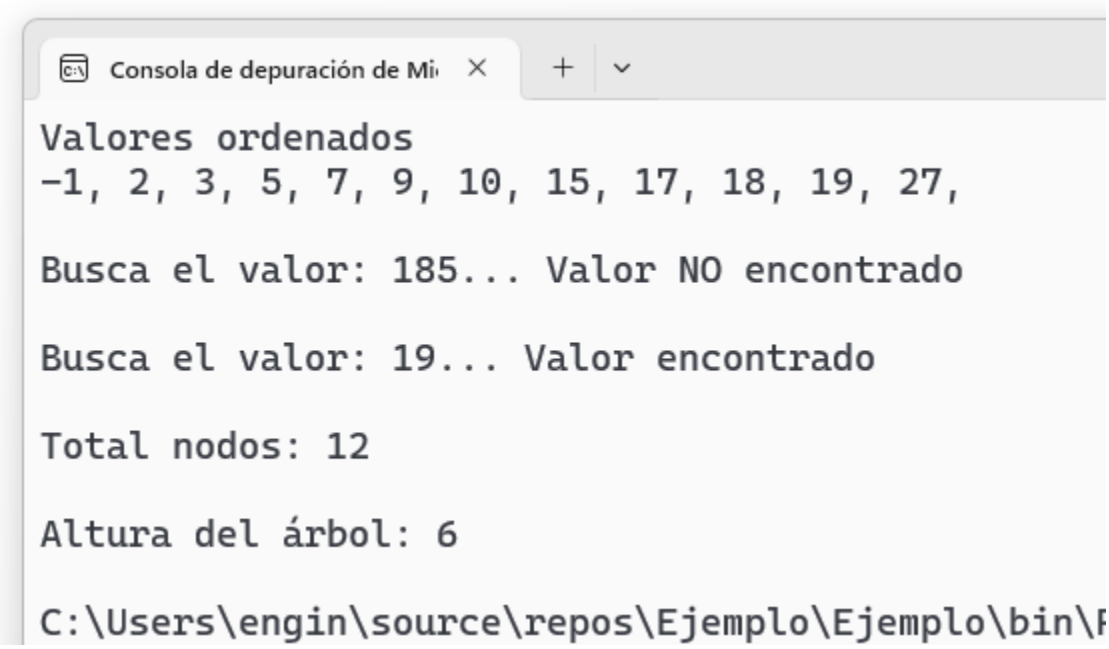

Ilustración 44: Buscar en árbol binario ordenado, número de nodos y altura del árbol

Dibujar un árbol binario

En http://viz-js.com/ se encuentra este servicio:

G/018.cs

```
namespace Ejemplo;

//Dibujar el árbol en http://viz-js.com/
class Nodo {
   public char Letra { get; set; }
   public Nodo Izquierda;
   public Nodo Derecha;

   public Nodo(char Letra) {
      this.Letra = Letra;
      this.Izquierda = null;
      this.Derecha = null;
   }
}

class Program {

   public static void Main() {

      //Crea el árbol
      Nodo Arbol = new('P');
      Arbol.Izquierda = new('F');
      Arbol.Derecha = new('S');
      Arbol.Izquierda.Izquierda = new('B');
      Arbol.Izquierda.Derecha = new('H');
      Arbol.Izquierda.Derecha.Izquierda = new('G');
      Arbol.Derecha.Izquierda = new('R');
      Arbol.Derecha.Derecha = new('Y');
      Arbol.Derecha.Derecha.Izquierda = new('T');
      Arbol.Derecha.Derecha.Derecha = new('Z');
      Arbol.Derecha.Derecha.Izquierda.Derecha = new('W');

      //Probarlo en: http://viz-js.com
      Console.WriteLine("digraph testgraph{");
      Dibujar(Arbol);
      Console.WriteLine("}");
   }

   static void Dibujar(Nodo Arbol) {
      if (Arbol != null) {
         if (Arbol.Izquierda != null) {
            Console.Write(Arbol.Letra + "->");
```

```
            Console.WriteLine(Arbol.Izquierda.Letra);
            Dibujar(Arbol.Izquierda);
         }
         if (Arbol.Derecha != null) {
            Console.Write(Arbol.Letra + "->");
            Console.WriteLine(Arbol.Derecha.Letra);
            Dibujar(Arbol.Derecha);
         }
      }
   }
}
```

