C# Essentials
Variabelen & operatoren

Koen Bloemen
Wim Bervoets
Kristof Palmaers

DE HOGESCHOOL MET HET NETWERK

Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be

- Variabelen vs constanten
- Datatypes
- Naming conventions
- Operatoren
- Numerieke conversies
- String conversies (TryParse)

Variabelen vs constanten

- Variabele
  - Tijdelijke opslag voor gegevens
  - **Verwijst naar een geheugenlocatie**
  - Heeft een bepaald gegevenstype (datatype)
  - Geen waarde = null reference voor bepaalde datatypes
- Constante
  - Beschrijvende naam voor “variabele”
  - Read-only
  - Waarde verandert niet
- Beide hebben een bepaald bereik (scope)
  - Afhankelijk waar ze gedeclareerd zijn
  - Buiten scope = niet (meer) toegankelijk

Datatypes

- int
  geheel getal
- float
  7 digits (32 bit)
- double
  15-16 digits (64 bit)
- decimal
  28-29 significant digits (128 bit)
- string
  tekst
- bool
  true or false

Variabelen

- Declaratie

```
static void Main(string[] args)
  {
    int eenGeheelGetal;
    bool a;
  }
```

- Initialisatie

```
static void Main(string[] args)
  {
     int eenGeheelGetalMetWaardeNul = 0;
     bool a = true;
     float eenKommaGetal = 2.5F;
     double eenAnderKommaGetal = 0.5;
     string eenStukTekst = "tekst";
  }
```

Variabelen

- Welke regel bevat een probleem

```
static void Main(string[] args)
  {
    // 1
    int leeftijd = 21;
    // 2
    bool isMan = true;
    // 3
    float totaleUitgaveKlant = 2015.61;
    // 4
    string naam = Johan Jansen;
  }
```

Variabelen

- Een variabele heeft een bepaalde scope of bereik
- Bepaald of variabele bereikbaar is

```
static string bereikbaarInClass;
static void Main(string[] args)
{
  string bereikbaarInMain;
  bereikbaarInMain = "PXL";
  bereikbaarInClass = "Main";
  Console.WriteLine(bereikbaarInMethod);
}
private static void TweedeMethode()
{
  string bereikbaarInTweedeMethode;
  bereikbaarInTweedeMethode = "PXL";
  bereikbaarInClass = "TweedeMethode";
  Console.WriteLine(bereikbaarInTweedeMethode);
}
```

Numerieke variabelen

- Geheel getal => kleinst mogelijk (uint, …)
- Decimaal getal => automatisch *double*
- Ander datatype => suffix gebruiken

Type	Suffix	Example
uint	U or u	100U
long	L or l	100L
ulong	UL or ul	100UL
float	F or f	123.45F
decimal	M or m	123.45M

Alfanumerieke variabelen

- Gewone tekenreeks
  - 0 of meer tekens tussen “”
  - Escape sequences (bijv. **\t**)
- Verbatim tekenreeks
  - @ prefix
  - Geen escape sequences
  - Alles letterlijk

```
string a = "hallo, C #";                          hallo, C #                  --- gewone tekenreeks
string b = @"hallo, C #";                         hallo, C #                  --- verbatim tekenreeks

string c = "hallo \t C #";                        hallo     C #               --- gewone tekenreeks
string d = @"hallo \t C #";                       hallo \t C #                --- verbatim tekenreeks

string e = "Paul is \"lector\" in klas B";        Paul is "lector" in klas B   --- gewone tekenreeks
string f = @"Paul is ""lector"" in klas B";       Paul is "lector" in klas B  --- verbatim tekenreeks

string g = "\\\\server\\share\\file.txt";         \\server\share\file.Txt      --- gewone tekenreeks
string h = @"\\server\share\file.txt";            \\server\share\file.Txt     --- verbatim tekenreeks

string i = "Visual\r\nStudio\r\n2017";            Visual                      --- gewone tekenreeks
string j = @"Visual                               Studio                      --- verbatim tekenreeks
Studio                                            2017                           over meerdere regels
2017";
```

Naming conventions

- camelCasing
  - Begint met een kleine letter
  - Elk woord in de samenstelling begint met een hoofdletter
  - Gebruikt voor variabelen en functieparameters
  - bijv.: naamCursist, rekeningBank, studentNaamPerAfdeling
- PascalCasing
  - Ook wel UppercamelCasing genoemd
  - Gebruikt voor functienamen, class definities, e.d.

