

# **Programming Basics**

Fontys ICT teachers

2023-10-02

# Table of contents

<b>Preface</b>	<b>9</b>
<b>1 Introduction</b>	<b>10</b>
1.1 Intro personas . . . . .	10
<b>I Tools</b>	<b>12</b>
<b>2 Visual Studio installatie</b>	<b>13</b>
2.1 Wat heb ik nodig? . . . . .	13
2.1.1 Default: VS for MS-Windows . . . . .	13
2.1.2 Alternatief: Visual Studio Code . . . . .	13
2.1.3 En op een Mac? . . . . .	13
2.2 Installatie . . . . .	14
2.2.1 Een Manier . . . . .	14
2.3 Je eerste programma (WinForm, werkt alleen in VS for MS-Windows) . . . . .	14
2.3.1 Windows Forms App C . . . . .	16
2.3.2 Tikken C# Sourcecode . . . . .	17
2.3.3 Opstarten eerste programma . . . . .	18
2.3.4 Het werkt niet? . . . . .	19
2.3.5 Hoe verder? . . . . .	19
2.4 Inleveren van Visual Studio solutions . . . . .	20
2.5 Short cut keys in Visual Studio . . . . .	20
<b>II Theory</b>	<b>23</b>
<b>3 De Debugger</b>	<b>24</b>
3.1 Inleiding . . . . .	24
3.2 Opdracht . . . . .	24
3.2.1 Debug Toolbar . . . . .	24
3.3 Breakpoints . . . . .	28
3.4 Menu Debug . . . . .	29
3.5 Watch Windows . . . . .	31
3.6 Locals . . . . .	32
3.7 Autos . . . . .	32

3.8	Watch . . . . .	32
3.9	Call Stack . . . . .	33
3.10	Debugging tips . . . . .	34
<b>4</b>	<b>Naslag basiskennis: Variables</b>	<b>35</b>
4.0.1	Typen variabelen . . . . .	35
4.0.2	Variabele aanmaken (declareren) . . . . .	35
4.0.3	Waarde aan variabele geven (toekenning of assignment) . . . . .	36
4.0.4	Variabele aanmaken en direct een waarde geven (declare en initialize) . . . . .	38
4.0.5	Waarden omzetten naar andere typen (convert) . . . . .	38
<b>5</b>	<b>Naslag basiskennis: Int en double bewerkingen (operatoren)</b>	<b>40</b>
<b>6</b>	<b>Naslag basiskennis: Keuzestructuren in C</b>	<b>42</b>
6.0.1	if-statement . . . . .	42
6.0.2	if ... else ... statement . . . . .	43
6.0.3	Voorbeelden “if ...” statement en “if ... else ...” statement . . . . .	43
<b>7</b>	<b>Naslag basiskennis: String bewerkingen (String methods)</b>	<b>46</b>
7.1	String’s samenvoegen . . . . .	46
7.2	IndexOf . . . . .	46
7.3	Substring . . . . .	47
7.4	Length . . . . .	48
<b>8</b>	<b>Naslag basiskennis: String bewerkingen (String methods)</b>	<b>49</b>
8.1	String’s samenvoegen . . . . .	49
8.2	IndexOf . . . . .	49
8.3	Substring . . . . .	50
8.4	Length . . . . .	51
<b>9</b>	<b>Naslag basiskennis: While</b>	<b>52</b>
9.0.1	Voorbeelden while en do while statement . . . . .	53
<b>10</b>	<b>Naslag basiskennis: For</b>	<b>54</b>
10.0.1	for statement . . . . .	54
10.0.2	Voorbeelden for statement . . . . .	55
<b>11</b>	<b>Explanation: Methods in C</b>	<b>58</b>
11.0.1	Algemene structuur methoden . . . . .	58
11.0.2	Belangrijkste voordelen van het gebruik van methoden: . . . . .	58
11.0.3	Voorbeelden Methoden . . . . .	59
<b>12</b>	<b>Explanation - Array &amp; Lists</b>	<b>61</b>
12.1	Array’s . . . . .	61

12.2	Lists . . . . .	62
12.2.1	List methoden . . . . .	63
12.3	Voorbeelden . . . . .	64
12.3.1	List en foreach . . . . .	64
12.3.2	Array en for-loop . . . . .	65
12.3.3	List en Contains . . . . .	65
12.3.4	List en IndexOf . . . . .	66
<b>III</b>	<b>Training assignments</b>	<b>67</b>
<b>13</b>	<b>Training Goeroe-calc: variabelen, bewerkingen en conversies: Een Console app(lication)</b>	<b>68</b>
13.1	Vorbereiding . . . . .	68
13.2	Inleiding . . . . .	68
13.3	Opdracht . . . . .	68
13.3.1	Enkele stappen uitgelegd... . . . .	68
13.4	Extra's . . . . .	70
<b>14</b>	<b>Training Goeroe-calc: variabelen, bewerkingen en conversies</b>	<b>71</b>
14.1	Vorbereiding . . . . .	71
14.2	Inleiding . . . . .	71
14.3	Opdracht . . . . .	71
14.4	Extra's . . . . .	72
<b>15</b>	<b>Software baas</b>	<b>73</b>
<b>16</b>	<b>Euro-Dollar Converter</b>	<b>74</b>
16.1	Doelen . . . . .	74
16.2	Inleiding . . . . .	74
16.3	Bronnen . . . . .	74
16.4	Opdracht . . . . .	74
16.5	Uitbreiding (niveau 3 van 5) . . . . .	75
16.6	Uitbreiding (niveau 4 van 5) . . . . .	75
16.7	Checklist . . . . .	75
16.8	Versies . . . . .	76
<b>17</b>	<b>Training Computer-rekenen</b>	<b>77</b>
17.1	Recht-toe-recht-aan . . . . .	77
17.2	Al wandelend... . . . .	77
17.3	Ik een beetje meer dan jij... . . . .	78
17.4	Ik een beetje meer dan jij... aanzet tot een oplossing . . . . .	78
17.5	Een 1-2-3-tje . . . . .	79

17.6	Hoeveel palindromen kan ik maken met ...?	80
17.6.1	Bronnen	81
17.7	Spoilers...	81
<b>18</b>	<b>Extra opgaven variabelen/strings</b>	<b>82</b>
18.1	Casus 1 - Vind het woord	82
18.2	Casus 2 - BMI-calculator	82
18.3	Casus 3 - Oppervlakte-calculator	82
18.4	Casus 4 - Listbox-vuller	83
18.5	Casus 5 - Ak-tester	83
18.6	Casus 6 - Je werkelijke leeftijd	83
18.7	Casus 7 - Raad-een-getal (if-statement)	83
18.8	Casus 8 - Afstanden omrekenen naar km	84
<b>19</b>	<b>Training - Prijzen met BTW berekenen</b>	<b>85</b>
<b>20</b>	<b>Training - Hoger-lager</b>	<b>86</b>
20.1	Uitbreiding	86
<b>21</b>	<b>Training Calorieën-tracker</b>	<b>87</b>
21.1	Analyse	87
21.2	Ontwerp	87
21.3	Realisatie	87
21.4	Onderhoud	88
<b>22</b>	<b>Training - Dobbelsteen</b>	<b>89</b>
22.1	Opdracht	89
22.2	Uitbreidingen	89
22.3	Checklist	91
22.4	Bronnen	91
<b>23</b>	<b>Training - Pincode reminder</b>	<b>92</b>
23.1	Opdracht	92
23.1.1	Tips	93
23.2	Uitbreidingen	93
23.3	Checklist	93
<b>24</b>	<b>Training - Wachtwoord check</b>	<b>94</b>
<b>25</b>	<b>Training - Dr. Bobby</b>	<b>95</b>
<b>26</b>	<b>Training - Batman vs Superman</b>	<b>96</b>
<b>27</b>	<b>Training - For - Drankjesschema</b>	<b>97</b>

<b>28 Training - For - Wiskunde en zo</b>	<b>98</b>
28.1 Casus 1 - De tafel van 3 . . . . .	98
28.2 Casus 2 - De tafel van N . . . . .	98
28.3 Casus 3 - Lus met een if er in . . . . .	98
28.4 Casus 4 - Geneste lus . . . . .	99
28.5 Casus 5 - Som of product van bepaalde reeks getallen . . . . .	99
28.6 Casus 6 - Som van een reeks getallen . . . . .	99
<b>29 Training - For - Worpengenerator</b>	<b>100</b>
29.1 Deel 1 . . . . .	100
29.1.1 Tips . . . . .	100
29.2 Deel 2 . . . . .	101
29.2.1 Tips . . . . .	101
29.3 Deel 3 . . . . .	101
29.4 Uitbreidingen . . . . .	102
29.5 Checklist . . . . .	102
<b>30 Training - For - Muziekgenerator</b>	<b>103</b>
<b>31 Training - While - Het grote geheel</b>	<b>104</b>
<b>32 Training- While - Het goede getal</b>	<b>105</b>
<b>33 Training - Christmas Tree Generator</b>	<b>106</b>
33.1 Stappenplan . . . . .	106
33.2 Uitbreidingen . . . . .	106
33.3 Bronnen . . . . .	108
<b>34 Training - Emmer vol laten lopen</b>	<b>109</b>
34.1 Bronnen . . . . .	110
<b>35 Training method Ampere</b>	<b>111</b>
<b>36 Training method MaalDrie</b>	<b>112</b>
<b>37 Training De Drie Vierkanten</b>	<b>113</b>
<b>38 Training - Listige listboxen</b>	<b>114</b>
<b>39 Training - Mad lad teksten</b>	<b>115</b>
<b>40 Training met methode ElkaarAchterstevoren</b>	<b>116</b>
<b>41 Training - Cijfers door woorden vervangen</b>	<b>117</b>

<b>42 Training method marathon</b>	<b>118</b>
<b>43 Heb ik het allemaal goed gedaan?</b>	<b>121</b>
<b>44 Eerste Hulp Bij Vasthangen</b>	<b>122</b>
<b>45 Training - Galgje</b>	<b>123</b>
45.1 Uitbreidingen . . . . .	123
<b>46 Training: Method cijfers naar woorden</b>	<b>124</b>
<b>47 Training - Boter, kaas en eieren tegen de computer</b>	<b>125</b>
<b>48 Training - Array - Televisie</b>	<b>126</b>
48.1 Deel 1 - De televisie aan en uit . . . . .	126
48.2 Deel 2 - Zenders toevoegen . . . . .	127
48.3 Deel 3 - Zenders laten zien in een listbox . . . . .	127
48.4 Deel 4 - Zenders toevoegen via een textbox . . . . .	128
48.5 Deel 5 - Zappen en huidige zender laten zien . . . . .	128
48.5.1 Uitbreiding . . . . .	129
<b>49 Training - Array - Netflix and random</b>	<b>130</b>
49.1 Uitbreidingen . . . . .	130
<b>50 Training - Array - Fruitautomaat</b>	<b>132</b>
<b>51 Training - List - Prijzen toevoegen</b>	<b>133</b>
<b>52 Training - List - Ticketmeester</b>	<b>134</b>
52.1 Uitbreidingen . . . . .	134
52.2 Bronnen . . . . .	135
<b>53 Training - Listmethodes - Digitale muziekcollectie</b>	<b>136</b>
<b>54 Training - Listmethodes - Raad het getal</b>	<b>137</b>
<b>55 Training - Listmethodes - Vlaggen van de wereld</b>	<b>138</b>
<b>56 Training - Array - Woordvervormer</b>	<b>139</b>
56.1 Tips . . . . .	139
<b>57 How To ... (problem oriented)</b>	<b>140</b>
57.1 WinForms . . . . .	140
57.1.1 Meerdere Forms in een app . . . . .	140
57.1.2 Informatie doorgeven tussen Forms . . . . .	140

57.2 Bron . . . . .	141
<b>58 Training Rock Paper Scissors</b>	<b>142</b>
<b>59 Training kaartspel</b>	<b>143</b>
<b>60 Training van getal naar woord</b>	<b>144</b>
60.1 Uitbreidingsmogelijkheden . . . . .	144
<b>61 Training Blackjack (light)</b>	<b>145</b>
61.1 Regels . . . . .	145
61.2 Puntentelling . . . . .	145
61.3 Uitbreidingen . . . . .	145
<b>62 Summary</b>	<b>146</b>
<b>References</b>	<b>147</b>



# Preface

This is a Quarto book.

To learn more about Quarto books visit <https://quarto.org/docs/books>.

# 1 Introduction

Dit is een verzameling van programmeerassignments die als training gebruikt kunnen worden. Concepten die hierin aan bod komen:

- Variable
- Conditional statement
- Loop
- Method
- Parameters bij Methods
- Returnvalues bij Methods
- List
- Array
- Enum
- Type int, string, double, bool

Ook vaardigheden als:

- Leesbaarheid / Onderhoudbaarheid
- Algoritmie
- Feedback vragen en dat verwerken
- Professioneel handelen

## 1.1 Intro personas

In den beginne... had nog niemand programmeerervaring... Op het moment dat je aan dit semester begint heb je misschien al ergens ervaring opgedaan, of wellicht niet. Kortom: we zijn allen gelijk in dat we allemaal uniek zijn!

Als je al ervaring hebt is het jammer helemaal aan het begin te beginnen. Vandaar dat we verschillende persona's beschreven hebben. Bepaal welke het best bij je past en kijk eens wat er over jouw persona gezegd wordt.

---

Bas (of Barbara) the  
Absolute Beginner

Intermediate Inge (of Ivo)

OOP-experienced Olaf (of  
Olivia)

---



BAS



INGE



OLAF

---

# **Part I**

## **Tools**

## 2 Visual Studio installatie

### 2.1 Wat heb ik nodig?

Een ontwikkelomgeving (kort: IDE). Als je nog geen ervaring hebt raden we de **default** aan:

#### 2.1.1 Default: VS for MS-Windows

Visual Studio (afgekort: VS). De default is **VS for MS-Windows**. In sommige opdrachten zie je (nog) screenshots van WinForms-apps, die je alleen in deze versie kunt maken. In andere omgevingen moet je eerst bedenken hoe je hetzelfde effect kunt bereiken. Ook kan het zijn dat een docent of mede-student iets met WinForms voordoet: Het is dan het makkelijkste als jij dezelfde versie gebruikt.

#### 2.1.2 Alternatief: Visual Studio Code

Als je nog geen programmeerervaring hebt raden we dit af, omdat sommige dingen net anders werken dan bij je mede-studenten en docenten: wellicht moet je zelf wat dingen uitzoeken.

- Installatie is lastiger.
- WinForms wordt niet ondersteund in ‘VS Code’.
- verder een prima programmeeromgeving.
- wordt ook bij Technology Verdieping gebruikt.

#### 2.1.3 En op een Mac?

- Default: We raden aan Bootcamp of eventueel een Virtual Machine (bijvoorbeeld VMWare of Parallels) te draaien met MS-Windows en daarin **VS for Windows**. Dit vreet wel schijfruimte en kan zo nu en dan wat trager zijn, maar je hebt dezelfde IDE als medestudenten onder Windows.
  - <https://support.apple.com/nl-nl/boot-camp>
- **VS for Mac**: ondersteunt geen WinForms; short-cut-keys zijn anders; verder een prima programmeeromgeving.

- **Rider:** Fast & powerful cross-platform .NET IDE: ondersteunt WinForms. Het is een onderdeel van JetBrains en via je studenten account kan je gratis deze software gebruiken.
- **Visual Studio Code:** zie boven.
- De ‘Online FHICT werkplek’: zie boven.

Voor zowel bootcamp alsmede VM oplossing heb je extra software nodig. Deze kun je gratis vinden in de webshop die hierboven gelinkt is.

## 2.2 Installatie

### 2.2.1 Een Manier

Je zoekt op internet naar "Visual Studio 2017 community". [Bijvoorbeeld hier](#)

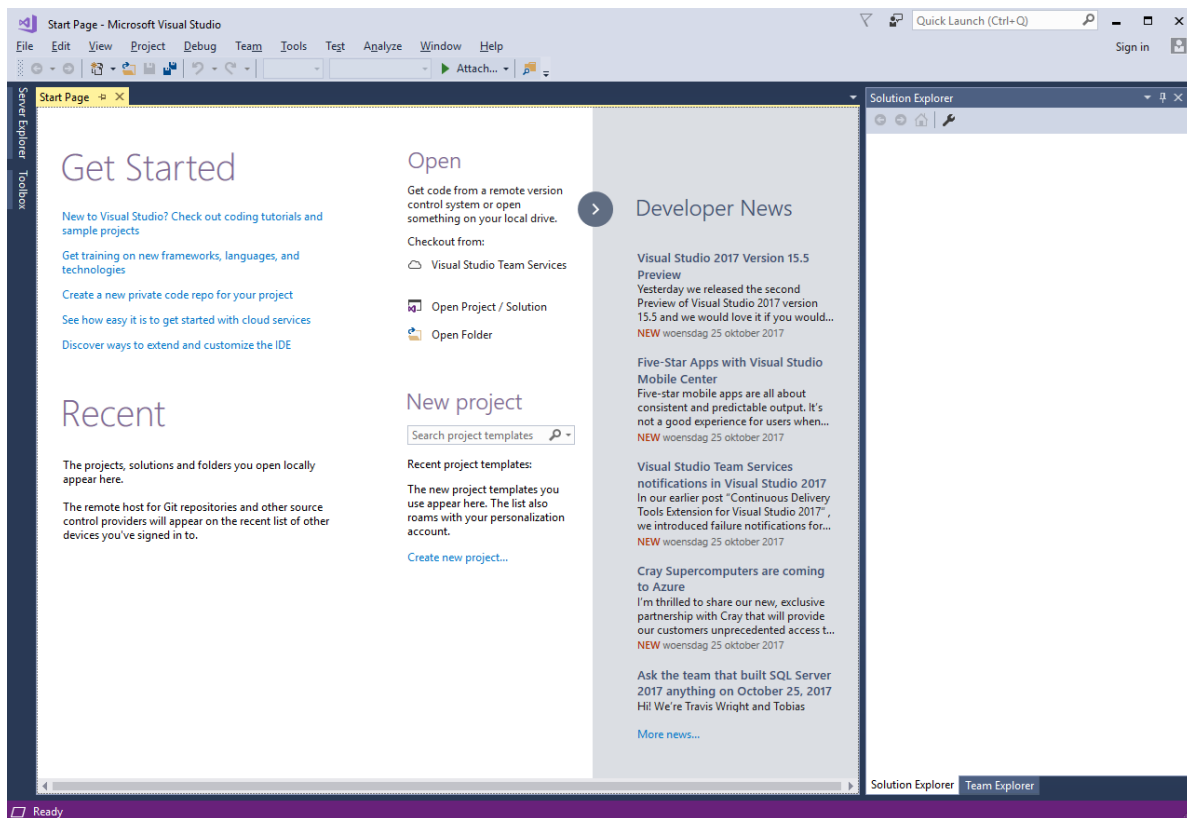
De gratis versie van VS noemen ze de Community Edition. Kies bij het installeren voor ".NET desktop development". Deze versie is voor het startsemester voldoende. Maar....

#### 2.2.1.1 Een andere manier: Via Studentenplein

Het kan ook via [studentplein](#) op de [FHICT-portal](#), onder het kopje ‘Software’.

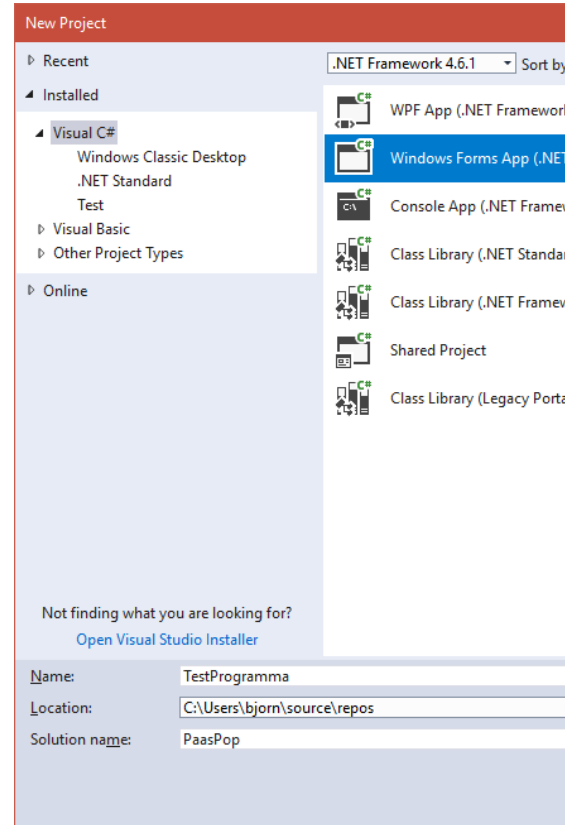
## 2.3 Je eerste programma (WinForm, werkt alleen in VS for MS-Windows)

Wil je weten of je VS werkt? Start Visual Studio op. We maken eerst een nieuw project aan en gaan daarna C# sourcecode intikken. Vervolgens starten we de C# sourcecode op. Je eerste programma!



Als je dit ziet na de installatie, dan ben je startklaar voor de volgende stap.

### 2.3.1 Windows Forms App C



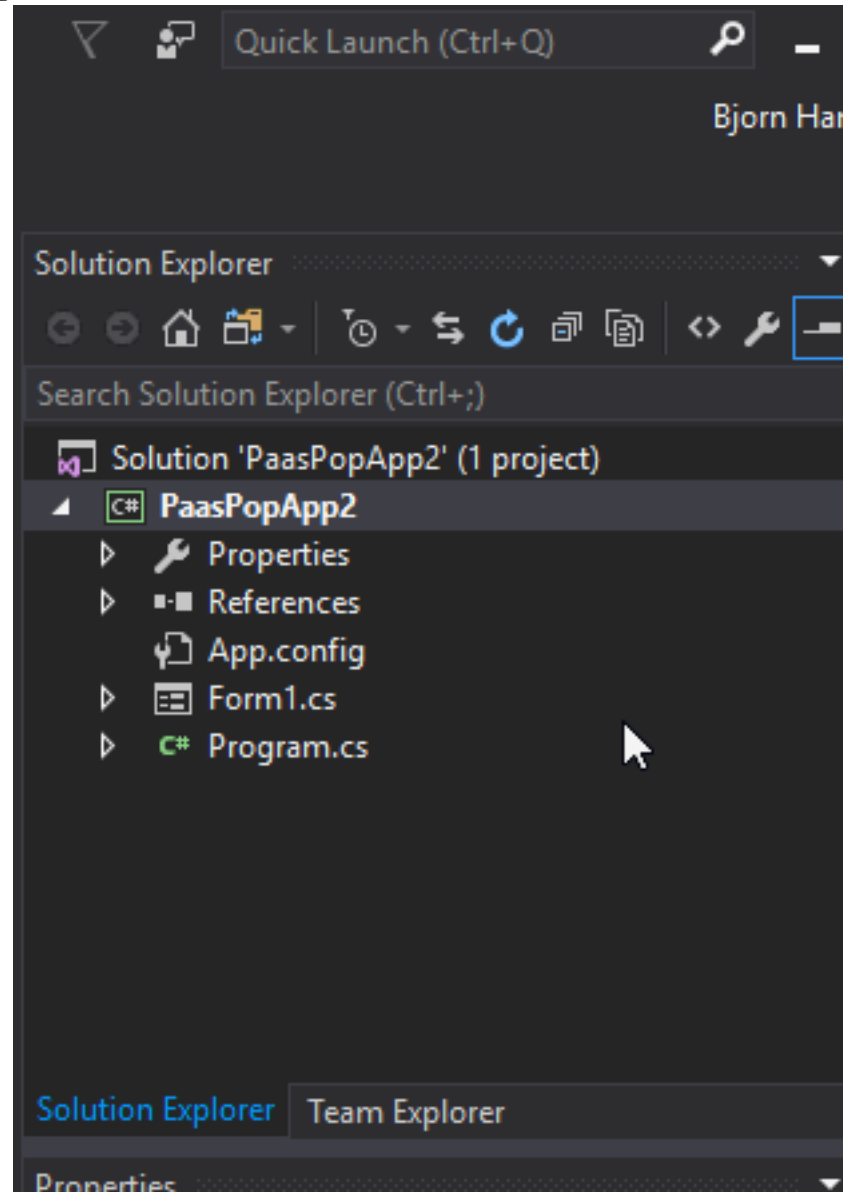
Kies in het menu 'File' voor 'New' en dan 'Project'. Zie New Project.

Zorg ervoor dat je het blauwe deel exact hetzelfde laat zien als bij jou op het scherm. Sleutelwoorden: "Visual C#" staat links geselecteerd en er staat "Windows Forms App (.NET Framework) Visual C#". Geef het project een zelfgekozen naam bij "Name" en klik vervolgens op "OK".



### 2.3.2 Tikken C# Sourcecode

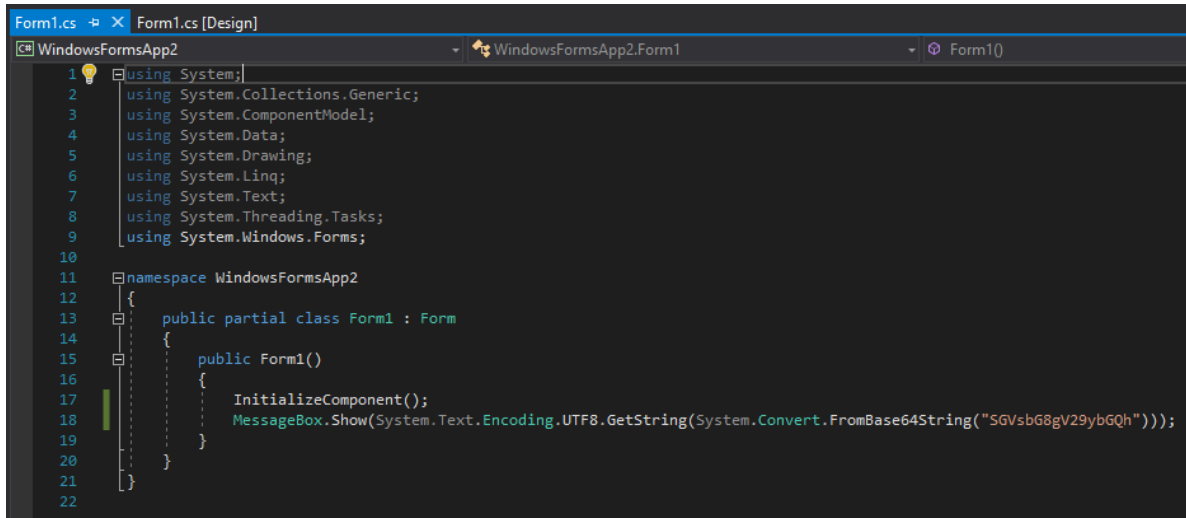
Om bij de source code van je eerste (lege) programma te komen moeten we die zichtbaar maken.



