

Module PRC-I

Programmeren in C#

1 Over dit document

Blok: B
Opleiding: Team ICO / Software Developer / Breda
Vak(ken): WIN
Weken 1 t/m 10

1.1 Versiebeheer

Versie	Datum	Auteur	Aanpassingen
1.0	11-01-2021	C.Wagtmans	

2 Inhoud

1	Over dit document	2
1.1	Versiebeheer	2
3	Over deze module	7
3.1	Doelstelling en relevantie	7
3.2	Samenhang met andere onderdelen.....	7
3.3	Materiaal- en middelenlijst.....	7
3.4	Feedback-momenten.....	7
3.5	Samenhang met de ICO-V	7
4	Studiewijzer	8
5	Feedback-momenten.....	9
5.1	Feedback-moment 1.....	9
5.2	Feedback-moment 2.....	9
5.3	Leeruitkomsten	10
6	Voor de docent	14
7	Kwaliteit en verbeteracties.....	15
8	Kennismaken met C#	16
8.1	Inleiding	16
8.2	Visual Studio Installatie	16
8.3	Lesopbouw	17
8.4	Voortgangsmeting.....	18
9	Werken met Visual Studio	19
9.1	Opdracht – Project aanmaken.....	21
9.2	De IDE	23
10	Applicatie - Hello Developer Land	26
10.1	Aanmaken project.....	26
10.2	Opdracht – Hello World	26
10.3	De code lezen.....	28
10.4	Class, Object, Property en Method.....	29
10.5	Conventies.....	32
10.6	Opdracht – Hello Developer Land	33
10.7	Opdracht - IntelliSense	34
10.8	Opdracht - Properties	35
10.9	Interactie met de gebruiker	36
10.10	Variabelen.....	36
10.10.1	Plaatsing van variabelen.....	37
10.10.2	Value type variabele	37
10.10.3	Reference type variable.....	37
10.11	Eindopdracht – Interactie met de gebruiker	39

10.12	Wat heb je geleerd?	41
11	Console Applicatie – Rekenen	42
11.1	Bits en Bytes.....	42
11.2	Binair.....	43
11.3	Datatypes.....	44
11.4	Omzetten datatypen - getal < - > string	46
11.5	Omzetten datatypen – getal < - > getal	47
11.6	Operatoren.....	48
11.7	Opdracht - Rekenmachine.....	49
11.8	Opdracht - BTW berekenen.....	50
11.9	Opdracht - BTW vrije dagen	51
11.10	Eindopdracht - Kassa	51
11.11	Wat heb je geleerd?	51
12	Windows Form - Rekenmachine	52
12.1	Controls	52
12.2	Opdracht - Rekenmachine UI	53
12.3	Opdracht - Events.....	57
12.4	Opdracht – Foutafhandeling	60
12.5	Opdracht – Personaliseren met PictureBox	63
12.6	Eindopdracht - BTW	64
12.7	Wat heb je geleerd?	64
13	Introductie Methoden	65
13.1	Methoden die iets doen	65
13.2	Methoden die een vraag beantwoorden.....	65
13.3	Aanmaken van een methode.....	66
13.4	Aanroepen Methode	67
13.5	Opdracht - RekenMachine aanpassen.....	69
13.6	Opdracht - Het nut van methoden	70
13.7	Eindopdracht Methoden	73
13.8	Wat heb je geleerd.	73
14	Date Time Variabele	74
14.1	Opdracht - DateTime variabele	74
14.2	Wat heb je geleerd.	76
15	Selecties - If..Else	77
15.1	If....Else	77
15.2	If...Else - Doorlopen van de code.....	81
15.3	Opdracht – Aanpassen Applicatie LeeftijdCheck	83
15.4	Opdracht – Actiedagen bij Kubus.Com	84
15.5	Wat heb je geleerd?	84
16	Geneste If Statements	85

16.1	Opdracht – Geneste IF Statements.....	86
16.2	Eindopdracht – Foutafhandeling RekenMachine	91
16.3	Wat heb je geleerd?	94
17	Selecties - Switch	95
17.1	Opdracht – Kassa.....	98
17.2	Opdracht – Welke les heb ik vandaag?.....	98
17.3	Eindopdracht – Welke les heb ik vandaag?	98
17.4	Wat heb je geleerd?	99
18	Comments	99
19	Loops (herhalingen).....	100
19.1	While loop.....	101
19.2	Opdracht - While Loop.....	104
19.3	Do ... While.....	105
19.4	Opdracht – Do...While Loop.....	107
19.5	For Loop	108
19.6	Eindopdracht - For Loop	110
19.7	Wat heb je geleerd?	110
20	Gegevens, dataset en Informatie	111
21	Array	113
21.1	Het maken van een Array.....	114
21.2	Opdracht – Aanmaken van een Array	115
21.3	Opdracht - Ophalen van gegevens uit een Array	116
21.4	Opdracht - Arrays manipuleren	116
21.4.1	Sorteren van de gegevens.....	117
21.4.2	Gegevens toevoegen aan een array	117
21.5	Loops en arrays	119
21.6	Opdracht - Loops en Arrays	121
21.7	Opdracht - Multi Dimensional Array.....	122
21.8	Opdracht - Control DataGridView.....	125
21.8.1	Werking van de loop in de code	129
21.9	Eindopdracht – Dessert Applicatie	131
21.10	Wat heb je geleerd?	131
22	Extra Opdrachten	132
22.1	Hoofdstuk 10 - Met je gezin naar Developer Land in december.....	132
22.2	Hoofdstuk 11 – Werken met operatoren	132
22.2.1	Applicatie – Temperatuur omrekenen.....	132
22.2.2	Applicatie – Omrekenen TV grootte.....	132
22.2.3	Applicatie – Entree prijs berekenen	132
22.3	Hoofdstuk 12 – Controls, Events en foutafhandeling	133
22.3.1	Wisselkoers	133

22.3.2	Pythagoras.....	133
22.4	Hoofdstuk 14 – DateTime Variabele.....	135
22.4.1	Vakantie App	135
22.4.2	ToetsInfo Applicatie	135
22.5	Hoofdstuk 15 – Selecties.....	136
22.5.1	DatatypeApp	136
22.6	Hoofdstuk 16, 17 – Geneste Selecties	137
22.6.1	Applicatie – ToetsScore app.....	137
22.6.2	Applicatie – NakijkApp	137
22.6.3	DatatypeApp 2.0 – (pittige opdracht)	138
22.6.4	Nieuwe Actiedagen bij Kubus.Com	139
22.6.5	Selecties Switch - Palatia Village Cocktails	139
22.7	Hoofdstuk 19 – Loops.....	140
22.7.1	Getallen raden	140
22.8	Hoofdstuk 21 – DataGridView en Arrays.....	140
22.8.1	Gemiddeld gebruik van een auto	140
22.8.2	Puppyhulp.....	140
23	Bijlage 1 – Naamgevingsconventies	143
24	Bijlage 2 - Uitwerking - Filmlijst.....	144
25	Bijlage 2 - Index.....	147

3 Over deze module

3.1 Doelstelling en relevantie

Aan het eind van deze module ben je in staat om een eenvoudige interactieve Windows Forms Applicatie te bouwen waarbij je gebruik maakt van de programmeeromgeving in Visual Studio. De applicatie voldoet aan de wensen van de opdrachtgever en je maakt gebruik van de naamgevingsconventies welke zijn opgesteld door Microsoft zodat het maken van fouten tot een minimum wordt beperkt. In je applicatie maak je gebruik van de juiste technieken waardoor er geen onnodige code en geheugen wordt gebruikt.

3.2 Samenhang met andere onderdelen

In deze module ga je technieken zoals variabelen en loops toepassen welke je al eerder hebt gebruikt. Je krijgt nogmaals de uitleg hoe deze technieken werken en hoe deze kunnen worden gebruikt in je code met de daarbij behorende syntax. Je maakt gebruik van je GitHub account om code te delen en versiebeheer toe te passen.

3.3 Materiaal- en middelenlijst

Voor deze module maak je gebruik van je laptop met daarop Microsoft Visual Studio, Xampp, Github-client. De module bevat de theorie welke klassikaal wordt behandeld en de opdrachten welke uitgevoerd moeten worden. Indien gewenst kun je het boek Leren Programmeren in C# aanschaffen waarin dezelfde onderwerpen worden behandeld maar met een iets andere uitleg en oefeningen. Naast dit boek kun je uiteraard gebruik maken van het internet maar let op dat je gebruik maakt van de technieken zoals ze in de les worden uitgelegd.

3.4 Feedback-momenten

Iedere les bestaat uit een theoretische uitleg en opdrachten waarmee je kunt oefenen. Opdrachten welke je niet in de les hebt kunnen maken dienen thuis afgemaakt te worden. Aan het begin van de volgende les wordt altijd gestart met een feedback moment waarin de vorige les nog kort wordt besproken. In dit feedbackmoment ben je in staat om vragen te stellen over het eventuele huiswerk.

In week 5 en 9 staat een toets op de planning waarmee wordt gekeken hoe goed je de behandelde stof tot je hebt genomen. De toets zal bestaan uit een aantal vragen en opdrachten waarin (een deel van) de onderwerpen uit de voorgaande weken aan bod komen. In week 6 en 10 wordt de toets besproken zodat je een indicatie hebt wat jouw voortgang is voor deze module.

3.5 Samenhang met de ICO-V

De volgende activiteiten uit het ICO-V model, die je in het blokboek B terug kunt vinden, komen in deze module aan bod.



4 Studiewijzer

Week	Onderwerpen	Opdrachten	Bronnen / websites / literatuur
1	IDE Conventies Aanmaken Project Code lezen Console Applicatie	Kennismaking met Visual Studio Hello Developer Land IntelliSense Properties	Hoofdstuk 8,9,10 Filmpjes: bit.ly/PRCI-1A-Uitleg bit.ly/PRCI-1A-AanmakenProject
1b	Variabelen	Interactie met de gebruiker	Hoofdstuk 10
2a	Datatypen Variabelen Operatoren	Rekenen BTW Berekenen	Hoofdstuk 11 Filmpjes: bit.ly/PRCI-OmzettenDatatypen
2b		BTW Vrije Dagen Eindopdracht Kassa	Hoofdstuk 11
3a	Windows Forms Controls Events Foutafhandeling	Rekenmachine UI Personaliseren BTW	Hoofdstuk 12 Filmpjes: bit.ly/PRCI-EventsEnFoutafhandeling
3b		Picturebox Eindopdracht BTW berekenen.	Hoofdstuk 12
4a	Methoden	Rekenmachine aanpassen	Hoofdstuk 13
4b		Het nut van Methoden Eindopdracht Methoden	Hoofdstuk 13
5a	DateTime Variabele Selecties If Else	DateTime variabele Aanpassen Applicatie leeftijdCheck. Actiedagen bij kubus.Com	Hoofdstuk 14,15
5b	Geneste If Else	Geneste IF Statements Nieuwe actiedagen bij kubus.Com Eindopdracht Foutafhandeling RekenMachine	Hoofdstuk 16
6a	Selecties – Switch Comments	Kassa Welke les heb ik vandaag? Eindopdracht Welke les heb ik vandaag?	Hoofdstuk 17,18
6b	Feedback moment	Toets	
7a	Feedback moment	Bespreken	
7b	While Loop Do While For Loop	Opdracht While Loop Opdracht Do While loop Eindopdracht For Loop	Hoofdstuk 19
8a	Gegevens, datasets en informatie Array	Aanmaken van een Array Ophalen van gegevens Array manipuleren	Hoofdstuk 20,21
8b	Loops en Arrays	Loops en Array	Hoofdstuk 21
9a	Multidimensionale Array DataGridView	Eindopdracht Dessert Applicatie	Hoofdstuk 21
9b	Feedback moment	Voorbereiding	

10.1	Feedback	Toets	
10.2	Reflectie Voorbereiding PRC-II		

5 Feedback-momenten

De feedback-momenten gaan in je portfolio zodat je voortgang duidelijk te zien is.

5.1 Feedback-moment 1

Week: 6

Vorm: Samengestelde toets

Voorbereiden: Lees de theorie van de weken 1 t/m 5 nogmaals door en bekijk je opdrachten. Bedenk eventueel een andere toepassing waarbij je de technieken uit deze weken toepast.

Meer info: Deze toets bestaat uit;

- Een multiplechoicetest met een aantal praktische vragen.
- Een code van applicaties die niet werken waaruit jij de fouten moet halen zodat ze wel werken.
- Het bouwen van een eenvoudige Windows Forms Applicatie

5.2 Feedback-moment 2

Week: 10

Vorm: Theoretische toets

Voorbereiden: Lees de theorie van de voorgaande weken nogmaals door en bekijk je opdrachten. Doorloop de toets uit week 6 nogmaals. Bedenk of zoek op internet naar een applicatie welke je kunt bouwen waarin je de geleerde technieken uit afgelopen weken toepast.

Meer info: Deze toets bestaat uit;

- Een multiplechoicetest met een aantal praktische vragen.
- Een code van applicaties die niet werken waaruit jij de fouten moet halen zodat ze wel werken.
- Het bouwen van een eenvoudige Windows Forms Applicatie

5.3 Leeruitkomsten

Aspect	Niet voldaan	Voldaan
PRC.01	Je kunt werken met Visual Studio (project maken, bestaand project openen, code delen, enzovoort).	
IDE		Je kunt zonder problemen de Solution Explorer, Properties Venster, Toolbox, Error-List en Data Sources vinden.
Project openen		Je kunt een project openen vanuit de recente projecten lijst of door te browsen naar een project.
Project aanmaken		Je kunt een Windows Forms- of Console applicatie aanmaken en past de juiste benaming toe.
Code Delen		Je weet hoe je code kunt delen via Github of door de projectmap in te pakken en te delen via een zip bestand.
Applicatie Klonen		Je kunt een bestaande applicatie klonen van GitHub of een bestaand project, om zo een nieuwe versie van de applicatie te maken.
Opmaak Applicatie		Je kunt het uiterlijk van zowel de Console als Windows Forms aanpassen met behulp van properties.
PRC.02	Je kunt de basisconcepten (keuzes, herhalingen, variabelen) toepassen in C#.	
Selecties		Je kunt selecties inbouwen in je applicatie door gebruik te maken van het IF...ELSE statement.
Geneste selecties		Je kunt complexe selecties inbouwen in je applicatie door gebruik te maken van geneste IF...ELSE statements.
Switch		Je snapt het concept van het Switch statement en bent in staat om deze te gebruiken indien deze geschikt is voor je applicatie.
Variabelen		Je kunt beschrijven wat een variabele is, waar deze voor gebruikt wordt en gebruiken in je applicatie.
Variabelen		Je weet op welke plek een variabele gedeclareerd moet worden om deze te kunnen gebruiken in je code.
Variabelen		Je snapt het concept van het aanmaken, vullen en legen van een variabele.
Loops		Je snapt het concept van loops en past deze toe indien er een repeterende handeling plaatsvindt in je code.
Loops		Je kent het verschil tussen een For-, While-, Do While en For Each Loop en kunt bepalen welke loop je het beste kunt gebruiken bij het maken van je applicatie.
PRC.03	Je begrijpt hoe de code wordt uitgevoerd (wat de <i>flow</i> is); je begrijpt en gebruikt het concept van event-handlers.	
Uitvoer van de code		Je kunt beschrijven hoe de code van je applicatie wordt uitgevoerd en wat er in de verschillende stappen gebeurt.
Werken met blokken		Je kunt beschrijven hoe het programma omgaat met blokken zoals selecties en loops in combinatie met variabelen.
Event-handlers		Je maakt gebruik van Event-Handlers en schrijft code voor het event, zoals de klik op een knop, om interactie met de gebruiker in te bouwen in je applicatie.

PRC.04	Je kunt een eenvoudig interactief programma maken met WinForms of WPF; je kunt werken met de <i>designer</i> en ook direct vanuit de code <i>properties</i> aanpassen.	
Winforms		Je kunt een Windows Forms applicatie maken met één formulier met daarop verschillende controls.
Designer		Je kunt een visueel aantrekkelijke applicatie bouwen en gebruikt de hulpmiddelen in Visual Studio om de controls netjes uit te lijnen.
Designer		Je ben in staat om de eigenschappen van controls aan te passen met behulp van het properties venster in Visual Studio.
Properties		Je kunt vanuit de code dynamisch een label aanpassen of een textbox vullen met gegevens.
Properties		Je bent in staat om de opmaak van tekst en controls aan te passen met behulp van properties in je code.
PRC.05	Je kent verschillende datatypes, je kunt deze inzetten zoals bedoeld en je kunt waar nodig converteren tussen verschillende datatypes.	
Datatypes		Je kent verschillende datatypes, weet het verschil tussen de datatypes en bent in staat om de juiste keuze te maken voor het gebruik van een datatype in je applicatie.
Datatypes omzetten		Je bent in staat om datatypes om te zetten naar andere datatypes met behulp van methodes welke beschikbaar zijn in de programmeertaal.
Opmaak		Je bent in staat om de opmaak van het datatype aan te passen zodat de gegevens op het scherm voldoen aan de eisen van de klant.
PRC.06	Je kunt zelf een methode aanmaken die verantwoordelijk is voor één handeling.	
Methode		Je kent het verschil tussen methoden die een waarde teruggeven en methoden die iets doen.
Methode Argumenten		Je bent in staat een methode aan te maken waarbij argumenten meegegeven worden.
Methode Return		Je bent in staat een methode te maken welke één of meerdere resultaten teruggeeft.
PRC.07	Je kent verschillende componenten en kunt deze gebruiken (waaronder <i>button</i> , <i>label</i> , <i>textbox</i> en eventueel <i>datagridview</i>).	
Button		Je gebruikt buttons in je applicatie en past de opmaak, naam, opmaak en tekst aan zodat deze past binnen de stijl van de applicatie.
Label		Je gebruikt labels om tekst toe te voegen aan je applicatie of om resultaten te laten zien.
DataGridView		Je bent in staat een DataGridView te gebruiken binnen je applicatie om gegevens uit een array, gegevensbestand of database te tonen.
TextBox		Je gebruikt één of meerdere textboxes in je applicatie voor het gebruik van input door de gebruiker of het weergeven van een resultaat.
DateTimePicker		Je bent in staat om een DateTimePicker in te zetten indien in de applicatie wordt gevraagd om een datum in te geven. Je kent verschillende properties van deze control om specifieke gegevens op te halen.

PRC.08	Je kunt werken met Arrays (aanmaken, loop-en, opvragen, enzovoort); je begrijpt de concepten hierachter.	
Array		Je weet hoe gegevens in een (multi dimensionale) array worden opgeslagen op basis van een index.
Array Aanmaken		Je bent in staat om een (multi dimensionale) Array aan te maken en te vullen met gegevens.
Toevoegen gegevens		Je bent in staat om gegevens toe te voegen aan een (multi dimensionale) array
Verwijderen gegevens		Je bent in staat om gegevens uit een (multi dimensionale) array te verwijderen.
Loop		Je bent in staat om met behulp van een loop gegevens op te halen uit een (multi dimensionale) array.

En daarnaast wordt aandacht besteed aan de leeruitkomsten van het V-model:

Aspect	Niet voldaan	Voldaan
13.04	Je programmeert de applicatie zodat deze aan alle klanteisen voldoet; je integreert assets en/of werkt met een framework. Je werkt efficiënt met je editor en presteert ook onder tijdsdruk.	
Klanteisen Uiterlijk		Het uiterlijk voldoet aan de eisen van de klant zoals in de opdracht beschreven staat.
Klanteisen functionaliteit		Je bent in staat om een applicatie te programmeren waarin alle functionele eisen van de klant aanwezig zijn. Je zorgt ervoor dat deze functionaliteiten altijd werken en dat fouten gemaakt door de klant worden afgevangen.
Controls		Je kiest de juiste controls die het beste aansluiten bij de te bouwen functionaliteit op basis van de eisen van de klant.
IntelliSense		Je maakt effectief gebruik van IntelliSense tijdens het programmeren om fouten te voorkomen en tijd te besparen.
13.05	Tijdens het programmeren gebruik je de gestelde <u>conventies</u> .	
Controls		De namen van controls zijn voorzien van een prefix zodat in de code duidelijk is met welke control gewerkt wordt. De controls hebben een logische naam zodat duidelijk is wat het doel is van de control en de naam wordt geschreven in camelCase .
Variabelen		De variabelen hebben een logische naam zodat duidelijk is waarvoor ze gebruikt worden. De naam wordt geschreven in camelCase.
Methoden		De namen van methoden zijn logisch en bij voorkeur een werkwoord. De namen zijn geschreven in PascalCase.
13.08	Je kunt zelfstandig je code <u>debuggen</u> ; je leest foutmeldingen nauwkeurig en je gaat systematisch te werk om een fout op te sporen. Waar nodig schakel je hulp in.	
Debuggen		Na het bouwen van je applicatie test je grondig alle functionaliteiten binnen de applicatie. Bij foutmeldingen ben je in staat te achterhalen waar deze fout optreedt en deze te verhelpen eventueel met hulp van anderen.
Testen		Je weet hoe je een goede test kunt uitvoeren door een lijst te maken met alle mogelijke situaties die zich voor kunnen doen bij het werken met de applicatie.

13.09	Je kent de <u>documentatie</u> van de programmeertaal waar je mee werkt en je kunt deze raadplegen; je leest nauwkeurig en analytisch. Eventueel gebruik je ook de bijbehorende community om antwoorden te zoeken of vragen te stellen.	
Moduleboek		Je volgt het moduleboek stap voor stap en gebruikt het internet of andere middelen indien iets niet duidelijk is of als aanvullende informatie noodzakelijk is.
Vragen		Je stelt concrete vragen aan de docent of medestudent indien je vastloopt bij het maken van je applicatie.
13.10	Code van een ander kun je vlot lezen en begrijpen; waar nodig maak je schema's en zoek je dingen op. Je onderzoekt de huidige werking en controleert of je geen ongewilde wijzigingen maakt. Je werkt verder volgens de gebruikte conventies.	
Code		Je weet hoe code is opgebouwd en je kunt de flow van je applicatie beschrijven.
Schema's		Je bent in staat een flow-chart te lezen en maken bij complexere delen in je code zoals bijvoorbeeld geneste if statements.
14.02	<u>Tijdens het programmeren test je</u> steeds de gemaakte wijzigingen. Je blijft controleren of je geen ongewilde wijzigingen maakt, of de functionaliteit nu voldoet aan alle eisen en of je veilig geprogrammeerd hebt.	
Breakpoint		Je bent in staat met een breakpoint een deel van de code uit te voeren om fouten te achterhalen.
Testen		Bij het aanpassen van code test je alle functionaliteiten binnen de applicatie om te controleren of de wijziging geen problemen
Error List		Je bent in staat de foutmeldingen in de error list te interpreteren en de code waarnaar de foutmelding verwijst, indien nodig met hulp, aan te passen.

6 Voor de docent

7 Kwaliteit en verbeteracties

Er zijn nog geen verbeteracties uitgevoerd op dit document.

8 Kennismaken met C#

8.1 Inleiding

Vandaag ga je een start maken met het leren programmeren in C#. C# is een door Microsoft ontwikkelde programmeertaal, die wereldwijd wordt gebruikt bij het ontwikkelen van zakelijke applicaties en games. In de komende weken ga je onder andere leren wat functies en methoden zijn en ga je zien dat de basis van het programmeren ook in deze taal hetzelfde is als in Python. De syntax (of grammatica) zal anders zijn en per programmeertaal heb je verschillende regels waar je aan moet houden.