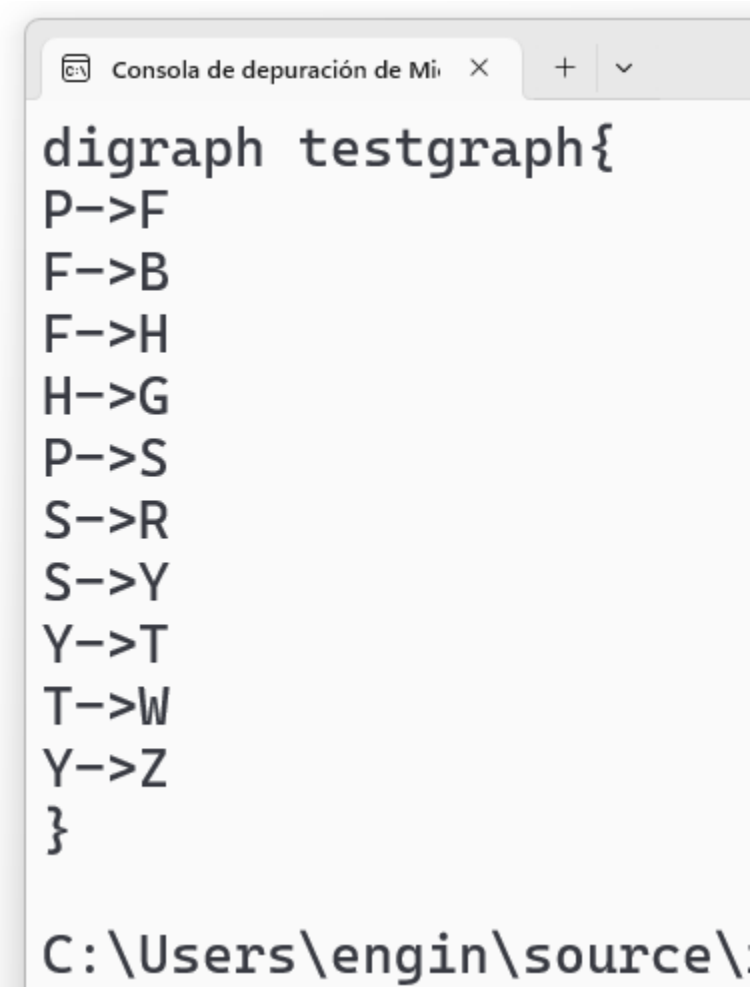

Ilustración 45: Dibujar un árbol binario

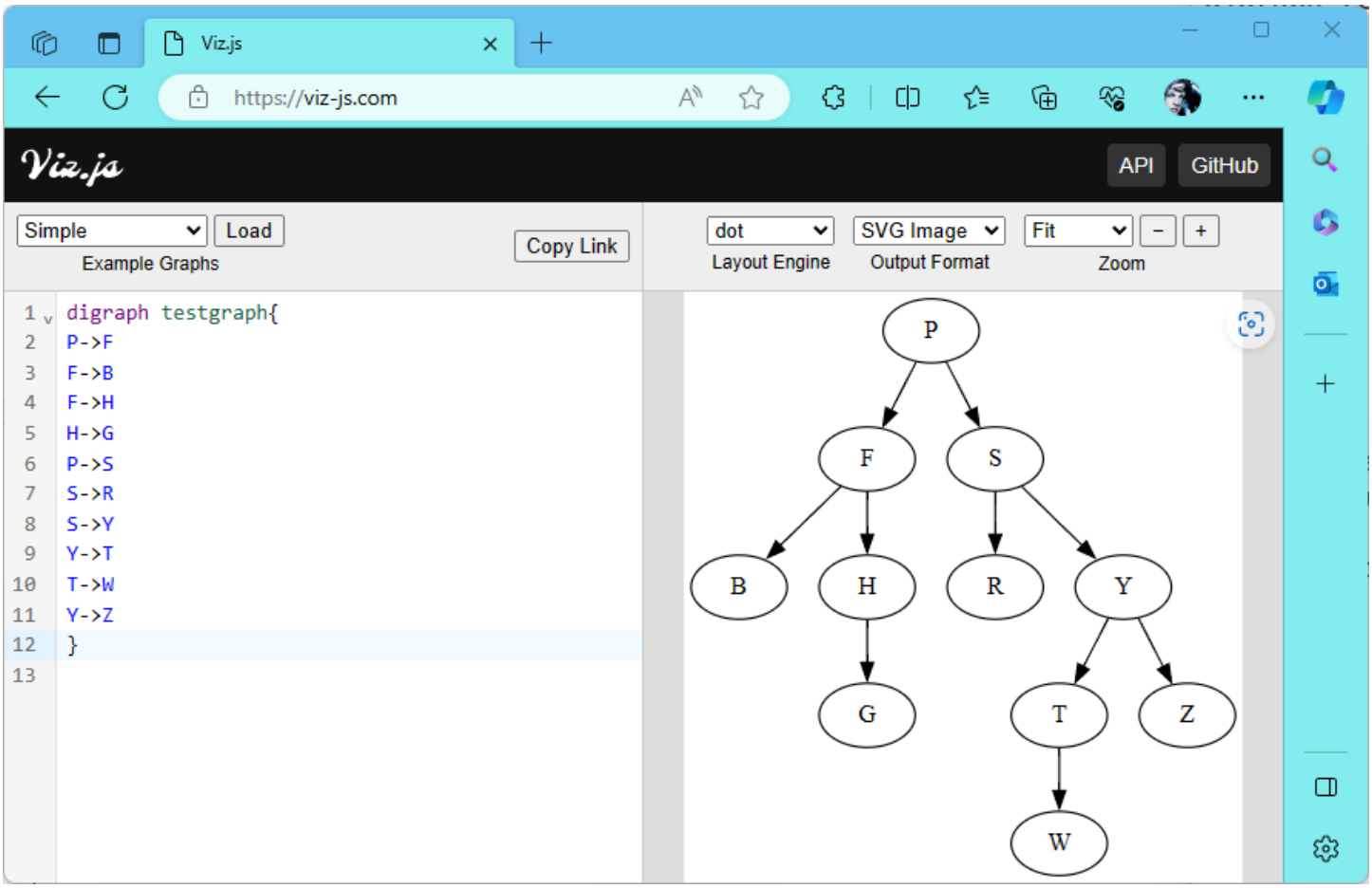

Ilustración 46: Dibujar un árbol binario

Recorrer un árbol binario por niveles

Se hace uso de una lista para almacenar dos datos de cada nodo del árbol: el nodo y su altura. Luego se recorre varias veces esa lista, mostrando los nodos de cada determinada altura.

G/019.cs

```
namespace Ejemplo;

//Recorrido por niveles de un árbol binario
class Nodo {
   public char Letra { get; set; }
   public Nodo Izquierda; //Apuntador
   public Nodo Derecha; //Apuntador

   //Constructor
   public Nodo(char Letra) {
      this.Letra = Letra;
   }
}

class NodosNivel {
   public int Altura { get; set; }
   public Nodo nodo;

   public NodosNivel(int Altura, Nodo nodo) {
      this.Altura = Altura;
      this.nodo = nodo;
   }
}

class Program {
   static void Main(string[] args) {
      List<NodosNivel> niveles = [];

      //Crea el árbol
      Nodo Arbol = new('P');
      Arbol.Izquierda = new('F');
      Arbol.Derecha = new('S');
      Arbol.Izquierda.Izquierda = new('B');
      Arbol.Izquierda.Derecha = new('H');
      Arbol.Izquierda.Derecha.Izquierda = new('G');
      Arbol.Derecha.Izquierda = new('R');
      Arbol.Derecha.Derecha = new('Y');
      Arbol.Derecha.Derecha.Izquierda = new('T');
      Arbol.Derecha.Derecha.Derecha = new('Z');
      Arbol.Derecha.Derecha.Izquierda.Derecha = new('W');
```

```
      //Recorrido por niveles
      Console.WriteLine("Recorrido por niveles");

      //Arma la lista con la información de Nodo y Altura
      ArmaLista(niveles, Arbol, 0);

      //Una vez armada la lista entonces la explora
      //usando como llave la altura
      bool ExisteNivel;
      int Altura = 0;
      do {
         ExisteNivel = false;

         //Muestra los nodos de esa altura en particular
         for (int cont = 0; cont < niveles.Count; cont++)
            if (niveles[cont].Altura == Altura) {
               Console.Write(niveles[cont].nodo.Letra + " -- ");
               ExisteNivel = true;
            }

         //Salta al siguiente nivel
         Console.WriteLine(" ");
         Altura++;
      } while (ExisteNivel);
   }

   //Arma la lista con la información del nodo y su altura
   public static void ArmaLista(List<NodosNivel> niveles,
                     Nodo arbol, int altura) {
      niveles.Add(new NodosNivel(altura, arbol));
      if (arbol.Izquierda != null)
         ArmaLista(niveles, arbol.Izquierda, altura + 1);

      if (arbol.Derecha != null)
         ArmaLista(niveles, arbol.Derecha, altura + 1);
   }
}
```

