



2021

C#: Árboles binarios, n-arios, grafos, listas simple y doblemente enlazadas

Rafael Alberto
Moreno Parra

Contenido

Otros libros del autor 2

Página web del autor y canal en Youtube 3

Sitio en GitHub 3

Licencia del software 3

Marcas registradas 3

Dedicatoria 4

Introducción..... 5

La lista simplemente enlazada..... 6

 Concepto 6

 Enlazamiento continuo 9

 Recorriéndola 11

 Recorriendo la lista usando un método..... 13

 Tamaño 15

 Traer un determinado nodo..... 17

 Adicionar un nodo en determinada posición 19

 Borrar un nodo de una determinada posición 22

La lista doblemente enlazada 25

 Definición..... 25

 Adicionar un nodo en determinada posición 28

 Borrar un nodo de una determinada posición 32

Árbol binario 36

 Definición y recorridos recursivos 36

 Primer ejemplo..... 36

 Segundo ejemplo..... 38

 Recorrido iterativo (no recursivo)..... 40

 Generar árboles binarios al azar..... 42

 Ordenamiento usando un árbol binario..... 44

 Buscar en árbol binario ordenado, número de nodos y altura del árbol 46

 Dibujar un árbol binario..... 48

 Recorrer un árbol binario por niveles 51

Árbol N-ario 53

 Definición..... 53

 Recorriéndolo 55

Grafos 58

 Generando un grafo al azar y dibujarlo 60

Bibliografía..... 62

Otros libros del autor

Libro 17: "C#. Estructuras dinámicas de memoria". En Colombia 2021. Págs. 82. Libro y código fuente descargable en: <https://github.com/ramsoftware/CSharpDinamica>

Libro 16: "C#. Programación Orientada a Objetos". En Colombia 2020. Págs. 90. Libro y código fuente descargable en: <https://github.com/ramsoftware/C-Sharp-POO>

Libro 15: "C#. Estructuras básicas de memoria.". En Colombia 2020. Págs. 60. Libro y código fuente descargable en: <https://github.com/ramsoftware/EstructuraBasicaMemoriaCSharp>

Libro 14: "Iniciando en C#". En Colombia 2020. Págs. 72. Libro y código fuente descargable en: <https://github.com/ramsoftware/C-Sharp-Iniciando>

Libro 13: "Algoritmos Genéticos". En Colombia 2020. Págs. 62. Libro y código fuente descargable en: <https://github.com/ramsoftware/LibroAlgoritmoGenetico2020>

Libro 12: "Redes Neuronales. Segunda Edición". En Colombia 2020. Págs. 108. Libro y código fuente descargable en: <https://github.com/ramsoftware/LibroRedNeuronal2020>

Libro 11: "Capacitándose en JavaScript". En Colombia 2020. Págs. 317. Libro y código fuente descargable en: <https://github.com/ramsoftware/JavaScript>

Libro 10: "Desarrollo de aplicaciones para Android usando MIT App Inventor 2". En Colombia 2016. Págs. 102. Ubicado en: <https://openlibra.com/es/book/desarrollo-de-aplicaciones-para-android-usando-mit-app-inventor-2>

Libro 9: "Redes Neuronales. Parte 1.". En Colombia 2016. Págs. 90. Libro descargable en: <https://openlibra.com/es/book/redes-neuronales-parte-1>

Libro 8: "Segunda parte de uso de algoritmos genéticos para la búsqueda de patrones". En Colombia 2015. Págs. 303. En publicación por la Universidad Libre – Cali.

Libro 7: "Desarrollo de un evaluador de expresiones algebraicas. **Versión 2.0.** C++, C#, Visual Basic .NET, Java, PHP, JavaScript y Object Pascal (Delphi)". En: Colombia 2013. Págs. 308. Ubicado en: <https://openlibra.com/es/book/evaluador-de-expresiones-algebraicas-ii>

Libro 6: "Un uso de algoritmos genéticos para la búsqueda de patrones". En Colombia 2013. En publicación por la Universidad Libre – Cali.

Libro 5: Desarrollo fácil y paso a paso de aplicaciones para Android usando MIT App Inventor. En Colombia 2013. Págs. 104. Estado: Obsoleto (No hay enlace).

Libro 4: "Desarrollo de un evaluador de expresiones algebraicas. C++, C#, Visual Basic .NET, Java, PHP, JavaScript y Object Pascal (Delphi)". En Colombia 2012. Págs. 308. Ubicado en: <https://openlibra.com/es/book/evaluador-de-expresiones-algebraicas>

Libro 3: "Simulación: Conceptos y Programación" En Colombia 2012. Págs. 81. Ubicado en: <https://openlibra.com/es/book/simulacion-conceptos-y-programacion>

Libro 2: "Desarrollo de videojuegos en 2D con Java y Microsoft XNA". En Colombia 2011. Págs. 260. Ubicado en: <https://openlibra.com/es/book/desarrollo-de-juegos-en-2d-usando-java-y-microsoft-xna> . ISBN: 978-958-8630-45-8

Libro 1: "Desarrollo de gráficos para PC, Web y dispositivos móviles" En Colombia 2009. ed.: Artes Gráficas Del Valle Editores Impresores Ltda. ISBN: 978-958-8308-95-1 v. 1 págs. 317

Artículo: "Programación Genética: La regresión simbólica". Entramado ISSN: 1900-3803 ed.: Universidad Libre Seccional Cali v.3 fasc.1 p.76 - 85, 2007

Página web del autor y canal en Youtube

Investigación sobre Vida Artificial: <http://darwin.50webs.com>

Canal en Youtube: <http://www.youtube.com/user/RafaelMorenoP> (dedicado principalmente al desarrollo en C#)

Sitio en GitHub

El código fuente se puede descargar en <https://github.com/ramsoftware/C-Sharp-Arboles>

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 2019 ® Enlace: <https://visualstudio.microsoft.com/es/vs/>

Dedicatoria

A mis padres, a mi hermana....

Y a mi tropa gatuna: Sally, Suini, Vikingo, Grisú, Capuchina, Milú y mi recordada Tammy.

Introducción

Siguiendo con la serie de libros de C# que he escrito anteriormente:

1. Iniciando en C# <https://github.com/ramsoftware/C-Sharp-Iniciando>
2. C#. Estructuras básicas de memoria <https://github.com/ramsoftware/EstructuraBasicaMemoriaCSharp>
3. C#. Programación Orientada a Objetos <https://github.com/ramsoftware/C-Sharp-POO>
4. C#. Estructuras de datos dinámicas <https://github.com/ramsoftware/CSharpDinamica>

Este libro finalizado en marzo de 2021 se concentra en las estructuras de datos implementadas a bajo nivel, es decir, no hace uso de APIs (funciones del lenguaje) sino que son construidas haciendo uso de apuntadores. Las listas simples y doblemente enlazadas son construidas para conocer el concepto, pero no las recomiendo cuando existe el List o el ArrayList, mucho más fáciles de usar y optimizadas. El concepto aprendido con las listas simples y doblemente enlazadas es aplicado en estructuras con otras topologías como son los árboles binarios, los árboles n-arios y los grafos, los cuales no tienen APIs nativas.

El código fuente se puede descargar de GitHub en: <https://github.com/ramsoftware/C-Sharp-Arboles>

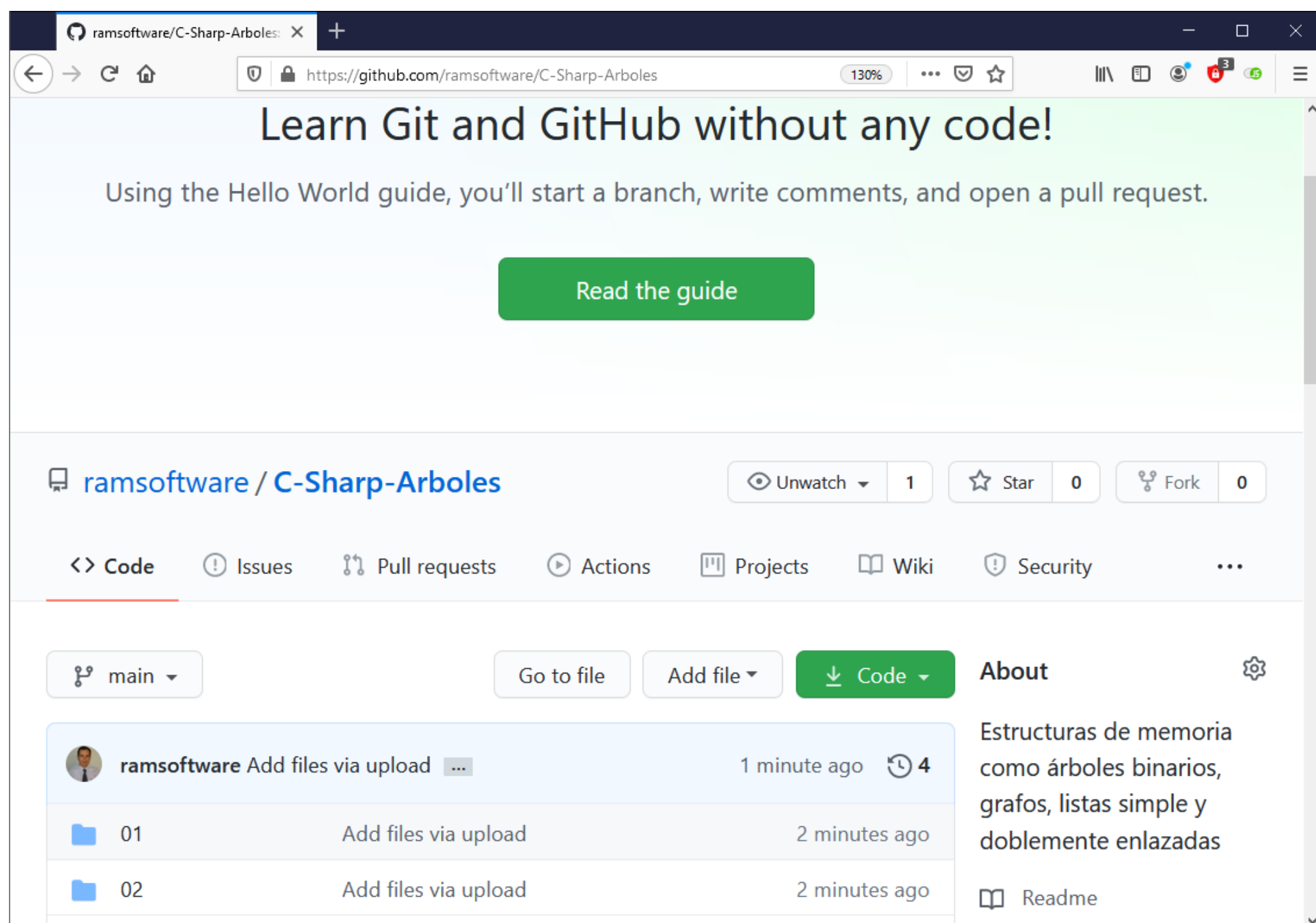


Ilustración 1: Sitio GitHub del libro y el código fuente: <https://github.com/ramsoftware/C-Sharp-Arboles>

La lista simplemente enlazada

Concepto

Se representa de esta forma:

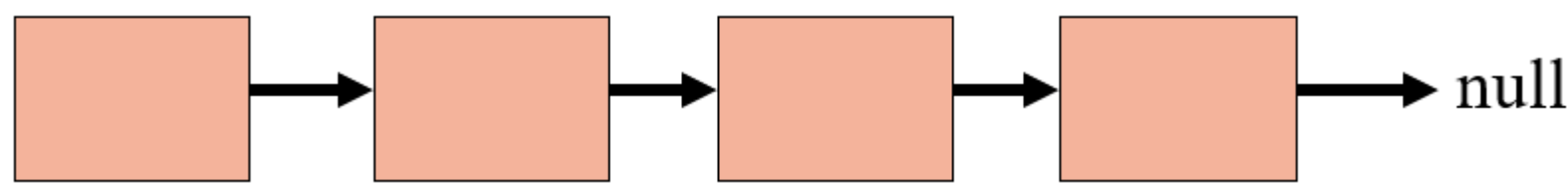


Ilustración 2: Representación gráfica de una lista simplemente enlazada

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

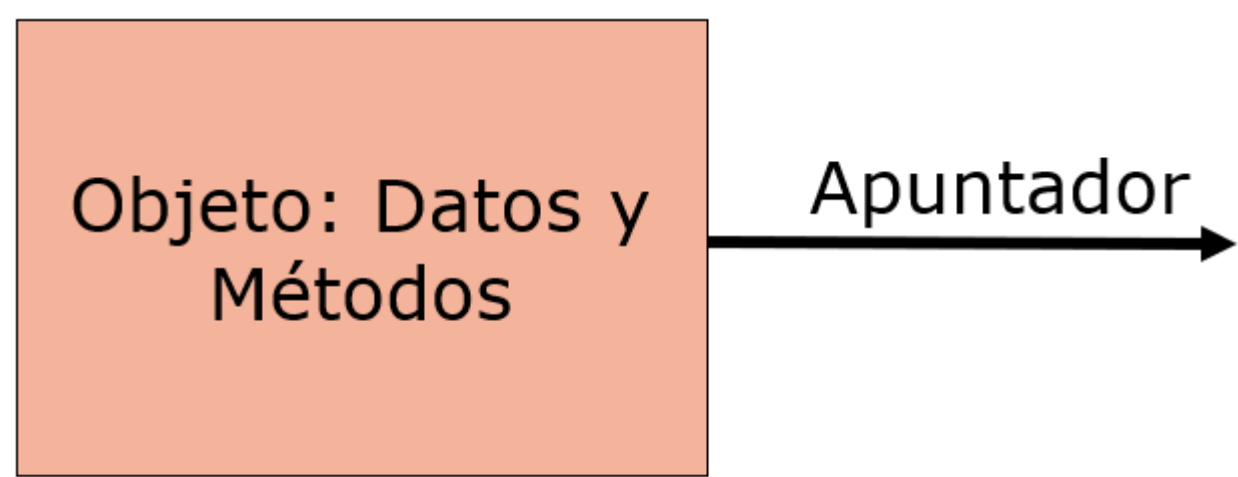


Ilustración 3: El nodo es un objeto con sus atributos, métodos y un apuntador

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

Directorio 01. Nodo.cs

```
using System;

namespace ArbolGrafoLista {
    class Nodo {
        //Atributos propios
        public string Cadena { get; set; }
        public char Caracter { get; set; }
        public int Entero { get; set; }
        public double NumReal { get; set; }

        //Apuntador para lista simplemente enlazada
        public Nodo Apuntador;

        //Constructor
        public Nodo(string Cadena, char Caracter, int Entero, double NumReal) {
            this.Cadena = Cadena;
            this.Caracter = Caracter;
            this.Entero = Entero;
            this.NumReal = NumReal;
        }

        //Imprime Contenido
        public void Imprime() {
            Console.WriteLine("Cadena: " + Cadena + " Caracter: " + Caracter.ToString());
            Console.WriteLine("Entero: " + Entero.ToString() + " Real: " + NumReal.ToString());
        }
    }
}
```

Se crean tres nodos:

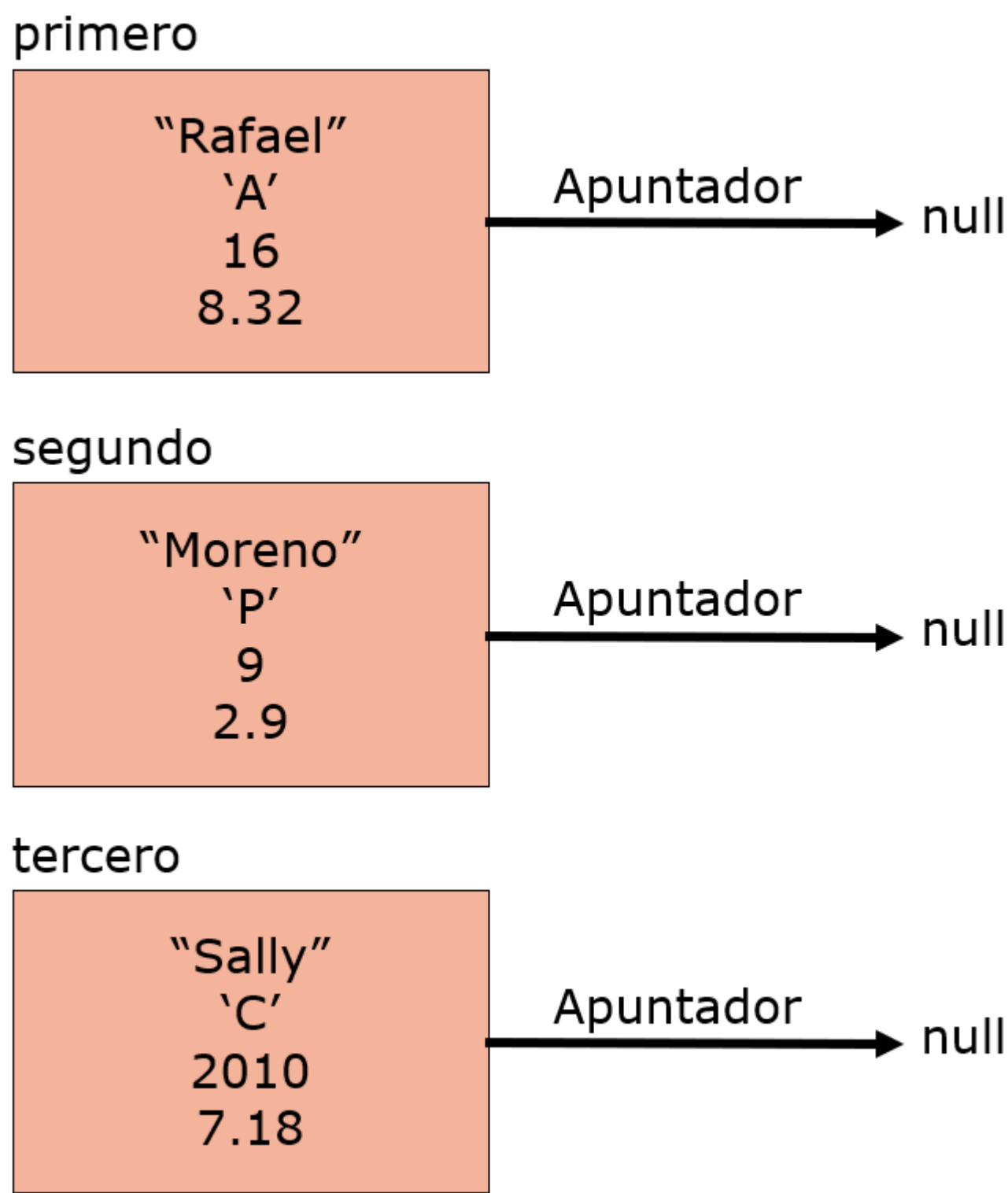


Ilustración 4: Se crean tres nodos

Este sería el código en C#:

```
//Crea tres nodos separados
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:

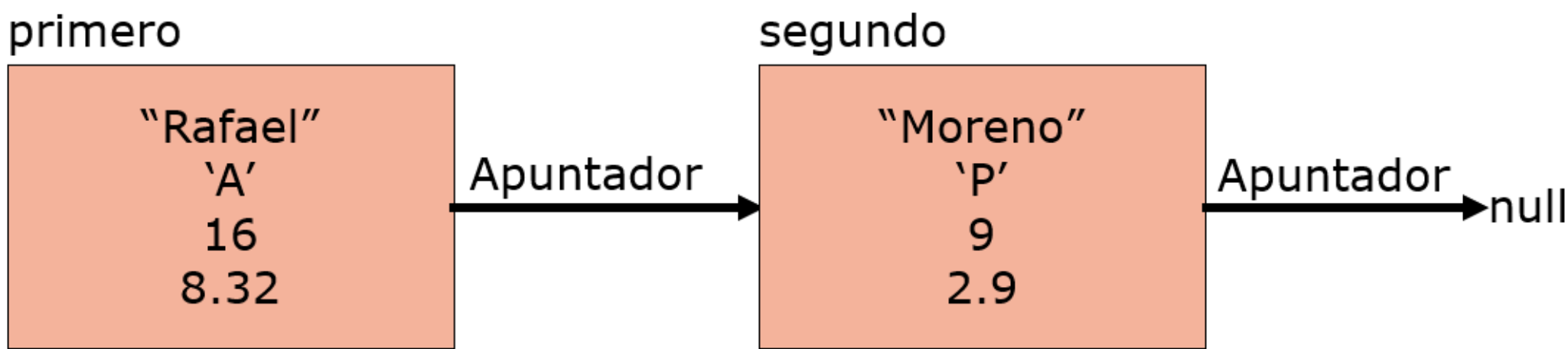
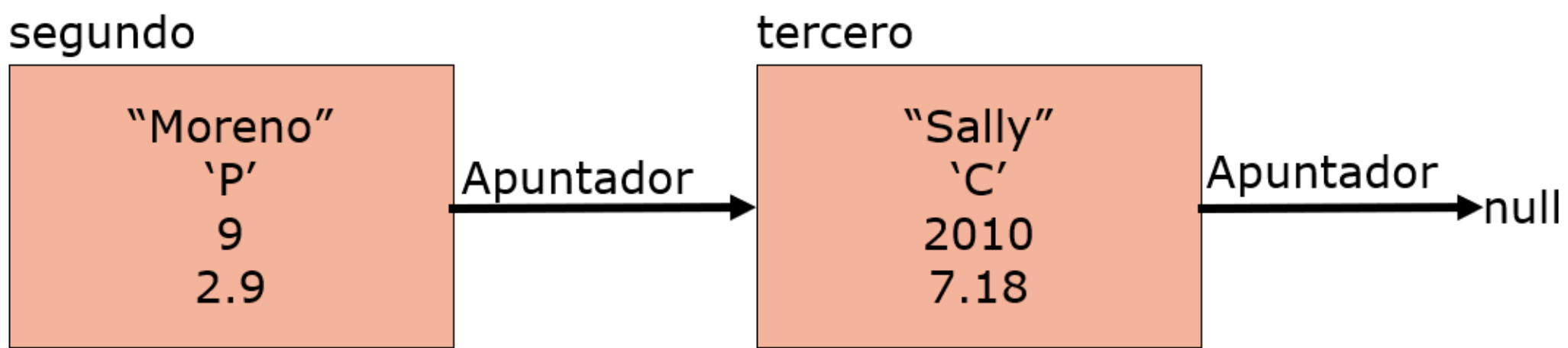


Ilustración 5: Conectar el nodo primero con el nodo segundo

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:



Este sería el código en C#:

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

El código completo:

Directorio 01. Program.cs

