C#Indice

* GENERAL
  1. [Tipo de variables](#Variables)
  2. [Variable global](#VariablesGlobales)
  3. [Color en consola](#ColorEnConsola)
  4. [Tamaños y rangos](#TamañosYRangos)
  5. [Parsear](#Parsear)
     1. TryParse
  6. [Console](#Console)
  7. [Foreach(recorriendo string)](#ForeachParaString)
  8. [String](#string)
  9. [Math](#Math)
  10. [Random](#Random)
  11. [Sleep](#sleep)
  12. [DateTime](#DateTime)

|  |
| --- |
| //Soy un comentario  /\* soy un comentario  multiple\*/ |

|  |  |  |
| --- | --- | --- |
| Notas varias  concatenación implícita  automaticamente int pasa a ser string  De forma explicita  Exactamente lo mismo que arriba pero explicita | varString + varInt  varString + varInt.ToString() | |
| Al castear un numero a entero se redondea hacia abajo | (int)1.2  (int)1.9 | 1  1 |
| var.GetType();  Comparar dos objetos para obtener su clase  obj1.GetType()==obj2.GetType()  obj1.GetType()==typeof(Persona) | Devuelve el tipo de variable.  Sirve para clases etc  true si pertenecen a la misma clase.  True si el objeto pertenece a la clase Persona. | |
| nameof(var1); | Devuelve el nombre de la variable en string | |
| Guid id= Guid.NewGuid(); | Genera un valor aleatorio(ideal para Ids) (alfanumerico)  tiene como ventaja que nunca se repite. | |
| Metodo1(ref string varString); | Pasamos la variable por referencia.  Tambien el la firma del metodo debe tener ref | |

**Variables en c#**

|  |  |
| --- | --- |
| string nombre;  char nombre;  bool nombre;  int nombre;  float nombre;  double nombre;  decimal nombre;  int[] nombre=new int[3];  int[] nombre= [1,2,3,4,5]; | Solo acepta true o false  presicion simple (12.45F)  presicion doble (123.44)  presicion exacta (123.45M)  array fijo |
| public static string mensaje;  Program.mensaje=”hola mundo” | Variables globales (se pono a la altura ,arriba del main)  para llamarlo desde cualquier parte del programa |

**Varible globales**

|  |  |
| --- | --- |
| Clase1.Metodo1(out var) | Desde el programa |
| public void Metodo1(out string var){  var=””;  } | Desde la clase  Es obligatoria darle un valor; |

**Color en consola**

|  |  |
| --- | --- |
| Console.ForegroundColor=ConsoleColor.Red; | Cambia el color de las letras de consola. Recordar de volver a ponerlo en blanco |
| ConsoleColor var=ConsoleColor.Red  ForegroudColor=var; | Guardamos el color en una variable  Cambia el color |
| Color.Red; | Guardar un color |
| Color.FromArgb(0,100,100,100); | Guardar un colo con alpha rgb |

**Tamaños y rangos**

|  |  |
| --- | --- |
| sizeof(int)  int.MaxValue  int.MinValue | Devuelve el tamaño en bytes  devuelve el numero máximo que puede ser  el mínimo  **funciona solo para tipos numéricos** |

**Parsear**

|  |  |
| --- | --- |
| varInt=int.Parse(varString);  varChar=char.Parse(varString);  varInt=(int) 20; | Pasar string a int, tambien en double  pasar string a un char  castear de double a int |
| **int.TryParse(Console.ReadLine(), out int var);** | **Es un metodo especial que intenta transformar el string a numero, si lo logra lo guarda en la variable (var, que es entero), sino no hace nada** |

**Console**

|  |  |
| --- | --- |
| vasString=Console.ReadLine(); | Pide ingreso de dato del bufer como string |
| Console.WriteLine(variable);  Console.Write(variable)  Console.WriteLine(“Soy un texto”);  Console.WriteLine(“El valor es: {0}{1}”, var1, var2); | Muestra por pantalla  sin salto de linea |
| Console.ReadKey();  Console.Clear(); | Para que no se cierre la consola, en espera de una tecla  limpiar consola |
| Environment.Exit(0);. | Salir de consola |

**Foreach para recorrer string**

|  |  |
| --- | --- |
| Foreach (char i in varString){  i  } | For(int i=0; i<string,Length; i++){  string[i]  } |

