

Programmeren 1 C#

DATATYPES

Data Type - Karakteristieken

Data type heeft:

- Naam (C# keyword or .NET type)
- Grootte (hoeveel geheugen wordt voorzien)
- Default waarde

Bijvoorbeeld:

Geheel getal in C#

Naam datatype: `int`

Grootte: 32 bits (4 bytes)

Default waarde: 0

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

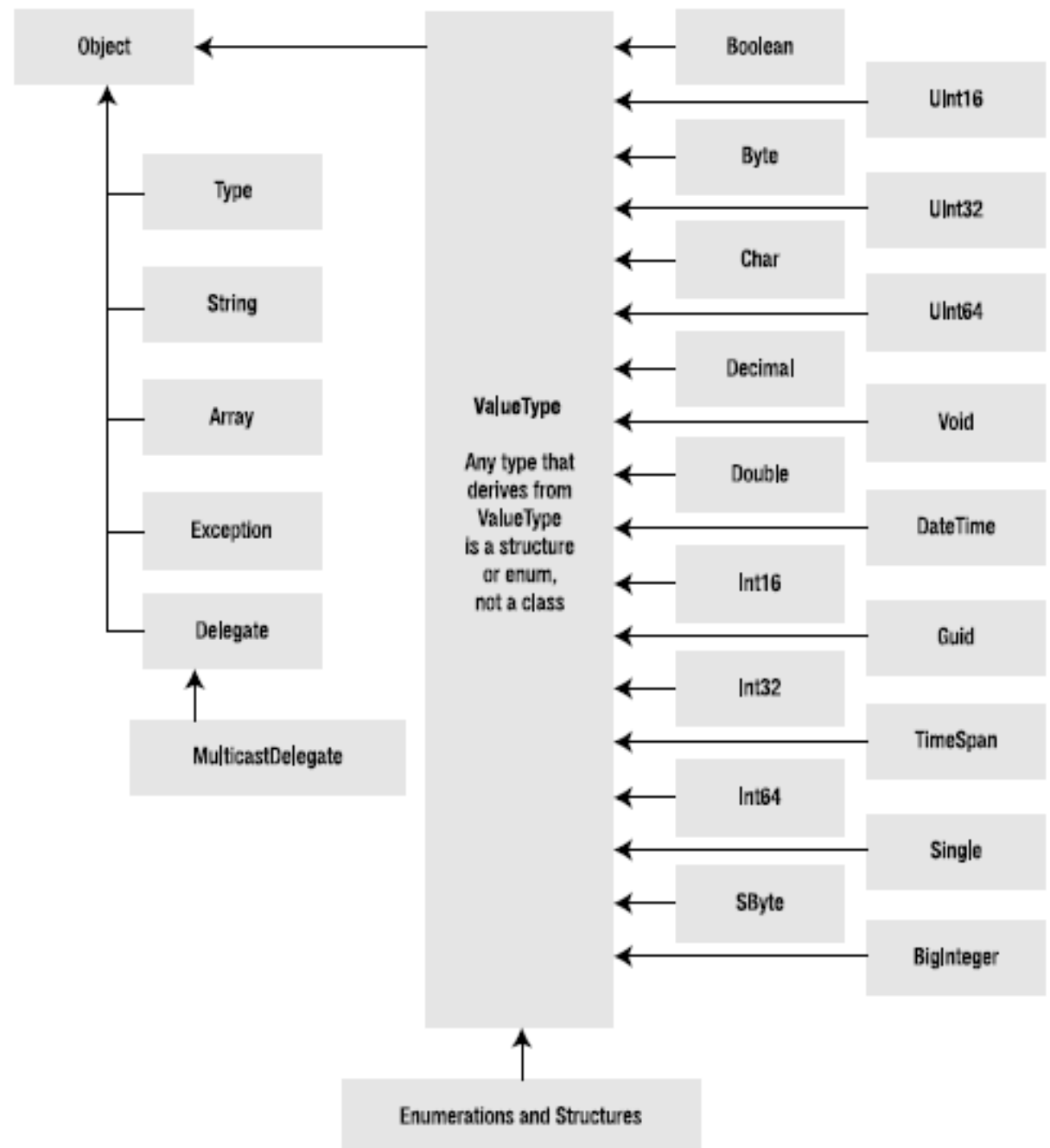
C# type	.NET type
<u>bool</u>	<u>System.Boolean</u>
<u>byte</u>	<u>System.Byte</u>
<u>sbyte</u>	<u>System.SByte</u>
<u>char</u>	<u>System.Char</u>
<u>decimal</u>	<u>System.Decimal</u>
<u>double</u>	<u>System.Double</u>
<u>float</u>	<u>System.Single</u>
<u>int</u>	<u>System.Int32</u>
<u>uint</u>	<u>System.UInt32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