Klik rechts op 'Form1' en kies 'View Code F7'.

Je ziet nu in het midden C# source code. Kopieer de regel hieronder en plak die onder de regel die er al staat. Maak het zo dat je scherm er hetzelfde uit ziet als hieronder. Het gaat om het toevoegen van alleen deze regel. Sommige namen in de regels erboven en eronder heten wat anders. Dat komt omdat in het voorbeeldprogramma de namen misschien net wat anders gekozen zijn. Dat is niet erg.

```
MessageBox.Show(System.Text.Encoding.UTF8.GetString(System.Convert.FromBase64String("SGVsbG8gV29ybGQh"));
```

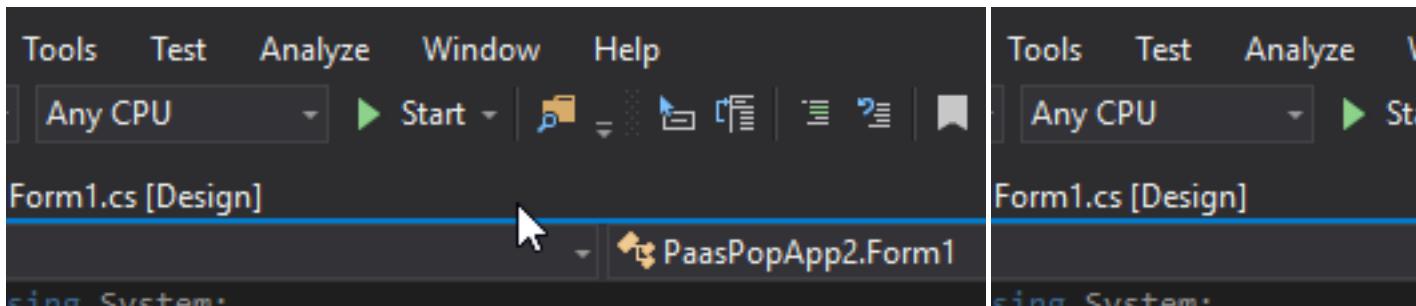


```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace WindowsFormsApp2
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18             MessageBox.Show(System.Text.Encoding.UTF8.GetString(System.Convert.FromBase64String("SGVsbG8gV29ybGQh")));
19         }
20     }
21 }
22
```

Wezenlijk voeg je dus alleen maar "MessageBox..."-regel toe onder de "InitializeComponent()" -regel. Deze regel bevat een geheime code die een boodschap aan je laat zien bij het opstarten.

### 2.3.3 Opstarten eerste programma

Nadat je C# sourcecode hebt getikt moet je Visual Studio de opdracht geven om het programma te maken (compileren) en uit te voeren (runnen). Dat doe je met de knop "Start". Je krijgt na het uitvoeren een boodschap te zien.



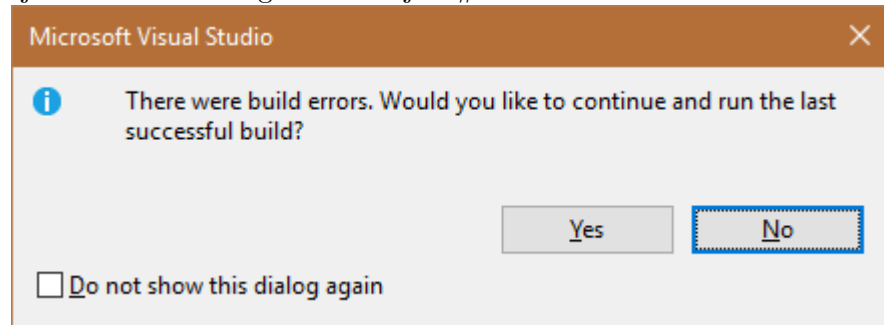
Het valt je op dat Visual Studio twee "gezichten" heeft. Een bewerkingsmodus en een uitvoeringsmodus.

- De **bewerkingsmodus** gebruiken we het meest. Daar kun je knoppen programmeren en C# sourcecode tikken. Visual Studio start op in deze modus en in deze modus hebben we net die regel C# sourcecode toegevoegd. In deze modus kunnen we ons nieuwe programma opstarten met "Start".

- Nadat je op "Start" klikt verspringen alle icoontjes. Soms blijft je C# sourcecode nog staan, soms niet. Je ziet Visual Studio je programma opstarten en grafieken tekenen van je CPU/Memory. Visual Studio is in deze modus bezig met het uitvoeren van je programma. Klik op het stop-blokje (zie afbeelding hierboven) om te stoppen met het uitvoeren en terug te gaan naar de bewerkingsmodus.

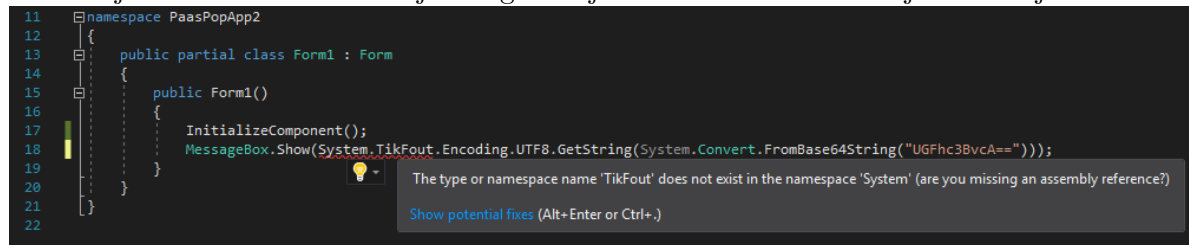
### 2.3.4 Het werkt niet?

Nu kan het gebeuren dat je een tikfout hebt gemaakt in je C# sourcecode. Dat laat Visual Studio zien met dit scherm:



dit zien met dit scherm:

Kies altijd voor "No"! Dan kun je terug naar je Visual Studio om te kijken waar je tikfout zit.



Tikfouten geeft Visual Studio aan met rode kringeltjes onder de woorden. Net zoals bij Word. Deze zijn soms moeilijk te begrijpen. Haal dan de regel weg en tik hem opnieuw. Je kunt ook het programma afsluiten en terug gaan naar een laatst werkende versie. (Niet opslaan.)

### 2.3.5 Hoe verder?

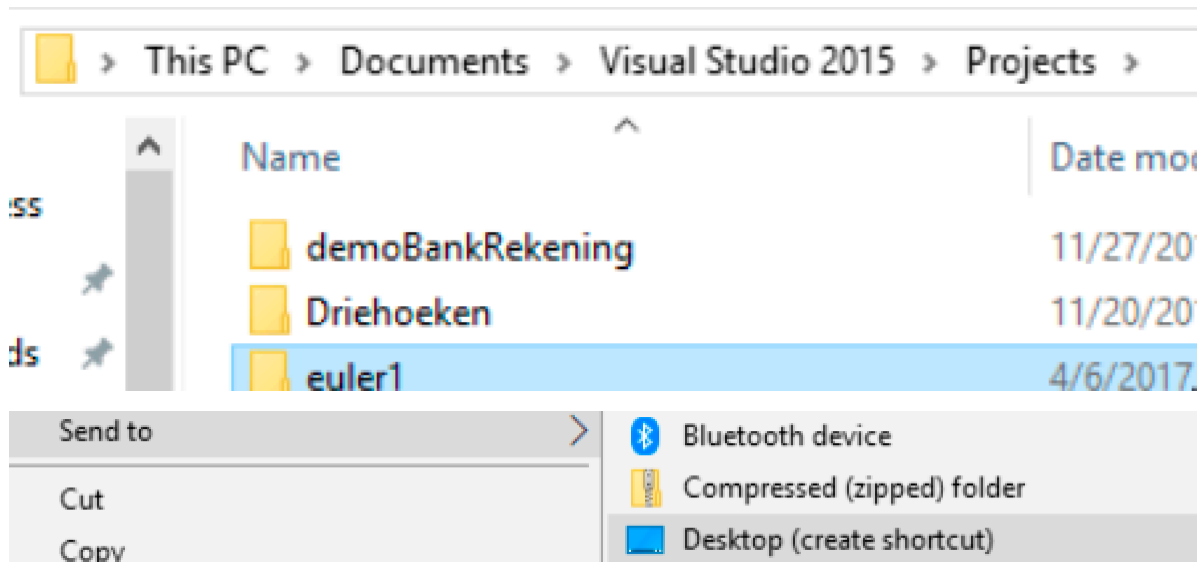
Probeer de designer te openen van Form1 (rechtsklik weer op Form1, net als bij het openen van de code). Sleep wat knoppen (buttons), tekst (labels), en dergelijke op je Form1. Pas de kleur en tekst aan. Maak er iets moois van! Als je een knop dubbelklikt kun je deze code gebruiken om neer te zetten dat dan zichtbaar wordt als je op de knop drukt. (Plak deze code op de lege regel tussen de { en }.)

```
MessageBox.Show("Dank je wel voor het klikken!");
```

## 2.4 Inleveren van Visual Studio solutions

Als je gemaakte Visual Studio solutions wil inleveren in Canvas, hou dan rekening met het volgende:

- Laat Visual Studio tijdens het programmeren alle files opslaan op de door Visual Studio voorgestelde plek. (ga dus *niet* losse files met behulp van **save as** op andere plekken opslaan).
- Het gevolg hiervan is dat alle files van een solution in 1 directory bij elkaar staan.
- De default plek waar Visual Studio solutions opslaat is C:\Users\- Als je 1 solution wilt inleveren kun je de directory van die solution zippen door in het *rechtermuisknop* van de directory **send to**|Compressed (zipped) Folder te kiezen.
- Als je meerdere solutions in 1 zip wilt inleveren kun je er meerdere selecteren (met **ctrl** of **shift** ingedrukt en dan de directories aan te klikken), ook weer met hetzelfde *rechtermuismenu* te zippen.
- Gebruik in Canvas de Submit-button om de zip in te leveren.



## 2.5 Short cut keys in Visual Studio

Als in een sneltoets combinatie een komma staat, dan moet eerst het deel voor de komma worden ingedrukt, dan laat je de toetsen los en druk je daarna de letter in die na de komma staat.

Sneltoets	Menu	Toelichting
CTRL-W, X	View → Toolbox	Maakt het Toolbox scherm met alle visuele objecten zichtbaar.
CTRL-W, P	View → Properties	Window Scherm met de eigenschappen van visuele objecten tonen. Hier vind je ook de events van de visuele objecten.
CTRL-W, E	View → Error List	Toont het scherm met de eventuele fouten die in je code gevonden zijn (ververs het scherm door je project opnieuw te build-en met de F6 knop, zie hieronder). Dubbelklikken op een error navigeert naar de plaats in je code waar de fout is gevonden. Tip: bekijk eerst de eerste fout, andere fouten kunnen hier een gevolg van zijn.
F6	Build → Build Solution	Controleert je code op fouten en bouwt vervolgens een uitvoerbaar bestand.
F5	Debug → Start Debugging	Build je project (zie F6 hierboven) als dat nog niet gebeurd is, en voert het uitvoerbaar bestand vervolgens uit. Dit doet hetzelfde als een klik op de knop

Sneltoets	Menu	Toelichting
Shift-F5		met het groene pijltje (playknop). Stop de uitvoer van je programma. Handig als je programma vast loopt.

# **Part II**

# **Theory**

## 3 De Debugger

Activiteit: Kennismaken met de debugger van Visual Studio

### 3.1 Inleiding

Bij softwareontwikkeling nemen testen en fouten herstellen meer tijd in beslag dan het daadwerkelijk code schrijven. Met een debugger kun je geen fouten opsporen maar alleen lokaliseren. Een debugger bestaat uit een verzameling tools waarmee je **breakpoints** (stoppunten) kunt zetten, en waarmee je **stepping** (stap voor stap uitvoeren) kunt toepassen. Tevens kun je variabelen inspecteren en kun je de **call stack** en nog veel meer aspecten bekijken.

```
kjhdsfkjhdsfkjhdsf ## kjhdfkjehskfg [[debugger_breakpoint.png]]
```

### 3.2 Opdracht

In deze tutorial ga je een aantal mogelijkheden van de debugger onderzoeken. Maak een **Console Application** project aan. Noem het project “Debuggen”.

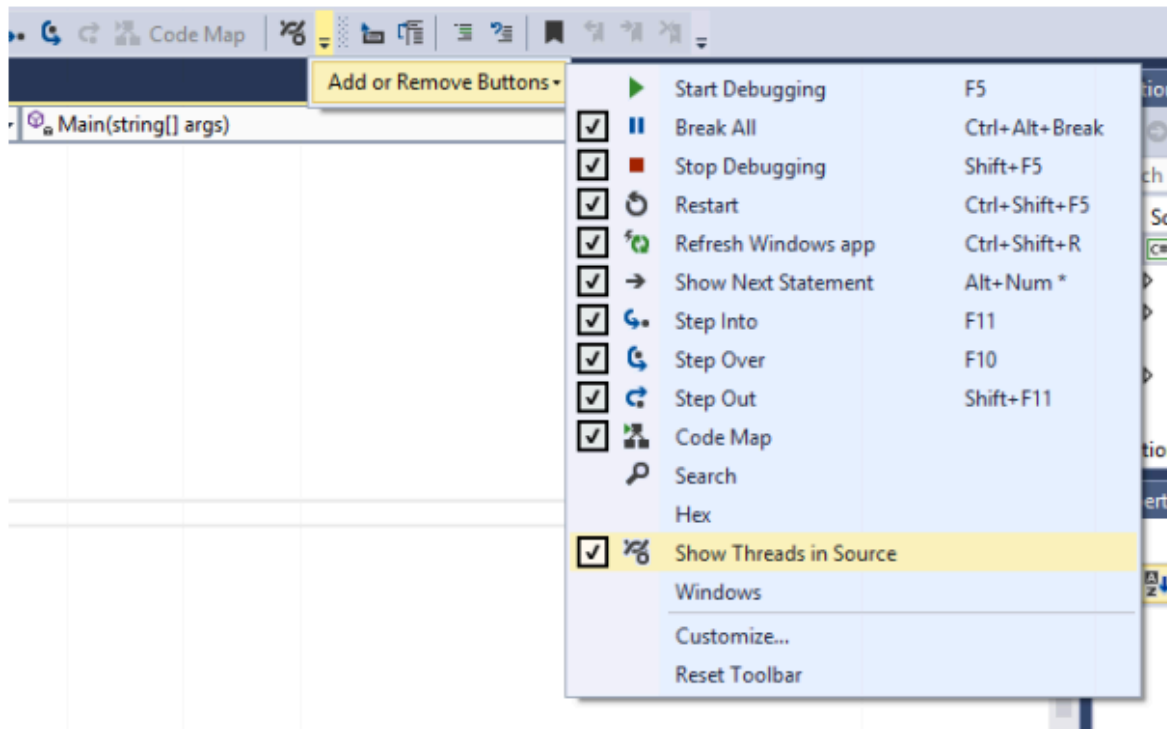
#### 3.2.1 Debug Toolbar

Open in Visual Studio het menu **View > Toolbars**. Je krijgt dan alle toolbars die beschikbaar zijn voor de gebruiker te zien. Zorg dat **Debug** aangevinkt is. Dan krijg je de onderstaande toolbar.



Het drop-down menu naast de **Breakpoints**-knop bevat de mogelijkheid om meer of minder buttons zichtbaar te laten zijn in de toolbar. Vink bijvoorbeeld “Show/Hide Thread” uit.





Zoals je ziet zijn ook de sneltoetsen in dit menu weergegeven. Dubbelklik in de **Solution Explorer** van Visual Studio op “Program.cs” om dit bestand te openen. Je ziet dan de code van de class “Program” in de editor.

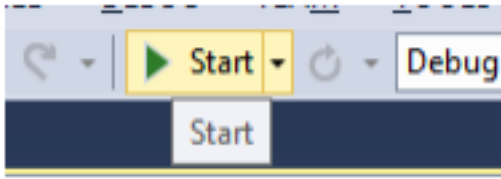
Type de onderstaande code in de “Main”-method (tussen de accolades)

```
string[] namen = { "naam1", "naam2", "naam3", "naam4", "naam5", "naam6" };

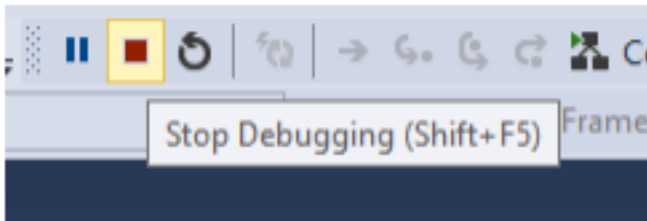
foreach (string naam in namen)
    Console.WriteLine(naam);
    Console.ReadLine();
```

Wat gebeurt er als deze code wordt uitgevoerd? Beschrijf het voordat je het programma uitvoert.

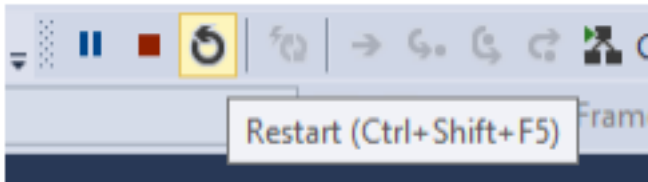
Een druk op de **Start**-knop heeft hetzelfde effect als F5. De applicatie runt na een eventuele build.



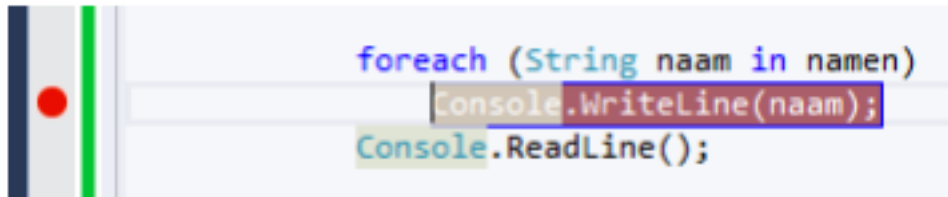
Stop Debugging stopt de debugger met runnen en gaat terug naar de ontwerpomgeving.



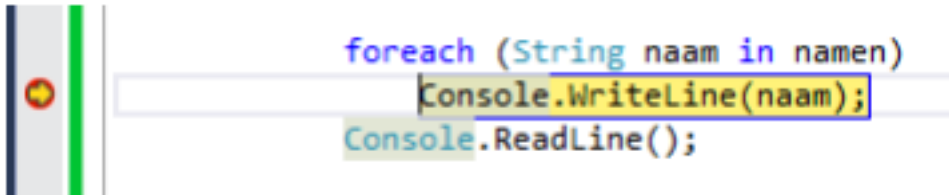
Restart is de herstart van de debugger vanaf het begin, alle variabelen worden gereset.



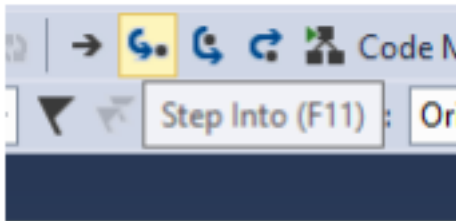
Ga met de cursor op de regel `Console.WriteLine(naam);` staan en druk op F9. Je ziet nu een **BreakPoint** in de kantlijn verschijnen (Het rode bolletje).



Als je nu het programma runt pauzeert het bij het **BreakPoint**. Deze regel wordt nog niet uitgevoerd. Je ziet dat de regel nu geel is gekleurd en een gele pijl wijst naar de regel die nog niet uitgevoerd is.



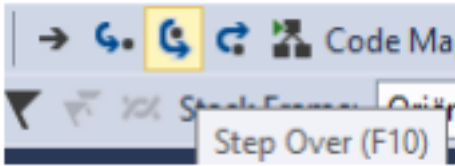
**Step Into** laat regel voor regel door de code stappen. Dit betekent dat je ook in een methode kunt gaan die worden aangeroepen.



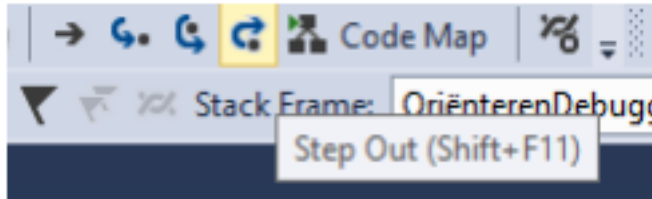
Stap met F11 door deze code heen. In het Command Window zie je het resultaat hiervan weergegeven.



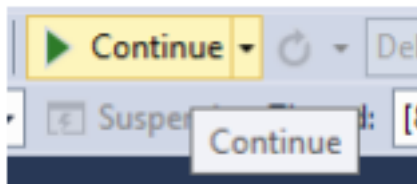
**Step Over** laat je ook stap voor stap door de code lopen maar gaat niet de methodes in. Dat is handig als je alleen het resultaat van een methode wilt zien. Dat kan is onze code niet bekeken worden, we roepen namelijk geen andere methode aan.



**Step Out** is handig als je niet langer door een methode wilt lopen. Dat heb je nodig als je in een systeemfunctie bent beland. Maar ook als je gewoon snel naar het volgende statement wilt kun je deze knop gebruiken.



**Continue** is handig als je tijdens het debuggen gezien hebt wat je moet zien. Dus als je al gezien hebt dat het goed gaat. Als je op Continue klikt gaat de debugger naar het volgende breakpoint of naar het einde van het programma als er geen breakpoint meer is.

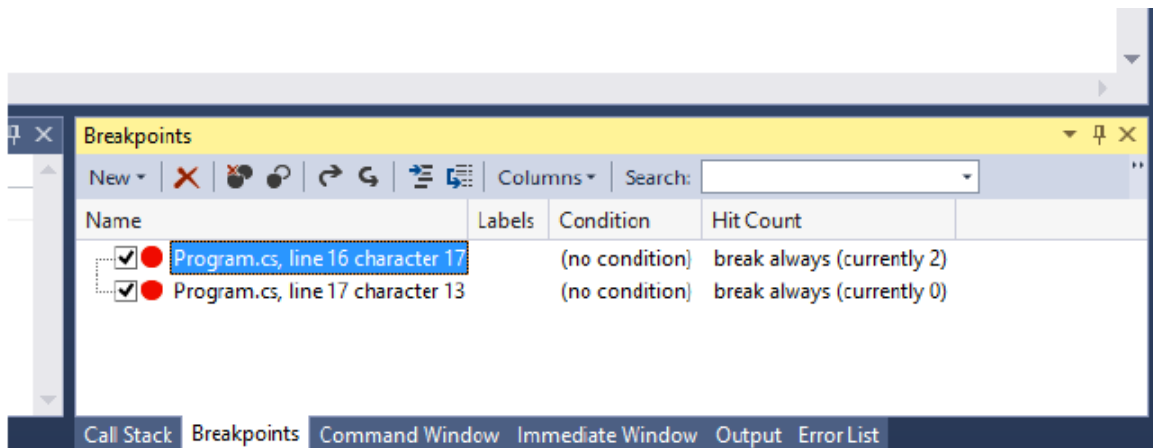


### 3.3 Breakpoints

**BreakPoints** zijn erg belangrijk bij debuggen. Je kunt een **BreakPoint** ook zetten door éénmaal op de grijze verticale balk aan de linkerkant van je scherm te klikken.

**BreakPoints** opent het **BreakPoint Window** en integreert het samen met andere windows onderaan in de Visual Studio omgeving. Dit window bevat informatie over alle **BreakPoints** die op dit moment zijn gezet.

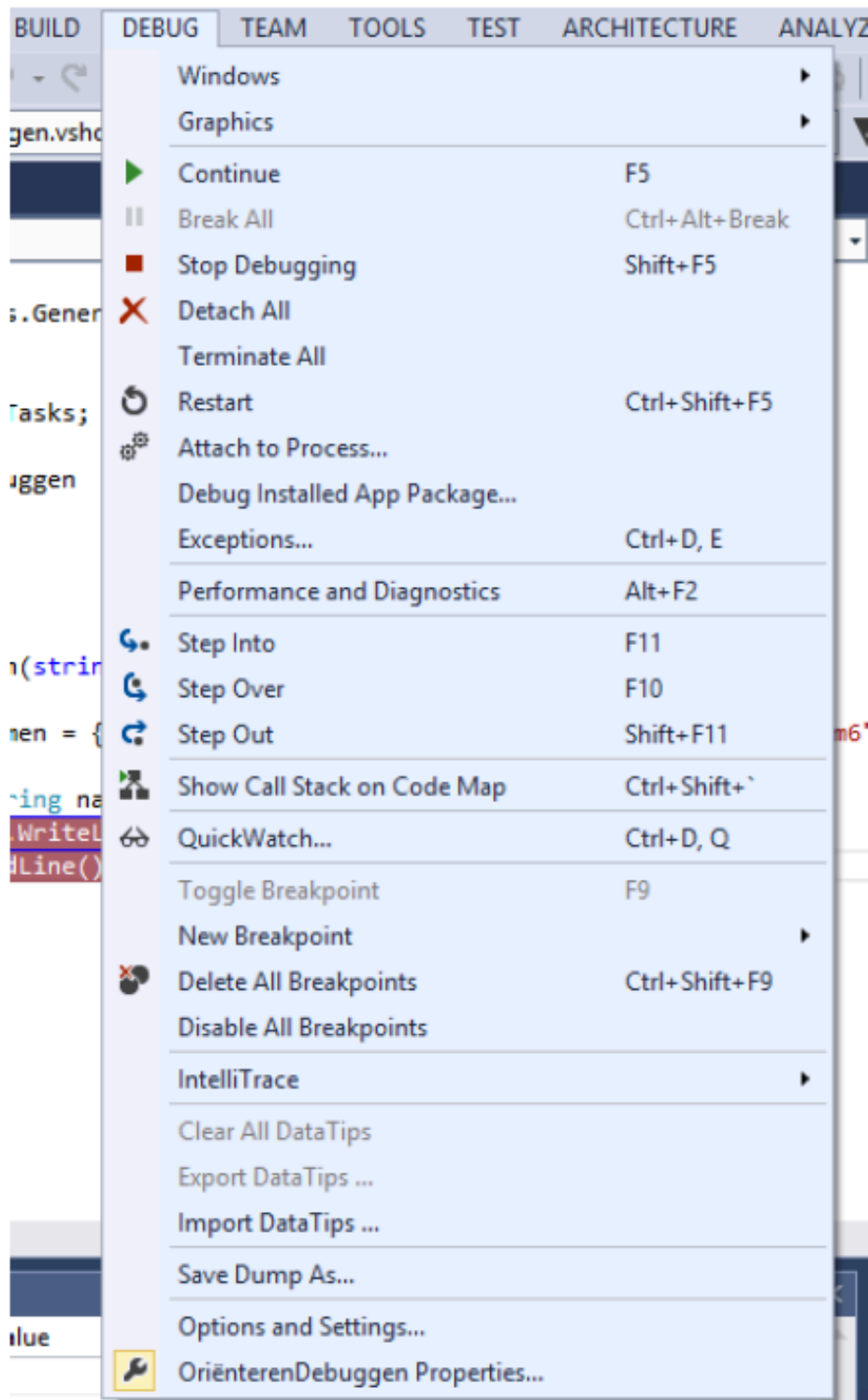
Zet in je programma ook nog een **BreakPoint** op methodenaam “Main” en bekijk het **BreakPoints Window**.