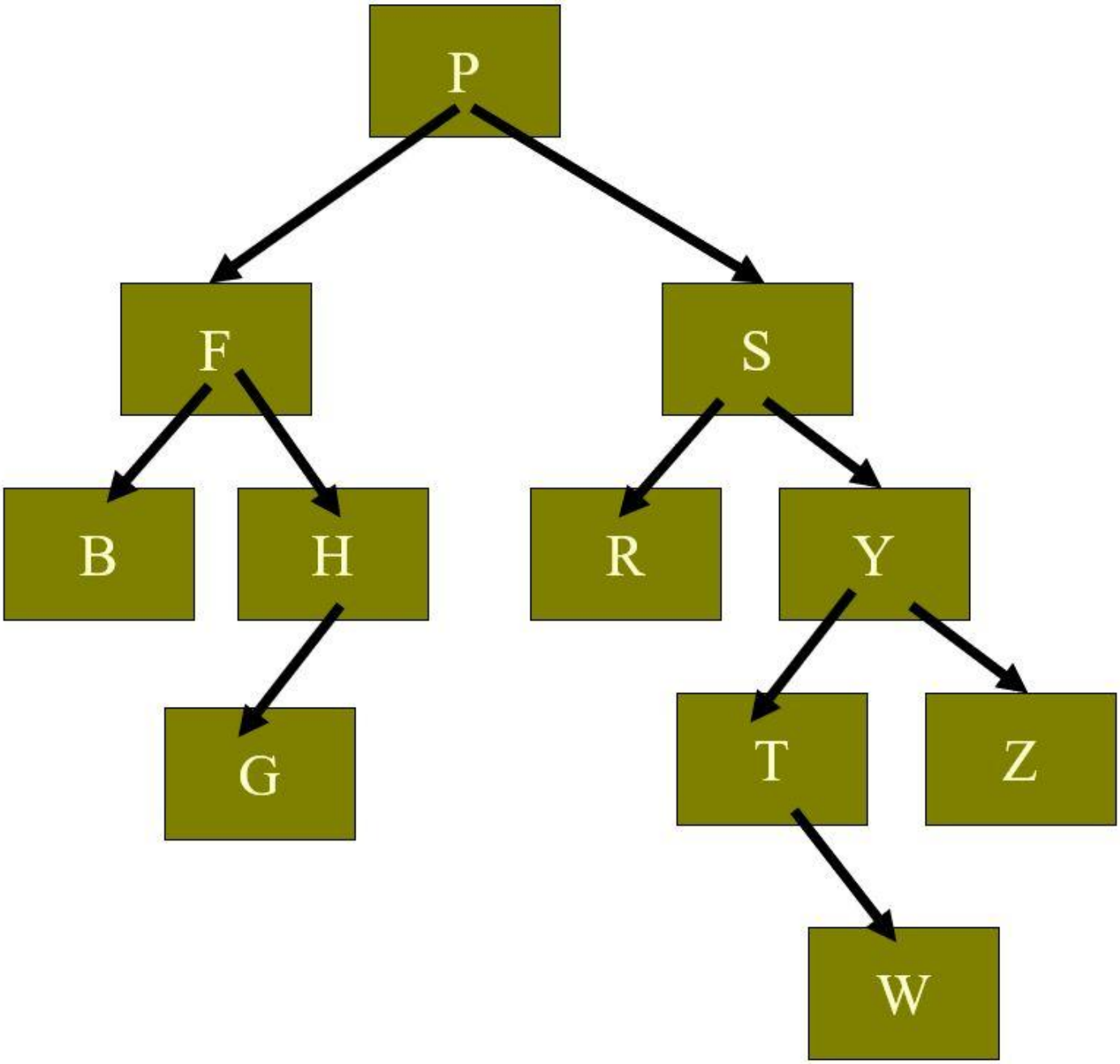

Ilustración 47: Árbol binario de ejemplo

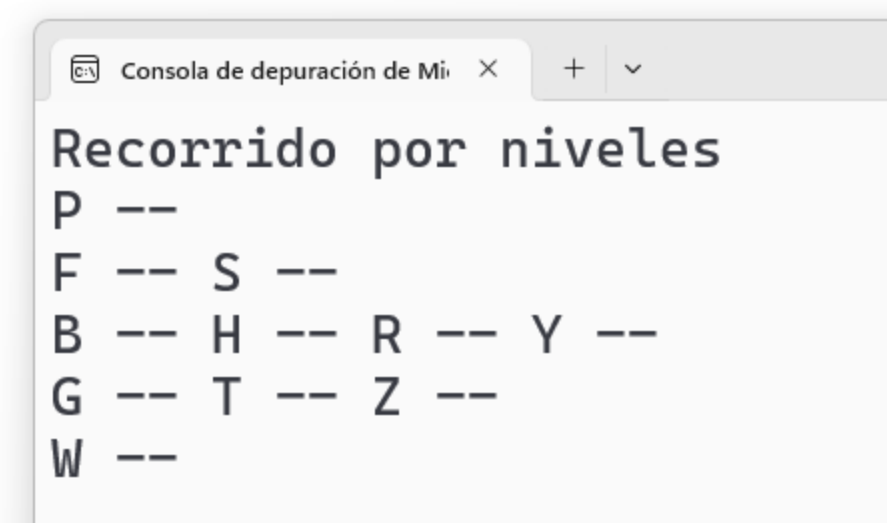

Ilustración 48: Recorrer un árbol binario por niveles

Árbol N-ario

Un árbol donde puede haber de 0 a N hijos por rama

G/020.cs

```
namespace Ejemplo;

class Nodo {
   //Atributos propios
   public string Cad { get; set; }
   public char Car { get; set; }
   public int Entero { get; set; }
   public double Num { get; set; }

   //Uso de una Lista para sostener los hijos del nodo
   public List<Nodo> Hijos;

   public Nodo(string Cad, char Car, int Entero, double Num) {
      this.Cad = Cad;
      this.Car = Car;
      this.Entero = Entero;
      this.Num = Num;
      Hijos = []; //Crea la lista vacía
   }

   public void AgregaHijo(Nodo hijo) {
      Hijos.Add(hijo); //Agrega hijo a la lista
   }

   //Imprime Contenido
   public void Imprime() {
      Console.Write("Cad: " + Cad + " Car: " + Car);
      Console.Write(" Entero: " + Entero + " Real: " + Num);
      Console.WriteLine(" Número de hijos: " + Hijos.Count + "\r\n");
   }
}


class Program {
   static void Main() {
      //Crea la raíz del árbol N-ario
      Nodo arbolN = new("AAAA", 'a', 1, 0.1);

      //Agrega varios hijos a esa raíz
      arbolN.AgregaHijo(new("BBBB", 'b', 2, 0.2));
      arbolN.AgregaHijo(new("CCCC", 'c', 3, 0.3));
```

```
        arbolN.AgregaHijo(new("DDDD", 'd', 4, 0.4));
        arbolN.AgregaHijo(new("EEEE", 'e', 5, 0.5));
        arbolN.AgregaHijo(new("FFFF", 'f', 6, 0.6));

        //Agrega varios hijos al nodo "BBBB"
        arbolN.Hijos[0].AgregaHijo(new("Bhhh", 'h', 7, 0.7));
        arbolN.Hijos[0].AgregaHijo(new("Biii", 'i', 8, 0.8));
        arbolN.Hijos[0].AgregaHijo(new("Bjjj", 'j', 9, 0.9));

        //Agrega varios hijos al nodo "EEEE"
        arbolN.Hijos[3].AgregaHijo(new("Ekkk", 'k', 10, 1.1));
        arbolN.Hijos[3].AgregaHijo(new("Elll", 'l', 11, 1.2));

        //Imprime el árbol
        arbolN.Imprime();
        arbolN.Hijos[0].Imprime();
        arbolN.Hijos[1].Imprime();
        arbolN.Hijos[2].Imprime();
        arbolN.Hijos[3].Imprime();
        arbolN.Hijos[4].Imprime();
        arbolN.Hijos[0].Hijos[0].Imprime();
    }
}
```