**String**

|  |  |
| --- | --- |
| String.Equals(miString, “salir”); | Retorna true si son iguales |
| String.Concat(miString1,miString2) | Retorna un string con los dos strings concatenados |
| String.IsNullOrWhiteSpace(varString); | True: si la variable es nula o en blanco  false: si contiene texto. |
| miString[1]; | Retorna el valor en el index 1 |
| miString.Length | Devuelve el tamaño del string |
| miString.Trim()  miString.Trim(new char[]{' ','!','?' })  miString.TrimStart()  miString.TrimEnd()  miString.TrimStart(new char[]{' ','!','?' }) | **Devuelve una string:**  con los espacios borrados al inicio y al final  con el primero de esos caracteres que se encuentre al IyF  con los espacios borrados al inicio  con los espacios borrados al final  También aplica |
| miString.Contains(“ad”); | True si el string contiene al otro(pequeño)  false si no lo contiene |
| miString.SubString(2)  miString.SubString(1,2) | Devuelve una string desde el 0 a 2  Devuelve una string desde el 1 a 2 |
| miString.CompareTo(varString2); | 0→si son iguales  1 si el primero es mayor que el segundo  -1 si el primero es menor que el segundo |
| miString.ToLower();  miString.ToUpper(); | Retorna una copia del string en miniscula  Retorna una copia del string en mayuscula |
| string[] auxArray= miString.Split(‘.’);  string[] auxArray= miString.Split(‘.’,2);  string[] auxArray= miString.Split(‘.’,’/’);  string[] auxArray= miString.Split(new char[] {'.','/'});  Parametros extras:  miString.Split(‘.’,StringSplitOptions.TrimEntries);  miString.Split(‘.’,StringSplitOptions.RemoveEmptyEntries); | Retorna un array de string separando por lo indicado nuestros string  limita a 2 string de separacion(len=2)  separación por mas de un carácter  igual mas eficiente  elimina los espacios en blanco al inicio y fin de las cadenas  Elimina los campos vacíos del array. |

**Math**

|  |  |
| --- | --- |
| Math.Pow(varNunm, 3); | Eleva a la potencia en este caso al cubo. Devuelve un Double |
| Math.Sqrt(numDouble) | Calcula la raiz cuadrada |
| Math.Abs(num); | Devulve int en modulo. |

**Random**

|  |  |
| --- | --- |
| Random rd =new Random(); | Crear objeto e instanciarlo |
| rd.Next() | Genera un numero aleatorio positivo |
| rd.Next(25); | Genera un número aleatorio entre el 0 -24 |
| rd.Next(1,25); | Genera 1-24 |
| rd.Next(20,100)/10; | Para generar números randoms con decimales |

**Sleep**

|  |  |
| --- | --- |
| Thread.Sleep(2000); | Duerme 2s |

**DateTime**

|  |  |
| --- | --- |
| **DateTime dt;** | **Se crea una clase(obligatoria)** |
| dt=DateTime.Now  dt=DateTime.MinValue;  dt=DateTime.MaxValue; | Fecha y hora actual  1/1/0001 12:00:00 AM  12/31/9999 11:59:5 PM |
| DateTime dt=new(aaaa,mm,dd,hh,mm,ss) | Se crea una clase e instancia |
| dt.addDays(20); | Devuelve otro objeto aumentando 20 dias. Puede ser cualquier condición…. |
| DateTime dt=DateTime.Now  dt.Year;  dt.Month;  dt.Day;  dt.Hour;  dt.Minute;  dt.Second; | Devuelve el año en **ENTERO**  etc |
| PARSEAR  string varSring=”02/04/2000”;  DateTime dt=DateTime.Paser(varString); | Parseamos el string a un objeto datetime |
| string varString= dt.ToString("dd/MM/yyyy" ); | Darle formato al DateTime |
| TimeSpan dif= dt1-dt2;  varInt=dif.Days; | Obtenemos la diferencia entre las dos fechas  nos devuelve esa diferencia en días o lo que queramos |

**Equals**

|  |  |
| --- | --- |
| class1.Equals(class2); | True si las dos clases guardan la misma instancia (dirección de memoria)  False si son dos objetos diferentes  (equals compara contenido?) |
| Equals(clase1,clase2); | True si las dos clases guardan la misma instancia (dirección de memoria)  False si son dos objetos diferentes |

**Arrays**

|  |  |
| --- | --- |
| Int[] miArray =new int[5] | Creamos un array de enteros con 5 posiciones |
| MiClase[] miArrarDeClases = new MiClase[5]; | Creamos un array de objetos. |
| Al crear el array automaticamente tienen valores por default  (0 para int)  (null para objetos)  (“”para string) | |
| MiArray.Length | Devuelve el tamaño del array |
| Int[,] miMatriz= new int[2,2]; | Creamos una matriz de enteros de 2x2 |

**Listas**

|  |  |
| --- | --- |
| List <int> miLista = new List<int>();  List <int> miLista = new List<int>(){1,2,3,}; | Instanciar una lista  Instanciar y cargar datos |
| miLista.Add(5);  miLista.Remove(5);  miLista.RemoveAll(5);  miLista.Clear();  miLista.Contain(5);  miLista[0];  miLista.Sort();  miLista.Reverse();  miLista.Count(); | Agregar una variable  remueve el primer valor que aparezca desde la izquierda(true si removio algo, false si no removio nada)  Remueve todas las apariciones.  Elimina todos los elementos de la lista  true si contiene. False si no . **También funciona con objetos**  Devuelve el elemento en esta posición  ordena de la a→z  voltea la lista  devuelve la cantidad de elementos |
| **Sortear lista con delegados**  miLista.Sort(PuntFunc);  *btnDrecho sobre PuntFunc→acciones rapidas→generar metodo*  private static int PuntFunc(int x, int x){  if(x>y){  return 1;  }  return -1;  }  **Sortear un string**  private static int PuntFunc(string x, string y){  if(x.CompareTo(y)>0){  return 1;  }  return -1;  } | Llamamos al puntero a función  hack  Creamos el método  método para comparar strings(también sirve con objetos) |