In het BreakPoints Window zie je dan weergegeven waar het breakpoint in je code staat.

### 3.4 Menu Debug

Het Visual Studio Debugmenu bevat alle bovenstaande selecties voor degene die liever met menu's werken dan met toolbars.



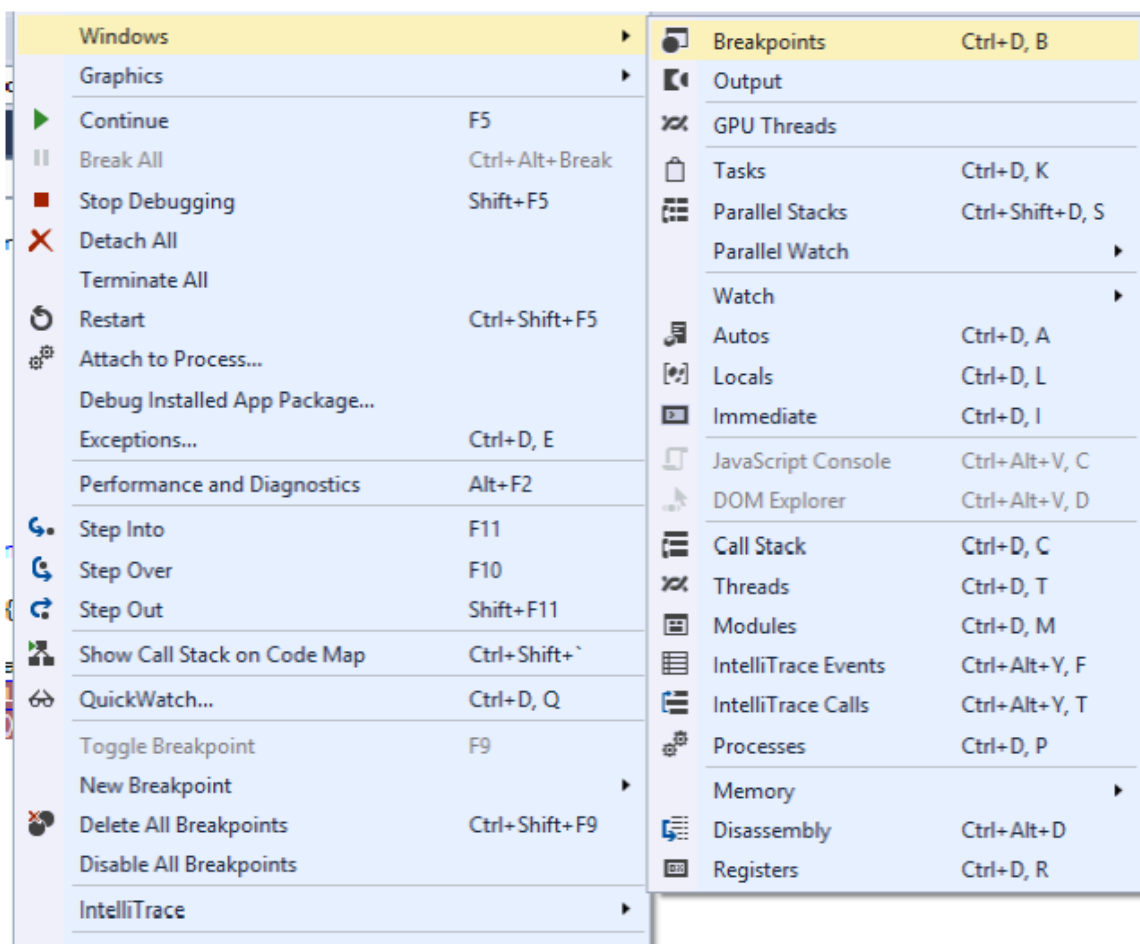
## 3.5 Watch Windows

Er is een aantal vensters tot je beschikking om de variabelen van het programma te bekijken. Tot nu toe zagen we het **BreakPoints Window** onderaan in de IDE weergegeven. Met het pijltje naast het icoon van de **BreakPoints** in de toolbar krijg een dropdownmenu met andere mogelijkheden.

Maar ook komt dit menu als je kiest voor **Windows** onder **Debug**.

Let op, je krijgt alleen alle mogelijkheden te zien als je bij een breakpoint staat.

Er zijn verschillende soorten vensters zoals Autos, Local, etc.

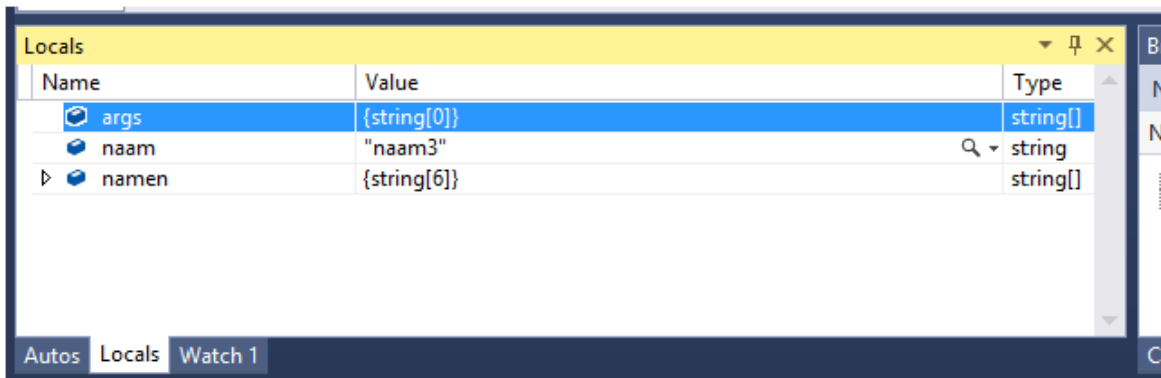


Als je programma pauzeert bij een **BreakPoint** kun je via zo'n venster bijvoorbeeld de waarden van het huidige object of variabelen inspecteren. Een mooie optie is ook de muis hover, als

je over de variabele schuift met de cursor krijg je via een `ToolTip` ook de informatie van de variabele te zien.

## 3.6 Locals

Dit scherm laat automatisch de lijst van variabelen zien die binnen de scope van een methode vallen.



Wat gebeurt er als je de Hex knop van de Debugtoolbar indrukt?

## 3.7 Autos

In dit venster worden de variabelen weergegeven waar de debugger staat en de variabelen in het statement wat ervoor staat.

## 3.8 Watch

In dit venster kun je zelf variabelen toevoegen die je wilt bekijken. Dit doe je door met je rechter muisknop te klikken op de variabele en dan “Add To Watch” te kiezen. Je kunt de variabelen ook naar het venster slepen.

Om dit uit te proberen moet er wat code worden toegevoegd aan je programma. Vervang de regel `Console.ReadLine();` in de `Main`-methode door:

```
int temp = 4;
for (int i = 0; i < 10; i++)
{
```



```

        if (i > 5)
            temp = 5;
            Console.WriteLine(temp);
    }
    Console.ReadLine();

```

Wat gebeurt er als deze code wordt uitgevoerd? Beschrijf het voordat je het programma runt.

Voeg de variabele `temp` toe aan het Watch scherm en volg de waarde van deze variabele terwijl je door de code heenloopt met de stepfunctie.

### 3.9 Call Stack

Call Stack informatie kan belangrijk zijn om te achterhalen hoe een programma in een bepaald stuk van de code terecht kwam.

Als je meerdere methodes aangeroepen hebt en je wilt zien of de geneste methoden wel worden aangeroepen dan is de Call Stack nodig.

Zet de onderstaande code in het programma buiten de Main-methode. Snap je hoe deze code werkt?

```

public static void Methode1()
{
    Console.WriteLine("Dit is methode1");
    Methode2();
}

public static void Methode2()
{
    Console.WriteLine("Dit is methode2");
    Methode3();
}

public static void Methode3()
{
    Console.WriteLine("En dit is methode3");
    Console.ReadLine();
}

```

Vervang alle code in **Main**-methode (dus tussen de accolade) door een aanroep van **Methode1**. Dus door de code **Methode1()**;

Je hebt nu geneste methodenaanroepen gemaakt. **Methode1** wordt aangeroepen in de **Main**-methode. Dan roept **Methode1** na een **WriteLine**-statement **Methode2** aan. De **Methode2** roept op zijn beurt na een **WriteLine**-statement **Methode3** aan. **Methode3** heeft eerst een **WriteLine**-statement en wacht dan tot dat er een toets wordt in gedrukt.

De **Call Stack** laat tot een **BreakPoint** de nesting zien. Zet een **BreakPoint** bij de aanroep van **Methode1** en loop met de debugger door de code en bekijk hoe de **Call Stack** de aanroepen laat zien.

Als je in de **Call Stack** op een rij klikt, zie je in de code de bijbehorende code groen kleuren.

Wen je zelf eraan om gebruik te maken van de debugger. Op de duur kost het minder tijd om fouten op te sporen.

### 3.10 Debugging tips

- [Debugging in .NET 6 / nov2021](#): behoorlijk zinvolle tips over hoe je moet debuggen in Visual Studio. Zeker nuttig, want als ontwikkelaar ben je aanzienlijk meer tijd kwijt met het opsporen en oplossen van fouten dan met het schrijven ervan :^) Als je skipped naar 9:00 ofzo, kun je het hele voorstelrondje van de presentatoren overslaan en meteen de diepte in. De audio is af en toe een beetje brak, maar de tips zijn top-notch.

## 4 Naslag basiskennis: Variables

Dit hoofdstuk is geschreven als een *naslagwerk*, het is niet specifiek geschreven om uit te leren hoe je met variabelen programmeert.

### 4.0.1 Typen variabelen

Een variabele is een stukje geheugen waarin tijdelijk een waarde kan worden opgeslagen. De veelgebruikte typen variabelen zijn:

Inhoud	Naam	Voorbeelden
Stukje tekst	String	“abcde” “dit is een tekst” “ ” etc.
Geheel getal	Int	12 -1337 0 etc.
Komma getal	Double	10.2 -12.3 5.0 etc.
Waar of niet waar	Bool	true, false

### 4.0.2 Variabele aanmaken (declareren)

Variabelen kunnen op verschillende manieren worden aangemaakt, enkele voorbeelden staan hier-onder. Merk op dat: - Je de variabele naam zelf kunt kiezen - De regel moet worden beëindigd met een `”;`-teken

Op verschillende manieren kunnen variabelen worden aangemaakt. Programmeer op een lege regel het type van de variabele (zie hierboven), de naam die je de variabele wil geven (deze kies je zelf) en een `”;` teken om het programmeercommando af te sluiten.

Voorbeeld	Effect
String s;	Variabele met de naam s wordt aangemaakt. De default waarde is “”.
int i;	Variabele met de naam i wordt aangemaakt. De default waarde is 0.
double d;	Variabele met de naam “d” wordt aangemaakt. De default waarde is 0.0
Bool b;	Variabele met de naam “b” wordt aangemaakt. De default waarde is false
String mijnString;	Variabele met de naam “mijnString” wordt aangemaakt. De default waarde is “”
int getal;	Variabele met de naam “getal” wordt aangemaakt. De default waarde is 0
double straal;	Variabele met de naam “straal” wordt aangemaakt. De default waarde is 0.0

Direct na het aanmaken heeft een **variabele** een **waarde** die we de **default waarde** noemen. Dit kan per programmeertaal enigszins verschillen. Daarom is het een goede gewoonte variabelen waarvan je wil dat ze een specifieke waarde hebben deze waarde expliciet toe te kennen.

#### 4.0.3 Waarde aan variabele geven (toekenning of assignment)

Als een variabele eenmaal is aangemaakt kan hier een waarde aan worden toegekend. Merk op:

- Alleen geldige waarden kunnen worden toegekend (string waarden aan strings, getallen aan int, etc.), het programmeren van een niet geldige toekenning levert een fout op waardoor het programma niet kan worden uitgevoerd.
- De variabele waaraan een waarde moet worden toegekend staat aan de linkerkant van het “=” teken, en de waarde welke in de variabele moet worden gestopt staat rechts van het “=” teken.
- De regel code wordt weer beëindigd met het “;”-teken.

Hier volgen enkele voorbeelden. In commentaar staat erbij uitgelegd wat het betekent.

```

String s;      // maak een variabele aan met naam "s".
s = "test";    // Variabele met de naam "s" krijgt de waarde "test".

int i;
i = 10; // maak variabele met naam "i" aan en geef die waarde 10

double d;
d = 1.52; //   Nieuwe variabele genaamd "d" krijgt de waarde 1,52

bool b;
b = true;    // Nieuwe variabele "b" krijgt de waarde true

String string1;
string 1 = "abc";
String string2;
string2 = string1; // Variabele met de naam "string2" krijgt
                  // de waarde van "string1", namelijk "abc"

int getalA;
getalA = 5;
int getalB;
getalB = getalA; // Variabele met de naam "getalB" krijgt
                // de waarde van "getalA", namelijk 5

double kommaGetalA;
kommaGetalA = 1.32;
double kommaGetalB;
kommaGetalB = kommaGetalA; // Variabele met de naam "kommaGetalB" krijgt
                          // de waarde van "kommaGetalA",
                          // namelijk 1.32

String s;
s = textBox1.Text;
    // Variabele met de naam "s" krijgt
    // als waarde de tekst die in de
    // TextBox genaamd "textBox1" staat.

```

Dit werkt omdat de `Text` property van de `TextBox` ook van het type `string` is.

#### 4.0.4 Variabele aanmaken en direct een waarde geven (declare en initialize)

Variabele met de naam *s* aanmaken en waarde "test" toekennen:

```
String s = "test";
```

Variabele met de naam *i* aanmaken en waarde 10 toekennen:

```
int i =10;
```

Variabele met de naam *d* aanmaken en waarde 1,52 toekennen:

```
double d = 1.52;
```

Variabele met de naam *b* aanmaken en waarde true toekennen:

```
bool b = true;
```

#### 4.0.5 Waarden omzetten naar andere typen (convert)

Merk op: het omzetten van een `int` of `double` naar een `String` lukt altijd, andersom lukt niet altijd en kan een foutmelding opleveren tijdens het uitvoeren van het programma (`crash` of `Unhandled Exception`).

Een `bool` variabele kan niet worden geconverteerd.

Zet de waarde van *i* om naar een tekst met dezelfde waarde. Het resultaat van de laatste regel is dat variabele *s* de waarde 81 krijgt.

```
int i = 81;  
String s;  
s = Convert.ToString(i);
```

Zet de waarde van *d* om naar een tekst met dezelfde waarde. Het resultaat van de laatste regel is dat variabele *s* de waarde "12.33" krijgt:

```
double d =12.33;  
String s;  
s = Convert.ToString(d);
```

Zet de waarde van *s* om naar een geheel getal (*integer*) met dezelfde waarde als dat lukt (anders krijg je een foutmelding). Het resultaat van de laatste regel is dat variabele *i* de waarde 7 krijgt:

```
int i;  
String s = "7";  
i = Convert.ToInt32(s);
```

Zet de waarde van *s* om naar een *kommagetal* met dezelfde waarde als dat lukt (anders krijg je een foutmelding). Het resultaat van de laatste regel is dat variabele *d* de waarde 12.129 krijgt:

```
double d;  
String s = "12.129";  
d = Convert.ToDouble(s);
```

## 5 Naslag basiskennis: Int en double bewerkingen (operatoren)

Onderstaande bewerkingen zijn zowel op int typen als op double typen van toepassing:

```
int k;  
k = 5 + 10;
```

Aan variabele  $k$  wordt in de laatste regel code de waarde 15 toegekend omdat het +teken de waarden 5 en 10 bij elkaar optelt.

```
int i = 2;  
int k;  
k = i + 1;
```

Aan variabele  $k$  wordt in de laatste regel code de waarde 3 toegekend omdat het +teken de waarden 2 en 1 bij elkaar optelt.

```
int i = -8;  
int k;  
k = 1 + i;
```

Aan variabele  $k$  wordt in de laatste regel code de waarde -7 toegekend omdat het +teken de waarden 1 en -8 bij elkaar optelt.

```
int i = 5;  
int j = 3;  
int k;  
k = i + j;
```

Aan variabele  $k$  wordt in de laatste regel code de waarde 8 toegekend omdat het +teken de waarden uit  $i$  en  $j$  bij elkaar op telt. Bij bovenstaande voorbeelden kan de operator (het +teken) worden vervangen door één van de volgende mogelijkheden:



Symbool	Uitwerking
+	Optellen
-	Aftrekken
*	Vermenigvuldigen
/	Delen
%	Geeft de rest na deling. Bijvoorbeeld: $7 \% 5 = 2$ $11 \% 2 = 1$ $6 \% 2 = 0$

## 6 Naslag basiskennis: Keuzestructuren in C

Als een stukje code soms wel en soms niet moet worden uitgevoerd, dan heb je een `if` of `if ... else statement` nodig. Moet een stukje code soms één keer en soms vaker worden herhaald, dan heb je een `for` of `while statement` nodig.

### 6.0.1 if-statement

Deze structuur wordt gebruikt om een stukje code uit te voeren afhankelijk van een bepaalde situatie (de *conditie* genoemd). Algemene vorm:

```
if ([conditie])
{
    [Uit te voeren code als conditie waar is]
}
```

waarbij *conditie* is een stelling die de waarde `true` (*waar*) of `false` (*niet waar*) heeft.

Voorbeelden van condities:

Conditie	Betekenis
<code>true</code>	Waar
<code>false</code>	Niet waar
<code>i &gt; 5</code>	Is i groter dan 5?
<code>i &lt; 7</code>	Is i kleiner dan 7?
<code>i &gt;= 1</code>	Is i groter of gelijk aan 1?
<code>i &lt;= 2</code>	Is i kleiner of gelijk aan 2?
<code>i == 3</code>	Is i precies gelijk aan 3?
<code>i != 3</code>	Is i ongelijk aan 3?
<code>stukjeText == "abcde"</code>	Is stukjeText precies gelijk aan "abcde"?
<code>stukjeText &lt; "abcde"</code>	Komt stukjeText eerder in het alfabet dan "abcde"?
etc.	

Verder staat *[Uit te voeren code als conditie waar is]* voor een stukje code (dit kunnen meerdere regels code zijn) dat moet worden uitgevoerd als de conditie `true` (*waar*) is. Als precies één

regel code moet worden uitgevoerd zou je ervoor kunnen kiezen de accolades openen en sluiten weg te laten, maar dit maakt de kans op bugs een stuk groter, dus dat raden we af.

### 6.0.2 if ... else ... statement

Een if statement kan uitgebreid worden met een "else" blok. Als de conditie niet "waar" oplevert dan wordt de code in het else blok uitgevoerd. Algemene vorm:

```
if ([conditie])
{
    [Uit te voeren code als conditie waar is]
}
else
{
    [Uit te voeren code als conditie niet waar is]
}
```

Merk op: of de conditie nu wel of niet waar is, altijd wordt één van de twee stukjes code uit-ge-voerd.

### 6.0.3 Voorbeelden "if ..." statement en "if ... else ..." statement

```
if (true)
{
    TextBox1.Text = "test";
}
```

Het stukje code tussen { en } wordt altijd uitgevoerd, dus de Text van de *TextBox* wordt altijd "test" gemaakt.

```
if (false)
{
    TextBox1.Text = "test";
}
```

Het stukje code tussen { en } wordt nooit uitgevoerd.

```
bool b = true;
if (b)
{
```

```

    TextBox1.Text = "test";
}

```

Als `b` de waarde `true` (= *waar*) heeft wordt de `Text` in de *TextBox* "test" gemaakt. Dit is hier nu altijd het geval omdat in dit stukje code aan variabele `b` alleen de waarde "true" wordt toegekend.

```

int i = 10;
if (i < 5)
{
    i = i + 1;
}

```

Als getal `i` kleiner dan 5 is, dan wordt bij de waarde van `i` één opgeteld, anders gebeurt er niets.

```

TextBox1.Text = "test2";
if (TextBox1.Text != "test")
{
    TextBox1.Text = "test3";
}

```

Als de tekst in de textbox niet gelijk is aan "test" (dat is hier het geval) dan wordt de tekst van de textbox veranderd in "test3".

```

if (true)
{
    TextBox1.Text = "test";
}
else
{
    TextBox1.Text = "test2";
}

```

Het stukje code tussen de eerste `{` en `}` wordt altijd uitgevoerd, dus de `Text` van de *TextBox* wordt altijd "test" gemaakt. Het stukje code tussen de tweede `{` en `}` wordt nooit uitgevoerd.

```

int i = 5;
if (i >= 10)
{
    i = i + 1;
}

```

```

}
else
{
    i = i + 5;
}

```

Als getal  $i$  groter of gelijk aan 10 is dan wordt bij getal  $i$  1 opgeteld. Dit is hier niet het geval, dus wordt bij  $i$  5 opgeteld. Resultaat:  $i$  krijgt de waarde 10.

```

int i = 5;
if (i >= 10)
{
    i = i + 1;
}
else
{
    i = i + 5;
    if (i >= 10)
    {
        i = 20;
    }
}

```

Als getal  $i$  groter of gelijk aan 10 is dan wordt bij getal  $i$  1 opgeteld. Dit is hier niet het geval, dus wordt bij  $i$  5 opgeteld. Resultaat:  $i$  krijgt de waarde 10, vervolgens wordt gecontroleerd of  $i >= 10$ , dat is nu het geval dus krijgt  $i$  uiteindelijk de waarde 20 toegekend.

## 7 Naslag basiskennis: String bewerkingen (String methods)

Hieronder worden enkele veelgebruikte String functies gedemonstreerd en kort toegelicht. Dit document is geschreven als een *naslagwerk*, het is niet specifiek geschreven om uit te leren hoe je met variabelen programmeert.

### 7.1 String's samenvoegen

Met het plus teken kunnen strings aan elkaar worden geplakt.

```
string tekst = "een tekst.";
string woorden = "Hier staat";
string s = woorden + tekst;
```

De `s` variabele krijgt hier de waarde "Hier staateen tekst." Merk op dat niet automatisch spaties worden toegevoegd.

```
string tekst = "tekst.";
string woorden = "Hier staat";
string s = woorden + " een " + tekst;
```

Met het "+"-teken kunnen strings aan elkaar worden geplakt. De "s" variabele krijgt hier de waarde "Hier staat een tekst."

### 7.2 IndexOf

De plaats van een String binnen een andere String bepalen: De *Positie* variabele krijgt de waarde 1. Merk op dat de positie van de eerste gevonden "e" in de String wordt gevonden (waarbij vanaf 0 wordt geteld):

```
string tekst = "regel tekst";  
int positie = tekst.IndexOf("e");
```

Er kan ook naar meerdere letters achter elkaar gezocht worden:

```
string tekst = "regel tekst";  
int positie = tekst.IndexOf("tek");
```

De "Positie" variabele krijgt de waarde 6. **Niet gevonden** geeft -1:

```
string tekst = "regel tekst";  
int positie = tekst.IndexOf("a");
```

De "Positie" variabele krijgt de waarde -1. De waarde -1 betekent dus: de String komt niet voor binnen de andere String.

## 7.3 Substring

Een stukje uit een string kopiëren:

```
string tekst = "regel tekst";  
string deelTekst = tekst.Substring(0, 1);
```

wat heeft als resultaat dat in deelTekst de waarde "r" komt te staan omdat van de oorspronkelijke tekst vanaf positie 0 precies 1 letter gekopieerd wordt.

Nog enkele voorbeelden met Substring.

```
"abc".Substring(0,1) // dit geeft "a" (begin vanaf karakter met index 0, neem 1 karakter)  
"abc".Substring(0,2) // geeft "ab" (begin vanaf karakter met index 0, neem 2 karakters)  
"abc".Substring(1,1) // geeft "b" (begin vanaf index 1, neem 1 karakter)
```

Goeie oefening: Typ deze eens in in een Console app, probeer te voorspellen wat er uitkomt en print de waarde met `Console.WriteLine()`, controleer of het klopt wat je dacht. Speel hiermee tot je snapt hoe het werkt. Dan kun je vast ook voorspellen wat hier uit komt:

```
Console.WriteLine("abcdef".Substring(1,1)); // hoe lang en welke letters?  
// Voer regel voor regel in (nadat je gezien hebt of je regel ervoor snapt):  
Console.WriteLine("abcdef".Substring(3,2));  
Console.WriteLine("abcdef".Substring(0,6));
```

```
// Wat zou er gebeuren bij:  
Console.WriteLine("abcdef".Substring(0,7));  
// of bij:  
Console.WriteLine("abcdef".Substring(7,1));  
// belangrijk dat je gezien hebt wat er bij die laatste 2 gebeurt!
```

```
string tekst = "regel tekst";  
string deelTekst = tekst.Substring(6, 5);
```

Deze code heeft als resultaat dat in `deelTekst` de waarde "tekst" komt te staan omdat van de oorspronkelijke tekst vanaf positie 6 precies 5 letters gekopieerd worden.

## 7.4 Length

Aantal tekens van de String bepalen. Achter `Length` hoeven geen haakjes openen en sluiten geplaatst te worden omdat het een **property** (eigenschap) van de string is en niet een **method** die je uitvoert.

```
string tekst = "regel tekst";  
int lengte = tekst.Length;
```

Deze code heeft als resultaat dat in `lengte` de waarde 11 komt te staan omdat de tekst precies elf lang is. Merk op: dit is inclusief spaties in de tekst. De **double quotes** om begin en einde van de String waarde aan te geven worden niet meegeteld.

```
string tekst = "";  
int lengte = tekst.Length;
```

Deze code heeft als resultaat dat in `lengte` de waarde 0 komt te staan omdat geen tekens in de string staan.



## 8 Naslag basiskennis: String bewerkingen (String methods)

Hieronder worden enkele veelgebruikte String functies gedemonstreerd en kort toegelicht. Dit document is geschreven als een *naslagwerk*, het is niet specifiek geschreven om uit te leren hoe je met variabelen programmeert.

