

Van C++ naar C#

**door:
Tim Langens**

1 Inhoudsopgave

1	Inhoudsopgave	2
2	Inleiding	Fout! Bladwijzer niet gedefinieerd.
3	1 + 1 = 2, console	4
3.1	Het programma	4
3.2	Namespaces	4
3.3	Klassen.....	4
3.4	De Main Methode	5
3.5	Programma Statements	5
4	1 + 1 = 2, windows	6
4.1	Het formulier.....	6
4.1.1	Namespaces	6
4.1.2	Klasse Declaratie	6
4.1.3	Constructor.....	6
4.1.4	Main Executie.....	7
4.2	Titel toevoegen	7
4.3	Tekst toevoegen met label	7
4.3.1	Het label definiëren.....	7
4.3.2	De constructor aanpassen.....	7
4.3.3	Extra namespace	8
4.3.4	Het volledige programma	8
4.4	Tekst rechtstreeks toevoegen	9
4.4.1	Namespaces	9
4.4.2	De code	9
4.5	Een eigen tekstmethode	10
5	Interactieve methodes.....	11
5.1	Een knop definiëren	11
5.2	De constructor aanpassen.....	11
5.3	Instructie toevoegen.....	12
5.4	De volledige code	12
6	Initialisatie en dispose instructies.....	14
6.1	Initialisatie instructie.....	14
6.2	Dispose instructie.....	14
6.3	Alles op een rijtje.....	15
7	Input.....	17
7.1	Tekstvak.....	17
7.2	Input lezen en gebruiken.....	17
7.3	De volledige code	18

8	Synthese	20
8.1	Verschillende bewerkingen.....	20
8.2	1 tekstvak	20
8.3	Deling controleren	20
8.4	Kommagetallen.....	20
8.5	Mijn code	20
9	Menu toevoegen.....	21
9.1	Het hoofdmenu	21
9.2	Exit.....	21
9.3	Meer menu's	21
9.4	About	22
9.4.1	Linklabel	22
9.5	Contextmenu.....	22
10	Appendix.....	24
A	Bronnen en nuttige informatie	24
A.1	Websites.....	24
A.2	Cursussen	24
B	Code's	25
B.1	Het rekenmachine 1	25

2 1 + 1 = 2, console

2.1 Het programma

Aangezien je met C++ voornamelijk consolegericht geprogrammeerd zult hebben. Maken we nu een programma, dat in de console weergeeft dat $1+1 = 2$.

```
// Namespace Declaratie
using System;

// Start klasse van het programma
class Reken
{
    // Programma executie begint met Main()
    public static void Main()
    {
        // Naar de console schrijven
        Console.WriteLine( "1 + 1 = 2" );
        Console.ReadLine();
    }
}
```

Kopieer deze code en compileer het programma en voer uit.

2.2 Namespaces

De eerste twee lijnen code handelen met namespace. De eerste lijn is commentaar

```
// Namespace Declaratie
using System;
```

Een Namespace, ken je ook vanuit C++, het is eigenlijk een verwijzing dat je bepaalde code gebruikt, in dit programma is dat dus code vanuit System.

In C++ was dit vooral

```
using namespace std;
```

Wanneer we deze namespace System niet gebruiken zouden we moeten typen:

```
System.Console.WriteLine( "1 + 1 = 2" );
```

2.3 Klassen

De volgende twee lijnen code zorgen voor de klasdeclaratie. Ook hier is de eerste lijn commentaar. De tweede lijn declareert de klasse Reken.

```
// Start klasse van het programma
class Reken
```

2.4 De `Main` Methode

De volgende twee lijnen code geven de enige methode weer in het programma. Eerste lijn is weer commentaar. De tweede lijn definieert de main methode, de methode waar elk programma mee start, net zoals in C++.

```
// Programma executie begint met Main()  
public static void Main()
```

2.5 Programma Statements

De laatste drie lijnen of code, zorgen ervoor dat $1 + 1 = 2$ in de console wordt weergegeven. De eerste lijn is weer commentaar. De tweede lijn, drukt de tekst $1 + 1 = 2$ af. De derde lijn zorgt ervoor dat er gewacht wordt tot een toets wordt ingedrukt.

```
// Naar de console schrijven  
Console.WriteLine( "1 + 1 = 2" );  
Console.ReadLine();
```

3 1 + 1 = 2, windows

3.1 Het formulier

Wanneer we een windows programma schrijven, moeten we eerst een formulier definiëren. Dit doen we als volgt:

```
// Namespace Declaration
using System;
using System.Windows.Forms;

// Class Declaration
class MyForm : Form
{
    // Constructor
    public MyForm()
    {
    }

    // Main begins program execution
    public static void Main()
    {
        Application.Run( new MyForm() );
    }
}
```

Wanneer je deze code uitvoert zal je enkel een leeg venster krijgen.

3.1.1 Namespaces

We gebruiken nu niet enkel System maar ook System.Windows.Forms. Dit hebben we nodig om een formulier te kunnen definiëren.

