La Güeb de Joaquín
Mi sitio de Internet

Operadores de Conversión

[Descripción general]

Si hemos definido una clase o estructura, podemos definir un operador de conversión que permita realizar conversiones de tipos entre el tipo que hemos definido y otros tipos de datos (por ejemplo, integer, doublé, string)

Este operador de conversión permite realizar de forma natural operaciones de conversión explicitas e implícitas, tanto de ampliación como de restricción

Contenido

Introducción

Se denomina conversiones al hecho de cambiar el Tipo de dato, por ejemplo, pasar de un tipo Decimal a un Tipo Integer.

Las conversiones pueden realizarse de forma Implícita o explícita, y pueden ser de ampliación o de restricción.

Se puede cambiar el comportamiento del compilador redefiniendo el Operador de conversión CType.

Las conversiones explicitas e implícitas

Observa el siguiente código

```
[1]         Dim i As Integer = 25
[2]         Dim L As Long
[3]         L = i '       Implícita de ampliación
[4]         i = CInt(L) ' Explícita de reduccion
```

La conversión realizada en la línea [3] es una conversión Implícita y además es de ampliación de datos. Una conversión implícita no requiere escribir ningún código extra, solo hay que utilizar el operador de igualdad.

La conversión realizada en la línea [4] es una conversión explicita, y además de restricción. Las conversiones explicitas utilizan SIEMPRE una palabra clave de conversión de tipos

Las conversiones de ampliación y de restricción

Las conversiones de ampliación

Observa el siguiente código

```
Dim i As Integer = 25I
Dim L As Long = 10L
Dim D As Decimal = 50D
' Conversion de Ampliacion
'-----------------------------------------------
' Implícita de ampliación - Long en Decimal
'Siempre se realiza independientemente del valor de [Option Strict]
D = L
'-----------------------------------------------
' Explicta de ampliación - Long en Decimal
' Siempre se realiza independientemente del valor de [Option Strict]
' Realizada con la función de conversión [CDec]
D = CDec(L)
' Realizada con la función de conversión [CType]
D = CType(L, Decimal) ' Explicita
```

En el código que se muestra más arriba, se muestran dos conversiones de ampliación, una realizada implícitamente [D=L] y otra realizada explícitamente utilizando dos funciones de conversión [D = CDec (L)], y [D = CType (L, Decimal)]

Esta conversión se denomina **conversión de ampliación** porque la variable amplia el tamaño del valor que puede contener. (Una variable Decimal puede contener un valor numérico mucho mayor que una variable Long). En otras palabras, una conversión numérica de Ampliación nunca falla, porque por muy grande que sea el número, el contenedor donde va a ir es mayor todavía

Las conversiones de ampliación son siempre satisfactorias en tiempo de ejecución y no provocan nunca pérdida de datos.

Siempre se pueden realizar de forma implícita, independientemente del valor [on],[off] que pueda tener Option Strict (Instrucción)

Como las conversiones de ampliación siempre son satisfactorias, no producen excepciones.

Las conversiones de restricción

Observa el siguiente código

```
Dim i As Integer = 25I
Dim L As Long = 10L
Dim D As Decimal = 50D

'-----------------------------------------------
' Conversion de Restriccion
'-----------------------------------------------
' Implicta de restriccion - Decimal en Long
' + Con Option Strict = Off Se realiza la conversión
' + Con Option Strict = On  El compilador muestra un error
'                           y no permite la conversión
L = D '       Implicta de restriccion - Decimal en Long - Error de conversion

'-----------------------------------------------
' Explicta de restriccion Long en Integer
' Siempre se realiza independientemente del valor de [Option Strict]
' Realizada con la función de conversión [CInt]
i = CInt(L) ' Explicita
' Realizada con la función de conversión [CType]
i = CType(L, Integer) ' Explicita
```

El caso contrario se denomina **conversión de restricción** cuando la variable receptora tiene un rango numérico inferior a la variable Origen, en este caso, puede ocurrir que en la variable receptora no quepa el numero de origen, en este caso se pueden generar errores o perdidas de datos.