### 8.1 String's samenvoegen

Met het plus teken kunnen strings aan elkaar worden geplakt.

```
string tekst = "een tekst.";
string woorden = "Hier staat";
string s = woorden + tekst;
```

De `s` variabele krijgt hier de waarde "Hier staateen tekst." Merk op dat niet automatisch spaties worden toegevoegd.

```
string tekst = "tekst.";
string woorden = "Hier staat";
string s = woorden + " een " + tekst;
```

Met het "+"-teken kunnen strings aan elkaar worden geplakt. De `s` variabele krijgt hier de waarde "Hier staat een tekst."

### 8.2 IndexOf

De plaats van een String binnen een andere String bepalen: De *Positie* variabele krijgt de waarde 1. Merk op dat de positie van de eerste gevonden "e" in de String wordt gevonden (waarbij vanaf 0 wordt geteld):

```
string tekst = "regel tekst";  
int positie = tekst.IndexOf("e");
```

Er kan ook naar meerdere letters achter elkaar gezocht worden:

```
string tekst = "regel tekst";  
int positie = tekst.IndexOf("tek");
```

De "Positie" variabele krijgt de waarde 6. **Niet gevonden** geeft -1:

```
string tekst = "regel tekst";  
int positie = tekst.IndexOf("a");
```

De "Positie" variabele krijgt de waarde -1. De waarde -1 betekent dus: de String komt niet voor binnen de andere String.

## 8.3 Substring

Een stukje uit een string kopiëren:

```
string tekst = "regel tekst";  
string deelTekst = tekst.Substring(0, 1);
```

wat heeft als resultaat dat in deelTekst de waarde "r" komt te staan omdat van de oorspronkelijke tekst vanaf positie 0 precies 1 letter gekopieerd wordt.

Nog enkele voorbeelden met Substring.

```
"abc".Substring(0,1) // dit geeft "a" (begin vanaf karakter met index 0, neem 1 karakter)  
"abc".Substring(0,2) // geeft "ab" (begin vanaf karakter met index 0, neem 2 karakters)  
"abc".Substring(1,1) // geeft "b" (begin vanaf index 1, neem 1 karakter)
```

Goeie oefening: Typ deze eens in in een Console app, probeer te voorspellen wat er uitkomt en print de waarde met `Console.WriteLine()`, controleer of het klopt wat je dacht. Speel hiermee tot je snapt hoe het werkt. Dan kun je vast ook voorspellen wat hier uit komt:

```
Console.WriteLine("abcdef".Substring(1,1)); // hoe lang en welke letters?  
// Voer regel voor regel in (nadat je gezien hebt of je regel ervoor snapt):  
Console.WriteLine("abcdef".Substring(3,2));  
Console.WriteLine("abcdef".Substring(0,6));
```

```
// Wat zou er gebeuren bij:  
Console.WriteLine("abcdef".Substring(0,7));  
// of bij:  
Console.WriteLine("abcdef".Substring(7,1));  
// belangrijk dat je gezien hebt wat er bij die laatste 2 gebeurt!
```

```
string tekst = "regel tekst";  
string deelTekst = tekst.Substring(6, 5);
```

Deze code heeft als resultaat dat in `deelTekst` de waarde "tekst" komt te staan omdat van de oorspronkelijke tekst vanaf positie 6 precies 5 letters gekopieerd worden.

## 8.4 Length

Aantal tekens van de String bepalen. Achter `Length` hoeven geen haakjes openen en sluiten geplaatst te worden omdat het een **property** (eigenschap) van de string is en niet een **method** die je uitvoert.

```
string tekst = "regel tekst";  
int lengte = tekst.Length;
```

Deze code heeft als resultaat dat in `lengte` de waarde 11 komt te staan omdat de tekst precies elf lang is. Merk op: dit is inclusief spaties in de tekst. De **double quotes** om begin en einde van de String waarde aan te geven worden niet meegeteld.

```
string tekst = "";  
int lengte = tekst.Length;
```

Deze code heeft als resultaat dat in `lengte` de waarde 0 komt te staan omdat geen tekens in de string staan.

## 9 Naslag basiskennis: While

### 9.0.0.1 while statement

Deze structuur wordt gebruikt om een stukje code uit te voeren zolang aan bepaalde voorwaarden is voldaan. Dit varieert van 0 keer de code uitvoeren tot het in de oneindigheid aantal keer uitvoeren van de code). Algemene vorm:

```
while ([conditie])  
{  
    [Uit te voeren code zolang de conditie waar is]  
}
```

Na de eerste regel staat geen ";" teken.

Eerst wordt gecontroleerd of aan een voorwaarde is voldaan, dan pas wordt eventueel code uitgevoerd.

### 9.0.0.2 do while statement

Deze structuur wordt gebruikt om een stukje code uit te voeren. Elke keer nadat het stukje code is uitgevoerd wordt gecontroleerd of nog aan bepaalde voorwaarden is voldaan, zo ja, dan wordt de code opnieuw uitgevoerd. Het aantal keer uitvoeren van de code varieert van 1 keer de code uitvoeren tot het in de oneindigheid aantal keer uitvoeren van de code. Algemene vorm:

```
do  
{  
    [Uit te voeren code zolang de conditie waar is]  
} while ([conditie]);
```

Na de laatste regel staat een ";" teken.

Eerst wordt de code één keer uitgevoerd, dan pas wordt gecontroleerd of de code eventueel vaker moet worden uitgevoerd.

### 9.0.1 Voorbeelden while en do while statement

```
int i = 0;
while(i < 10)
{
    MessageBox.Show("Test");
    i = i + 1;
}
```

Variabele *i* krijgt in het begin de waarde 0 en er wordt net zo lang doorgegaan met *MessageBoxes* weergeven totdat *i* kleiner dan 10 is. De code wordt dus doorlopen met achtereenvolgens de waarden 0, 1, 2, 3, 4, 5, 6, 7, 8 en 9. Er worden daarom 10 *Messageboxes* getoond met de tekst "Test".

```
int i = 5;
while(i > 0)
{
    MessageBox.Show("Test");
    i = i - 1;
}
```

Variabele *i* krijgt in het begin de waarde 5 en er wordt direct gestopt als *i* de waarde 0 krijgt toegekend. De code wordt dus doorlopen met de waarden 5, 4, 3, 2, 1. Er worden daarom 5 *Messageboxes* getoond met de tekst "Test".

```
int i = 10;
do
{
    MessageBox.Show("Test");
    i = i + 1;
}
while (i < 5);
```

Variabele *i* krijgt in het begin de waarde 10, de code wordt uitgevoerd, en vervolgens wordt net zo lang doorgegaan met *Messageboxes* weergeven totdat *i* kleiner dan 5 is. De code wordt dus doorlopen met de waarde 10. Er wordt daarom 1 *Messagebox* getoond met de tekst "Test".

# 10 Naslag basiskennis: For

## 10.0.1 for statement

Deze structuur wordt gebruikt om een stukje code een vooraf bekend aantal keer uit te laten voeren. Algemene vorm:

```
for([teller variabele aanmaken]; [herhalingsconditie]; [teller variabele aanpassen])
{
    [herhaaldelijk uit te voeren code]
}
```

waarbij *[teller variabele aanmaken]* een `variabele` met zelf te kiezen variabelenaam wordt aangemaakt en van een waarde voorzien. Veel gebruikte variabele namen voor een for statement zijn "i", "j", "k" omdat deze een hele korte naam hebben, dat leest in veel gevallen prettig. Ook "index", "count" of "teller" worden vaak gebruikt. Het type variabele is meestal `int`. De waarde waarmee de teller wordt gevuld is afhankelijk van wat je aan het programmeren bent. In veel gevallen heeft deze de waarde 0. Voorbeelden:

```
int i = 0
```

```
int j = 100
```

Dan *[herhalingsconditie]*: deze uit te voeren code wordt net zo lang herhaald als uit de voorwaarde de waarde `true` komt. Hierin verwijst je naar de *teller* variabele. Voorbeelden:

```
i < 10
j > 0
```

*[teller variabele aanpassen]* Het verhogen of verlagen van de teller. Vaak wordt deze met 1 verhoogd of verlaagd, soms in grotere stappen (bijv. 10). Voorbeelden:

```
i = i + 1
j = j - 10
```

[herhaaldelijk uit te voeren code] Het stukje code (dit kunnen meerdere regels code zijn) dat moeten worden uitgevoerd zolang de herhalingsconditie "true" (waar) is. Ieder `for`statement is om te zetten naar een `while` statement dat hetzelfde doet, en andersom.

## 10.0.2 Voorbeelden for statement

```
for(int i =0 ; i < 10 ; i = i + 1)
{
    MessageBox.Show("Test");
}
```

Variabele `i` krijgt in het begin de waarde 0 en er wordt direct gestopt als `i` de waarde 10 krijgt toegekend. De code wordt dus doorlopen met de waarden 0,1,2,3,4,5,6,7,8 en 9. Er worden daarom 10 messagebox-en getoond met de tekst "Test".

```
for(int i =5;i > 0; i = i - 1)
{
    MessageBox.Show("Test");
}
```

Variabele `i` krijgt in het begin de waarde 5 en er wordt direct gestopt als `i` de waarde 0 krijgt toegekend. De code wordt dus doorlopen met de waarden 5,4,3,2,1. Er worden daarom 5 messagebox-en getoond met de tekst "Test".

```
for(int i =0;i < 10;++i)
{
    MessageBox.Show("Test");
}
```

Hetzelfde resultaat als het eerste voorbeeld, maar dan in een verkorte schrijfwijze:

```
i = i + 1;
```

wordt van oudsher ook wel geschreven als

```
i++;
```

of

```
++i;
```

Hetzelfde resultaat als het tweede voorbeeld, maar dan in een verkorte schrijf-wijze:

```
for(int i =5;i > 0; --i)
{
    MessageBox.Show("Test");
}
```

```
i=i-1;
```

wordt van oudsher ook wel geschreven als

```
i--;
```

of

```
--i;
```

De code

```
for(int i =0;i < 10; ++i)
{
    MessageBox.Show("Test "+i);
}
```

heeft als resultaat dat *MessageBoxes* verschijnen met achtereenvolgens:

```
"Test 0"
"Test 1"
"Test 2"
"Test 3"
"Test 4"
"Test 5"
"Test 6"
"Test 7"
"Test 8"
"Test 9"
```

De code

```
for(int i =5;i > 0; i = i - 2)
{
```



```
    MessageBox.Show("Test "+i);  
}
```

laat messagebox-en verschijnen met achtereenvolgens:

```
"Test 5"  
"Test 3"  
"Test 1"
```

en tot slot geeft

```
for(int y =0;y < 2; ++y)  
{  
    for(int x =0;x < 3; ++x)  
    {  
        MessageBox.Show("(" +x+", "+y+"");  
    }  
}
```

als resultaat *MessageBoxes* verschijnen met:

```
"(0,0)"  
"(1,0)"  
"(2,0)"  
"(0,1)"  
"(1,1)"  
"(2,1)"
```

# 11 Explanation: Methods in C

## 11.0.1 Algemene structuur methoden

Een methode is een stukje code dat vanuit een ander stukje code is aan te roepen. Als een methode een waarde terug geeft welke gebruikt gaat worden in het stukje code waar vanuit deze is aangeroepen spreek je over een methode welke "een waarde teruggeeft". Ook kunnen aan een methode één of meer waarden worden meegegeven. Dit worden argumenten genoemd.

## 11.0.2 Belangrijkste voordelen van het gebruik van methoden:

1. Overzichtelijkheid: Als alle code in één enkele event handler (bijv. *ButtonX\_Click*) wordt geplaatst wordt je code al snel erg overzichtelijk.
2. Werk verdelen: Als je voordat je gaat programmeren het programmeerwerk wilt verdelen kun je de te maken code opdelen in methoden en deze met verschillende programmeurs tegelijkertijd programmeren.
3. Onderhoudbaarheid & herbruikbaarheid: Als je op verschillende plaatsen in je programma hetzelfde stuk code vaker uit wilt voeren kun je vanaf de verschillende plaatsen een methode aanroepen die je maar één keer hoeft te programmeren. Dat scheelt code en is gemakkelijker te onderhouden dan dat je code verschillende keren in je programma kopieert en plakt.

Een methode heeft de volgende structuur:

```
private [returnType] [methodeNaam]([parameters])
{
    ...
    [return returnWaarde]
}
```

**private**

geeft aan dat de **methode** alleen binnen het huidige bestand (lees: **Form1**) kan worden aangeroepen. Wat dit precies inhoudt is voor dit vak niet interessant, hier wordt later op teruggekomen.

[returnType]

Het type van de waarde die de methode terug geeft. Als de methode geen waarde terug geeft, is dit `void` (*niets*). Voorbeelden: `int`, `double`, `bool`, `string`, `void`.

[methodeNaam]

Zelf gekozen naam voor de methode, te vergelijken met een zelf gekozen naam voor een variabele. Voorbeelden: *TelOp*, *ToonMelding*, *Methode1*, *Abc*.

[parameters]

Optioneel onderdeel. Hiermee wordt opgegeven welk(e) type(n) waarde(n) moet worden meegegeven aan de methode en onder welke naam deze waarde binnen de method kunnen worden gebruikt. Meerdere parameters worden gescheiden met een `,`-teken. Voorbeelden: `int deler, int getalA, int getalB, bool isIngelogd, double eenKommaGetal`.

[return returnWaarde]

Met `return` gevolgd door een waarde die aan het *[returnType]* voldoet wordt een waarde teruggegeven vanuit de methode aan het stukje code dat de methode heeft aangeroepen. Als *[returnType]* `void` is hoeft er geen `return` worden gebruikt. Voorbeelden: `return uitkomst;`, `return 10;;`, `return mijnTekst;;`, `return "Hallo"+" daar!";`, `return getal > 10;`

### 11.0.3 Voorbeelden Methoden

Een aantal voorbeeldmethoden:

```
private int AddTwoNumbers(int number1, int number2)
{
    int som;
    som = number1 + number2;
    return som;
}

private int SquareANumber(int number)
{
    return number * number;
}
```

Bovenstaande methoden zijn als volgt aan te roepen:

```
int sum = AddTwoNumbers(8765, 287);
```

of:

```
int kwadraat = SquareANumber(63);
```

of, beiden:

```
int total = SquareANumber(AddTwoNumbers(1, 2));
```

#### 11.0.3.1 Tekst en uitleg (engelstalig) over methoden en parameters

Zie bijvoorbeeld [C-sharpcorner over methods](#) voor meer uitleg.

## 12 Explanation - Array & Lists

C# kent, net als de meeste andere programmeertalen, naast klassen nog meer complexe datatypes. Twee voorbeelden daarvan zijn **array's** en **lists**. Beide types zijn bedoeld om meerdere waarden van een type te kunnen bewaren. Bijvoorbeeld een verzameling getallen, een lijst van strings of een aantal boten. Soms weet je van te voren niet hoeveel waarden je precies moet onthouden en soms zijn het er gewoon te veel om allemaal op te slaan als losse variabelen. Dit zijn allemaal goede redenen om collecties te gebruiken.

We kijken eerst naar array's. Dit is het basistype om meerdere waarden van een en hetzelfde type te kunnen bewaren. Dit datatype werkt ook in de niet object-georiënteerde programmeertalen, zoals bijvoorbeeld C.

### 12.1 Array's

Bekijk het volgende voorbeeldje:

```
double[] getallen = new double[10];
getallen[0] = 2.5;
MessageBox.Show("Getal nummer 1 = " + getallen[0]);
```

Hier wordt een array van doubles aangemaakt. Om precies te zien een array van precies 10 doubles. Op de tweede regel code wordt het eerste getal in de rij gelijk gemaakt aan 2,5. Met de [blokhaken] kun je elementen een nieuwe waarde geven. Ook kun je getallen opvragen met de [blokhaken]. Dit zie je op de regel daaronder, in de messagebox. Simpel toch? Wat voor getallenreeks wordt er in onderstaand stukje code dan opgeslagen in de array?

```
int[] getallen = new int[10];
for (int i = 0; i < getallen.Length; i++)
{
    getallen[i] = (i + 1) * 5;
}
```

Zoals je ziet heeft een array altijd een vaste grootte. In beide voorbeelden hierboven wordt plek gereserveerd voor 10 getallen (doubles of integers). Met de Length eigenschap van een array (ook te zien in bovenstaande stukje code) kun je opvragen hoeveel elementen je maximaal in

de array kunt opslaan. Uiteraard kun je ook andere zaken dan integers en doubles opslaan met een array. Zo kun je bijvoorbeeld ook een array van strings of je eigen objecten aanmaken. Probeer eens uit! Een array kan wel altijd maar 1 soort (type) variabele opslaan.

Net als bij klassen moet je een array aanmaken met `new`. Wanneer je dat niet doet krijg je daar van Visual Studio een melding over.

**Let op:** We beginnen te tellen bij **0**. Het eerste element in een array staat op positie 0. Het getal op *index* 5 is het zesde getal in de reeks.

**Breïnbrekers:** Wist je dat een string eigenlijk een `char[]` array is? En wist je dat je ook een array van array's (van array's (van array's...)) kunt maken (2D- en 3D-array's)? Leuke uitdagingen om mee te spelen!

## 12.2 Lists

Het nadeel van een array is dat je van te voren moet specificeren hoeveel plekken je wilt reserveren voor je data. Echter, niet altijd is dit van te voren bekend. Wanneer je een gebruiker zelf elementen laat toevoegen in het systeem weet je niet hoeveel er toegevoegd zullen worden. Daarom is de List bedacht: de meegroeiende array. Wanneer je meer plek nodig hebt zal de List automatisch groeien in maximale grootte.

```
int[] getallen = new int[10];
for (int i = 0; i < 10; i++)
{
    getallen[i] = i * 10;
}

getallen[10] = 10 * 10;
```

In bovenstaand stukje code worden een array van maximaal 10 integers aangemaakt. Vervolgens worden de tafel van 10 getal voor getal aan de array toegevoegd. Na de for-loop wordt het laatste element aan de array toegevoegd, 10 x 10. Echter, deze zal niet meer passen. Er kunnen maximaal 10 integers in de array en 0 x 10 tot en met 9 x 10 zijn al toegevoegd. Dat zijn er dus al 10 in totaal. Het getal 100 zal niet meer passen en geeft dan een *IndexOutOfRangeException* foutmelding. In onderstaand voorbeeld wordt een alternatief met een List gegeven.

```
List<int> getallen = new List<int>();
for (int i = 0; i < 10; i++)
{
    getallen.Add(i * 10);
}
```

```
}

getallen.Add(10 * 10);
```

Dit stuk code zal niet meer de aangeven dat er geen plek meer is voor het getal 100. De List groeit mee met de vraag. Als er meer plek nodig is zal de List dus plek vrijmaken. Een array kan dit niet uit zichzelf!

Hieronder zie je een voorbeeld met een List van strings. In plaats van de for-loop wordt de foreach gebruik. Dit is een nieuw soort herhalingsstructuur waarin er om de loop al een element uit de collectie wordt gehaald om er mee verder te werken. Dit scheelt dus weer code! Dit werkt overigens ook met array's.

```
string s = "";
List<string> woorden = new List<string>();
woorden.Add("Hallo");
woorden.Add("allemaal!");
foreach (var woord in woorden)
{
    s = s + woord + " ";
}
MessageBox.Show(s);
```

Waarschijnlijk zie je het al, maar de tekst “Hallo allemaal!” zal verschijnen in een MessageBox. Bij elke iteratie (elke keer dat C# door de collectie loopt) pakt hij het volgende element uit de verzameling. Deze stopt hij in de variabele woord. Deze variable bevat altijd het element wat op dat moment actueel is in de herhaling. Hij pakt dus eest element met index 0, daarna die met index 1, index 2, en zo voorts. Je hoeft dus niet zelf meer de index bij te houden en ook het element wordt automatisch voor je uit de collectie gepakt.

### 12.2.1 List methoden

Het leuke van alles opslaan in een List is dat je er gemakkelijk elementen in terug kunt vinden. Daar gebruiken we twee methodes voor, de Contains en IndexOf. Met de Contains methode kun je controleren of een element voorkomt in de List. Deze geeft dus true of false terug. De IndexOf methode geeft de index terug van het element dat je zoekt. Als het element niet is gevonden zal -1 terug worden gegeven.

```
List<string> talen = new List<string>();
talen.Add("Java");
...
```

```

if (talen.Contains("C#"))
{
    MessageBox.Show("C# komt voor in de lijst!");
}

```

In bovenstaand voorbeeld wordt het gebruik van de Contains methode laten zien. Aan de talen List worden verschillende strings toegevoegd. Allereerst wordt de string “Java” aan de List toegevoegd. Op de drie punten kunnen nog andere talen zijn toegevoegd.

Als je weet dat een bepaald element in de List voorkomt kun je met de IndexOf methode kijken op welke index het element staat. Bekijk onderstaand voorbeeld.

```

List<string> talen = new List<string>();
talen.Add("Java");
...

int index = talen.IndexOf("C#");
MessageBox.Show("C# staat op index " + index);

```

Wanneer C# niet in de lijst zou staan zou de IndexOf methode aangeven dat de index gelijk is aan -1. Dat is een code die wordt gebruikt wanneer het element waarvan de index van wordt opgevraagd niet wordt gevonden in de List. Dat kun je natuurlijk ook voorkomen door eerst met Contains te kijken of het element voorkomt in de List. Probeer het eens zelf uit en kijk of je de twee kunt combineren!

**Tip:** De Contains en IndexOf methodes werken ook voor strings!

## 12.3 Voorbeelden

### 12.3.1 List en foreach

```

public List<int> GeefTafelVan(int tafelVan)
{
    List<int> getallen = new List<int>();
    for (int i = 0; i <= 10; i++)
    {
        getallen.Add(i * tafelVan);
    }
    return getallen;
}

```



In de code hierboven zie je een methode die de tafel van een getal teruggeeft, in de vorm van een List van integers. Je zou deze als volgt kunnen gebruiken:

```
List<int> tafelVanVijf = GeefTafelVan(5);
foreach (var getal in tafelVanVijf)
{
    Console.Out.WriteLine(getal);
}
```

### 12.3.2 Array en for-loop

```
char[] woord = { 'H', 'a', 'l', 'l', 'o', '!' };
for (int i = 0; i < woord.Length; i++)
{
    Console.Out.Write(woord[i]);
}

Console.Out.Write(Environment.NewLine);
```

Hierboven zie je een stukje code om een array van characters op te slaan. Met een for-loop kun je de gegevens ophalen en afdrucken naar de console. Probeer het zelf eens uit!

### 12.3.3 List en Contains

```
List<double> inworp = new List<double>();
inworp.Add(.5);
inworp.Add(2.0);
inworp.Add(.5);

if (inworp.Contains(.5))
{
    MessageBox.Show("Er is minstens 1 " + "muntstuk van vijftig cent ingeworpen!");
}
```

Hierboven zie je een stukje code met een lijst van prijzen, opgeslagen als lijst van doubles. Wanneer een gebruiker een muntstuk van 50 cent heeft ingeworpen zal er een MessageBox worden weergegeven.

#### 12.3.4 List en IndexOf

```
List<double> inworp = new List<double>();  
if (inworp.IndexOf(.5) == -1)  
{  
    MessageBox.Show("Er is geen muntstuk " + "van vijftig cent ingeworpen!");  
}
```

Bijna hetzelfde voorbeeld als hierboven. Hier wordt de IndexOf methode gebruikt om de index van het eerste vijftig cent muntstuk in de lijst op te vragen. Als deze -1 teruggeeft is er geen muntstuk van vijftig cent gevonden.

## **Part III**

# **Training assignments**

# 13 Training Goeroe-calc: variabelen, bewerkingen en conversies: Een Console app(lication)

(Concepten: int, double, bewerkingen en conversies)

## 13.1 Voorbereiding

Ken je het verschil tussen een int, double en string?

## 13.2 Inleiding

We gaan een werkende calculator maken. Het is een minimalistische calculator, maar hij kan zaken voor je uitrekenen die je zelf niet kunt. Hoeveel is 655 maal 23623? De meeste mensen gebruiken daar liever een programma voor. Jij kunt dat programma maken.

## 13.3 Opdracht

We vragen de gebruiker om een getal in te typen (en op enter te drukken), nog een getal (weer enter), dan vertellen we de gebruiker hoeveel je krijgt als je de getallen optelt, maar ook hoeveel je krijgt als je de getallen vermenigvuldigt.

### 13.3.1 Enkele stappen uitgelegd...

- Maak een Console app aan en geef die voor de onderhoudbaarheid een duidelijke naam, zoals bijvoorbeeld 'Calculator'.
- Met het commando `Console.WriteLine("bloemkool")` kun je de letterlijke tekst tussen de dubbele quotes (") aan gebruiker laten zien. Vraag de gebruiker om een getal in te typen.

- Na ieder commando (ook wel **statement** genaamd) wil C# graag een punt-komma ; zien.
- Een `Console.ReadLine()` wacht totdat de gebruiker iets intypt en daarna op **enter** drukt. Door een variabele van type `String` aan te maken met bijvoorbeeld de naam `textTypedByUser` en dan een toekenning (**assignment**) te gebruiken (in C# te herkennen aan het `=`-teken) wordt de tekst die door de user werd ingetypt in die variabele opgeslagen.
- De eerste keer dat je een nieuwe variabele gebruikt noemen we de **declaratie**: je moet dan het 'type' er voor zetten. Daarna moet dat juist niet meer: zo weet C# wanneer je een nieuwe variabele aan maakt en wanneer niet.
- Let op: als je een tekst op het scherm zet gebruik je de taal (NL?) waarmee het programma met de gebruiker interacteert, maar voor variabelennamen wordt bijna altijd Engels gebruikt.
- Een `String` is een keten van karakters. Om ermee te gaan rekenen moeten we C# vertellen dat we de waarde willen omzetten (converteren) naar een `int` (voor een geheel getal, ook wel **integer** genoemd). Dat doen we door de methode `Convert.ToInt32()` aan te roepen en tussen de haakjes de waarde mee te geven die we willen converteren. Het resultaat (dat we aan een variabele van type `int` kunnen assignen) is een integer getal: `int numberTypedByUser = Convert.ToInt32(textTypedByUser);`.
- Om de waarden van twee bestaande integer variabelen `a` en `b` op te tellen kunnen we een nieuwe integer `int c`; maken en daar de waarde van `a + b` aan assignen.
- Tot slot moeten we de **integer** waarde weer terug converteren naar een `String` om het op het scherm te zetten.