3.1.2 Klasse Declaratie

```
class MyForm : Form
```

We leiden ons formulier af van de System klasse Form. Dit doen we door een : te gebruiken.

3.1.3 Constructor

Een constructor is een methode die een klasse definieert. Hierin staan alle elementen die de klasse heeft. Op dit ogenblik dus nog geen. De constructor methode heeft dezelfde naam als de klasse.

3.1.4 Main Executie

Deze code start de toepassing door de static `Run()` methode van de `Application` class in `System.Windows.Forms` namespace op te roepen, als parameter wordt het startformulier van de applicatie meegegeven.

```
public static void Main()
{
    Application.Run( new MyForm() );
}
```

3.2 Titel toevoegen

Nu gaan we ervoor zorgen dat het programma, de titel Rekenmachine krijgt. Dit doen we door in de constructor een extra regel toe te voegen

```
public MyForm()
{
    // Text to be Displayed in the Caption-Title Bar
    this.Text = "Rekenmachine";
}
```

Probeer dit, en bekijk de veranderingen.

3.3 Tekst toevoegen met label

Na de titel te hebben toegevoegt gaan we de tekst, $1 + 1 = 2$, toevoegen. Dit doen we d.m.v. een label. Om een label toe te voegen moeten we twee dingen doen:

1. Een label definiëren.
2. De constructor aanpassen om het label weer te geven.

3.3.1 Het label definiëren

```
// Create an instance of a Label.
private Label label1;
```

Door deze code regel toe te voegen aan onze klasse, definiëren we een label genaamd `label1`.

3.3.2 De constructor aanpassen

De constructor code bestaat uit 3 delen.

Het eerste deel past het lettertype aan.

```
// Changes the font name and size
this.Font = new Font( "MS Sans Serif", 20 );
```

In deze code, veranderen we het lettertype in MS Sans Serif met grootte 20.

Het tweede deel beschrijft het label.

```
// Label initialization and setup
    label1 = new Label();
    label1.Location = new Point( 24, 16 );
    label1.Size = new Size( 150, 24 );
    label1.Text = "1 + 1 = 2";
    label1.TextAlign = ContentAlignment.MiddleCenter;
```

De regels willen in volgorde zeggen:

1. Maak een nieuw object van het type label aan.
2. Definiëert de plaats waar het object moet komen.
3. Bepaalt de grootte van het label.
4. De tekst die het label bevat.
5. Zegt dat de uitlijning van de tekst horizontaal en verticaal gecentreerd moet zijn.

Het laatste deel code plaatst het label op het formulier

```
// Add the label to the Form
    this.Controls.Add( label1 );
```

3.3.3 Extra namespace

Wanneer we iets willen tekenen op een formulier hebben we de extra namespace vermelding, `using System.Drawing`, nodig.

3.3.4 Het volledige programma

De volledige programmacode zou er dan ongeveer zo uit moeten zien:

```
// Namespace Declaration
using System;
using System.Drawing;
using System.Windows.Forms;

// Class Declaration
class MyForm : Form
{
    // Create an instance of a Label.
    private Label label1;

    // Constructor
    public MyForm()
    {
        // Text to be Displayed in the Caption-Title Bar
        this.Text = "Rekenmachine";

        // Changes the font name and size
        this.Font = new Font( "MS Sans Serif", 20 );

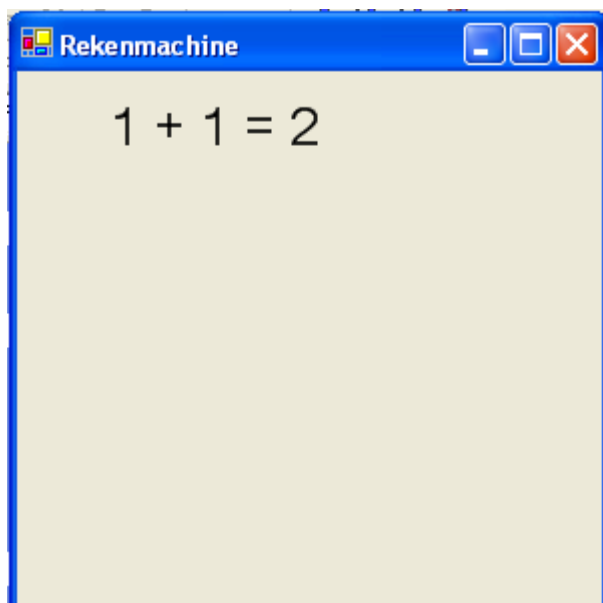
        // Label initialization and setup
```

```
label1 = new Label();
label1.Location = new Point( 24, 16 );
label1.Size = new Size( 150, 24 );
label1.Text = "1 + 1 = 2";
label1.TextAlign = ContentAlignment.MiddleCenter;

// Add the label to the Form
this.Controls.Add( label1 );
}

// Main begins program execution
public static void Main()
{
    Application.Run( new MyForm() );
}
}
```

Wanneer we dit uitvoeren krijgen we het volgende te zien:



3.4 Tekst rechtstreeks toevoegen

Zojuist heb je geleerd hoe je tekst d.m.v. een label kunt toevoegen. Maar er is ook een manier om tekst te plaatsen zonder label.