Se produce un error si el tipo de datos de destino no puede recibir el valor que se está convirtiendo. Por ejemplo, una conversión numérica puede provocar un desbordamiento.

Importante: El compilador no le permite realizar las conversiones de restricción implícitamente a menos que Option Strict (Instrucción) establezca el modificador de comprobación de tipos en Off.

Únicamente se debe utilizar una conversión de restricción cuando se sabe que el valor de origen se puede convertir en el tipo de datos de destino sin error o pérdida de datos. Por ejemplo, si tiene un tipo de datos String y sabe que contiene "True" o "False", puede utilizar la palabra clave CBool para convertirlo en Boolean.

En las conversiones de restricción, cuando se produce un error, suele ser uno de los siguientes:

- InvalidCastException: si no hay definida ninguna conversión entre los dos tipos
- OverflowException: si el valor convertido es demasiado grande para el tipo de destino

La función CType

Observa el siguiente código

```
Dim i As Integer = 25I
Dim L As Long = 10L
Dim D As Decimal = 50D
'-----------------------------------------------
' Conversion de Restriccion
'-----------------------------------------------
' Implicta de restriccion - Decimal en Long
' + Con Option Strict = Off Se realiza la conversión
' + Con Option Strict = On  El compilador muestra un error
'                           y no permite la conversión
L = D '       Implicta de restriccion - Decimal en Long - Error de conversion

'-----------------------------------------------
' Explicta de restriccion Long en Integer
' Siempre se realiza independientemente del valor de [Option Strict]
' Realizada con la función de conversión [CInt]
i = CInt(L) ' Explicita
' Realizada con la función de conversión [CType]
i = CType(L, Integer) ' Explicita
```

CType (Función) actúa sobre dos argumentos. El primero es la expresión que va a convertirse y el segundo es la clase de objeto o el tipo de datos de destino. Observa que el primer argumento debe ser una expresión, no un tipo.

Los valores utilizados con una palabra clave de conversión deben ser válidos para el tipo de datos de destino; de lo contrario, se produce un error. Por ejemplo, si intenta convertir un tipo Long en Integer, el valor de Long debe estar dentro del intervalo válido para el tipo de datos Integer.

La acción de realizar una conversión explícita se denomina también convertir una expresión en un determinado tipo de datos o clase de objeto.