Operatoren

- Rekenkundige (aritmetische)
  Berekeningen met getallen
- String operatoren
  Bewerkingen met tekst
- Toewijzingsoperatoren
  Nauw samenhangend met rekenkundige
- Relationele
  Vergelijkingen tussen twee variabelen
- Logische
  Berekeningen met booleans

Rekenkundige operatoren

- Binaire operatoren (werken met 2 operators)

Operator	Beschrijving
+x, -x	Teken veranderen
x+y, x-y	Som en verschil
x*y, x/y	Product en quotiënt
x%y	Rest na deling

- Unaire operatoren (werken met 1 operator)

Operator	Beschrijving
x++	Verhoog met 1
x--	Verminder met 1

String operatoren

Operator	Beschrijving
"tekst"+"tekst"	Teksten worden aan elkaar geplakt (ook wel concatenatie genoemd)

Toewijzingsoperatoren

Operator	Alternatief	Beschrijving
x=10	x=10	Waarde toekennen
x+=y	x=x+y	Verhogen met waarde
x-=y	x=x-y	Verminderen met waarde
x*=y	x=x*y	Vermenigvuldigen met waarde
x/=y	x=x/y	Delen door waarde
x%=y	x=x%y	Modulus toekennen

Relationele operatoren

Operator	Beschrijving
x==y	is gelijk aan
x!=y	is verschillend van
x<y	is kleiner dan
x>y	is groter dan
x<=y	is kleiner dan of gelijk aan
x>=y	is groter dan of gelijk aan

Logische operatoren

- ! (not operator)
  - keert de booleaanse expressie om
  - voorbeeld: `bool expressie = !(getal > 0); // wordt getal <= 0`

Voorwaarde	!
true	false
false	true

Logische operatoren

- &&, & (and operator)
  - Aan alle voorwaarden moet voldaan zijn om true te zijn
  - && (meeste gebruikt): stopt met voorwaardes checken als er eentje false is
  - & controleert alle voorwaardes, zelfs al is er eentje false
  - voorbeeld: `bool expressie = (getal > 0) && (getal < 10) && (getal != 5);`

Voorwaarde 1	Voorwaarde 1	&& of &
true	true	true
false	true	false
true	false	false
false	false	false

Logische operatoren

- ||, | (or operator)
  - Aan minstens 1 voorwaarden moet voldaan zijn om true te zijn
  - || (meeste gebruikt): stopt met voorwaardes checken als er eentje true is
  - | controleert alle voorwaardes, zelfs al is er eentje true
  - voorbeeld: `bool expressie = (getal > 0) || (getal < 10) || (getal != 5);`

Voorwaarde 1	Voorwaarde 1	|| of |
true	true	true
false	true	true
true	false	true
false	false	false

Logische operatoren

- ^ (xor operator)
  - Aan minstens 1 voorwaarden moet voldaan zijn om true te zijn
  - Is hetzelfde als or (|| of |), MAAR als alle voorwaarden true zijn => toch false als resultaat. Daarom spreken we exclusive or (xor)
  - voorbeeld: `bool expressie = (getal == 5) ^ (getal == 10);`

Voorwaarde 1	Voorwaarde 1	^
true	true	false
false	true	true
true	false	true
false	false	false

Evaluatie prioriteit

Rekenkundige operatoren	- (negatie)
	*, /, +, -, %
	++, -- (voorvoegsel)
Vergelijkingsoperatoren	<, >, <=, >=, ==, !=
Logische operatoren	!
	&, &&
	|, ||
	^
Toewijzingsoperatoren	=, *=, /=, +=, -=, %=
	++, -- (achtervoegsel)