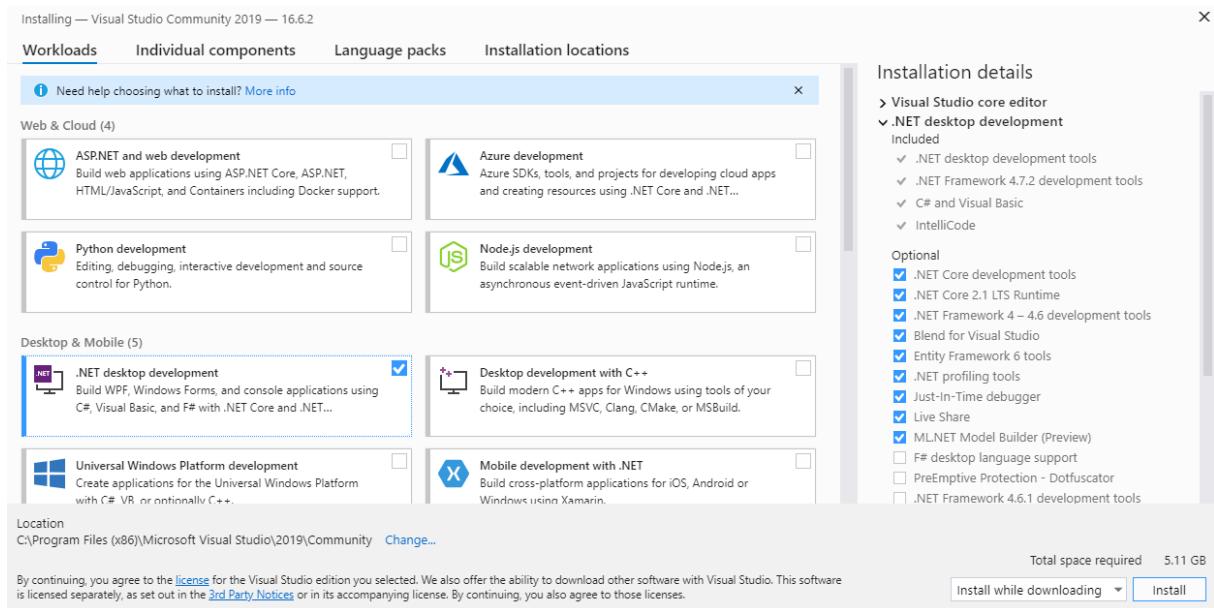
In de komende weken en de rest van je carrière als Software Developer ga je veel gebruik maken van het internet. Het zoeken naar oplossingen voor een bepaald vraagstuk of het verhelpen van een fout in je code zal onderdeel zijn van je dagelijkse werkzaamheden. Je zult al snel merken dat je voor deze vraagstukken op zoek moet gaan naar antwoorden in boeken of het internet. Voor deze lessenserie is er gekozen om het boek 'Leren Programmeren in C#' van BrinkmanICT als basis te hanteren. De lessen zijn zo geschreven dat je dit boek niet hoeft te kopen maar het kan wel dienen als extra naslagwerk indien je hier behoeft aan hebt. Je zult gaan merken dat er verschillende mogelijkheden zijn bij het ontwikkelen van een applicatie en op internet zul je voor het oplossen van een probleem ook verschillende oplossingen vinden. De ene oplossing is beter dan de andere ook al hebben ze hetzelfde resultaat. Het boek van Brinkman houdt vast aan de standaarden die Microsoft heeft gesteld en ook wij volgen deze standaarden in de les.

8.2 Visual Studio Installatie

Voor het ontwikkelen in C# ga je gebruik maken van Visual Studio, een zogenaamde IDE (Integrated Developer Environment). Deze applicatie is ontwikkeld door Microsoft en Visual Studio Community is gratis te gebruiken. Naast de Community versie bestaan er ook de betaalde Pro en Enterprise versies. Deze laatste twee zijn bedoeld voor bedrijven en hebben meer specifieke tools en functionaliteiten voor het gebruik binnen een bedrijf. Je bent met de Community versie in staat om exact dezelfde applicaties te bouwen en deze versie ga je dan ook gebruiken hier op school.

Je start deze les met de installatie van Visual Studio volgens onderstaande stappen.

1. Ga naar <https://visualstudio.microsoft.com/vs/community/> en klik op de 'Download Visual Studio' knop.
2. Voer het gedownloade bestand uit.
3. Op je laptop wordt de Visual Studio Installer geïnstalleerd waarmee je de verschillende workloads kunt selecteren welke je binnen Visual Studio beschikbaar wilt hebben.



Afbeelding 1 - Selecteren van workloads

4. Vink de ‘.NET desktop development’ workload aan en klik op install.
5. Mocht je in de toekomst andere workloads nodig hebben dan kun je deze installer eenvoudig via het startmenu weer openen en andere workloads installeren.
6. De installatie zal enige tijd in beslag nemen aangezien alle bestanden gedownload moeten worden.
7. In de tussentijd kun je alvast doorgaan met het volgende hoofdstukken.

8.3 Lesopbouw

Op je rooster heb je gezien dat er iedere week twee lessen gepland staan van ieder twee uur. De eerste les zal voor een deel bestaan uit een aansturing en evaluatie. In de aansturing kijk je met de docent vooruit en krijg je informatie die je nodig hebt in de lessen van deze week. De evaluatie zal bestaan uit een terugbliek op de les van de vorige week en het bespreken van het huiswerk zodat je altijd op het juiste niveau kunt starten met de les. Na de aansturing en evaluatie ga je zelfstandig aan de slag met de verschillende opdrachten in dit document. De tweede les in de week zal voornamelijk bestaan uit zelfstandige opdrachten.

Gedurende de les loopt de docent rond om je te helpen mocht je vastlopen bij een opdracht. Je gaat merken dat de docent meestal niet de oplossing geeft maar probeert jou zo te sturen dat je zelf de oplossing gaat vinden. Je kunt natuurlijk ook hulp vragen aan een medestudent naast je. Zorg er wel

Het schrijven van code moet je letterlijk ‘in de vingers krijgen’. Dit wordt ook wel “muscle memory” genoemd en je gebruikt dit al dagelijks. Je hoeft niet meer na te denken bij bijvoorbeeld het fietsen of het vinden van de rem bij het autorijden, dat gaat allemaal automatisch. Het beste voorbeeld is het typen op een toetsenbord. Als je veel typt hoef je op den duur niet meer op je toetsenbord te kijken waar alles staat, je vingers weten automatisch de juiste toetsen te vinden.

voor dat je zelf je code typt en neem niet letterlijk de code van een medestudent over. Gebruik ook zo min mogelijk de knip- en plakfunctie en typ zoveel mogelijk zelf ook al typ je het over.

8.4 Voortgangsmeting

Algemene voortgangsmeting

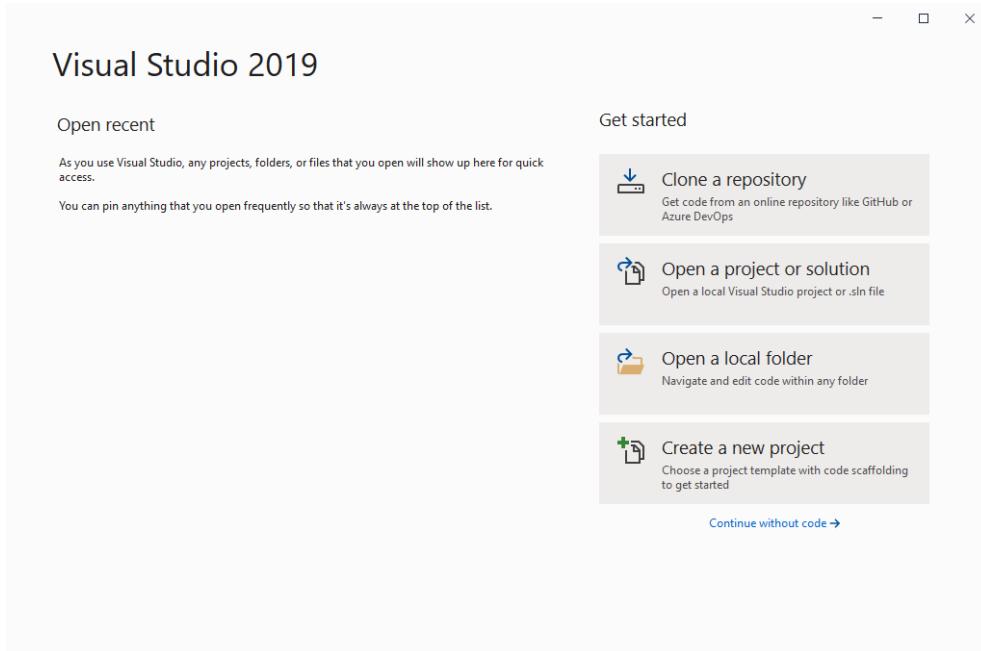
De docent houdt gedurende de lesweken de globale voortgang bij van de groep. Jouw persoonlijke voortgang wordt bijgehouden aan de hand van wat de docent ziet in de les, het ingeleverde huiswerk en de vragen die je stelt. Stel je veel vragen dan ziet de docent dat je met de stof bezig bent wat ook zal resulteren in het beter kunnen maken van de opdrachten. Het niet stellen van vragen geeft dus niet per definitie aan dat je goed bezig bent, alleen als de docent ziet dat het ook goed gaat met de opdrachten. Zeker als je wat meer moeite hebt met de behandelde stof is het dus belangrijk om vragen te stellen. De docent ziet dan dat je goed bezig bent ook al lukken de opdrachten nog niet zo goed.

Tussentijdse voortgangsmeting

In het blokboek kun je terugvinden dat in week 6 en in week 10 een evaluatiemoment gepland staat. Op deze momenten heb je een overleg met de SLB en wordt jouw voortgang besproken. Voorafgaande aan deze evaluatie gaan we meten hoe ver je bent. Deze meting bestaat uit een toets welke bestaat uit verschillende onderdelen welke staan beschreven in hoofdstuk 5. Het resultaat van deze meting in combinatie met de algemene voortgang wordt opgeslagen in jouw dossier en met jou besproken.

9 Werken met Visual Studio

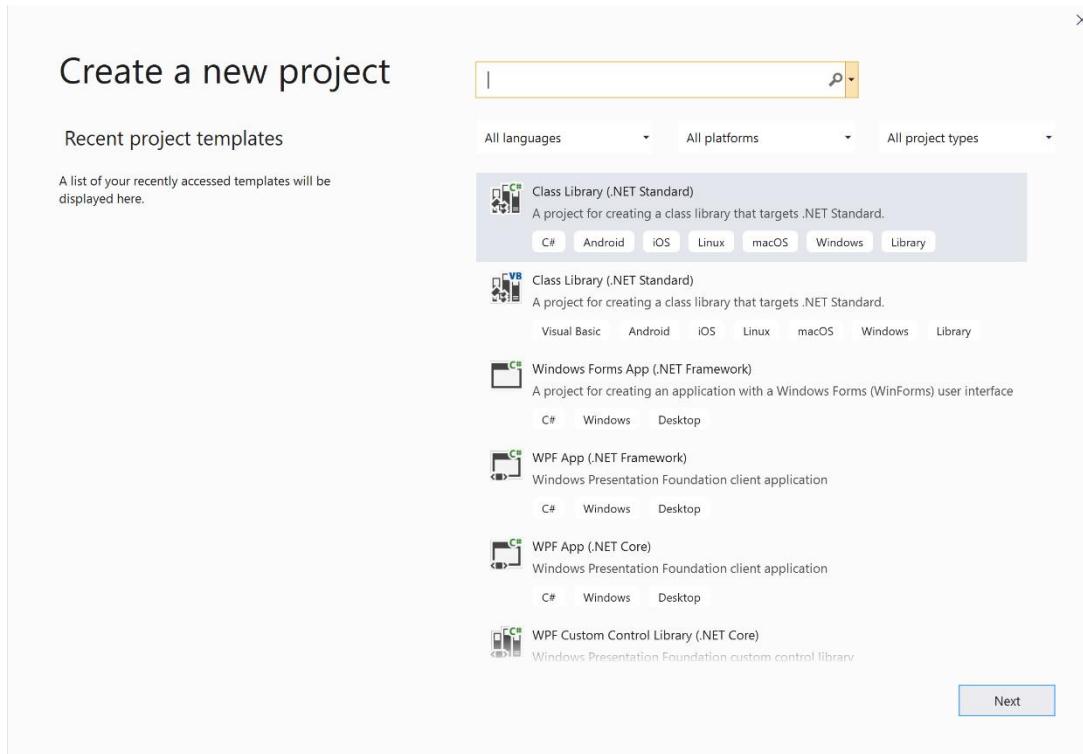
De installatie van Visual Studio zal nu ongeveer klaar zijn of het duurt niet lang meer. In dit hoofdstuk gaan we kijken naar de omgeving waarin je de komende tijd gaat werken. Bij het starten van Visual Studio wordt het scherm getoond welke lijkt op het scherm in Afbeelding 2.



Afbeelding 2 - Het opstartschermscherm

In dit venster heb je de mogelijkheid om een project op te halen van een externe server, het openen van een bestaand project, het openen van een map waarin code staat of het aanmaken van een nieuw project.

- Klik nu op Create a new Project.
- Een scherm welke lijkt op Afbeelding 3 zal verschijnen.



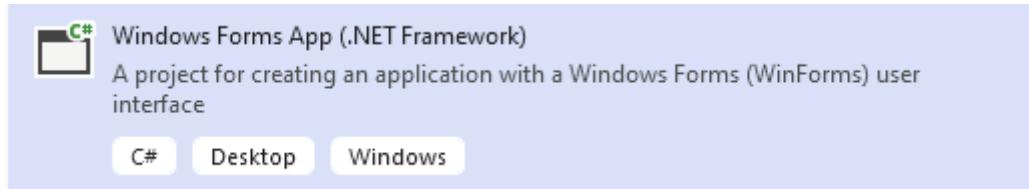
Afbeelding 3 - Nieuw project aanmaken

Je hebt nu de mogelijkheid om verschillende typen projecten aan te maken. Heb je bij de installatie van Visual Studio meerdere workloads geïnstalleerd, dan zul je hier meer opties terugvinden. In de komende periode ga je vooral werken met Windows Forms App projecten maar deze week houden we het nog bij console applicaties. Om kennis te maken met de Integrated Development Environment (IDE) ga je eerst een Windows Forms App aanmaken zodat je direct alle verschillende vensters in de IDE krijgt te zien.

Een video van deze handelingen is [hier](https://youtu.be/HFd65Lph3UU) (<https://youtu.be/HFd65Lph3UU>) te vinden.

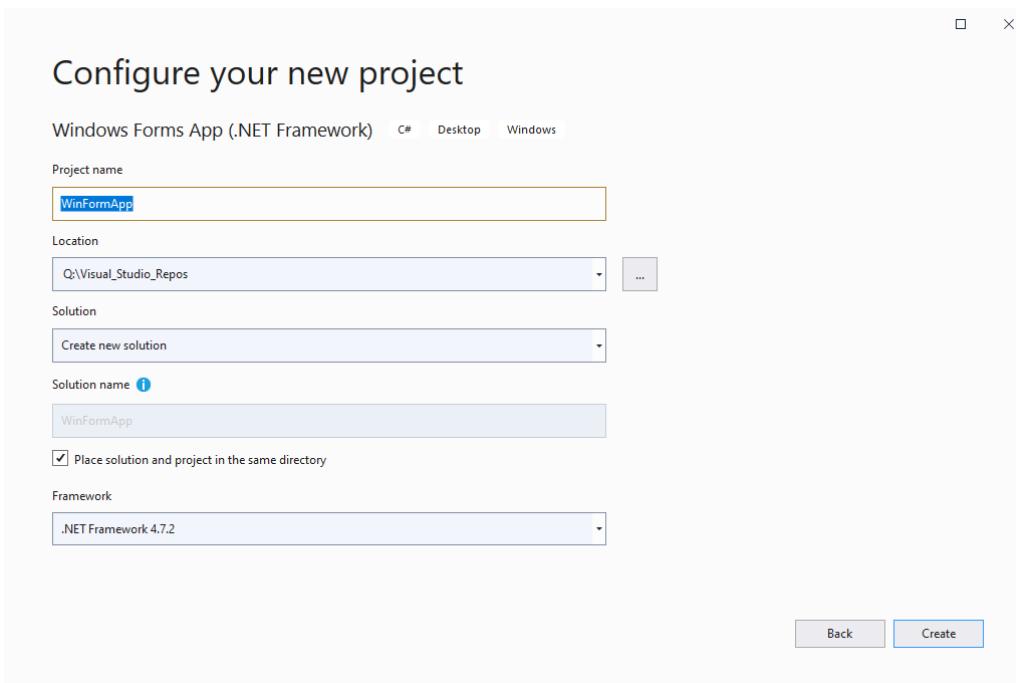
9.1 Opdracht – Project aanmaken

Maak een Windows Forms App (.Net Framework) project. Kies hier voor de optie welke is aangegeven in Afbeelding 4



Afbeelding 4 - Windows Forms App (.Net Framework)

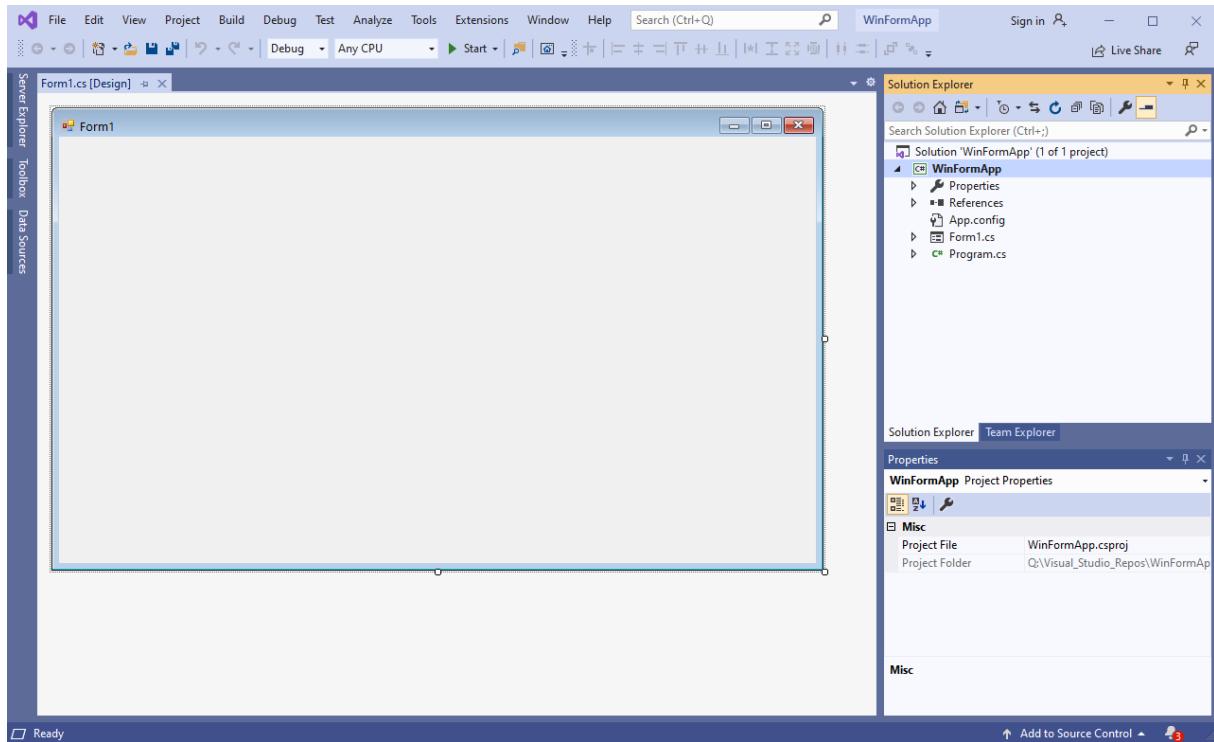
1. Zie je deze optie niet direct terug in het venster dan kun je naar beneden scrollen of in het vak 'Search for templates' zoeken op 'Windows Forms'.
2. Selecteer **Windows Forms App (.NET Framework)** zoals in Afbeelding 4 en klik op next.
3. Het venster uit Afbeelding 5 zal verschijnen.



Afbeelding 5 - Configure your new project.

- Maak voor de **Location** een map aan op je pc waar je de projecten op gaat slaan. Je kunt de standaard instelling laten staan maar het is prettiger om een map te kiezen welke eenvoudig te vinden is op je laptop.
- Geef dit project de naam 'WinFormApp'.
- De Solution Name zal automatisch ingevuld worden en dit kun je zo laten staan.
Een solution omvat alle code welke nodig is voor de applicatie. Een solution kan bestaan uit meerdere projecten en om deze reden wordt aan een solution vaak een andere naam gegeven dan aan het project. Aangezien de applicatie die je gaat maken bestaat uit één project is het geen probleem dat de naam gelijk is aan het project.
- Vink de optie 'Place solution and project in the same directory' aan zodat alle bestanden in dezelfde map worden geplaatst.

- Laat de instelling bij Framework staan. Dit zal automatisch het nieuwste Framework zijn welke beschikbaar is.
Deze instelling pas je alleen aan op het moment dat binnen het bedrijf met oudere Frameworks wordt gewerkt vanwege bijvoorbeeld compatibiliteit met andere applicaties.
- Klik op Create en je project wordt aangemaakt en de IDE in Afbeelding 6 zal verschijnen.

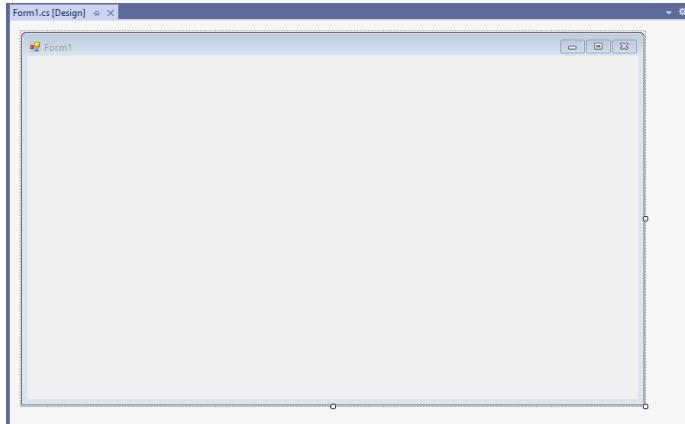


Afbeelding 6 - De IDE

9.2 De IDE

Het venster in Afbeelding 6 wordt de “Integrated Development Environment” (IDE) genoemd, die bestaat uit verschillende vensters en uiteraard de menubalk waarin je alle functionaliteiten kunt vinden. In dit hoofdstuk kijken we eerst naar de opbouw van het scherm.

Het hoofdvenster



Afbeelding 7 - Het hoofdvenster design

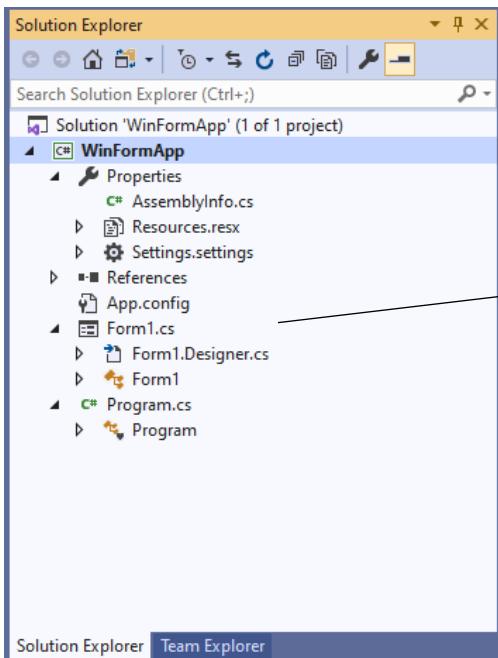
Het grootste venster is het hoofd- of werkvenster waar je onder andere je formulieren opmaakt en je code straks gaat typen. Bovenin dit venster zie je een tabblad met je huidige Project. In dit venster vinden de meeste bewerkingen plaats en maak je het formulier op, typ je de code, maak je formulieren en voer je andere bewerkingen uit.

```
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 WinFormApp
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19     }
20 }
21
```

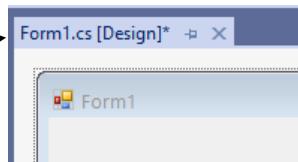
Afbeelding 8 - Het hoofdvenster code

Selecteer het formulier door er met de muis op te klikken en druk op F7 op je toetsenbord. Er wordt een nieuw tabblad geopend waarin je de code van het formulier kunt bewerken.