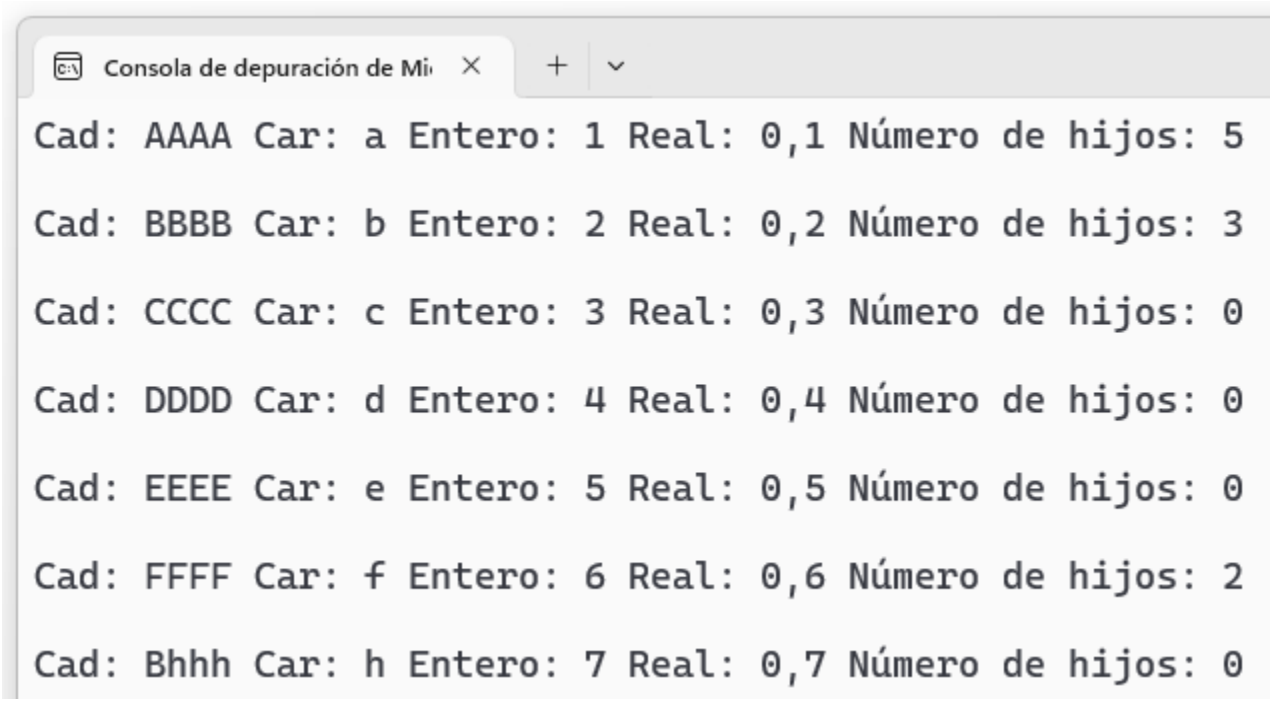

Ilustración 49: Árbol N-ario

Recorriéndolo

Se usa un procedimiento recursivo.

G/021.cs

```
namespace Ejemplo;

class Nodo {
   //Atributos propios
   public string Cad { get; set; }
   public char Car { get; set; }
   public int Entero { get; set; }
   public double Num { get; set; }

   //Uso de una Lista para sostener los hijos del nodo
   public List<Nodo> Hijos;

   public Nodo(string Cad, char Car, int Entero, double Num) {
      this.Cad = Cad;
      this.Car = Car;
      this.Entero = Entero;
      this.Num = Num;
      Hijos = new List<Nodo>(); //Crea la lista vacía
   }

   public void Nuevo(Nodo hijo) {
      Hijos.Add(hijo); //Agrega hijo a la lista
   }

   //Imprime Contenido
   public void Imprime() {
      Console.Write("Cad: " + Cad + " Car: " + Car);
      Console.Write(" Entero: " + Entero + " Real: " + Num);
      Console.WriteLine(" Número de hijos: " + Hijos.Count);
   }
}

class Program {
   static void Main() {
      //Crea la raíz del árbol N-ario
      Nodo arbolN = new("AAAA", 'a', 1, 0.1);

      //Agrega varios hijos a esa raíz
      arbolN.Nuevo(new("BBBB", 'b', 2, 0.2));
      arbolN.Nuevo(new("CCCC", 'c', 3, 0.3));
      arbolN.Nuevo(new("DDDD", 'd', 4, 0.4));
```

```
      arbolN.Nuevo(new("EEEE", 'e', 5, 0.5));
      arbolN.Nuevo(new("FFFF", 'f', 6, 0.6));

      //Agrega varios hijos al nodo "BBBB"
      arbolN.Hijos[0].Nuevo(new("Bhhh", 'h', 7, 0.7));
      arbolN.Hijos[0].Nuevo(new("Biii", 'i', 8, 0.8));
      arbolN.Hijos[0].Nuevo(new("Bjjj", 'j', 9, 0.9));

      //Agrega varios hijos al nodo "EEEE"
      arbolN.Hijos[3].Nuevo(new("Ekkk", 'k', 10, 1.1));
      arbolN.Hijos[3].Nuevo(new("Elll", 'l', 11, 1.2));
      arbolN.Hijos[3].Nuevo(new("Emmm", 'm', 12, 1.3));

      //Agrega varios hijos al nodo "Biii"
      arbolN.Hijos[0].Hijos[1].Nuevo(new("Biiia", 'n', 13, 1.4));
      arbolN.Hijos[0].Hijos[1].Nuevo(new("Biiib", 'o', 14, 1.5));
      arbolN.Hijos[0].Hijos[1].Nuevo(new("Biiic", 'p', 15, 1.6));
      arbolN.Hijos[0].Hijos[1].Nuevo(new("Biiid", 'q', 16, 1.7));
      arbolN.Hijos[0].Hijos[1].Nuevo(new("Biiie", 'r', 17, 1.8));

      //Imprime el árbol
      RecorreArbolN(arbolN);
   }

   //Recorre el árbol
   static void RecorreArbolN(Nodo Arbol) {
      if (Arbol != null) {
         Arbol.Imprime();
         for (int cont = 0; cont < Arbol.Hijos.Count; cont++)
            RecorreArbolN(Arbol.Hijos[cont]);
      }
   }
}
```

