REPORTE DE PRÁCTICA

| Nombre Completo | Machado Sanchez Javier |
| --- | --- |
| Número de la Unidad a Evaluar | 5 |
| Tema a Evaluar | Quick Sort Descendente |
| Fecha de Entrega | 02 de Noviembre del 2023 |

**Programa Fuente**

| using System;  namespace P35\_Quick\_Sort\_Descendente  {  internal class Program  {  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Declaración de método para ordenar  public static int[] QuickSort(int[] Numeros, int primero, int ultimo)  {  int i, j, central, pivote, temp;  central = (primero + ultimo) / 2;  pivote = Numeros[central];  i = primero; j = ultimo;  do  {  while (Numeros[i] > pivote) i++;  while (Numeros[j] < pivote) j--;  if (i <= j)  {  temp = Numeros[i];  Numeros[i] = Numeros[j];  Numeros[j] = temp;  i++; j--;  }  } while (i <= j);  if(primero < j)  {  Numeros = QuickSort(Numeros, primero, j);  }  if(i < ultimo)  {  Numeros = QuickSort(Numeros, i, ultimo);  }  return Numeros;  }  //Fin declaración de método para ordenar  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Inicio declaración método para mostrar  public static void Mostrar(int[] Numeros)  {  for (int i = 1; i <= Numeros.Length; i++)  {  if (i == (Numeros.Length / 2) + 2) Console.Write("\n");  Console.Write($"[{Numeros[i - 1]}] ");  }  }  //Fin declaración de método mostrar  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  static void Main(string[] args)  {  Console.Title = "P35 Quick Sort Descendente";  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Declaración de variables locales  int[] Numeros = new int[25];  int Numero; char opcion;  Random NumerosRandom = new Random();  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  do //Do while para controlar menú  {  //Despliegue de menú  Console.Write("\n\t\t.: MENÚ DE OPCIONES :." +  "\n\n\t[ 1 ] - Capturar Números" +  "\n\t[ 2 ] - Ordenar Números Descendentemente" +  "\n\t[ 3 ] - Salir del Programa" +  "\n\n\tIngrese el número de la opción deseada: ");  if (char.TryParse(Console.ReadLine(), out opcion) != false) //Try catch para controlar errores de formato  {  Console.Write($"\n\n\tLa opción {opcion} se ha capturado correctamente" +  $"\n\n\tPresione la tecla <INTRO> para continuar...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  //Switch case para controlar opciones  switch (opcion)  {  case '1':  {  Console.WriteLine("\n\t\t.: CAPTURA DE DATOS :." +  "\n\n\t| La captura de datos se ha realizado por medio de números random |");  for (int i = 1; i <= Numeros.Length; i++) //Ciclo para capturar todos los números  {  Numero = NumerosRandom.Next(-100, 100);  Console.Write($"\n\t{i}.- Ingrese un numero: {Numero}");  Console.WriteLine($"\n\tSe ha capturado el número {Numero} correctamente.");  Numeros[i - 1] = Numero;  }  Console.Write("\n\tCaptura de datos terminada, presione la tecla <INTRO> para continuar...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  break;  case '2':  {  Console.WriteLine("\n\t\t.: ORDENANDO DATOS DESCENDENTEMENTE :." +  "\n\n\t| Este es el arreglo original sin ordenar |\n\t");  Mostrar(Numeros); //Llamada a método para mostrar  Console.WriteLine("\n\n\t| Este es el arreglo ordenado descendentemente |\n\t");  Numeros = QuickSort(Numeros, 0, Numeros.Length - 1);//llamada a método para ordenar  Mostrar(Numeros);  Console.Write("\n\n\tOrdenamiento terminado, presione la tecla <INTRO> para continuar...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  break;  case '3':  {  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:  {  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\n\tLa opción no se ha podido capturar, asegúrese de escribirla correctamente" +  "\n\n\tPresione la tecla <INTRO> para continuar...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  } while (opcion != '3');  }  }  } |
| --- |

**Ventana de Entrada de datos**

|  |
| --- |

**Ventana de Resultados**

|  |
| --- |

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

| Con este tipo de ordenamiento tampoco había trabajado, pero se me hizo muy interesante que trabajara con números negativos también.  No tuve muchos problemas para hacer el programa, utilice la misma base de programas anteriores y modifiqué la función para hacerla acorde al quicksort. La recursividad es muy útil para este tipo de algoritmos y ayuda a entender mejor su funcionamiento iteración a iteración. |
| --- |