De Solution Explorer

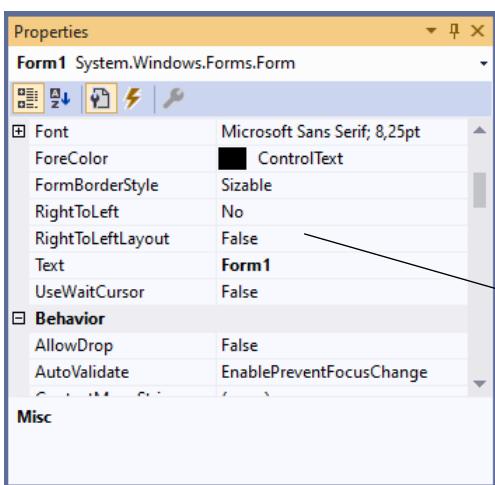


Rechts bovenin het scherm vind je de Solution Explorer. In dit venster kun je de hele structuur van het project vinden waaronder de bestanden en instellingen. Dit venster zul je later vaak gaan gebruiken om verschillende formulieren te openen.

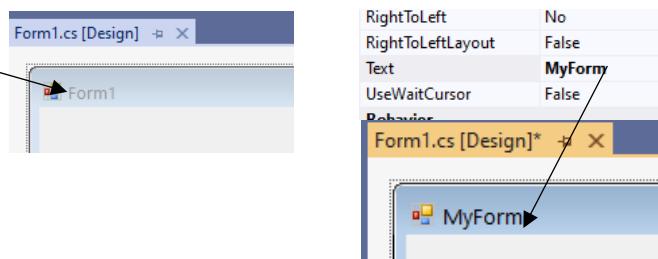


Afbeelding 9 - De Solution Explorer

Het Properties Venster



Rechts onderin het scherm vind je het Properties venster. In dit venster kun je alle eigenschappen van objecten vinden zoals bijvoorbeeld knoppen, invulvelden en andere items. In Afbeelding 10 kun je zien dat je bijvoorbeeld met de eigenschap **Text** de tekst bovenin het formulier kunt aanpassen.



Afbeelding 10 - Het Properties Venster

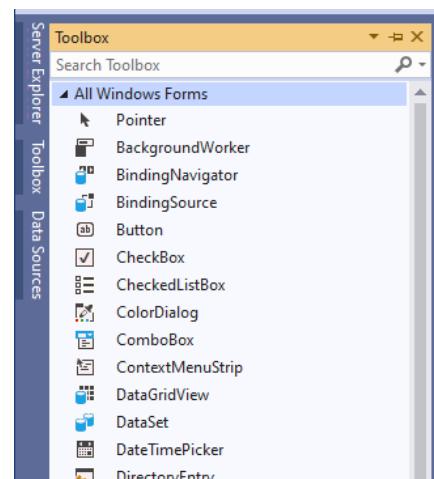
Toolbox

Links in het venster vind je een aantal knoppen. Als je op deze knoppen klikt zal een extra venster openen.

Server Explorer: Hierin staan de servers zoals bijvoorbeeld een database server waar het project een connectie mee kan maken.

Toolbox: In Afbeelding 11 zie je de Toolbox waar objecten in staan welke je op je formulier kunt plaatsen zoals knoppen en invulvelden.

Data Sources: Klik je op Data Sources dan krijg je databronnen te zien die gekoppeld zijn aan je project.



Afbeelding 11 - De Toolbox

De Menubalk

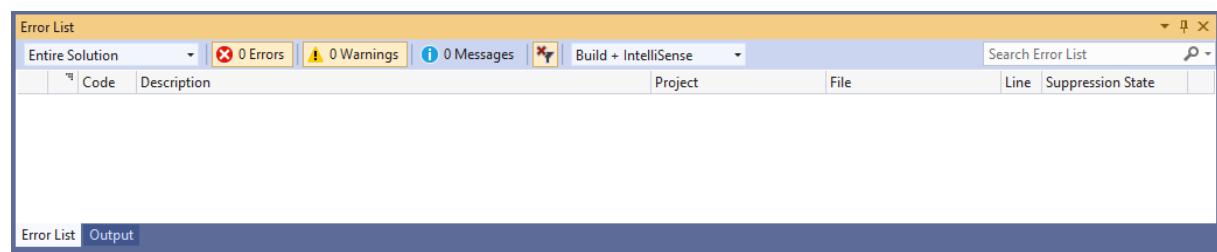


Afbeelding 12 - De Menubalk

In de menubalk vind je uiteraard de meeste functies terug. Gedurende de komende weken zul je de menustructuur verkennen en ga je meerdere functies gebruiken. Voor nu is de belangrijkste knop de Debug knop die je in het midden van de balk kunt vinden. Met deze knop kun je je programma debuggen wat betekent dat je gaat testen of het programma werkt.

De error/output list

In de error list onderin het scherm verschijnen eventuele foutmeldingen tijdens het maken van je applicatie.



10 Applicatie - Hello Developer Land

10.1 Aanmaken project

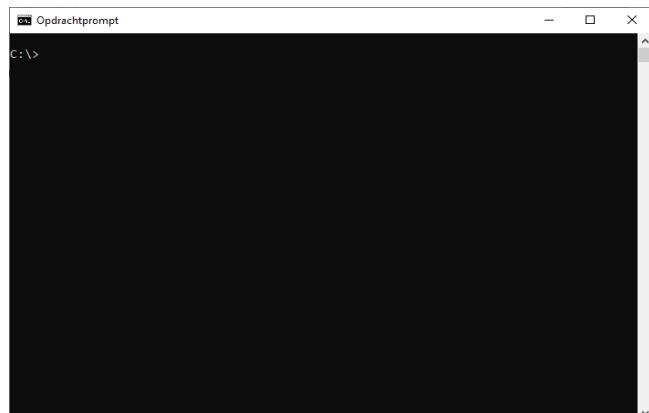
Het is nu tijd om je eerste programma te schrijven. Sluit het project die je in het vorige hoofdstuk hebt aangemaakt. Je hoeft dit niet op te slaan.

Het eerst programma dat je gaat schrijven wordt een Console Applicatie met als naam 'HelloDeveloperLand'. Dit is uiteraard een afgeleide van 'Hello World' wat, in bijna alle ontwikkel cursussen die je kunt vinden, een standaard is voor je eerste project.

'Hello World' is een simpel computerprogramma wat niets meer doet dan de tekst 'Hello World' tonen op het scherm. Het is bedacht door de ontwikkelaars van de programmeertaal C en met dit eenvoudige programma konden ze laten zien dat de programmeertaal geschikt was om iets op een scherm te laten zien.

Bron: [https://nl.wikipedia.org/wiki/Hello_world_\(programma\)](https://nl.wikipedia.org/wiki/Hello_world_(programma))

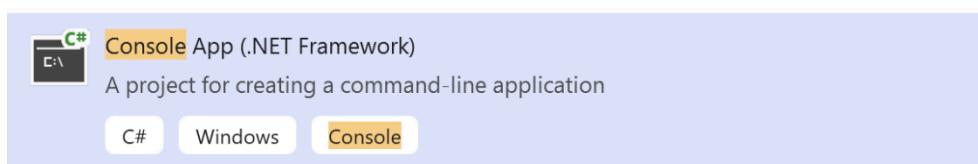
Een console applicatie is een applicatie welke in de cmd venster wordt uitgevoerd en dus volledig bestaat uit tekst. Je kent dit venster waarschijnlijk wel uit Windows en het lijkt op de terminal uit Linux. Voor de eerste kennismaking met C# is het prettig omdat je in een console applicatie geen rekening hoeft te houden met de verschillende objecten en properties welke wel in een Windows Forms applicatie aanwezig zijn.



Afbeelding 13 - Het console venster

10.2 Opdracht – Hello World

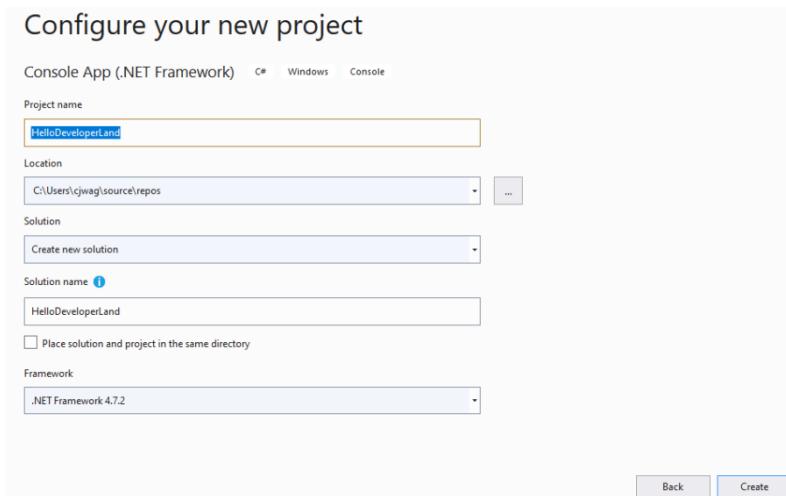
1. Maak nu een project aan en kies voor Console App (.Net Framework) zoals aangegeven in Afbeelding 14.



Afbeelding 14 - Console App (.Net Framework)

2. Een venster dat lijkt op Afbeelding 15 zal verschijnen.

- Je ziet dat dit iets anders is dan bij het aanmaken van de Windows Form applicatie uit de vorige hoofdstukken.



Afbeelding 15 - Naam en opslag van je project.

- Vul bij Project Name in HelloDeveloperLand en kies de locatie waar je het project op wilt slaan.
- Klik op Create en in het hoofdvenster zal de code uit Afbeelding 16 verschijnen.
- In de Solution Explorer en het Properties Venster is nu niet veel te zien.

```

Program.cs  X
HelloDeveloperLand\Program.cs  HelloDeveloperLand
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace HelloDeveloperLand
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13         }
14     }
15 }
16

```

Afbeelding 16 - HelloDeveloperLand Console App

- Ga naar regel 13 en druk op Enter zodat er tussen de accolades een regel open staat.
- Ga nu weer op regel 13 staan. Je staat nu in de **scope** van **static void Main**. De code welke wordt uitgevoerd als het programma start moet in deze scope komen te staan. De **scope** is dus de 'verzamelbak' met code welke hoort bij een class of method.
- Typ onderstaande code over op regel 13 zodat deze in de scope van static void Main staat en dus wordt uitgevoerd zodra het programma start.

```
Console.WriteLine("Hello World!");
```

- In de volgende paragrafen krijg je uitleg over de code in je applicatie. Laat je applicatie even open staan want je gaat zo meteen meer code toevoegen.

10.3 De code lezen

Voor je start met het programmeren is het uiteraard belangrijk dat je begrijpt wat er op het scherm staat. Je moet de code eerst kunnen lezen voordat je in staat bent om de code aan te passen of functionaliteiten toe te voegen. Visual Studio heeft al een groot deel zelf ingevuld en in de voorgaande paragraaf heb je zelf al een stukje code toegevoegd. De code ziet er nu als volgt uit.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HelloDeveloperLand
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World");
        }
    }
}
```

Een aantal delen in de code worden later uitvoerig behandeld dus hier gaan we nu maar kort op in. Je hoeft dit niet allemaal te onthouden. Gedurende de opleiding komen deze onderdelen continu voorbij en leer je gaandeweg wat ze doen. Voor nu is het een verduidelijking van wat je op je scherm ziet.

using System;	In dit deel geef je aan dat jouw programma gebruik maakt van de System Library. Dit is een set met objecten, methodes en functies die je beschikbaar hebt waaronder bijvoorbeeld het object Console.
namespace HelloDeveloperLand {}	Namespace een soort map waarin alle code en objecten staan die bij het project horen. Zie dit als een map op je pc waarin je documenten opslaat die bij elkaar horen. Visual Studio heeft deze map nodig omdat het moet weten waar alles moet worden opgeslagen.
class Program {}	Je gaat een klasse aanmaken met als naam Program. Hierin komen alle functies en methoden welke bij het object Program horen. Hier leer je later meer over als je met verschillende klassen gaat werken.
static void Main(string[] args) {}	Met Static Void Main maak je een methode aan in je klasse. Je voegt een stukje functionaliteit toe aan je programma. Hierover leer je later meer.
Console.WriteLine("Hello World!");	Dit onderdeel komt zo meteen uitgebreider terug.
;	De puntkomma of ; gebruik je aan het einde van elk stuk code als dit is afgerond. C# weet dan dat dat stuk code compleet is.
{ en }	Met { en } geef je de scope aan wat er allemaal tot het object behoort. Lees je de code volledig dan staat er:
	<ul style="list-style-type: none">• Console.WriteLine("Hello World!"); = onderdeel van static void Main(string[] args)• static void Main(string[] args) = onderdeel van class Program• class Program = onderdeel van namespace HelloDeveloperLand

- In het programma zie je op regel 13 de code waarmee je aangeeft dat in de Console de tekst '**Hello World!**' moet worden getoond.
- Deze code is als volgt opgebouwd.

```
Console.WriteLine("Hello World!");
```

- De class `Console` is het zwarte cmd venster wat je al eerder hebt gezien.
- Je kunt dit object gebruiken in je code omdat je bovenaan de code `using System;` hebt gebruikt.

```
Console.WriteLine("Hello World!");
```

- `WriteLine()` is een methode die er voor zorgt dat er iets op het scherm wordt weergegeven.
- Op de achtergrond wordt er dus een stuk code uitgevoerd die ervoor zorgt dat de tekst op het scherm komt. Het voordeel is dat je deze code nu niet helemaal zelf hoeft te typen.

```
Console.WriteLine("Hello World!");
```

- Tussen de haakjes staat wat de methode `WriteLine` in de console moet schrijven wat in dit geval de tekst **Hello World** is. Je ziet dat de volledige code hiervoor "**Hello World!**", dus tussen aanhalingstekens staat. Met aanhalingstekens wordt in C# aangegeven dat de waarden tussen de tekens een **string** (tekst) is. Later leer je waarom je een tekst tussen aanhalingstekens moet plaatsen.

```
Console.WriteLine("Hello World!");
```

- In C# dien je na elke opdracht aan te geven dat deze klaar is door een puntkomma (;) achter de code te zetten. Hierdoor weet het programma dat de volgende opdracht uitgevoerd kan worden.

10.4 Class, Object, Property en Method

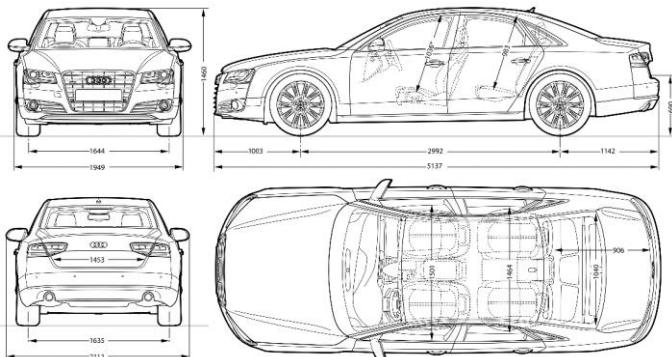
C# is een object georiënteerde programmeertaal wat betekent dat je aan het programmeren bent rondom objecten. Het daadwerkelijk object georiënteerd programmeren komt later in de opleiding uitgebreid aan bod maar in deze lessen ga al je regelmatig de begrippen Class (Klasse), Object (Object), Method (Methode) en Property (Eigenschap) tegenkomen. Om die reden kun je in deze paragraaf, aan de hand van een alledaags voorbeeld, lezen wat de begrippen globaal inhouden. Het is voor nu behoorlijk complex en je hoeft dit niet uit je hoofd te leren. Zorg ervoor dat je het in ieder geval doorleest zodat je de termen herkent als je ze later in de lessen tegen komt. Snap je het na het lezen nog niet helemaal dan is dat helemaal niet erg, dit komt vanzelf gedurende je opleiding.

Om het iets eenvoudiger te maken wordt er geen programma als voorbeeld gebruikt, maar een fabriek waar ze de Audi A8 bouwen.

Class

Een **class** kun je zien als een bouwtekening van een **object**. In de fabiek is de class dus de bouwtekening die gebruikt wordt voor alle Audi's A8. In deze class staat dus beschreven waaraan een Audi A8 minimaal moet voldoen om een Audi A8 te zijn zoals bijvoorbeeld;

- De Audi A8 moet vier wielen hebben.
- De Audi A8 moet ramen en deuren hebben.
- De Audi A8 moet carrosserie hebben.
- De Audi A8 moet een kleur krijgen.



Afbeelding 17- Class - Blauwdruk Audi A8

Een **class** is een stuk code dat bij elkaar hoort en een duidelijke functie heeft. Een **class** kan ook bestaan uit meerdere andere **classes**. Zo kun je in het voorbeeld van de fabiek ook stellen dat er een **class Auto** bestaat waarin de eigenschappen van alle auto's zijn benoemd en dat binnen deze **class** een **class** is aangemaakt voor de Audi A8 met daarin de specifieke eigenschappen waaraan alle Audi's A8 minimaal moeten voldoen. In dat geval hoeft in de **class** Audi A8 bijvoorbeeld niet te staan dat er deuren aanwezig moeten zijn want dat staat al in de **class** Auto.

Object

Niet alle Audi's A8 zijn exact hetzelfde en in de class staan ook niet alle details van de auto die gebouwd gaan worden. In de class staat bijvoorbeeld dat de auto een laklaag krijgt maar niet welke kleur. Op basis van de class (blauwdruk) wordt de auto gemaakt en in de fabiek gaat hij bijvoorbeeld een bepaalde kleur krijgen. Deze gebouwde auto wordt een **object** genoemd wat dus eigenlijk een kopie of **instance** is van de **class** waarbij de bouwer heeft aangegeven wat bijvoorbeeld de kleur wordt.

Van Class naar Object staat in dit voorbeeld dus gelijk aan "van tekening naar echte auto"



Afbeelding 18 - Object – "Een echte" Audi A8

Property

Met properties worden eigenschappen bepaald welke uniek zijn voor het object. Met een property wordt in het voorbeeld onder andere de kleur van de auto bepaald of bijvoorbeeld dat de ramen geblindeerd zijn zoals in Afbeelding 18. Ook de soort bekleding, soort velgen etc zijn eigenschappen van het object dat je op de foto ziet.

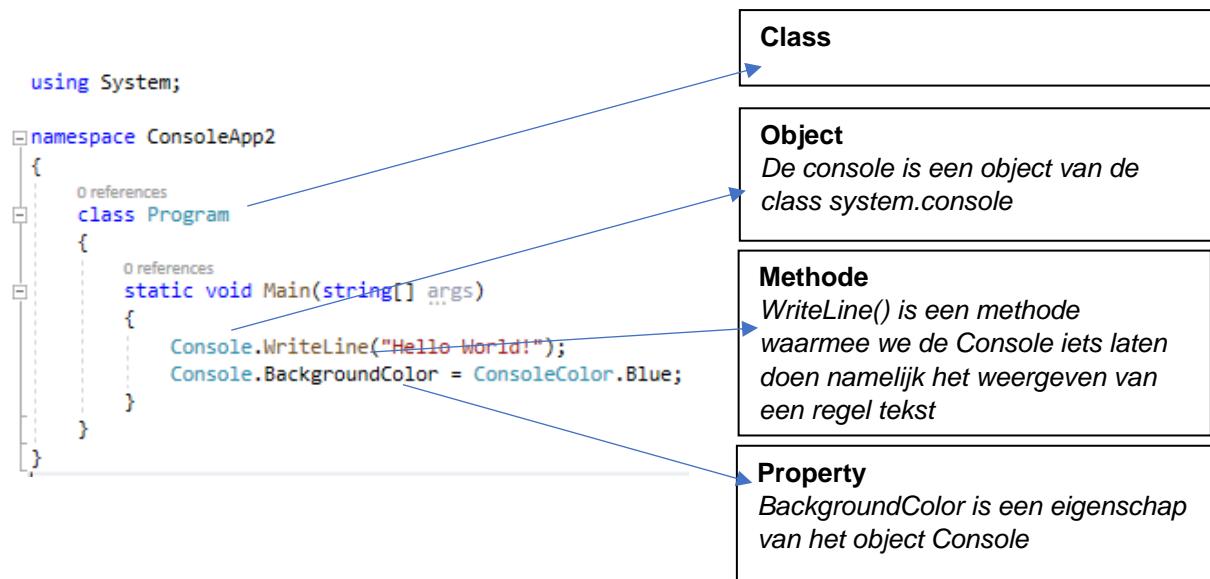
Method

Een method is een stuk functionaliteit dat wordt toegevoegd aan een object waardoor het object iets kan ‘uitvoeren’ of waardoor er iets met het object kan worden gedaan. Methoden van het object “Audi A8 met kenteken 52-ZVN-4” zijn bijvoorbeeld remmen, rijden, sturen.



Afbeelding 19 - Remmen

Het gebruik van de fabriek is natuurlijk om het wat duidelijker te maken. Maar hoe ziet dit er nu uit als je naar de code van jouw programma kijkt. Hieronder wordt dit schematisch weergegeven.



Kleuren in Visual Studio

In Visual Studio zie je dat bepaalde onderdelen een andere kleur krijgen. Op deze manier wordt het eenvoudig om deze onderdelen te herkennen.

- Een Class - Blauw
- Een Object - Geel
- Een Method - Oker
- Een string - Rood

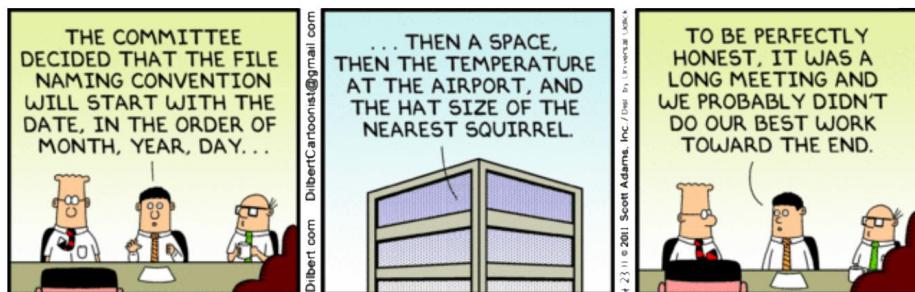
De Console is dus eigenlijk geen object maar een class. Dit wordt een Static Class genoemd, dat is een klasse die je gebruikt als een object (de blauwdruk is direct het “eindproduct”).

10.5 Conventions

In eerdere lessen heb je wellicht al iets gehoord over het gebruik van naamgevingsconventies. C# is een hoofdlettergevoelige programmeertaal waardoor het nog belangrijker is om je aan deze conventies te houden. Je maakt bijvoorbeeld een variabele aan voor het opslaan van de leeftijd van de gebruiker en noemt deze **leeftijdGebruiker**. Roep je deze variabele verderop in je code aan met **ageUser**, dan zal C# een melding geven dat hij **leeftijdGebruiker** niet kent omdat de naam van de variabele **leeftijdgebruiker** is en niet **leeftijdGebruiker**. Naast de hoofdlettergevoelheid kan het ook voorkomen dat je namen gaan gebruiken welke al door Visual Studio gebruikt worden voor bijvoorbeeld functies. Door je te houden aan de naamgevingsconventies voorkom je dit.

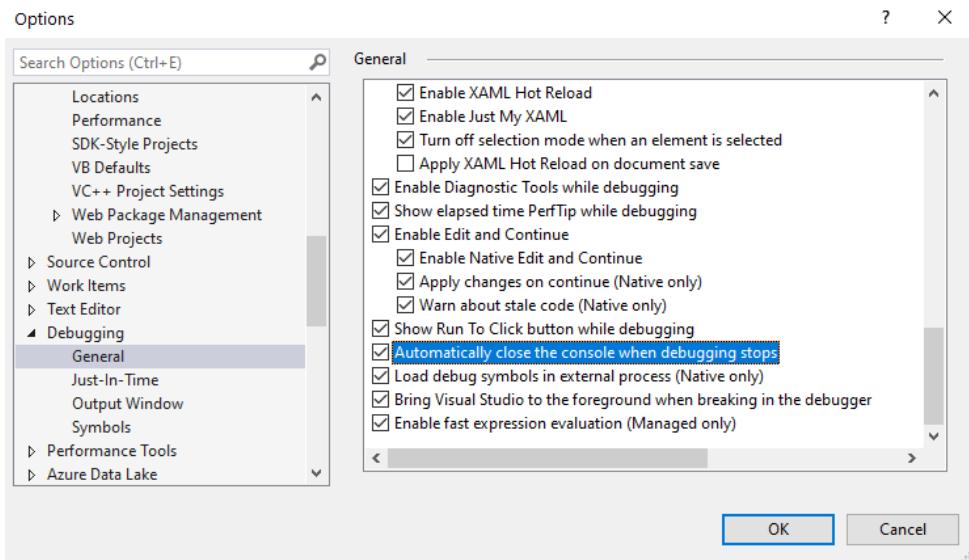
Op internet en bij bedrijven kun je soms verschillende naamgevingsconventies tegen. Op school hanteren we de naamgevingsconventies die worden geadviseerd door Microsoft. Hieronder een aantal voorbeelden. In de bijlage 1 vind je een uitgebreidere lijst.

- Gebruik zelfstandige naamwoorden of zinnen met een zelfstandig naamwoord voor classes.
- Gebruik **PascalCase** voor classes, properties en bestandnamen.
(Bij PascalCase wordt ieder woord met een hoofdletter geschreven zoals **AutoSize**)
- Gebruik **camelCase** voor fields, variabelen en argumenten.
(Bij camelCase wordt het eerste woord met een kleine letter geschreven de volgende woorden met een hoofdletter zoals in **leeftijdGebruiker**)
- Gebruik zo min mogelijk afkortingen.
- Gebruik geen – (streepje of minteken) en liever ook geen _ (underscore).



10.6 Opdracht – Hello Developer Land

1. Het is belangrijk tijdens het programmeren dat je de werking van het programma krijgt te zien zoals de gebruiker dat gaat zien. Om deze reden ga je een aanpassing doorvoeren bij de instellingen van Visual Studio.
2. Ga in de menubalk naar Tools > Options en vink de optie ***Automatically close the console when debugging stops*** aan zoals aangegeven in Afbeelding 20.



Afbeelding 20 - Close after debugging

3. Klik op OK om dit scherm te sluiten.
4. Start nu het programma uit de vorige paragraaf door op de debug knop te klikken in de menubalk.
5. Waarschijnlijk heb je niet veel gezien. Wel is er in Visual Studio onder in je scherm een nieuw venster 'Output' geopend.
6. Dit venster geeft weer wat het programma heeft gedaan en zal ook eventuele fouten in je code tonen. Dit venster ga je later veel gebruiken bij het opsporen van de fouten of bugs in je code.
7. De code die je nu hebt bestaat alleen uit het tonen van de tekst 'Hello World!' op het scherm en daarna is het programma klaar. Zodra het programma klaar is zal het cmd venster gesloten worden.
8. Dit gaat zo snel dat je het eigenlijk helemaal niet kunt zien.

9. Je gaat in de volgende paragraaf je programma aanpassen zodat het venster pas wordt gesloten zodra er op een toets wordt gedrukt. Hiervoor ga je een andere methode gebruiken genaamd **.ReadKey()**.

Weetje

De term BUG of fout komt (ten onrechte) uit 1947 toen er een storing was in de toenmalige ‘computers’ die toen nog bestonden uit grotere elektronische componenten. Bij het uitzoeken van een storing werd een nachtvlinder gevonden welke kortsluiting veroorzaakte in een Relais (één van de componenten). Er werd toen door de beheerders in het logboek geschreven "first actual case of bug being found". De echte term BUG komt waarschijnlijk uit de oude Welsch taal waar ‘bwg’ staat voor ‘een onverwacht defect, fout’.

Bron: [https://nl.wikipedia.org/wiki/Bug_\(technologie\)](https://nl.wikipedia.org/wiki/Bug_(technologie))

10.7 Opdracht - IntelliSense

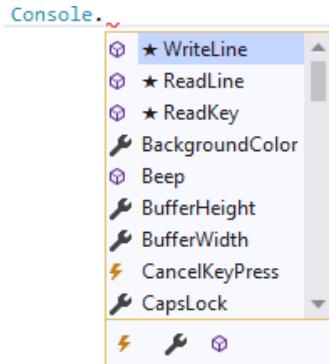
In de voorgaande oefeningen heb je alleen maar naar code gelezen en nu ga je daadwerkelijk een stukje code schrijven. Een handig hulpmiddel in Visual Studio is IntelliSense. IntelliSense draait op de achtergrond en gaat jou ondersteunen bij het schrijven van Code.

1. Geef een Enter op regel 14 in de code van je programma zodat de } op regel 15 komt te staan.
2. Begin op regel 14 met het typen van het woord **Console**.
3. Je zult merken dat zodra je **Con** op je scherm hebt staan dat Visual Studio een suggestie doet van wat het moet worden (Afbeelding 21 - IntelliSense).
4. Aangezien je een Console project hebt aangemaakt en Console al in je code voorkomt zal Visual Studio de meest logische opties voor de situatie bovenaan zetten in de lijst. In dit geval is dat uiteraard de klasse Console. Je kunt op TAB of ENTER drukken en Visual Studio maakt voor jou het woord af.



Afbeelding 21 - IntelliSense

5. Typ nu achter Console een punt en je zult zien dat er wederom een lijst verschijnt. In deze lijst staan **properties**, **methoden** en **events** die gebruikt kunnen worden bij de klasse Console.
(Je leert later wat **events** zijn en hoe je deze kunt gebruiken)



Afbeelding 22 - IntelliSense

6. Zorg dat onderstaande code in je programma komt te staan.