3.4.1 Namespaces

Zoals al eerder gezien heb je om een object op een formulier te tekenen de namespace, `System.Drawing`, nodig.

3.4.2 De code

```
public MyForm()
{
```

```
        this.Text = "Rekenmachine";
    }

    protected override void OnPaint( PaintEventArgs e )
    {
        // Call base class OnPaint method
        base.OnPaint(e);

        // Method under System.Drawing.Graphics
        e.Graphics.DrawString( "1 + 1 = 2", new Font( "Verdana", 20 ),
                               new SolidBrush( Color.Tomato ),
                               40,
                               40 );
    }
}
```

Het enige wat we nog nodig hebben om rechtstreeks tekst weer te geven is de bovenstaande code. Deze code is de nieuwe constructor code.

Om op het formulier te kunnen schrijven heeft elk formulier een OnPaint methode. In deze overschreven functie, schrijven we de tekst die we willen, $1 + 1 = 2$. Hiervoor gebruiken we de DrawString methode, als argumenten geven we mee, de te tekenen string, het lettertype, de kleur, en de linksbovenhoekcoördinaten.

3.5 Een eigen tekstmethode

Daarjuist hebben we tekst toegevoegd, door een label of door de OnPaint methode te overschrijven, nu gaan we ons eigen paint methode schrijven.

```
public MyForm()
{
    this.Text = "Rekenmachine";
    this.Paint += new PaintEventHandler( f1_paint );
}

private static void f1_paint( object sender, PaintEventArgs e )
{
    // Get Graphics Object
    Graphics g = e.Graphics;
    // Method under System.Drawing.Graphics
    g.DrawString( "1 + 1 = 2", new Font( "Verdana", 20 ),
                 new SolidBrush( Color.Tomato ),
                 40,
                 40 );
}
```

De regel:

```
this.Paint += new PaintEventHandler( f1_paint );
```

zorgt ervoor dat we een nieuwe paintmethode toewijzen aan ons formulier i.p.v. het OnPaint commando te gebruiken. Uiteraard moeten we deze methode ook beschrijven.

4 Interactieve methodes

Normaal gezien moet het programma wachten op de gebruiker voor het iets doet. Bij ons voorbeeldprogramma zal dit gebeuren door op een knop te drukken. Een knop toevoegen doen we in enkele stappen:

1. De knop definiëren
2. De constructor aanpassen om de knop weer te geven
3. Een instructie aan de knop toevoegen.

Om hieraan te beginnen verwijder je best alles uit je programmacode zodat je vanaf volgende code kan beginnen

```
// Namespace Declaration
using System;
using System.Drawing;
using System.Windows.Forms;

public class MyForm : Form
{
    // Constructor
    public MyForm()
    {
        this.Text = "Rekenmachine";
    }
    // Main begins program execution
    public static void Main()
    {
        Application.Run( new MyForm() );
    }
}
```

4.1 Een knop definiëren

Als eerste stap moeten we ons programma laten weten dat het er een knop aanwezig is. Dit doen we op dezelfde manier als dat we een label toevoegen:

```
// Create an instance of a Button.
private Button plus;
```

4.2 De constructor aanpassen

Vervolgens moeten we de constructor aanpassen zodat de gebruiker de knop ook kan zien:

```
// Button initialization and setup
plus = new Button();
plus.Text = "+";
plus.Location = new Point( 25, 15 );
plus.Size = new Size( 20, 20 );

// Add the Button to the Form
this.Controls.Add( plus );
```

Ook hier spreekt de code weer voor zich.

4.3 Instructie toevoegen

Als laatste moeten we een instructie toevoegen aan de knop, zodat wanneer je op de knop klikt, er een boodschap verschijnt.

Dit doen we door in de constructor te vermelden welke methode gebruikt moet worden wanneer er geklikt wordt:

```
// Add Event for Button Click
plus.Click += new EventHandler(Plus_Clicked);
```

En de methode Plus_Clicked toe te voegen aan de klasse MyForm:

```
// Event for Button Click
public void Plus_Clicked( object ob, EventArgs e )
{
    MessageBox.Show( "1 + 1 = 2", "MessageBox" );
}
```

Deze methode zal dus een messagebox geven met daarin de melding $1 + 1 = 2$ wanneer er op de knop + geklikt wordt.