Hier C#-code die de boven besproken concepten laat zien:

```
Console.WriteLine("Beste gebruiker,");
Console.WriteLine("Typ een getal svp (en druk op enter)");

String textTypedByUser = Console.ReadLine();
Console.WriteLine("U hebt ingetypt: " + textTypedByUser);

int numberTypedByUser = Convert.ToInt32(textTypedByUser);

int a = 42;
int b = 365;
int c = a + b;

String antwoord = Convert.ToString(c);
Console.WriteLine(antwoord);
```

Het is mogelijk dit korter op te schrijven, maar hou dan in de gaten of het leesbaar blijft!

Met deze kennis is het mogelijk de eerdergenoemde calculator te maken: De gebruiker kan integers intypen. Je kunt er bijvoorbeeld voor kiezen de gebruiker te vertellen zowel wat de som als het product van de getallen is.

Is het gelukt? Dan heb je nu zelf een programma geschreven dat meer kan dan jijzelf (binnen een milliseconde de getallen 7225 en 5588 met elkaar vermenigvuldigen bijvoorbeeld) en heb je de eerste stap gezet om een ervaren software engineer te worden.

Zit je vast? Stel een vraag aan je buurman of -vrouw! Als jullie er samen niet uitkomen vraag je het aan je docent. In het begin kan dit programmeren best moeilijk zijn.

Als de calculator werkt dan kun je nog de laatste 2 requirements programmeren. Dat zijn deze: + De uitkomst wordt getoond als “Uitkomst: 123”, dus met de tekst “Uitkomst:” voor de daadwerkelijke uitkomst. + De calculator moet werken met gebroken getallen. Dus 3.14 maal 2.0 moet 6.28 opleveren. Je moet dan in plaats van `int` `double` gebruiken.

Misschien had je die al, dan ben je niet alleen goed in programmeren, maar heb je de analyse-fase ook eervol doorlopen.

Bespreek opdrachten regelmatig met je docent en voer dan feedback in in Feedpulse.

## 13.4 Extra's

Om het echt goed te leren is het goed om uitbreidingen op de opdrachten te maken, en hier ook feedback op te vragen. Dat kan leiden tot een hogere beoordeling (let op: je moet het wel zelf kunnen programmeren, code-kopie van internet is niet voldoende). Enkele mogelijke uitbreidingen:

- Breid de calculator uit met een functie voor worteltrekken (hint: square root in het Engels, dat helpt bij het gebruik van een zoekmachine).
- Als de uitkomst onder de nul is laat je dit duidelijk zien, bijvoorbeeld door er achter de tekst ‘LET OP: is negatief’ te zetten.
- Breidt de calculator uit met een functie voor delen. Zorg ervoor dat delen door 0 netjes wordt afgevangen en laat een nette foutmelding zien.

# 14 Training Goeroe-calc: variabelen, bewerkingen en conversies

(Concepten: int, double, bewerkingen en conversies)

## 14.1 Voorbereiding

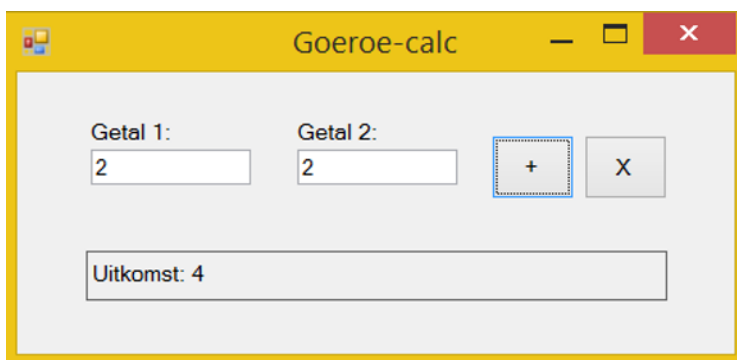
Ken je het verschil tussen een int, double en string?

## 14.2 Inleiding

We gaan een werkende calculator maken. Het is een minimalistische calculator, maar hij kan zaken voor je uitrekenen die je zelf niet kunt. Hoeveel is 655 maal 23623? De mens gebruikt daar liever een programma voor. Jij kunt dat programma maken.

## 14.3 Opdracht

Maak de user interface van de calculator waarbij je minimaal twee Textboxen, een Label en twee Buttons (een voor optellen en een voor vermenigvuldigen) hebt.



De gebruiker kan gehele getallen (integers) invullen in de tekstboxen. Programmeer nu de functionaliteit van de calculator conform de volgende specificatie:

- Als de gebruiker op de knop Plus klikt dan worden de twee ingevoerde getallen bij elkaar opgeteld en in een label getoond als uitkomst.
- Als de gebruiker op de knop Vermenigvuldig klikt dan worden de twee ingevoerde getallen met elkaar vermenigvuldigd en in een label getoond als uitkomst.

Is het gelukt? Dan heb je nu zelf een programma geschreven dat meer kan dan jijzelf (binnen een milliseconde de getallen 7225 en 5588 met elkaar vermenigvuldigen bijvoorbeeld) en heb je de eerste stap gezet om een ervaren software engineer te worden.

Zit je vast? Stel een vraag aan je buurman of -vrouw! Als jullie er samen niet uitkomen vraag je het aan je docent. In het begin kan dit programmeren best moeilijk zijn.

Als de calculator werkt dan kun je nog de laatste 2 requirements programmeren. Dat zijn deze:  
 + De uitkomst wordt getoond als “Uitkomst: 123” in een label dus met de tekst “Uitkomst: “: voor de daadwerkelijke uitkomst. + De calculator moet werken met gebroken getallen. Dus 3.14 maal 2.0 moet 6.28 opleveren.

Misschien had je die al, dan ben je niet alleen goed in programmeren, maar heb je de analyse-fase ook eervol doorlopen.

Bespreek deze opdracht met je docent en voer feedback van je docent in in Feedpulse.

## 14.4 Extra's

Wil je graag lachende smiley's in Feedpulse? Verzin dan uitbreidingen op de opdrachten of eigen opdrachten en vraag feedback (let op: je moet deze extra's zelf kunnen programmeren voor een hogere beoordeling, code-kopie van internet is niet voldoende). Enkele mogelijke uitbreidingen:

- Breid de calculator uit met een functie voor worteltrekken.
- Maak de tekst van het label rood (rode letters) als de uitkomst onder de nul is.
- Breidt de calculator uit met een functie voor delen. Zorg ervoor dat delen door 0 netjes wordt afgevangen en laat een nette foutmelding zien.



## 15 Software baas

Maak deze applicatie.

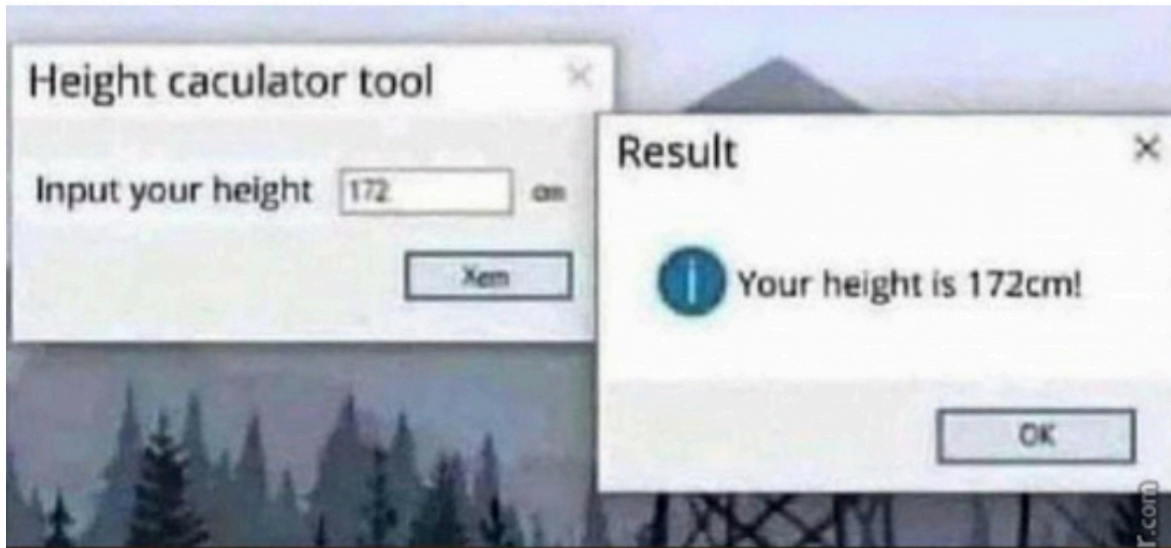


Figure 15.1: Aangeleverd ontwerp user interface

# 16 Euro-Dollar Converter

## 16.1 Doelen

- Omgaan met verschillende typen (int, double, string).
- Gebruik van variabelen.
- Converteren van typen.

## 16.2 Inleiding

Reisbureau “FLUX vakanties” organiseert rondreizen in landen van alle continenten. Veel van de geboekte reizen gaan naar landen waar niet met euro’s maar met dollars betaald moet worden. Om de reisvoorbereidingen voor klanten wat te vergemakkelijken wil FLUX vakanties daarom een applicatie laten ontwikkelen om euro’s gemakkelijk naar dollars om te kunnen rekenen en andersom.

## 16.3 Bronnen

Voorbeelden over het werken met variabelen in C#: zie ‘Reference: variable’.

## 16.4 Opdracht

De user interface is al ontworpen:

- Bouw deze user interface.
- Zorg ervoor dat de koers per cent wordt opgehoogd/verlaagd als op de pijltjes omhoog/omlaag wordt geklikt (dit is een Property van de NumericUpDown), en stel de standaardwaarde in op 2,00 (of 2.00?). Controleer of het werkt door het project uit te voeren.
- Programmeer de functionaliteit achter de Button “<” en Button “>”:
  - Als op “>” wordt geklikt worden de Euro’s in de linker TextBox omgerekend naar dollars, en worden deze dollars in de rechter TextBox getoond.

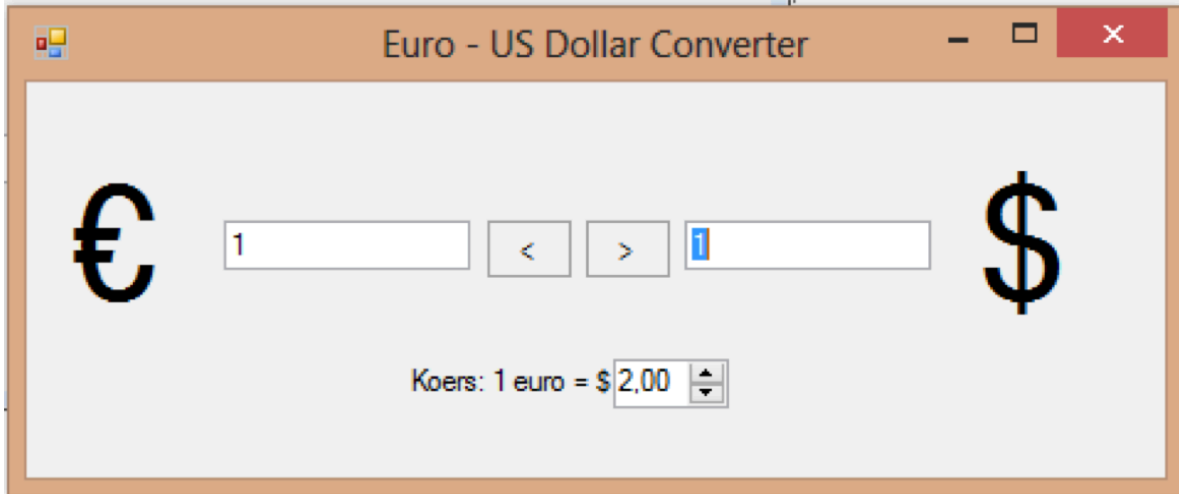


Figure 16.1: Aangeleverd ontwerp user interface

- Als op “<” wordt geklikt worden de dollars rechter TextBox omgerekend naar Euro’s, en worden deze Euro’s in de linker TextBox getoond.
- In beide gevallen wordt de ingestelde koers gebruikt.

Het is niet mogelijk om met variabelen van een tekst type te rekenen. Met variabelen van

## 16.5 Uitbreiding (niveau 3 van 5)

Voeg een controle toe op de invoer van bedragen: als de invoer geldig is, moet de berekening worden uitgevoerd. Is de invoer niet geldig, dan wordt een duidelijke foutmelding getoond en wordt er niets berekend.

## 16.6 Uitbreiding (niveau 4 van 5)

Voeg de mogelijkheid toe om op een gebruikersvriendelijke manier te kunnen kiezen tussen twee vreemde valuta’s: Dollar of Yen. Er wordt wel altijd omgerekend van of naar Euro’s. De user interface moet hier uiteraard op aangepast worden.

## 16.7 Checklist

Als je de opdracht op de juiste manier hebt uitgevoerd heb je voldaan aan onderstaande punten:  
+ Vul bij koers 1,50 in, vul bij euro’s 3 in en druk op de “>” knop. Het aantal dollars zou nu

4,5 moeten worden. + Vul bij koers 1,50 in, vul bij dollars 2 in en druk op de “<” knop. Het aantal euros zou nu 1,333... moeten worden. + Controleer of de koers met centen kan worden verhoogd en verlaagd.

Is alles in orde? Feedback vragen maar...

## 16.8 Versies

- Mei 2015 Marcel Veldhuijzen (KAL, standaard uitwerking verwijderd)
- 2014-01-10 Bas Michielsen (Template)
- 2014-01-09 Lindy Hutz (VS 2013)
- 2012 Sjaak Verwaaijen
- 2011 Tom Broumels

# 17 Training Computer-rekenen

Als de computer dan toch zo'n krachtige calculator is, laten we hem dan ook als zodanig gebruiken...

## 17.1 Recht-toe-recht-aan

- $100 \times 0,7 - 25 \times 0,6 = \dots$
- $18.0 / 5.0 = \dots$
- $100 / 30 = \dots$

Schrijf een programma dat deze antwoorden berekent.

Als je het op een rekenmachine (bijvoorbeeld de calculator-app op je laptop) berekent: komt er dan dezelfde uitkomst uit? Zo nee, probeer erachter te komen waarom dat zo is.

## 17.2 Al wandelend...

Patrick maakt een wandeling van A naar B. De afstand is 20 km.

Hij begint in A en als hij  $1/4$  deel van de afstand heeft afgelegd, rust hij een poosje (rustpunt 1). Na korte tijd hervat hij de wandeling en loopt  $1/4$  deel van het resterende gedeelte, Daar rust hij weer even uit (rustpunt 2). Op dezelfde manier wandelt hij verder. Elke keer rust hij een poosje als hij weer  $1/4$  deel van het resterende gedeelte heeft gelopen.

De afstand van rustpunt 5 tot het eindpunt B is meter. (Vul een geheel getal in, indien nodig afronden.)

Na rustpunt 5 maakt hij geen stops meer, omdat hij anders nooit het eindpunt zal halen."

(bron: vraag 28aug2020, betterrekenen)

Hint: Teken allereerst een plaatje. Pauzepunt 1 ligt op  $3/4$  van 20km van punt B. Pauzepunt 2 ligt op  $3/4$  van het getal op de vorige regel... En zo voort...

## 17.3 Ik een beetje meer dan jij...

Je verdeelt een geldbedrag over vier personen (A, B, C en D). Elke persoon krijgt een ander bedrag: A krijgt 20% meer dan het gemiddelde van de vier personen. Voor de duidelijkheid: uiteindelijk krijgen de vier personen samen het hele bedrag. A krijgt daarvan een kwart + 20% van zo'n kwart. Daarna verlaat persoon A de kamer.

Er blijven drie personen over, met wie een soortgelijke verdeling plaatsvindt: B krijgt 20% meer dan het gemiddelde van de 3 overgebleven personen en verlaat de kamer. C krijgt 20% meer dan het gemiddelde van de 2 overgebleven personen. D ontvangt de overgebleven 168 euro.

Het oorspronkelijke, te verdelen bedrag is ... euro.

Schrijf een computerprogramma dat dit getal berekent.

Hieronder een deel van de oplossing, maar probeer het eerst zelf! (Afkomstig van [beterreken.nl](http://beterreken.nl), 12aug2020; een som van Henk van Huffelen)

## 17.4 Ik een beetje meer dan jij... aanzet tot een oplossing

Als je het in C# doet: Maak een Console app aan en vul de Main-methode als volgt in:

```
static void Main(string[] args)
{
    // Ik een beetje meer dan jij...
    // ----- een oplossing -----

    // Je verdeelt een geldbedrag over vier personen(A, B, C en D). Elke persoon krijgt ee
    // A krijgt 20 % meer dan het gemiddelde van de vier personen.
    // Voor de duidelijkheid: uiteindelijk krijgen de vier personen samen het hele bedrag.
    // A krijgt daarvan een kwart + 20 % van zo'n kwart. Daarna verlaat persoon A de kamer

    // Er blijven drie personen over, met wie een soortgelijke verdeling plaatsvindt:
    // B krijgt 20 % meer dan het gemiddelde van de 3 overgebleven personen en verlaat de

    // Het oorspronkelijke, te verdelen bedrag is ... euro.

    // Schrijf een computerprogramma dat dit getal berekent.

    // ----- een oplossing -----
    // "Achteruit terugredeneren: "
```

```

// Het bedrag dat persoon D krijgt noemen we 'd', c is het bedrag dat persoon C, krijg
// Voor bedragen raden we het type 'decimal' aan:
decimal d = 168;
    Console.WriteLine("d: {0}",d);

// c kreeg 20% meer dan het gemiddelde van de laatste 2 personen: c en d.
// Het gemiddelde van c en d is (c + d) / 2
// c is 20% meer ofwel 1.2 * dat gemiddelde:
// Er geldt: c == 1.2 * ((c + d) / 2)
// ofwel: c == 0.6 * c + 0.6 * d
// ofwel: 0.4 * c == 0.6 * d
// dus c = 0,6 / 0,4 * d
decimal c = 3 / 2 * 168;
    Console.WriteLine("c: {0}", c);

// b is 20% meer dan (b + c + d) / 3,
// ofwel: b == 1.2 * ((b + c + d) / 3),
// ofwel:

maak dit programma zelf verder af!


decimal b = 0; // pas dit aan...
Console.WriteLine("b: {0}", b);

decimal a = 0; // pas dit aan...
Console.WriteLine("a: {0}", a);
Console.WriteLine("Hello World!");
}

```

Maak het programma hierna af.

## 17.5 Een 1-2-3-tje

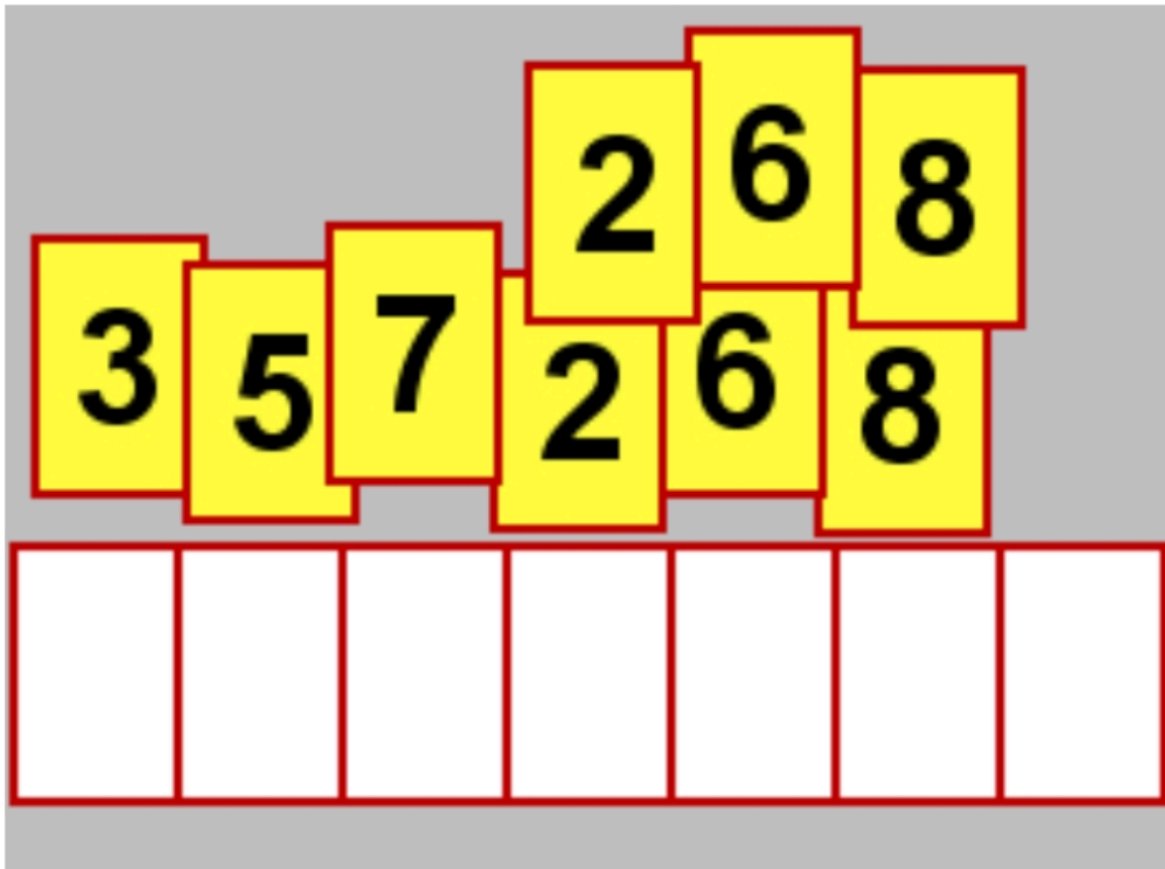
In de winkel liggen 3 artikelen naast elkaar met elk een verschillende prijs in hele euro's. De prijzen zijn genoteerd zonder komma, dus bijvoorbeeld: "32 euro".

De prijs van het duurste artikel is een bedrag van 3 cijfers. Als je het eerste cijfer van dat bedrag weglaat, heb je de prijs van het tweede artikel. Laat je daarvan weer het eerste cijfer weg, dan heb je de prijs van het derde artikel.

De artikelen kosten samen 589 euro. Het middelste artikel is minder dan 30 keer zo duur als het goedkoopste artikel. Het verschil in prijs tussen het duurste en goedkoopste artikel is ... euro.

Schrijf een computerprogramma dat dit getal berekent. (of kun je het zelfs zonder computerprogramma?)

## 17.6 Hoeveel palindromen kan ik maken met ...?



Een **palindroom** is een woord of getal dat achterstevoren geschreven hetzelfde is. Bijvoorbeeld het woord LEPEL of het getal 121.

Je hebt de beschikking over een maal de cijfers 3, 5 en 7 en tweemaal over de cijfers 2, 6 en 8 (zie plaatje).

Hiermee kun je ... verschillende palindromen van 7 cijfers samenstellen.

(afkomstig van [beterrekenen.nl](http://beterrekenen.nl), 10aug2020, een som van Henk van Huffelen).



### 17.6.1 Bronnen

Een aantal van deze opgaven komen van [www.betterrekenen.nl](http://www.betterrekenen.nl) (ook als mobile app): Elke dag een paar minuten aan uitdagingen, soms makkelijk, soms moeilijker, some recht-toe-recht-aan, soms echt nodig om een plaatje te tekenen of een laptop te hulp te roepen... hersengym dus... een aanrader!

### 17.7 Spoilers...

Enkele antwoorden: + A1 wandelend...: 4746 meter

## 18 Extra opgaven variabelen/strings

### 18.1 Casus 1 - Vind het woord

Schrijf een programma dat voldoet aan de volgende requirements: + Gebruiker kan een zin (een regel tekst) invoeren. + Gebruiker kan een woord (een stuk tekst zonder spaties) invoeren in een tweede invoerveld. + Het programma toont de positie (index) op het scherm waar het ingevoerde woord zich in de zin bevindt. De positie begint bij 1 te tellen. Als het woord niet voorkomt in de zin dan is de positie 0.

Bijvoorbeeld in de zin “De appel valt niet ver van de boom.” komt het woord “appel” voor op positie 4.

### 18.2 Casus 2 - BMI-calculator

Schrijf een programma dat de BMI van een persoon uitrekent en op het scherm laat zien. Tip: voorafgaand aan het programmeren (=realisatiefase) ga je een kort vooronderzoek (=analyse) uitvoeren waarbij je de volgende vragen beantwoordt: + Wat is BMI? + Hoe reken je voor een man/vrouw van een bepaalde leeftijd de BMI uit?

### 18.3 Casus 3 - Oppervlakte-calculator

Schrijf een programma dat de oppervlakte van een cirkel uitrekent. De gebruiker vult de middellijn van de cirkel in en het programma toont de oppervlakte van de cirkel met die middellijn op het scherm.

Tip: voer een korte analyse uit waarbij je het antwoord op de vraag Hoe reken je de oppervlakte van een cirkel uit? achterhaald.

## 18.4 Casus 4 - Listbox-vuller

Maak een programma met een ListBox, een TextBox en een Button. Als de gebruiker op de Button klikt dan wordt de tekst die in de TextBox staat (de gebruiker kan er intypen wat hij wil) toegevoegd aan de lijst in de ListBox.

## 18.5 Casus 5 - Ak-tester

Maak een programma waarmee de gebruiker een tekst kan invoeren en op een knop kan klikken. Na het klikken op de knop verschijnt er een tekst in beeld die aangeeft op welke positie de zinsnede ak voorkomt.

Voorbeeld-scenario: de gebruiker vult het woord “goudakaas” in. Het programma moet dan na een druk op de knop de uitkomst 5 tonen (omdat de - eerstvoorkomende - substring ak op de 5e plaats staat).

## 18.6 Casus 6 - Je werkelijke leeftijd

Maak een programma dat gegeven de naam van de gebruiker (voornaam + achternaam) en zijn leeftijd de “werkelijke leeftijd” die gebruiker op het scherm toont.

De “werkelijke leeftijd” van een gebruiker wordt als volgt bepaald: + De ingevulde leeftijd minus 12 vermenigvuldigt met het tiende deel van het aantal letters van zijn volledige naam.

## 18.7 Casus 7 - Raad-een-getal (if-statement)

Opdracht: schrijf een applicatie die voldoet aan de volgende requirements: + De gebruiker moet een getal kunnen invullen in een TextBox (dus niet een ander control gebruiken dan een TextBox). De gebruiker vult een getal in en klikt op een knop. + Indien na het klikken op de knop het getal gelijk is aan 3 dan heeft de gebruiker het goed en verschijnt er een tekst die dat aan de gebruiker mededeelt. Als er een ander getal dan 3 wordt ingevuld verschijnt er een andere tekst met de melding dat dit niet goed is.

Zodra dit werkt breidt dan je applicatie uit met de volgende functionaliteit: + Je programma geeft bij een fout antwoord van de gebruiker aan of het getal hoger is of lager dan het te raden getal 3.