```
Console.ReadKey();
```

7. Je maakt nu gebruik van de methode `.ReadKey()` wat een onderdeel is van het object `Console`. De methode `.ReadKey()` zorgt ervoor dat je programma wacht tot de gebruiker op een toets drukt. Je herkent methodes aan het  "doosje".
8. Zorg er nu voor dat in plaats van de tekst '**Hello World!**' onderstaande tekst in de console verschijnt. *Hiervoor pas je één regel aan en voeg je één regel toe.*

Welkom in Developer Land!
Het attractiepark voor ontwikkelaars van over de hele wereld.

10.8 Opdracht - Properties

Eerder heb je kunnen lezen dat objecten en classes ook eigenschappen hebben; zo ook de console. In onderstaande opdrachten leer je een paar eigenschappen van de class Console aan te passen. Ook hier gaat IntelliSense je uiteraard helpen. Je kunt Properties in IntelliSense herkennen aan het steeksleutel icoontje .

1. Pas de achtergrond kleur van het console venster aan naar wit door onderstaande code boven de `Console.WriteLine(.....);` te plaatsen op regel 13.

```
Console.BackgroundColor = ConsoleColor.White;
```

2. Een witte tekst op een witte achtergrond is natuurlijk niet te lezen dus voeg onderstaande code toe op regel 14 om de tekst zwart te maken.

```
Console.ForegroundColor = ConsoleColor.Black;
```

3. Voeg onderstaande methode toe op regel 15 zodat de kleuren worden toegepast bij het uitvoeren van het programma.

```
Console.Clear();
```

4. Voer je programma uit en kijk of het werkt. Als het goed is zit je console eruit als in Afbeelding 23



Afbeelding 23 – Console met aangepaste kleuren

5. Probeer nog eens wat andere kleuren of properties aan te passen.

10.9 Interactie met de gebruiker

Ondanks alle aanpassingen aan je applicatie in de voorgaande paragraaf is het programma eigenlijk een beetje saai en simpel want er is totaal geen interactie met de gebruiker. De volgende stap is dan ook een stukje interactie met de gebruiker toe te voegen aan je programma. Hiervoor ga je gebruik maken van een nieuwe methode `ReadLine()` waarmee het programma wacht op een invoer van de gebruiker zoals dat ook gebeurd bij de methode `.ReadKey()` die je hiervoor hebt gebruikt. Met `ReadLine()` kun je echter hele zinnen of getallen typen en gaat het programma verder als je op enter hebt gedrukt. De code die je hier zo meteen voor gaat gebruiken staat hieronder weergegeven.

```
Console.ReadLine();
```

Om met de getypte waarde iets te kunnen doen ga je deze waarde opslaan in een **variabele** zodat je deze verder kunt gebruiken in je programma. Dus in het volgende hoofdstuk eerst iets meer over variabelen voordat je verder kunt met je programma.

10.10 Variabelen

Zoals je wellicht al weet is een variabele een stukje geheugen wat je ‘declareert’ of reserveert in het geheugen van de computer voor het programma zodat het programma tijdelijk iets op kan slaan en dit later gebruiken.

Variabelen worden geschreven in camelCase dus beginnend met een kleine letter en ieder vervolgwoord met een hoofdletter zoals in de variabele `leeftijdGebruiker`.

10.10.1 Plaatsing van variabelen

Aangezien een computer de code van jouw applicatie van boven naar beneden leest dien je de variabele altijd eerst te declareren voordat je deze in je code kunt gebruiken. Dit klinkt logisch maar je zult zien dat je in de toekomst regelmatig 'even' nog een variabele aanmaakt en deze op de verkeerde plek zet. Leer jezelf aan om altijd de variabele bovenin het stuk code te declareren waar je mee bezig bent. Later in de opleiding zul je nog gaan leren dat er Global Variables en Private variables bestaan maar daar ga je je nu nog niet mee bezig houden. Voor nu ligt de focus op de twee typen variabelen welke in de volgende paragrafen staat beschreven.

10.10.2 Value type variabele

In deze variabele wordt direct een waarde opgeslagen bijvoorbeeld;

```
string gebruikersNaam = "Jan";
```

Bovenstaande code zorgt ervoor dat het programma de string Jan in het geheugen van de computer staat. Deze waarde kan later in het programma gebruikt worden. Dit wordt duidelijk met de volgende opdracht.

1. Pas je programma aan en voeg bovenstaande regel toe aan je programma en plaats deze op de regel boven je eerste **Console.WriteLine**.
2. Vervang de string **Jan** door de tekst **Hello Developer Land**.
3. Pas de regel **Console.WriteLine("Hello Developer Land");** aan en vervang "**Hello Developer Land**" door **gebruikersNaam**. (Dus niet tussen "")
4. Start je programma en kijkt of het werkt.

Als alles goed is gegaan kreeg je hetzelfde resultaat als de vorige keer alleen maak je nu gebruik van een variabele. Het programma reserveert een stukje geheugen en slaat daar de string 'Hello Developer Land' in op. Met **gebruikersNaam** wordt deze string uit het geheugen opgehaald en geplaatst op de plek waar je deze variabele in je code hebt staan. Nu lijkt dit onzinnig omdat je de tekst ook gewoon in de code zelf had kunnen plaatsen maar stel je voor dat je later in een complex programma de tekst Hello Developer World tien keer nodig hebt. Je dient dan op alle plekken op de tekst aan te passen. Door het gebruik van een variabele hoeft je het maar op één plek aan te passen en is de kans op het maken van fouten een stuk kleiner. De naam van de variabele is nu gebruikersNaam maar dat is natuurlijk niet de juiste benaming want er wordt geen gebruikersnaam opgeslagen. Voor deze opdracht mag je dat even negeren.

Een belangrijke regel is dat je nooit vaste/constante waarden in je code opneemt en dus altijd een variabele gebruikt.

10.10.3 Reference type variable

In deze variabele wordt een verwijzing opgeslagen naar bijvoorbeeld een klasse, object of index. De waarde wordt dus later door je programma opgehaald nadat het programma al is gestart. Een voorbeeld welke je in je code kunt gebruiken is de volgende.

```
string gebruikersNaam = Console.ReadLine();
```

- In de code zie je dat de **Console.ReadLine()** in de variabele **gebruikersNaam** wordt geplaatst.
- Bij het starten van het programma is dus nog niet duidelijk wat de waarde van de variabele is want deze is pas duidelijk op het moment dat de `Console.ReadLine();` is uitgevoerd en de gebruiker een waarde heeft getypt.
- Je programma zal dan ook de `Console.ReadLine();` uitvoeren zodat de waarde duidelijk wordt.
- In de code zie je dat er voor de variabele **gebruikersNaam** ook nog **string** staat. Hiermee geef je aan dat de data in de variabele van het type string is.
- In C# is het belangrijk dat je vooraf aangeeft wat voor waarde er in de variabele wordt opgeslagen zodat C# weet wat hij met deze waarde kan doen. Later ga je hier uitgebreider mee aan de slag in het onderdeel **datatype's**.

Dit alles lijkt ingewikkeld maar je zult zien dat je straks zonder na te denken gebruik maakt van deze twee typen variabelen. In de volgende opdrachten leer je hoe je de variabele verder kunt gebruiken in je code.

10.11 Eindopdracht – Interactie met de gebruiker

In deze opdracht ga je stap voor stap je programma aanpassen zodat er interactie plaatsvindt met de gebruiker. Het doel is dat de gebruiker zijn naam invult en dat hij dan persoonlijk wordt begroet met onderstaande tekst.

**Welkom in Developer World <naam van de gebruiker>!
Het attractiepark voor ontwikkelaars van over de hele wereld.**

1. Pas je code aan zodat deze dezelfde lay-out heeft als onderstaande code. (Regel 12 en 13 zijn de twee lege regels in onderstaande code. Hier ga je zo meteen code toevoegen).

```
using System;

namespace HelloDeveloperLand
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.BackgroundColor = ConsoleColor.White;
            Console.ForegroundColor = ConsoleColor.Black;
            Console.Clear();

            Console.WriteLine("Welkom in Developer World!");
            Console.WriteLine("Het attractiepark voor ontwikkelaars van over
de hele wereld.");
            Console.ReadKey();

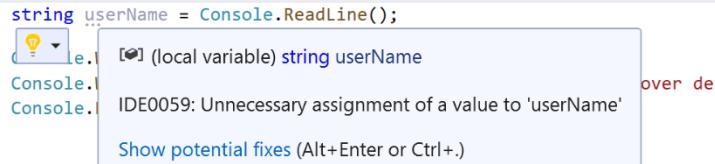
        }
    }
}
```

2. De code wordt van boven naar beneden doorlopen dus voordat het programma de eerste regel op het scherm kan tonen moet het eerst weten wat de naam van de gebruiker is.
3. De code die je toe gaat voegen om de naam van de gebruiker te vragen komt dus voor de code welke er nu voor zorgt dat de tekst op het scherm komt te staan.
4. Voeg een stuk code in op regel 12 waarmee de zin “Wat is uw naam?” op het scherm komt te staan.
5. Op regel 13 voeg je de code in die besproken is in dit hoofdstuk.

```
string gebruikersNaam = Console.ReadLine();
```

6. Voer je nu het programma uit dan kan de gebruiker zijn naam invullen maar er wordt nog niets mee gedaan. Je zult eerst aan moeten geven waar het programma de waarde van de variabele (de naam van de gebruiker) moet gebruiken.

7. C# zal dit ook aangeven in je code. Je ziet onder de variabele **gebruikersNaam** drie puntjes staan en de kleur van de variabele is lichtgrijs.
8. Ga met je muis op de drie puntjes staan en je krijgt onderstaande melding.



Afbeelding 24 - Unnecessary assignment

9. C# geeft hiermee aan dat je onnodig een variabele GebruikersNaam hebt gedeclareerd in je code.
10. Dat is natuurlijk niet het geval want je hebt de variabele wel degelijk nodig. Je hebt alleen nog niet aangegeven waar de variabele gebruikt moet worden.
11. Pas de code van regel 14 aan zodat deze hetzelfde is als onderstaande code.

```
Console.WriteLine("Welkom in Developer Land " + gebruikersNaam + "!");
```

Bij het invoeren van bovenstaande code heb je wellicht gemerkt dat de door jou gedeclareerde variabele GebruikersNaam ook beschikbaar is in IntelliSense.

12. In bovenstaande code heb je jouw variabele **gebruikersNaam** toegevoegd aan de **Console.WriteLine()** waarmee de waarde van de variabele op het scherm wordt getoond.
13. Door gebruik te maken van de + tekens plak je de eerste string "**Welkom in Developer Land** ", **gebruikersNaam** en **"!"** aan elkaar. Op deze manier kun je een variabele ook midden in een zin gebruiken. *Je moet rekening houden met spaties aangezien de stukken tekst letterlijk aan elkaar worden geplakt. Aan het einde van "Welkom in Developer Land" is dus bewust een spatie gebruikt.*
14. Voer je programma nu uit om te testen of het werkt. Als je alles goed hebt gedaan krijg je eerst de vraag om je naam in te vullen en daarna wordt de tekst getoond waarin de naam verwerkt is.

Tip - De **Console.WriteLine()** methode toont de opgegeven tekst altijd op een nieuwe regel. Wil je een tekst op dezelfde regel plaatsen als de voorgaande tekst, dan kun je gebruik maken van de **Console.Write()** methode.

10.12 Wat heb je geleerd?

In de voorgaande hoofdstukken heb je geleerd;

- De IDE van Visual Studio.
- Het lezen van code.
- Je weet wat IntelliSense is.
- Je weet wat een Scope is.
- Je weet wat een class, object, property en methode is.
- Hoe je een variabele kunt declareren en gebruiken in C#
- Je weet het verschil tussen een Value Type Variabele en een Reference Type Variabele.
- Je kunt een eenvoudige console applicatie bouwen met interactie.

Meer informatie

Boek – Leren programmeren in C#

Onderwerp	Hoofdstuk / Paragraaf	Pagina
IDE	De programmeeromgeving (IDE)	9
Naamgeving	De programmeeromgeving (IDE)	8
Aanmaken Project	De eerste handeling	10

11 Console Applicatie – Rekenen

In dit hoofdstuk ga je een nieuwe applicatie bouwen waarmee je verschillende eenvoudige berekeningen uit kunt voeren en verschillende datatypen gaan gebruiken. Voor je deze applicatie kunt gaan bouwen is het noodzakelijk dat je eerst leert hoe een computer omgaat met deze data.

11.1 Bits en Bytes

In een computer zitten veel onderdelen welke met elkaar samenwerken de onderdelen en het hart van de computer bestaan uit de processor en het geheugen. Een processor is eigenlijk niets meer dan een heleboel kleine schakelaars (transistoren) die aan of uit kunnen staan. Ook het geheugen en een opslagmedium zoals een harddisk is op deze manier opgebouwd.

Een schakelaar kan aan of uit staan. Vertalen we dit naar een getal dan is uit 0 en aan 1. Dit noemen we ook wel binair. Één schakelaar wordt dan een bit genoemd. Deze term ken je al in de vorm van Mb (Megabit), Gb (Gigabit), Tb(Terrabit). Acht van deze **bits** vormen samen een **byte**.



Afbeelding 25 - Een processor



Afbeelding 26 - Ziggo's Gigabit netwerk

Wellicht heb je je wel eens afgevraagd waarom je thuis met je gigabit aansluiting toch maar kunt downloaden met ongeveer 112 MB/s (*let op de hoofdletter B in MB*). Dit komt dus omdat de leveranciers het hebben over **Gigabit** verbindingen en je download wordt weergegeven in **Byte** (in dit voorbeeld MB oftewel **MegaByte** per seconde). Aangezien er 8 bits zitten in een Byte moet je dus het getal delen door 8 om op de snelheid uit te komen die je op je scherm gaat zien. (*Als je rekent kom je uit op 125 MB/s maar er gaat er ook nog wat snelheid verloren door de netwerkconfiguratie maar dat is voor de systeembeheerder en netwerkbeheerders, daar gaan we jou niet mee lastig vallen.*)

11.2 Binair

Aangezien een computer alleen bits en bytes kan lezen is er een nieuw talstelsel ontwikkeld genaamd het binair talstelsel. Dit talstelsel bestaat uit 1-en en 0-en; oftewel ‘schakelaar aan’ en ‘schakelaar uit’. In het onderstaande schema zie je dit talstelsel terug.

	Bit	Byte = 8 Bits							
Binair	0	0	0	0	0	0	0	0	
Omrekenen	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
Getal	128	64	32	16	8	2	1	0	
<hr/>									
Binair	Omrekenen								Waarde
0000 0000									0
0000 0001	2^0								1
0000 0010	2^1								2
0000 0011	$2^1 + 2^0$								3
<hr/>									
1111 1110	$2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1$								254
1111 1111	$2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$								255

Het klinkt misschien vreemd maar dit talstelsel is niet veel anders dan het talstelsel waar je je hele leven al mee werkt, het decimale talstelsel. Het grote verschil is dat in het decimale talstelsel je niet alleen 0-en en 1-en hebt maar 0 t/m 9. De structuur is verder nagenoeg hetzelfde. Pak het getal 44723

Dit getal is als volgt opgebouwd

Decimaal	4	4	7	2	3
Omrekenen	4×10^4	4×10^3	7×10^2	2×10^1	3×10^0
Waarde	40000	4000	700	20	3
$40000 + 4000 + 700 + 20 + 3 = 44723$					

In het binair talstelsel lees je altijd van achteren naar voren omdat je er dan vanuit kan gaan dat links van het getal altijd nullen staan.

10 = 2 maar ook 0000000010 = 2

Op het rekenmachine van Windows (Afbeelding 27) kun door op het menu te klikken de ‘programmeren calculator’ openen. Klik om BIN (Binair) en vul 0-en en 1-en in. Je krijgt de waarde dan in de andere talstelsels te zien. Er zijn nog meer talstelsels zoals het Hexadecimaal talstelsel en het Octaal talstelsel. Het Hexadecimaal talstelsel zul je nog regelmatig tegen komen omdat dit talstelsel vaak wordt gebruikt bij bijvoorbeeld het aanduiden van kleuren op een website. Deze talstelsel gaan we nu niet behandelen.



Afbeelding 27 – Calculator

Nu weer terug naar de bits en bytes waar we het eerder over hadden. Je hebt gelezen dat een computer alleen maar 0-en en 1-en begrijpt dus je zult als programmeur met deze 0-en en 1-en moeten leren werken. Gelukkig wordt dit voor jou door Visual Studio voor het grootste gedeelte geregeld al kom je de termen nog wel regelmatig tegen. Zo is dat ook het geval in de volgende paragraaf waar er wordt gesproken over Datatypes.

11.3 Datatypes

In hoofdstuk 5 heb je al gelezen dat je bij het programmeren gebruik maakt van verschillende datatypes. Maar waarom is het nu zo belangrijk om hier rekening mee te houden? De computer kan niet zelf bepalen wat een gebruiker typt. Typt de gebruiker '1988' dan kan dit een getal zijn, een jaartal of gewoon een tekst waar verder niets mee wordt gedaan. Het is dus belangrijk om in je programma aan te geven met wat voor type waarde er wordt gewerkt zodat de computer weet hoe hier mee om te gaan. Daarnaast is het ook belangrijk dat er voldoende geheugen wordt gereserveerd om de waarde in op te kunnen slaan.

Het gebruik van de juiste datatypes is dan ook erg belangrijk. In onderstaande tabel staat een overzicht van een aantal veelgebruikte datatypen. Later zul je nog kennis maken met andere datatypen.

Weetje

Een deel van een processor is speciaal gericht op het rekenen met zwevende-kommagetallen (floating points). Dit zorgt voor veel snelheidswinst als er complexe berekeningen worden gemaakt in een applicatie.

Datatype	Omschrijving	Omvang	Waarde	
Tekst				
string	Tekenreeks		2 bytes per karakter	
Hele getallen	Positief / negatief		Minimum	Maximum
byte	Positief	1 byte	0	255
sbyte	Positief en negatief	1 byte	-128	127
short	Positief en negatief	2 bytes	-32.768	32.767
ushort	Positief	2 bytes	0	65.535
Integer (int)	Positief en negatief	4 bytes	-2^{-31}	2^{31}
long	Positief en negatief	8 bytes	-2^{-63}	2^{63}
Gebroken getallen				
float	Kommagetal	4 bytes	$-3,4 \times 10^{-45}$	$3,4 \times 10^{38}$
double	Kommagetal	8 bytes	-5×10^{-324}	$1,7 \times 10^{308}$
decimal	Financieel Getal	16 bytes	$-7,9 \times 10^{28}$	$7,9 \times 10^{28}$
Andere types				
char	één teken	2 bytes		
bool	Booleaanse waarde	1 bit	Onwaar (0)	Waar (1)

In de bovenstaande tabel kun je zien dat er drie verschillende typen getallen zijn:

- Gehele getallen
- Gebroken getallen (kommagetallen)
- Financiële Getallen

Ga je werken met getallen in je applicatie dan zul je eerst moeten bepalen;

- Is het een heel- of gebroken getal?
- Hoe groot wordt het getal?
- Komen er ook negatieve waarden voor?

In de tabel zie je een kolom omvang staan waarin staat aangegeven hoeveel bytes er worden gebruikt in het geheugen om de waarde op te slaan. Deze ruimte moet dus groot genoeg zijn om het getal op te kunnen slaan, je weet immers uit het vorige hoofdstuk dat je 1 byte nodig hebt om het getal 255 op te kunnen slaan want dat is binair 1111 1111. Gebruik je een datatype dat te klein is om de waarde in op te slaan, dan zal Visual Studio en foutmelding geven.

Voorbeeld

Stel, in je applicatie sla je in een variabele de leeftijd van een persoon op.

Je kunt dan het volgende concluderen:

- Een leeftijd wordt in hele jaren ingevoerd dus je gaat met hele getallen werken.
- De maximale leeftijd van een persoon ligt nu rond de 120 jaar.
- Een negatieve leeftijd bestaat niet.

Kijk je in de tabel dan zie je dat het juiste datatype in dit geval de **byte** is.

Nu zul je misschien denken dat het verstandig is om altijd voor een double te gaan want dan kun je altijd grote getallen opslaan. Dit is niet verstandig omdat je applicatie dan onnodig veel geheugen gaat gebruiken.

Gebruik je in je applicatie het datatype **int** om de leeftijd op te slaan, dan zal het programma 3 bytes teveel geheugen declareren waar niets mee wordt gedaan. Nu is dit bij kleine applicaties natuurlijk niet zo erg maar je kunt je voorstellen dat bij hele grote applicaties (*en later zeker ook als je gaat werken met databases*) dat dit uiteindelijk kan oplopen tot meerdere megabytes aan onnodig geheugenverbruik. Het is verstandig om jezelf aan te leren direct met de juiste datatypen te werken.

Signed en Unsigned

Je gaat in de toekomst termen Signed en Unsigned vaak tegen komen. Een Signed variabele betekent dat de variabele een 'vlaggetje' (sign) heeft wat aangeeft of het een positief getal of een negatief getal is. Unsigned heeft dit 'vlaggetje' niet wat betekent dat je in deze variabele geen negatieve getallen kunt opslaan. In de tabel In deze paragraaf kun je al een aantal signed en unsigned variabele herkennen. Bij unsigned komt er soms een kleine 'u' voor te staan zoals bij uShort, maar het kan ook voorkomen dat er een s komt te staan voor een signed variabele zoals sByte.

11.4 Omzetten datatypen - getal < - > string

Bij het werken met getallen krijg je zoals je eerder hebt gelezen te maken met verschillende datatypen waar je rekening mee moet houden. Gebruik je in de console de mogelijkheid om een gebruiker iets in te laten voeren met de `Console.ReadLine();`; dan zal C# de ingevoerde waarde altijd zien als een string. Dit is natuurlijk niet handig want de uitkomst van de som `10 + 11` wordt dan `1011` want je hebt eerder geleerd dat het optellen van twee strings in C# betekent dat deze achter elkaar worden gezet. Dit kun je oplossen door je programma de ingevoerde waarde eerst om te laten zetten naar een getal. Hiervoor kun je de **`parse()`** methode gebruiken. Deze methode kijkt naar de ingevoerde waarde en zal deze proberen om te zetten naar een getal.

Onthouden

Zie je in een applicatie een getal op je scherm, dan ziet C# dit als een tekst en niet een getal. Dit kan een String of een Char datatype zijn. Je dient deze waarde dus om te zetten als je hier mee wilt gaan rekenen.

De code om een string bijvoorbeeld om te zetten naar een integer ziet er als volgt uit.