4.4 De volledige code

```
// Namespace Declaration
using System;
using System.Drawing;
using System.Windows.Forms;

public class MyForm : Form
{
    // Create an instance of a Button.
    private Button plus;

    // Constructor
    public MyForm()
    {
        this.Text = "Rekenmachine";

        // Button initialization and setup
        plus = new Button();
        plus.Text = "+";
        plus.Location = new Point( 25, 15 );
        plus.Size = new Size( 20, 20 );

        // Add the Button to the Form
        this.Controls.Add( plus );

        // Add Event for Button Click
        plus.Click += new EventHandler(Plus_Clicked);
    }

    // Event for Button Click
```

```
public void Plus_Clicked( object ob, EventArgs e )
{
    MessageBox.Show( "1 + 1 = 2", "MessageBox" );
}

// Main begins program execution
public static void Main()
{
    Application.Run( new MyForm() );
}
}
```

5 Initialisatie en dispose instructies

Deze instructies lijken niets speciaals bij te brengen aan het programma, het programma zal nog net hetzelfde reageren

5.1 Initialisatie instructie

Een initialisatie instructie is eigenlijk hetzelfde als de constructor, maar je verwijst in je constructor naar je initialisatie instructie en in je initialisatie instructie zet je wat je voordien in je constructor zette. Moeilijke uitleg? Kijk gewoon naar de code:

```
// Initialize form components
private void InitializeComponent()
{
    this.Text = "Event Driven Form 2";

    // Button initialization and setup
    btn1 = new Button();
    btn1.Text = "Click Me";
    btn1.Location = new Point( 25, 15 );
    btn1.Size = new Size( 70, 20 );

    // Add the Button to the Form
    this.Controls.Add( btn1 );

    // Add Event for Button Click
    btn1.Click += new EventHandler(Btn1_Clicked);
}
```

Dit is de initialisatie instructie, nu nog de nieuwe constructor:

```
// Constructor
public EventFrm2()
{
    InitializeComponent();
}
```

5.2 Dispose instructie

De dispose instructie zorgt voor een zuinig gebruik van het geheugen, Alle gebruikte bronnen worden terug afgesloten. Voor ons programma is het geen ramp indien dit niet netjes gebeurt, maar wanneer je programma, duizenden instructies gebruikt, kan dit uitdraaien in systeemfouten en het vastlopen van je computer. Iets wat je niet echt wilt meemaken.

```
// Clean up any resources being used
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if( components != null )
        {
```

```
        components.Dispose();
    }
}
base.Dispose( disposing );
}
```

Dus om een roodgloeiende helpdesklijn te mijden, kan je best leren om dit altijd in je programma te zetten.

5.3 Alles op een rijtje

Als we alles op een rijtje zetten krijgen we dus de volgende code:

```
using System;
using System.Drawing;
using System.Windows.Forms;

public class MyForm : Form
{
    private System.ComponentModel.Container components = null;

    // Create an instance of a Button
    private Button plus;

    // Constructor
    public MyForm()
    {
        InitializeComponent();
    }

    // Clean up any resources being used
    protected override void Dispose( bool disposing )
    {
        if( disposing )
        {
            if( components != null )
            {
                components.Dispose();
            }
        }
        base.Dispose( disposing );
    }

    // Initialize form components
    private void InitializeComponent()
    {
        this.Text = "Rekenmachine";

        // Button initialization and setup
        plus = new Button();
        plus.Text = "+";
        plus.Location = new Point( 25, 15 );
        plus.Size = new Size( 20, 20 );

        // Add the Button to the Form
        this.Controls.Add( plus );
    }
}
```

```
        // Add Event for Button Click
        plus.Click += new EventHandler(plus_Clicked);
    }

    // The main entry point for the application.
    [STAThread]
    public static void Main()
    {
        Application.Run( new MyForm() );
    }

    // Event for Button Click
    public void plus_Clicked( object ob, EventArgs e )
    {
        MessageBox.Show( "1 + 1 = 2", "Plus" );
    }
}
```

6 Input

Uiteraard willen we telkens een andere optelling doen en niet enkel laten zien dat $1 + 1 = 2$. Hiervoor moeten we dan wel getallen kunnen inlezen. Daarvoor hebben we tekstvakken nodig.

6.1 Tekstvak

De bedoeling is om 2 tekstvakken te creëren die de twee termen van de som voorstellen.

Een tekstvak wordt op dezelfde manier aangemaakt als een label en een knop. We moeten daarvoor dus volgende code toevoegen om twee tekstvakken te maken:

De tekstvakken definiëren:

```
private TextBox term1, term2;
```

De tekstvakken initialiseren in de constructor:

```
// term1 initialization and setup
term1 = new TextBox();
term1.Text = "0";
term1.Location = new Point( 25, 40);
term1.Size = new Size(70, 20);
// term2 initialization and setup
term2 = new TextBox();
term2.Text = "0";
term2.Location = new Point( 110, 40);
term2.Size = new Size(70,20);
```

De tekstvakken op het formulier weergeven.

```
this.Controls.Add( term1 );
this.Controls.Add( term2 );
```

6.2 Input lezen en gebruiken

De bedoeling is nu om de input te gaan lezen en op te tellen, wanneer er op de plusknop gedrukt wordt. Dit gaan we dus doen door de `Plus_Clicked` methode aan te passen. De uitkomst gaan we dan melden d.m.v. een messagebox.