```
using System;

namespace ArbolGrafoLista {
    class Program {
        static void Main() {
            //Crea tres nodos separados
            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);

            //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(); //Primer nodo
            primero.Apuntador.Imprime(); //Segundo nodo
            primero.Apuntador.Apuntador.Imprime(); //Tercer nodo
            Console.ReadKey();
        }
    }
}
```

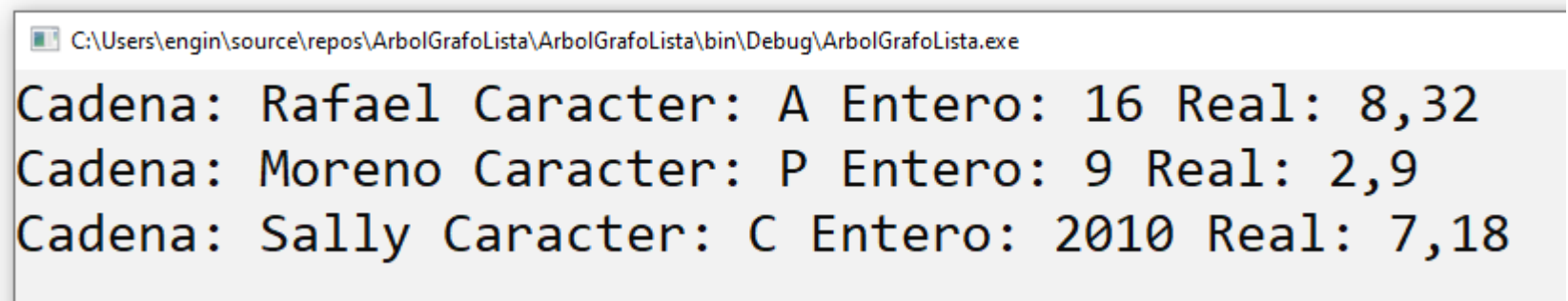


Ilustración 6: Lista simplemente enlazada. Definición.

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.

Directorio 02. Nodo.cs

```
using System;

namespace ArbolGrafoLista {
    class Nodo {
        //Atributos propios
        public string Cadena { get; set; }
        public char Caracter { get; set; }
        public int Entero { get; set; }
        public double NumReal { get; set; }

        //Apuntador para lista simplemente enlazada
        public Nodo Apuntador;

        //Constructor
        public Nodo(string Cadena, char Caracter, int Entero, double NumReal, Nodo Apuntador) {
            this.Cadena = Cadena;
            this.Caracter = Caracter;
            this.Entero = Entero;
            this.NumReal = NumReal;
            this.Apuntador = Apuntador;
        }

        //Imprime Contenido
        public void Imprime() {
            Console.WriteLine("Cadena: " + Cadena + " Caracter: " + Caracter.ToString());
            Console.WriteLine(" Entero: " + Entero.ToString() + " Real: " + NumReal.ToString());
        }
    }
}
```

Crea el primer nodo:

lista

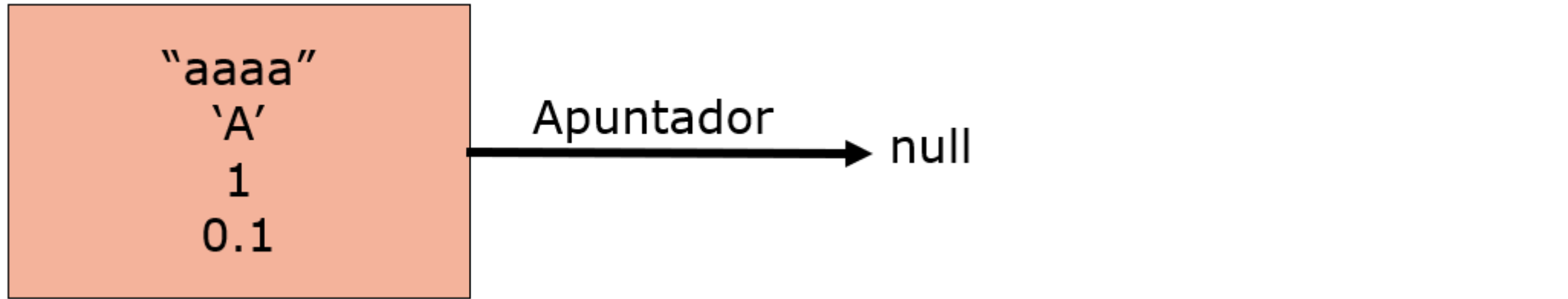


Ilustración 7: Crea el primer Nodo

Con esta instrucción en C#:

```
//Crea la lista
Nodo lista = new Nodo("aaaa", 'A', 1, 0.1, null);
```

Luego crea y conecta el segundo nodo:

lista

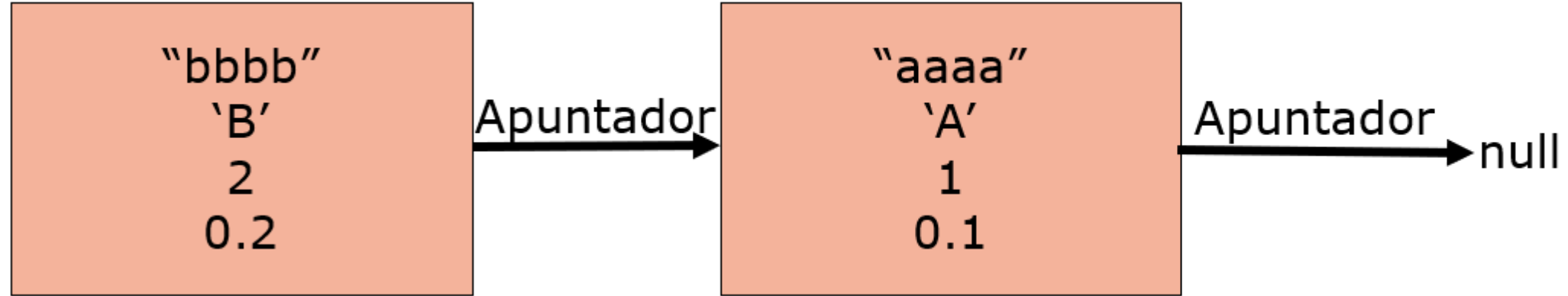


Ilustración 8: Crea el segundo Nodo y lo conecta con el anterior

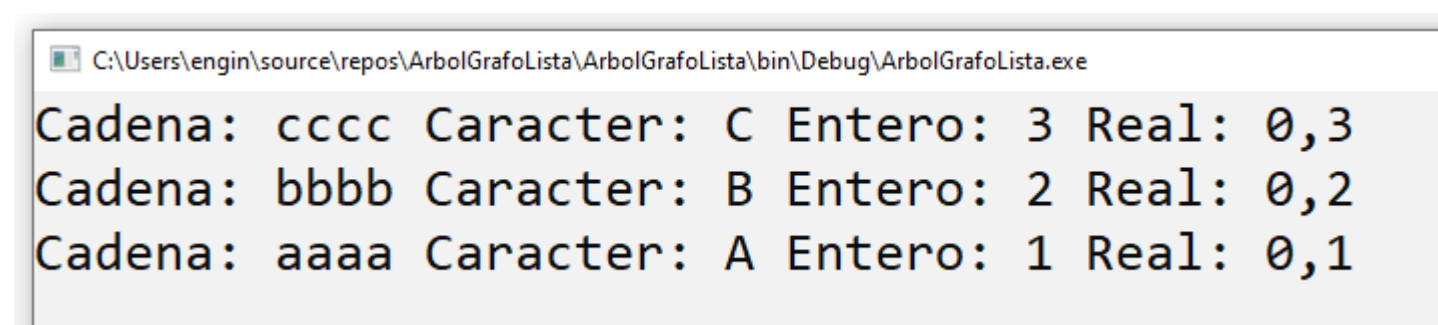
Con esta instrucción en C#:

```
lista = new Nodo("bbbb", 'B', 2, 0.2, lista);
```

```
using System;

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

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



```
C:\Users\engin\source\repos\ArbolGrafoLista\ArbolGrafoLista\bin\Debug\ArbolGrafoLista.exe
Cadena: cccc Character: C Entero: 3 Real: 0,3
Cadena: bbbb Character: B Entero: 2 Real: 0,2
Cadena: aaaa Character: A Entero: 1 Real: 0,1
```

Ilustración 9: Lista simplemente enlazada. Enlazamiento continuo.

Recorriéndola

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, ise estará borrando!

lista

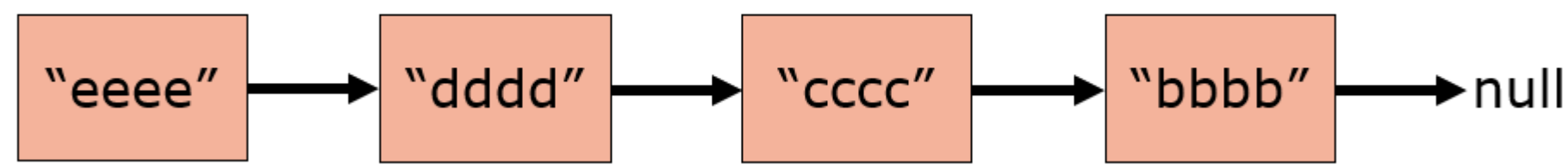


Ilustración 10: Variable "lista" sostiene la lista simplemente enlazada

Directorio 03. Nodo.cs

```
using System;

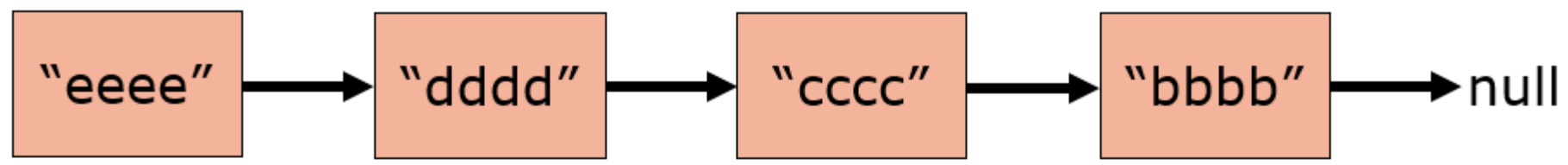
namespace ArbolGrafoLista {
    class Nodo {
        //Atributos propios
        public string Cadena { get; set; }
        public char Caracter { get; set; }
        public int Entero { get; set; }
        public double NumReal { get; set; }

        //Apuntador para lista simplemente enlazada
        public Nodo Apuntador;

        //Constructor
        public Nodo(string Cadena, char Caracter, int Entero, double NumReal, Nodo Apuntador) {
            this.Cadena = Cadena;
            this.Caracter = Caracter;
            this.Entero = Entero;
            this.NumReal = NumReal;
            this.Apuntador = Apuntador;
        }

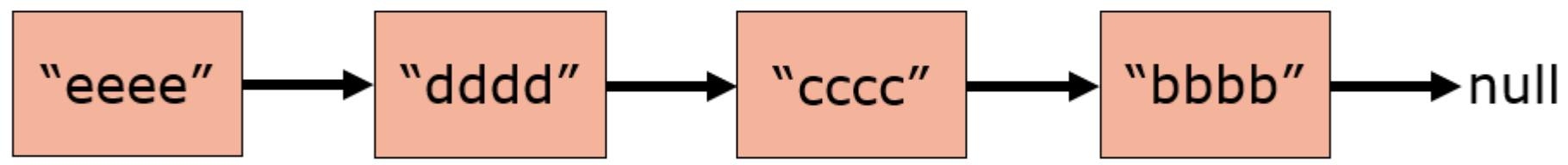
        //Imprime Contenido
        public void Imprime() {
            Console.WriteLine("Cadena: " + Cadena + " Caracter: " + Caracter.ToString());
            Console.WriteLine(" Entero: " + Entero.ToString() + " Real: " + NumReal.ToString());
        }
    }
}
```

pasea = lista



lista

pasea



lista

pasea

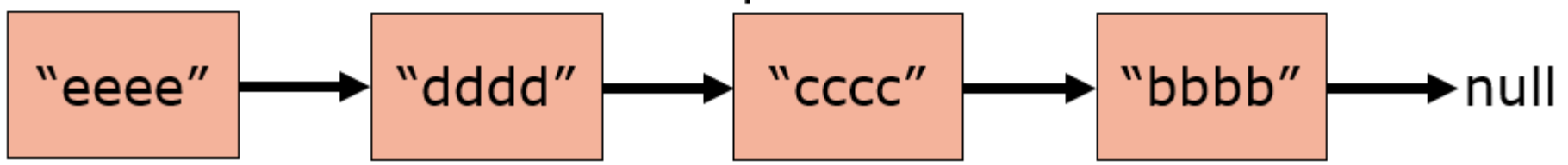


Ilustración 11: La variable "pasea" va de nodo en nodo. Con esta variable se muestra el contenido de cada nodo.

```

using System;

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

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

            Console.ReadKey();
        }
    }
}

```

```

C:\Users\engin\source\repos\ArbolGrafoLista\ArbolGrafoLista\bin\Debug\ArbolGrafoLista.exe
Cadena: iiii Character: I Entero: 9 Real: 0,9
Cadena: hhhh Character: H Entero: 8 Real: 0,8
Cadena: gggg Character: G Entero: 7 Real: 0,7
Cadena: ffff Character: F Entero: 6 Real: 0,6
Cadena: eeee Character: E Entero: 5 Real: 0,5
Cadena: dddd Character: D Entero: 4 Real: 0,4
Cadena: cccc Character: C Entero: 3 Real: 0,3
Cadena: bbbb Character: B Entero: 2 Real: 0,2
Cadena: aaaa Character: A Entero: 1 Real: 0,1

```

Ilustración 12: Recorrer una lista simplemente enlazada

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.

Directorio 04. Nodo.cs

```
using System;

namespace ArbolGrafoLista {
    class Nodo {
        //Atributos propios
        public string Cadena { get; set; }
        public char Caracter { get; set; }
        public int Entero { get; set; }
        public double NumReal { get; set; }

        //Apuntador para lista simplemente enlazada
        public Nodo Apuntador;

        //Constructor
        public Nodo(string Cadena, char Caracter, int Entero, double NumReal, Nodo Apuntador) {
            this.Cadena = Cadena;
            this.Caracter = Caracter;
            this.Entero = Entero;
            this.NumReal = NumReal;
            this.Apuntador = Apuntador;
        }

        //Imprime Contenido
        public void Imprime() {
            Console.WriteLine("Cadena: " + Cadena + " Caracter: " + Caracter.ToString());
            Console.WriteLine(" Entero: " + Entero.ToString() + " Real: " + NumReal.ToString());
        }
    }
}
```

Directorio 04. Program.cs

```
using System;

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

            Console.ReadKey();
        }

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

RECORRE PRIMERA VEZ

Cadena: iiii Character: I Entero: 9 Real: 0,9
Cadena: hhhh Character: H Entero: 8 Real: 0,8
Cadena: gggg Character: G Entero: 7 Real: 0,7
Cadena: ffff Character: F Entero: 6 Real: 0,6
Cadena: eeee Character: E Entero: 5 Real: 0,5
Cadena: dddd Character: D Entero: 4 Real: 0,4
Cadena: cccc Character: C Entero: 3 Real: 0,3
Cadena: bbbb Character: B Entero: 2 Real: 0,2
Cadena: aaaa Character: A Entero: 1 Real: 0,1

RECORRE SEGUNDA VEZ

Cadena: iiii Character: I Entero: 9 Real: 0,9
Cadena: hhhh Character: H Entero: 8 Real: 0,8
Cadena: gggg Character: G Entero: 7 Real: 0,7
Cadena: ffff Character: F Entero: 6 Real: 0,6
Cadena: eeee Character: E Entero: 5 Real: 0,5
Cadena: dddd Character: D Entero: 4 Real: 0,4
Cadena: cccc Character: C Entero: 3 Real: 0,3
Cadena: bbbb Character: B Entero: 2 Real: 0,2
Cadena: aaaa Character: A Entero: 1 Real: 0,1

Ilustración 13: Recorre una lista simplemente enlazada usando una función

Tamaño

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

Directorio 05. Nodo.cs

```
using System;

namespace ArbolGrafoLista {
    class Nodo {
        //Atributos propios
        public string Cadena { get; set; }
        public char Caracter { get; set; }
        public int Entero { get; set; }
        public double NumReal { get; set; }

        //Apuntador para lista simplemente enlazada
        public Nodo Apuntador;

        //Constructor
        public Nodo(string Cadena, char Caracter, int Entero, double NumReal, Nodo Apuntador) {
            this.Cadena = Cadena;
            this.Caracter = Caracter;
            this.Entero = Entero;
            this.NumReal = NumReal;
            this.Apuntador = Apuntador;
        }

        //Imprime Contenido
        public void Imprime() {
            Console.Write("Cadena: " + Cadena + " Caracter: " + Caracter.ToString());
            Console.WriteLine(" Entero: " + Entero.ToString() + " Real: " + NumReal.ToString());
        }
    }
}
```



```

using System;

