REPORTE DE PRÁCTICA

| Nombre Completo | Machado Sanchez Javier |
| --- | --- |
| Número de la Unidad a Evaluar | 4 |
| Tema a Evaluar | Árboles Binarios Inserción y Recorridos |
| Fecha de Entrega | 24 de Octubre del 2023 |

**Programa Fuente**

| using System;  using System.Collections.Generic;  namespace P29\_Arboles\_Binarios\_Insercion\_Recorridos  {  internal class Program  {  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Declaración de clase árbol  public class Arbol  { //Elemento  public int numero;  //Apuntadores  public Arbol Derecha;  public Arbol Izquierda;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // VARIABLES GLOBALES  public static int elemento;  public static bool flag;  public static Queue<Arbol> ArbolAmplitud = new Queue<Arbol>();  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // DECLARACIÓN DE ARBOLES  public static Arbol Raiz, Resultado, Temp1, Temp2;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  static void Main(string[] args)  {  Console.Title = "P29 Árboles Binarios Inserción y Recorridos";  //Declaración de variables auxiliares para control de menú  char opcion; bool aux\_repetir;  //Declaración de menú  do  {  Console.Write("\n\t\t.: MENÚ OPCIONES :." +  "\n\n\t[1] - Insertar" +  "\n\t[2] - PreOrden" +  "\n\t[3] - InOrden" +  "\n\t[4] - PostOrden" +  "\n\t[5] - Amplitud" +  "\n\t[6] - Salir" +  "\n\n\tIngrese el número de la opción deseada: ");  if (char.TryParse(Console.ReadLine(), out opcion))  {  Console.Write($"\n\n\tSe ha capturado la opción \"{opcion}\" correctamente" +  $"\n\n\tPresione la tecla <INTRO> para continuar...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  //Switch case para controlar las opciones  switch (char.ToUpper(opcion))  {  case '1':  {  do  { //Captura de datos  Console.Write("\n\t\t.: INSERCCIÓN :." +  "\n\n\tIngrese el número a insertar: ");  Program.elemento = int.Parse(Console.ReadLine());  //Llamada a método de insertar  InsertarElemento();  Program.Resultado = Raiz;  do  {  Console.Write("\n\t¿Desea insertar otro elemento [S=Sí][N=No] ?" +  "\n\tIngrese la letra de la respuesta: ");  if (char.TryParse(Console.ReadLine(), out opcion)) aux\_repetir = false;  else  {  aux\_repetir = true;  Console.Write("\n\n\tLa opción no se ha podido capturar" +  "\n\n\tPresione la tecla <INTRO> para intentarlo de nuevo...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  } while (aux\_repetir);  Console.Write("\n\n\tPresione la tecla <INTRO> para continuar...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  } while (char.ToUpper(opcion) != 'N');  }  break;  case '2':  {  Console.Write("\n\t\t.: RECORRIDO PREORDEN :." +  "\n\n\t| Este es el recorrido en PreOrden del árbol |\n\n\t");  //Llamada a método para recorrer en PreOrden  PreOrden(Program.Resultado);  Console.Write("\n\n\tPresione la tecla <INTRO> para continuar...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  break;  case '3':  {  Console.Write("\n\t\t.: RECORRIDO INORDEN :." +  "\n\n\t| Este es el recorrido en InOrden del árbol |\n\n\t");  //Llamada a método para recorrer en InOrden  InOrden(Program.Resultado);  Console.Write("\n\n\tPresione la tecla <INTRO> para continuar...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  break;  case '4':  {  Console.Write("\n\t\t.: RECORRIDO POSTORDEN :." +  "\n\n\t| Este es el recorrido en PostOrden del árbol |\n\n\t");  //Llamada a método para recorrer en PostOrden  PostOrden(Program.Resultado);  Console.Write("\n\n\tPresione la tecla <INTRO> para continuar...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  break;  case '5':  {  Console.Write("\n\t\t.: RECORRIDO AMPLITUD :." +  "\n\n\t| Este es el recorrido en Amplitud del árbol |\n\n\t");  //Condición para validar si esta vacío el arbor  if (Raiz == null) Console.WriteLine("\n\tÁrbol vacío");  else  {  ArbolAmplitud.Clear();  ArbolAmplitud.Enqueue(Raiz);  //Llamada a método para recorrer en Amplitud  Amplitud(Program.ArbolAmplitud);  }  Console.Write("\n\n\tPresione la tecla <INTRO> para continuar...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  break;  case '6':  {  //Caso 4. Salida del programa  Console.Write("\n\t\t:. SALIDA :." +  "\n\n\tGracias por utilizar nuestro programa" +  "\n\n\tPresione la tecla <Esc> para salir...");  while (Console.ReadKey().Key != ConsoleKey.Escape) ;  }  break;  default:  {  //En caso de ingresar una opción inválida  Console.Write("\n\t\t.: OPCIÓN INVALIDA :." +  "\n\n\tPor favor ingrese una opción existente" +  "\n\n\tPresione la tecla <INTRO> para continuar...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  break;  }  }  else  {  Console.Write("\n\t\t.: OPCIÓN INVALIDA :." +  "\n\n\tPor favor ingrese una opción existente" +  "\n\n\tPresione la tecla <INTRO> para continuar...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  } while (opcion != '6');  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Inicio procedimiento insertar    public static void InsertarElemento()  {  if(Raiz == null)  { //Creación de raíz en caso de estar vacío el árbol  Raiz = new Arbol  {  numero = elemento,  Derecha = null,  Izquierda = null  };  Console.WriteLine($"\n\n\tSe ha ingresado el número {elemento} con éxito.");  }  else  { //Cuándo ya hay raíz  Program.flag = false;  Temp1 = Raiz;  //Ciclo para encontrar el lugar del valor dentro del árbol  do  { //Condición para encontrar el subárbol al que pertenece el número  if(elemento > Temp1.numero)  { //Comprobando si existe elemento en la rama derecha  if (Temp1.Derecha == null)  { //Insertando número  Temp2 = new Arbol  {  numero = elemento,  Derecha = null,  Izquierda = null  };  Temp1.Derecha = Temp2;  flag = true; //Bandera para saber que ya se insertó  }  else Temp1 = Temp1.Derecha;  }  else  { //Comprobando si existe elemento en la rama izquierda  if (Temp1.Izquierda == null)  { //Insertando número  Temp2 = new Arbol  {  numero = elemento,  Derecha = null,  Izquierda = null  };  Temp1.Izquierda = Temp2;  flag = true; //Bandera para saber que ya se insertó  }  else Temp1 = Temp1.Izquierda;  }  } while (flag != true);  Console.WriteLine($"\n\n\tSe ha ingresado el número {elemento} con éxito.");  }  }  //Fin procedimiento insertar  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Inicio procedimiento PreOrden  public static void PreOrden(Arbol Resultado)  {  if(Resultado != null) //Condición para caso base  {  Console.Write($"[ {Resultado.numero} ] "); //Raíz  PreOrden(Resultado.Izquierda); //Llamador más pequeño para subárbol izquierdo  PreOrden(Resultado.Derecha); //Llamador más pequeño para subárbol derecho  }  }  //Fin procedimiento PreOrden  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Inicio procedimiento InOrden  public static void InOrden(Arbol Resultado)  {  if (Resultado != null) //Condición para caso base  {  InOrden(Resultado.Izquierda); //Llamador más pequeño para subárbol izquierdo  Console.Write($"[ {Resultado.numero} ] "); //Raíz  InOrden(Resultado.Derecha); //Llamador más pequeño para subárbol derecho  }  }  //Fin procedimiento InOrden  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Inicio procedimiento PostOrden  public static void PostOrden(Arbol Resultado)  {  if (Resultado != null) //Condición para caso base  {  PostOrden(Resultado.Izquierda); //Llamador más pequeño para subárbol izquierdo  PostOrden(Resultado.Derecha); //Llamador más pequeño para subárbol derecho  Console.Write($"[ {Resultado.numero} ] "); //Raíz  }  }  //Fin procedimiento PostOrden  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Inicio procedimiento Amplitud  public static void Amplitud(Queue<Arbol> ArbolAmplitud)  {  if(ArbolAmplitud.Count > 0) //Condición para caso base  {  Arbol Auxiliar = ArbolAmplitud.Dequeue(); //Pasando un apuntador hacia otro temporal y eliminando de la cola  Console.Write($"[ {Auxiliar.numero} ] "); //Raíz  if (Auxiliar.Izquierda != null)  {  ArbolAmplitud.Enqueue(Auxiliar.Izquierda);  }  if (Auxiliar.Derecha != null)  {  ArbolAmplitud.Enqueue(Auxiliar.Derecha);    }  Amplitud(ArbolAmplitud); //Llamador más pequeño  }  }  //Fin procedimiento Amplitud  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  }  } |
| --- |