Allereerst moet je weten dat een tekstvak een string bevat. We gaan die string dus nog moeten omzetten in een integer. Zodat we deze kunnen optellen. Dit doen we met de volgende instructie:

```
Int32.Parse(term1.Text);
```

Wanneer we dus `term1` en `term2` willen optellen bekomen we een integer. Deze integer zullen we dan terug moeten omzetten naar een string. Dit kunnen we d.m.v. Het `ToString` commando:

```
(Int32.Parse(term1.Text) + Int32.Parse(term2.Text)).ToString(s);
```

Wanneer we bovenstaand commando in een messagebox zetten krijgen we de uitkomst van de som in een messagebox.

Uiteraard moeten we ook een string `s` declareren waarnaar de string gekopieerd wordt.

6.3 De volledige code

```
using System;
using System.Drawing;
using System.Windows.Forms;

public class MyForm : Form
{
    private System.ComponentModel.Container components = null;

    private Button plus;
    private TextBox term1, term2;
    private String s;

    // Constructor
    public MyForm()
    {
        InitializeComponent();
    }

    // Clean up any resources being used
    protected override void Dispose( bool disposing )
    {
        if( disposing )
        {
            if( components != null )
            {
                components.Dispose();
            }
        }
        base.Dispose( disposing );
    }

    // Initialize form components
    private void InitializeComponent()
    {
        this.Text = "Rekenmachine";

        // Button initialization and setup
        plus = new Button();
        plus.Text = "+";
        plus.Location = new Point( 105, 15 );
        plus.Size = new Size( 20, 20 );

        // term1 initialization and setup
        term1 = new TextBox();
        term1.Text = "0";
        term1.Location = new Point( 25, 15);
        term1.Size = new Size(70, 20);

        // term2 initialization and setup
```

```
term2 = new TextBox();
term2.Text = "0";
term2.Location = new Point( 135, 15);
term2.Size = new Size(70,20);

// Add the objects to the Form
this.Controls.Add( plus );
this.Controls.Add( term1 );
this.Controls.Add( term2 );

// Add Event for Button Click
plus.Click += new EventHandler(plus_Clicked);
}

// The main entry point for the application.
[STAThread]
public static void Main()
{
    Application.Run( new MyForm() );
}

// Event for Button Click
public void plus_Clicked( object ob, EventArgs e )
{
    MessageBox.Show( (Int32.Parse(term1.Text) +
Int32.Parse(term2.Text)).ToString(s) );
}
}
```

7 Synthese

Via deze synthese krijg je een overzicht van wat je geleerd hebt. De bedoeling is om zelf je rekenmachine uit te bouwen met de dingen die je geleerd hebt. Op het einde zal je een uitgebreide code zien, maar dit wil niet zeggen dat dit de enige oplossing is. Het kan zijn dat jouw programma werkt maar niet hetzelfde is. Het handigste is om op het einde de code van hier over te nemen zodat je verder kunt met de hier aangereikte code, maar niets is verplicht natuurlijk.

7.1 Verschillende bewerkingen

Voeg ook nog andere bewerkingen toe:

- Vermenigvuldigen
- Aftrekken
- Delen

De uitkomsten laat je ook in een messagebox komen. De deling is een gehele deling.

7.2 1 tekstvak

Maak van je rekenmachine een echt rekenmachine, Dit betekent dat je nog maar 1 tekstvak hebt, een extra knop = hebt en je de berekeningen gewoon verder rekt op het vorige getal. De uitkomst geef je uiteraard weer in het tekstvak. Zorg ook voor een knop die je rekenmachine reset (C) en knoppen om de getallen mee in te geven.

7.3 Deling controleren

Bouw met je kennis van C++ de deling zo uit dat je controleert op een getal verschillend van nul.

7.4 Kommagetallen

Zorg ervoor dat je met kommagetallen kunt rekenen. (double)

7.5 Mijn code

De code die ik heb kan je bekijken in appendix B: code's, het rekenmachine 1.

8 Menu toevoegen

Nu gaan we aan ons rekenmachine een menu toevoegen.

8.1 Het hoofdmenu

Een menu toevoegen is niet zo moeilijk in C#. We moeten enkel volgende lijnen code toevoegen:

Bij het declaratiegedeelte:

```
Private MainMenu mainMenu
```

In public MyForm:

```
// Add File Menu  
MenuItem File = mainMenu.MenuItems.Add("&File");  
File.MenuItems.Add( new MenuItem("&Exit" ) );
```

in InitializeComponents:

```
mainMenu = new MainMenu();  
this.menu = mainMenu;
```

Zoals je ziet zal je nu een menu krijgen met 1 (nog niet werkende) mogelijkheid namelijk exit.