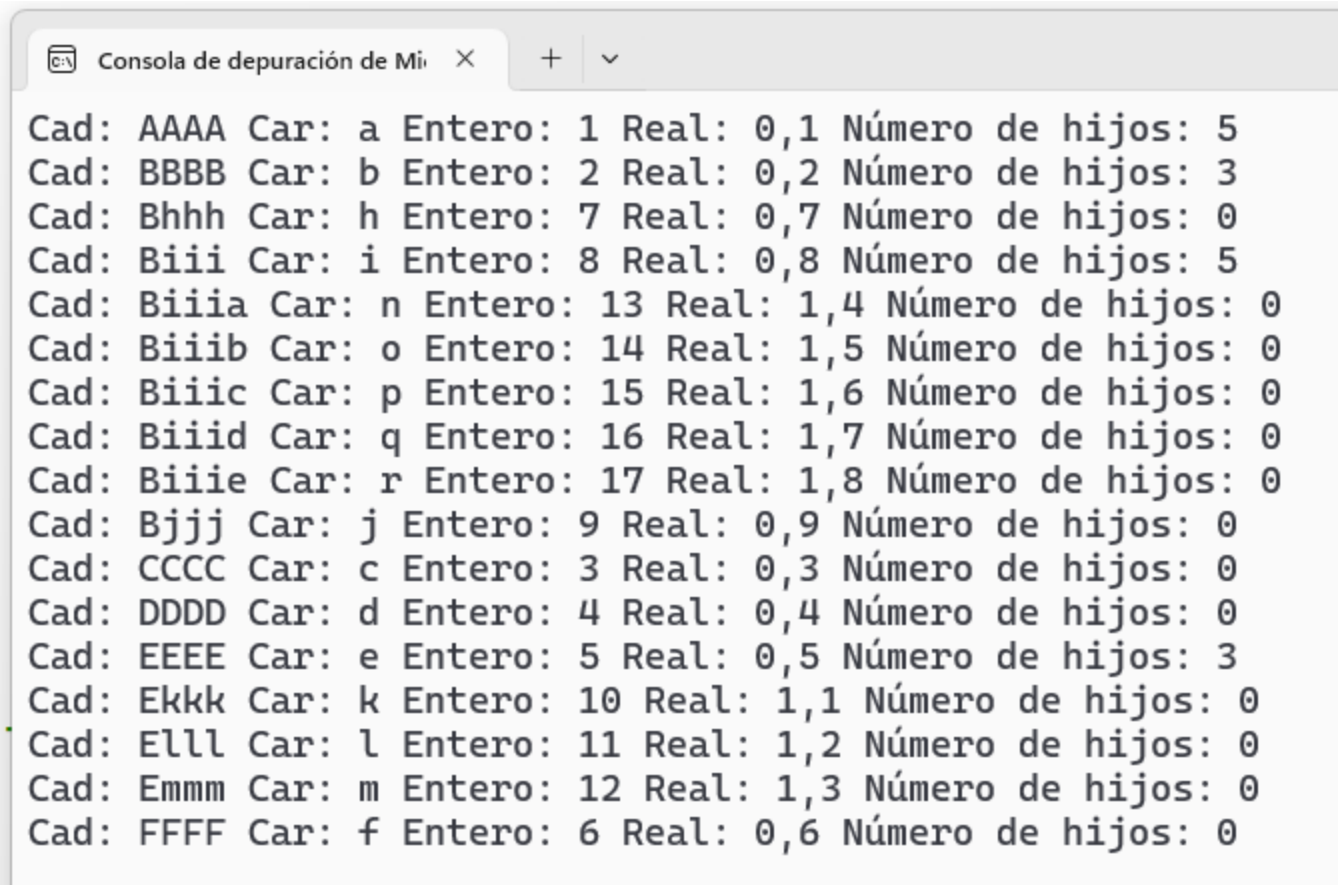

Ilustración 50: Recorriendo árbol N-ario

Grafos

Para generar cualquier otra estructura de nodos interconectados, se hace uso de grafos.

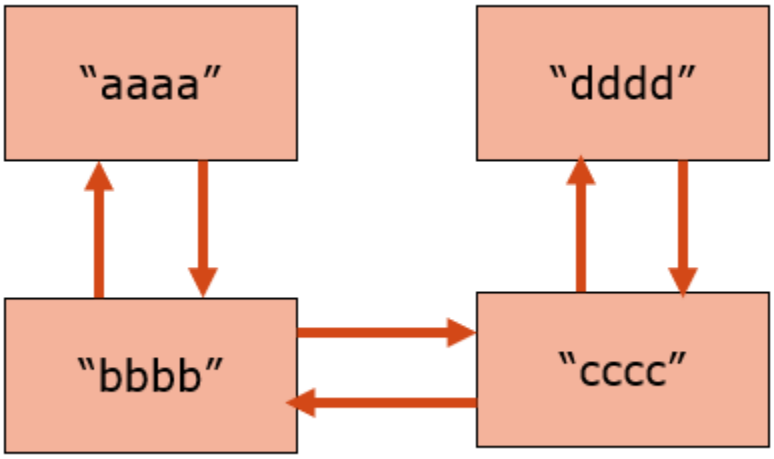

Ilustración 51: Grafo generado

G/022.cs

```
namespace Ejemplo;

//Grafo básico
//Unidad básica en el grafo cuadrado.
//Apuntará Arriba, Abajo, Derecha e Izquierda
class Nodo {
   public string Cad { get; set; }

   //Apuntadores en las 4 direcciones
   public Nodo Arriba;
   public Nodo Abajo;
   public Nodo Derecha;
   public Nodo Izquierda;

   //Constructor
   public Nodo(string Cad) {
      this.Cad = Cad;
   }
}

class Program {
   public static void Main() {
      //Genera los nodos
      Nodo nodoA = new("aaaa");
      Nodo nodoB = new("bbbb");
      Nodo nodoC = new("cccc");
      Nodo nodoD = new("dddd");
```

```
        //Une los nodos para crear el grafo
        nodoA.Abajo = nodoB;
        nodoB.Arriba = nodoA;

        nodoB.Derecha = nodoC;
        nodoC.Izquierda = nodoB;

        nodoC.Arriba = nodoD;
        nodoD.Abajo = nodoC;

        //Imprime
        Console.WriteLine("nodoA: " + nodoA.Cad);

        Console.WriteLine("nodoA->Abajo: " + nodoA.Abajo.Cad);

        Console.Write("nodoA->Abajo->Derecha: ");
        Console.WriteLine(nodoA.Abajo.Derecha.Cad);

        Console.Write("nodoA->Abajo->Derecha->Arriba: ");
        Console.WriteLine(nodoA.Abajo.Derecha.Arriba.Cad);
    }
}
```

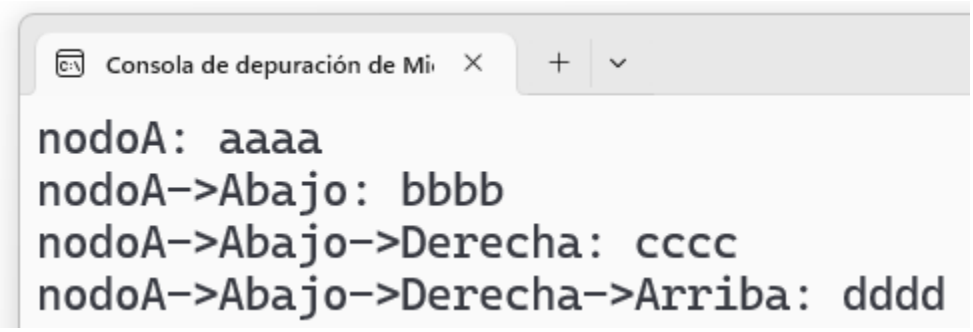

Ilustración 52: Grafos

Generando un grafo aleatoriamente

Aquí se hace uso de un truco: se generan los nodos del grafo agregándolos a un List y luego se recorre nodo por nodo del List para interconectar los nodos con otros nodos.

G/023.cs

```
namespace Ejemplo;

//Generando un grafo aleatoriamente
//Unidad básica en el grafo cuadrado.
//Apuntará Arriba, Abajo, Derecha e Izquierda
class Nodo {
   public int Numero { get; set; }

   //Apuntadores en las 4 direcciones
   public Nodo Arriba;
   public Nodo Abajo;
   public Nodo Derecha;
   public Nodo Izquierda;

   //Constructor
   public Nodo(int Numero) {
      this.Numero = Numero;
   }
}

class Program {
   public static void Main() {
      Random azar = new();

      //Usa una lista para guardar los nodos
      List<Nodo> listado = [];

      //Genera los nodos dentro de un List
      int Total = azar.Next(20, 30);
      for (int cont = 1; cont <= Total; cont++) {
         listado.Add(new(cont));
      }

      //Ahora interconecta los nodos al azar
      Total = azar.Next(50, 200);
      for (int cont = 1; cont <= Total; cont++) {
         int nodoA = azar.Next(listado.Count);
         int nodoB;
         do {
            nodoB = azar.Next(listado.Count);
```

```
            } while (nodoA == nodoB);

            switch (azar.Next(4)) {
               case 0:
                  listado[nodoA].Arriba = listado[nodoB];
                  break;

               case 1:
                  listado[nodoA].Abajo = listado[nodoB];
                  break;

               case 2:
                  listado[nodoA].Izquierda = listado[nodoB];
                  break;

               case 3:
                  listado[nodoA].Derecha = listado[nodoB];
                  break;
            }
         }

         //Imprime el grafo para ser interpretado por viz.js
         Console.WriteLine("digraph testgraph{");
         for (int cont = 0; cont < listado.Count; cont++) {
            if (listado[cont].Arriba != null) {
               Console.Write(listado[cont].Numero + "->");
               Console.WriteLine(listado[cont].Arriba.Numero);
            }

            if (listado[cont].Abajo != null) {
               Console.Write(listado[cont].Numero + "->");
               Console.WriteLine(listado[cont].Abajo.Numero);
            }

            if (listado[cont].Izquierda != null) {
               Console.Write(listado[cont].Numero + "->");
               Console.WriteLine(listado[cont].Izquierda.Numero);
            }

            if (listado[cont].Derecha != null) {
               Console.Write(listado[cont].Numero + "->");
               Console.WriteLine(listado[cont].Derecha.Numero);
            }
         }
         Console.WriteLine("}");
      }
   }
```