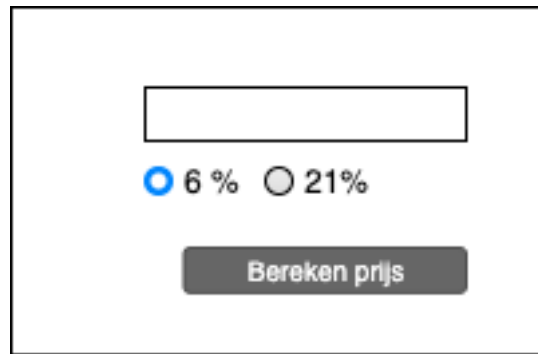
## 18.8 Casus 8 - Afstanden omrekenen naar km

In Engeland gebruiken ze ‘inch’, ‘foot’, ‘yard’ en ‘mile’. Maak een programma met een textbox, radio button en knop om dit om te rekenen naar kilometers. De gebruiker kan een aantal invoeren in de textbox en dan met radio buttons kiezen wat dat voorstelt.

Bijvoorbeeld 1 invoeren in de textbox en dan de radio button voor ‘mile’ selecteren. Als er op de knop wordt gedrukt wordt het aantal kilometers weergegeven. In dit voorbeeld 1,6 km. Rond altijd af op 1 cijfer achter de komma.

## 19 Training - Prijzen met BTW berekenen

Maak een programma waarbij je een prijs in een textbox kunt zetten. Dan moet je als gebruiker nog de keus hebben tussen BTW tarief hoog en laag. Zet deze twee keuzes in een combobox of gebruik een radio button. Afhankelijk van de keus moet je na het drukken op de knop de prijs inclusief 6% of 21% BTW tonen.



The image shows a user interface design for a program that calculates prices with BTW. It consists of a rectangular container with a light gray background. Inside the container, there is a white rectangular text input field at the top. Below the input field, there are two radio buttons. The first radio button is selected (indicated by a blue dot) and is followed by the text "6 %". The second radio button is not selected (indicated by a gray dot) and is followed by the text "21%". Below the radio buttons, there is a dark gray rectangular button with the text "Bereken prijs" in white.

Figure 19.1: Aangeleverd ontwerp user interface

## 20 Training - Hoger-lager

Schrijf een programma dat voldoet aan de volgende requirements: 1. Het programma maakt een willekeurig getal aan en onthoudt dit in een variabele, maar laat het niet zien op het scherm. 2. Gebruiker kan getal invullen. 3. Het programma toont in een label “Hoger” als het door de gebruiker getal lager is dan het in stap 1 onthouden getal en “Lager” indien de gebruiker te hoog zat. Als het getal precies goed is (geraden) dan wordt er een MessageBox getoond “Geraden!”.



Figure 20.1: Aangeleverd ontwerp user interface

### 20.1 Uitbreiding

Heb je naast *keuzestructuren* extra oefening nodig op het gebied van *variabelen*? Maak dan deze casus. Breid casus 1 uit met de volgende functionaliteiten: 1. Als het geraden getal te laag is dan wordt de tekstkleur van het label blauw, en als het getal te hoog is dan wordt de tekstkleur rood. 2. Het aantal beurten dat de gebruiker nodig heeft om het getal te raden wordt door het programma bijgehouden en getoond in de de MessageBox: “Geraden in 123 keer!”.

## 21 Training Calorieën-tracker

Voor je gezondheid is het belangrijk om in de gaten te houden hoeveel calorieën je dagelijks nodig hebt. Jouw taak is een applicatie te maken die uitrekent hoeveel calorieën de gebruiker dagelijks nodig heeft.

### 21.1 Analyse

Je collega heeft een vooronderzoek uitgevoerd en de volgende informatie verzameld:

- Een vrouw heeft per dag gemiddeld 2.000 kilocalorieën (kcal) nodig. Voor de man is dat gemiddeld 2.500 kcal per dag (bron voedingscentrum).
- Maar als je een niet-actieve levensstijl hebt (als je minder dan 30 minuten per dag beweegt) dan heb je 10% minder calorieën nodig.
- En als je boven de 50 bent dan heb je 200 kilocalorieën minder nodig.

*Tip: je ziet in deze analyse twee keer het woord ALS staan. Dat worden in C# vermoedelijk if statements*

### 21.2 Ontwerp

Het ontwerp voor de user interface is reeds gedaan. De gebruiker kan met een radiobutton zijn/haar geslacht en levensstijl aangeven, zijn/haar leeftijd invullen en op de knop **Bereken** klikken. Nadat er op die knop is geklikt verschijnt de caloriebehoefte in beeld (hoe dat mag jij weten: met een messagebox of je voegt een label toe op het form).

### 21.3 Realisatie

Programmeer de applicatie. Probeer van te voren een stappenplan te maken hoe je dit gaat aanpakken. Bijvoorbeeld: 1. Ik ga eerst het scherm (het form) maken met de groupboxen, radiobuttons, enz. 2. Dan programmeer ik eerst een applicatie die kijkt of iemand een man of vrouw is en laat ik als antwoord 2000 of 2500 zien. 3. Ik heb nu de uitkomst (2000 of 2500) in een variabele staan. Nu ga ik kijken naar de radiobuttons voor de levensstijl. Als

Calorieën-tracker

Geslacht

☒ Man

☐ Vrouw

Levensstijl

☒ Actief (normaal)

☐ Niet actief (couch potato)

Leeftijd:

Bereken

Figure 21.1: Aangeleverd ontwerp user interface

(if-statement) hij **Niet Actief** kiest dan haal ik 10% van de uitkomst af. En ik laat dan die uitkomst zien. 4. Nu ga ik kijken wat zijn leeftijd is. Ik haal de leeftijd op uit de textbox en zet die in een variable. Als dat hoger is dan 50 dan trek ik nog eens 200 af van de uitkomst.

*Tip: het voordeel van dit stappenplan is dat je vanaf stap 2 elke keer een werkende applicatie hebt.*

Je hebt nu een werkende applicatie en kunt deze bij de klant opleveren.

## 21.4 Onderhoud

Na een half jaar komt de klant terug en wil hij een aantal nieuwe features aan je applicatie toegevoegd zien.

- Als de leeftijd van de gebruiker kleiner is dan 12 dan moet het aantal calorieën nog eens met 180 vermindert worden.
- Zwangere vrouwen in de leeftijd tot en met 30 jaar hebben 2600 calorieën nodig en zwangere vrouwen boven de 30 hebben 2500 calorieën nodig. Zorg ervoor dat de gebruiker kan invullen of ze zwanger is of niet en pas de berekening aan.
- Testen: test of je met jouw applicatie ook een zwangere man kunt zijn. Zo niet, dan heb je dat goed geprogrammeerd.



## 22 Training - Dobbelsteen

Als je een game wilt programmeren dan stuit je zelfs bij de meest eenvoudige games (boter-kaas-en-eieren) op het probleem: hoe programmeer ik willekeur? Doe een willekeurige zet? Hoe programmeer je dat?

Om dit te ervaren ga je een klein programma schrijven waarmee je het willekeurige gedrag van een dobbelsteen simuleert. Als dit zogenaamde “proof-of-concept”-programma werkt weet je beter hoe je tijdens het ontwerpen rekening met de benodigde willekeur kunt houden en heb je meer zekerheid over de kans van slagen van de game.

### 22.1 Opdracht

Maak een user interface met 6 picturebox-en en een button die er zo uit ziet:

Er staan zes objecten van het type **PictureBox** en een “Werp dobbelsteen!”-Button op het formulier. De **PictureBox**-objecten zijn allemaal onzichtbaar als het programma wordt uitgevoerd (de **Visible Property** staat op **false**).

Het is de bedoeling dat telkens als op de “Werp dobbelsteen!”-**Button** geklikt wordt, precies één willekeurige PictureBox zichtbaar wordt. De andere pictureboxen worden dus onzichtbaar. Programmeer deze functionaliteit.

### 22.2 Uitbreidingen

- Niveau \* - Plaats alle **PictureBox**-objecten precies over elkaar heen (zo lijkt het alsof telkens dezelfde dobbelsteen opnieuw wordt gegooit).
- Niveau \*\* - Maak gebruik van één switch statement (tip: Zoek op internet naar een bron) in plaats van verschillende “if ... else ...” statements.

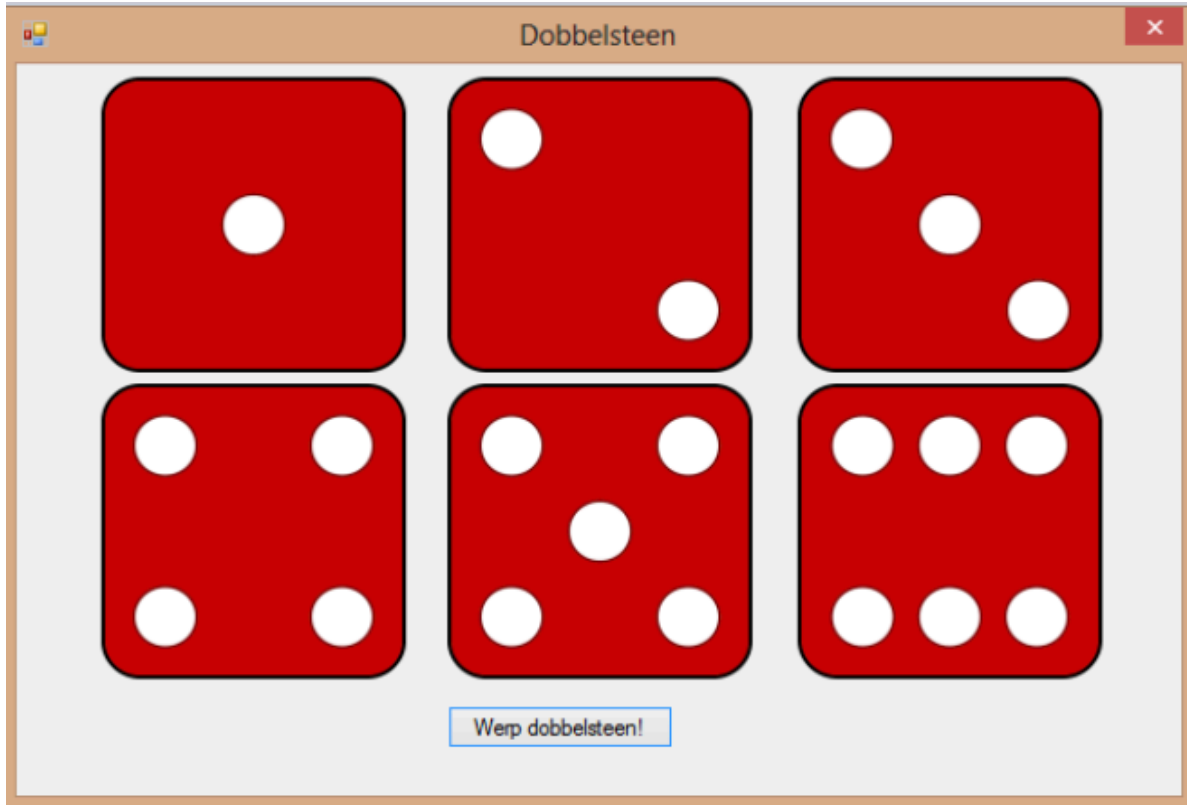


Figure 22.1: Aangeleverd ontwerp user interface

## 22.3 Checklist

Als je de opdracht op de juiste manier hebt uitgevoerd heb je voldaan aan onderstaande punten:

- Bij het starten van het programma zijn geen dobbelsteenafbeeldingen zichtbaar. - Bij een druk op de knop wordt precies één afbeelding van een dobbelsteen zichtbaar. - Bij de volgende druk op de knop wordt weer precies één afbeelding (dit kan toevallig dezelfde zijn) van een dobbelsteen zichtbaar. - Als je tientallen keren na elkaar op de knop klikt zie je de afbeeldingen voor dobbelsteen 1, 2, 3, 4, 5 en 6 in ieder geval één keer getoond worden. - Als je tientallen keren na elkaar op de knop klikt worden nooit twee afbeeldingen tegelijkertijd getoond. - Je hebt niet meer “if ... else ...” constructies gebruikt dan nodig. - Je hebt in iedere “if ... else ...” constructie in het “if” blok regels code staan. - Je hebt in iedere “if ... else ...” constructie in het “else” blok regels code staan of je hebt het “else” blok weggelaten. - Uitbreiding a: Ongeacht welke waarde wordt gegooid, de afbeelding met het aantal ogen wordt altijd op dezelfde plek getoond. - Uitbreiding b: Naast het “switch” statement heb je geen “if...else...” statements meer gebruikt en je hebt geen “default” case binnen het switch statement gebruikt.

## 22.4 Bronnen

- [Keuzestructuren](#)
- [Random class Microsoft Docs](#)

## 23 Training - Pincode reminder

Een pincode is vaak moeilijker te onthouden dan een wachtwoord, en de bank raadt je af je pincode ergens te noteren om te voorkomen dat iemand er gemakkelijk met je geld vandoor gaat. Het zou handig zijn om een programma te hebben waarmee je met het invoeren van het juiste wachtwoord je pincode te zien krijgt. Een zelfgekozen wachtwoord is immers vaak een stuk gemakkelijker te onthouden dan een pincode.

### 23.1 Opdracht

Maak in je nieuwe project een grafische user interface voor het programma waarmee je een wachtwoord kunt invoeren in een textbox en je invoer kunt bevestigen met een button. Verder moeten er twee labels gemaakt worden waarmee je de gebruiker voorziet van informatie als op de knop gedrukt is: als het wachtwoord juist is wordt de ene tekst zichtbaar, als het wachtwoord onjuist is wordt de andere tekst zichtbaar.

Je user interface zou er bijvoorbeeld zo uit kunnen zien:

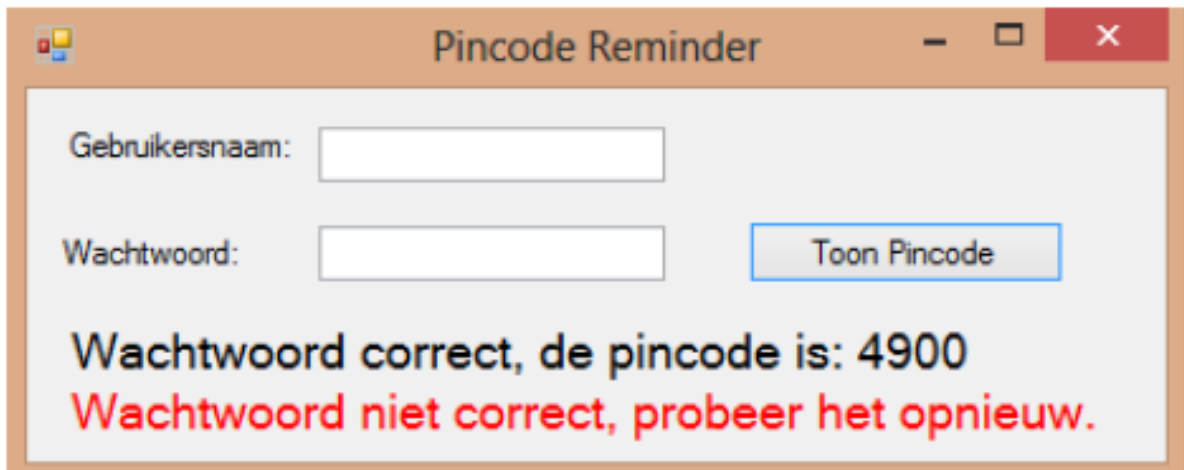


Figure 23.1: Aangeleverd ontwerp user interface

Geef, zoals je in een eerdere opdracht gedaan hebt, de objecten die je gebruikt een duidelijke naam. Als je straks namelijk code toe gaat voegen is het handiger om te werken met namen

als **labelRight**, **labelWrong** en **showPincodeButton** dan met namen als **label1**, **label2** en **button1**.

### 23.1.1 Tips

- Zorg dat de zichtbaarheidseigenschap (**Visibility**) van de objecten goed wordt ingesteld.
- Zorg dat de EventHandler voor de Button wordt aangemaakt en voorzien van code zodanig dat het programma doet wat het zou moeten doen.
- Als je iets onzichtbaar maakt zie je dit niet direct in de editor maar pas als je het programma draait.
- Door te dubbelklikken op de knop kun je de voor de Button meest gebruikte **EventHandler**, namelijk “Click” (lees: de gebruiker heeft op de knop geklikt), automatisch aanmaken. Je zult zien dat je daardoor direct in een nog niet ingevulde methode in de programmacode van je formulier springt.

## 23.2 Uitbreidingen

- Niveau \* - Zorg dat als het wachtwoord wordt ingevoerd alleen sterretjes/bolletjes te zien zijn (dit is een **Property** van een TextBox). Ook moet, nadat de invoer gecontroleerd is de tekst in het wachtwoord-tekstvak altijd leeg worden gemaakt (of het wachtwoord nu klopt of niet).
- Niveau \*\*\* - Zorg dat het programma gebruikt kan worden door twee personen (gebruikers). Er moet niet alleen gevraagd worden om een wachtwoord, maar om een naam -en een wachtwoord. Pas als de combinatie naam/wachtwoord klopt wordt de pincode voor die gebruiker getoond.

## 23.3 Checklist

Als je de opdracht op de juiste manier hebt uitgevoerd heb je voldaan aan onderstaande punten:

- Bij het starten van het programma zijn de pincode en de foutmelding onzichtbaar.
- Alleen als het wachtwoord klopt wordt de pincode getoond, anders de foutmelding.
- Als je na het invoeren van een fout wachtwoord het goede intypt moet het programma nog naar behoren werken.
- Als je na het invoeren van het juiste wachtwoord een verkeerd wachtwoord invoert moet het programma nog naar behoren werken.
- Je code is duidelijk te lezen doordat je de objecten duidelijke namen hebt gegeven.
- De GUI ziet er verzorgd uit (duidelijke teksten, goede uitlijning).
- Uitbreiding a: De getoonde pincode hoort bij de betreffende gebruiker
- Uitbreiding b: Bij het invoeren van het wachtwoord zie je alleen sterretjes/bolletjes in het tekstvak verschijnen.
- Uitbreiding b: Na een druk op de knop wordt altijd het tekstvak automatisch leeggemaakt, of het wachtwoord nu goed is of niet.

## 24 Training - Wachtwoord check

Schrijf een programma dat voldoet aan de volgende requirements:

1. Gebruiker kan een wachtwoord invullen (wachtwoord is niet zichtbaar, er staan bolletjes of sterretjes). Na het invullen van het wachtwoord klikt de gebruiker op een knop.
2. Een panel geeft met 4 kleuren aan hoe sterk het wachtwoord is. Indien het wachtwoord sterk is dan is het panel groen, middelmatig sterk dan is het geel, zwak dan is het panel rood en heel zwak dan is het panel zwart.

a. Sterk (groen): er zit een cijfer 1, 2 of 3 in het wachtwoord EN het wachtwoord heeft een lengte groter dan 3.

1. Sterk (groen): er zit een cijfer 1, 2 of 3 in het wachtwoord EN het wachtwoord heeft een lengte groter dan 3.
2. Zwak (rood): er zit geen cijfer 1, 2 of 3 in het wachtwoord EN het wachtwoord heeft een lengte groter dan 3.
3. Zeer zwak (zwart): de lengte van het wachtwoord is korter dan 3 karakters.
4. Middelmatig (geel): alle overige gevallen.

## 25 Training - Dr. Bobby

Dr. Bobby is een computerpsychiater. Jij gaat Dr. Bobby programmeren door een programma te schrijven dat voldoet aan de volgende requirements:

De gebruiker kan een vraag stellen aan het systeem door deze in te typen in een textbox en op een knop te klikken. Na het klikken op de knop verschijnt het antwoord. 1. Als de vraag begint met het woord *waarom* dan antwoord Dr. Bobby met “Waarom denkt u zelf?”. 2. Als de vraag begint met het woord *wie* dan antwoord Dr. Bobby met “Wie denkt u zelf, ” waarbij gelijk is aan de door de gebruiker ingevulde tekst minus *wie*. Voorbeeld: De gebruiker typt “Wie is Sinterklaas?” Dr. Bobby: Wie denkt u zelf, is Sinterklaas?” 3. Als de vraag begint met het woord *wat* dan antwoord Dr. Bobby met “Wat denkt u, ” waarbij gelijk is aan de door de gebruiker ingevulde tekst minus *wat*. Voorbeeld: De gebruiker typt “Wat is het doel van dit gesprek?” Dr. Bobby: Wat denkt u zelf, is het doel van dit gesprek?” 4. In overige gevallen antwoord Dr. Bobby “Waarom wilt u dat weten, denkt u?”

## 26 Training - Batman vs Superman

Batman en Superman hebben onenigheid gekregen. Ze beweren elk dat ze elkaar zouden kunnen verslaan in een gevecht. Jij moet ze helpen om te kijken wie er zou winnen.

Maak een applicatie en geef elk van de twee superhelden twee radio buttons. Batman heeft er twee die (1) Batarang en (2) Kung-fu heten. Superman heeft (1) Laser en (2) Superkracht. Elk van deze aanvallen delen schade uit aan de tegenstander. In onderstaande tabel zie je hoeveel schade.

De batarang doet tussen de 1 en 8 schade. Kung-fu doet tussen de 5 en 7 schade.

De laser doet tussen de 5 en 7 schade. Superkracht doet tussen de 6 en 9 schade.

Laat met een druk op de knop zien wie er heeft gewonnen bij het gebruik van bepaalde wapens.

The image shows a user interface for a simulation titled "Batman vs Superman". It is divided into two columns. The left column is for Batman, with the title "Batman" in bold. It contains two radio buttons: "Batarang" and "Kung-fu", both of which are currently unselected. Below the radio buttons, it says "Kracht over: 10". At the bottom of the column is a dark grey button labeled "Aanvallen". The right column is for Superman, with the title "Superman" in bold. It contains two radio buttons: "Laser" and "Superkracht". The "Laser" button is selected, indicated by a blue dot. Below the radio buttons, it says "Kracht over: 20". At the bottom of the column is a dark grey button labeled "Aanvallen".

Figure 26.1: Aangeleverd ontwerp user interface



## 27 Training - For - Drankjesschema

Maak een applicatie waarmee je drankjes kunt toevoegen aan een **Listbox**. Als gebruiker moet je de soort van het drankje kunnen invoeren, bijvoorbeeld Red Bull of Whisky. Ook moet je het aantal van dat soort drankje kunnen ingeven, bijvoorbeeld 1, 3 of 6. Met een druk op de knop wordt het juiste aantal drankjes toegevoegd aan de lijst. Zet een volgnummer achter de soort van de drankje. Stel dat je drie drankjes wilt toevoegen. Dan zal er achtereenvolgens “whisky 1”, “whisky 2” en “whisky 3” worden toegevoegd aan de lijst.

Houdt onderaan de lijst ook het totaal aantal drankjes bij.

Soort drankje	Whisky 1
Aantal	Whisky 2
Voeg toe	Redbull 1
	Redbull 2
	Redbull 3
	Spa Blauw 1
Totaal aantal drankjes: 6	

Figure 27.1: Userinterface

## 28 Training - For - Wiskunde en zo

Niets is zo leuk als wiskunde! Waarom zou je niet wiskunde gebruiken om herhalingsstructuren te leren?

### 28.1 Casus 1 - De tafel van 3

Schrijf een programma dat voldoet aan de volgende requirements: 1. Toon de tafel van 3 op het scherm door gebruik te maken van een for-lus. 2. Voeg een tweede project toe aan je solution en maak opnieuw een programma dat de tafel van 3 laat zien, maar dan met een while-lus.

### 28.2 Casus 2 - De tafel van N

Schrijf een programma dat voldoet aan de volgende requirements: 1. Toon de tafel van N op het scherm door gebruik te maken van een for-lus. N is door de gebruiker in te vullen in een TextBox. 2. Voeg een tweede project toe aan je solution en maak opnieuw een programma dat de tafel van N laat zien, maar dan met een while-lus.

### 28.3 Casus 3 - Lus met een if er in

Schrijf een programma dat voldoet aan de volgende requirements: 1. Maak een programma dat de tafel van 8 kan uitrekenen en op het scherm in een listbox (elke item op een nieuwe regel in de listbox) kan laten zien. Het programma moet werken met een for-lus. 2. Elke keer als de uitkomst deelbaar is door 3 komt er in de listbox de tekst “DEELBAAR” achter de uitkomst te staan. Hiervoor heb je de volgende technieken nodig (doe een vooronderzoek): 1. Gebruik de % operator om te bepalen of een getal deelbaar is door 3. 2. Het gebruik van een if-statement is verplicht.

## 28.4 Casus 4 - Geneste lus

Schrijf een programma (console app) dat de volgende tekst op het scherm kan laten zien:

```
1
22
333
4444
55555
```

Het algoritme moet werken met een **geneste lus** (een lus in een lus).

## 28.5 Casus 5 - Som of product van bepaalde reeks getallen

Maak een programma dat de gebruiker om een getal vraagt en vervolgens de som van alle getallen tot en met dit getal op het scherm toont. Dus, stel de gebruiker voert 3 in, dan is het resultaat  $1 + 2 + 3 = 6$ .

Pas het programma aan zodat alleen veelvouden van 3 en 5 meegenomen worden in de berekening. Dus, stel de gebruiker voert 7 in, dan is het resultaat  $3 + 5 + 6 = 14$ .

Pas het programma aan zodat de gebruiker kan kiezen of het de som of de vermenigvuldiging van alle getallen tot n berekend moet worden. Dus, stel de gebruiker voert 4 in en vraagt om de *som*, dan is het resultaat  $1 + 2 + 3 + 4 = 10$ . Stel de gebruiker voert 3 in en vraagt om het *product*, dan is het resultaat  $1 * 2 * 3 = 6$ .

## 28.6 Casus 6 - Som van een reeks getallen

Maak een programma dat de vermenigvuldigingstabel tot en met 10 laat zien van een door de gebruiker opgegeven getal. Dus, stel de gebruiker voert 7 in, dan is het resultaat  $1 * 7 = 7$ ,  $2 * 7 = 14$ , ...,  $10 * 7 = 70$ . Zet ieder product op een nieuwe regel.

## 29 Training - For - Worpengenerator

In veel bordspellen (of digitale varianten hiervan) worden worpen gedaan met meerdere dobbelstenen (in het bijzonder bij spellen als Yahtzee, Risk, etc.)

Indien je geen dobbelstenen voor handen hebt of je een digitale variant van een spel aan het programmeren bent, komt een programma waarmee een x aantal dobbelstenen met y aantal ogen gegooid kan worden goed van pas.