8.2 Exit

Laat ons nu eens zorgen dat die exit werkt. Eigenlijk heb je dit al eens gedaan, aangezien het bijna op dezelfde manier gedaan wordt als bij een knop.

Het grootste verschil is dat we onze eventhandler bij het initialiseren plaatsen.

```
File.MenuItems.Add( new MenuItem("&Exit", new  
EventHandler(this.FileExit_Clicked)));
```

De code voor FileExit_Clicked ziet er zo uit:

```
// File->Exit Menu item handler  
private void FileExit_Clicked(object sender, System.EventArgs e)  
{  
    this.Close();  
}
```

8.3 Meer menu's

Maak zelf een menu Edit bij dat de deelmenu's copy en paste heeft. De menu's moeten enkel maar een messagebox tonen.

8.4 About

We gaan uiteraard, zoals bij elk programma, een klein about formuliertje bij in het menu zetten. Deze optie is te vinden onder help en als men deze optie kiest dan krijgt men wat informatie over het programma

Het extra menu, help, en de aboutknop kan je zelf maken.

Wanneer je op de knop drukt komt er een nieuw formulier naar boven. Dat betekent dat we dus een nieuw formulier moeten definiëren. Met hierop wat tekst. Neem hiervoor de code van in hoofdstuk 4. Je kan de code uiteraard aanpassen. Een handigheidje hiervoor is een linklabel.

8.4.1 Linklabel

Een linklabel is, zoals de naam het zelf zegt, een label dat linkt naar een site. Je kan dit aanmaken zoals elk ander object:

```
Private LinkLabel link;
```

Uiteraard moet je dan ook nog het object initialiseren en instellen en het toevoegen aan het formulier.

Als je het hierbij laat dan zal het linklabel zich gedragen als een gewoon label met de opmaak van een hyperlink. Daarom moeten we nog een event aan het label hangen.

```
// link1 linken
protected void link1_Clicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    link1.LinkVisited = true;
    System.Diagnostics.Process.Start(
"http://users.telenet.be/langens/ikke" );
}
```

8.5 Contextmenu

Een contextmenu is een menu dat je verkrijgt wanneer je rechts klikt op een programma. Je kan het maken door volgende code toe te voegen aan de InitializeComponent() methode:

```
this.contextMenu1 = new ContextMenu();
this.menuItem1 = new MenuItem();

// contextMenu1
this.contextMenu1.MenuItems.AddRange(new MenuItem[] {this.menuItem1});

// menuItem1
this.menuItem1.Index = 0;
this.menuItem1.Text = "About";
this.menuItem1.Click += new EventHandler(this.about_Clicked);

// ContextMenuDemo
```

```
this.AutoScaleBaseSize = new Size(5, 13);  
this.ClientSize = new Size(292, 266);  
this.ContextMenu = this.contextMenu1;  
this.Name = "Form1";  
this.Text = "Form1";
```

