leren. durven. doen.

SYNTRA WEST

Programmeren 1 C#

DATA TYPE CONVERSIE

west-vlaanderen POM

Overzicht C# Data Types

Type	Represents	Range	Default Value
bool	Boolean value	True or False	False
byte	8-bit unsigned integer	0 to 255	0
char	16-bit Unicode character	U +0000 to U +ffff	'\0'
decimal	128-bit precise decimal values with 28-29 significant digits	$(-7.9 \times 10^{28}$ to $7.9 \times 10^{28}) / 10^{0 \text{ to } 28}$	0.0M
double	64-bit double-precision floating point type	$(+/-)5.0 \times 10^{-324}$ to $(+/-)1.7 \times 10^{308}$	0.0D
float	32-bit single-precision floating point type	-3.4×10^{38} to $+ 3.4 \times 10^{38}$	0.0F
int	32-bit signed integer type	-2,147,483,648 to 2,147,483,647	0
long	64-bit signed integer type	-923,372,036,854,775,808 to 9,223,372,036,854,775,807	0L
sbyte	8-bit signed integer type	-128 to 127	0
short	16-bit signed integer type	-32,768 to 32,767	0
uint	32-bit unsigned integer type	0 to 4,294,967,295	0
ulong	64-bit unsigned integer type	0 to 18,446,744,073,709,551,615	0
ushort	16-bit unsigned integer type	0 to 65,535	0

De built-in primitieve C# types zijn synoniemen voor de voorgedefinieer de types in .Net System namespace.

C# type	.NET type
bool	System.Boolean
byte	System.Byte
sbyte	System.SByte
char	System.Char
decimal	System.Decimal
double	System.Double
float	System.Single
int	System.Int32
uint	System.UInt32
long	System.Int64
ulong	System.UInt64
object	System.Object
short	System.Int16
ushort	System.UInt16
string	System.String

Data Type conversie

Data Type Conversie

Is enkel mogelijk indien de 2 data types niet te veel van elkaar verschillen, bv

- conversie tussen int en long is mogelijk
- conversie van decimal naar bool ????

Er zijn 2 mogelijke manieren om een Data type naar een ander te converteren, namelijk via:

- Implicite conversie: dit gebeurt automatisch
- Expliciete conversie: hiervoor moet de programmeur code voorzien

Impliciete Data type conversie

De compiler converteert automatisch short naar int.
Dit heet verbreding(widening)

```
class Program
{
    static void Main(string[] args)
    {
        short eersteGetal = 5;
        short tweedeGetal = 10;
        int resultaat = eersteGetal + tweedeGetal;

        Console.WriteLine("{0} + {1} = {2}",
         eersteGetal, tweedeGetal, resultaat);
        Console.ReadKey();
    }
}
```

Impliciete Data type conversie

De reden hiervoor is dat er geen dataverlies kan voorkomen bij het omzetten van het ene data type naar het andere.

"Bredere" types (met groter bereik) kunnen steeds impliciet worden omgezet naar "Smallere" types (met kleiner bereik, indien dit bereik volledig binnen dat van het ander type ligt)

De compiler verbreedt impliciet elke int naar een long

d.i. *"upward cast" (widening):*

```
int i = 5;
long l = i;
```

Type conversies zonder gegevensverlies

Type	Can be converted without data loss to
Byte	UInt16 , Int16, UInt32, Int32, UInt64, Int64, Single, Double, Decimal
SByte	Int16 , Int32, Int64, Single, Double, Decimal
Int16	Int32 , Int64, Single, Double, Decimal
UInt16	UInt32 , Int32, UInt64, Int64, Single, Double, Decimal
Char	UInt16 , UInt32, Int32, UInt64, Int64, Single, Double, Decimal
Int32	Int64 , Double, Decimal
UInt32	Int64 , UInt64, Double, Decimal
Int64	Decimal
UInt64	Decimal
Single	Double

Deze data types kunnen impliciet door de compiler geconverteerd worden

Type	Can be converted to
Int32	Single
UInt32	Single
Int64	Single , Double
UInt64	Single , Double
Decimal	Single , Double

Data type – Expliciete conversie

Expliciete conversie gebeurt manueel

Expliciete cast is mogelijk via c# casting operator ()

Noodzakelijk bij de mogelijkheid van dataverlies of preciesie bij de conversie: Bv1: van long naar int conversie met **(int)** casting:

```
long l = 5;
int i = (int) l;
```

Code voorbeeld:

van int naar byte conversie met `(byte)` casting

```
static void Main(string[] args)
 {
      byte myByte = 0;
      int myInt = 200;
    // Expliciete cast de int naar een byte (geen dataverlies):
      myByte = (byte)myInt;
      Console.WriteLine("waarde van myByte: {0}", myByte);
}
```

Data type conversie –Expliciete cast

Een expliciete cast laat om conversie (narrowing) te doen, zelfs als er verlies is van data.

```
float heightInMeters = 1.74f; // Expliciete conversie

double maxHeight = heightInMeters; // Impliciet

double minHeight = (double) heightInMeters; // Expliciet

float actualHeight = (float) maxHeight; // Expliciet

float maxHeightFloat = maxHeight; // compilatiefout!
```

Nota:

C# voorziet de keywords *checked* en *unchecked* om na te gaan of er data verlies is. (wordt weinig gebruikt in de praktijk)