Data types Overzicht en Hierarchie

Zelfs de primitive
.NET data types zijn
geregeld in een
classe hiërarchie.

Alle types erven van
system.Object

Bevat o.A. Methodes `tostring()`,
`equals()`



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	$(-7.9 \times 10^{28} \text{ to } 7.9 \times 10^{28}) / 10^0 \text{ to } 28$	0.0M
double	64-bit double-precision floating point type	$(+/-)5.0 \times 10^{-324} \text{ to } (+/-)1.7 \times 10^{308}$	0.0D
float	32-bit single-precision floating point type	$-3.4 \times 10^{38} \text{ to } +3.4 \times 10^{38}$	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

DataType string

Werken met String Data

Voorbeeld:

```
string firstName, lastName;  
firstName = "Jan";  
lastName = "Jansen";
```

```
//Example of string concatenation
```

```
string fullName = firstName + " " + lastName;
```

```
Console.WriteLine("Volledige naam: {0}", fullName);
```

Escape Characters

Hoe data moet geprint worden naar output stream

Elke escape character begint met een backslash, gevolgd met een specifiek teken.

Voorbeelden:

```
//Example of backslash in a string literal
```

```
Console.WriteLine("C:\\MyApp\\bin\\Debug");
```

```
//Example of double quote in a string literal
```

```
string message = "Last night I dreamt of \"San Pedro\"";
```

```
Console.WriteLine(message);
```

```
string cityName = message.Substring(23);
```

```
Console.WriteLine(cityName);
```


Overzicht Escape sequences

Escape Sequence	Name	Description
\a	Bell (alert)	Makes a sound from the computer
\b	Backspace	Takes the cursor back
\t	Horizontal Tab	Takes the cursor to the next tab stop
\n	New line	Takes the cursor to the beginning of the next line
\v	Vertical Tab	Performs a vertical tab
\f	Form feed	
\r	Carriage return	Causes a carriage return
\"	Double Quote	Displays a quotation mark (")
\'	Apostrophe	Displays an apostrophe (')
\?	Question mark	Displays a question mark
\\	Backslash	Displays a backslash (\)
\0	Null	Displays a null character

String en Gelijkheid

String equality is case-sensitive

- "Hello!" <> "HELLO!"

```
static void StringEquality()
{
    Console.WriteLine("=> String Gelijkheid:");
    string s1 = "Hello!";
    string s2 = "Yo!";
    Console.WriteLine("s1 = {0}", s1);
    Console.WriteLine("s2 = {0}", s2);
    Console.WriteLine();

    // Test these strings for equality.
    Console.WriteLine("s1 == s2: {0}", s1 == s2);
    Console.WriteLine("s1 == Hello!: {0}", s1 == "Hello!");
    Console.WriteLine("s1 == HELLO!: {0}", s1 == "HELLO!");
    Console.WriteLine("s1 == hello!: {0}", s1 == "hello!");
    Console.WriteLine("s1.Equals(s2): {0}", s1.Equals(s2));
    Console.WriteLine("Yo.Equals(s2): {0}", "Yo!".Equals(s2));
    Console.WriteLine();
}
```

Werken met String Data

System.String

String methoden

- Length *(geeft de lengte van string)*
- CompareTo() *(vergelijkt twee strings)*
- Contains() *(controleert of string een bepaalde substring bevat)*
- Equals() *(controleert of twee strings identieke data bevatten)*
- ToUpper() *(zet de string in uppercase)*
- ToLower() *(zet de string in lowercase)*
- SubString() *(om een stukje string uit een andere string te extracten)*

Oefeningen op Strings

1. Schrijf een programma dat de volgende getallen naar de console schrijft (elk getal op een nieuwe lijn:
5 6 3 11 8 13

2.A Schrijf een programma dat de huidige datum naar de console schrijft (bv 11/06/2019)

2. B Extract het jaartal uit de vorige datum

Strings zijn Immutable

Wat kunnen we hieruit leren?