Ilustración 53: Gafo generado

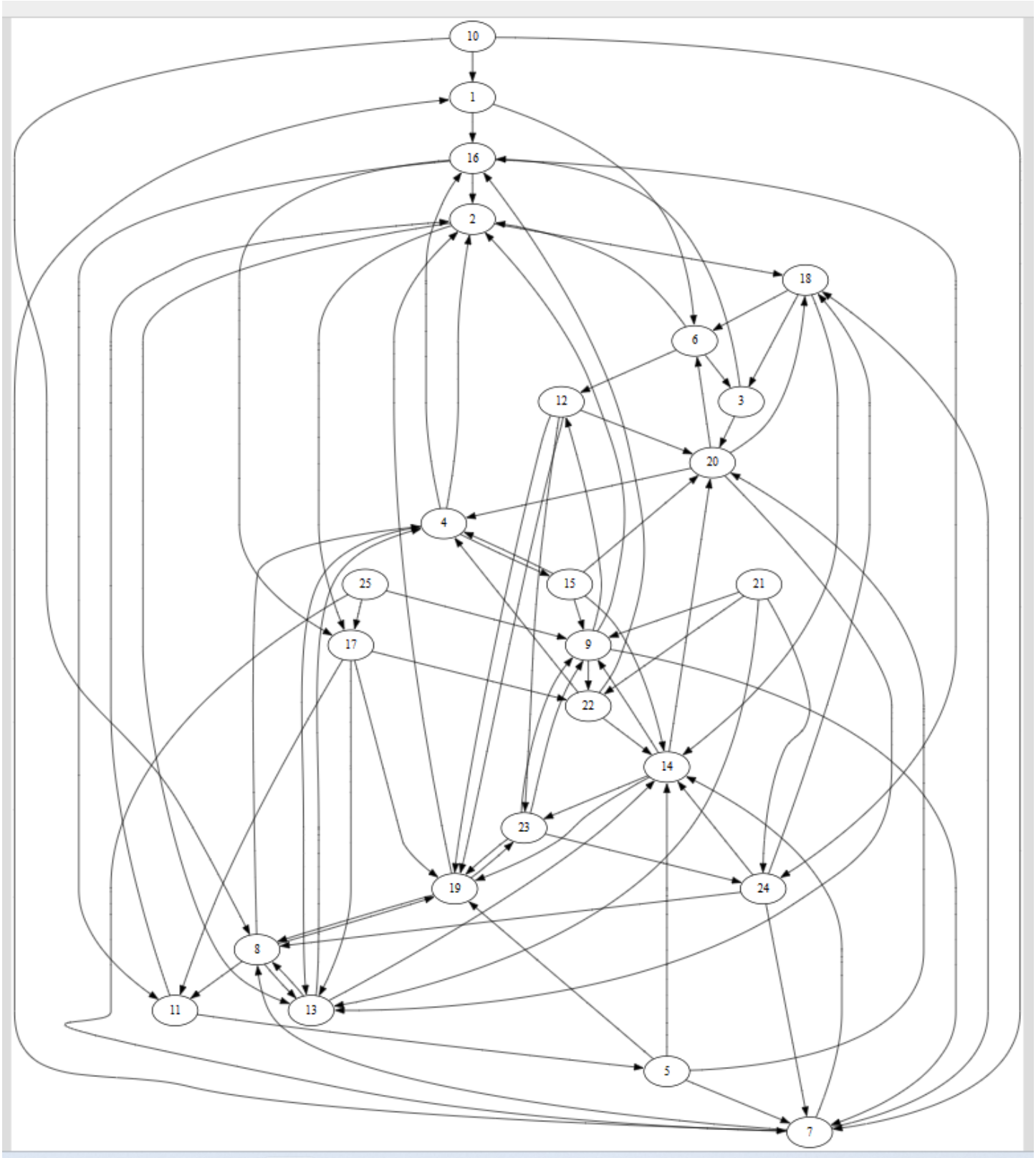

Ilustración 54: Grafo dibujado con https://viz-js.com/

Guardando un árbol binario en medio persistente

Se hace uso de JSON.

¡OJO! Los apuntadores deben ser propiedades (no sólo atributos)

G/024.cs

```
using System.Text.Json;

namespace Ejemplo;

//Recorrido de un árbol binario
class Nodo {
   public char Letra { get; set; }
   public Nodo Izquierda { get; set; } //Apuntador
   public Nodo Derecha { get; set; } //Apuntador

   //Constructor
   public Nodo(char Letra) {
      this.Letra = Letra;
   }
}

class Program {
   public static void Main() {
      //Crea el árbol
      Nodo Arbol = new('M');
      Arbol.Izquierda = new('E');
      Arbol.Derecha = new('P');
      Arbol.Izquierda.Izquierda = new('B');
      Arbol.Izquierda.Derecha = new('L');
      Arbol.Izquierda.Izquierda.Izquierda = new('A');
      Arbol.Izquierda.Izquierda.Derecha = new('D');
      Arbol.Derecha.Izquierda = new('N');
      Arbol.Derecha.Derecha = new('V');
      Arbol.Derecha.Derecha.Izquierda = new('T');
      Arbol.Derecha.Derecha.Derecha = new('Z');

      //Imprime árbol
      Console.WriteLine("ARBOL ORIGINAL");

      //Recorridos
      Console.WriteLine("PreOrden (raiz, izquierdo, derecho)");
      PreOrden(Arbol);

      Console.WriteLine("\n\nInOrden (izquierdo, raiz, derecho)");
      InOrden(Arbol);
```

```
        Console.WriteLine("\n\nPostOrden (izquierdo, derecho, raiz)");
        PostOrden(Arbol);

        // Guardar como JSON
        GuardarComoJSON(Arbol, "arbol.json");
        Console.WriteLine("\r\n\r\nÁrbol binario guardado\r\n");

        //Recupera el árbol
        Nodo Recupera = CargarDesdeJSON("arbol.json");
        Console.WriteLine("Árbol binario recuperado de archivo JSON");

        Console.WriteLine("PreOrden (raiz, izquierdo, derecho)");
        PreOrden(Recupera);

        Console.WriteLine("\n\nInOrden (izquierdo, raiz, derecho)");
        InOrden(Recupera);

        Console.WriteLine("\n\nPostOrden (izquierdo, derecho, raiz)");
        PostOrden(Recupera);

    }

    static void GuardarComoJSON(Nodo raiz, string rutaArchivo) {
        var opciones = new JsonSerializerOptions { WriteIndented = true };
        string json = JsonSerializer.Serialize(raiz, opciones);
        File.WriteAllText(rutaArchivo, json);
    }

    static Nodo CargarDesdeJSON(string rutaArchivo) {
        string json = File.ReadAllText(rutaArchivo);
        Nodo raiz = JsonSerializer.Deserialize<Nodo>(json);
        return raiz;
    }

    static void PreOrden(Nodo Arbol) {
        if (Arbol != null) {
            Console.Write(Arbol.Letra + ", ");
            PreOrden(Arbol.Izquierda);
            PreOrden(Arbol.Derecha);
        }
    }
    static void InOrden(Nodo Arbol) {
        if (Arbol != null) {
            InOrden(Arbol.Izquierda);
            Console.Write(Arbol.Letra + ", ");
            InOrden(Arbol.Derecha);
        }
```

```
    }

    static void PostOrden(Nodo Arbol) {
        if (Arbol != null) {
            PostOrden(Arbol.Izquierda);
            PostOrden(Arbol.Derecha);
            Console.Write(Arbol.Letra + ", ");
        }
    }
}
```