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

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

            Console.ReadKey();
        }

        //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;
        }
    }
}

```

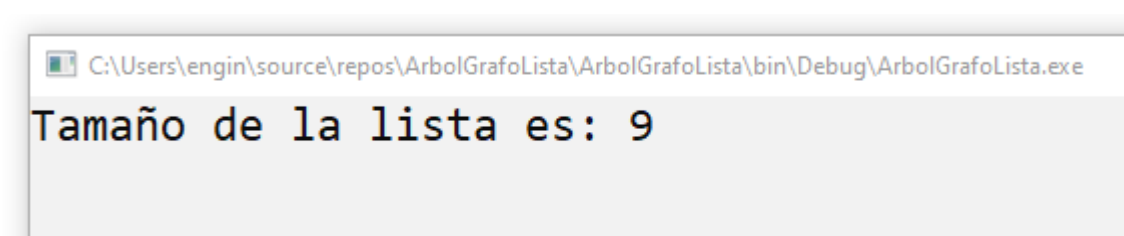


Ilustración 14: Tamaño de la lista simplemente enlazada

```

using System;

namespace ArbolGrafoLista {
    class Nodo {
        //Atributos propios
        public string Cadena { get; set; }
        public char Caracter { get; set; }
        public int Entero { get; set; }
        public double NumReal { get; set; }

        //Apuntador para lista simplemente enlazada
        public Nodo Apuntador;

        //Constructor
        public Nodo(string Cadena, char Caracter, int Entero, double NumReal, Nodo Apuntador) {
            this.Cadena = Cadena;
            this.Caracter = Caracter;
            this.Entero = Entero;
            this.NumReal = NumReal;
            this.Apuntador = Apuntador;
        }

        //Imprime Contenido
        public void Imprime() {
            Console.WriteLine("Cadena: " + Cadena + " Caracter: " + Caracter.ToString());
            Console.WriteLine(" Entero: " + Entero.ToString() + " Real: " + NumReal.ToString());
        }
    }
}

```

```

using System;

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

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

            Console.ReadKey();
        }

        //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;
        }
    }
}

```

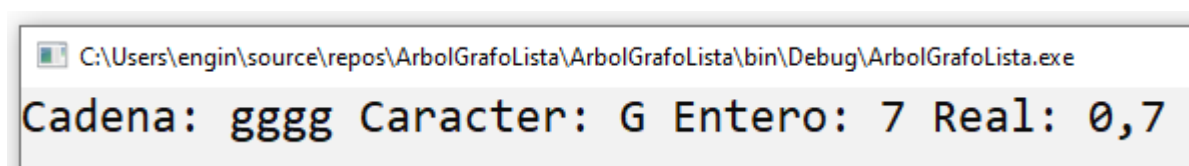
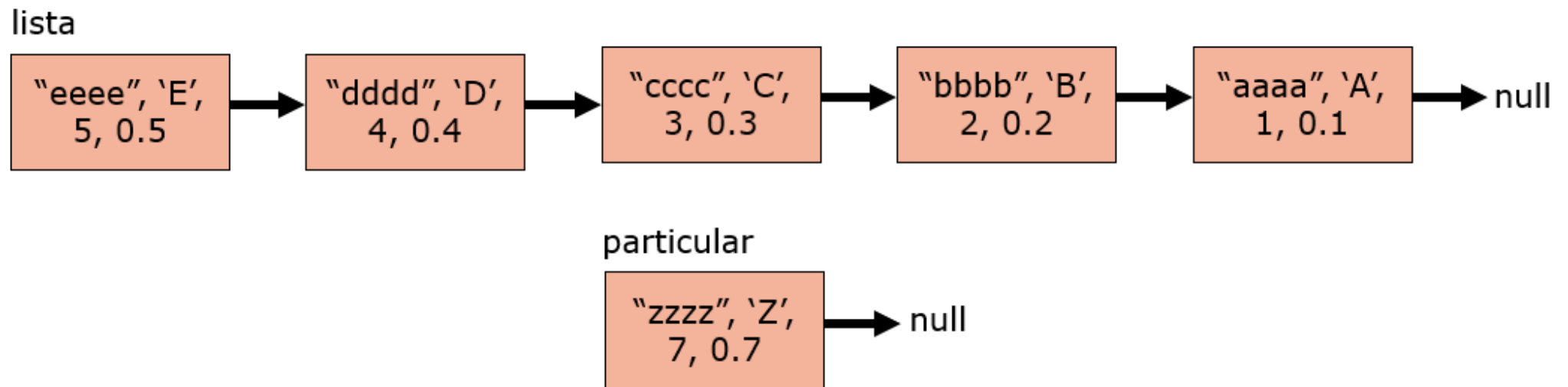


Ilustración 15: Trae un nodo en particular

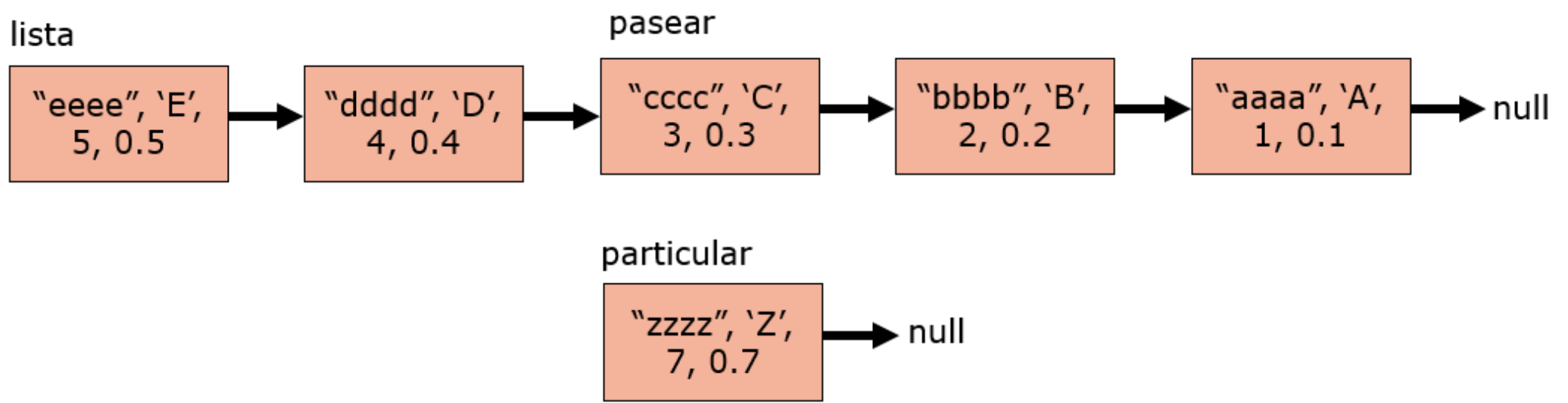
Adicionar un nodo en determinada posición

Para adicionar un nodo, se deben hacer varias operaciones:

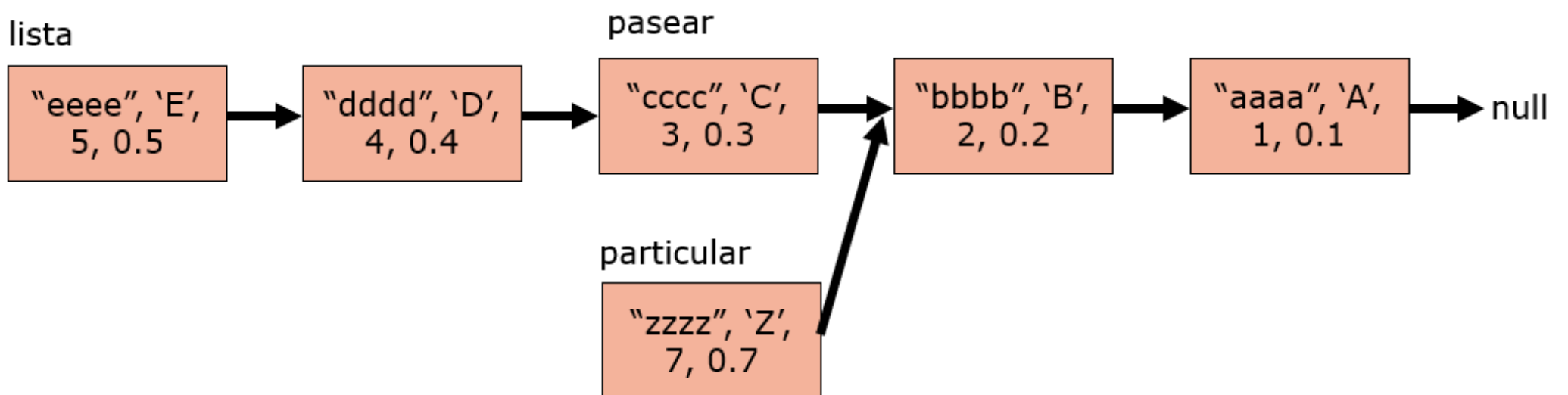
Paso 1: Lista existente y nodo nuevo a insertar

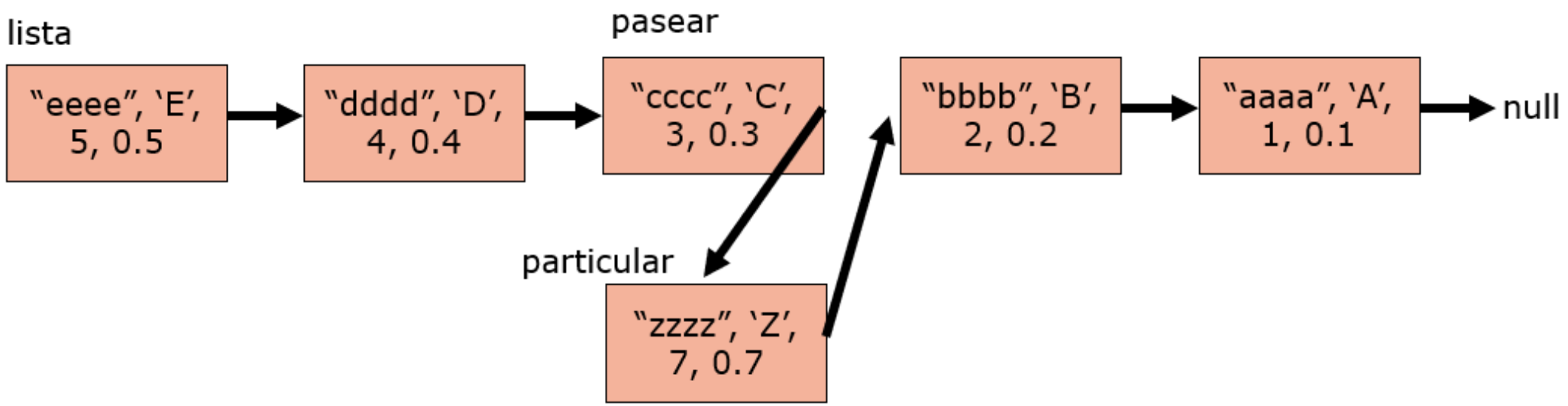


Paso 2: Ir hasta el punto de inserción



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





```
using System;

namespace ArbolGrafoLista {
    class Nodo {
        //Atributos propios
        public string Cadena { get; set; }
        public char Caracter { get; set; }
        public int Entero { get; set; }
        public double NumReal { get; set; }

        //Apuntador para lista simplemente enlazada
        public Nodo Apuntador;

        //Constructor
        public Nodo(string Cadena, char Caracter, int Entero, double NumReal, Nodo Apuntador) {
            this.Cadena = Cadena;
            this.Caracter = Caracter;
            this.Entero = Entero;
            this.NumReal = NumReal;
            this.Apuntador = Apuntador;
        }

        //Imprime Contenido
        public void Imprime() {
            Console.WriteLine("Cadena: " + Cadena + " Caracter: " + Caracter.ToString());
            Console.WriteLine(" Entero: " + Entero.ToString() + " Real: " + NumReal.ToString());
        }
    }
}
```

```

using System;

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

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

            Console.ReadKey();
        }

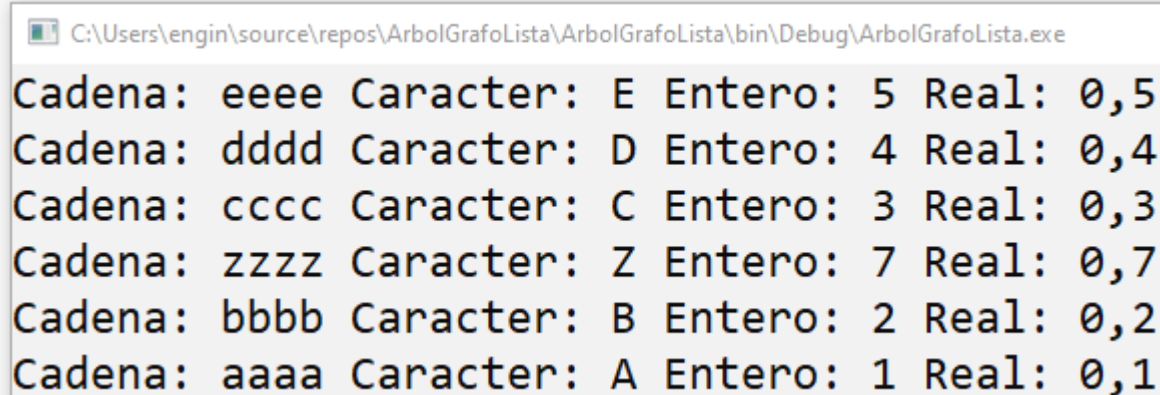
        //Adiciona un nodo en determinada posición
        static public Nodo AdicionaNodo(Nodo nodo, Nodo lista, int posicion) {
            //Si es al inicio de la lista
            if (posicion == 0) {
                nodo.Apuntador = lista;
                return nodo;
            }

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

```



```

C:\Users\engin\source\repos\ArbolGrafoLista\ArbolGrafoLista\bin\Debug\ArbolGrafoLista.exe
Cadena: eeee Character: E Entero: 5 Real: 0,5
Cadena: dddd Character: D Entero: 4 Real: 0,4
Cadena: cccc Character: C Entero: 3 Real: 0,3
Cadena: zzzz Character: Z Entero: 7 Real: 0,7
Cadena: bbbb Character: B Entero: 2 Real: 0,2
Cadena: aaaa Character: A Entero: 1 Real: 0,1

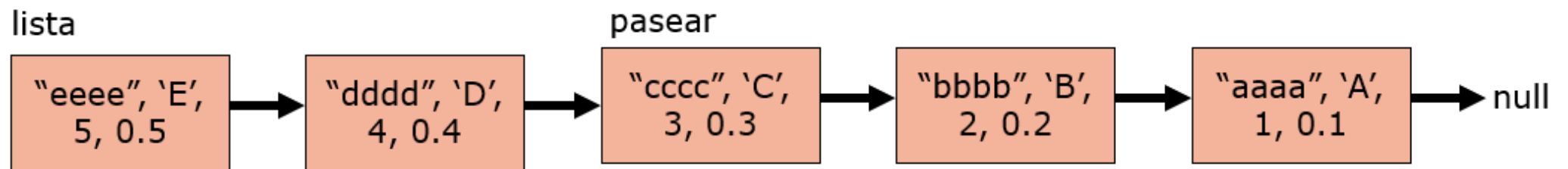
```

Ilustración 16: Adicionar un nodo a la lista

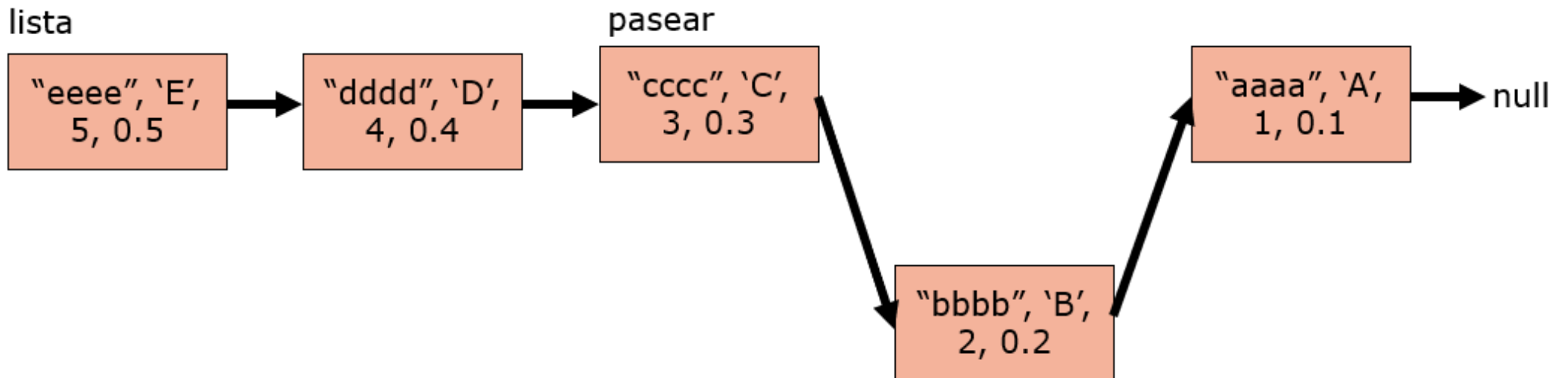
Borrar un nodo de una determinada posición

Para eliminar un nodo, se deben hacer varias operaciones:

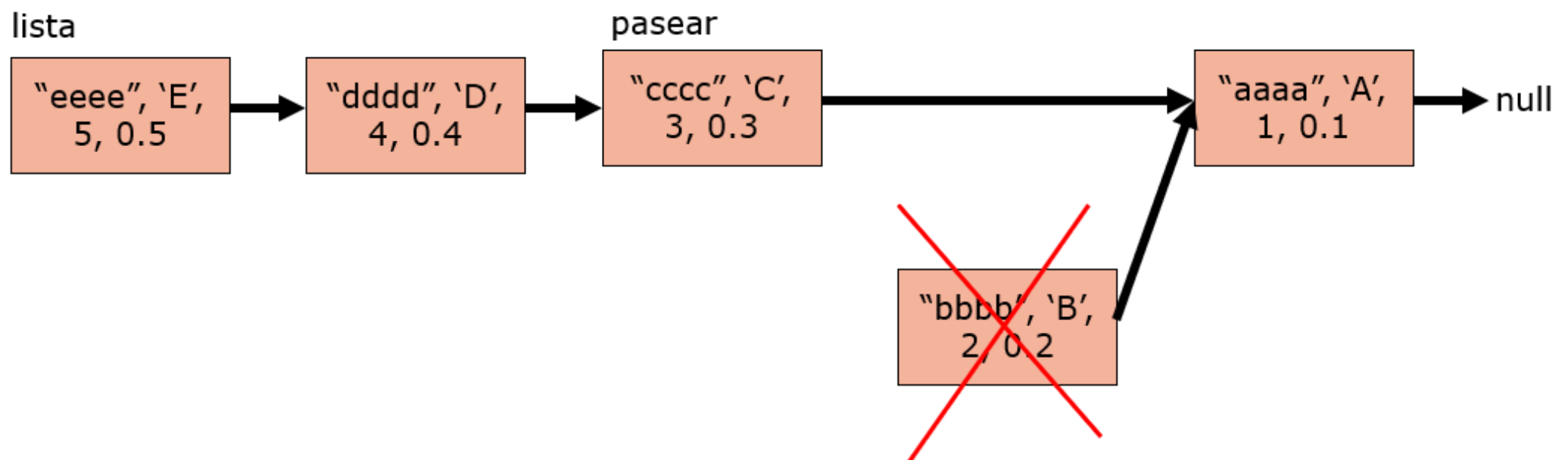
Paso 1: Lista existente e ir hasta el nodo al que se desee eliminar



Paso 2: El nodo que se va a eliminar



Paso 3: Se cambia el apuntador del nodo anterior al siguiente. El nodo sin conexión se borra automáticamente.



