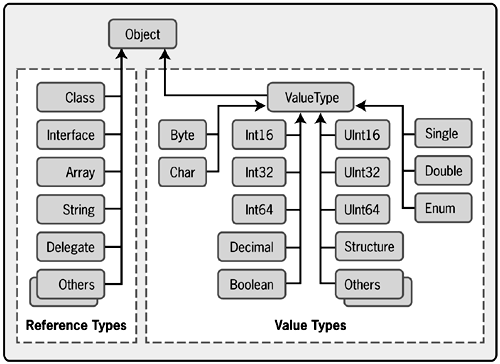
Common Type System

* Define un conjunto común de “tipos” de datos orientados a objetos.
* Todo lenguaje de programación .NET debe implementar los tipos definidos por el CTS.
* Todo tipo hereda directa o indirectamente del tipo System.Object.
* El CTS define tipos de VALOR y de REFERENCIA



Tipos de Datos

|  |  |  |  |
| --- | --- | --- | --- |
| **Categoría** | **Clase** | **Descripción** | **C# Alias** |
| **Enteros** | **Byte** | **Un entero sin signo (8-bit)** | **byte** |
|  | **SByte** | **Un entero con signo (8-bit)** | **sbyte** |
|  | **Int16** | **Un entero con signo (16-bit)** | **short** |
|  | **Int32** | **Un entero con signo (32-bit)** | **int** |
|  | **Int64** | **Un entero con signo (64-bit)** | **long** |
| **Punto Flotante** | **Single** | **Un número de punto flotante de simple precisión (32-bit)** | **float** |
|  | **Double** | **Un número de punto flotante de doble precisión (64-bit)** | **double** |
|  | **Decimal** | **Un número decimal de 96-bit** | **decimal** |
| **Lógicos** | **Boolean** | **Un valor booleano (true o false)** | **bool** |
| **Otros** | **Char** | **Un caracter Unicode (16-bit)** | **char** |
|  | **Object** | **La raíz de la jerarquía de objetos** | **object** |
|  | **String** | **Una cadena de caracteres unicode inmutable y de tamaño fijo** | **string** |

Valores Predeterminados

* Enteros
  + 0 (cero)
* Punto flotante
  + 0 (cero)
* Lógicos
  + False
* Referencias
  + Null

Conversiones Básicas en C#

Existen dos tipos de conversiones:

Implícitas:

* No interviene el programador.

Explícitas:

* Interviene el programador, porque puede haber pérdida de datos.

Conversiones Implícitas

|  |  |
| --- | --- |
| Tipo | Conversiones permitidas |
| **sbyte** | **short, int, long, float, double, decimal** |
| **byte** | **short, ushort, int, uint, long, ulong, float, double, decimal** |
| **short** | **int, long, float, double, decimal** |
| **ushort** | **int, uint, long, ulong, float, double, decimal** |
| **int** | **long, float, double, decimal** |
| **uint** | **long, ulong, float, double, decimal** |
| **long** | **float, double, decimal** |
| **ulong** | **float, double, decimal** |
| **float** | **double** |
| **double** | **Ninguna** |
| **decimal** | **Ninguna** |

Conversiones Explícitas - Ejemplo:

double pesoMax = 55.73;

float limite = pesoMax; //Error. Hay posible pérdida de datos.

//Hay que utilizar el ‘casting’ o casteo.

limite = (float) pesoMax; //Correcto.

//El programador se hace

//’responsable’ de la posible pérdida de datos.

Operadores Aritméticos

|  |  |
| --- | --- |
| **Descripción** | **C#** |
| Asignación | = |
| Adición | + |
| Sustracción | - |
| Multiplicación | \* |
| División | / |
| Negación | ! |
| Módulo (Parte entera de la división) | % |
| Mayor | > |
| Menor | < |
| Mayor o Igual | >= |
| Menor o Igual | <= |

Operadores Lógicos

|  |  |
| --- | --- |
| **Descripción** | **C#** |
| Operador lógico Y | && |
| Operador lógico O | || |
| Negación lógica | ! |
| Igualdad | == |
| Desigualdad | != |