- String class kan inefficiënt zijn en resulteren in opgeblazen code bij slecht gebruik
 - Zeker bij gebruik van *Concatenation*!
- Wil je een simpele data characters representeren zoals voornaam, familienaam,... is string class een perfecte keuze
- Vermijd gebruik bij textuele data die ook nog gewijzigd moet worden, leidt tot onnodige kopies!

StringBuilder type

In .NET bestaat er ook de klasse `StringBuilder`.

Deze is te vinden in de namespace `System.Text` en kan gebruikt worden door met volgend using statement:

```
using System.Text;
```

Bij het oproepen van members van dit type, wijzig je rechtstreeks het object intern character data.

Waarden kunnen gegeven worden via de verschillende *constructors* (later meer over constructors!)

StringBuilder type

```
static void FunWithStringBuilder()
{
    Console.WriteLine("=> Using the StringBuilder:");
    StringBuilder sb = new StringBuilder("**** Games ****");
    sb.Append("\n");
    sb.AppendLine("Half Life");
    sb.AppendLine("Beyond Good and Evil");
    sb.AppendLine("Deus Ex 2");
    sb.AppendLine("System Shock");
    Console.WriteLine(sb.ToString());
    sb.Replace("2", "Invisible War");
    Console.WriteLine(sb.ToString());
    Console.WriteLine("sb has {0} chars.", sb.Length);
    Console.WriteLine();
}
```

Data types

Integer Data Type

Wat zijn de integer types?

Integer types:

Stellen gehele getallen voor

Kunnen signed (met teken, ook negative) of unsigned (enkel positieve) zijn

Het bereik (min en max waarden) hangt af van de hoeveelheid gereserveerde geheugenruimte

De default waarde van integer types:

0 – voor integer types, behalve

0L – voor het **long** type



Overzicht Integer Types

De Integer types:

sbyte (-128 to 127): signed 8-bit

byte (0 to 255): unsigned 8-bit

short (-32,768 to 32,767): signed 16-bit

ushort (0 to 65,535): unsigned 16-bit

int (-2,147,483,648 to 2,147,483,647): signed 32-bit

uint (0 to 4,294,967,295): unsigned 32-bit

Integer types – Voorbeeld

Afhankelijk van het nodige bereik, wordt het gepaste Integer datatype gekozen:

```
// Declare some variables
byte centuries = 20;
ushort years = 2000;
uint days = 730480;
ulong hours = 17531520;
// Print the result on the console
Console.WriteLine(centuries + " centuries are " + years
+ " years, or " + days + " days, or " + hours + "
hours.");
// Console output:
// 20 centuries are 2000 years, or 730480 days, or
17531520
// hours.
ulong maxIntValue = UInt64.MaxValue;
Console.WriteLine(maxIntValue); // 18446744073709551615
```

Data types

Floating-Point en Decimal Types

Wat zijn Floating-Point Types?

Floating-point types:

- Zijn presentaties van reële getallen (komma getallen)
- Kunnen signed (met teken) of unsigned zijn
- Hebben een bepaald bereik en verschillende precisie (aantal cijfers na de komma), afhankelijk van de hoeveelheid gereserveerde geheugen
- Kunnen abnormale resultaten geven in berekeningen!