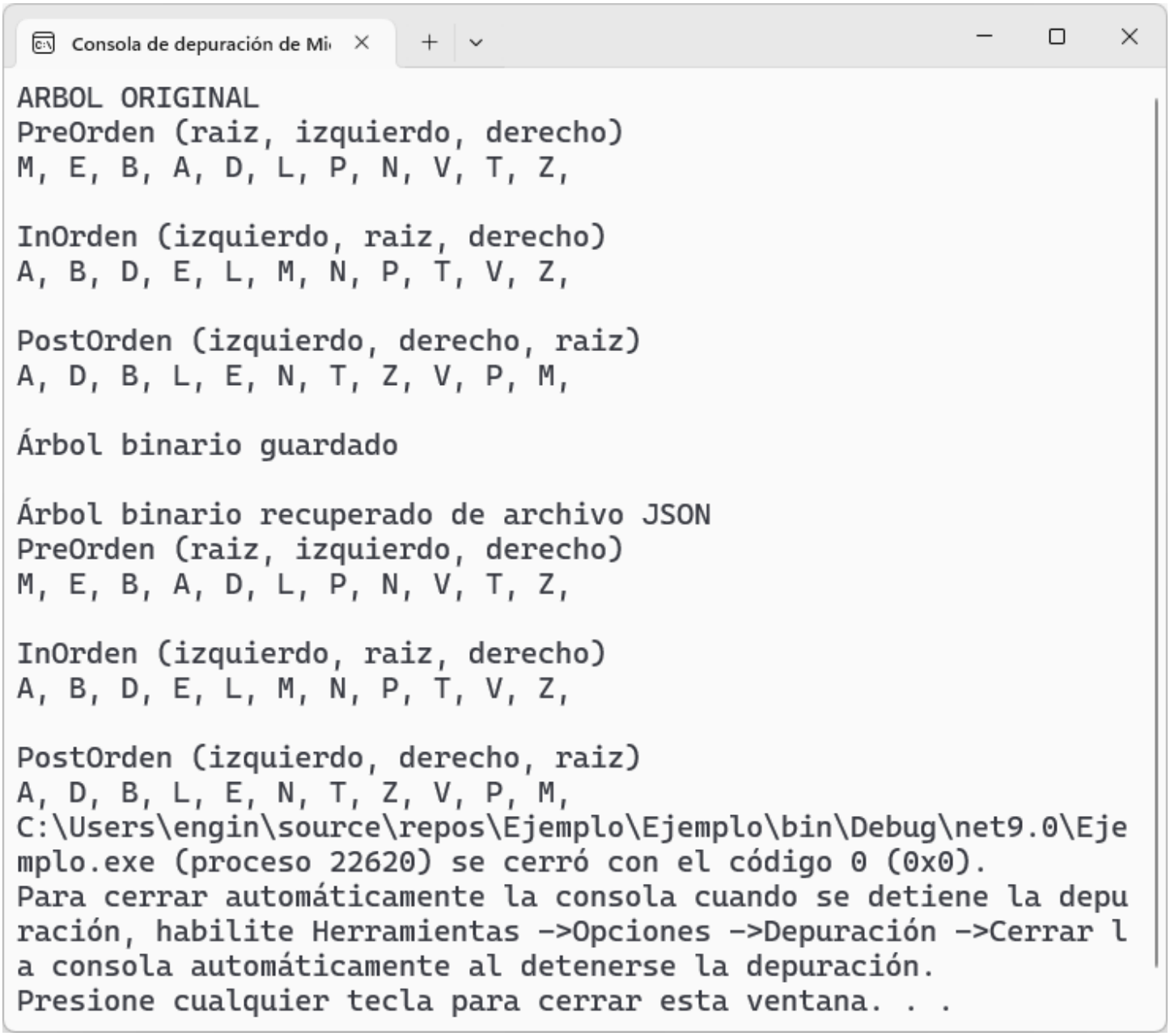

Ilustración 55: Árbol binario guardado usando JSON

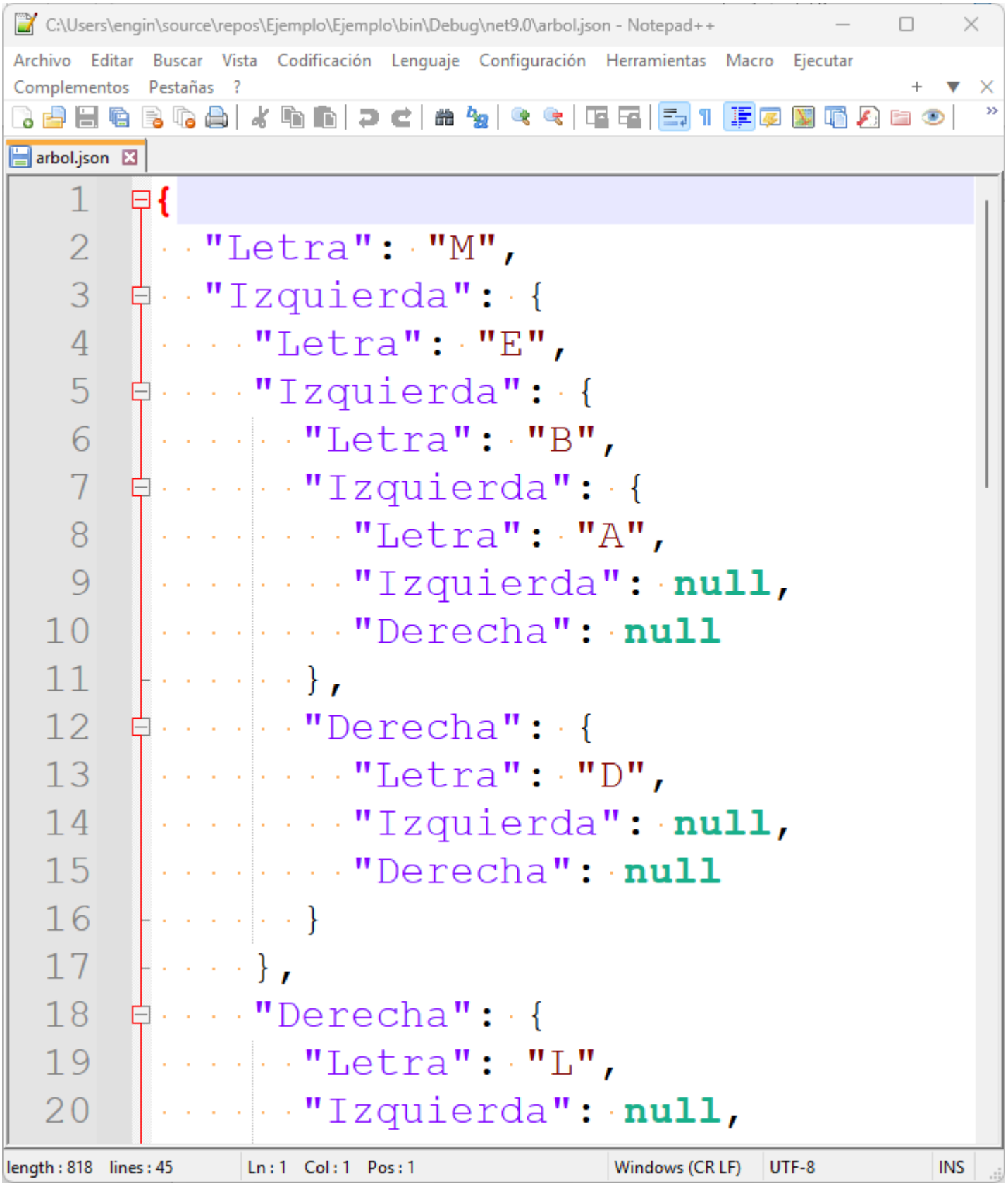

Ilustración 56: Archivo JSON

Guardando un árbol N-ario en un medio persistente

G/025.cs

```
using System.Text.Json;

namespace Ejemplo;

class Nodo {
   //Atributos propios
   public string Cad { get; set; }
   public char Car { get; set; }
   public int Entero { get; set; }
   public double Num { get; set; }

   //Uso de una Lista para sostener los hijos del nodo
   public List<Nodo> Hijos { get; set; } = new List<Nodo>();

   public Nodo(string Cad, char Car, int Entero, double Num) {
      this.Cad = Cad;
      this.Car = Car;
      this.Entero = Entero;
      this.Num = Num;
      Hijos = []; //Crea la lista vacía
   }

   public void AgregaHijo(Nodo hijo) {
      Hijos.Add(hijo); //Agrega hijo a la lista
   }

   //Imprime Contenido
   public void Imprime() {
      Console.Write("Cad: " + Cad + " Car: " + Car);
      Console.Write(" Entero: " + Entero + " Real: " + Num);
      Console.WriteLine(" Número de hijos: " + Hijos.Count + "\r\n");
   }
}


class Program {
   static void Main() {
      //Crea la raíz del árbol N-ario
      Nodo arbolN = new("AAAA", 'a', 1, 0.1);

      //Agrega varios hijos a esa raíz
      arbolN.AgregaHijo(new("BBBB", 'b', 2, 0.2));
      arbolN.AgregaHijo(new("CCCC", 'c', 3, 0.3));
```

```
        arbolN.AgregaHijo(new("DDDD", 'd', 4, 0.4));
        arbolN.AgregaHijo(new("EEEE", 'e', 5, 0.5));
        arbolN.AgregaHijo(new("FFFF", 'f', 6, 0.6));