Het programma dat je aan het eind van deze opdracht hebt gemaakt is ziet er als volgt uit:

### 29.1 Deel 1

Maak een nieuw C# Windows Forms-project aan. Maak een user interface welke er als hieronder uitziet (dit is een vereenvoudigde vorm van het eindresultaat, we gaan nog even uit van dobbelstenen met 6 ogen). De kaders (Instellingen en Resultaten) zijn **GroupBox** objecten. Hiermee kun je een aantal GUI-objecten die bij elkaar horen als groepje bij elkaar zetten en later, mocht het nodig zijn, bijv. als geheel verplaatsen. Het is het handigste om de **GroupBox**-en eerst aan te maken zodat je de andere objecten hier op kunt zetten. Het witte uitvoer vlak is een **ListBox**. - Geef de Button, NumericUpDown, ListBox, Labels en GroupBox-en duidelijke namen. - Zorg dat het aantal worpen dat gekozen mag worden minimaal 1 is en maximaal 1000. - Maak de EventHandler aan voor klikken op de Button “Gooi dobbelstenen!”. - Zorg dat als eerste de ListBox wordt leeggemaakt. Doe dit door de `Items.Clear()` methode aan te roepen van de ListBox. Dus: als je ListBox als naam `mijnListBox` heeft gaat dit zo: `mijnListBox.Items.Clear()` - In de EventHandler moet vervolgens de code komen te staan waardoor “aantal worpen” keer een willekeurig getal van 1 t/m 6 wordt toegevoegd aan de ListBox. Maak deze code.

#### 29.1.1 Tips

- Begin klein door eerst maar één worp uit te voeren en te tonen, dan kun je de code daarna gemakkelijker uitbreiden door een herhaling (for lus of while lus) toe te voegen.
- Aan een ListBox kun je een regel toevoegen door gebruik te maken van de methode `Items.Add()`. Dus voor de ListBox met de naam `mijnListBox` gaat het toevoegen van het getal tien zo: `mijnListBox.Items.Add(10);`

- Maak aan het begin van de EventHandler één keer een dobbelsteenobject aan en gooi hiermee “aantal worpen” keer. Dit voorkomt het telkens dezelfde hoeveelheid ogen gooien.
- Test of het programma goed werkt door veel dobbelstenen te gooien en te kijken of hierin alleen de waarden 1 t/m 6 voorkomen.

## 29.2 Deel 2

Breid de user interface uit met twee **Label** objecten zoals hieronder. De ene bevat de tekst “Totaal aantal ogen:”, de andere bevat de standaard waarde “0” en zal straks het totaal aantal ogen voor de worp bevatten. Geef de **Label** objecten duidelijke namen.

Zorg dat in de EventHandler van de Button de code wordt toegevoegd waarmee het totaal aantal gegooide ogen wordt bijgehouden. Maak hiervoor gebruik van een “totaalOgen” variabele van het type int.

### 29.2.1 Tips

- Als je een variabele binnen een for of while lus aanmaakt dan wordt deze tijdens elke herhaling aangemaakt en op het einde van de herhaling weer opgeruimd (dus voor iedere worp van iedere dobbelsteen wordt de variabele aangemaakt en weer weggegooid). Als je een variabele aanmaakt buiten, dus vlak voor een for of while lus, dan blijft deze variabele gedurende de uitvoering van de gehele for of while lus bestaan en te benaderen vanuit de for of while lus.
- Als je totaalOgen variabele na het uitvoeren van de for of while lus altijd precies de waarde heeft van de laatste worp, dan heb je heel waarschijnlijk één van de volgende fouten gemaakt:
  - Je hoogt de variabele totaalOgen niet op maar overschrijft deze met het aantal gegooide ogen.
  - Je hebt de totaalOgen variabele niet buiten de for of while lus gedeclareerd (aangemaakt), zie de tip hierboven voor wat je daaraan kunt doen.
- Test of het totaal aantal ogen juist wordt opgeteld door dit zelf na te tellen voor een aantal verschillende worpen, met telkens een andere hoeveelheid dobbelstenen.

## 29.3 Deel 3

Breid de user interface uit met een Label en een NumericUpDown zoals onderstaande afbeelding. Op het Label komt de tekst “ogen op dobbelsteen”, de NumericUpDown krijgt als minimumwaarde 2 en als maximumwaarde 100. De standaardwaarde komt op 6 te staan (dit is immers de meest voorkomende hoeveelheid ogen op een dobbelsteen).

Zorg dat in plaats van een waarde van 1 t/m 6 nu een waarde van 1 t/m de waarde in de nieuwe NumericUpDown wordt gegenereerd.

## 29.4 Uitbreidingen

- Niveau \*\* - Voeg twee extra Label objecten toe onder het totaal aantal ogen en houd hierin de “Hoogte worp tot nu toe” in bij. Doe hetzelfde voor het laagste aantal worpen tot nu toe.

## 29.5 Checklist

Als je de opdracht op de juiste manier hebt uitgevoerd heb je voldaan aan onderstaande punten:

- De NumericUpDown aantal-worpen heeft een minimumwaarde van 1 en een maximumwaarde van 1000.
- De NumericUpDown aantal-ogen-op-dobbelsteen heeft een minimumwaarde van 2, een maximumwaarde van 100 en een standaard waarde van 6.
- Het Label totaal-aantal-ogen staat bij het opstarten op '0'
- De gegenereerde waarden kloppen met de mogelijke waarden die je hebt opgegeven in de NumericUpDown aantal-ogen-op-dobbelsteen. De maximum- en minimumwaarde komen voor in het lijstje (als je lang genoeg dobbelstenen gooit). Test dit met een kleine hoeveelheid ogen, anders ben je lang bezig.
- Het aantal gegenereerde waarden komt precies overeen met de waarde in de NumericUpDown aantal-worpen (je zit er bijv. niet precies eentje naast).
- Uitbreiding: Controleer of het maximum en minimum kloppen na veel verschillende worpen.
- Is alles in orde? Kijk dan je uitwerking na met behulp van de standaarduitwerking op de volgende pagina. Heb je alles goed? Geef dan jezelf een **schouderklopje** of vraag je favoriete klasgenoot jou er een te geven, want je hebt 'm verdiend.

## 30 Training - For - Muziekgenerator

Maak een applicatie waarbij de gebruiker het aantal noten dat er achtereenvolgens gespeeld moeten worden kan aangeven. Vervolgens moet je willekeurig kiezen uit de noten C, D, E, F, G, A en B. Optioneel (50% kans) moet je bij de noten C, D, F, G en A een sharp noteren (#). Zet de uitkomst van het muziekstuk in een label, textbox, MessageBox of geef de output in de console.

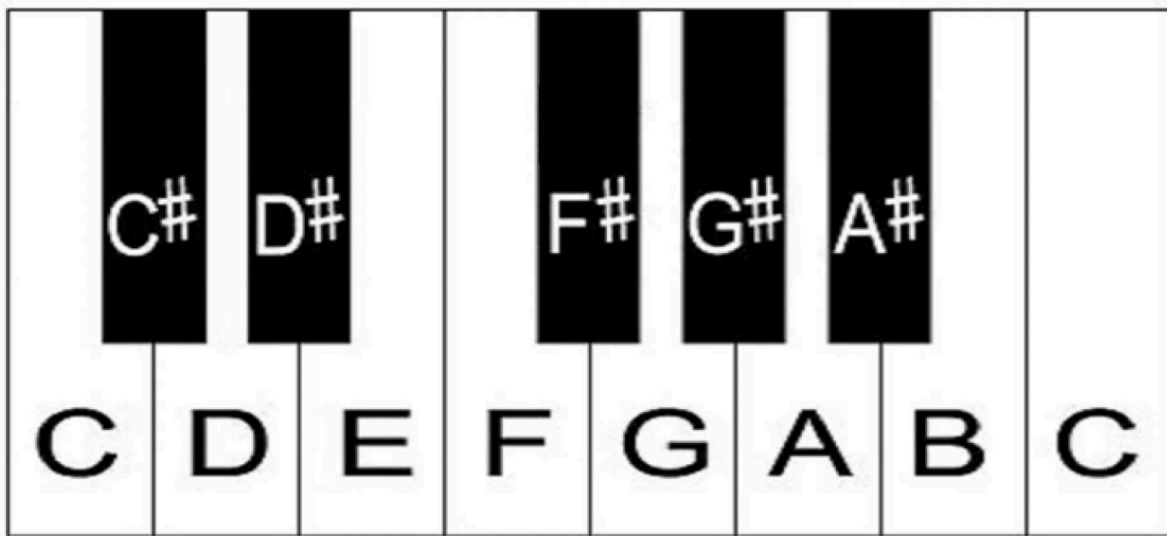


Figure 30.1: Noten op een piano

De gebruiker zou bijvoorbeeld 10 noten willen spelen. De applicatie zou dan willekeurig met de volgende partij noten kunnen komen:

**D – E – F# – F# – C – B – G# – A# – A – A#**

## 31 Training - While - Het grote geheel

Maak een applicatie met een while loop met daarbij een variabele “totaal”. In deze while loop wordt een willekeurig cijfer gegenereerd. Deze willekeurige cijfer wordt bij de variabele “totaal” opgeteld. Wanneer de variabele “totaal” hoger is dan 1000, wordt er een melding gegeven “Het getal is nu hoger!”.

Breid het programma uit dat je zelf het “hoger dan getal” kan invoeren.



## 32 Training- While - Het goede getal

Laat de gebruiker een getal invoeren. Controleer dat het een getal tussen de 0 en 10 is. Na een druk op de knop ga je willekeurige getallen genereren totdat je het getal van de gebruiker hebt. Dus stel dat het getal 6 is ingevoerd. Dan ga je een willekeurig getal genereren tussen de 0 en 10. Stel je trekt 3, dan moet je dus nogmaals gaan genereren. Dit blijf je net zolang herhalen tot je een 6 hebt getrokken. Je toont een melding aan de gebruiker: “Het goede getal is na 3 keer gevonden!”.

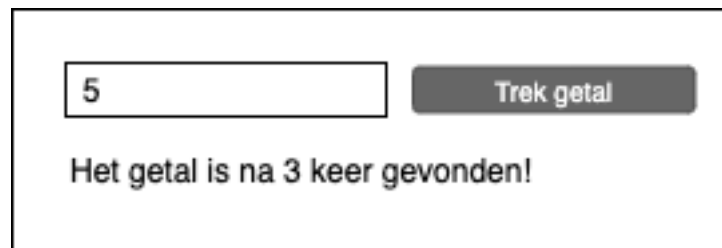


Figure 32.1: Userinterface

## 33 Training - Christmas Tree Generator

Maak een methode waarmee je op een willekeurige positie op het scherm een kerstboom van een bepaalde hoogte kan maken in een bepaalde kleur. Ook moet het symbool kunnen worden meegegeven. In het screenshot hieronder hebben we de kerstbomen de volgende parameters/eigenschappen:

Kleur	Hoogte	Left/x	Top/y	Tekensymbool
Rood	10	10	10	8
Groen	5	20	5	5

Gebruik een *Console* applicatie. Tip: gebruik **Console.ReadKey()** om te voorkomen dat het programma direct na opstarten weer afsluit.

### 33.1 Stappenplan

1. Een bepaald karakter op het scherm laten afdrukken
2. Een horizontale lijn van meerdere karakters laten afdrukken
3. Een horizontale lijn van meerdere karakters laten afdrukken op willekeurige positie
4. Een horizontale lijn van meerdere karakters laten afdrukken op willekeurige positie in een bepaalde kleur
5. Een kerstboom vorm maken
6. Meerdere kerstbomen

### 33.2 Uitbreidingen

- Niveau \*\* Vul de kerstboom met de karakters van een gegeven string, dus bijv. “abc” dan wordt de hele kerstboom gevuld met deze letters
- Niveau \*\* Vraag aan de gebruiker op welke positie de kerstboom moet worden geplaatst
- Niveau \*\*\* Plaats een willekeurig aantal kerstbomen met allemaal verschillende eigenschappen op een willekeurige plek, echter wel geheel binnen het venster.



Figure 33.1: Chistmas tree

- Niveau \*\* Versier de kerstboom met willekeurig op te hangen gekleurde kerstbomen (een letter O). De kerstballen mogen niet te dicht bij elkaar hangen, er moeten minimaal 2 vakjes tussen zitten.
- Niveau \*\*\* Maak een grafische variant in WinForms die van Windows GDI Graphics gebruik maakt.

### 33.3 Bronnen

- [Color in Console App](#)
- [Console.SetCursorPosition](#)
- [Console.ReadLine](#)

## 34 Training - Emmer vol laten lopen

Maak gebruik van de herhalingsstructuur **while** om een emmer te vullen door herhalend een beker water erin te doen totdat ie vol is. Visualiseer de emmer met een meegeleverde custom control **VerticalProgressBar** (zie Bronnen). Deze class kan je gewoon toevoegen aan je Project via RMB op je Project > Add > Existing Item of door het bestand te slepen in je project (+ moet verschijnen tijdens slepen). Als je daarna je Project Build vindt je de control in je ToolBox.

De gebruiker kan instellen hoe groot de emmer (in liters) is en hoe groot de beker (in centiliter) is.

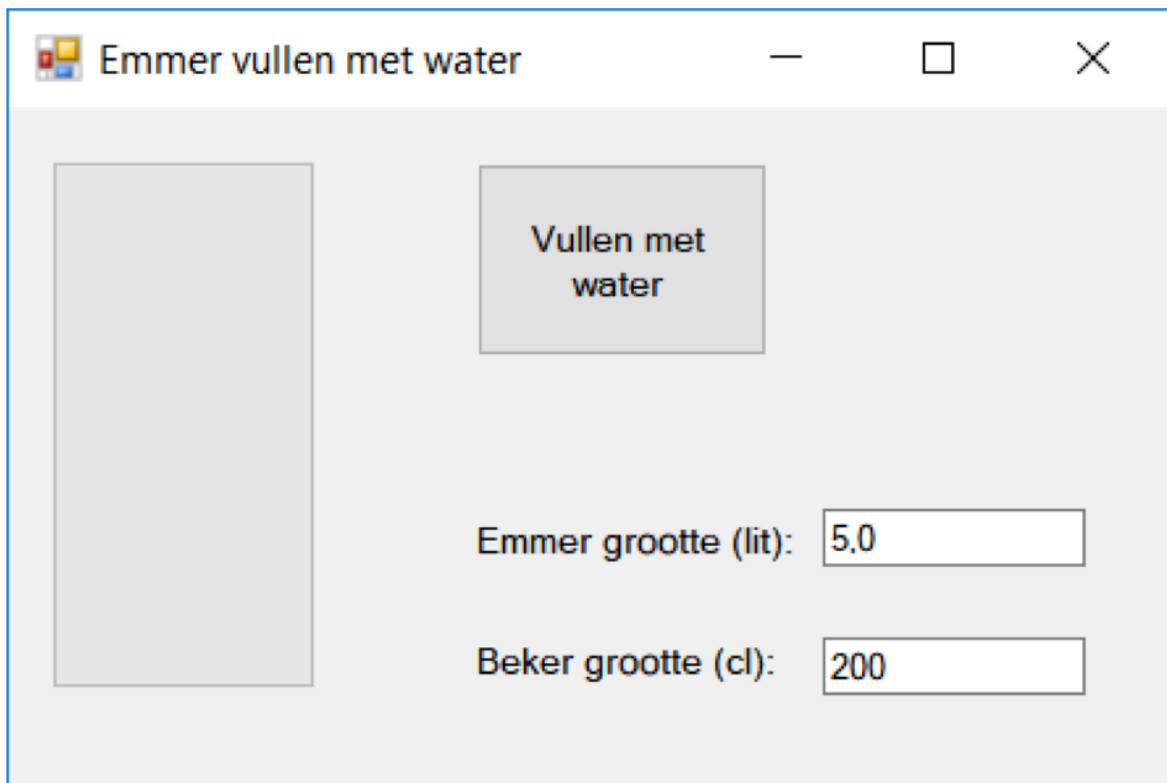


Figure 34.1: Userinterface

Tip: Om ervoor te zorgen dat de emmer niet meteen vol is (doordat de computer te snel de

lus uitvoert) én om de UI te updaten tijdens het uitvoeren van de lus, plaats onderstaande code **in** de lus structuur:

```
Thread.Sleep(100); //doe niks voor 100 ms  
Application.DoEvents(); //update UI
```

Plaats nog wel even de volgende regel bovenin bij de overige using's om de **Thread** class te kunnen gebruiken:

```
using System.Threading;
```

## 34.1 Bronnen

- [Vertical Progressbar](#)

## 35 Training method Ampere

Voer een analyse uit en onderzoek hoe je het volgende kunt uitrekenen:

Gegeven een stroom in Ampère en een spanning in Volt, wat is de waarde van de weerstand (in Ohm) waar die spanning over valt (en die stroom door loopt)? (wet van Ohm)

Schrijf een programma dat voldoet aan de volgende eisen:

- De gebruiker kan de spanning in V en de stroom in A invullen en jouw programma slaat die twee waarden op in twee variabelen. De gebruiker kan 5.5 invullen voor een spanning van vijfeneenhalf volt (gebruik type decimal voor de waarden van spanning en stroom).
- Schrijf code die de waarde van de weerstand uitrekent en die waarde opslaat in een derde variabele.
- Schrijf code die de waarde van de derde variabele laat zien als “De weerstand is 1234.65 ohm”.
- Als je programma nu werkt dan moet je de code die je hebt geschreven overzetten naar een nieuwe methode. De methode ziet er zo uit:

```
decimal Weerstand(decimal volt, decimal ampere)
{
    // zet in plaats van deze regel je eigen code!
}
```

De in-te-vullen code heb je al geschreven, verplaats die bestaande code. Verander hierbij desgewenst de namen van de variabelen. + Roep de methode Weerstand aan en test je programma.

## 36 Training method MaalDrie

Deze opdracht is ooit geschreven als een WinForm-opdracht, maar je mag ook een variatie hierop maken, zoals een Console app.

Gegeven is de volgende methode:

```
int MaalDrie(int input)
{
    int output = input * 3;
    return output;
}
```

- Maak een programma met onder meer een ListBox dat bovenstaande methode aanroept met parameterwaarde 2. De returnwaarde van de methode wordt dan 6. Sla die returnwaarde op in een variabele met de naam output.
- Voeg de waarde van de returnwaarde (6, die je in de variabele uitkomst hebt staan) toe aan de ListBox met `listBoxX.Items.Add(uitkomst);` waarbij `listBoxX` de duidelijke naam is die jij de ListBox gegeven hebt.
- Test je programma.
- Breid je programma uit met een lus die 10 maal doorlopen wordt. Roep in die lus de methode `MaalDrie` aan maar nu met parameter 1, 2, 3, 4... t/m 10. Dus de methode `MaalDrie` wordt 10 maal aangeroepen.

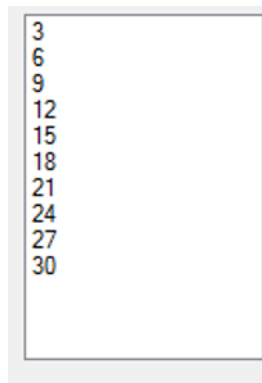


Figure 36.1: listbox



## 37 Training De Drie Vierkanten

Schrijf een programma dat voldoet aan de volgende requirements. Het mag een Console app zijn, of een WinForm app, of misschien nog wel iets anders?

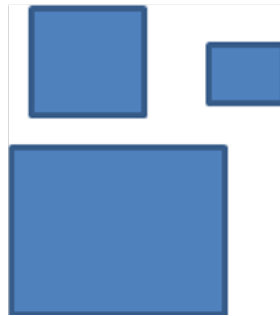


Figure 37.1: 3 vierkanten

- De gebruiker voert de lengte van een zijde van drie verschillende vierkanten in. Deze drie waarden worden in drie variabelen van het type `int` opgeslagen.
- Van alle drie de vierkanten wordt de oppervlakte uitgerekend.
- De totale oppervlakte van alle drie de vierkanten samen (dus `oppervlakte1 + oppervlakte2 + oppervlakte3`) wordt aan de gebruiker getoond.
- Schrijf een `method` die de oppervlakte van een vierkant kan uitrekenen. De method heeft een parameter (die de lengte van een zijde van het vierkant voorstelt) en retourneert (`return`) de oppervlakte van het vierkant.
- Gebruik je methode om de hierboven gemaakte code te verbeteren (methode aanroepen).

## 38 Training - Listige listboxen

Maak een scherm met daarop **2 listboxen**, **1 knop** en **een textbox**. Het scherm ziet er dan bijvoorbeeld als volgt uit.

Vul de listboxen met 10 willekeurige getallen. Gebruik hierbij de **random generator** en de methode **listbox.items.add**

Zorg er voor dat er na het klikken op de button een event wordt gestart die onderstaande handelingen uitvoert: 1. Ga in een **methode** met een **FOR**-loop door de 1e listbox en tel de getallen op in een variabele. Je moet dus deze methode programmeren en die vanuit het button-click-event aanroepen. Geef de waarde van de variabele terug (return-statement in de methode). 2. Programmeer een 2e methode (ook aanroepen vanuit de button-click) als volgt: ga in die 2e **methode** met een **WHILE**-loop door de 2e listbox en tel de getallen op in een 2e variabele. De methode geeft de som van alle getallen van listbox2 terug. 3. Afhankelijk van welke variabele het hoogste resultaat heeft toon je in de resultaat-textbox een tekst ('Listbox ?? heeft de hoogste waarde namelijk: ???'). Indien listbox 1 de hoogste waarde heeft maak je die listbox **groen** en de andere **rood** en andersom.

## 39 Training - Mad lad teksten

Een Mad lad tekst is een tekst waarbij het aantal karakters even is en waarbij de tweede helft van de string het spiegelbeeld is van de eerste helft. Bijvoorbeeld de strings **koffeiffok**, **waterretaw** en **brillirb** zijn allemaal Mad lad teksten.

Schrijf een programma waar de user een tekst in kan voeren en het programma als output “You are a mad lad!” of “You are sooo basic!” indien het geen Mad lad tekst is.

## 40 Training met methode ElkaarAchterstevoren

Maak een methode `ElkaarAchterstevoren` die als returnwaarde een `bool` heeft en 2 parameters van het type `String`. De methode geeft `true` terug precies dan als de ene string gelijk is aan de andere achterstevoren.

Hint: Gebruik de `String`-methode `SubString` om losse letters van woorden te isoleren.

## 41 Training - Cijfers door woorden vervangen

Schrijf een programma die de tekst van de gebruiker oppakt en daar alle cijfers vervangt door woorden. Dus als de gebruiker *Ik ben 20 jaar* invoert dan is de uitvoer *Ik ben tweenuul jaar*. Voorwaarden:

1. De gebruiker moet zijn tekst met meerdere regels (multiline) kunnen invoeren.
2. De functionaliteit die één cijfer verandert moet in een aparte methode komen (en die methode mag GEEN interactie met de user interface hebben). Noem deze methode **VeranderCijfer**.
3. De functionaliteit die de hele tekst ontleed en aanpast moet in een tweede methode komen. Deze tweede methode maakt gebruik van de methode **VeranderCijfer**.

## 42 Training method marathon

Maak een console-app aan en programmeer hierin de hieronder genoemde methods. Roep ze allemaal aan. Controleer of je overtuigd bent dat het antwoord klopt! Wellicht is het hier en daar handig een flowchart of ander hulpmiddel te gebruiken. Maak daar dan een foto van en voeg die aan je Trainingsopdrachten toe.

Tussendoor vind je hier en daar quotes van beroemde informatici.

Quote: Understand well as I may, my comprehension can only be an infinitesimal fraction of all I want to understand. (Ada Lovelace)

- (i) Maak een method *FullName* met de parameters *forename* en *surname* (voor- en achternaam, als je het liever in het NL doet). Aanroep *FullName*("Ada", "Lovelace") geeft terug: "Ada Lovelace", Zij was de eerste programmeur ooit! De aanroep *FullName*("Alan", "Kay") geeft terug: "Alan Kay". Denk er ook aan dat er een spatie tussen voor- en achternaam moet staan!
- (ii) Maak een method genaamd *Times* met dezelfde parameters als de vorige. De return-waarde is een *int* die het aantal letters in voornaam maal het aantal letters van de achternaam bevat.
- (iiia) Maak een method genaamd *IsIn* met een string-parameter **character** en een string-parameter 'word'. De method geeft een boolean terug: true als het 'character' in 'word' voorkomt, anders false. Als je hier niet zelf uit komt kijk dan eens beneden aan deze pagina naar 'Eerste Hulp Bij Vasthangen'.
- (iiib) Maak weer een method genaamd *InCommon* met de string-parameters *forename* en *surname*, die precies de letters teruggeeft die in voor- én achternaam voorkomen (als String of als List).

Quote: Always remember, however, that there's usually a simpler and better way to do something than the first way that pops into your head." (Donald Knuth)

- (iv) Method *HowMuchLonger*, zelfde parameters, die een getal teruggeeft: het aantal letters dat de achternaam langer is dan de voornaam, óf 0 als de achternaam niet langer is dan de voornaam.
- (v) Method *VoornaamVan* met 1 parameter: de *fullName* (bv. "Donald Knuth", een bekende informaticus). De method retournt de voornaam.

(vi) Idem *AchternaamVan*.

Quote: Most software today is very much like an Egyptian pyramid with millions of bricks piled on top of each other, with no structural integrity, but just done by brute force and thousands of slaves. (Alan Kay)

(vii) Maak een method *VoornaamAchterstevoren* die van de parameter `fullName` de voornaam pakt en de letters achterstevoren achter elkaar plakt in een String: dit is de returnwaarde van deze methode.

(viii) Maak een method *AchternaamAchterstevoren* die van de parameter `fullName` de achternaam pakt en de letters achterstevoren achter elkaar plakt in een String: dit is de returnwaarde van deze methode.

Quote: If debugging is the process of removing software bugs, then programming must be the process of putting them in. (Edsger W. Dijkstra)

(ix) Maak een method *UmEnUmVoorAchter*, die van parameter `fullName` eerste de 1e letter van de voornaam pakt, dan de 1e van de achternaam, dan de 2e letter van de voornaam, 2e van de achternaam, 3e vd voornaam, 3e vd achternaam, 4e ... nou, je snapt het denk ik wel. Als de ene naam 'op' is en de andere nog niet, plak dan de overgebleven letters er nog achter en geef de opgebouwde string terug. Roep deze methode een paar keer aan en controleer het resultaat! Voorbeeld: `UmEnUmVoorAchter("Edsger Dijkstra")` geeft: "EDdisjkesrtra"

Quote: The art of programming is the art of organizing complexity. (Edsger W. Dijkstra)

(x) Je raadt het al: we willen ook een *UmEnUmAchterVoor*, die hetzelfde doet als de *UmEnUmVoorAchter*, maar de 1e letter komt van de achternaam in plaats van de voornaam. Voorbeeld: `UmEnUmVoorAchter("Edsger Dijkstra")` geeft: "DEidjskgsetrra"

Quote: The best programs are written so that computing machines can perform them quickly and so that human beings can understand them clearly. A programmer is ideally an essayist who works with traditional aesthetic and literary forms as well as mathematical concepts, to communicate the way that an algorithm works and to convince a reader that the results will be correct. (Donald Knuth)

(xi) Maak een method *WordCount* die een string binnenkrijgt met een tekst (bijvoorbeeld een van de quotes op deze pagina). De method returnt een geheel getal: het aantal woorden in de string.