```
using System;

namespace ArbolGrafoLista {
    class Nodo {
        //Atributos propios
        public string Cadena { get; set; }
        public char Caracter { get; set; }
        public int Entero { get; set; }
        public double NumReal { get; set; }

        //Apuntador para lista simplemente enlazada
        public Nodo Apuntador;

        //Constructor
        public Nodo(string Cadena, char Caracter, int Entero, double NumReal, Nodo Apuntador) {
            this.Cadena = Cadena;
            this.Caracter = Caracter;
            this.Entero = Entero;
            this.NumReal = NumReal;
            this.Apuntador = Apuntador;
        }

        //Imprime Contenido
        public void Imprime() {
            Console.WriteLine("Cadena: " + Cadena + " Caracter: " + Caracter.ToString());
            Console.WriteLine(" Entero: " + Entero.ToString() + " Real: " + NumReal.ToString());
        }
    }
}
```



```

using System;

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

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

            Console.ReadKey();
        }

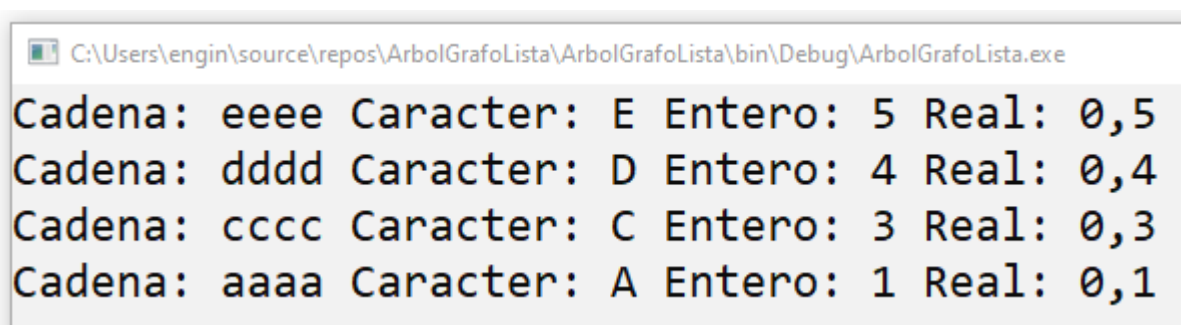
        //Borra nodo de una determinada posición
        static public Nodo BorraNodo(Nodo lista, int posicion) {
            //Si es al inicio de la lista
            if (posicion == 0) {
                lista = lista.Apuntador;
                return lista;
            }

            //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;
            }
        }
    }
}

```



```

C:\Users\engin\source\repos\ArbolGrafoLista\ArbolGrafoLista\bin\Debug\ArbolGrafoLista.exe
Cadena: eeee Character: E Entero: 5 Real: 0,5
Cadena: dddd Character: D Entero: 4 Real: 0,4
Cadena: cccc Character: C Entero: 3 Real: 0,3
Cadena: aaaa Character: A Entero: 1 Real: 0,1

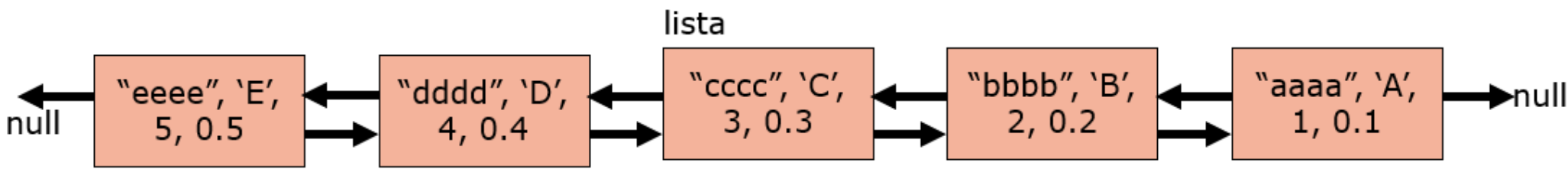
```

Ilustración 17: Eliminar nodo

La lista doblemente enlazada

Definición

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.



Directorio 09. Nodo.cs

```
using System;

namespace ArbolGrafoLista {
    class Nodo {
        //Atributos propios
        public string Cadena { get; set; }
        public char Caracter { get; set; }
        public int Entero { get; set; }
        public double NumReal { get; set; }

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

        //Constructor
        public Nodo(string Cadena, char Caracter, int Entero, double NumReal, Nodo NodoDer) {
            this.Cadena = Cadena;
            this.Caracter = Caracter;
            this.Entero = Entero;
            this.NumReal = NumReal;
            NodoIzq = null;
            this.NodoDer = NodoDer;
            if (NodoDer != null) NodoDer.NodoIzq = this;
        }

        //Imprime Contenido
        public void Imprime() {
            Console.WriteLine("Cadena: " + Cadena + " Caracter: " + Caracter.ToString());
            Console.WriteLine(" Entero: " + Entero.ToString() + " Real: " + NumReal.ToString());
        }
    }
}
```

```

using System;

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

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

        //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;
            }
        }
    }
}

```

```
C:\Users\engin\source\repos\ArbolGrafoLista\ArbolGrafoLista\bin\Debug\ArbolGrafoLista.exe

De izquierda a derecha
Cadena: eeee Caracter: E Entero: 5 Real: 0,5
Cadena: dddd Caracter: D Entero: 4 Real: 0,4
Cadena: cccc Caracter: C Entero: 3 Real: 0,3
Cadena: bbbb Caracter: B Entero: 2 Real: 0,2
Cadena: aaaa Caracter: A Entero: 1 Real: 0,1

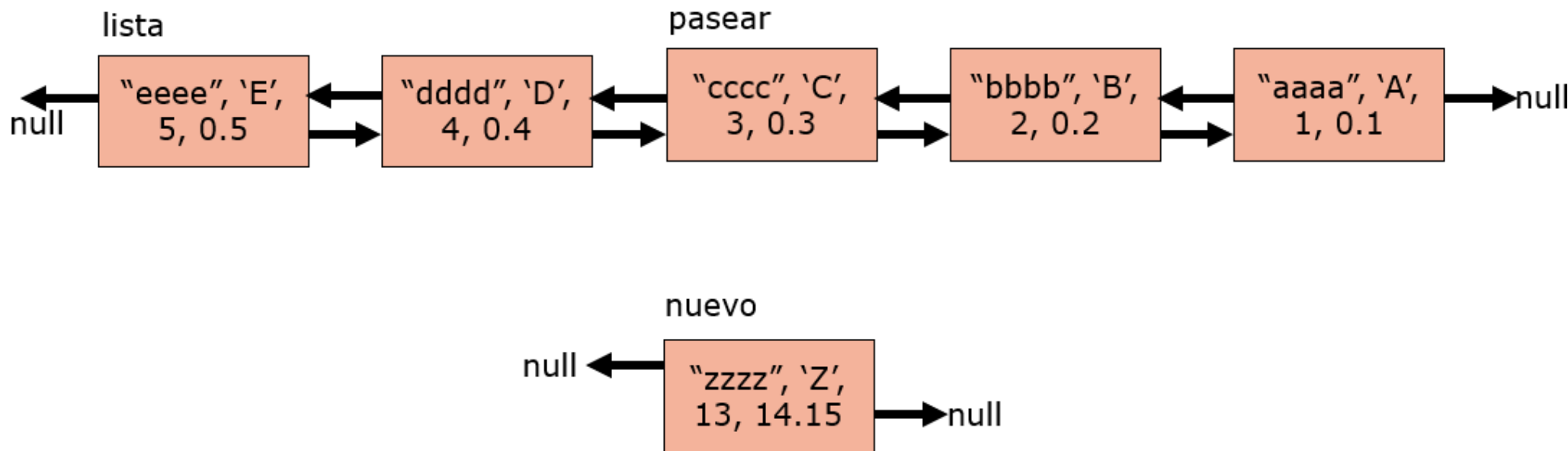
De derecha a izquierda
Cadena: aaaa Caracter: A Entero: 1 Real: 0,1
Cadena: bbbb Caracter: B Entero: 2 Real: 0,2
Cadena: cccc Caracter: C Entero: 3 Real: 0,3
Cadena: dddd Caracter: D Entero: 4 Real: 0,4
Cadena: eeee Caracter: E Entero: 5 Real: 0,5
```

Ilustración 18: Lista doblemente enlazada. Recorrido en ambos sentidos.

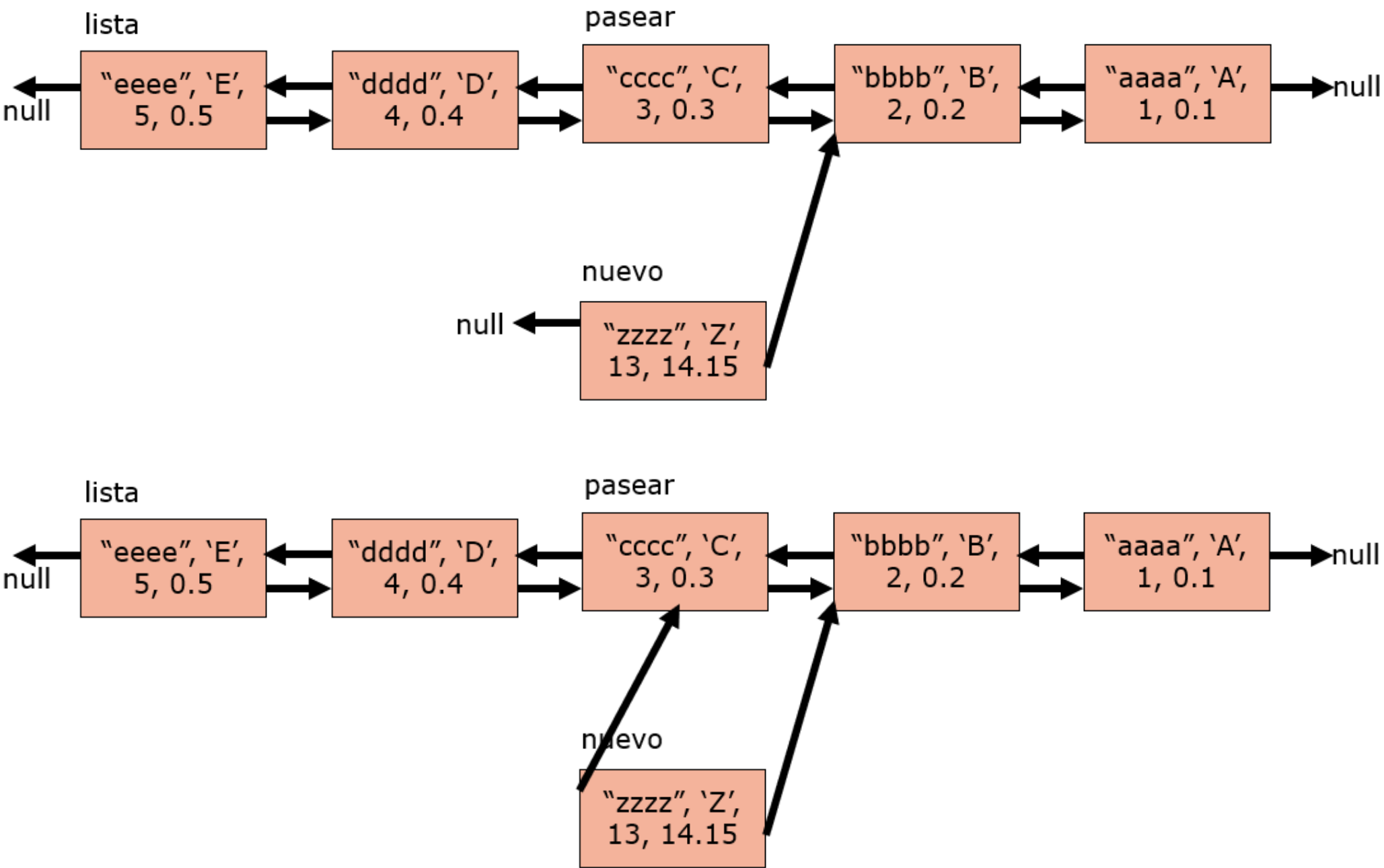
Adicionar un nodo en determinada posición

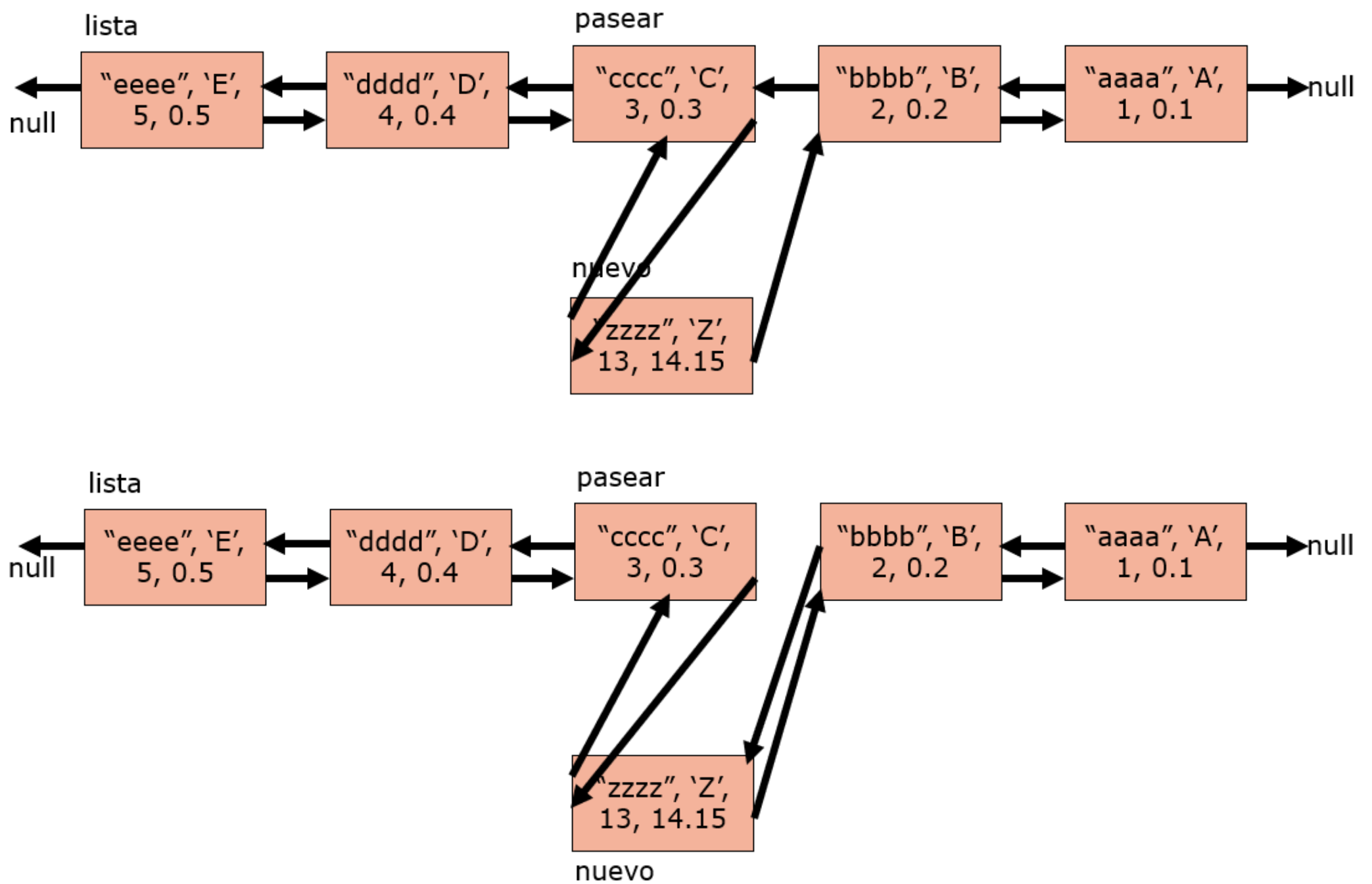
Para adicionar un nodo, se deben hacer varias operaciones:

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



Paso 2: Poner los enlaces





Directorio 10. Nodo.cs

```
using System;

namespace ArbolGrafoLista {
    class Nodo {
        //Atributos propios
        public string Cadena { get; set; }
        public char Caracter { get; set; }
        public int Entero { get; set; }
        public double NumReal { get; set; }

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

        //Constructor
        public Nodo(string Cadena, char Caracter, int Entero, double NumReal, Nodo NodoDer) {
            this.Cadena = Cadena;
            this.Caracter = Caracter;
            this.Entero = Entero;
            this.NumReal = NumReal;
            NodoIzq = null;
            this.NodoDer = NodoDer;
            if (NodoDer != null) NodoDer.NodoIzq = this;
        }

        //Imprime Contenido
        public void Imprime() {
            Console.WriteLine("Cadena: " + Cadena + " Caracter: " + Caracter.ToString());
            Console.WriteLine(" Entero: " + Entero.ToString() + " Real: " + NumReal.ToString());
        }
    }
}
```

```

using System;

namespace ArbolGrafoLista {
    class Program {
        static void Main() {
            //Crea la lista
            Nodo lista = new Nodo("aaaa", 'A', 1, 0.1, null);
            lista = new Nodo("bbbb", 'B', 2, 0.2, lista);
            lista = new Nodo("cccc", 'C', 3, 0.3, lista);
            lista = new Nodo("dddd", 'D', 4, 0.4, lista);
            lista = new Nodo("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 Nodo("zzzz", 'Z', 13, 14.15, null);
            lista = AgregaNodo(nuevo, lista, 3);

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

            Console.ReadKey();
        }

        //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;
    }
}
}
}
}

```

```

C:\Users\engin\source\repos\ArbolGrafoLista\ArbolGrafoLista\bin\Debug\ArbolGrafoLista.exe
De izquierda a derecha
Cadena: eeee Character: E Entero: 5 Real: 0,5
Cadena: dddd Character: D Entero: 4 Real: 0,4
Cadena: cccc Character: C Entero: 3 Real: 0,3
Cadena: bbbb Character: B Entero: 2 Real: 0,2
Cadena: aaaa Character: A Entero: 1 Real: 0,1

De derecha a izquierda
Cadena: aaaa Character: A Entero: 1 Real: 0,1
Cadena: bbbb Character: B Entero: 2 Real: 0,2
Cadena: cccc Character: C Entero: 3 Real: 0,3
Cadena: dddd Character: D Entero: 4 Real: 0,4
Cadena: eeee Character: E Entero: 5 Real: 0,5

De izquierda a derecha
Cadena: eeee Character: E Entero: 5 Real: 0,5
Cadena: dddd Character: D Entero: 4 Real: 0,4
Cadena: cccc Character: C Entero: 3 Real: 0,3
Cadena: zzzz Character: Z Entero: 13 Real: 14,15
Cadena: bbbb Character: B Entero: 2 Real: 0,2
Cadena: aaaa Character: A Entero: 1 Real: 0,1

De derecha a izquierda
Cadena: aaaa Character: A Entero: 1 Real: 0,1
Cadena: bbbb Character: B Entero: 2 Real: 0,2
Cadena: zzzz Character: Z Entero: 13 Real: 14,15
Cadena: cccc Character: C Entero: 3 Real: 0,3
Cadena: dddd Character: D Entero: 4 Real: 0,4
Cadena: eeee Character: E Entero: 5 Real: 0,5