Numerieke conversies

- Omzetten van datatypes

Verbredend	Versmallend
Gegevensverlies mogelijk	Kunnen (compile)errors geven
Worden automatisch uitgevoerd	Casting om gegevensverlies op te vangen
Ook wel impliciete conversies genoemd	Ook wel expliciete conversies genoemd

```
short shortResult, shortVal = 4;
int integerVal = 67;
long longResult;
float floatVal = 10.5F;
double doubleResult, doubleVal = 99.999;
string stringResult, stringVal = "17";
bool boolVal = true;
doubleResult = floatVal * shortVal; // impliciet: 10.5 * 4 = 42 (berekening in double)
shortResult = (short)floatVal; // expliciete casting: 10
longResult = integerVal + Convert.ToInt64(stringVal); // mix: 67 + 17 = 84
stringResult = Convert.ToString(boolVal) + Convert.ToString(doubleVal); // string: True99.999
```

Numerieke conversies

- System.Convert klasse
  - Bevat functies om te converteren
  - Enkel voor numerieke conversies
- Gehele getallen
  - short = Convert.ToInt16(val)
  - int = Convert.ToInt32(val)
  - long = Convert.ToInt64(val)
- Kommagetallen
  - float = Convert.ToSingle(val)
  - double = Convert.ToDouble(val)
  - decimal = Convert.ToDecimal(val)

Formattering van getallen

- string.format();

```
int getal1 = 123456;
float getal2 = 123.456F;
Console.WriteLine(string.Format("Geheel getal: {0} en floating point: {1}",getal1, getal2));
```

- string interpolation

```
int getal1 = 123456;
float getal2 = 123.456F;
Console.WriteLine($"Geheel getal: {getal1} en floating point: {getal2}");
```

String conversies

- Parsing wordt gebruikt om een string om te zetten naar een numerieke waarde

```
string tekst = "120";
int i = int.Parse(tekst); // geeft 120
float j = float.Parse(tekst) / 13; // geeft 9,230769
```

Conversie functie	Parse method
Convert.ToDecimal()	decimal.Parse()
Convert.ToDouble()	double.Parse()
Convert.ToInt32()	int.Parse()
Convert.ToInt64()	long.Parse()
Convert.ToSingle()	float.Parse()

String conversies

- TryParse
  - geeft true als conversie slaagt
  - testen tijdens conversie
  - geen runtime errors

```
string tekst = "5";
int cijfer;
bool resultaat = int.TryParse(tekst, out cijfer); // resultaat = true
//of
bool resultaat = int.TryParse(tekst, out int cijfer); // resultaat = true
```

```
string tekst = "PXL";
int cijfer;
bool resultaat = int.TryParse(tekst, out cijfer); // resultaat = false
//of
bool resultaat = int.TryParse(tekst, out int cijfer); // resultaat = false
```

String conversies

- Numerieke waarde omzetten naar string
  - Convert.ToString(variabeleNaam);
  - variabeleNaam.ToString();

```
int getal = 10;
string s1 = Convert.ToString(getal);
//of
string s1 = getal.ToString();
//ToString() heeft betere foutmeldingen en is performanter!
```

WPF controls

- Window

```
<Window x:Class="_51WpfTryCatchDatum.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:_51WpfTryCatchDatum"
        mc:Ignorable="d"
        Title="Correcte datum controleren...." Height="333.515" Width="536.376" FontSize="18" Loaded="Window_Loaded">
    <Window.Background>
        <ImageBrush ImageSource="Afbeeldingdatum.jpg"/>
    </Window.Background>
```

Eigenschappen	Name	Naam van formulier
	Title	Tekst in titelbalk
	Width	Breedte (in pixels)
	Height	Hoogte (in pixels)
	Background	Achtergrondkleur
	FontFamily	Lettertype voor alle objecten op het formulier
	WindowState	Venstergrootte (Minimized, Maximized, Normal)
Methods	Close()	Formulier wordt gesloten
Events	Loaded()	Wordt getriggerd wanneer formulier geladen wordt