```
int getalInteger = int.Parse("8");
```

De input in de console welke je ophaalt met de methode `Console.ReadLine()` levert ook een string op. Je kunt deze methode dan ook in bovenstaande voorbeeld plakken op de plek waar nu de string "8" staat. Je hebt dan een variabele van het type integer waarin de waarde wordt opgeslagen welke is ingevoerd door de gebruiker. De code komt er dan als volgt uit te zien.

```
Int getalInteger = int.Parse(Console.ReadLine());
```

Om de inhoud van een variabele op het scherm te tonen dient deze ook weer omgezet te worden naar een string. Net zoals de invoer van de gebruiker is ook de output op het scherm altijd een string. Om dit om te zetten kun je de methode **`ToString()`** gebruiken. Een datatype is ook een class met daarin methodes. Een variabele die je declareert met een bepaald datatype bevat dus ook methodes en **`ToString()`** is hier één van. Hieronder een voorbeeld waarin de variabele **`getalFloat`** wordt omgezet naar een string welke in de variabele **`tekstGetal`** komt te staan.

```
float getalFloat = 7,89  
string tekstGetal = getalFloat.ToString();
```

Later in dit hoofdstuk ga je bovenstaande methoden gebruiken bij het maken van een rekenmachine.

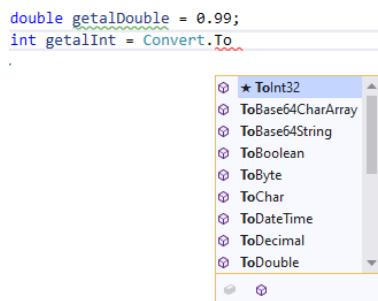
11.5 Omzetten datatypen – getal < - > getal

Naast het omzetten van een tekst naar een getal ga je ook regelmatig datatypen converteren naar een andere datatype. Hiervoor gebruik je de class **Convert** waarin verschillende methodes beschikbaar zijn voor conversie.

De code om een double om te zetten naar een int ziet er als volgt uit.

```
double getalDouble = 7.89  
int getalInt = Convert.ToInt32(getalDouble);
```

In de Convert class zijn verschillende conversiemethoden aanwezig. IntelliSense laat je de hele lijst zien als je in de code Convert aanroeft zoals je in afbeelding Afbeelding 28 kunt zien.



Afbeelding 28 - Convert

Video

In deze [video](#) kun je een voorbeeld zien van het gebruik van de **Parse()** en **ToString()** methoden.

Het is ook mogelijk om de class **Convert** te gebruiken om een getal om te zetten naar een string want de methode **ToString()** uit de vorige paragraaf is ook aanwezig in deze class. De reden dat er toch vaak wordt gekozen voor de methode in de variabele zelf (**getalDouble.ToString()**) is omdat de ontwikkelaar dan in staat is om hier een eigen methode voor te gebruiken met specifieke instellingen voor de conversie. Heeft de ontwikkelaar dit niet gedaan dan wordt alsnog dezelfde methode uit class Convert gebruikt. Het is dus verstandig om altijd de **.ToString()** achter je variabele te gebruiken omdat je dan altijd de juiste methode gebruikt.

11.6 Operatoren

Bij het programmeren ga je veel gebruik maken van berekeningen en vergelijkingen waarin je, net als in de wiskunde, gebruik gaat maken van operatoren. Klinkt ingewikkeld maar dat valt best mee en veel van deze operatoren ken je al heel lang. Bij het programmeren ga je in aanraking komen met drie verschillende typen operatoren.

Rekenkundige operatoren

Bij het gebruik van een rekenkundige operator is de uitkomst altijd een nieuw getal.

teken	Naam	Uitleg
$x * y$	Product	Vermenigvuldigt x met y
x / y	Deling	Deelt x door y.
$x + y$	Som	Telt y op bij x.
$x - y$	Verschil	Trekt y van x af.
$++$	Verhoog	Verhoogt de waarde van het getal met 1
$--$	Verminder	Verlaagt de waarde van het getal met 1.

Relationele operatoren

Bij het gebruik van een relationele operator is de uitkomst altijd de waarde WAAR of NIET WAAR.

teken	Naam	Uitleg
$x == y$	gelijk aan	Kijkt of de waarde van x gelijk is aan de waarde van y.
$x != y$	niet gelijk aan	Kijkt of de waarde van X niet gelijk is aan de waarde van y..
$x > y$	groter dan	Kijkt of de waarde van X groter dan de waarde van y..
$x < y$	Kleiner dan	Kijkt of de waarde van X kleiner is dan de waarde van Y.
$x >= y$	Groter dan of gelijk aan	Kijkt of de waarde van X groter of gelijk is aan de waarde van Y
$x <= y$	Kleiner dan of gelijk aan	Kijkt of de waarde van X kleiner of gelijk is aan de waarde van Y

Logische operatoren

Logische operatoren worden gebruikt om combinaties van uitkomsten toe te staan.

Teken	Naam	Uitleg
$&&$	AND	AND wordt gebruikt om meerdere voorwaarden op te geven voor een vergelijking.
$ $	OR	OR wordt gebruikt om aan te geven dat er meerdere uitkomsten goed zijn.
!	NOT	NOT wordt gebruikt om aan te geven dat het alles mag zijn behalve de opgegeven waarde.

11.7 Opdracht - Rekenmachine

In de voorgaande paragrafen heb je veel gelezen over datatypen, omzetten van getallen en operatoren. Het is nu tijd om daar eens mee te gaan oefenen. Hiervoor ga je een eenvoudige rekenmachine maken in de console.

1. Maak een nieuw project aan voor een Console App en geef deze als naam RekenMachine.
2. De code komt uiteraard weer in de scope van **Main**.

```
static void Main(string[] args)
{
    // Hier ga je de code schrijven
}
```

3. Maak eerst twee variabelen van het type **int** aan en noem deze **getal1** en **getal2**. De variabelen **getal1** en **getal2** moeten door de gebruiker worden ingevoerd dus gebruik **Console.ReadLine()**; als waarde voor deze variabelen.
4. Maak een derde variabele van het type **int** met als naam **uitkomst**. De waarde van **uitkomst** wordt het product (= vermenigvuldiging) van de waarden van **getal1** en **getal2**. Je gaat dus de * operator gebruiken voor deze berekening.
In je code komt dit er als volgt uit te zien.

```
int uitkomst = getal1 * getal2;
```

1. Zorg met een **Console.WriteLine()** dat de gebruiker twee keer de vraag krijgt om een getal in te voeren.
5. Zorg met een **Console.WriteLine()** dat de uitkomst op het scherm komt te staan. De uitkomst wordt door je programma opgeslagen in de variabele **uitkomst** dus deze kun je gebruiken om deze op het scherm te laten zien.
6. Onder in je scherm zul je zien dat er een aantal keer dezelfde foutmelding staat.



Afbeelding 29 - Rekenmachine Console Foutmelding

7. In voorgaande paragrafen heb je geleerd dat een getal op het scherm door C# niet als een getal maar als een string (tekst) wordt gezien. Zodra een gebruiker dus een getal invoert in jouw applicatie moet deze eerst worden omgezet naar een getal en als er iets op het scherm moet worden getoond dan moet het eerst worden omgezet naar een string.
8. Pas je code aan zodat de foutmeldingen verdwijnen en test je programma.

9. Bereken onderstaande uitkomsten met jouw rekenmachine en pas je code zo nodig aan zodat je de berekening kunt maken. (je past dus telkens je rekenmachine aan. Het is niet de bedoeling dat je applicatie onderstaande berekeningen allemaal in één keer kan maken.)
 - a. $15621 * 17821$
 - b. $176352 + 167100$
 - c. $68777 - 71225$
 - d. $3 / 4$
 - e. $23,45 + 78,73$
10. Bij de laatste twee sommen gaat er iets niet goed. De uitkomst van d klopt niet. Deze is niet 0 maar moet 0,75 zijn.
11. Eerder heb je gelezen dat het datatype int een type is dat alleen gebruikt wordt voor gehele getallen. In dit geval is 0,75 kleiner dan 1 dus zal C# 0 weergeven. Het wordt dus niet in afgerond naar boven zoals je dat gewend bent bij rekenen of wiskunde.
12. Bij e gaat het helemaal mis want je programma stopt met werken. Hier heb je hetzelfde probleem. Het programma verwacht een geheel getal aangezien je datatype int gebruikt. De gebruiker voert een kommagetal in en je programma heeft geen instructies gekregen hoe hier mee om te gaan en zal 'crashen'.
13. Beide problemen los je eenvoudig op door te kiezen voor een ander datatype. In dit geval zijn een **Float** of een **Double** de datatypen welke je het best kunt gebruiken.
14. Gebruik hiervoor de *Parse()* methode die is uitgelegd in paragraaf 11.4
15. Pas je programma aan zodat ook **d** en **e** uitgerekend kunnen worden.

? **Het gaat natuurlijk ook fout als de gebruiker een tekst intypt in plaats van een getal. Je leert later hoe je met deze gebruikersfouten om kunt gaan zonder dat je programma crasht.**

11.8 Opdracht - BTW berekenen

De volgende stap is om een iets nuttigere berekening te maken die je later wellicht ook wel eens in een programma gaat gebruiken; het berekenen van de BTW. Om de BTW te berekenen deel je het originele bedrag door 100 en vermenigvuldig je dit met het BTW percentage. Een broek die 200 euro kost zonder BTW en het BTW tarief is 21% dan wordt de som dus $(200 / 100) * 21$ en dit is 42 euro. Tel dit op bij het originele bedrag en je krijgt het bedrag inclusief btw. In dit geval is dat $200 + 42$ euro = 242 euro.

1. Maak eenzelfde programma als in paragraaf 11.7 maar met onderstaande wijzigingen.
 - a. De eerste variabele krijgt de naam **bedragZonderBtw**.
 - b. Deze variabele moet van het type **Decimal** zijn want het gaat om een bedrag.
 - c. De tweede variabele krijgt de naam **btwTarief** en is van het type **Byte**.
 - d. De code welke tussen haakjes komt in de `Console.WriteLine` wordt:

$((bedragZonderBtw / 100) * btwTarief) + bedragZonderBtw$

2. Pas de tekst aan welke op het scherm komt zodat de gebruiker weet wat er ingevuld moet worden wat de uitkomst is.
3. Start je programma en kijkt of het werkt.

11.9 Opdracht - BTW vrije dagen

Bij de Mediamarkt zie je een aantal keer per jaar de **BTW weg ermee** actie waarbij je niet de BTW hoeft te betalen over een bepaald item.

1. Pas je programma van paragraaf 11.8 aan zodat je kunt uitrekenen wat het bedrag is wat je aan de kassa moet betalen als je iets gaat kopen.
(De prijs in de winkel is de prijs inclusief BTW dus $100\% + 21\% = 121\%$. Je dient dus de prijs in de winkel te delen door 121 en de uitkomst vermenigvuldigen met 100 om op de prijs zonder BTW uit te komen.)
2. Zorg ervoor dat je programma ook werkt als in de toekomst het BTW percentage veranderd.



11.10 Eindopdracht - Kassa

Aan de kassa van het restaurant in Developer Land is het natuurlijk niet handig als de kassière ieder bedrag telkens opnieuw moet invoeren. Je gaat een eenvoudig kassa ‘rekenmachine’ maken die uitrekent wat een gezin moet betalen op basis van het aantal producten wat ze kopen.

In onderstaande tabel staan de prijzen van de producten die te koop zijn.

Product	Prijs
Tosti	6,50
Uitsmijter	7,95
Koffie	2,25
Melk	2,00
Frisdrank	2,50

Het programma moet op de volgende manier werken:

- De medewerker kan van ieder item de hoeveelheid invoeren hoeveel tosti's er zijn besteld en kan dit invullen.
- Hierna komt dezelfde vraag maar dan voor het aantal uitsmijters en dit gaat verder tot en met frisdrank.
- Het programma berekent eerst de kosten voor alle tosti's die mogelijk besteld zijn. (indien er geen tosti is besteld kan de medewerker uiteraard 0 invullen.)
- Het totaalbedrag voor de tosti's wordt opgeslagen.
- Hierna wordt hetzelfde gedaan voor alle andere producten.
- Aan het eind worden alle totalen bij elkaar opgeteld en krijgt de medewerker te zien wat de klant moet betalen.