Clase para ver el ejemplo

```
''' <summary>
'''    Estructura que representa a un numero racional
''' </summary>
''' <remarks>
'''   Un número racional es un número representado por el cociente
'''   de dos números enteros (lo que normalmente llamamos quebrado),
'''   como 5/7. El nuero de la izquierda se denomina numerador y el
'''   de la derecha denominador
''' </remarks>
<Serializable()> _
Public Structure NumeroRacional

#Region "Campos"

    '--------------------------------------------------------
    ' La interfaz de usuario del Numero Racional debe tratar al
    ' número como un objeto indivisible, esto es, el usuario
```

```
    ' no tiene que tener la posibilidad de acceder a los campos
    ' numerador y denominador de forma independiente
    '-----------------------------------------------------------
    Private _numerador As Long
    Private _denominador As Long

    Public ReadOnly Property Valor() As Decimal
        Get
            Return CType(Me._numerador / Me._denominador, Decimal)
        End Get
    End Property


#End Region


#Region "Constructores"

    ''' <summary>
    '''   Costructor con argumentos
    ''' </summary>
    ''' <param name="numerador">El numerador del numero racional</param>
    ''' <param name="denominador">El denominador del numero racional</param>
    ''' <remarks>
    ''' Las operaciones que debe realizar el constructor son:
    ''' Si el denominador es cero, forzar a que sea 1.
    ''' Si el denominador es negativo, invertiremos el signo del
    ''' numerador y del denominador.
    ''' Simplificará la fracción siempre que sea posible
    ''' </remarks>
    Public Sub New(ByVal numerador As Long, ByVal denominador As Long)
        Me._numerador = numerador
        Me._denominador = denominador

        If Me._denominador = 0 Then
            Me._denominador = 1
        End If

        If Me._denominador < 0 Then
            Me._numerador = -Me._numerador
            Me._denominador = -Me._denominador
        End If

        Call Simplificar()

    End Sub

    ''' <summary>
    ''' La función simplificar utiliza el algoritmo de Euclides para
    ''' obtener el máximo común divisor (mcd) del numerador y denominador,
    ''' y simplifica el numero racional dividiendo ambos números por su mcd.
    ''' </summary>
    Private Sub Simplificar()

        ' Calculo del máximo común divisor (mcd)
        Dim mcd As Long = 0L
        Dim temp As Long = 0L
        Dim resto As Long = 0L
        mcd = Math.Abs(Me._numerador)
        temp = Math.Abs(Me._denominador)

        'Algoritmo de Euclides
        Do While (temp > 0)
            resto = mcd Mod temp
            mcd = temp
            temp = resto
        Loop

        'Simplificar
        If mcd > 1 Then
            Me._numerador = CType(Me._numerador / mcd, Long)
            Me._denominador = CType(Me._denominador / mcd, Long)
        End If

    End Sub
```

```
    ''' <summary>
    '''    Constructor de número Entero
    ''' </summary>
    ''' <param name="numerador">Un numero entero</param>
    Public Sub New(ByVal numerador As Long)
        Me._numerador = numerador
        Me._denominador = 1L
    End Sub

#End Region

End Structure
```

El operador de conversión CType

Cuando definimos un Tipo propio, sea una clase o una Estructura, las conversiones entre tipos se complican y el compilador, simplemente no acepta las conversiones.
Vamos a ver como se resuelve este problema:

Conversión de Número Racional a ALGO

Por ejemplo, queremos convertir un número Racional en un Integer
Observa el siguiente código

```
Dim nr As New NumeroRacional(54, 23) '= 24,086956
'-------------------------------------------------
' Conversion de NumeroRacional a ALGO
'-------------------------------------------------
' Implícita
' No funciona, Independientemente del valor de Option Strict
' Solo funciona si Numero Racional define el operador [Narrowing CType]
' + Con Option Strict = Off Se realiza la conversión
' + Con Option Strict = On  El compilador muestra un error
'                           y no permite la conversión
i = nr
'-------------------------------------------------
' Explicta
' No funciona, Independientemente del valor de Option Strict
' Solo funciona si Numero Racional define el operador [Narrowing CType]
' Realizada con la función de conversión [CDec]
i = CInt(nr)
' Realizada con la función de conversión [CType]
i = CType(nr, Integer) ' Explicita
```

En las conversiones implícitas [i = nr], independientemente del valor del [Option Strict], (el compilador no permite hacerlas, genera un error y me dice: 'Un tipo NumeroRacional no se pude convertir en un Integer). La única forma de poder realizar esta conversión es redefiniendo el operador [CType] y poniendo Option Strict a Off

Las conversiones explicitas tampoco funcionan hasta que se redefine el operador [CType], en ese momento se pueden realizar las operaciones independientemente del valor de Option Strict