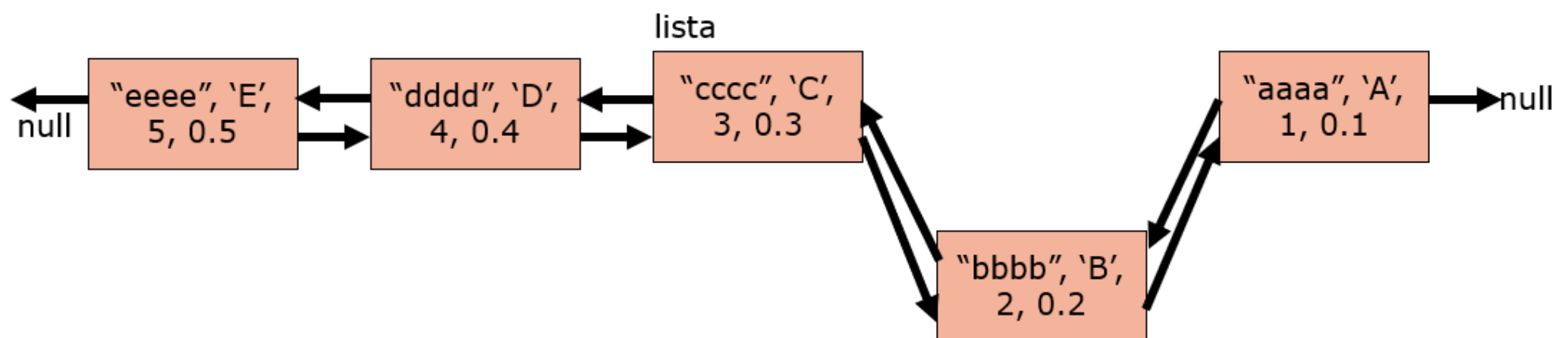
```

Ilustración 19: Adicionar un nodo en una lista doblemente enlazada

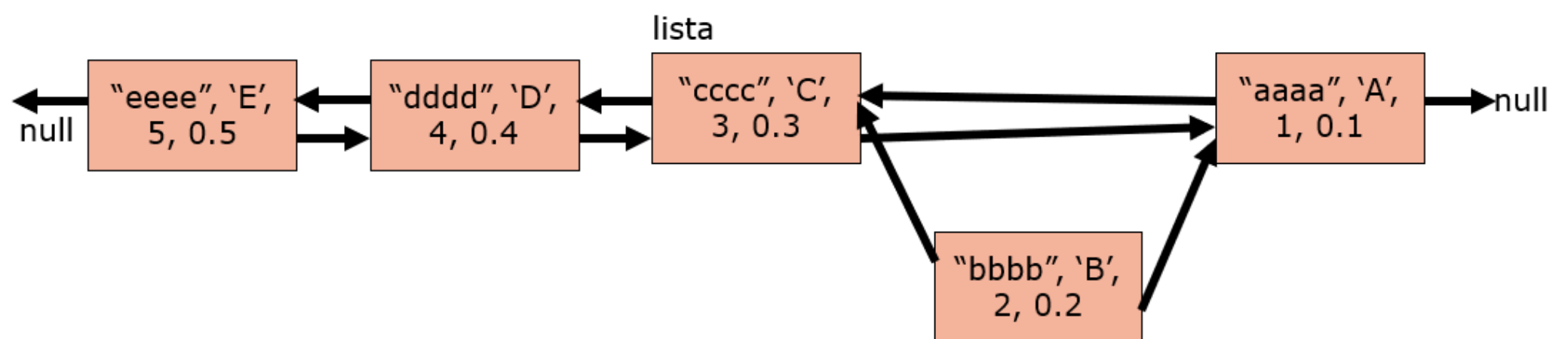
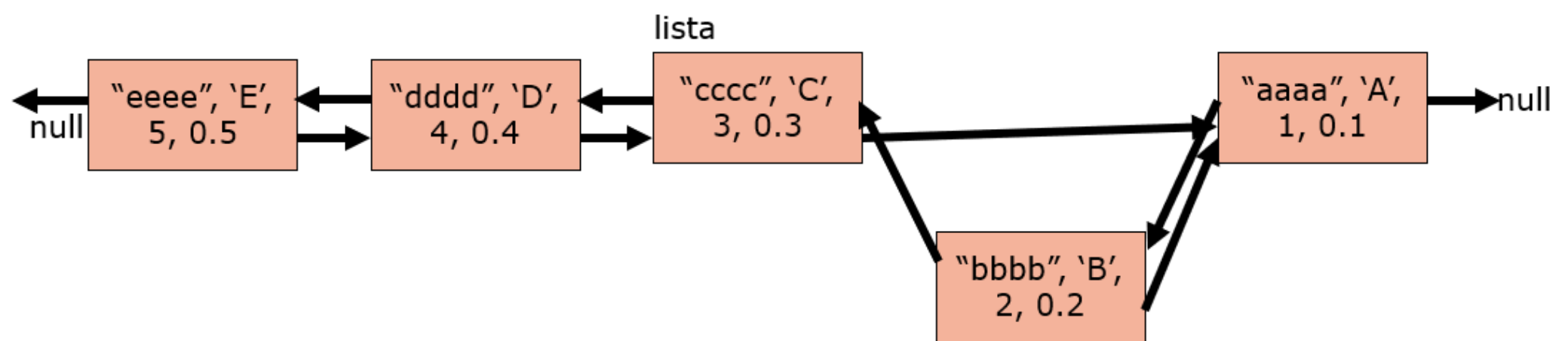
Borrar un nodo de una determinada posición

Para eliminar un nodo, se deben hacer varias operaciones:

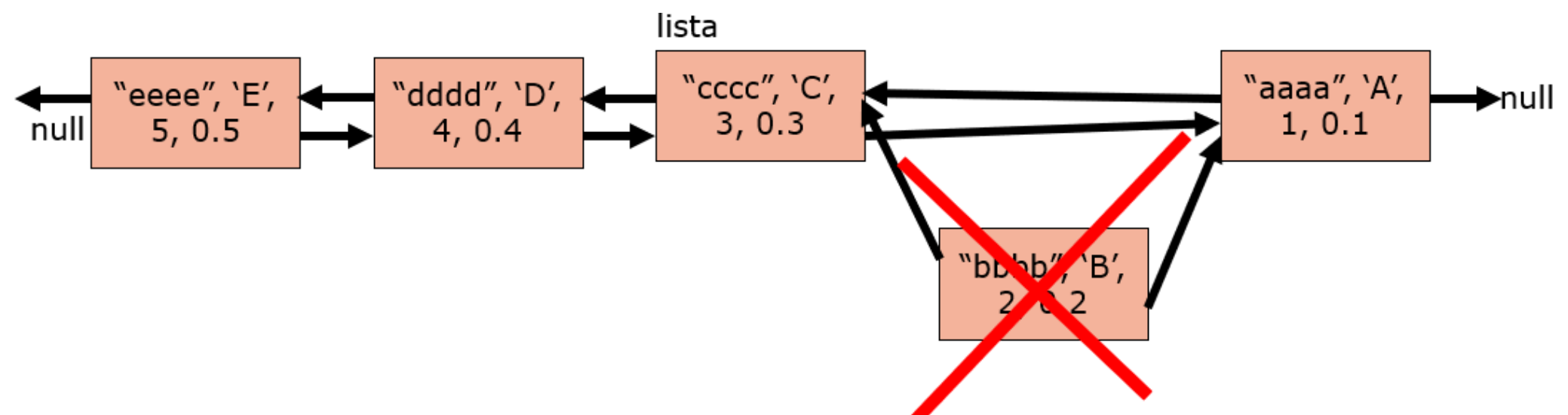
Paso 1: Lista existente e ir hasta el nodo al que se desee eliminar



Paso 2: Modificar los apuntadores



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



```

using System;

namespace ArbolGrafoLista {
    class Nodo {
        //Atributos propios
        public string Cadena { get; set; }
        public char Caracter { get; set; }
        public int Entero { get; set; }
        public double NumReal { get; set; }

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

        //Constructor
        public Nodo(string Cadena, char Caracter, int Entero, double NumReal, Nodo NodoDer) {
            this.Cadena = Cadena;
            this.Caracter = Caracter;
            this.Entero = Entero;
            this.NumReal = NumReal;
            NodoIzq = null;
            this.NodoDer = NodoDer;
            if (NodoDer != null) NodoDer.NodoIzq = this;
        }

        //Imprime Contenido
        public void Imprime() {
            Console.WriteLine("Cadena: " + Cadena + " Caracter: " + Caracter.ToString());
            Console.WriteLine(" Entero: " + Entero.ToString() + " Real: " + NumReal.ToString());
        }
    }
}

```

```

using System;

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

            Console.ReadKey();
        }

        //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;
        }
    }
}

```

```
C:\Users\engin\source\repos\ArbolGrafoLista\ArbolGrafoLista\bin\Debug\ArbolGrafoLista.exe

De izquierda a derecha
Cadena: eeee Character: E Entero: 5 Real: 0,5
Cadena: dddd Character: D Entero: 4 Real: 0,4
Cadena: cccc Character: C Entero: 3 Real: 0,3
Cadena: bbbb Character: B Entero: 2 Real: 0,2
Cadena: aaaa Character: A Entero: 1 Real: 0,1

De derecha a izquierda
Cadena: aaaa Character: A Entero: 1 Real: 0,1
Cadena: bbbb Character: B Entero: 2 Real: 0,2
Cadena: cccc Character: C Entero: 3 Real: 0,3
Cadena: dddd Character: D Entero: 4 Real: 0,4
Cadena: eeee Character: E Entero: 5 Real: 0,5

De izquierda a derecha
Cadena: eeee Character: E Entero: 5 Real: 0,5
Cadena: dddd Character: D Entero: 4 Real: 0,4
Cadena: cccc Character: C Entero: 3 Real: 0,3
Cadena: aaaa Character: A Entero: 1 Real: 0,1

De derecha a izquierda
Cadena: aaaa Character: A Entero: 1 Real: 0,1
Cadena: cccc Character: C Entero: 3 Real: 0,3
Cadena: dddd Character: D Entero: 4 Real: 0,4
Cadena: eeee Character: E Entero: 5 Real: 0,5
```

Ilustración 20: Borrar un nodo de una lista doblemente enlazada

Árbol binario

Definición y recorridos recursivos

Primer ejemplo

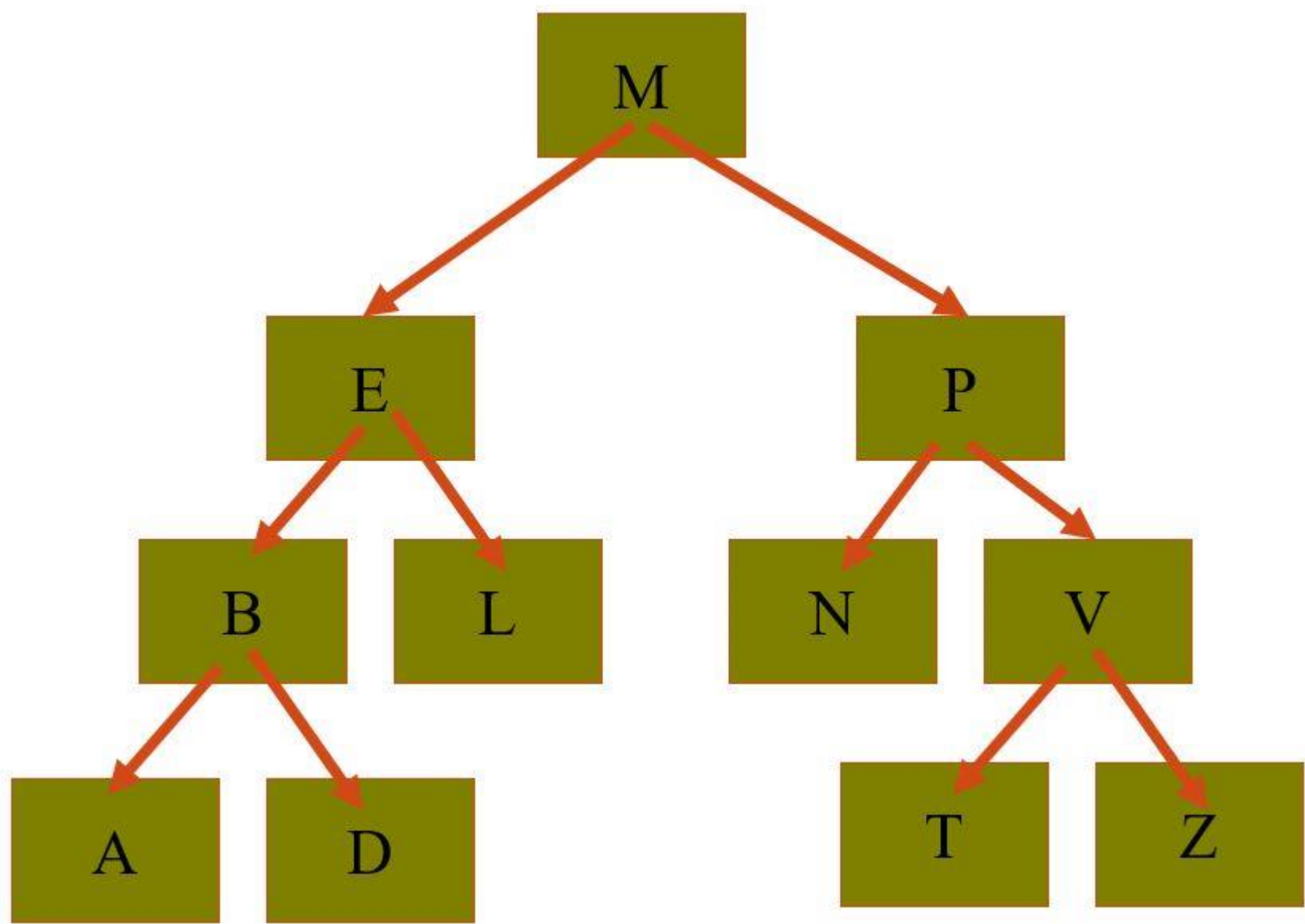


Ilustración 21: Ejemplo de un árbol binario

Directorio 12. Nodo.cs

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

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

```
//Recorrido de un árbol binario
using System;

namespace ArbolGrafoLista {
    class Program {

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

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

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

            Console.WriteLine("\n\nRecorrido postOrden (izquierdo, derecho, raiz)");
            PostOrden(Arbol);

            Console.ReadKey();
        }

        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 + ", ");
            }
        }
    }
}
```

```
C:\Users\engin\source\repos\ArbolGrafoLista\ArbolGrafoLista\bin\Debug\ArbolGrafoLista.exe

Recorrido preOrden (raiz, izquierdo, derecho)
M, E, B, A, D, L, P, N, V, T, Z,

Recorrido inOrden (izquierdo, raiz, derecho)
A, B, D, E, L, M, N, P, T, V, Z,

Recorrido postOrden (izquierdo, derecho, raiz)
A, D, B, L, E, N, T, Z, V, P, M,
```

Ilustración 22: Creación y recorrido de un árbol binario

Segundo ejemplo

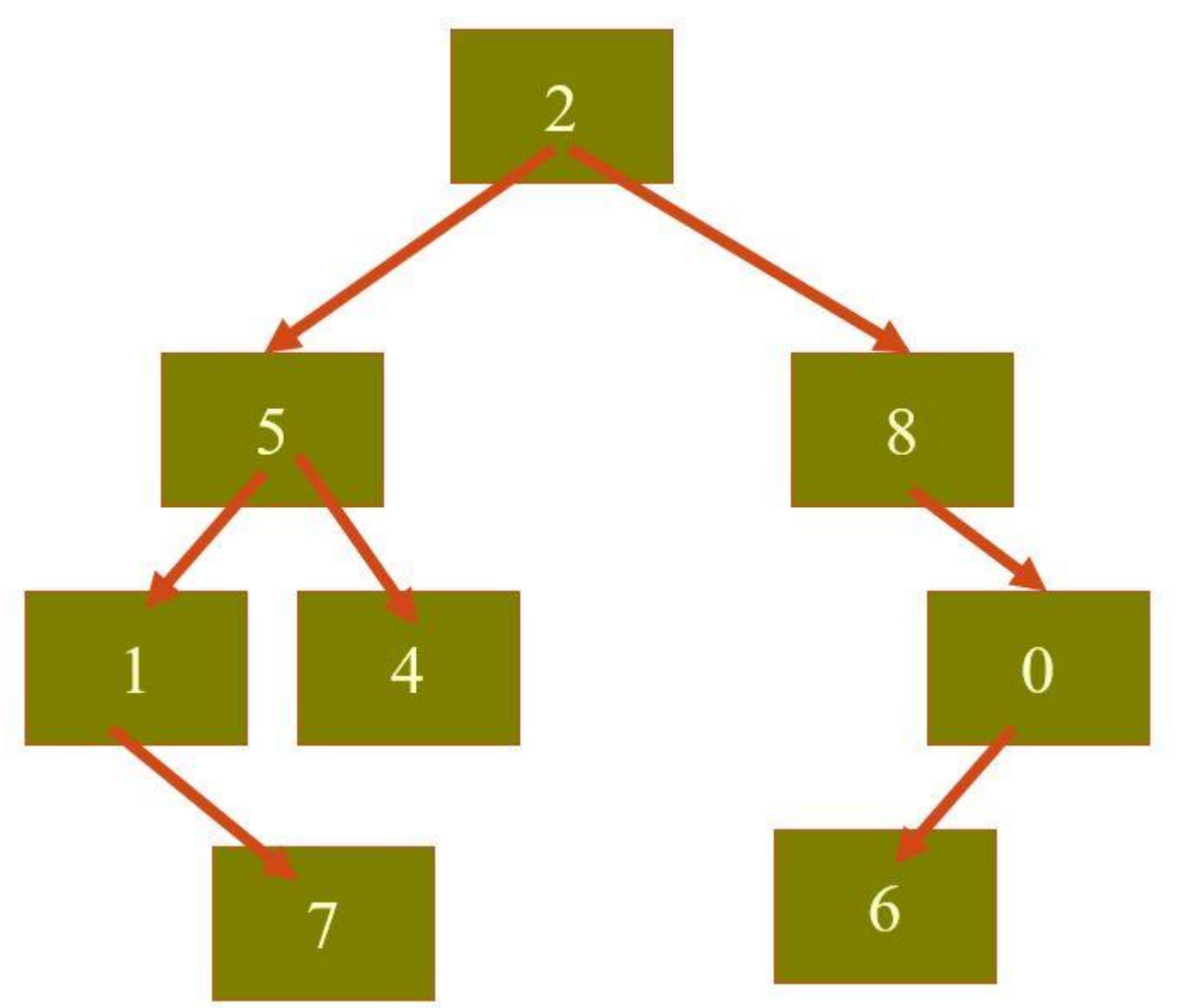


Ilustración 23: Ejemplo de árbol binario

Directorio 13. Nodo.cs

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

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



```
//Recorridos de un árbol binario
using System;

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

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

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

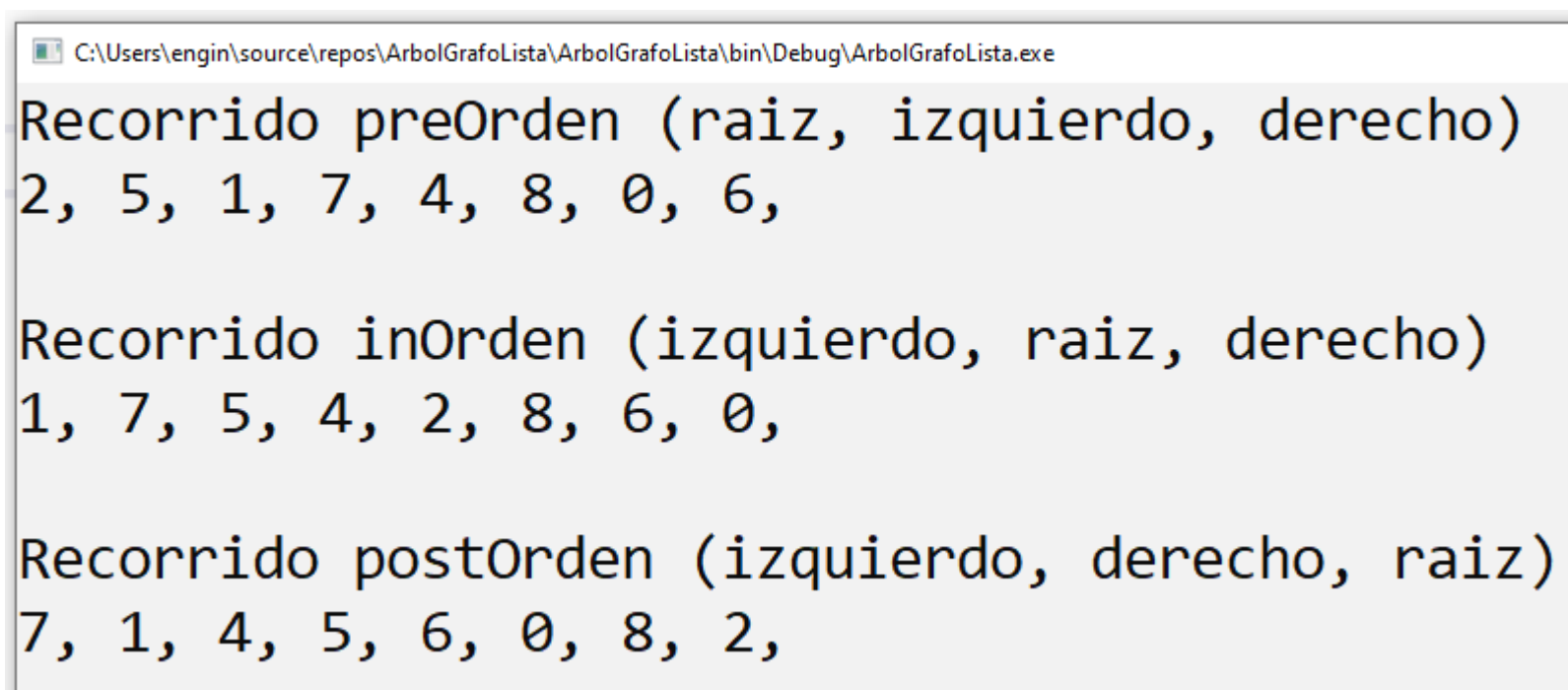
            Console.WriteLine("\n\nRecorrido postOrden (izquierdo, derecho, raiz)");
            PostOrden(Arbol);