```
Vul aantal Tosti's in:  
2  
Vul aantal Uitsmijters in:  
3  
Vul aantal Koffie in:  
4  
Vul aantal Melk in:  
3  
Vul aantal Frisdrank in:  
1  
Dan wordt het: 54,35 Euro
```

11.11 Wat heb je geleerd?

- Wat een Bit en een Byte zijn.
- Wat een variabele is en waar ze voor dienen.
- Hoe een variabele te declareren.
- Je weet wat een datatype is en waarom er verschillende datatypen worden gebruikt.

- Welke datatype vaak gebruikt worden.
- Je kunt gebruik maken van de methoden Parse() en ToString()
- Je kunt de class Convert inzetten om andere methoden voor conversie te gebruiken.
- Je kunt verschillende operatoren gebruiken.

12 Windows Form - Rekenmachine

In dit hoofdstuk ga je de volgende stap zetten en ga je in C# aan de slag met formulieren of "forms" met als doel een eenvoudige visuele applicatie te gaan maken. Op basis van een aantal opdrachten maak je kennis met verschillende controls, properties en events. De kennis en vaardigheden die je in de voorgaande hoofdstukken hebt opgedaan ga je uiteraard gebruiken om je applicatie te laten werken en je start dan ook met het maken met een gebruikersinterface (UI of User Interface) voor de rekenmachine welke je in het vorige hoofdstuk hebt gemaakt.

12.1 Controls

Controls zijn stukjes programma die je op je formulier kunt gebruiken. Er zijn veel verschillende controls en deze kun je vinden in de Toolbox. In paragraaf 0 staat kort uitgelegd waar je deze Toolbox kunt vinden mocht je dit nog niet weten. In de volgende paragrafen staat een korte uitleg over veelgebruikte controls en staan de belangrijkste eigenschappen of properties beschreven.

Button

	Doel	Een button wordt gebruikt om de gebruiker de optie te bieden om een stukje van het programma te starten.
	Properties	<p>Anchor: Hiermee kun je de knop vastmaken aan één kant van je formulier.</p> <p>TabStop: Wanneer een gebruiker vaak de TAB toets gebruikt kun je hiermee de volgorde aangeven.</p> <p>Enabled: Hiermee kun je de Button aan-/uitzetten. Zo kun je bijvoorbeeld programmeren dat de button pas actief wordt als een gebruiker zijn naam heeft ingevuld.</p>

TextBox

	Doel	De TextBox wordt gebruikt voor het invoeren van gegevens in je applicatie. Je kunt de Textbox ook gebruiken om gegevens weer te geven maar hier heb je ook andere mogelijkheden voor.
	Properties	<p> TextAlign: Hiermee kun je net als in Word aangeven of de tekst links, rechts of in het midden van de TextBox komt te staan.</p> <p> MultiLine: Hiermee kun je aangeven of er meerdere regels in de TextBox gebruikt mogen worden.</p> <p> BorderStyle: Hiermee kun je het uiterlijk van de rand van de TextBox aanpassen.</p> <p> MaxLength: Hiermee kun je het maximaal aantal</p>

		karakters dat ingevoerd kan worden bepalen. Standaard is dit 32767 tekens.
--	--	--

Label

	Doel	Een label wordt gebruikt om een tekst weer te geven op het formulier.
	Properties	AutoSize: Deze optie zorgt ervoor dat het label altijd iets groter is dan de tekst die getypt is zodat wordt voorkomen dat delen van de tekst niet zichtbaar zijn omdat deze buiten het label valt.

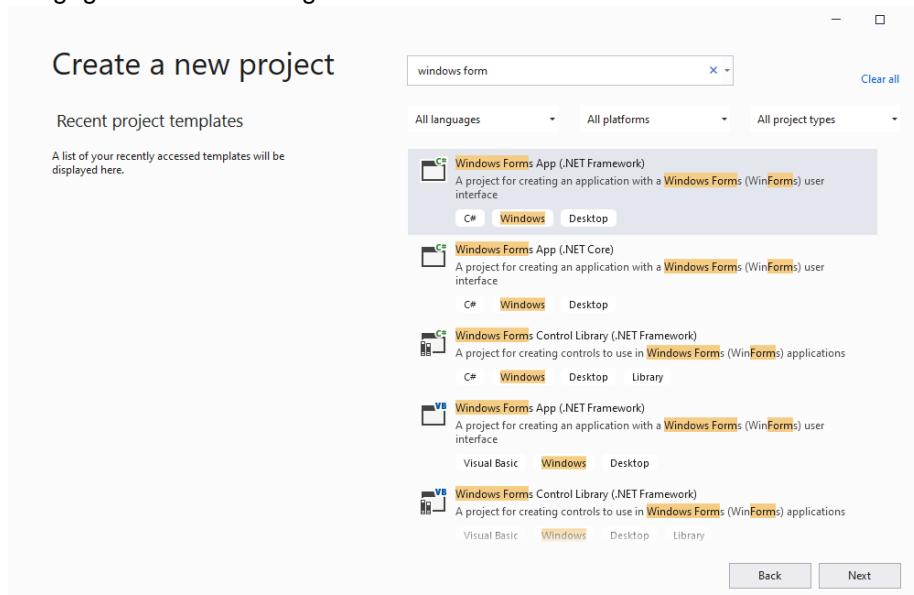
RadioButton

	Doel	Met RadioButtons kun je de gebruiker een enkele keuze geven en deze keuze kun je gebruiken in je applicatie. Bij het aanklikken van één button wordt automatisch de andere uitgezet.
	Properties	Checked: Hiermee kun je aangeven welke RadioButton standaard geselecteerd moet zijn.

12.2 Opdracht - Rekenmachine UI

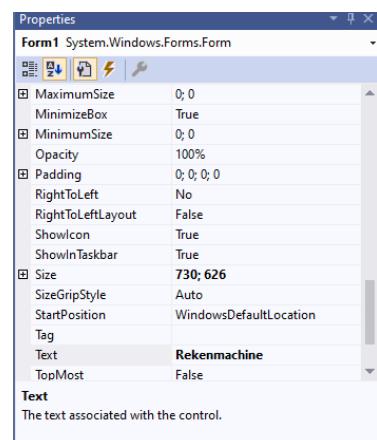
In deze opdracht ga je een UI (User Interface) maken voor de rekenmachine uit het vorige hoofdstuk. De code om deze te laten werken komt pas in het volgende hoofdstuk aan bod dus het gaat nu puur om het uiterlijk.

- Maak een nieuw project aan en kies voor een Windows Form App (.NET Framework) zoals aangegeven in Afbeelding 30.

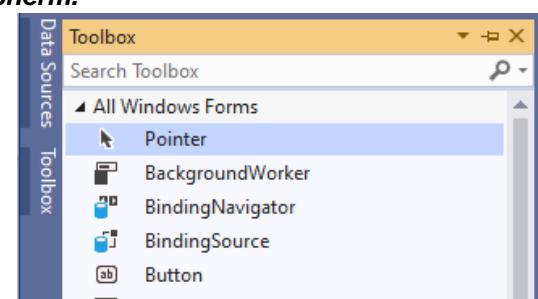


Afbeelding 30 – Windows Forms App

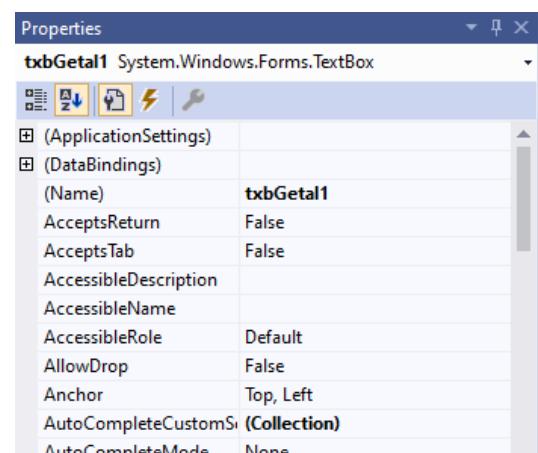
2. Geef het project de naam RekenMachineUI.
3. In het midden van je scherm heb je nu een leeg formulier staan waarop je de controls kunt gaan slepen vanuit de Toolbox.
4. De eerste eigenschap die je gaat aanpassen is de tekst van dit formulier zodat er geen *form1* meer staat maar *Rekenmachine*. Klik op het formulier en ga rechts onder in je scherm naar het **properties venster**.
5. Verander de waarde van de eigenschap Text en geef hier Rekenmachine in zoals aangegeven in Afbeelding 31
6. Bovenin je formulier zul je nu de tekst Rekenmachine zien staan.
7. Later in de opleiding ga je grotere programma's ontwikkelen waarin je gebruik maakt van meerdere formulier. Het is dan ook belangrijk om het formulier een duidelijke naam te geven zodat je weet welk formulier waarvoor gebruikt wordt.
8. Pas de naam van het formulier aan naar ***frmHoofdscherm***.
9. De volgende stap is toevoegen van de controls. Deze controls kun je vinden in de Toolbox aan de linkerkant van het scherm. Zie je de controls niet staan klik dan in de linker balk op Toolbox en het venster zal verschijnen. Zodra deze zichtbaar is kun je op de punaise bovenin dit venster klikken zodat deze zichtbaar blijft.
10. Zoek naar de control TextBox en sleep twee TextBoxes naar je formulier zodat deze in het midden onder elkaar komen te staan.
11. Om een goed overzicht te houden van alle controls op je formulier is het belangrijk om de naam van de control te wijzigen. Nu heb je twee TextBoxes staan maar daar komen nog buttons bij. Ga je later de code aanpassen dan weet je op den duur bijvoorbeeld niet meer waar Button1 of Button9 voor wordt gebruikt.
12. Pas de naam van de twee TextBoxes aan en geef de bovenste de naam ***txbGetal1*** en de onderste ***txbGetal2***. Door te beginnen met ***txb*** weet je straks in je code dat het om een TextBox gaat en Getal1 geeft duidelijkheid wat het doel is van de TextBox.
13. Plaats nu vier buttons op het formulier voor optellen, aftrekken, vermenigvuldigen en delen.
14. Pas de naam van de buttons aan waarbij de naam begint met ***btn*** zodat je later in je code kunt zien dat deze control een Button is en daarachter een logische benaming voor de functie van de button. Voor optellen is dit bijvoorbeeld ***btnOptellen***.



Afbeelding 31

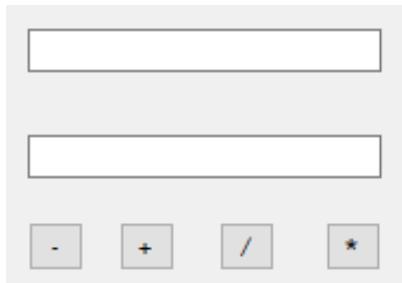


Afbeelding 32



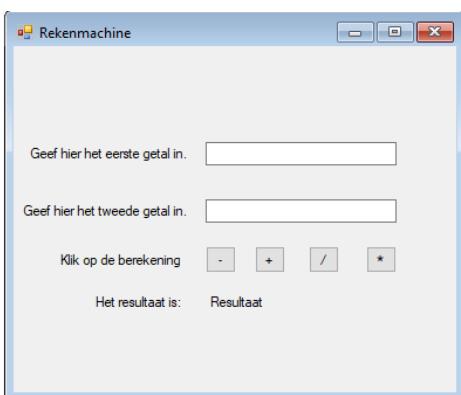
Afbeelding 33

15. Op de Buttons staat nu de tekst Button 1, Button 2 etc. Dit is natuurlijk niet handig. Deze tekst kun je aanpassen door de property Text van iedere button te wijzigen.
16. Zorg ervoor dat op de juiste buttons de tekens voor optellen (+) , aftrekken (-), vermenigvuldigen (*) en delen (/) komen te staan.
17. Pas nu alle controls aan zodat het formulier eruit komt te zien zoals in Afbeelding 34



Afbeelding 34

18. Voeg nu 5 labels toe zodat je applicatie eruit komt te zien als in Afbeelding 35. Let op dat de tekst Resultaat onder de knoppen een los label is.



Afbeelding 35

19. Zorg ervoor dat de namen van de labels logisch zijn en beginnen met **Ib1** zodat je later in de code weet met welke control je aan het werk bent. Je kunt bijvoorbeeld **Ib1InfoGetal1** gebruiken. Op deze manier weet je dat het om het label gaat die voor de TextBox staat behorende bij getal 1.

20. Pas met behulp van de properties de volgende onderdelen aan:

- a. De achtergrond van je rekenmachine krijgt een witte kleur.
- b. De tekst van de labels krijgt het font Arial en wordt dikgedrukt weergegeven.
- c. Het resultaat wordt weergegeven in een lettergrootte van 12 punten.
- d. Zorg ervoor dat de * duidelijker wordt door deze groter te maken en zorg dat alle tekst in de knoppen dikgedrukt is.
- e. Pas de TabIndex aan zodat bij het starten van de applicatie txbGetal1 standaard geselecteerd is. (TabIndex 0).
- f. Pas de TabIndex aan zodat txbGetal2 als tweede wordt geselecteerd zodra op de Tab toets wordt gedrukt.
- g. Bij iedere volgende druk op de Tab toets wordt de volgende knop geselecteerd waarbij je de volgorde aanhoudt van links naar rechts.

De UI van je rekenmachine is nu klaar. In de volgende paragrafen ga je aan de slag met het schrijven van de code om de rekenmachine te laten werken.

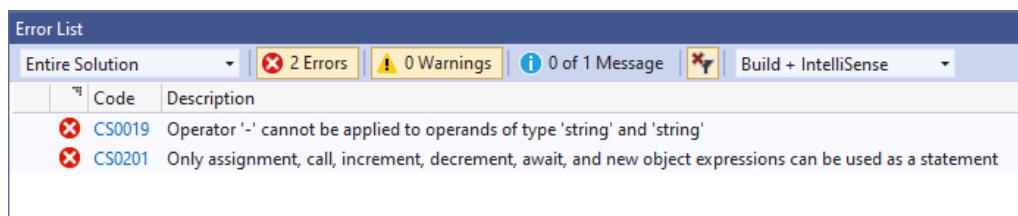
12.3 Opdracht - Events

Bij het programmeren van een applicatie ga je vaak gebruik maken van Events. Een event is, zoals de naam al aangeeft, een gebeurtenis waardoor je applicatie een stuk code gaat uitvoeren. Een voorbeeld van een event is het klikken op een knop door de gebruiker.

1. Dubbelklik op de **btnMin** knop. De code editor zal verschijnen en er wordt automatisch een event **btnMin_Click** aangemaakt. Tussen de accolades (de **Scope** van dit event) ga je de code plaatsen welke wordt uitgevoerd op het moment dat er op de **btnMin** wordt geklikt.

```
private void btnMin_Click(object sender, EventArgs e)
{
}
```

2. Zodra er op deze knop wordt geklikt wil je dat de waarde in de **txbGetal2** wordt afgetrokken van de waarde in **txbGetal1**.
3. De waardes welke je in de TextBox ziet worden opgeslagen in de property **Text**.
4. Net als in de console applicatie welke je in voorgaande hoofdstukken hebt gemaakt kun je de waarde van de eigenschap van een object ophalen in je code door gebruik te maken van de syntax **<objectnaam>.<property>**. In dit geval wordt dat **txbGetal1.Text** en **txbGetal2.Text**.
5. De code om deze twee waarden van elkaar af te trekken wordt dan:
txbGetal1.Text - txbGetal2.Text;
6. Je zegt hiermee ‘trek de waarde van de property **Text** van **txbGetal2** af van de waarde van de property **Text** van **txbGetal1**’.
7. Voer deze code in tussen de accolades van het event **btnMin_Click**.
8. IntelliSense helpt je weer met typen en je zult zien dat de naam die jij hebt gegeven aan de knoppen of textboxes in de lijst staan.
9. Na het invoeren van de code valt je waarschijnlijk al op dat er rode kringeltjes onder de code staan.
txbGetal1.Text - txbGetal2.Text;
10. Dit betekent dat er iets niet goed is in de code. De error melding kun je terugvinden in de **Error List** onderin je scherm. Je ziet daar twee foutmeldingen staan.



Afbeelding 36

11. De eerste foutmelding geeft aan dat je de ‘-‘ **operator** niet kunt gebruiken in combinatie met het datatype **String**.
12. De tweede foutmelding krijg je omdat de code nog niet compleet is. Je laat het programma iets berekenen maar je geeft niet aan wat hier mee moet gebeuren.
13. Je begint eerst met het oplossen van de tweede foutmelding door de code af te maken.
14. De uitkomst van de berekening moet worden getoond in het label **lblResultaat** van je rekenmachine.
15. Om hiervoor te zorgen dien je de property **Text** van object **lblResultaat** aan te passen. Dit doe je door je berekening toe te wijzen (=) aan de property **lblResultaat.Text** net zoals je dat met een variabele doet. In onderstaande code kun je dit zien. Neem onderstaande code over in je applicatie.

```
lblResultaat.Text = txbGetal1.Text - txbGetal2.Text;
```

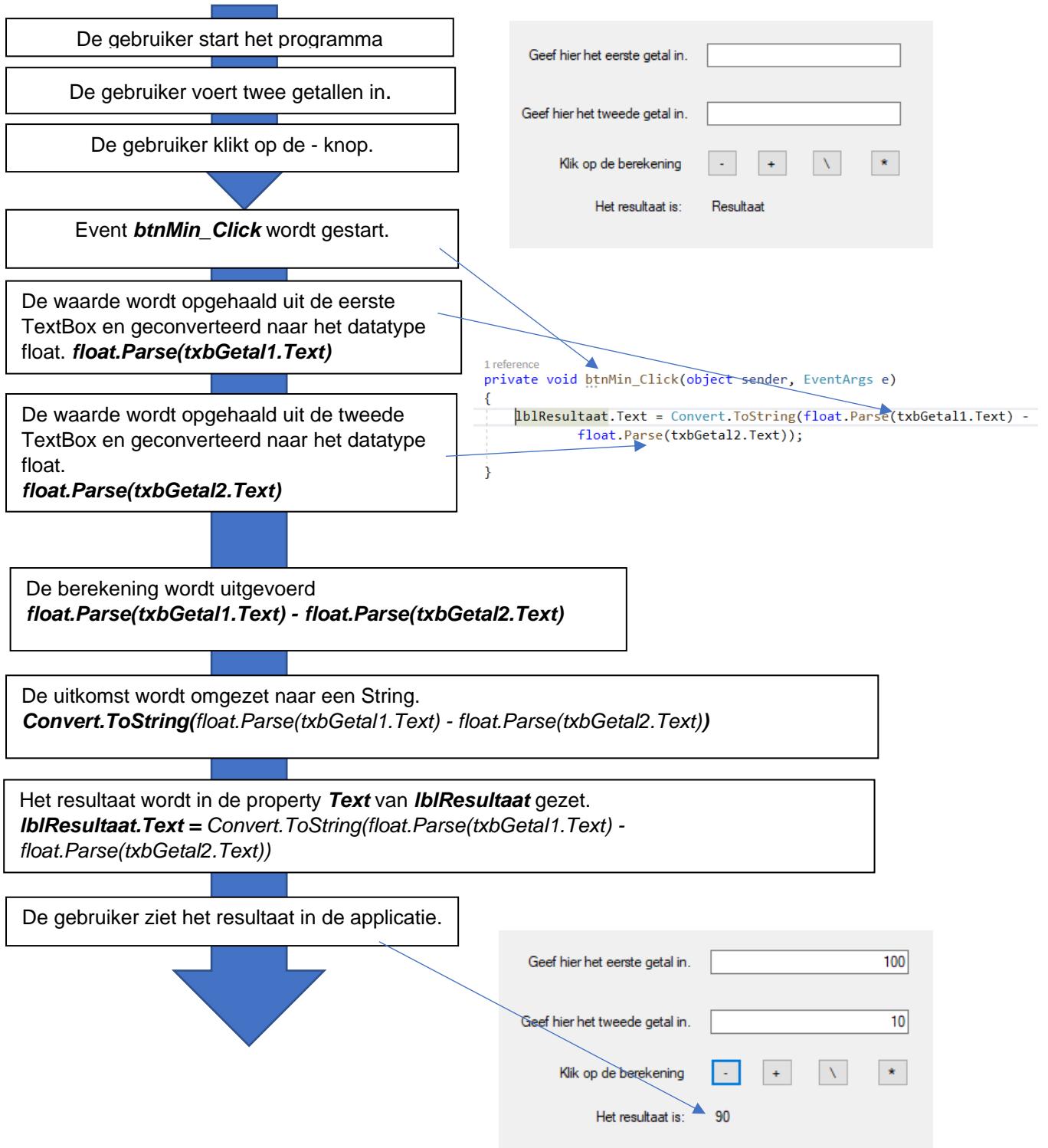
16. Bovenstaande code geef je aan dat de property **Text** van **lblResultaat** de waarde moet krijgen van de uitkomst van **txbGetal1.Text - txbGetal2.Text**.
17. Met deze aanpassing zie je dat de tweede foutmelding in de errorlist is verdwenen.
18. De eerste foutmelding lijkt op foutmeldingen die je bij eerdere opdrachten al eens tegen bent gekomen. De property **Text** is zoals de naam al zegt een **String**. De applicatie verwacht echter een getal op het moment dat je de – **operator** gebruikt.
19. Hiervoor kun je dan ook weer de **Parse()** methode inzetten om de waarde te converteren naar een getal. Aangezien de kans bestaat dat er een kommagetal wordt ingevoerd, ga je beide waarden afzonderlijk van elkaar omzetten naar het datatype **float** of **double**. Gebruik hiervoor onderstaande code.

```
lblResultaat.Text = float.Parse(txbGetal1.Text) - float.Parse(txbGetal2.Text);
```

21. Je zult zien dat er na deze wijziging een andere error melding verschijnt.
22. Er staat nu dat de applicatie niet in staat is om de uitkomst, een getal, in **lblResultaat.Text** te plaatsen. Dit komt omdat het programma een tekst (String) verwacht en niet een float.
23. Dit los je op door de uitkomst weer om te zetten naar een String met behulp van de methode **ToString()**.
24. Deze methode is onderdeel van de class **Convert** welke onderdeel is van de System Library die standaard wordt toegevoegd aan je applicatie als je deze gaat bouwen.
25. Om deze methode te gebruiken typ je **Convert.ToString()**; en tussen de haakjes van de methode komt je hele berekening te staan.
Pas de code aan zoals hieronder aangegeven.

```
lblResultaat.Text = Convert.ToString(float.Parse(txbGetal1.Text) - float.Parse(txbGetal2.Text));
```

26. In het volgende schema staat aangegeven hoe jouw applicatie werkt.



27. Als je alles goed hebt overgenomen heb je geen errors meer in je lijst staan en is het tijd voor een eerste test.
28. Start je programma en test of je twee getallen van elkaar af kunt trekken.
29. Pas nu je programma met behulp van bovenstaande stappen zodat ook de het optellen, vermenigvuldigen en delen werkt.
30. Dat kost niet veel tijd.....toch?

Een andere manier...

31. De code welke in deze opdracht wordt gebruikt ziet er ingewikkeld uit. Je kunt deze code ook op een andere manier opbouwen door gebruik te maken van variabelen. Je kunt dan de **geel** en **groen** gemaakte code in twee variabelen plaatsen.

```
lblResultaat.Text = Convert.ToString(float.Parse(txbGetal1.Text) - float.Parse(txbGetal2.Text));
```

32. Maak in het event van **btnMin_Click** twee variabelen aan van het type float met als naam **getal1** en **getal2**.
33. Gebruik de geel en groen gemaakte code om de variabelen te vullen.
34. Nu heb je twee variabelen waar de juiste waarden in worden opgeslagen dus kun je de geel en groen gemaakte code vervangen door deze twee variabelen en krijg je:

```
float getal1 = float.Parse(txbGetal1.Text);
float getal2 = float.Parse(txbGetal2.Text);
lblResultaat.Text = Convert.ToString(getal1 - getal2);
```

35. Test je programma en kijk of het werkt.
36. Het voordeel van de eerste methode (zonder variabelen) is dat je in totaal minder code gebruikt voor dezelfde handeling en iets minder geheugen verbruikt. Door wel gebruik te maken van variabelen wordt je code een stuk overzichtelijker. Voor nu kun je zelf een keuze maken wat je het prettigst vindt. Later zul je tegen situaties aanlopen waarbij je verplicht één van deze opties moet gebruiken.

12.4 Opdracht – Foutafhandeling

Je hebt nu een eenvoudig maar werkende rekenmachine gemaakt en ondanks dat je code goed is kan het toch gebeuren dat je applicatie crasht. Voer onderstaande stappen maar eens uit.

1. Start je applicatie en typ in het eerste veld het woord **honderd**.
2. Typ in het tweede veld het woord **tien**.
3. Tel deze twee bij elkaar op en kijk wat er gebeurt.
4. Zodra je op de - knop klikt crasht je programma en krijg je een errormelding.

De kans bestaat altijd dat een gebruiker niet juist omgaat met je applicatie en als ontwikkelaar dien je hier dan ook rekening mee te houden. Je hoeft natuurlijk niet ver te zoeken om te bepalen waarom dit fout gaat. Je voert geen cijfers in maar letters. De applicatie probeert letters om te zetten in een getal en dat gaat natuurlijk niet. De applicatie weet niet wat er moet gebeuren en crasht. Je kunt als ontwikkelaar ervoor

zorgen dat dit soort fouten worden afgevangen of hoe je de gebruiker erop kunt wijzen dat deze iets niet goed heeft gedaan.

In onderstaande stappen leer je hoe je een pop-up venster te zien krijgt op het moment dat er iets fout gaat zonder dat je applicatie direct crasht. Hierbij ga je gebruik maken van **Try Catch** blokken.

Een Try ... Catch blok ziet er als volgt uit.

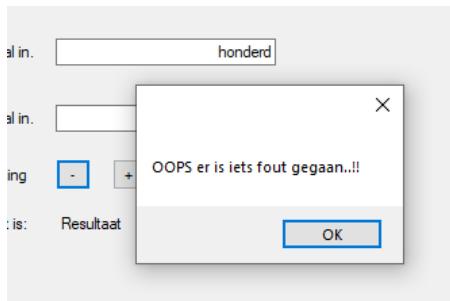
```
try
{
}
catch (Exception)
{
    throw;
}
```

- In de scope (*dus tussen de accolades*) van **Try** wordt het stuk code geplaatst wat uitgevoerd moet worden maar mogelijk fout kan gaan.
- In de scope (*dus tussen de accolades*) van **Catch (Exception)** komt de code die wordt uitgevoerd zodra er een fout wordt gemaakt.

1. Plaats een Try ... Catch blok tussen de accolades van het **btnMin_Click** event van jouw applicatie door **Try** te typen en hierna snel tweemaal op de tabtoets te drukken.
2. De code welke moet worden uitgevoerd als het event wordt gestart (dus bij het klikken op de knop) komt in het **Try** blok.
3. Knip de code welke je al had staan en plak deze in de scope van **Try** zoals hieronder aangegeven.

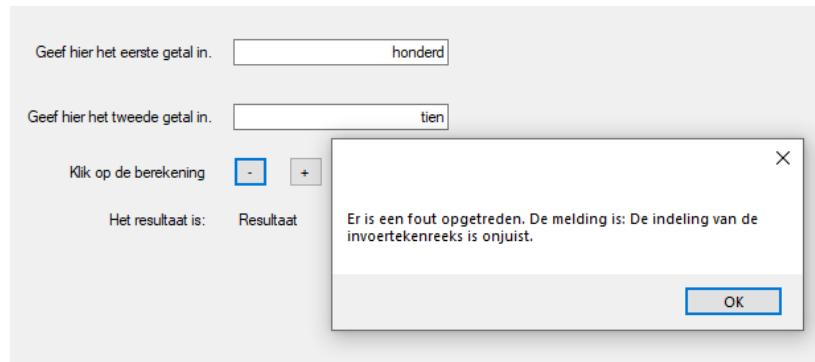
```
try
{
    float getal1 = float.Parse(txbGetal1.Text);
    float getal2 = float.Parse(txbGetal2.Text);
    lblResultaat.Text = Convert.ToString(getal1 - getal2);
}
```

4. Voer het programma nog een keer uit en kijk wat er gebeurt als je weer de woorden honderd en tien invult.
5. Je programma crasht nog steeds met een foutmelding.....
6. De reden hiervoor is dat er in de scope van **Catch** een **throw** staat die de foutmelding letterlijk over de schutting gooit en ervanuit gaat dat er een onderdeel is in je applicatie die deze melding 'vangt' en er iets mee gaat doen. Het schrijven van zo'n onderdeel in je applicatie is voor nu te complex dus ga je het simpeler op te lossen door de throw te vervangen door een andere opdracht die uitgevoerd moet worden als er iets misgaat.
7. Een goede optie is om hiervoor het object **MessageBox** te gebruiken. De **MessageBox** is een pop-up venster welke je waarschijnlijk wel kent.



Afbeelding 37 - MessageBox

8. De MessageBox kun je tevoorschijn laten komen met een eigen tekst zoals in Afbeelding 37, maar het is ook mogelijk om in de MessageBox de foutmelding te laten zien. Dit kan natuurlijk handig zijn om bijvoorbeeld fouten te achterhalen als je de applicatie laat testen door gebruikers. Je kunt ze dan vragen welke foutmelding ze krijgen en dan kun jij daarmee aan de slag.
 9. De foutmelding die je krijgt is de **Exception** welke je ook in het Try...Catch blok ziet staan want daar staat **Catch (Exception)**
 10. Om deze foutmelding te kunnen gebruiken moet deze worden opgeslagen in een variabele. Exception is van zichzelf al een variabele maar je moet deze wel een naam geven zodat je het kunt gebruiken in je code. Net als bij andere variabelen zet je de naam van de variabele achter het type.
 11. Verander de code van **catch (Exception)** naar **catch (Exception Foutmelding)**. Hiermee geef je aan dat de foutmelding (Exception) die Visual Studio genereert moet worden opgeslagen in de variabele **foutmelding**.
 12. De volgende stap is een MessageBox te laten verschijnen waarin deze foutmelding wordt getoond zodra het programma crasht. Om dit te regelen vervang je de **throw** door jouw eigen code.
 13. Vervang in de scope van catch de code **throw;** door onderstaande code.
- ```
MessageBox.Show("Er is een fout opgetreden. De melding is: " + foutmelding.Message);
```
14. Bovenstaande code begint met een **MessageBox.Show**
  15. Een **MessageBox** is een object welke beschikbaar is in de system library welke je tot je beschikking hebt.
  16. **Show()** is een methode van het object MessageBox welke ervoor zorgt dat het pop-up venster verschijnt.
  17. Tussen de ( ) komt te staan wat er in het venster moet komen te staan. In dit geval is dit een stuk tekst en de melding welke wordt opgehaald met **foutmelding.Message**. (*Message is een property van de variabele foutmelding waarin de meldingstekst is opgeslagen. Dit kun je vergelijken met de text property van een textbox.*)
  18. Test je programma door wederom de woorden honderd en tien in te vullen in de TextBoxes en op de – knop te klikken.
  19. Als je alles goed hebt gedaan is het resultaat gelijk aan Afbeelding 38.



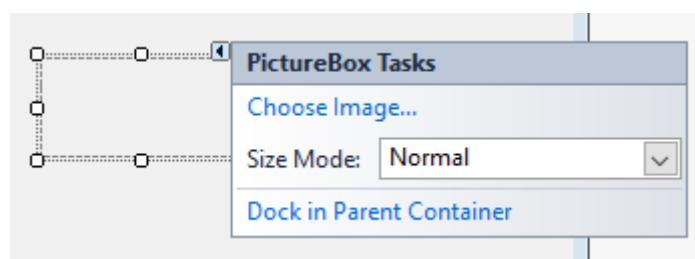
Afbeelding 38 - MessageBox foutmelding

22. Pas nu de rest van de code van je rekenmachine aan zodat ook een MessageBox verschijnt bij het optellen, delen en vermenigvuldigen als de gebruiker geen getallen maar tekst intypt.
23. Test je applicatie en ga door met de volgende opdracht.

## 12.5 Opdracht – Personaliseren met PictureBox

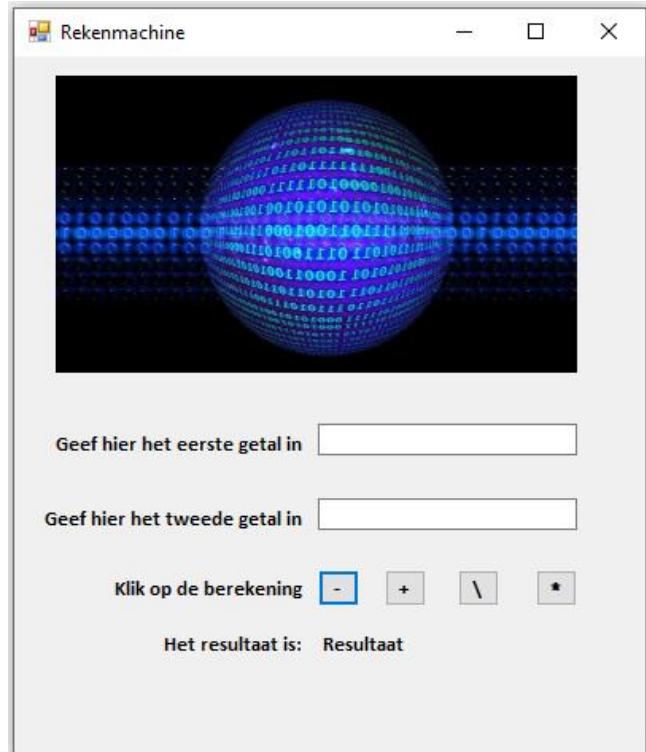
De rekenmachine is nu klaar maar je kunt er nog een persoonlijk tintje aan geven door er **PictureBox** aan toe te voegen. Een **PictureBox** is een plek in je formulier waar je een afbeelding kunt laden.

1. Voeg een **PictureBox** toe aan je formulier.
2. Klik op het kleine pijltje rechts bovenin de PictureBox



Afbeelding 39 - PictureBox

3. Klik op **Choose Image**.
4. Selecteer een afbeelding die je op je pc hebt staan.
5. Kies bij **Size Mode**: voor **Stretch** zodat het afbeelding automatisch wordt aangepast aan de grootte van de PictureBox. Let op dat de afbeelding er vervormt uit kan zien als de verhoudingen niet overeenkomen met de PictureBox.
6. Pas de rest van je Rekenmachine aan zodat het uiterlijk (met je eigen afbeelding uiteraard) gelijk is aan Afbeelding 40.



Afbeelding 40 - Rekenmachine

## 12.6 Eindopdracht - BTW

Je gaat in deze opdracht de laatste functionaliteit toevoegen aan je rekenmachine.

- Voeg een knop toe voor het uitreken van het bedrag van een artikel als je geen btw hoeft te betalen zoals in paragraaf 11.9.
- Pas de tekst van de labels aan zodat het duidelijk is dat je bij het eerste veld een getal of bedrag kunt invullen en bij het tweede veld een getal of percentage.
- Zorg ervoor dat zodra er op de knop BTW wordt geklikt bij het resultaat het bedrag komt te staan wat je moet betalen tijdens de BTW vrije dagen.
- Test je applicatie.
- Maak een screenshot van jouw code en plaats deze op It's Learning.

## 12.7 Wat heb je geleerd?

- Je kunt een Windows Forms applicatie maken.
- Je weet wat een Event is.
- Je weet wat Controls zijn.
- Je kunt de eigenschappen van controls aanpassen.
- Je weet wat foutafhandeling is en je kunt deze in de basis toepassen.

# 13 Introductie Methoden

---

Eerder heb je al iets gelezen over methoden wat functionaliteiten zijn van een object of class. Eigenlijk is een methode een stuk code binnen een class welke meerdere keren gebruikt en ‘aangeroepen’ kan worden. Dit aanroepen heb je al meerdere keren gedaan met bijvoorbeeld de methode **ReadLine()** van de **Console** om ervoor te zorgen dat de gebruiker een tekst kan invoeren.

Naast al deze methoden die je in C# al tot je beschikking hebt kun je ook zelf methoden schrijven waarmee je dus efficiënter kunt programmeren want je gaat dan minder code gebruiken. Dit is vooral nuttig als je later echt object georiënteerd gaat programmeren. Ook nu kun je al gebruik maken van methoden.

Je gaat in dit hoofdstuk je rekenmachine applicatie hiervoor inzetten uit paragraaf 11.7. Dit is natuurlijk een mooi voorbeeld van een class (want deze applicatie bestaat uit één class) met daarin meerdere methoden voor de functionaliteiten optellen, aftrekken, vermenigvuldigen en delen.

Er zijn twee typen methoden welke je kunt aanmaken.

## 13.1 Methoden die iets doen

Een methode die iets doet herken je aan het woordje **void** en deze methode slaat niets in het geheugen op. Kijken we terug op de AudiA8 dan zou een methode **Remmen()** een goed voorbeeld zijn.

```
static void Remmen()
{
}
```

Deze methode voert iets uit maar geeft geen waarde terug.

## 13.2 Methoden die een vraag beantwoorden

Een methode die een vraag beantwoordt is een methode die dus waarde(n) teruggeeft welke je verder in je applicatie kunt gebruiken. Er wordt dus ook een stukje geheugen gedeclareerd waarin de waarde wordt opgeslagen. Bij dit type methode dien je dus ook aan te geven om wat voor datatype welke teruggeven gaat worden. In de Audi A8 zou een bijvoorbeeld **Snelheid()** een methode kunnen zijn welke een waarde teruggeeft.

```
static int snelheid()
{
 return: 100;
}
```

De waarde die je terugkrijgt van deze methode kan dan in een ander (deel van je) programma gebruikt worden. Denk hierbij aan de snelheidsmeter die deze waarde laat zien op je dashboard.

### 13.3 Aanmaken van een methode

Je gaat voor de rekenmachine welke je eerder hebt gemaakt methoden schrijven.

#### Stap 1:

Bij het aanmaken van een methode bepaal je eerst of de methode iets terug moet geven zoals bijvoorbeeld de uitkomst van een berekening of dat de methode alleen iets voor je moet uitvoeren.

- Gebruik **static void** voor methoden die alleen iets uitvoeren en niets teruggeven.
- Gebruik een type aanduiding voor methoden die een waarde teruggeven (bijv. **int**, **string**, **double**, **bool** etc.)

In het geval van de rekenmachine willen we een waarde terug krijgen dus wordt er gekozen voor een type aanduiding.

#### Stap 2:

Je geeft de methode een logische naam waaruit af te leiden is wat de methode doet. Bij voorkeur is dit een werkwoord.

```
static double Optellen()
{
}
```

#### Stap 3:

Bepaal of je parameters mee wilt geven bij het aanroepen van de methode.

Vaak worden bij methoden zogenaamde argumenten of parameters gebruikt die je mee kunt geven op het moment dat je een methode aanroeft. De methode kan deze waarden dan gebruiken in de code om bijvoorbeeld een berekening uit te voeren. Deze parameters worden geplaatst tussen de haakjes van de methoden en worden net als een variabele gedeclareerd dus met het datatype ervoor. Hieronder staat een voorbeeld van een methode Optellen() waaraan de **parameters** zijn toegevoegd.

```
static double Optellen(double getal1, double getal2)
{
}
```

Je gebruikt dus parameters in een methode om ervoor te zorgen dat een ander deel van de applicatie waarden mee moet kunnen geven waar de methode iets mee moet doen.

#### Stap 4:

Schrijf de code welke moet worden uitgevoerd bij het aanroepen van de methode en plaats deze in de scope van de methode. In het geval van de methode Optellen() moeten de twee getallen bij elkaar opgeteld worden dus wordt **de code** als volgt.

```
static double Optellen(double getal1, double getal2)
{
 double uitkomst = getal1 + getal2;
}
```

## Stap 5:

Moet de code een waarde teruggeven dan gebruik je een `return` om deze waarde terug te geven aan de applicatie. Voor de methode optellen() wordt dit:

```
static double Optellen(double getal1, double getal2)
{
 double uitkomst = getal1 + getal2;
 return uitkomst;
}
```

## 13.4 Aanroepen Methode

Na het doorlopen van de stappen uit de vorige paragraaf heb je een methode welke je kunt gebruiken in je code. Wil je deze methode in een ander deel van je applicatie aanroepen dan kun je onderstaande voorbeeld code gebruiken.

Bijvoorbeeld:

`Optellen(variabele1, variabele2)`

of

`Optellen(100, 50);`

- In dit laatste voorbeeld wordt de methode aangeroepen met vaste waarden en worden de getallen 100 en 50 meegegeven als parameters. We gebruiken dit voorbeeld om het verloop uit te leggen. Let op dat je zelf altijd met variabelen werkt.
- Deze parameters worden dus in getal1 en getal2 geplaatst.

```
private void btnPlus_Click(object sender, EventArgs e)
{
 Optellen(100, 50);
}

static double Optellen(double getal1, double getal2)
{
 double uitkomst = getal1 + getal2;
 return uitkomst;
}
```

- De waarde wordt teruggegeven aan de applicatie met `return` wordt 150 en dat is te zien als je de variabelen even vervangt door de waarden die worden gebruikt

```
static double Optellen(100, 50)
{
 double uitkomst = 100 + 50;
 return 150;
}
```

- De waarde die je terugkrijgt van de methode is van het datatype **Double** want dit heb je aangegeven

```
static double Optellen(double getal1, double getal2)
{
 double uitkomst = getal1 + getal2;
 return uitkomst;
}
```

- Om deze waarde in de **TextBox** weer te geven dien je deze dus om te zetten naar een string. De code welke je in de event van je knop kunt gebruiken wordt dan: (let op, de getallen moeten variabelen worden in je daadwerkelijke code).

```
lblResultaat.Text = (Optellen(100, 50)).ToString();
```

## 13.5 Opdracht - RekenMachine aanpassen

Je gaat nu de functionaliteiten uit de rekenmachine die je hebt gemaakt in het vorige hoofdstuk omzetten naar methoden.

1. Maak een kopie van de projectmap van je RekenMachine uit het vorige hoofdstuk, hernoem de kopie naar **RekenMachineMethode** en open de **RekenMachine.SLN** in deze map.
2. Hernoem je solution naar **RekenMachineMethode** door met de rechter muis op de solution naam te klikken en te kiezen voor *rename*.
3. Maak de methode **Optellen()** zoals in de vorige paragraaf besproken is en plaats deze onder het laatste event in je code maar wel binnen de scope van de class. Hieronder staat aangegeven waar je de code moet plaatsen.

De **groen** gemarkeerde { en } geven de scope van de namespace aan.

De **geel** gemarkeerde { en } geven de scope van de class aan.

De **roze** gemarkeerde { en } geven de scope van een event in je code.

Het **blauwe blok** geeft aan waar je de methode moet aanmaken.

```
namespace RekenMachine
{
 public partial class frmHoofdscherm : Form
 {
 private void btnVermenigvuldigen_Click(object sender, EventArgs e)
 {
 }

 private void btnDelen_Click(object sender, EventArgs e)
 {
 }

 Hier komt je methode
 }
}
```

4. Maak ook de een methode aan voor **Aftrekken()**, **Vermenigvuldigen()** en **Delen()**.
5. Pas al je events aan zodat ze gebruik maken van de methodes welke je zojuist hebt aangemaakt om de berekening uit te voeren.

## 13.6 Opdracht - Het nut van methoden

Je hebt in voorgaande opdracht gezien dat het gebruik van methoden nog niet veel heeft opgeleverd voor jouw applicatie. De hoeveelheid code is niet minder geworden en het ziet er wellicht nog complexer uit dan het al was. Dit komt omdat je eigenlijk nog niet echt een hele complexe applicatie hebt gemaakt waarbij je de methode meerdere keren kunt gebruiken. Later in de opleiding ga je hier nog uitgebreid mee aan de slag op het moment dat de applicaties complexer worden en je ook met meerdere classes gaat werken.

Nog even een paar voordelen op een rij om het nut van methoden toe te lichten.

- Bij complexe applicaties kun je code hergebruiken zodat je niet elke keer dezelfde code hoeft te typen.
- Je code staat op één plaats dus bij wijzigingen hoeft je de code ook maar op één plaats aan te passen. Dit scheelt tijd maar het voorkomt ook fouten.

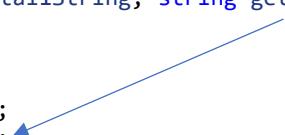
In je rekenmachine applicatie heeft het gebruik van de methoden niet echt nut. Alle methoden zijn losse berekeningen en je gebruikt ze maar één keer in je code. Wat je wel vaker gebruikt is de foutafhandeling met je try...catch blok. Dit kun je in een losse methode plaatsen maar om dit op een echt nette manier te doen is op dit moment nog te complex. Om toch aan te tonen dat het nuttig is ga je het op een wat minder nette manier uitvoeren.

1. Maak wederom een kopie van je rekenmachine, pas de naam van de map aan naar RekenMachineFoutAfhandeling en verander de naam van je Solution ook naar RekenMachineFoutAfhandeling.
2. Met de foutafhandeling (try...Catch) controleer je eigenlijk alleen of de waarden in de twee TextBoxen omgezet kunnen worden naar het datatype Double. Kan dit niet dan krijgt de gebruiker een popup venster. Deze controle kun je in een methode stoppen zodat je niet in iedere functionaliteit een Try Catch blok hoeft te plaatsen.
3. Maak een methode aan met als naam `FoutAfhandeling()` en gebruik hiervoor onderstaande code.

```
static (double, double) FoutAfhandeling(string getal1String, string getal2String)
{
 double getal1 = double.Parse(getal1String);
 double getal2 = double.Parse(getal2String);
}
```

4. Je ziet dat dit voor de naam van de methode het datatype twee keer is benoemd (`double?`, `double?`). Dit worden Tuples genoemd en deze maken het mogelijk om meer dan één waarde terug te geven. Aangezien je twee getallen terug wilt krijgen uit de methode maak je hier gebruik van.
5. Maak een try...catch blok aan in deze methode en declareer twee variabelen `getal1` en `getal2`.

```
static (double, double) FoutAfhandeling(string getal1String, string getal2String)
{
 try
 {
 double getal1 = double.Parse(getal1String);
 double getal2 = double.Parse(getal2String);
 }
}
```



- De code gaat dus proberen om de twee parameters van het type strings **getal1String** en **getal2String** om te zetten naar een double.
- De methode geeft twee waarden terug dus je dient hier een iets andere code voor te gebruiken. Neem onderstaande (geel gemarkeerde) code over.

```
static (double, double) FoutAfhandeling(string getal1String, string getal2String)
{
try
{
 double getal1 = double.Parse(getal1String);
 double getal2 = double.Parse(getal2String);
 return (getal1, getal2);
}
```

- Het Try blok is nu klaar en je kunt de code voor het Catch blok maken. Neem onderstaand code over zodat er een venster verschijnt. Let op dat er in de MessageBox nu een andere melding wordt getoond.

```
catch (Exception foutMelding)
{
 MessageBox.Show(foutMelding.Message + "Er wordt nu standaard 0 ingevuld voor beide getallen!!");
}
```

- Aangezien je een methode aanmaakt welke een waarde teruggeeft dient ook het Catch blok een waarde terug te geven anders gaat je applicatie niet werken. Je kunt dit zien door met je muis naar FoutAfhandeling te gaan. Hier staan ook rode kringeltjes onder.

FoutAfhandeling(string getal1String, string getal2String)

💡 ⓘ (double, double) frmHoofdscherm.FoutAfhandeling(string getal1String, string getal2String)

:tall1 = dou :tall2 = dou :getal1, get

CS0161: 'frmHoofdscherm.FoutAfhandeling(string, string)': not all code paths return a value

Show potential fixes (Alt+Enter or Ctrl+.)

- De melding geeft aan dat niet alle mogelijkheden een waarde teruggeven. Dit klopt want jouw Catch blok laat alleen een popup venster zien en geeft geen waarde terug.
- Dit is dan ook de reden waarom dit niet de juiste methode is om te werken met foutmeldingen. Voor nu gaan we met een workaround werken. Voeg daarom onderstaande code toe aan de Catch scope.

```
double getal1 = 0;
double getal2 = 0;
return (getal1, getal2);
```

- Je declareert opnieuw de twee variabelen en geeft de waarde 0 mee.
- Met **return** worden deze waarden teruggegeven zodat je straks als uitkomst van de berekening 0 krijgt.  
*Aangezien dit eigenlijk niet waar is, want er moet helemaal geen berekening worden uitgevoerd, maak je dit duidelijk in de MessageBox zodat de gebruiker weet wat er gebeurd.*

14. Je kunt nu de methode gebruiken in alle events van je code. Aangezien je nu de Try ...Catch in de methode **FoutAfhandeling()** hebt staan kun je de Try en Catch uit al je events verwijderen.
15. Bij het uitvoeren van de methode komen er twee waarden terug welke je op kunt slaan in een variabele. Maak een variabele aan met de naam **waarden** van het type var waarin de waarden komen van de methode.

```
var waarden = FoutAfhandeling(txbGetal1.Text, txbGetal2.Text);
```

16. De parameters welke je mee gaat geven naar de methode zijn van het type string, je kunt dus txbGetal1.Text en txbGetal2.Text zonder om te zetten meegeven als parameter bij het aanroepen van de methode.
17. De twee getallen welke in de variabele **waarden** zijn opgeslagen kun je ophalen met **waarden.Item1** (Dit is de waarde van getal1) en **waarden.Item2** (Dit is de waarde van getal2).
18. Pas nu de rest van je applicatie aan zodat de rekenmachine werkt en er een foutmelding komt op het moment dat er geen getallen worden ingevoerd.
19. Je kunt nu zien dat er veel minder code in je applicatie staat omdat je de try....catch in een methode hebt geplaatst waardoor je die niet in ieder event hoeft aan te maken.

*Zoals al eerder aangegeven is dit NIET de juiste manier om met foutAfhandeling om te gaan. De opdracht is alleen om aan te tonen wat het nut is van methoden ook als je minder complexe applicaties gebruikt. Later in deze module ga je leren hoe je hier wel mee om kunt gaan.*

#### Variabelen in een scope

Bij het aanmaken van de methode heb je gezien dat je tweemaal de variabele getal1 en getal2 declareert. Één keer in de scope van **Try** en één keer in de scope van **Catch**. Dit is nodig omdat je de variabelen aanmaakt binnen een scope en alleen binnen dezelfde scope kun je deze gedeclareerde variabelen gebruiken. Je zult in een nieuwe scope dus altijd een nieuwe variabele moeten maken en deze kan dezelfde naam krijgen als de variabele in een andere scope. Later leer je ook om variabelen aan te maken welke binnen meerdere scopes te gebruiken zijn.

## 13.7 Eindopdracht Methoden

Een lokale makelaar is geïnteresseerd in een applicatie waarmee hij eenvoudig een schatting kan maken van de inhoud van het huis om deze waarden op Funda te zetten bij de verkoop van een huis. Dit hoeft allemaal niet op de centimeter nauwkeurig dus hij is van plan om alleen de benedenverdieping op te meten en op basis daarvan een berekening uit te laten voeren.



1. Maak een nieuwe Windows Forms applicatie aan met als naam **HuisBerekenen**.

The screenshot shows a Windows application window titled "Berekenen Huis". Inside, there are five text input fields for dimensions and one for the number of floors. To the right of each input field is a corresponding output text field showing "00000". At the bottom is a "Bereken" button.

Afbeelding 41 - Huis Berekenen

2. Maak bovenstaande applicatie na en gebruik de juiste naamgevingsconventies.
3. Zorg ervoor dat zodra er op Bereken wordt geklikt de ingevulde waarden worden gebruikt en de uitkomsten rechts op het formulier komen te staan waar nu de 00000 staan.
4. Maak methoden voor de berekeningen en probeer zo efficiënt mogelijk te werken (zo min mogelijk dubbele code).
5. Je zorgt voor fout afhandeling maar je zet dit NIET in een methode want er mogen natuurlijk geen 'valse' waarden worden gebruikt.
6. Maak een screenshot van jouw code en plaats deze op It's Learning.

## 13.8 Wat heb je geleerd.

- Je weet wat een methode is.
- Je kent de verschillen tussen de twee methodes.
- Je kunt een methode aanmaken en aanroepen in je code.
- Je weet wat het nut is van het gebruik van methoden.

# 14 Date Time Variabele

Een handige variabele welke je in je projecten kunt gebruiken is de DateTime variabele en deze wordt vaak gebruikt in applicaties. De DateTime variabele heeft een groot bereik van het jaar 0001 om 00:00:00 tot het jaar 9999 11:59:59. De tijd wordt gemeten in zogenaamde ‘ticks’ en deze zijn 100 nanoseconden. Zodra je deze variabele gebruikt gaat de computer rekenen met deze ticks vanaf 0001 00:00:00 tot het moment waarop je de gegevens ophaalt. Deze variabele heeft verschillende methoden en eigenschappen waar je mee kunt werken. Zo kun je bijvoorbeeld de dag van de week bepalen of kijken of het dit jaar een schrikkeljaar is maar er zijn nog veel meer mogelijkheden.

## 14.1 Opdracht - DateTime variabele

Voor deze opdracht ga je applicatie maken waarmee je kunt ‘rekenen’ met tijden en leeftijden. Gebruik je kennis uit voorgaande oefeningen.

1. Maak een nieuw Windows Forms project aan en geef dit de naam TimeCalculator.
2. Plaats onderstaande controls op het formulier. Je zult wellicht controls tegenkomen welke je nog niet kent maar je ziet vanzelf wat ze doen.

| Control        | Actie                                        | Naam             |
|----------------|----------------------------------------------|------------------|
| Buttons        | Ophalen jaartal                              | btnJaar          |
|                | Uitrekenen leeftijd                          | btnLeeftijd      |
|                | Controleren of het een schrikkeljaar is.     | btnSchrikkelJaar |
|                | Ophalen Dag van de week                      | btnWeek          |
|                | Ophalen weeknummer                           | bntWeekNummer    |
|                | Uitreken hoe lang nog tot een bepaalde datum | btnDagenTot      |
| DateTimePicker | Datum selecteren                             | dtpDatum         |
| Label          | Weergeven resultaat                          | lblResultaat     |

3. Pas de eigenschap van het formulier en de controls aan zodat de applicatie er netjes uitziet en verander de eigenschap van de buttons zodat je weet wat er gebeurd zodra je erop klikt.
4. De eerste stap is het ophalen van het jaartal.
5. Dubbelklik op de button om naar het code venster te gaan.
6. Maak twee variabelen met onderstaande namen. Tussen de haakjes staat aangegeven wat voor variabele het moet worden.  
**jaar** (int)  
**vandaag** (DateTime)
7. De waarde van de variabele **vandaag** wordt gevuld met de datum en tijd van het moment waarop er op de knop wordt gedrukt. Dit realiseer je door de eigenschap **DateTime.Now** uit te lezen.
8. De code wordt als volgt.