La función que redefine el operador [CType] es la siguiente:

```
''' <summary>
'''   Convierte una estructura NumeroRacional
'''   en un entero de 32 bits (Integer) con signo.
''' </summary>
''' <param name="value">Estructura NumeroRacional que se va a convertir. </param>
''' <returns> Entero de 32 bits con signo, resultado de la conversión </returns>
''' <remarks>
'''   Narrowing - Indica que un operador de conversión (CType) convierte una clase o
'''   una estructura en un tipo que quizá no pueda incluir algunos de los
'''   valores posibles de la clase o la estructura original.
''' </remarks>
''' <exception cref="System.OverflowException"> value es menor que
''' <seealso cref="Int32.MinValue">Int32.MinValue</seealso> o mayor que
''' <seealso cref="Int32.Maxvalue">Int32.Maxvalue</seealso>.
''' </exception>
Public Shared Narrowing Operator CType(ByVal value As NumeroRacional) As Integer
    Return CType(value.Valor, Integer)
End Operator
```

Observa como hace la conversión:
Primero convierto el número racional en un número decimal con la propiedad Valor. Por ejemplo 54/23 = 24,086956, y después utilizo el operador CType para convertir explícitamente este número decimal en un numero Integer, el resultado de la conversión será = 24, y ese valor es el que se devuelve

Es una conversión de restricción, se utiliza la palabra Narrowing que indica que puede haber pérdidas de información en el proceso de conversión, en este caso se ve claramente que se pierden los valores decimales del numero.

Si ahora corremos el código inicial vemos que funciona perfectamente, porque ahora el compilador sabe cómo hay que hacer la conversión.

Conversión de ALGO a Número Racional

Por ejemplo, queremos convertir un Integer en un número Racional
Observa el siguiente código

```
Dim nr As New NumeroRacional(54, 23) '= 24,086956
' ------------------------------------------------
' Conversion de ALGO a NumeroRacional
'-------------------------------------------------
' Implicta
' No funciona, Independientemente del valor de Option Strict
' Solo funciona si Numero Racional define e operador [Widening CType]
nr = i

'-------------------------------------------------
' Explicta
' No funciona, Independientemente del valor de Option Strict
' Solo funciona si Numero Racional define e operador [Widening CType]
'
' Realizada con la función de conversión [CInt]
nr = CInt(i) ' Explicita
' Realizada con la función de conversión [CType]
nr = CType(i, Integer) ' Explicita
```

En las conversiones implícitas, independientemente del valor del [Option Strict], (el compilador no permite hacerlas, genera un error y me dice: 'Un tipo [Integer] no se pude convertir en un [NumeroRacional].). La única forma de poder realizar esta conversión es redefiniendo el operador [CType] y poniendo Option Strict a Off

Las conversiones explicitas tampoco funcionan hasta que se redefine el operador [CType]. En ese momento se pueden realizar las operaciones independientemente del valor de Option Strict

Para resolver ese problema tengo que sobrecargar (otra vez) el operador CType de la siguiente manera:

```
''' <summary>
''' Convierte un entero de 32 bits con signo en una estructura NumeroRacional.
''' </summary>
''' <param name="value">Entero de 32 bits con signo</param>
''' <returns>una estructura NumeroRacional</returns>
''' <remarks>
'''   Widening - Indica que un operador de conversión (CType) convierte una clase o
'''   una estructura en un tipo que puede incluir todos los valores posibles
'''   de la clase o la estructura original.
'''</remarks>
Public Shared Widening Operator CType(ByVal value As Integer) As NumeroRacional
    Return New NumeroRacional(CType(value, Long))
End Operator
```

Observa que lo que hace es utilizar el constructor de [NumeroRacional] que admite números del tipo Long. La función CType, convierte el número Integer en un numero Long para que lo acepte el constructor.

Es una conversión de ampliación, Además, usa la palabra reservada Widening porque en este caso no se va a perder ninguna información en el camino (es una conversión de ampliación)

Si ahora corremos el código inicial vemos que funciona perfectamente, porque ahora el compilador sabe cómo hay que hacer la conversión.