        //Agrega varios hijos al nodo "BBBB"
        arbolN.Hijos[0].AgregaHijo(new("Bhhh", 'h', 7, 0.7));
        arbolN.Hijos[0].AgregaHijo(new("Biii", 'i', 8, 0.8));
        arbolN.Hijos[0].AgregaHijo(new("Bjjj", 'j', 9, 0.9));

        //Agrega varios hijos al nodo "EEEE"
        arbolN.Hijos[3].AgregaHijo(new("Ekkk", 'k', 10, 1.1));
        arbolN.Hijos[3].AgregaHijo(new("Elll", 'l', 11, 1.2));

        //Imprime el árbol
        RecorrerPreorden(arbolN);

        //Guarda el árbol N-ario
        Console.WriteLine("Guarda árbol N-ario");
        GuardarArbolNarioComoJSON(arbolN, "arbol.json");

        //Restaura el árbol N-ario
        Console.WriteLine("Restaura árbol N-ario");
        Nodo Restaura = CargarArbolNarioDesdeJSON("arbol.json");

        //Imprime el árbol N-ario restaurado
        RecorrerPreorden(Restaura);
    }

    static void RecorrerPreorden(Nodo nodo) {
        if (nodo == null) return;
        nodo.Imprime();
        foreach (var hijo in nodo.Hijos) {
            RecorrerPreorden(hijo);
        }
    }

    static void GuardarArbolNarioComoJSON(Nodo raiz, string rutaArchivo) {
        var opciones = new JsonSerializerOptions { WriteIndented = true };
        string json = JsonSerializer.Serialize(raiz, opciones);
        File.WriteAllText(rutaArchivo, json);
    }

    static Nodo CargarArbolNarioDesdeJSON(string rutaArchivo) {
        string json = File.ReadAllText(rutaArchivo);
        return JsonSerializer.Deserialize<Nodo>(json);
    }
}
```

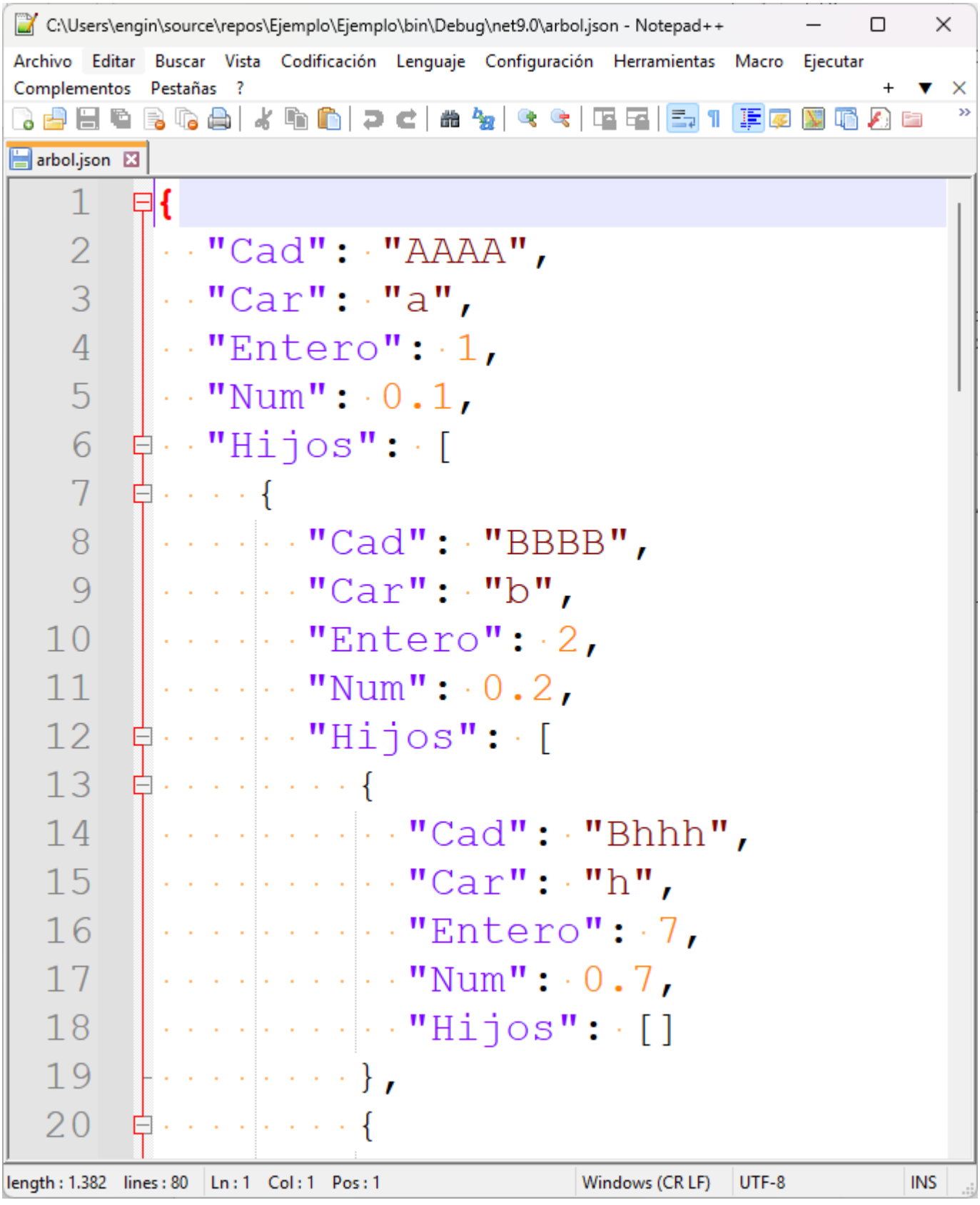

Ilustración 57: JSON de un árbol N-ario