9 Appendix

A Bronnen en nuttige informatie

A.1 Websites

<http://www.publicjoe.f9.co.uk/csharp/tut.html>

http://www.c-sharpcorner.com/2/convert_class.asp

A.2 Cursussen

MELL 2.0 Microsoft Visual Studio .NET Collection

B Code's

B.1 Het rekenmachine 1

```
using System;
using System.Drawing;
using System.Windows.Forms;

public class MyForm : Form
{
    private System.ComponentModel.Container components = null;

    private Button
plus,min,maal,deel,uitk,reset,een,twee,drie,vier,vijf,zes,zeven,acht,negen
,nul,komma,neg;
    private TextBox reken;
    private String s;
    private double term;
    private byte bewerk = 0; // de byte bewerk heeft een andere waarde
naar gelang de bewerking, 0 betekent niets doen
    private bool cijfer = true; // staat op false als er laatst op een
cijfer is geklikt.

    // Constructor
    public MyForm()
    {
        InitializeComponent();
    }

    // Clean up any resources being used
    protected override void Dispose( bool disposing )
    {
        if( disposing )
        {
            if( components != null )
            {
                components.Dispose();
            }
        }
        base.Dispose( disposing );
    }

    // Initialize form components
    private void InitializeComponent()
    {
        this.Text = "Rekenmachine";

        // Plus initialization and setup
        plus = new Button();
        plus.Text = "+";
        plus.Location = new Point(100, 40 );
        plus.Size = new Size( 20, 20 );
        plus.TabIndex = 1;

        // Min initialization and setup
```

```
min = new Button();
min.Text = "-";
min.Location = new Point( 100,70 );
min.Size = new Size( 20, 20 );
min.TabIndex = 2;

// Maal initialization and setup
maal = new Button();
maal.Text = "x";
maal.Location = new Point( 100,100);
maal.Size = new Size( 20, 20 );
maal.TabIndex = 3;

// Deel initialization and setup
deel = new Button();
deel.Text = "/";
deel.Location = new Point( 100 ,130);
deel.Size = new Size( 20, 20 );
deel.TabIndex = 4;

// uitk initialization and setup
uitk = new Button();
uitk.Text = "=";
uitk.Location = new Point( 130,100);
uitk.Size = new Size( 20, 50 );
uitk.TabIndex = 5;

// reset initialization and setup
reset = new Button();
reset.Text = "C";
reset.Location = new Point( 130, 40);
reset.Size = new Size( 20, 20 );
reset.TabIndex = 6;

// nul initialization and setup
nul = new Button();
nul.Text = "0";
nul.Location = new Point( 40,130);
nul.Size = new Size( 20, 20 );
nul.TabIndex = 7;

// een initialization and setup
een = new Button();
een.Text = "1";
een.Location = new Point( 10,100);
een.Size = new Size( 20, 20 );
een.TabIndex = 8;

// twee initialization and setup
twee = new Button();
twee.Text = "2";
twee.Location = new Point( 40 ,100);
twee.Size = new Size(20,20);
twee.TabIndex = 9;

// drie initialization and setup
drie = new Button();
drie.Text = "3";
drie.Location = new Point( 70 ,100);
```

```
drie.Size = new Size( 20, 20 );
drie.TabIndex = 10;

// vier initialization and setup
vier = new Button();
vier.Text = "4";
vier.Location = new Point( 10 , 70);
vier.Size = new Size( 20, 20 );
vier.TabIndex = 11;

// vijf initialization and setup
vijf = new Button();
vijf.Text = "5";
vijf.Location = new Point( 40, 70);
vijf.Size = new Size( 20, 20 );
vijf.TabIndex = 12;

// zes initialization and setup
zes = new Button();
zes.Text = "6";
zes.Location = new Point( 70 , 70);
zes.Size = new Size( 20, 20 );
zes.TabIndex = 13;

// zeven initialization and setup
zeven = new Button();
zeven.Text = "7";
zeven.Location = new Point( 10, 40);
zeven.Size = new Size( 20, 20 );
zeven.TabIndex = 14;

// acht initialization and setup
acht = new Button();
acht.Text = "8";
acht.Location = new Point( 40 , 40);
acht.Size = new Size( 20, 20 );
acht.TabIndex = 15;

// negen initialization and setup
negen = new Button();
negen.Text = "9";
negen.Location = new Point( 70, 40);
negen.Size = new Size( 20, 20 );
negen.TabIndex = 16;

// neg initialization and setup
neg = new Button();
neg.Text = "±";
neg.Location = new Point( 10, 130);
neg.Size = new Size( 20, 20 );
neg.TabIndex = 17;

// komma initialization and setup
komma = new Button();
komma.Text = ",";
komma.Location = new Point( 70, 130);
komma.Size = new Size( 20, 20 );
komma.TabIndex = 18;
```

```
// reken initialization and setup
reken = new TextBox();
reken.Text = "0";
reken.Location = new Point( 10, 15);
reken.Size = new Size(140, 20);
reken.TabIndex = 0;
reken.TextAlign = Sys-
tem.Windows.Forms.HorizontalAlignment.Right;

// Add the objects to the Form
this.Controls.Add( plus );
this.Controls.Add( min );
this.Controls.Add( maal );
this.Controls.Add( deel );
this.Controls.Add( uitk );
this.Controls.Add( reset );
this.Controls.Add( nul );
this.Controls.Add( een );
this.Controls.Add( twee );
this.Controls.Add( drie );
this.Controls.Add( vier );
this.Controls.Add( vijf );
this.Controls.Add( zes );
this.Controls.Add( zeven );
this.Controls.Add( acht );
this.Controls.Add( negen );
this.Controls.Add( komma );
this.Controls.Add( neg );
this.Controls.Add( reken );

// Add Event for Button Click
plus.Click += new EventHandler(plus_Clicked);
min.Click += new EventHandler(min_Clicked);
maal.Click += new EventHandler(maal_Clicked);
deel.Click += new EventHandler(deel_Clicked);
uitk.Click += new EventHandler(uitk_Clicked);
reset.Click += new EventHandler(reset_Clicked);
nul.Click += new EventHandler(nul_Clicked);
een.Click += new EventHandler(een_Clicked);
twee.Click += new EventHandler(twee_Clicked);
drie.Click += new EventHandler(drie_Clicked);