Conversiones que no son posibles ni deseables

Existen algunas conversiones que no son posibles, por ejemplo convertir una fecha a un numero racional y viceversa. En estos casos hay que definir el operador Ctype de forma que genere un error

```
''' <summary>
'''   [NO SOPORTADO]  Esta conversion no es posible
''' </summary>
Public Shared Narrowing Operator CType(ByVal value As Date) As NumeroRacional
    Using SW As New System.IO.StringWriter( _
                    System.Globalization.CultureInfo.CurrentCulture)
        SW.WriteLine("Estructura NumeroRacional")
        SW.WriteLine("Un valor[Date], [" & value & "] no puede ")
        SW.WriteLine("ser transformado en una Estructura NumeroRacional [a/b]")
        SW.WriteLine("Utiliza en su lugar un valor numerico de tipo [System.Decimal] ")
        SW.Flush()
        Throw New System.InvalidCastException(SW.ToString)
    End Using
End Operator
```

```
''' <summary>
'''   [NO SOPORTADO] Convierte una Estructura NumeroRacional en una fecha -
'''   Esta conversion no es posible
''' </summary>
''' <param name="value">una Estructura NumeroRacional
'''         que se va a convertir. </param>
''' <returns>Una Exception <seealso cref="System.OverflowException">
'''          OverflowException</seealso> Esta conversion no es posible. </returns>
''' <exception cref="System.OverflowException">OverflowException-
'''       Esta conversion no es posible.
''' </exception>
Public Shared Narrowing Operator CType(ByVal value As NumeroRacional) As Date
    Throw New System.InvalidCastException("Esta conversion no es posible")
End Operator
```

En otras ocasiones no me interesa que se realice la conversión aunque esta sea posible. Por ejemplo, existen problemas de redondeo con los números de coma flotante y se pierden decimales cuando se transforman, (por ejemplo un tipo Double en un Tipo Decimal). Podemos estar trabajando en un entrono en el que no quiero que esto suceda, y por lo tanto impido que se pueda realizar la conversión.

En el ejemplo siguiente se ilustra este concepto, un número decimal no se puede convertir en un número racional (porque a mí no me interesa), y sin embargo se permite la conversión de un número racional en decimal.

```
''' <summary>
'''   [NO SOPORTADO] Convierte un número de punto flotante de precisión doble [Double]
'''   en una Estructura NumeroRacional Esta conversion no es posible
'''   por problemas de redondeo de cantidades, utiliza en su lugar un valor [Decimal]
''' </summary>
''' <param name="value">un valor Double</param>
''' <returns>Una Exception  InvalidCastException</returns>
''' <exception cref="System.InvalidCastException">InvalidCastException-
'''       Esta conversion no es posible.
''' </exception>
''' <remarks>
'''   Narrowing - Indica que un operador de conversión (CType) convierte una clase o
'''   una estructura en un tipo que quizá no pueda incluir algunos de los
'''   valores posibles de la clase o la estructura original.
''' </remarks>
Public Shared Narrowing Operator CType(ByVal value As Double) As NumeroRacional
    Using SW As New System.IO.StringWriter( _
                    System.Globalization.CultureInfo.CurrentCulture)
     SW.WriteLine("Estructura NumeroRacional")
     SW.WriteLine("El valor de punto flotante [System.Double][" & value & "] no puede ")
     SW.WriteLine("ser una Estructura NumeroRacional por problemas de ")
     SW.WriteLine("redondeo de cantidades ")
     SW.WriteLine("Utiliza en su lugar un valor numerico de tipo [System.Decimal] ")
     SW.Flush()
     Throw New System.InvalidCastException(SW.ToString)
    End Using
End Operator


''' <summary>
'''   Convierte una Estructura NumeroRacional
'''   en un número de punto flotante de precisión doble en un Double.
''' </summary>
''' <param name="value">una Estructura NumeroRacional </param>
''' <returns> Número de punto flotante de precisión doble.
'''          <seealso cref="System.Decimal">Double</seealso>,
'''          que representa una Estructura NumeroRacional convertido. </returns>
''' <remarks>
'''  Esta operación puede producir errores de redondeo porque un número
'''  de punto flotante de precisión simple tiene menos dígitos significativos
'''  que el valor Long subyacente en la estructura NumeroRacional.
'''
'''   Narrowing - Indica que un operador de conversión (CType) convierte una clase o
'''   una estructura en un tipo que quizá no pueda incluir algunos de los
'''   valores posibles de la clase o la estructura original.
''' </remarks>
```

```
''' <exception cref="System.OverflowException">OverflowException-  value es menor que
''' <seealso cref="Double.MinValue">doble.MinValue</seealso> o mayor que
''' <seealso cref="Double.Maxvalue">doble.Maxvalue</seealso>.  O bien, value es
''' <seealso cref="Double.NaN">Double.NaN</seealso>,
''' <seealso cref="Double.PositiveInfinity">Double.PositiveInfinity</seealso> o
''' <seealso cref="Double.NegativeInfinity">Double.NegativeInfinity</seealso>.
''' </exception>
Public Shared Narrowing Operator CType(ByVal value As NumeroRacional) As Double
    Return CType(value.Valor, Double)
End Operator
```