**Ventana de Entrada de datos**

| PreOrden(RID)  56, 34, 34, 21, 12, 55, 78, 67, 89  InOrden(IRD)  12, 21, 34, 34, 55, 56, 67, 78, 89  PostOrden(IDR)  12, 21, 34, 55, 34, 67, 89, 78, 56  Amplitud  56, 34, 78, 34, 55, 67, 89, 21, 12 |
| --- |

**Ventana de Resultados**

|  |
| --- |

**EXPLICA BREVEMENTE QUÉ TE PARECIÓ EL PROBLEMA, QUE SE TE DIFICULTO Y COMO LO SOLUCIONASTE.**

| Realizar árboles binarios a mano me parece muy divertido y entretenido, tanto realizarlos como obtener sus recorridos. Hacerlo en programación me pareció igual de divertido, además de que utilizar recursividad en los métodos facilita mucho la comprensión de la lógica y en lo personal me ayudó a tener una nueva perspectiva del algoritmo detrás de los recorridos, ya que C# tiene una opcion de “paso por paso” donde pude ir viendo como se iba recorriendo el arbol termino a termino.  Al inicio se me complicó realizar el método del recorrido por amplitud en recursividad, pues el programa corria pero el recorrido no era correcto. Para solucionarlo me acordé del caso base, llamador más pequeño y caso general, definiendo eso pude darme cuenta porque con ciclo while funcionaba y en recursividad no. Al final pude corregirlo y tener todos los recorridos con recursividad. Me divertí y me gustó mucho realizar este programa. |
| --- |