vier.Click += new EventHandler(vier_Clicked);
vijf.Click += new EventHandler(vijf_Clicked);
zes.Click += new EventHandler(zes_Clicked);
zeven.Click += new EventHandler(zeven_Clicked);
acht.Click += new EventHandler(acht_Clicked);
negen.Click += new EventHandler(negen_Clicked);
neg.Click += new EventHandler(neg_Clicked);
komma.Click += new EventHandler(komma_Clicked);
}

// Event for plus Click
public void plus_Clicked( object ob, EventArgs e )
{
    bereken();
    bewerk = 1; // 1 betekent dat er opgeteld moet worden
    reken.Text = term.ToString(s);
}
```

```
// Event for min Click
public void min_Clicked( object ob, EventArgs e )
{
    bereken();
    bewerk = 2; // 2 betekent dat er afgetrokken moet worden
    reken.Text = term.ToString(s);
}

// Event for maal Click
public void maal_Clicked( object ob, EventArgs e )
{
    bereken();
    bewerk = 3; // 3 betekent dat er vermenigvuldigt moet worden
    reken.Text = term.ToString(s);
}

// Event for deel Click
public void deel_Clicked( object ob, EventArgs e )
{
    bereken();
    bewerk = 4; // 4 betekent dat er gedeelt moet worden
    reken.Text = term.ToString(s);
}

// Event for uitk Click
public void uitk_Clicked( object ob, EventArgs e )
{
    bereken();
    bewerk = 0; // 4 betekent dat er gedeelt moet worden
    reken.Text = term.ToString(s);
}

// Event for reset Click
public void reset_Clicked( object ob, EventArgs e )
{
    bewerk = 0; // 4 betekent dat er gedeelt moet worden
    reken.Text = "0";
}

// Event for nul Click
public void nul_Clicked( object ob, EventArgs e )
{
    if(cijfer | reken.Text == "0")
    {
        reken.Text = "0";
        cijfer = false;
    }
    else
        reken.Text = reken.Text + "0";
}

// Event for een Click
public void een_Clicked( object ob, EventArgs e )
{
    if(cijfer | reken.Text == "0")
```

```
        {
            reken.Text = "1";
            cijfer = false;
        }
        else
            reken.Text = reken.Text + "1";
    }

    // Event for twee Click
    public void twee_Clicked( object ob, EventArgs e)
    {
        if(cijfer | reken.Text == "0")
        {
            reken.Text = "2";
            cijfer = false;
        }
        else
            reken.Text = reken.Text + "2";
    }

    // Event for drie Click
    public void drie_Clicked( object ob, EventArgs e)
    {
        if(cijfer | reken.Text == "0")
        {
            reken.Text = "3";
            cijfer = false;
        }
        else
            reken.Text = reken.Text + "3";
    }

    // Event for vier Click
    public void vier_Clicked( object ob, EventArgs e)
    {
        if(cijfer | reken.Text == "0")
        {
            reken.Text = "4";
            cijfer = false;
        }
        else
            reken.Text = reken.Text + "4";
    }

    // Event for vijf Click
    public void vijf_Clicked( object ob, EventArgs e)
    {
        if(cijfer | reken.Text == "0")
        {
            reken.Text = "5";
            cijfer = false;
        }
        else
            reken.Text = reken.Text + "5";
    }

    // Event for zes Click
    public void zes_Clicked( object ob, EventArgs e)
    {
```

```
        if(cijfer | reken.Text == "0")
        {
            reken.Text = "6";
            cijfer = false;
        }
        else
            reken.Text = reken.Text + "6";
    }

    // Event for zeven Click
    public void zeven_Clicked( object ob, EventArgs e)
    {
        if(cijfer | reken.Text == "0")
        {
            reken.Text = "7";
            cijfer = false;
        }
        else
            reken.Text = reken.Text + "7";
    }

    // Event for acht Click
    public void acht_Clicked( object ob, EventArgs e)
    {
        if(cijfer | reken.Text == "0")
        {
            reken.Text = "8";
            cijfer = false;
        }
        else
            reken.Text = reken.Text + "8";
    }

    // Event for negen Click
    public void negen_Clicked( object ob, EventArgs e)
    {
        if(cijfer | reken.Text == "0")
        {
            reken.Text = "9";
            cijfer = false;
        }
        else
            reken.Text = reken.Text + "9";
    }

    // Event for neg Click
    public void neg_Clicked( object ob, EventArgs e)
    {
        if(double.Parse(reken.Text) < 0)
            reken.Text = (- double.Parse(reken.Text)).ToString(s);
        else
            reken.Text = "-" + reken.Text;
    }

    // Event for komma Click
    public void komma_Clicked( object ob, EventArgs e)
    {
        reken.Text = reken.Text + ",";
    }
}
```

```
// Functie die de vorige berekening uitvoert.
public void bereken()
{
    switch(bewerk)
    {
        case 1:
            term += Double.Parse(reken.Text);
            reken.Focus();
            break;
        case 2:
            term -= Double.Parse(reken.Text);
            reken.Focus();
            break;
        case 3:
            term *= Double.Parse(reken.Text);
            reken.Focus();
            break;
        case 4:
            if (Double.Parse(reken.Text) != 0)
            {
                term /= Double.Parse(reken.Text);
                reken.Focus();
            }
            else
            {
                MessageBox.Show("Delen door nul gaat
niet!");
                reken.Focus();
            }
            break;
        default:
            term = Double.Parse(reken.Text);
            reken.Focus();
            break;
    }
    cijfer = true;
}

// The main entry point for the application.
[STAThread]
public static void Main()
{
    Application.Run( new MyForm() );
}
}
```