A modo de resumen

Las conversiones de restricción no son siempre satisfactorias en tiempo de ejecución y pueden generar errores o provocar pérdida de datos. Se produce un error si el tipo de datos de destino no puede recibir el valor que se está convirtiendo. Por ejemplo, una conversión numérica puede provocar un desbordamiento.

Las conversiones de ampliación son siempre satisfactorias en tiempo de ejecución y no provocan nunca pérdida de datos. Siempre puede realizarlas implícitamente, independientemente de si Option Strict (Instrucción) establece el modificador de comprobación de tipos en On o en Off.

El compilador no le permite realizar las conversiones de restricción implícitamente a menos que Option Strict (Instrucción) establezca el modificador de comprobación de tipos en Off.

Si estamos definiendo un tipo numérico o que contiene un valor numérico, como es el caso de una estructura que contiene un numero Racional (un quebrado de toda la vida), es conveniente redefinir los operadores de conversión explicita e implícita para evitar errores, perdidas inoportunas de información y facilitarnos el uso natural del 'nuevo' número

Más información:

¿Qué es un número Racional?

Un número racional es un número representado por el cociente de dos números enteros (lo que normalmente llamamos quebrado), como 5/7. El número de la izquierda se denomina numerador y el de la derecha denominador

Una clase que envuelva un número racional es útil porque muchos de estos números NO pueden ser representados exactamente utilizando el tipo Double.
Por ejemplo Realizamos una operación:

- Con Números Racionales 1/3 + 1/3+ 1/3 =1,00
- Con Números Double 0,33333+ 0,33333 + 0,33333 = 0,99999.
- Puedes observar que hay una diferencia en el sexto decimal, que la representación del valor no es exacta con valores Double.

Código de ejemplo

En el siguiente enlace hay un fichero ZIP que contiene ejemplo

- Fichero con el código de ejemplo: [2008_09_27_NumeroRacionalEstructura.zip]
- MD5 checksum: [B6680B49C257AC47F6F11B4423C522D0]
- MD5 checksum: Información
  - [http://www.elguille.info/colabora/MD5_checksum.aspx]

Bibliografía

- *Definición de un numero racional copiada de*
  Java 2 Curso de Programación Pág. 306
  Fco. Javier Ceballos Sierra
  ISBN: 84-7897-430-X
  RA MA 2000

Información de este Documento

[Historial de este documento]

- *Autor*:
  - *Nombre*: Joaquín Medina Serrano
  - *Email* : joaquin@medina.name
  - *Web*: http://joaquin.medina.name

- *Publicador*:
  - *Nombre*: Joaquín Medina Serrano

- *Fechas* (Formato de fecha ISO 8610:2004. [yyyy-MM-ddThh:mm:ss])
  - *Fecha de Creación*: 2008-09-14T12:20
  - *Fecha de la última modificación*: 2008-10-09T10:03
  - *Fecha de Impresión*: 2008-10-09T10:03