Diccionarios

|  |  |
| --- | --- |
| Dictionary<string,int> miDicc=new Dictionary<string,int>();  {  {“jonathan”,1},  {“alanoca”,3},  }; | Intanciamos un diccionario (las keys no se pueden repetir)  instanciar con valores(poner antes del ;) |
| miDicc.add(“gerardo”, 2);  miDicc.Count  miDicc[“jonathan”]; | Agrega al diccionario  devuelve la cantidad de registros que tiene  devuelve el valor de la llave |
| miDicc.Keys  miDicc.Keys.toList() | devuelve una colección de keys (tratarlo como una list)  para recorrer en un **foreach** con **var**  Transformar a lista  (Misma logica para las **Values**) |
| miDicc.Keys.OrderBy(x=>x);  miDicc.Keys.OrderByDescending(x=>x); | Devuelve una “lista” ordenada a→z de las llaves  Devuelve una “lista” ordenada z→a de las llaves |
| miDicc.ContainKey(“jonathan”);  miDicc.Reverse(); | True si contiene; False si no  devuelve un “diccionario” con los registros invertidos (Guardarlos en un var) |
| from entry in a orderby entry.Value descending select entry | * Guardar en un var * descending o ascending (cambiable) * .Value o .Key (cambiable) |

Queue

|  |  |
| --- | --- |
| miQueue.Enqueue(2);  miQueue.Dequeue();  miQueue.Last();  miQueue.Peek();  miQueue.ToList();  miQueue.Contains(2); | Agrega a la cola  Borra el primer valor y lo retorna  retorna el ultimo registro  retorna el primer registro  Transforma en lista  True si contiene; false si no |
| miQueue.Copy(miArray,0); | Inserta los valores del queue en el array desde la posicion indicada |

Stack

|  |  |
| --- | --- |
| Stack<int> miStack =new Stack<int>();  miStack .Push(5);  miStack.CopyTo(miArray,0);  miStack.Pop()  miStack.Sum(); | Instanciamos la pila  agregamos al final de la pila  Inserta los valores del stack en el array desde la posicion indicada  devuelve el ultimo valor agregado  devuelve la sumatoria. Solo si son int |

SortedList

|  |  |
| --- | --- |
| SortedList<string,int> miSList =new SortedList<string,int>();  miSList.Add(“a”,3); | Instanciar  Agregar un registro |

**POO**

nombre de una clase o metodos de un a clase usamos la escritura PascalCase

para agregar un metodo:

1- sobre el proyecto→btn derecho ~~>agregar~~ →nuevo elemento → seleccionar clase y poner nombre

ejemplo de clase:

|  |
| --- |
| …  public static class Nombre{  public static string PonerNombre(){…}  public static int SumarNumeros(int num1, int num2){...}  } |
| Para hacer refencia o llamarlo (desde cualquier parte)  Nombre.PonerNombre();  Nombre.SumarNumeros(4,6); |

Buenas practicas: en los atributos es mejor ponerlos en privado para que los de afuera no tegan acceso

|  |
| --- |
| public class Clase1{  private string nombre;  private int edad;  } |
| Los atributos estáticos se comparten entre todas las clases (si se modifica en uno se modifica en todos)(pertenecen a la clase)  public class Clase1{  private string nombre;  private int edad;  **public static id=0;**  }  para referirse a el en un metodo se antepone el nombre de la clase. **No se usa thisp**  public Clase1(){  Clase1.id++;  }  cada vez que es llamado se sumaria 1.  -Desde el programa (no se usa el objeto sino directamente a la clase)  Console.WriteLine(Clase1.id); |

Constructores: tiene es un metodo con el mismo nombre de la clase mas () tiene la fución de inicializar los atributos

|  |
| --- |
| public clase1(){  this.nombre =”nadie”;  this.edad=0;  } |
| public clase1(string nombre, int edad) |

Objeto e instancia

|  |  |
| --- | --- |
| En programa  Clase1 nombreObjeto;  nombreObjeto= new Clase1(“nomre”,edad); | Se crea el objeto  se reserva espacio en la MD con los datos del contructor. |

SobreCarga (tener muchos constructores)

el programa llama al constructores que tiene mas parecido al invocador

|  |
| --- |
| Public |

Clases Para el workspace

Crear una hoja de clases para el workspace:

1. sobre la carpeta agregar nuevo elemento
2. seleccionar y dar nombre a clase de c#

importar una hoja de clase a un proyecto en el mismo workspace

En el proyecto al que vamos a agregar ir a la parte de sus **dependencias → opciones →agregar referencia del proyecto → tildar el proyecto del cual vamos a importar sus clases**

Para usar los clases vamos añadirlo al código. **En la parte superiro (linea 1) poner → using ejercicio01**

Para llamar una clase hacemos (se crea un objeto o instancia y se llama con su obejeto)