Floating-Point Types (komma-getallen)

Floating-point types:

`float` ($\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$): 32-bits, precisie van 7 cijfers

`double` ($\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$): 64-bits, precisie van 15-16 cijfers

De default value van floating-point types:

`0.0F` voor `float` type

`0.0D` voor `double` type

Precisie van PI – Voorbeeld

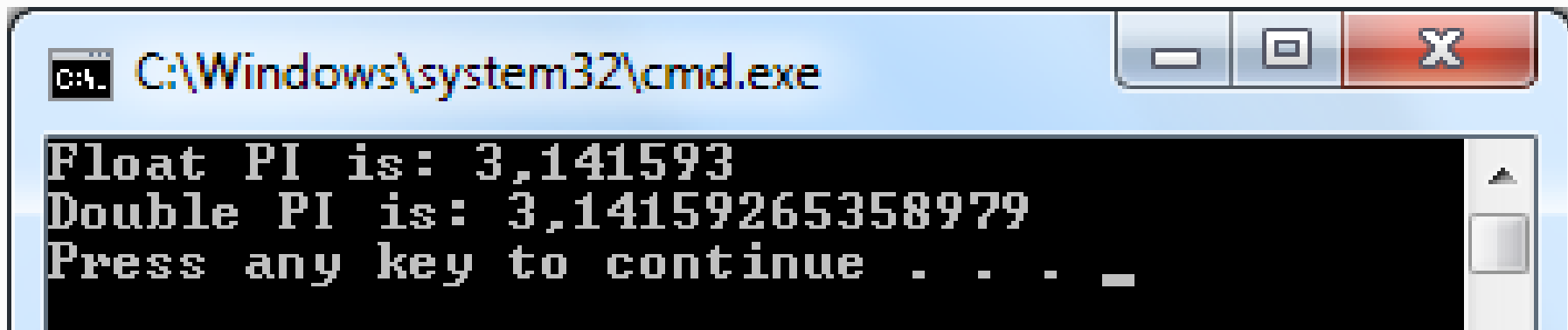
Hieronder zie je het verschil in precisie tussen **float** en **double**:

```
float floatPI = 3.141592653589793238f;  
double doublePI = 3.141592653589793238;  
Console.WriteLine("Float PI is: {0}", floatPI);  
Console.WriteLine("Double PI is: {0}", doublePI);
```

Bemerk de suffix "**f**" in de eerste lijn.

Reële getallen worden standaard als **double** Geïnterpreteerd!

Deze moet **expliciet** geconverteerd worden naar **float**



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed in white text on a black background:

```
Float PI is: 3,141593  
Double PI is: 3,14159265358979  
Press any key to continue . . . _
```

Abnormale resultaten bij Floating-Point berekeningen

Soms zijn er abnormale resultaten bij het gebruik van floating-point getallen

Het vergelijken van komma getallen kan niet met de `==` operator

Voorbeeld:

De suffix `f` is voor float types, maar de compiler aanvaardt dat je deze aan het type `double` toekent, want `double` heeft grotere precisie dan `float`. Er zijn echter precisie-problemen wanneer je deze dan optelt, waardoor onderstaande vergelijking `false` teruggeeft:

```
double a = 1.0f;
double b = 0.33f;
double sum = 1.33f;
bool equal = (a + b == sum); // False!!!
Console.WriteLine("a+b={0}    sum={1}    equal={2}",
    a+b, sum, equal);
```


Decimal Type (komma-getallen)

Er is een decimal type in C#:

decimal ($\pm 1,0 \times 10^{-28}$ to $\pm 7,9 \times 10^{28}$): 128-bits, precisie van 28-29 cijfers

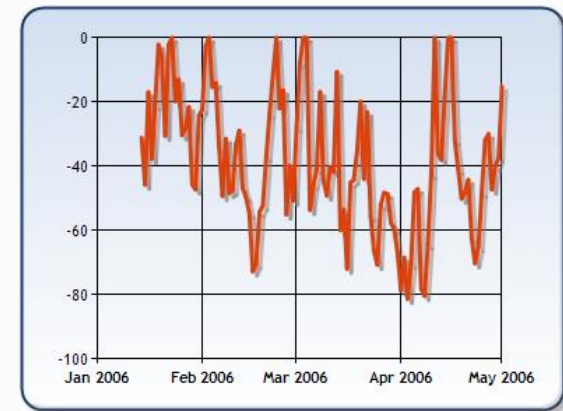
Wordt gebruikt voor wetenschappelijke berekeningen

Geen afrondingsfouten

Bijna geen verlies van precisie

De standaard waarde van een **decimal** type is:

0.0M (**M** is suffix voor decimale getallen)



Boolean Type



Data Type Boolean

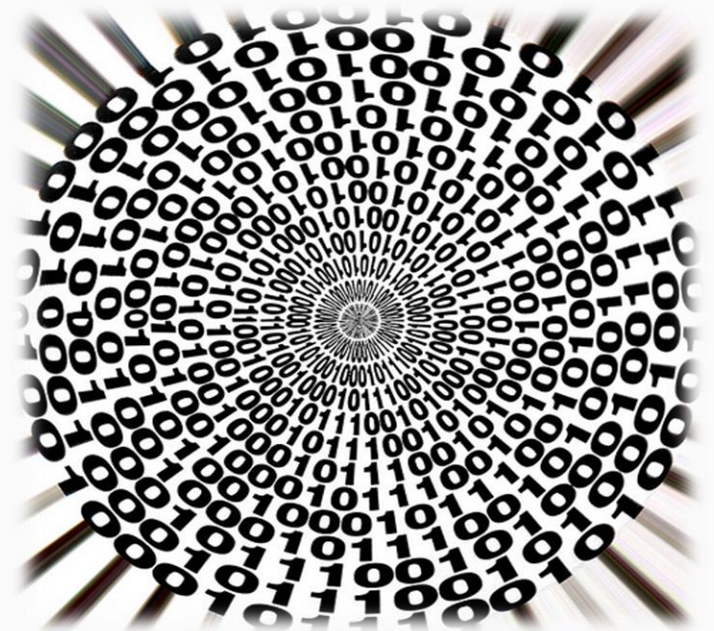
Boolean data type:

Wordt gedeclareerd via `bool` sleutelwoord

Heeft 2 mogelijke waarden: `true` en `false`

Is nuttig in logische uitdrukkingen (expressions)

De standaard waarde is `false`



Boolean Waarden – Voorbeeld

Voorbeeld van boolean variabelen die de waarden **true** of **false** aannemen:

```
int a = 1;  
int b = 2;  
bool greaterAB = (a > b);  
Console.WriteLine(greaterAB);    // False  
bool equalA1 = (a == 1);  
Console.WriteLine(equalA1);      // True
```



Character Type



Data Type Character

character data type:

Voorstelling van symbolische informatie

Declaratie met `char` sleutelwoord

Geeft elk symbool een overeenkomstig geheel getal

Standaard waarde = `'\0'`

Neemt 16 bits geheugen in (van `U+0000` tot `U+FFFF`)



A grid of 20 characters arranged in 4 rows and 5 columns, illustrating various Unicode scripts. The characters are: Row 1: A, a, B, b, C, c, D, d, E; Row 2: N, O, Y, Δ, Ξ, Ж, й, υ, ψ; Row 3: س, ع, ك, あ, に, サ; Row 4: 女, 花, 雪, 雨, 라, 바.

Characters en Codes

Elk symbool heeft zijn unieke Unicode code

Voorbeeld:

```
char symbol = 'a';  
Console.WriteLine("The code of '{0}' is: {1}",  
symbol, (int) symbol);  
symbol = 'b';  
Console.WriteLine("The code of '{0}' is: {1}",  
symbol, (int) symbol);  
symbol = 'A';  
Console.WriteLine("The code of '{0}' is: {1}",  
symbol, (int) symbol);
```




Character Data Type

Demo



Object Type

Data Type Object

object type:

Declaratie via **object** sleutelwoord

Is het basistype van de andere types

Kan waarden van alle verschillende types bevatten



Gebruik maken van datatype object

Een variabele van type object kan verschillende soorten datatypes bevatten:

Voorbeeld:

```
object dataContainer = 5;  
Console.Write("De waarde is: ");  
Console.WriteLine(dataContainer);  
dataContainer = "Vijf";  
Console.Write("De waarde is: ");  
Console.WriteLine(dataContainer);
```

leren. durven. doen.



Vragen?

DATA TYPES

REFERENTIES

[HTTPS://WWW.LEARNCS.ORG/](https://www.learncs.org/)

FUNDAMENTALS OF COMPUTER PROGRAMMING WITH C#

© SVETLIN NAKOV & CO.,