- *Copyright*
  -

[Direcciones Web Interesantes]

Más Información sobre números racionales:

- Wiki - Números Racionales
  - [http://es.wikipedia.org/wiki/N%C3%BAmero_racional]
- Wiki -Categoría: Fracciones
  - [http://es.wikipedia.org/wiki/Categor%C3%ADa:Fracciones]
- Tipos de números Racionales
  - [http://platea.pntic.mec.es/anunezca/ayudas/raiz_de_2_irracional/r_irracional.htm]

Guía de programación de Visual Basic
[http://msdn.microsoft.com/es-es/library/y4wf33f0.aspx]

- Cómo: Definir un operador de conversión
  - [http://msdn.microsoft.com/es-es/library/yf7b9sy7.aspx]
- Conversiones de ampliación y de restricción
  - http://msdn.microsoft.com/es-es/library/k1e94s7e.aspx
- Conversiones implícitas y explícitas
  - http://msdn.microsoft.com/es-es/library/kca3w8x6.aspx

Biblioteca de clases de .NET Framework
[http://msdn.microsoft.com/es-es/library/ms229335.aspx]

- Boolean (Estructura) [http://msdn.microsoft.com/es-es/library/system.boolean.aspx]
- SByte (Estructura)[http://msdn.microsoft.com/es-es/library/system.sbyte.aspx]
- Byte (Estructura) [http://msdn.microsoft.com/es-es/library/system.byte.aspx]
- Int16 (Estructura) [http://msdn.microsoft.com/es-es/library/system.int16.aspx]
- UInt16 (Estructura) [http://msdn.microsoft.com/es-es/library/system.uint16.aspx]
- Int32 (Estructura) [http://msdn.microsoft.com/es-es/library/system.int32.aspx]
- UInt32 (Estructura) [http://msdn.microsoft.com/es-es/library/system.uint32.aspx]
- Int64 (Estructura) [http://msdn.microsoft.com/es-es/library/system.int64.aspx]
- UInt64 (Estructura) [http://msdn.microsoft.com/es-es/library/system.uint64.aspx]
- Single (Estructura) [http://msdn.microsoft.com/es-es/library/system.single.aspx]
- Double (Estructura) [http://msdn.microsoft.com/es-es/library/system.double.aspx]
- Decimal (Estructura) [http://msdn.microsoft.com/es-es/library/system.decimal.aspx]
- DateTime (Estructura) [http://msdn.microsoft.com/es-es/library/system.datetime.aspx]
- Char (Estructura) [http://msdn.microsoft.com/es-es/library/system.char.aspx]
- String (Clase) [http://msdn.microsoft.com/es-es/library/system.string.aspx]

- OverflowException (Clase)
  - [http://msdn.microsoft.com/es-es/library/system.overflowexception.aspx]
- InvalidCastException (Clase)
  - [http://msdn.microsoft.com/es-es/library/system.invalidcastexception.aspx]

Referencia del lenguaje Visual Basic
[http://msdn.microsoft.com/es-es/library/sh9ywfdk.aspx]

- Option Strict (Instrucción)
  - [http://msdn.microsoft.com/es-es/library/zcd4xwzs.aspx]
- CType (Función)
  - [http://msdn.microsoft.com/es-es/library/4x2877xb.aspx]
- Operator (Instrucción)
  - [http://msdn.microsoft.com/es-es/library/hddt295a.aspx]

[Palabras Clave]

- ‘Conversión de tipos’, ‘Función CType’, ‘Operador CType’, explicitas, ‘conversiones explicitas’, implícitas, ‘conversiones implícitas’, operadores, ‘operadores de conversión’, restricción, ‘conversiones de restricción’, ‘conversiones de ampliación’,

[Grupo de documentos]

- [Documento Index]
- [Documento Start]