Data type conversie: check overflow na casting

```
static void ProcessBytes()
{
    byte b1 = 100; byte b2 = 250;
    byte sum = (byte)Add(b1, b2);
    // sum should hold the value 350, but
    // we find the value 94!
    Console.WriteLine("sum = {0}", sum);
    Console.ReadKey();
}

static int Add(int x, int y)
{
    return x + y;
}
```

```
 static void ProcessBytes()
{
    byte b1 = 100;
    byte b2 = 250;
    // This time, tell the compiler to throw an
    //exception if there is overflow/underflow
    try
    {
        byte sum = checked((byte)Add(b1, b2));
        Console.WriteLine("sum = {0}", sum);
    }
    catch (OverflowException ex)
    {
        Console.WriteLine(ex.Message);
    }
    Console.ReadKey();
}
```

Conversie naar String

Het converteren van en naar het type String is niet mogelijk via de typecasting operator ()

Om te converteren **naar** een string bestaat er een ingebouwde methode (**ToString()**), die voor alle .NET data types beschikbaar is.

Een voorbeeld:

Data type conversie

System.Convert class

Taal-neutrale manier om conversies te doen (conversie syntax van vb.net is verschillend van c#)

Voor de manier waarop er wordt geconverteerd is de keuze aan de programmeur: via casting of System.Convert

```
String aantalText= "45000";
Int32 aantalInt = Convert.ToInt32(aantalText);
Console.WriteLine("Text omzetten tot Int32: {0}",
aantalInt);
```

Impliciet getypeerde variabelen

var

var: impliciet getypeerde locale variabelen

`var` keyword kan voor declaratie van een lokale variabele worden gebruikt i.p.v. om het even wel datatype (int,bool,string,..).

De compiler zal dan dit dan **automatisch omzetten naar het data type van de initiële waarde** waarmee de lokale variabele werd geïnitialiseerd.

impliciete typering van locale variabelen resulteert in **sterk getypeerde data** (dus, de var bij C# is niet vergelijkbaar met de var van Javascript(zwak getypeerd)

Bv:

```
var myInt = 0;
var myBool = true;
var myString = "Time, marches on...";
```

Nut van impliciete type variabele (var)

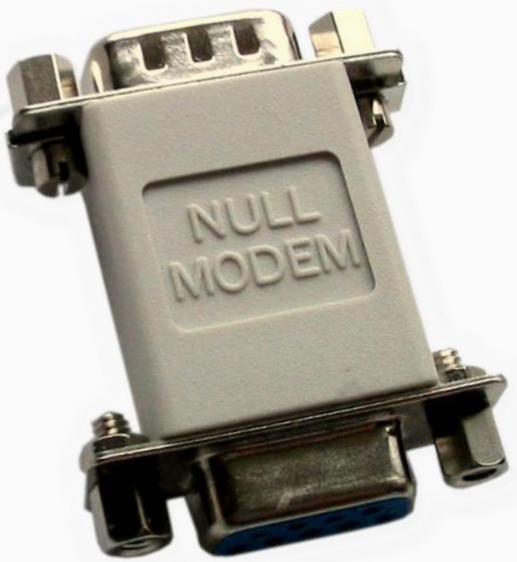
Good practice: Gebruik steeds wanneer mogelijk het data type (bv int, bool,…) indien je weet dat je dit datatype nodig hebt

Wanneer var declaratie gebruiken?

Bij LINQ (Language Integrated Query) technologie wordt gebruik gemaakt van query expressions die dynamisch gecreeerde resultaten kunnen opleveren gebasserd op de query zelf.(data type is soms moeilijk of zelfs onmogelijk te achterhalen)

In dit geval is *implicit typing (var declaratie)*extreem hulpvol (gebruik dan var als declaratie voor variabele) (In latere modules komt LINQ aan bod)

Nullable Types



Nullable Types

`Nullable` types zijn instanties van `System.Nullable`

=Wrapper over de primitive datatypes

Voorbeelden `int?`, `double?` `,...`

`Nullabe` types kunnen naar het gespecifieerde datatype ook een ***null*** waarde aannemen.

De nullable types zijn vooral nuttig bij het aanspreken van o.a. `Databases` en andere datastructuren die als standaard waarde `null` bevatten.

Nullable Types Voorbeeld

- Voorbeeld met geheel getal `int`:

```
int? someInteger = null;
Console.WriteLine("int met Null waarde ->" + someInteger);
someInteger = 5;
Console.WriteLine("int met waarde 5 ->" +  someInteger);
```

- Voorbeeld met geheel getal `double`:

```
double? someDouble = null;
Console.WriteLine("kommagetal met null waarde ->" +
someDouble);
someDouble = 2.5;
Console.WriteLine("kommagetal met waarde 2.5->" +
someDouble);
```

leren. durven. doen.

SYNTRA WEST

Vragen?

west-vlaanderen POM

leren. durven. doen.

SYNTRA WEST

REFERENTIES

PRO C# 7 WITH .NET AND .NET CORE – ANDREW TROELSEN – PHILIP JAPIKSE

HTTPS://WWW.LEARNCS.ORG/

FUNDAMENTALS OF COMPUTER PROGRAMMING WITH C#

© SVETLIN NAKOV & CO.

west-vlaanderen POM