WPF controls

- Label

```
<Label Content="Personeelslid" HorizontalAlignment="Left"
VerticalAlignment="Top" Margin="60,51,0,0" HorizontalContentAlignment="Right"/>
```

Eigenschappen	Name	Naam van label
	Content	Tekst in label
	Background	Achtergrondkleur
	Foreground	Tekstkleur
	FontFamily	Lettertype
	HorizontalAlignment	Plaats van tekst in label
	Visibility	Zichtbaar of niet

WPF controls

- Textbox

```
<TextBox x:Name="TxtResultaat" HorizontalAlignment="Left" Height="200"
Margin="65,190,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="395"
IsEnabled="False" Background="#FFEAE5E5" FontFamily="Consolas"/>
```

Eigenschappen	Name	Naam van tekstvak
	Tekst	Inhoud van tekstvak
	FontFamily/FontWeight	Lettertype, grootte, …
	Background	Achtergrondkleur
	Foreground	Tekstkleur
	IsEnabled	Beschikbaarheid (kan geklikt worden of niet)
	IsReadOnly	Inhoud kan gewijzigd worden (is altijd aanklikbaar)
	MaxLength	Maximaal aantal karakters
	MaxLine	Maximaal aantal lijnen
	VerticalScrollBarVisibility	Vertikale schuifbalk
	HorizontalScrollBarVisibility	Horizontale schuifbalk
	TextWrapping	Tekstterugloop
	Visibility	Zichtbaarheid

WPF controls

- Textbox

```
<TextBox x:Name="TxtResultaat" HorizontalAlignment="Left" Height="200"
Margin="65,190,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="395"
IsEnabled="False" Background="#FFEAE5E5" FontFamily="Consolas"/>
```

Methods	Focus()	Tekstvak krijgt de focus
	Clear()	Inhoud van tekstvak wordt gewist
Events	Click()	Getriggerd wanneer op tekstvak wordt geklikt
	TextChanged()	Getriggerd wanneer inhoud wordt aangepast
	GotFocus()	Tekstvak krijgt de focus
	LostFocus()	Tekstvak verliest de focus

WPF controls

- Button

```
<Button Name=“btnBerekenen" Content="_Berekenen" HorizontalAlignment="Left"
Margin="355,35,0,0" VerticalAlignment="Top" Width="105" Height="35" IsDefault="True"
Click=“BtnBerekenen_Click"/>
```

Eigenschappen	Name	Naam van knop
	Tekst	Tekst op knop
	Width	Breedte in pixels
	Height	Hoogte in pixels
	FontFamily	Lettertype, …
	FontStyle	Italic, …
	FontWeight	Bold, …
	Background	Achtergrondkleur
	Foreground	Tekstkleur
	IsDefault	Standaardknop (reageert op enter)
	IsEnabled	Beschikbaarheid
	Visibility	Zichtbaarheid

WPF controls

- Button

```
<Button x:Name="ButtonBerekenen" Content="_Berekenen" HorizontalAlignment="Left"
Margin="355,35,0,0" VerticalAlignment="Top" Width="105" Height="35" IsDefault="True"
Click="ButtonBerekenen_Click"/>
```

Methods	Focus()	Tekstvak krijgt de focus
Events	Click()	Getriggerd wanneer op knop geklikt wordt
	GotFocus()	Knop krijgt de focus
	LostFocus()	Knop verliest de focus
	MouseEnter()	Muis komt over de knop (zonder te klikken)

WPF controls

- TextBlock
  - zoals label gebruik om tekst te tonen
  - verschil met label
    - TextBlock kan geen rand hebben
    - TextBlock kan niet disabled worden
    - TextBlock kan geen afbeelding tonen
    - TextBlock kan geen sneltoets bevatten
    - TextBlock heeft geen ContentTemplate
  - voorkeur om enkel tekst te tonen
  - veel lichter object dan label

Oefeningen 3-7