            Console.ReadKey();
        }

        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 + ", ");
            }
        }
    }
}
```



```
C:\Users\engin\source\repos\ArbolGrafoLista\ArbolGrafoLista\bin\Debug\ArbolGrafoLista.exe

Recorrido preOrden (raiz, izquierdo, derecho)
2, 5, 1, 7, 4, 8, 0, 6,

Recorrido inOrden (izquierdo, raiz, derecho)
1, 7, 5, 4, 2, 8, 6, 0,

Recorrido postOrden (izquierdo, derecho, raiz)
7, 1, 4, 5, 6, 0, 8, 2,
```

Ilustración 24: Árbol binario y sus recorridos

Recorrido iterativo (no recursivo)

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

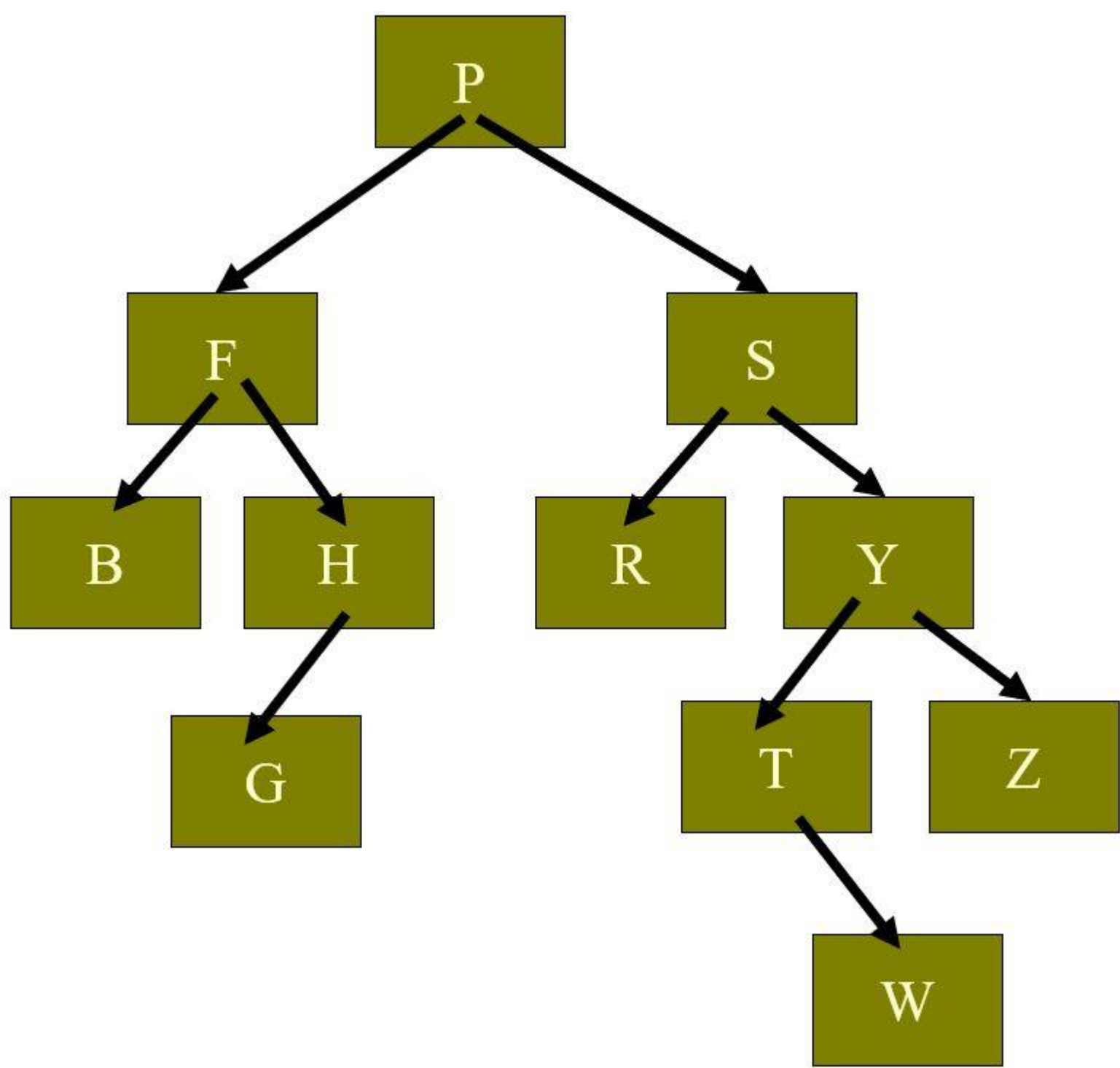


Ilustración 25: Árbol binario

Directorio 14. Nodo.cs

```
//Nodo de un árbol binario
namespace ArbolGrafoLista {
    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
namespace ArbolGrafoLista {
    class NodoPila {
        public NodoPila Flecha;
        public Nodo Raiz;
        public NodoPila(Nodo Raiz, NodoPila Flecha) {
            this.Raiz = Raiz;
            this.Flecha = Flecha;
        }
    }
}
```

```
//Recorrido (no recursivo) de un árbol binario
using System;

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

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

            Console.ReadKey();
        }

        public static void Iterativo(Nodo arbol) {
            //Usa una pila para guardar
            NodoPila inicia = new NodoPila(arbol, null);
            do {
                Nodo tmp = inicia.Raiz; //Una variable tmp para ver el nodo del árbol
                Console.Write(tmp.Letra + ", "); //Imprime el valor del nodo del árbol
                inicia = inicia.Flecha; //Se quita un elemento de la pila
                if (tmp.Izquierda != null) inicia = new NodoPila(tmp.Izquierda, inicia); //Si el nodo
de árbol tiene un hijo a la izquierda entonces agrega este a la pila
                if (tmp.Derecha != null) inicia = new NodoPila(tmp.Derecha, inicia); //Si el nodo de
árbol tiene un hijo a la derecha entonces agrega este a la pila
            } while (inicia != null); //Hasta que se vacíe la pila
        }
    }
}
```

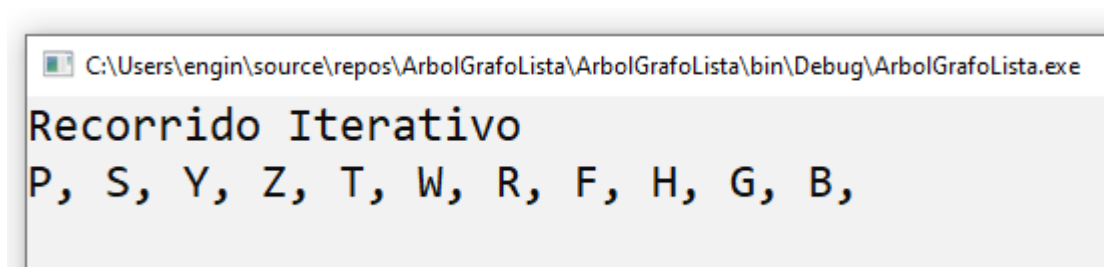


Ilustración 26: Recorrido iterativo de un árbol binario

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

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

```
//Crear un árbol binario al azar
using System;

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

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

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

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

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

            Console.ReadKey();
        }

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

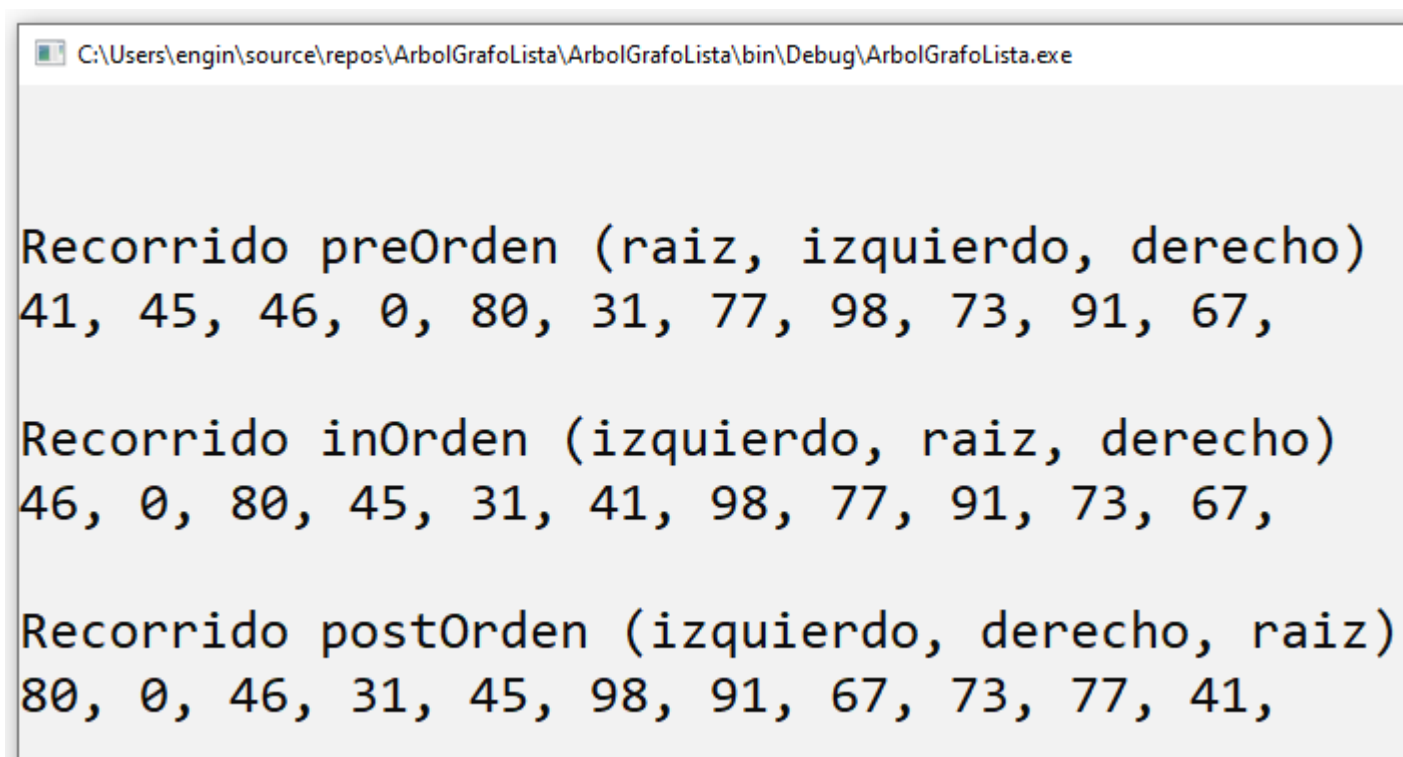
        static void inOrden(Nodo Arbol) {
            if (Arbol != null) {
                inOrden(Arbol.Izquierda);
                Console.Write(Arbol.Numero + ", ");
                inOrden(Arbol.Derecha);
            }
        }

        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.WriteLine(Arbol.Numero + ", ");
    }
}
}
}

```



```

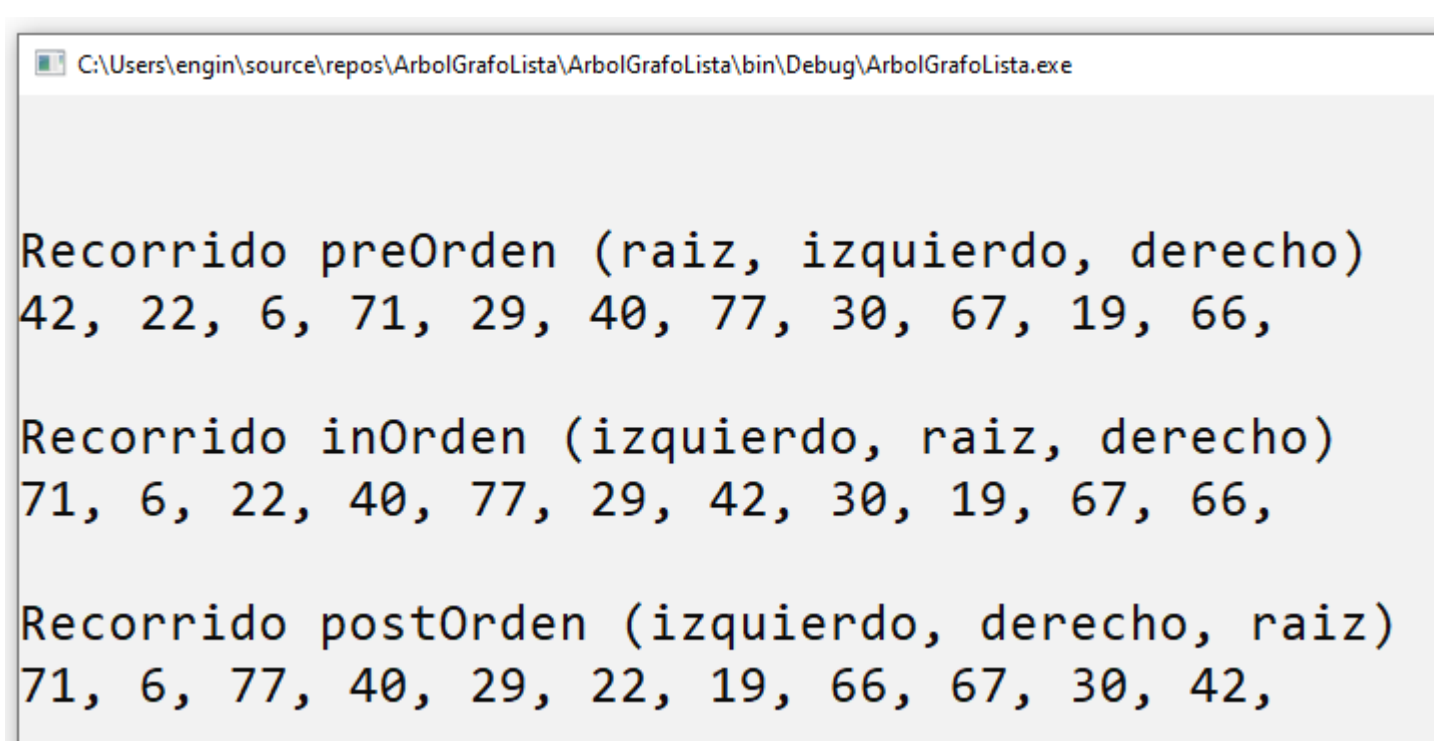
Recorrido preOrden (raiz, izquierdo, derecho)
41, 45, 46, 0, 80, 31, 77, 98, 73, 91, 67,

Recorrido inOrden (izquierdo, raiz, derecho)
46, 0, 80, 45, 31, 41, 98, 77, 91, 73, 67,

Recorrido postOrden (izquierdo, derecho, raiz)
80, 0, 46, 31, 45, 98, 91, 67, 73, 77, 41,

```

Ilustración 27: Ejemplo de recorrer un árbol binario generado al azar



```

Recorrido preOrden (raiz, izquierdo, derecho)
42, 22, 6, 71, 29, 40, 77, 30, 67, 19, 66,

Recorrido inOrden (izquierdo, raiz, derecho)
71, 6, 22, 40, 77, 29, 42, 30, 19, 67, 66,

Recorrido postOrden (izquierdo, derecho, raiz)
71, 6, 77, 40, 29, 22, 19, 66, 67, 30, 42,

```

Ilustración 28: Ejemplo de recorrer un árbol binario generado 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.

Directorio 16. Nodo.cs

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

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

Directorio 16. Program.cs

```
//Ordenar con un árbol binario
using System;

namespace ArbolGrafoLista {
    class Program {
        static void Main(string[] args) {
            //Va agregando nodo a nodo y los va ordenando
            Nodo Arbol = new Nodo(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\nRecorrido InOrden (izquierdo, raiz, derecho)");
            InOrden(Arbol);

            Console.ReadKey();
        }

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

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

```
C:\Users\engin\source\repos\ArbolGrafoLista\ArbolGrafoLista\bin\Debug\ArbolGrafoLista.exe

Recorrido InOrden (izquierdo, raiz, derecho)
-1,
2,
3,
5,
7,
9,
10,
15,
17,
18,
19,
27,
```

Ilustración 29: Ordenando con un árbol binario

```
namespace ArbolGrafoLista {
    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;
        }
    }
}
```

```
//Ordenar, buscar en árbol binario ordenado, número de nodos y altura del árbol
using System;

namespace ArbolGrafoLista {
    class Program {

        public static void Main() {

            Nodo Arbol = new Nodo(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\n\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));

            Console.ReadKey();
        }

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

```

```

        if (Raiz.Derecha == null)
            Raiz.Derecha = new Nodo(Valor);
        else
            AgregaNodo(Valor, Raiz.Derecha);
    }
}

//Recorrido del 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;
}
}
}