(xii) Maak een method *MeanLength* die een string binnenkrijgt met een tekst (bijvoorbeeld een van de quotes op deze pagina). De method returnt een double: het gemiddelde aantal letters per woord.

- (xiii) Maak een method *IsSchrikkel* die gegeven een int parameter **jaartal** een bool teruggeeft: **true** als het een schrikkeljaar is, **false** anders. Programmeer de berekening zelf. Wanneer is een jaar een schrikkeljaar? Als het deelbaar is door 4 (2020 is een schrikkeljaar), maar niet wanneer het deelbaar is door 100 (2100 is geen schrikkeljaar), behalve een keer in de 400 jaar (2000 was wél een schrikkeljaar). Voor de liefhebber: er is al heel wat geschreven over kalenderberekeningen, zie bijvoorbeeld [The Calender FAQ](#)
- (xiv) Maak een method *aantalDagenInFebruari* die gegeven een jaartal (int) het aantal dagen in februari in dat jaar.
- (xv) Maak een method *aantalDagenInJaar* die gegeven een jaartal (int) het aantal dagen in dat jaar teruggeeft.



## 43 Heb ik het allemaal goed gedaan?

Het is goed om zelf de volgende zaken te bekijken en/of testen:

- Er zijn géén variabelen buiten een method gedefinieerd: er zijn alleen maar lokale variabelen (en namen van lokale variabelen beginnen met een **kleine** letter).
- De **Main**-method is de enige plek waar `Console.ReadLine()` en `Console.WriteLine()` voorkomt!
- Roep elke methode meerdere keren aan! Bijvoorbeeld een methode die 2 strings als parameter heeft roep je minstens 1 keer aan met 2 strings van verschillende lengte, maar ook met 2 strings van gelijke lengte. Controleer het antwoord!
- De **happy flow** testen we tijdens het programmeren meestal wel. Probeer ook uitzonderingssituaties te testen! Geef eens een lege string mee aan een methode die een string-parameter heeft, en een methode die een List of een array verwacht wordt aangeroepen met een lege List of array.

## 44 Eerste Hulp Bij Vasthangen

Hoe pulk je letters uit een string? Hoe tel je het aantal letters in een string? Hoe kijk je of een string als deel van een andere string voorkomt? + [string manipulatie \(in Toolbox basis\)](#)

## 45 Training - Galgje

Programmeer het spel Galgje waarbij de gebruiker een vooraf ingesteld woord moet raden.

De gebruiker mag 10 keer een letter raden. Heeft hij daarna het woord goed dan heeft hij gewonnen, anders wint de computer.

Ontwerpideeën: houdt in een (string-) variabele het te raden woord bij en houdt in een andere variabele bij wat de gebruiker allemaal heeft geraden.

### 45.1 Uitbreidingen

1. Maak een PictureBox waarin je middels een aantal plaatjes van een galgje en poppetje het aantal fouten laat zien. Laad dynamisch elke keer het volgende plaatje in als er een fout wordt gemaakt. Kijk eens naar de methode `PictureBox.Load()`.
2. Laat zien welke letters geweest zijn. Je kunt bijv. alle letters laten zien in een zwarte kleur. Letters die goed geraadde zijn maak je groen, letters die fout geraadde zijn maak je rood.
3. Uitbreiding op feature 2: maak de (zwarte dus niet gekozen) letters clickable. Als je er op clickt wordt dezelfde event handler als onder de knop Indienen aangeroepen.
4. Zorg dat als je een letter intypt deze automatisch wordt “ingediend”, dus dat je niet ook nog eens op de knop moet clicken. Maak tevens na het indienen van een letter de TextBox leeg en zorg dat deze altijd de focus meteen weer heeft zodat je meteen een nieuwe letter kan intypen.
5. Zorg ervoor dat je alleen letters kan intypen in de TextBox. Accepteer tevens hoofdletters. Er zijn meerdere manieren om dit op te lossen. Kijk bijv. eens naar deze [StackOverflow](#) bron waar een vergelijkbaar probleem wordt besproken of gebruik een variant op de `TextBox` nl. `MaskedTextBox`. Type je geen letter in, bijv. een cijfer, dan speelt er een errorgeluidje af.
6. Onthoud de score(s) van een bepaalde speler en schrijf deze weg naar een highscore bestand. Laat hiervoor de speler zich eerst identificeren. Maak een highscore overzicht scherm waarin alle namen met de gemiddelde aantal benodigde pogingen worden getoond.
7. Uitbreiding op 6: Maak een login mogelijkheid waar je middels een wachtwoord jezelf moet identificeren zodat niet andere spelers met jouw highscore aan de haal gaan. Sla dit wachtwoord gehashed op in hetzelfde bestand waar ook de namen en highscores staan. Gebruik bijv. MD5 hiervoor. Elke regel in het bestand bestaat uit “ ”. Omdat MD5 niet veilig is, probeer een ander hashing algoritme te vinden.

## 46 Training: Method cijfers naar woorden

Maak een methode `CijfersNaarWoorden` die als parameter een `String` heeft en als return-waarde ook een `String`.

Schrijf een programma die de tekst van de gebruiker oppakt en daar alle cijfers vervangt door woorden. Dus als de gebruiker

```
Ik ben 20 jaar
```

invoert dan is de uitvoer

```
Ik ben tweekul jaar.
```

Voorwaarden: + De functionaliteit die één cijfer verandert moet in een aparte methode komen (en die methode mag GEEN interactie met een eventuele user interface hebben). Noem deze methode `VeranderCijfer`.

## 47 Training - Boter, kaas en eieren tegen de computer

Bedenk eerst een algoritme (dmv Flowchart) voor boter-kaas-en-eieren. Dat algoritme kun je zo inrichten zodat iemand tegen de computer boter-kaas-en-eieren kan spelen. Denk van tevoren na over hoe je dit gaat programmeren: welke variabelen heb je nodig en wanneer ga je die variabelen een waarde geven?



## 48 Training - Array - Televisie

We gaan een televisieapplicatie maken. In deze televisieapplicatie moet je kunnen zien of de TV aan of uit staat en zien welke zenders bekeken kunnen worden.

### 48.1 Deel 1 - De televisie aan en uit

De eerste stap is om een form met twee radiobuttons met de waarde “aan” en “uit” (rbAan en rbOff) te maken. Daarnaast een picturebox met een televisie en daarboven op een label (tbStatus). Wanneer de radiobutton “aan” staat moet de label naar “Aan” gaan en wanneer de radiobutton “uit” staat moet de label naar “Uit” gaan. Deze status onthoud je ergens met een boolean.



Figure 48.1: Deel 1

## 48.2 Deel 2 - Zenders toevoegen

Als je televisie kijkt is het belangrijk om te weten welke zenders je kunt bekijken. De televisie kan maximaal 100 zenders hebben. Omdat het maximale zenders vaststaat ga je gebruik maken van een array van strings. Hiervoor ga je een lege array aanmaken met ruimte voor 100 items. Zodoende heb je genoeg plek voor extra zenders in de toekomst. Hoe doe je dit?

```
string[] zenders = new string[100];
```

Je kunt hier nu ook zenders aan toe voegen:

```
string[] zenders = new string[100];  
zenders[0] = "NPO1";  
zenders[1] = "NPO2";  
zenders[2] = "NPO3";
```

Je ziet dat er wordt begonnen op positie 0 (zenders[0]).

## 48.3 Deel 3 - Zenders laten zien in een listbox

De volgende stap is om dit te laten zien met een listbox in je applicatie. Gebruik het voorbeeld hieronder en voeg een listbox lbZenders toe en laat de zenders uit de array zien.



Figure 48.2: Deel 2

## 48.4 Deel 4 - Zenders toevoegen via een textbox

Als je dit hebt gedaan is het nu tijd om er nog wat functionaliteit aan toe te voegen. De gebruiker van de applicatie moet zenders kunnen toevoegen. Hiervoor maak je eerst een textbox (tbZender) en button (btToevoegen) aan.



Figure 48.3: Deel 3

Daarna maak je de methode **public void VoegZenderToe(string zender)** aan. Hiermee kun je een zender kunt toevoegen aan je array waar alle zenders in opgeslagen gaan worden. Denk na over hoe je ervoor zorgt dat de zender op de juiste positie in de array opgeslagen kan worden.

## 48.5 Deel 5 - Zappen en huidige zender laten zien

Als dit is gelukt maak je nog twee extra buttons aan waarmee je een zender verder (btVerder) en terug (btTerug) kan “zappen” en een label (lbZender) waarin je laat zien op welke zender de televisie staat.

Maak **twee methodes** waarmee je een zender verder en terug kunt zappen. Het is handig om hiervoor een variabele `int` aan te maken die `huidigeZender` heet. Hier houd je bij welke zender er nu op staat. Maak ook een methode die **public string HaalHuidigeZenderOp()** heet. Deze laat de naam van de huidige zender zien.





Figure 48.4: Deel 4

#### 48.5.1 Uitbreiding

Maak nu de grafische kant van de applicatie waarin je bovenstaande functionaliteit aan de GUI gaat koppelen. Pak als televisieprogramma bijvoorbeeld een plaatje of zo. Elke zender laat een ander plaatje zien.

## 49 Training - Array - Netflix and random

Je zit met je (aanstaande?) geliefde op jouw bed in je kekke studentenkamer. Samen met je geliefde wil je op Netflix een film gaan kijken. Aangezien je alleen maar fan bent van Avant-garde films en je geliefde ook een plezier wilt doen, heb jij al jouw eigen gemaakte applicatie klaarliggen. Deze applicatie gaat jullie helpen met het kiezen van de film.

Maak deze applicatie waar er tussen vijf verschillende genres door middel van radiobuttons gekozen kan worden. Wanneer er op een radiobutton geklikt wordt moet er een willekeurige film uit dat genre zichtbaar worden.

Deze films komen uit vijf verschillende arrays (elke array heeft 1 genre).

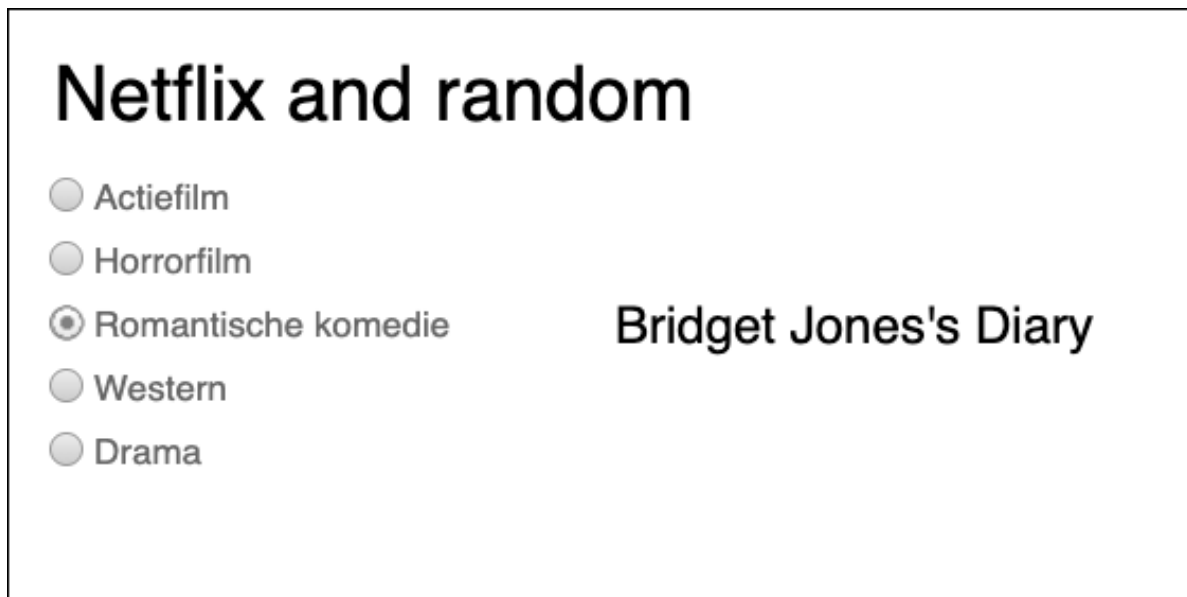


Figure 49.1: Netflix and random

### 49.1 Uitbreidingen

- Wanneer er een genre wordt gekozen: laat een plaatje van het genre zien

- Voor de echte diehards: koppel er de Movie API aan. <https://developers.themoviedb.org/3/getting-started/introduction>

## 50 Training - Array - Fruitautomaat

Maak een fruitautomaat-applicatie. Maak deze zo dat je met een druk op de knop drie verschillende plaatjes te zien krijgt. Zorg er voor dat je de code zo maakt dat deze makkelijk uit te breiden is van bijvoorbeeld 3 naar meerdere “slots” (plaatjes).

*Tip: Je gaat de Random class nodig hebben bij de implementatie. Denk na waar je een array zou kunnen gebruiken. Zonder een array kan het ook, maar dat is niet het doel van deze opdracht natuurlijk. Bovendien is de code dan niet uitbreidbaar naar een willekeurig aantal “slots”.*

## 51 Training - List - Prijzen toevoegen

Voeg een aantal prijzen toe in onderstaand stukje code, op de plek waar het commentaar staat. Herschrijf de for-loop naar een foreach herhalingsstructuur. Controleer je nieuwe code in een C# project.

```
List<double> prijzen = new List<double>();  
double totaalPrijs = 0.0;  
// Hier moet je nog een aantal prijzen (doubles) aan de List toevoegen.  
  
for(int i = 0; i <= prijzen.Count; i++)  
{  
    totaalPrijs = totaalPrijs + prijzen[i];  
}  
  
Console.Out.WriteLine("De totaalprijs is " + totaalPrijs.ToString("C"));
```

## 52 Training - List - Ticketmeester

Je werkt als software developer bij het bekende bedrijf “Ticketmeester”. Bij Ticketmeester worden online concerttickets verkocht voor uitlopende concerten.

Maak een applicatie waar de koper tickets kan kopen voor een concert. Laat in een overzicht zien welke tickets er zijn verkocht. Deze tickets moeten op alfabet gesorteerd zijn. Maak gebruik van een List waar strings inzitten.

The screenshot shows a web application titled "Ticketmeester". It is divided into two main sections: "Bestel ticket" (Order ticket) on the left and "Bestelde tickets:" (Ordered tickets:) on the right.

**Bestel ticket section:**

- Kies concert:** A list of three concert options: "K3 - Kleuren regenboog tour", "Rihanna - Diamonds are 4evar", and "Down the Code Hole". The third option is selected and highlighted in blue.
- Aantal tickets:** A text input field containing the number "4".
- Bestel button:** A dark grey button labeled "Bestel" (Order).

**Bestelde tickets: section:**

- A list of four ordered tickets, each in a separate box:
  - Down the Code Hole - 4 tickets
  - K3 - Kleurenregenboog tour - 10 tickets
  - K3 - Kleurenregenboog tour - 10 tickets
  - Snoop Dogg - Green is the new Black - 2 tickets

Figure 52.1: Ticketmeester

### 52.1 Uitbreidingen

- Voeg er een knop aan toe waar de gebruiker zijn tickets kan sorteren op alfabet ascending en descending.
- Voeg een extra veld toe zodat de gebruiker kan kiezen voor een sta- of zitplaats.

- Voor de diehards: maak van de strings een losse ticket object. Hierdoor kan je meer gegevens gebruiken.

## 52.2 Bronnen

- List Sort <https://www.dotnetperls.com/sort>

## 53 Training - Listmethodes - Digitale muziekcollectie

Maak een applicatie om je digitale muziekcollectie bij te houden. Laat de gebruiker van je app een album toevoegen aan je collectie. Gebruik daar een List voor. Geef alle albums weer in een ListBox.

De gebruiker moet ook albums kunnen zoeken in de collectie. Maak daar functionaliteit voor. Ook moeten er albums verwijderd kunnen worden. Hiervoor moet de gebruiker het regelnummer van het album doorgeven wat er verwijderd dient te worden. Bij het toevoegen moet er worden gecontroleerd of het album niet al in de collectie zit.

*Tip: Gebruik de Contains en IndexOf methodes van Lists.*



## 54 Training - Listmethodes - Raad het getal

We gaan een spel maken waar de speler moet gokken welke getallen er in willekeurig getrokken reeks voorkomen. Laat op voorhand de speler kiezen tussen welke grenzen de getallen mogen komen te liggen en geef hem of haar de mogelijkheid om het aantal getrokken getallen in te stellen. Het maximum minus het minimum moet twee maal zo groot zijn als het aantal getallen dat getrokken gaat worden. Anders is het spelletje te gemakkelijk.

Wanneer het spelletje wordt gestart met de door de speler ingevoerde spelparameters kan er geraden worden. Er zal moeten worden gekeken of het getal dat de speler ingeeft in de lijst van willekeurige getallen voorkomt. Zo ja, dan moet deze uit de lijst worden verwijderd. Zo niet, dan moet de applicatie dit laten weten.

Wanneer alle getallen uit de lijst zijn verwijderd moet er een melding door de applicatie worden gegeven dat de speler heeft gewonnen. Als je jezelf helemaal te buiten wilt gaan mag je de speler ook nog een aantal levens geven. Bij elke fout die de gebruiker maakt gaat er een leven van het totaal af. Wanneer het aantal levens op 0 staat heeft de speler verloren en wordt er Game Over weergegeven door het spel.

Probeer het spel er grafisch leuk uit te laten zien.

## 55 Training - Listmethodes - Vlaggen van de wereld

Maak een applicatie waarmee een gebruiker vlaggen kan maken. Een vlag bestaat uit drie kleuren. Deze drie moeten in een array worden opgeslagen. Wanneer een gebruiker drie kleuren heeft gekozen (een voor de bovenste balk, een voor de middelste en een voor de onderste) kan hij de vlag vervolgens invoeren in het systeem. De vlaggen worden opgeslagen in een List. Geef elke vlag ook een naam (van een fictief land). Dubbele landen mogen niet voorkomen in de lijst van vlaggen.

De gebruiker moet ook een vlag kunnen selecteren en bekijken. Wanneer er een vlag is geselecteerd worden de kleuren van de vlag afgebeeld op het scherm.

Zorg er voor dat de gebruikers interface zo intuïtief mogelijk is opgezet. Het moet bijvoorbeeld ook op een mobile device te bedienen zijn.

## 56 Training - Array - Woordvervormer

We gaan een woordvervormer maken. Maak een applicatie waar de gebruiker een woord kan invoeren. Tevens heeft de gebruiker de keus tussen de vervormopties “Spiegelen”, “Plussen” en “Onevenen”. Als de gebruiker op de knop met de tekst “Vervorm!” klikt moet zijn ingevoerde woord vervormd worden en moet hij het resultaat ervan zien. Hier volgt de uitleg van de verschillende opties in vervormen.

- Spiegelen: Het ingevoerde woord moet omgedraaid worden (“Hallo” wordt “ollaH”).
- Plussen: Een a wordt een b, een b wordt een c, ..., z wordt een a (“Hallo” wordt “Ibmmp”).
- Onevenen: Pak alleen de letters op oneven plekken in het woord (“Hallo” wordt “Hlo”).

Er geldt nog een extra spelregel! Je mag bij Spiegelen geen gebruik maken van de *Reverse* of andere string-gerelateerde methodes. Anders is het geen uitdaging natuurlijk ;-)

### 56.1 Tips

- Zie de strings als array's en behandel ze als zodanig. Je kunt in C# van en naar `char[]` converteren. Vergeet de ASCII tabel niet voor het plussen.
- Maak een class aan voor je vervormer. Wat voor eigenschappen en acties zouden daar handig voor zijn? Zie de Klassen module voor meer informatie over dit onderwerp. Als je daar nog niet aan toe bent kun je dit later altijd nog verbeteren.

## 57 How To ... (problem oriented)

Hoe kan ik in een WinForm-app met meerdere Forms vanuit het ene Form het andere openen?  
Het antwoord op deze en nog veel meer vragen: [How to...](#)

### 57.1 WinForms

#### 57.1.1 Meerdere Forms in een app

Als je in een WinForms-app de `Program.cs` opent zie je soortgelijke code als:

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new MainForm());
}
```

waarbij `MainForm` de naam is van je Form. Bij het opstarten van je app wordt met `new MainForm` een form aangemaakt. Vanuit dit form kun je het mainform zelf (`this`) hidden maken. Vervolgens een ander form aanmaken en daar de method `ShowDialog()` van aanroepen:

```
this.Hide();
SecondForm otherForm = new SecondForm();
otherForm.ShowDialog();
```

#### 57.1.2 Informatie doorgeven tussen Forms

In deze video wordt het uitgelegd: [Fox Learn](#)

## 57.2 Bron

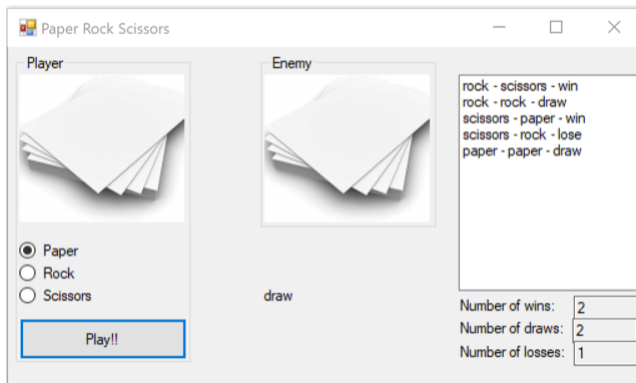
[how-to-open-a-second-form](#)

## 58 Training Rock Paper Scissors

Maak het spel “paper rock scissors” (Steen Papier Schaar).

Dat kan in een WinForm-app (zoals in de omschrijving hieronder) maar mag ook op de Console of nog anders.

Doormiddel van radiobuttons kan je kiezen of je kiest voor paper, rock of scissors. Wanneer je erna op de “Play” button klikt wordt er automatisch voor de tegenstander een paper, rock of scissor gekozen. Daarna wordt er gekeken wie de winnaar is. De uitslagen (draw, win of lose) worden in een listbox gezet en wordt bijgehouden hoe vaak je draw, win of lose hebt. Dit zet je in textboxes.

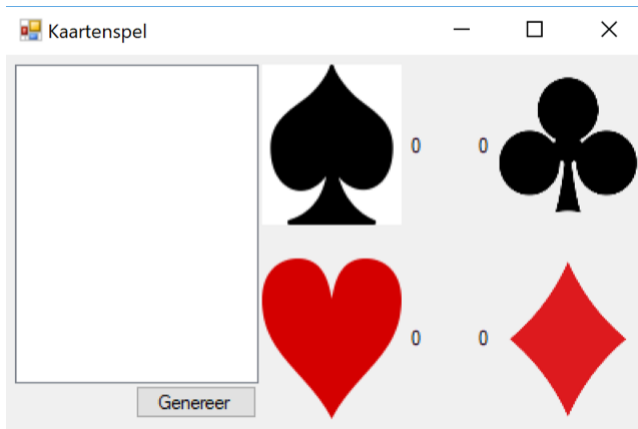


## 59 Training kaartspel

Concepten: arrays, methods

Dat kan in een WinForm-app (zoals in de omschrijving hieronder) maar mag ook op de Console of nog anders.

Maak een applicatie waarin je een listbox en een genereerbutton hebt gezet. Wanneer je op de genereerbutton klikt moet er een “kaart” (een string met bijv. H3-> harten 3) uit het kaartspel willekeurig worden gegenereerd. Deze kaart moet in de listbox terecht komen. Maak hiervoor een method “GenereerKaart” aan. Deze method moet een string met de kaartcode (bijv. H3-> harten drie, S2-> schoppen 2, SK->schoppen koning) terugsturen. Breid het spel uit door tevens te tellen hoeveel van elk type (schoppen, harten, klaveren of ruiten) gegenereerd zijn.



## 60 Training van getal naar woord

Maak een applicatie die gehele getallen in woorden af kan drukken.

### 60.1 Uitbreidingsmogelijkheden

1. geef gebruiker optie om “een” aan of uit te zetten (b.v. eenduizend/duizend)
2. geef gebruiker een optie om te wisselen tussen “duizendtweehonderd” en “twaalfhonderd”
3. maak het mogelijk voor meerdere talen: NL, EN, FR, etc.

Natuurlijk maak je nette en onderhoudbare code. Om dit te bereiken breid je je programma in kleine stappen (cijfers, tientallen, honderdtallen, etc.) uit. Vraag feedback voor iedere uitbreiding die je maakt.



# 61 Training Blackjack (light)

Maak het kaartenspel [blackjack](#). Het doel van het spel is om de bank (dealer) te verslaan. Hierbij moet men proberen dicht bij de 21 punten te komen dan de bank. Als de speler boven de 21 punten uitkomst heeft hij verloren, ongeacht wat de bank heeft. Maak het spel eerst zo dat er 1 bank is en 1 speler.

## 61.1 Regels

- de bank trekt eerst een kaart, daarna de speler. Als speler kan je niet de kaarten van de bank zien.
- wanneer de speler de kaart trekt, mag de speler kiezen om nog een kaart te trekken of stoppen. Wanneer de speler stopt, mag de bank kaarten blijven trekken totdat de bank “het goed vindt” (wat dit is mag je zelf bepalen en programmeren).
- als speler moet je zo dicht mogelijk in de buurt van 21 moeten komen. Als de speler boven de 21 punten uitkomst heeft hij verloren, ongeacht wat de bank heeft.
- gebruik een kaartenset van 52 kaarten

## 61.2 Puntentelling

- boer, vrouw en heer zijn 10 punten waard
- 2 - 10 kaarten hebben de waarde die zij aangeven
- de aas is 1 of 11 punten waard. Afhankelijk welke waarde het dichtst de 21 benadert.

## 61.3 Uitbreidingen

- meer spelers
- meer kaartensets
- breid dit uit met de regels van blackjack (verdubbelen, splitsen, enz..)
- scores bijhouden
- visuele weergave met “echte kaarten”

## 62 Summary

Basics of programming.

## References