* En C# todas las evaluaciones se hacen por “cortocircuito”

//Si Hacer1() es True, entonces //Si Hacer1() es False, entonces

//NO se evalua Hacer2() //NO se evalua Hacer2()

if (Hacer1() || Hacer2()) if (Hacer1() && Hacer2())

{ {

} }

Estructuras de Control

* + Condicionales
  + Repetitivas

Sentencias Condicionales

* C#: sentencia if con varios formatos

if (x > 10) if (x < 10) if (x < 10) if (x < 10)

HacerAlgo(); { {

Hacer1(); Hacer1(); Hacer1();

Hacer2(); } }

} else else if (x > 20)

{ {

Hacer2(); Hacer2();

} }

else

{

Hacer3();

}

* C#: sentencia switch

int a = 0;

switch (a)

{

case 1:

//CODIGO 1

break;

case 2:

//CODIGO 2

break;

default:

//CODIGO DEFAULT

break;

}

Sentencias Repetitivas

* C#: sentencia **while** C#: sentencia **do-while**

bool condicion = true;

do{

//En algún momento poner condicion = false

} while (condicion == true);

bool condicion = true;

while (condicion == true){

//En algún momento poner condicion = false

}

Entry Point

Primer programa en C#

class HolaMundo

{

static void Main(string[] args)

{

System.Console.WriteLine("Hola mundo C#");

// nombre completamente cualificado.

System.Console.ReadKey();

}

}

* El punto de entrada para los programas en C# es:

**static void Main(string[] args) { … }**

Dónde:

static: Es un modificador que permite ejecutar un método sin tener que instanciar a una variable (sin crear un objeto). El método Main() debe ser estático.

void: Indica el tipo de valor de retorno del método Main(). No necesariamente tiene que ser void.

string [] args: Es un Array de tipo string que puede recibir el método Main() como parámetro. Este parámetro es opcional.

CLASE CONSOLE

Generalidades

* public static class **Console**
* Representa la entrada, salida y errores de Streams para aplicaciones de consola.
* Es miembro del NameSpace **System**.

Métodos

* Clear()
  + Limpia el buffer de la consola. Equivalente a **clrscr()** de C.
* Read()
  + Lee el próximo carácter del stream de entrada. Devuelve un entero.
* ReadKey(bool)
  + Obtiene el carácter presionado por el usuario. La tecla presionada puede mostrarse en la consola. Equivalente a **getch() / getche()** de C.
* ReadLine()
  + Lee la siguiente línea de caracteres de la consola. Devuelve un string.

Equivalente a **gets()** de C.

* Write()
  + Escribe el string que se le pasa como parámetro a la salida estándar.

Equivalente a **printf()** de C.

* WriteLine()
  + Ídem método Write, pero introduce un salto de línea al final de la cadena.

Propiedades

* BackGroundColor
  + Obtiene o establece el color de fondo de la consola.
* ForeGroundColor
  + Obtiene o establece el color del texto de la consola.
* Title
  + Obtiene o establece el título de la consola.

La Clase Console – Formatos

* Console.Write() o Console.WriteLine()
* Con los marcadores (“{}”), además de indicar el número de parámetro que se usará, podemos indicar la forma en que se mostrará.
* Cuantos caracteres se mostrarán y si se formatearán a la derecha o la izquierda o también se pueden indicar otros valores de formato.
* La forma de usar los marcadores o las especificaciones de formato será la siguiente:

{ N [, M ][: Formato ] } (\*)

Dónde:

* **N** será el número del parámetro, empezando por cero.
* **M** será el ancho usado para mostrar el parámetro, el cual se rellenará con espacios. Si M es negativo, se justificará a la izquierda, y si es positivo, se justificará a la derecha.
* **Formato** será una cadena que indicará un formato extra a usar con ese parámetro.

Ejemplo:

Console.WriteLine(“{0,10}{1,-10}{2}”, 10, 15, 23);

Console.WriteLine(“---------- ----------”);

Console.WriteLine(“{0,10:#,###.00}{1,10}”, 10.476, 15.355);

El resultado es:

10 15 23

---------- ----------

10.48 15.355