```

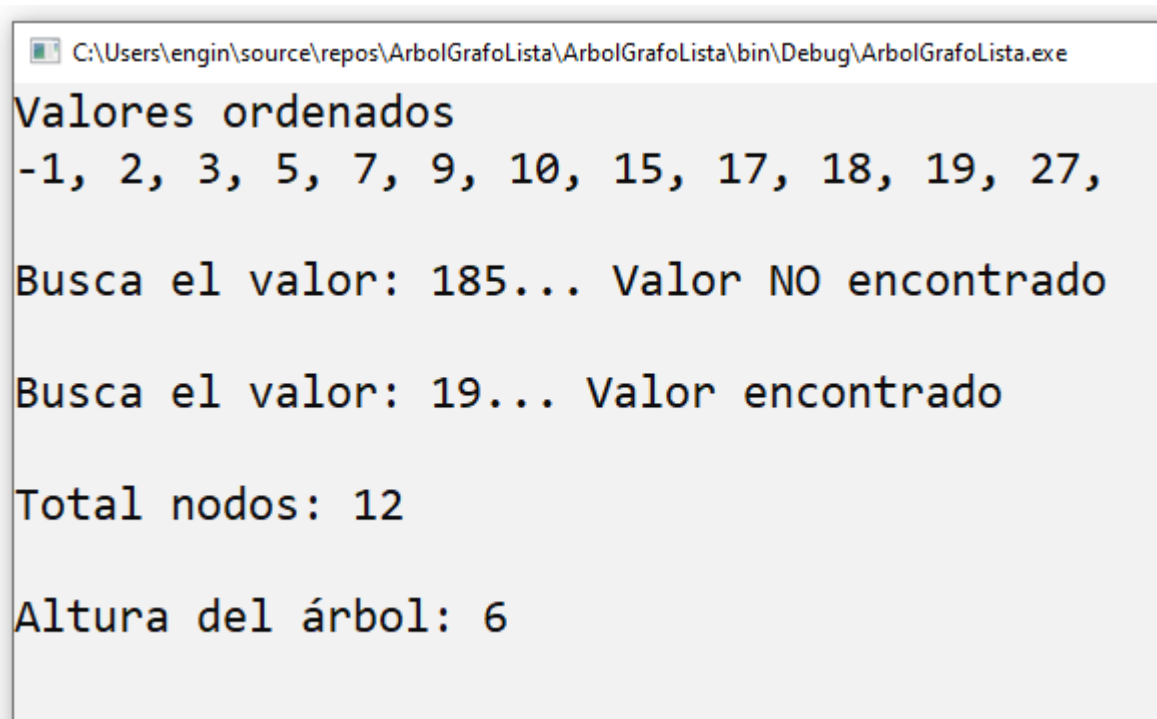


Ilustración 30: Varias operaciones en un árbol binario

Dibujar un árbol binario

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

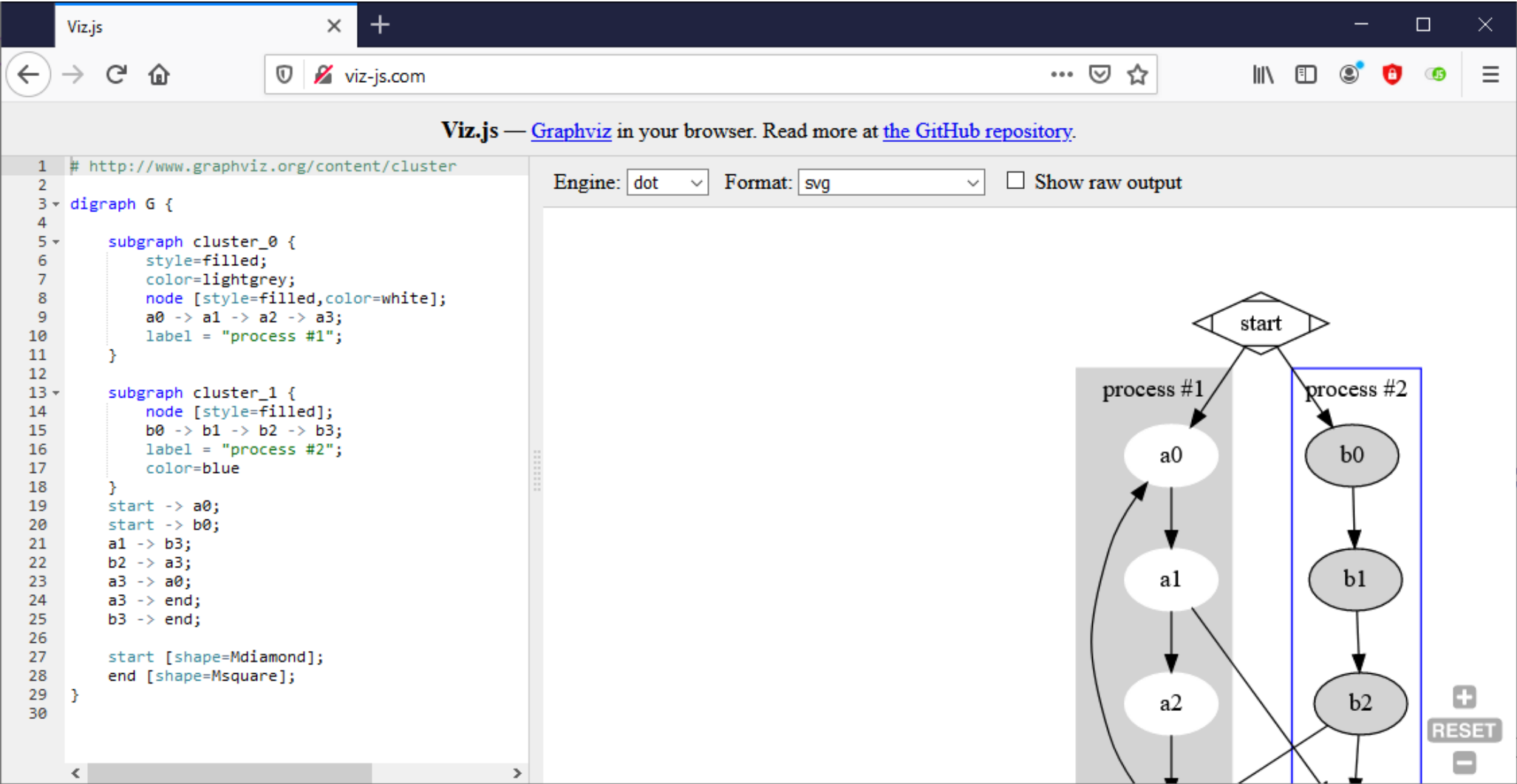


Ilustración 31: <http://viz-js.com/>

Este es el código para generar el árbol:

Directorio 18. Nodo.cs

```

namespace ArbolGrafoLista {
    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;
        }
    }
}

```

```
//Dibujar el árbol en http://viz-js.com/
using System;

namespace ArbolGrafoLista {
    class Program {

        public static void Main() {

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

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

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



```
C:\Users\engin\source\repos\ArbolGrafoLista\ArbolGrafoLista\bin\Debug\ArbolGrafoLista.exe
digraph testgraph{
P->F
F->B
F->H
H->G
P->S
S->R
S->Y
Y->T
T->W
Y->Z
}
```

Ilustración 32: Código para dibujar el árbol

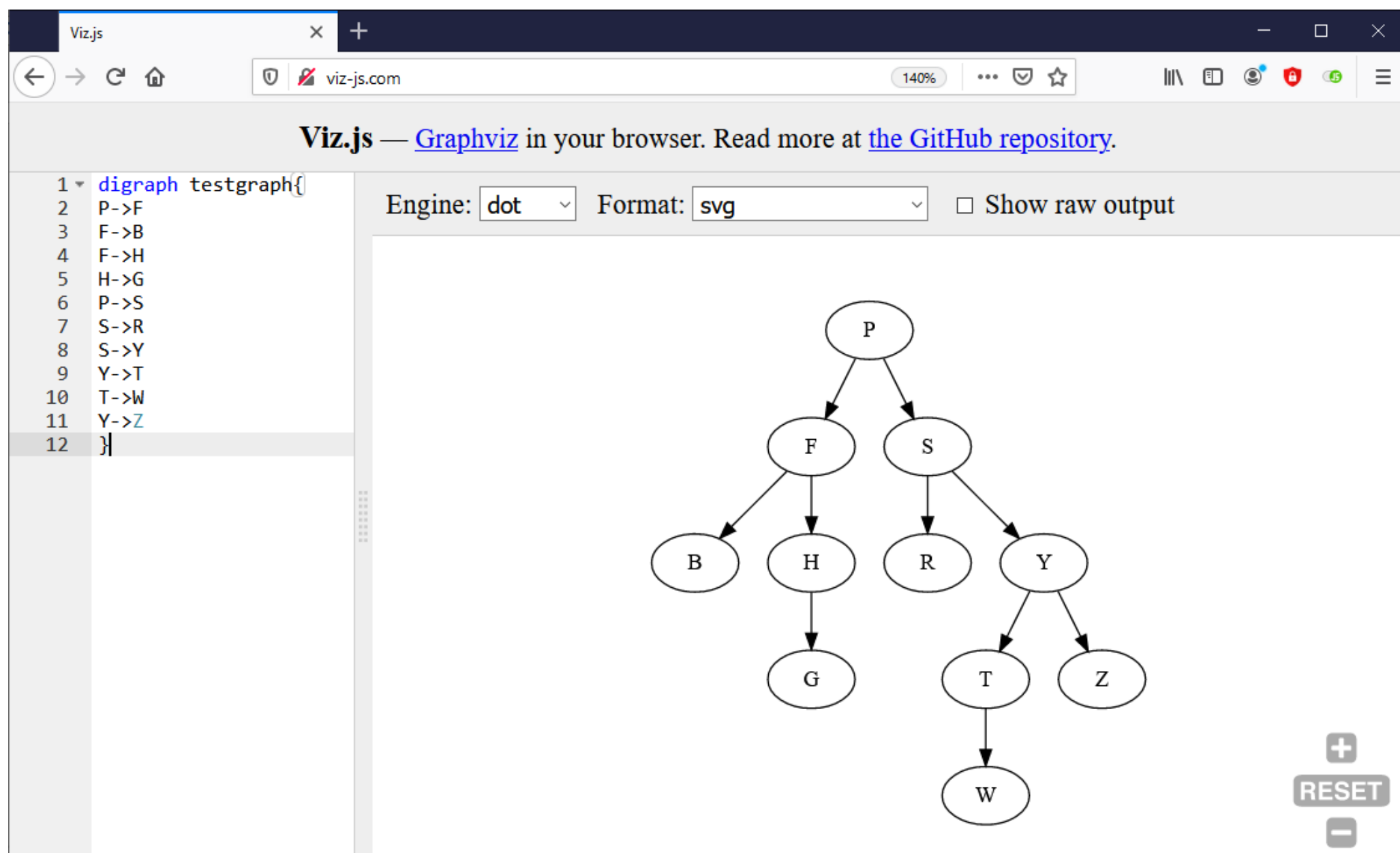


Ilustración 33: Árbol binario dibujado

Las opciones deben ser Engine: dot y Format: svg

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.

Directorio 19. Nodo.cs

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

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

Directorio 19. NodosNivel.cs

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

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

Directorio 19. Program.cs

```
//Recorrido por niveles de un árbol binario
using System;
using System.Collections.Generic;

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

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

    Console.ReadKey();
}

//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);
}
}
```

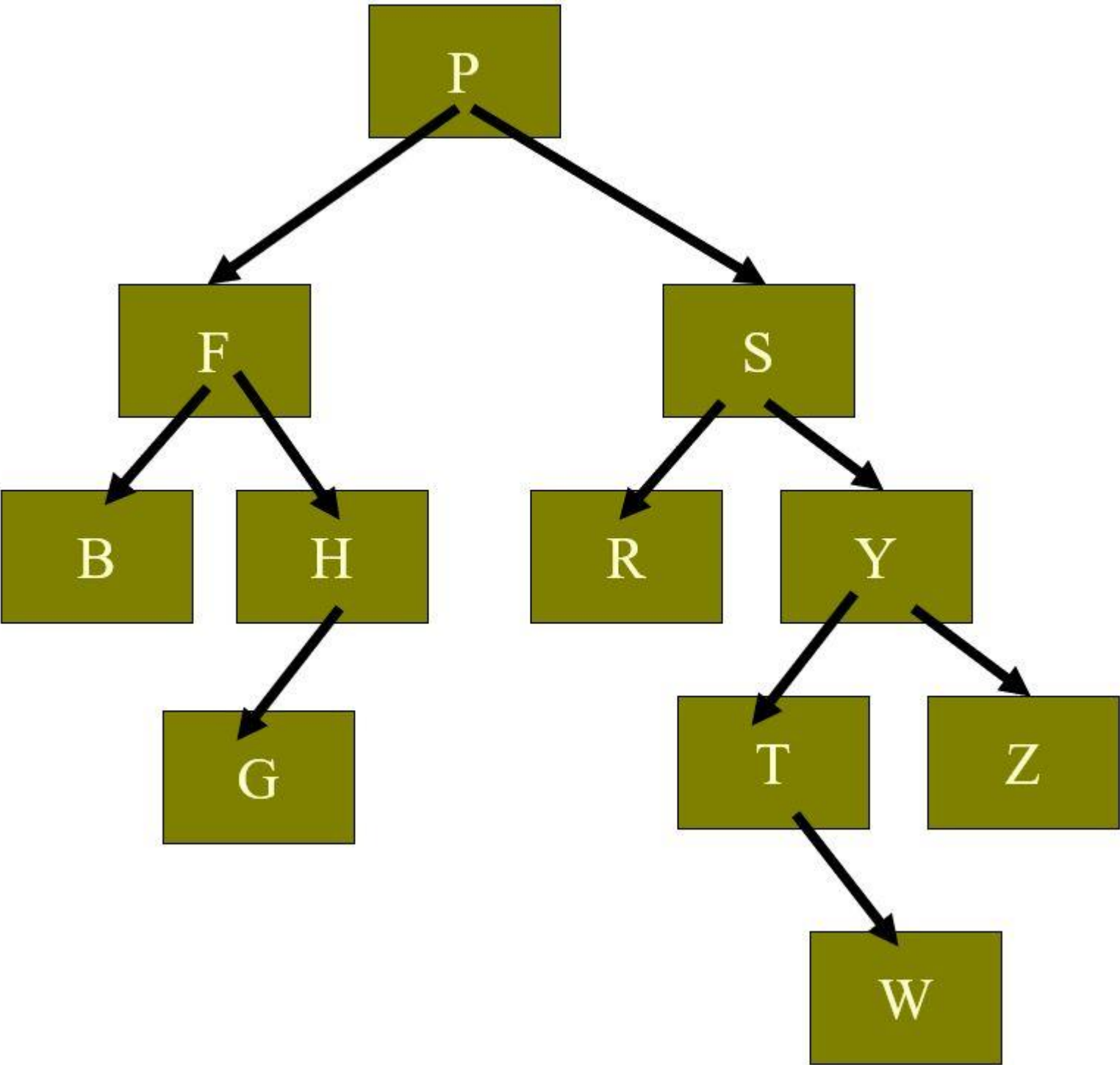


Ilustración 34: Árbol binario de ejemplo

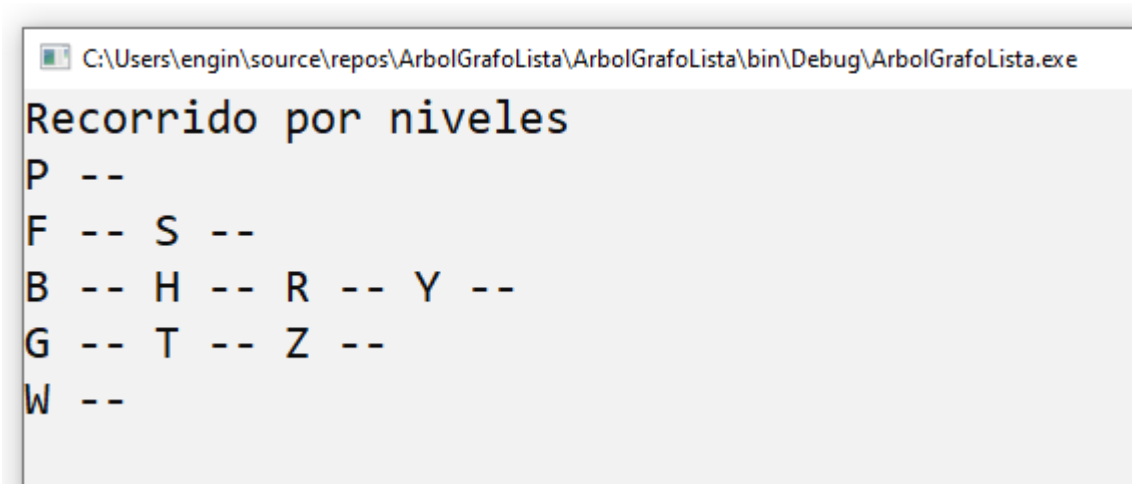


Ilustración 35: Recorrido por niveles

Árbol N-ario

Definición

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

Directorio 20. Nodo.cs

```
using System;
using System.Collections.Generic;

namespace ArbolGrafoLista {
    class Nodo {
        //Atributos propios
        public string Cadena { get; set; }
        public char Caracter { get; set; }
        public int Entero { get; set; }
        public double NumReal { get; set; }

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

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

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

        //Imprime Contenido
        public void Imprime() {
            Console.WriteLine("Cadena: " + Cadena + " Caracter: " + Caracter.ToString());
            Console.WriteLine(" Entero: " + Entero.ToString() + " Real: " + NumReal.ToString());
            Console.WriteLine(" Número de hijos: " + Hijos.Count + "\r\n");
        }
    }
}
```

Directorio 20. Program.cs

```
using System;

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

            //Agrega varios hijos a esa raíz
            arbolN.AgregaHijo(new Nodo("BBBB", 'b', 2, 0.2));
            arbolN.AgregaHijo(new Nodo("CCCC", 'c', 3, 0.3));
            arbolN.AgregaHijo(new Nodo("DDDD", 'd', 4, 0.4));
            arbolN.AgregaHijo(new Nodo("EEEE", 'e', 5, 0.5));
            arbolN.AgregaHijo(new Nodo("FFFF", 'f', 6, 0.6));

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

            //Agrega varios hijos al nodo "EEEE"
            arbolN.Hijos[4].AgregaHijo(new Nodo("Ekkk", 'k', 10, 1.1));
            arbolN.Hijos[4].AgregaHijo(new Nodo("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();

            Console.ReadKey();
        }
    }
}
```

```
}  
}
```

```
C:\Users\engin\source\repos\ArbolGrafoLista\ArbolGrafoLista\bin\Debug\ArbolGrafoLista.exe  
Cadena: AAAA Caracter: a Entero: 1 Real: 0,1 Número de hijos: 5  
Cadena: BBBB Caracter: b Entero: 2 Real: 0,2 Número de hijos: 3  
Cadena: CCCC Caracter: c Entero: 3 Real: 0,3 Número de hijos: 0  
Cadena: DDDD Caracter: d Entero: 4 Real: 0,4 Número de hijos: 0  
Cadena: EEEE Caracter: e Entero: 5 Real: 0,5 Número de hijos: 0  
Cadena: FFFF Caracter: f Entero: 6 Real: 0,6 Número de hijos: 2  
Cadena: Bhhh Caracter: h Entero: 7 Real: 0,7 Número de hijos: 0
```

Ilustración 36: Árbol N-Ario

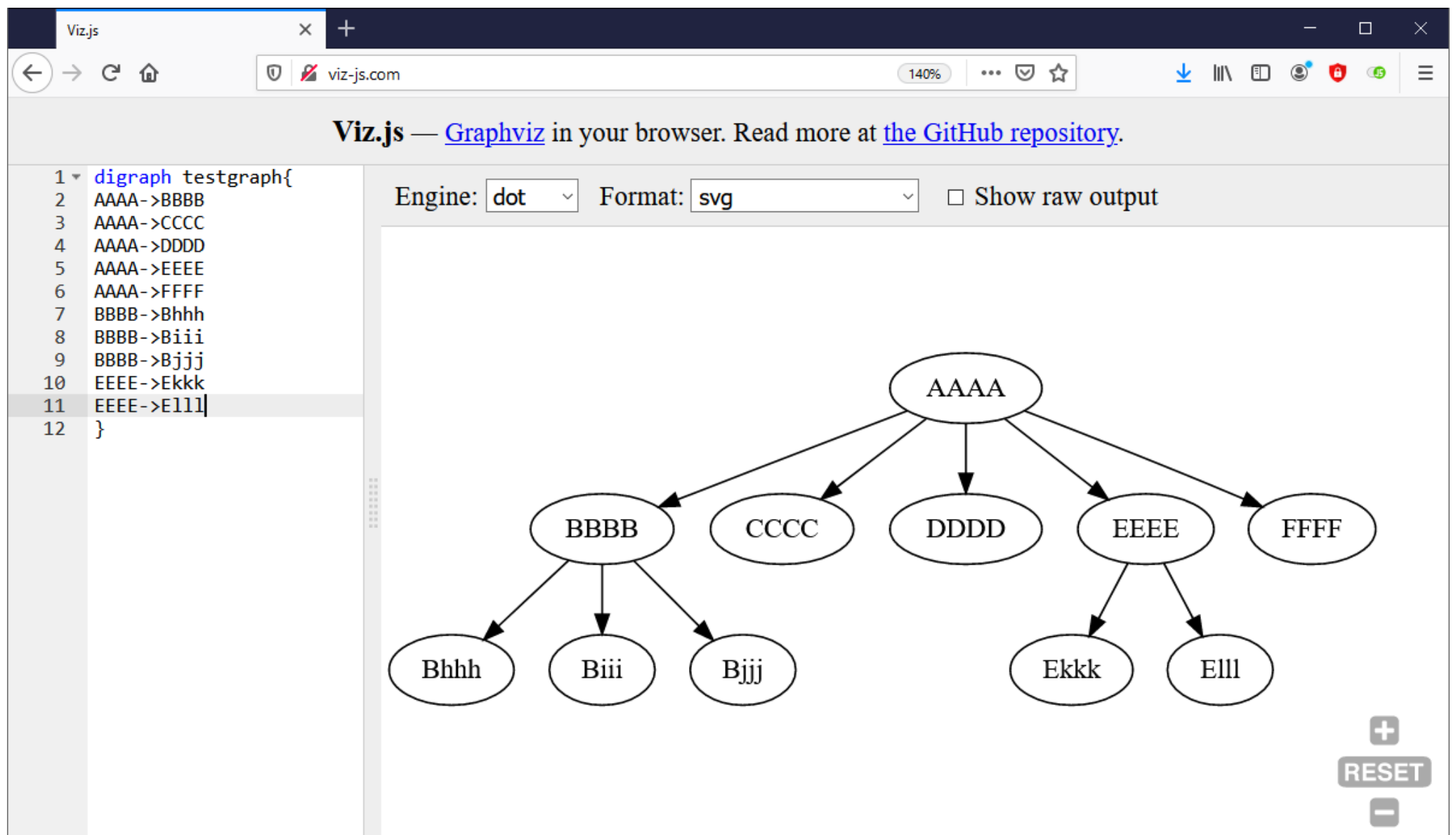


Ilustración 37: Dibujando el árbol N-Ario

Recorriéndolo

Se usa un procedimiento recursivo.

Directorio 21. Nodo.cs

```
using System;
using System.Collections.Generic;

namespace ArbolGrafoLista {
    class Nodo {
        //Atributos propios
        public string Cadena { get; set; }
        public char Caracter { get; set; }
        public int Entero { get; set; }
        public double NumReal { get; set; }

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

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

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

        //Imprime Contenido
        public void Imprime() {
            Console.WriteLine("Cadena: " + Cadena + " Caracter: " + Caracter.ToString());
            Console.WriteLine(" Entero: " + Entero.ToString() + " Real: " + NumReal.ToString());
            Console.WriteLine(" Número de hijos: " + Hijos.Count);
        }
    }
}
```



```

using System;

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

            //Agrega varios hijos a esa raíz
            arbolN.AgregaHijo(new Nodo("BBBB", 'b', 2, 0.2));
            arbolN.AgregaHijo(new Nodo("CCCC", 'c', 3, 0.3));
            arbolN.AgregaHijo(new Nodo("DDDD", 'd', 4, 0.4));
            arbolN.AgregaHijo(new Nodo("EEEE", 'e', 5, 0.5));
            arbolN.AgregaHijo(new Nodo("FFFF", 'f', 6, 0.6));

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

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

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

            //Imprime el árbol
            RecorreArbolN(arbolN);

            Console.ReadKey();
        }

        //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]);
            }
        }
    }
}

```

```

C:\Users\engin\source\repos\ArbolGrafoLista\ArbolGrafoLista\bin\Debug\ArbolGrafoLista.exe
Cadena: AAAA Character: a Entero: 1 Real: 0,1 Número de hijos: 5
Cadena: BBBB Character: b Entero: 2 Real: 0,2 Número de hijos: 3
Cadena: Bhhh Character: h Entero: 7 Real: 0,7 Número de hijos: 0
Cadena: Biii Character: i Entero: 8 Real: 0,8 Número de hijos: 5
Cadena: Biiia Character: n Entero: 13 Real: 1,4 Número de hijos: 0
Cadena: Biiib Character: o Entero: 14 Real: 1,5 Número de hijos: 0
Cadena: Biiic Character: p Entero: 15 Real: 1,6 Número de hijos: 0
Cadena: Biiid Character: q Entero: 16 Real: 1,7 Número de hijos: 0
Cadena: Biiie Character: r Entero: 17 Real: 1,8 Número de hijos: 0
Cadena: Bjjj Character: j Entero: 9 Real: 0,9 Número de hijos: 0
Cadena: CCCC Character: c Entero: 3 Real: 0,3 Número de hijos: 0
Cadena: DDDD Character: d Entero: 4 Real: 0,4 Número de hijos: 0
Cadena: EEEE Character: e Entero: 5 Real: 0,5 Número de hijos: 3
Cadena: Ekkk Character: k Entero: 10 Real: 1,1 Número de hijos: 0
Cadena: Elll Character: l Entero: 11 Real: 1,2 Número de hijos: 0
Cadena: Emmm Character: m Entero: 12 Real: 1,3 Número de hijos: 0
Cadena: FFFF Character: f Entero: 6 Real: 0,6 Número de hijos: 0

```

Ilustración 38: Recorrido del árbol

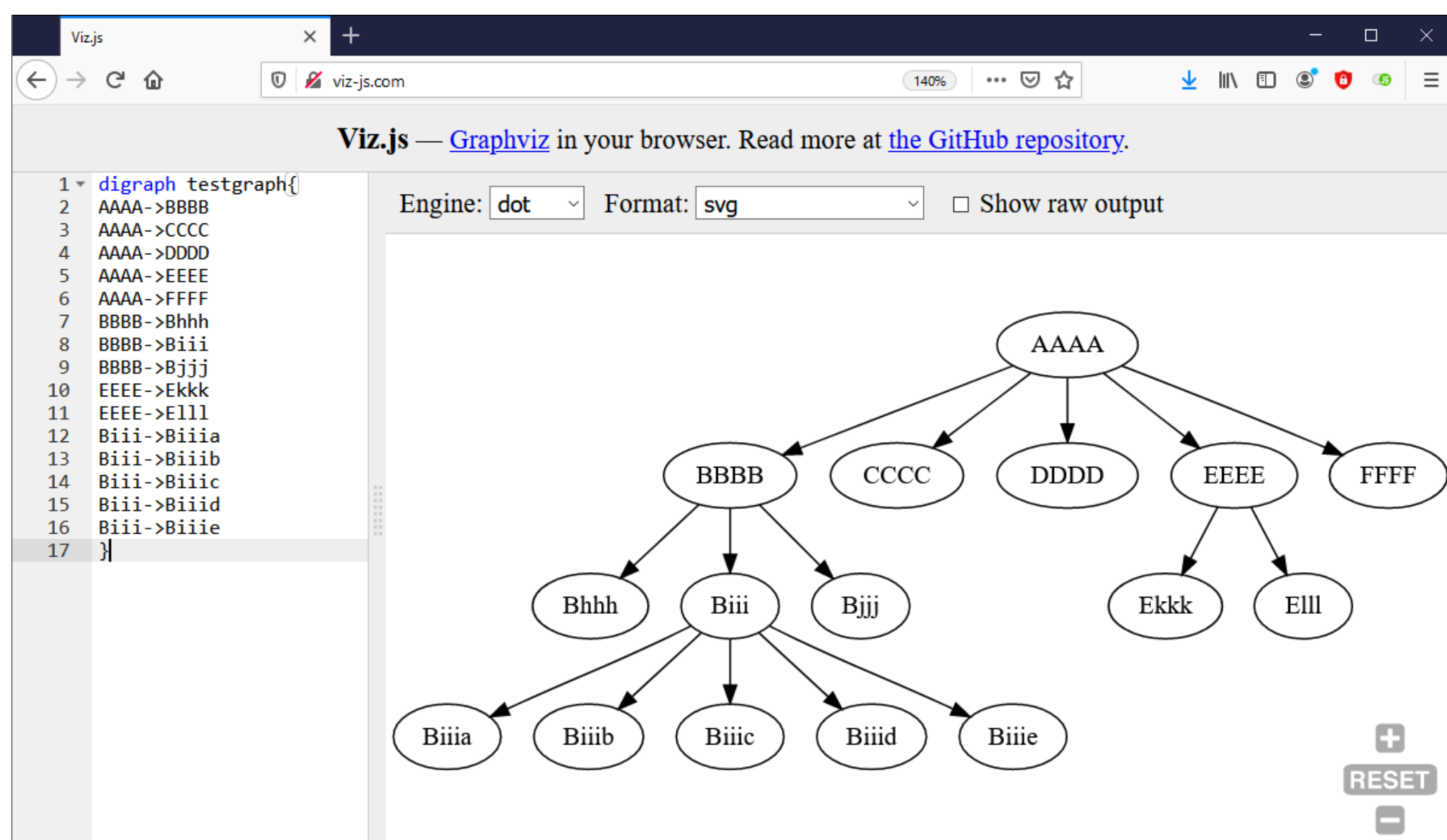


Ilustración 39: Árbol N-ario generado

Grafos

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

Directorio 22. Nodo.cs

```
//Unidad básica en el grafo cuadrado. Apuntará Arriba, Abajo, Derecha e Izquierda
namespace ArbolGrafoLista {
    class Nodo {
        public string Cadena { get; set; }

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

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

Directorio 22. Program.cs

```
//Grafo básico
using System;

namespace ArbolGrafoLista {
    class Program {
        public static void Main() {
            //Genera los nodos
            Nodo nodoA = new Nodo("aaaa");
            Nodo nodoB = new Nodo("bbbb");
            Nodo nodoC = new Nodo("cccc");
            Nodo nodoD = new Nodo("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.Cadena);
            Console.WriteLine("nodoA->Abajo: " + nodoA.Abajo.Cadena);
            Console.WriteLine("nodoA->Abajo->Derecha: " + nodoA.Abajo.Derecha.Cadena);
            Console.WriteLine("nodoA->Abajo->Derecha->Arriba: " + nodoA.Abajo.Derecha.Arriba.Cadena);
            Console.ReadKey();
        }
    }
}
```

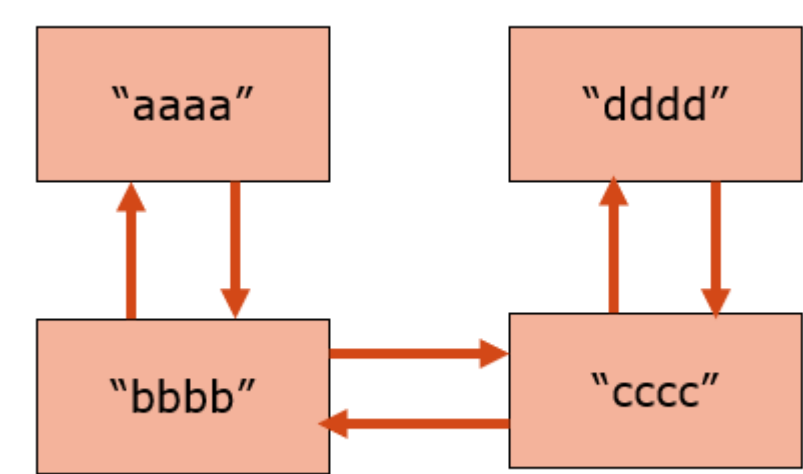
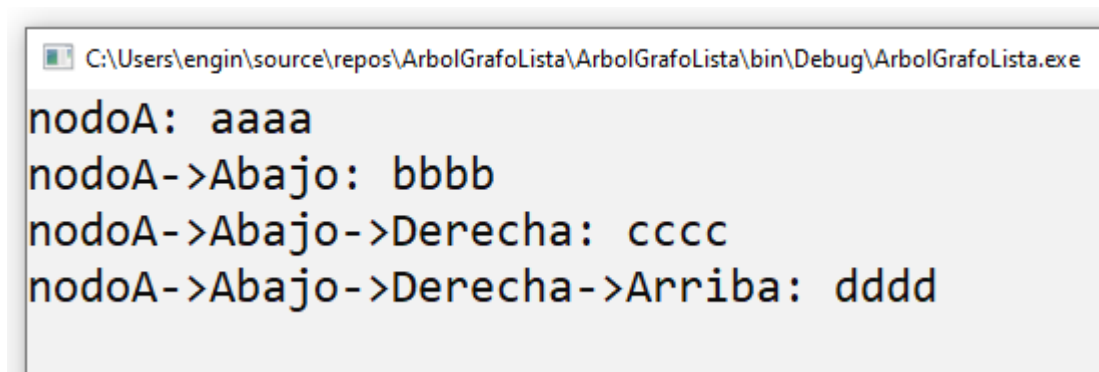


Ilustración 40: Grafo generado



```
C:\Users\engin\source\repos\ArbolGrafoLista\ArbolGrafoLista\bin\Debug\ArbolGrafoLista.exe
nodoA: aaaa
nodoA->Abajo: bbbb
nodoA->Abajo->Derecha: cccc
nodoA->Abajo->Derecha->Arriba: dddd
```

Ilustración 41: Imprimiendo el grafo

```
//Unidad básica en el grafo cuadrado. Apuntará Arriba, Abajo, Derecha e Izquierda
namespace ArbolGrafoLista {
    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;
        }
    }
}
```

```
//Generando un grafo aleatoriamente
using System;
using System.Collections.Generic;

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

            //Usa una lista para guardar los nodos
            List<Nodo> listado = new List<Nodo>();

            //Genera los nodos dentro de un List
            int Total = azar.Next(20, 30);
            for (int cont = 1; cont <= Total; cont++) {
                listado.Add(new Nodo(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.WriteLine(listado[cont].Numero + "->" +
listado[cont].Arriba.Numero);
                if (listado[cont].Abajo != null) Console.WriteLine(listado[cont].Numero + "->" +
listado[cont].Abajo.Numero);
                if (listado[cont].Izquierda != null) Console.WriteLine(listado[cont].Numero + "->" +
listado[cont].Izquierda.Numero);
                if (listado[cont].Derecha != null) Console.WriteLine(listado[cont].Numero + "->" +
listado[cont].Derecha.Numero);
            }
            Console.WriteLine("}");
            //Imprime
            Console.ReadKey();
        }
    }
}
```

```
C:\Users\engin\source\repos\ArbolGrafoLista\Ar
digraph testgraph{
1->21
1->5
2->12
3->10
3->13
3->23
4->19
5->3
5->12
5->1
5->19
7->25
7->2
```

Ilustración 42: Genera las instrucciones para dibujar

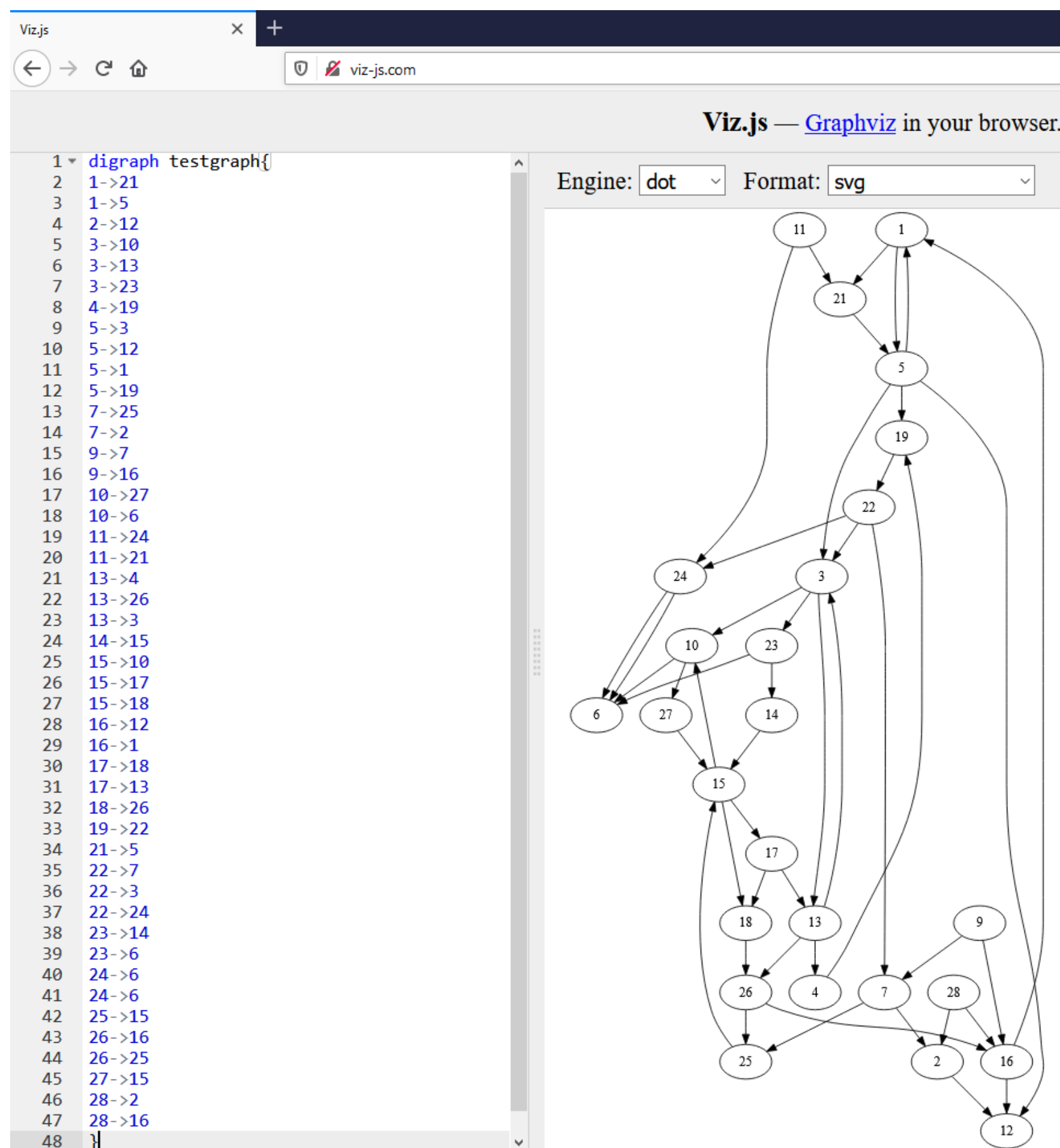


Ilustración 43: Un grafo generado aleatoriamente

Bibliografía

- [1] Wikipedia, «GNU Lesser General Public License,» 2017. [En línea]. Available: https://es.wikipedia.org/wiki/GNU_Lesser_General_Public_License. [Último acceso: mayo 2020].
- [2] GeeksforGeeks, «C# | ArrayList Class,» 03 abril 2019. [En línea]. Available: <https://www.geeksforgeeks.org/c-sharp-arraylist-class/>. [Último acceso: 07 enero 2021].
- [3] TutorialsTeacher, «C# - List<T>,» 2020. [En línea]. Available: <https://www.tutorialsteacher.com/csharp/csharp-list>. [Último acceso: 07 enero 2021].
- [4] Microsoft, «Dictionary<TKey,TValue> Clase,» 2021. [En línea]. Available: <https://docs.microsoft.com/es-es/dotnet/api/system.collections.generic.dictionary-2?view=net-5.0>. [Último acceso: 07 enero 2021].
- [5] Guru 99, «C# Queue with Examples,» 2021. [En línea]. Available: <https://www.guru99.com/c-sharp-queue.html>. [Último acceso: 07 enero 2021].
- [6] csharp.com.es, «Tutorial de C#,» 2021. [En línea]. Available: <https://csharp.com.es/la-pila-en-c-lifo-la-cola-lilo/>. [Último acceso: 07 enero 2021].
- [7] TutorialsPoint, «C# - Hashtable Class,» 2021. [En línea]. Available: https://www.tutorialspoint.com/csharp/csharp_hashtable.htm. [Último acceso: 07 enero 2021].
- [8] dot net perls, «C# SortedList,» 2021. [En línea]. Available: <https://www.dotnetperls.com/sortedlist>. [Último acceso: 07 enero 2021].
- [9] A. Sharma, «Implementing Linked List In C#,» 22 septiembre 2019. [En línea]. Available: <https://www.c-sharpcorner.com/article/linked-list-implementation-in-c-sharp/>. [Último acceso: 07 enero 2021].