```
DateTime vandaag = DateTime.Now;
```

- Je hebt nu een variabele **vandaag** van het type DateTime met de bijbehorende eigenschappen en methoden. Zo kun je eenvoudig gegevens afleiden uit de datum. Zo kun je bijvoorbeeld het jaartal eenvoudig ophalen door naar de eigenschap **vandaag.Year** in de variabele **ditJaar** te stoppen.

- De code wordt als volgt.

```
DateTime vandaag = DateTime.Now;
int ditJaar = vandaag.Year;
```

- De volgende stap is het weergeven van het resultaat in het label **lblResultaat**. Dit heb je al vaker gedaan.

```
lblResultaat.Text = Convert.ToString();
```

- In bovenstaande code wordt de waarde van variabele Jaar geplaatst in de Text eigenschap van het **object lblResultaat**. De waarde wordt uiteraard eerst omgezet naar een string.

- De code voor het converteren naar de string ziet er iets anders uit dan je gewend bent. Er zijn verschillende mogelijkheden en de bovenstaande manier is de kortste en meest leesbare. Onderstaande code welke je eerder hebt gezien kunt je ook gebruiken.

```
lblResultaat.Text = Convert.ToString(Jaar);
```

- Als je alles goed hebt overgenomen heb je geen meldingen in je error list en kun je het programma starten.

- Test of het jaartal correct wordt opgehaald.

In de volgende stappen ga je de code schrijven waarmee je de leeftijd kunt uitrekenen zodra je op de knop **btnLeeftijd** drukt. Hiervoor ga je een nieuw type variabele gebruiken genaamd **TimeSpan** wat simpel gezegd het verschil is tussen twee datums.

- Ga naar de code welke wordt uitgevoerd zodra je op de **btnLeeftijd** knop klikt.

- Maak een variabele aan met als naam **geboorteDag** waarin de geboorteDag van de gebruiker wordt opgeslagen.

- De waarde welke in deze variabele wordt opgeslagen is de datum welke wordt ingevoerd door de gebruiker met behulp van de DateTimePicker op je formulier.

- Om de ingevoerde waarde uit te lezen gebruik je de eigenschap **dtpDatum.Value**. De code ziet er als volgt uit.

```
DateTime geboorteDag = dtpDatum.Value;
```

- Nu je de datum hebt opgehaald kun je het verschil uitrekenen met de datum van vandaag en deze waarde opslaan in een variabele. Het is niet noodzakelijk om de datum van vandaag in een losse variabele op te slaan aangezien we geen specifieke eigenschap op gaan vragen.

- De code wordt in dit geval als volgt.

```
DateTime geboorteDag = dtpDatum.Value;
TimeSpan leeftijd = DateTime.Now - GeboorteDag;
```

22. In de variabele **leeftijd** is nu het verschil tussen de twee datums opgeslagen.
23. De waarde moet worden getoond in **lblResultaat**. De uitkomst wordt echter geen mooi rond getal. Om hiervoor te zorgen kun je een opmaak meegeven op het moment dat je de waarde converteert naar een string. Dit gaat op de volgende manier.

```
lblResultaat.Text = (leeftijd.TotalDays).ToString("111");
```

24. Door tussen de haakjes "N0" van de methode **.ToString()** te plaatsen geef je aan dat in de string geen getallen achter de komma komen te staan.

**Let op!** Er wordt in dit geval geen rekening gehouden met afronding. In bovenstaande code worden de eerste drie cijfers omgezet naar een string en alles wat erachter staat wordt 'weggegooid'. Je kunt deze methodiek dus niet gebruiken op het moment dat afronding noodzakelijk is. In dat geval kun je bijvoorbeeld eerst de waarde converteren naar een Integer (*int*) aangezien dat hele getallen zijn. En daarna de waarde converteren naar een string.

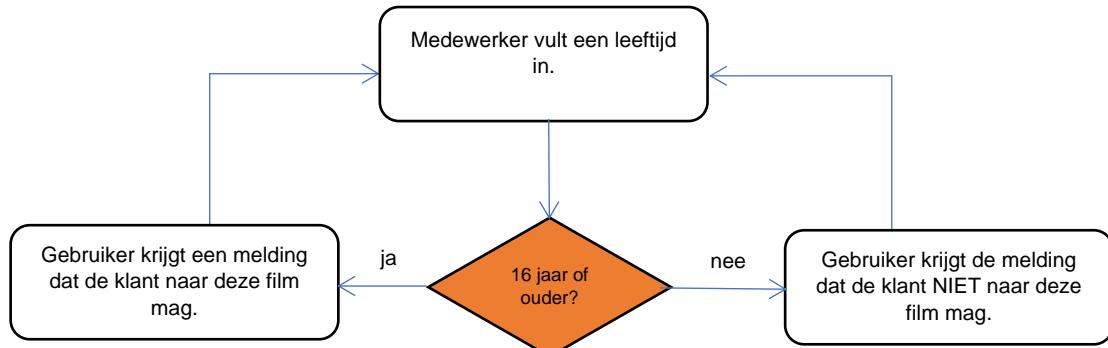
25. Test je programma en kijk of alles werkt.
26. Wil je de leeftijd in jaren weergeven dan dien je uiteraard het aantal dagen te delen door het aantal dagen in een jaar. Om dit nauwkeurig uit te rekenen gebruik je 365,25 dagen.
27. Pas je programma aan zodat de leeftijd in jaren wordt weergegeven.
28. Test het programma.
29. Maak nu je programma af en gebruik de kennis die je hebt opgedaan in voorgaande opdrachten om de code te schrijven voor de overige knoppen welke nog niet behandeld zijn.  
Je kunt op internet zoeken naar eventuele methodes welke je nodig hebt maar IntelliSense zal je al een eind op weg helpen.
30. Test je programma en kijk of alles werkt.
31. Maak een screenshot van jouw code en plaats deze op It's Learning.

## 14.2 Wat heb je geleerd.

- Je weet wat een Date Time Variabele is.
- Je kunt een Date Time Variabele toepassen in je applicatie.
- Je kunt rekenen met datums in je applicatie.

## 15 Selecties - If..Else

Selecties worden veel toegepast bij het programmeren en worden gebruikt om het programma keuzes te laten maken. Met een selectie kun je dus aangeven dat een bepaalde code wordt uitgevoerd als een bepaald getal wordt ingevuld. In onderstaande schema staat dit met een voorbeeld aangegeven voor een café waar een personen alcohol besteld



In het ‘wybertje’ wordt een vergelijking gemaakt en deze uitkomst kan waar (ja) of niet waar (nee) zijn. In dit voorbeeld is de vergelijking “is de ingevulde waarde hoger of gelijk aan 16”.

Selecties ga je in C# maken met zogenaamde IF statements. In de volgende paragrafen ga je hier mee aan de slag.

### 15.1 If....Else

In het voorbeeld van de vorige paragraaf is schematisch te zien hoe een selectie werkt in C#. Vertaal je dit naar de code die je moet gebruiken, dan ziet dat er als volgt uit.

In C# ziet zo'n selectie er als volgt uit.

```
if (true)
{
 //Code welke wordt uitgevoerd als de vergelijking WAAR is
}
else
{
 //Code welke wordt uitgevoerd als de vergelijking NIET WAAR is
}
```

Voor het voorbeeld van de bioscoop kan dit de code zijn:

```
if (leeftijd >= 16)

{
 MessageBox.Show("De klant mag naar deze film.");
}
else
{
 MessageBox.Show("De klant mag NIET naar deze film!");
}
```

- Het programma kijkt of de vergelijking tussen de haakjes bij IF **waar** is (TRUE).
- Indien dit het geval is zal de code tussen de accolades worden uitgevoerd.

- Indien dit niet het geval is zal het programma de code uitvoeren bij het ELSE statement.

Zodra de selectie is uitgevoerd gaat het programma verder met de code. Indien er geen code onder het IF...ELSE statement staat dan zal het programma de code van voor af aan doorlopen. Het programma wacht dan op een nieuwe input van de gebruiker. Zodra er wel code onder het statement staat zal deze eerst worden doorlopen. Het programma zal deze code direct na het uitvoeren van de selectie uitvoeren dus houdt er rekening mee dat het programma niet wacht tot de gebruiker iets doet.

In de selectie kunnen uiteraard ook andere vergelijkingen worden gebruikt zoals bijvoorbeeld “is het selectievakje in het formulier aangevinkt?”

Dit wordt duidelijk aan de hand van de volgende opdracht waarin je op basis van een leeftijd het programma laat bepalen of en persoon alcohol mag drinken of niet.

1. Maak een nieuw Windows Form Programma aan en noem het LeeftijdControle.
2. Plaats op het formulier een button en een checkbox en geef ze de namen chbKeuze en btnCheck.
3. Verander de tekst van de checkbox en knop zodat duidelijk is wat ze doen.

4. Maak een event aan door dubbel te klikken op de btnCheck en voeg onderstaande code in.

```
if (chbKeuze.Checked)
{
}
else
{
}
```

5. Zet tussen de accolades bij het IF deel de code welke een pop-up-venster laat zien waarin staat “Ja je mag alcohol drinken!”
6. Zet tussen de accolades bij het ELSE deel de code welke een pop-up-venster laat zien waarin staat “Nee je mag geen alcohol drinken!”
7. Controleer of je applicatie werkt.
8. De controle is natuurlijk ook uit te voeren door de GeboorteDag in te geven door gebruik te maken van een **DateTimePicker**.
9. Plaats een **DateTimePicker** met als naam **dtpGeboorteDag** op het formulier en een **label** met als naam **lblGeboorteDag**.
10. Zorg dat in de tekst komt te staan “Wat is uw GeboorteDag?“.
11. Nu kun je de code aanpassen.
12. Om de juiste leeftijd uit te kunnen rekenen kun je gebruik maken van het aantal dagen dat iemand oud is. Om dit uit te kunnen rekenen heb je een aantal gegevens nodig.
  - a. De datum van vandaag.
  - b. De GeboorteDag.
  - c. Het verschil tussen de GeboorteDag en de datum van vandaag.

13. Je hebt in het vorige hoofdstuk geleerd dat je de datum van vandaag eenvoudig op kunt halen. Zorg ervoor dat deze waarde wordt opgeslagen in de variabele:

```
DateTime vandaag =
```

14. De GeboorteDag wordt door de gebruiker ingevuld in de **DateTimePicker**. Zorg ervoor dat deze waarde wordt opgeslagen in de variabele:

```
DateTime geboorteDag =
```

15. Om het aantal dagen uit te rekenen kun je de twee waarden van bovenstaande variabelen van elkaar aftrekken. Deze uitkomst komt in de volgende variabele te staan.

```
TimeSpan leeftijdInDagen =
```

16. Je kunt nu de leeftijd in jaren uitrekenen door de waarde van **leeftijdInDagen** te delen door 365.25. Maak de volgende variabele aan waarin deze uitkomst wordt opgeslagen. (*Let op je maakt gebruik van een Double omdat je deelt door een kommagetal!*)

```
double leeftijd =
```

17. In je If statement kun je nu controleren of de GeboorteDag groter of gelijk is dan ( $\geq$ ) de waarde van de variabele **leeftijd**.

18. Deze vergelijking neem je op in je if statement dus pas deze aan zoals hieronder aangegeven.

```
if (leeftijd >= 18)
```

19. Test je programma en kies een datum in het jaar 2000 om te kijken of de juiste messagebox verschijnt. Test het nogmaals en vul een datum van vorig jaar in.

Je hebt nu een applicatie welke gebruik maakt van een selectie of vergelijking om te bepalen welke code verder wordt uitgevoerd. Je hebt opgegeven in het ELSE deel wat er moet gebeuren als er niet aan de voorwaarde is voldaan. Het is ook mogelijk om meerdere voorwaarden te scheppen door meerdere IF statements te gebruiken.

20. Je code ziet er nu als het goed is als volgt uit.

```
private void btnCheck_Click(object sender, EventArgs e)
{
 DateTime vandaag = DateTime.Today;
 DateTime geboorteDag = dtpBirthdate.Value;
 TimeSpan leeftijdInDagen = vandaag - geboorteDag;
 double leeftijd = (leeftijdInDagen.Days) / 365.25;

 if (leeftijd >= 18)
 {
 MessageBox.Show("De klant mag alcohol drinken");
 }
 else
 {
 MessageBox.Show("De klant mag GEEN alcohol drinken!");
 }
}
```

21. Je gaat nu een extra optie toevoegen waarmee wordt aangegeven over hoeveel dagen een persoon mag drinken. Dit gebeurt alleen als deze persoon 17 jaar of ouder is.
22. Voeg de gele gemarkeerde code toe. Je ziet hieronder waar dit moet komen te staan.

```
if (leeftijd >= 18)
{
 MessageBox.Show("U mag drinken!");
}
else if (leeftijd >= 17)
{

 MessageBox.Show("U mag over " + dagenResterend + " dagen drinken!");
}
else
```

23. Je ziet in bovenstaande code dat er een nieuwe variabele wordt gebruikt met als naam **dagenResterend**. In deze variabele moet het aantal dagen komen te staan vanaf Vandaag tot de 18<sup>e</sup> verjaardag van de betreffende persoon. *Je gaat deze zo meteen aanmaken*
24. Voordat je dit kunt uitrekenen heb je eerst een aantal gegevens nodig.
- Hoeveel dagen moet iemand oud zijn om te drinken.
  - Hoeveel dagen oud is de betreffende persoon.
25. Het eerste gegeven is eenvoudig uit te rekenen door het aantal jaren (18) te vermenigvuldigen met het aantal dagen in een jaar (365,25).
26. Maak een variabele aan van het type **double** en geeft het de naam **dagenDrinken**.

```
double dagenDrinken = 18 * 365.25;
```

27. Het tweede gegeven (hoeveel dagen oud de betreffende persoon is) heb je al eens eerder opgehaald uit de variabele **leeftijdInDagen**.

```
leeftijdInDagen.Days
```

28. Je hebt nu voldoende gegevens om het aantal dagen uit te kunnen rekenen.
29. Maak de variabele **dagenResterend** aan en zorg dat deze van het type **Double** is.
30. De berekening wordt nu <het aantal dagen dat iemand oud moet zijn> - < het aantal dagen dat iemand oud is)
31. De code wordt dus als volgt:

```
double dagenResterend = dagenDrinken - leeftijdInDagen.Days;
```

Met de methode AddYears() kun je de datum laten zien waarop ze mogen drinken. We gaan ervan uit dat ze dat zelf ook wel weten dus die gebruiken we hier niet.

Zou je dit wel willen gebruiken dan kun je onderstaande code gebruiken om de datum waarop de persoon 18 wordt weergeven.

```
MessageBox.Show("U mag drinken vanaf " + geboorteDag.AddYears(18).ToShortDateString());
```

32. Je bent nu klaar om je programma te testen. Test je programma door driemaal een geschikte datum in te vullen waarbij je één van de drie meldingen krijgt.

33. Zet je project op GitHub

## 15.2 If...Else - Doorlopen van de code

In voorgaande oefening heb je gezien dat je één of meerdere voorwaarden kunt toevoegen aan je applicatie waarna op basis van deze voorwaarde een bepaalde code wordt uitgevoerd. De eerste voorwaarde komt in het IF statement en eventueel aanvullende voorwaarden kun je toevoegen met een ELSE IF statement. Met het ELSE statement kun je aangeven wat er moet gebeuren als aan geen enkele voorwaarde wordt voldaan.

Je programma doorloopt de code altijd van boven naar beneden zet zoals je een boek leest. Je hebt echter gezien dat sommige delen van de code worden overgeslagen op het moment dat je werkt met IF en ELSE. In onderstaande afbeelding en bijbehorende uitleg kun je zien en lezen hoe de code doorlopen wordt.



Afbeelding 42 - Doorlopen van de code

1. Er wordt op de **btnCheck** geklikt waardoor de code van dit **Event** wordt gestart.

```
private void btnCheck_Click(object sender, EventArgs e)
```

2. De variabelen worden aangemaakt en gevuld zoals dit in de code staat.

```
DateTime vandaag = DateTime.Today;
DateTime geboorteDag = dtpBirthdate.Value;
TimeSpan leeftijdDagen = vandaag - geboorteDag;
double leeftijd = (leeftijdDagen.Days) / 365.25;
double dagenDrinken = 18 * 365.25;
double dagenResterend = dagenDrinken - leeftijdDagen.Days;
```

3. Je programma komt aan in het '**IF....ELSE blok**'.

4. De eerste voorwaarde wordt gecontroleerd.

```
if (leeftijd >= 18)
```

5. Wordt er aan de voorwaarde voldaan (true) dan wordt de code bij dit IF statement uitgevoerd. (**groene pijl**)

```
 MessageBox.Show("U mag drinken!");
```

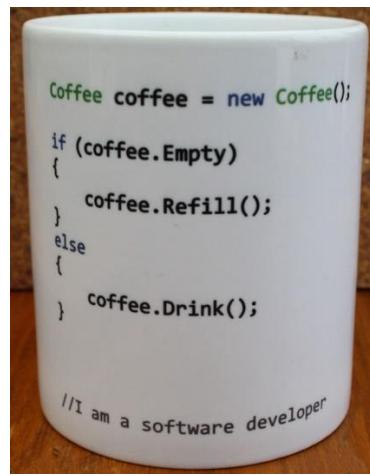
6. Na het uitvoeren van bovenstaande code 'springt' het programma uit het '**IF.....ELSE blok**' en gaat verder met de code onder het blok indien dit aanwezig is. (**lange groene pijl aan de rechter kant**)
7. Wordt er NIET aan de voorwaarde voldaan dan wordt er naar de volgende voorwaarde (else if) gekeken. (**rode pijl**)

```
else if (leeftijd >= 17)
```

8. Wordt er aan deze voorwaarde WEL voldaan (true) dan wordt de code in dit **ELSE IF** statement uitgevoerd.

```
MessageBox.Show("U mag over " + DagenResterend + " dagen drinken!");
```

9. Na het uitvoeren van bovenstaande code springt het programma uit het '**IF ELSE blok**' en gaat verder met de code onder het blok indien dit aanwezig is. (**lange groene pijl aan de rechter kant**)
10. In dit geval zijn er geen andere voorwaarden aanwezig dus gaat het programma naar het **ELSE** statement (**rode pijl**) en wordt de code uitgevoerd.
11. Na het uitvoeren van de code springt het programma uit het '**IF ELSE blok**' en gaat verder met de code onder het blok indien dit aanwezig is. (**lange groene pijl aan de rechter kant**)



Afbeelding 43 - If...Else

## 15.3 Opdracht – Aanpassen Applicatie LeeftijdCheck

Kijk je naar de applicatie dan zie je daar een aantal zaken in staan die niet echt duidelijk zijn. Waarom staat er bijvoorbeeld geboorteDag in plaats van geboorteDatum wat een stuk duidelijker is. Ook staat de variabele leeftijd niet correct geschreven want het begint niet met een hoofdletter.

Naast deze fouten is er ook een handigere manier om de dag te bepalen waarop iemand 18 jaar wordt namelijk met de volgende code.

```
geboorteDatum.AddYears(18).ToString()
```

**AddYears(18)** Telt 18 jaar op bij de ingevoerde datum.

**ToShortDateString()** Zorgt ervoor dat de datum wordt omgezet naar een string in het formaat DD-MM-YYYY. Je kunt ook de ToString() methode gebruiken maar dan krijg je het formaat DD-MM-YYYY HH-MM-SS wat niet altijd wenselijk is.

*(Je gebruikt de methode **ToShortDateString()** uiteraard alleen als je er een string van moet maken voor het bijvoorbeeld tonen in een Label of MessageBox. Je kunt de AddYears() methode uiteraard ook gebruiken zonder het om te zetten naar een string.)*

### Opdracht - wijzig onderstaande items in je applicatie

- Variabelen moeten beginnen met een kleine letter volgens de naamgevingsconventie. Indien de variabele bestaat uit twee woorden aan elkaar dan wordt camelCase gebruikt dus de eerste letter van het eerste woord is een kleine letter en de letter van het tweede woord en opeenvolgende woorden beginnen met een hoofdletter.
- Pas de variabelen aan zodat ze voldoen aan de naamgevingsconventie.
- Variabele leeftijdDagen is niet duidelijk genoeg, maak hier **leeftijdInDagen** van.
- De naam geboorteDag gebruiken we ook niet dus maak hier **geboorteDatum** van.
- De naam dagenDrinken klinkt ook niet lekker dus maak hiervan **drinkLeeftijdInDagen**.
- In de applicatie wordt niet consistent omgegaan met Nederlandse en Engelse benaming. Pas de namen van de items op je formulier en in je code aan zodat ze allemaal in het Nederlands staan.
- Bij het uitrekenen van het aantal dagen dat iemand nog moet wachten voordat hij/zij mag drinken reken je nu eerst alles om naar dagen. Gebruik de AddYears() methode in je applicatie zodat dit niet meer hoeft. (Het resultaat moet uiteraard wel in dagen worden weergegeven)
- Gebruik deze methode ook om de datum weer te geven waarop de persoon mag drinken indien hij/zij jonger is dan 17 jaar.
- Test je applicatie.
- Maak een screenshot van jouw code en plaats deze op It's Learning.

## 15.4 Opdracht – Actiedagen bij Kubus.Com

De winkelketen **Kubus.Com** gaat strooien met kortingen in de komende vier dagen. De CEO van het bedrijf mevrouw Gill Bates kwam op het idee om de klanten korting te geven op basis van de hoogte van de aankoop. Ze hoopt hiermee nog meer naamsbekendheid op te doen waardoor er in de toekomst meer klanten naar haar komen in plaats van de winkel van haar broer. Hoe duurder het product hoe hoger het kortingspercentage. Maak een applicatie genaamd KortingsBerekening welke laat zien wat een product kost op basis van onderstaande regels.

- De applicatie bevat een invoerveld waarin het bedrag getypt kan worden.
- De applicatie bevat een knop waarmee het bedrag wordt uitgerekend.
- In de applicatie worden de volgende items getoond nadat op de knop is geklikt:
  - Het percentage korting dat de klant krijgt op het product.
  - De korting in euro's die de klant krijgt.
  - Het te betalen bedrag.
- De volgende percentages kunnen worden aangehouden.
  - Producten tot 10 euro krijgen 5% korting.
  - Producten tot 40 euro krijgen 12,50% korting.
  - Producten tot 100 euro krijgen 15% korting.
  - Producten vanaf 200 euro krijgen 21,5% procent korting.
- Er staat uitleg bij bovenstaande items wat ze inhouden.
- Het formulier heeft een correcte naam.
- De naamgevingsconventies worden aangehouden.
- Er wordt foutafhandeling toegepast waarbij je gebruik maakt van een methode.

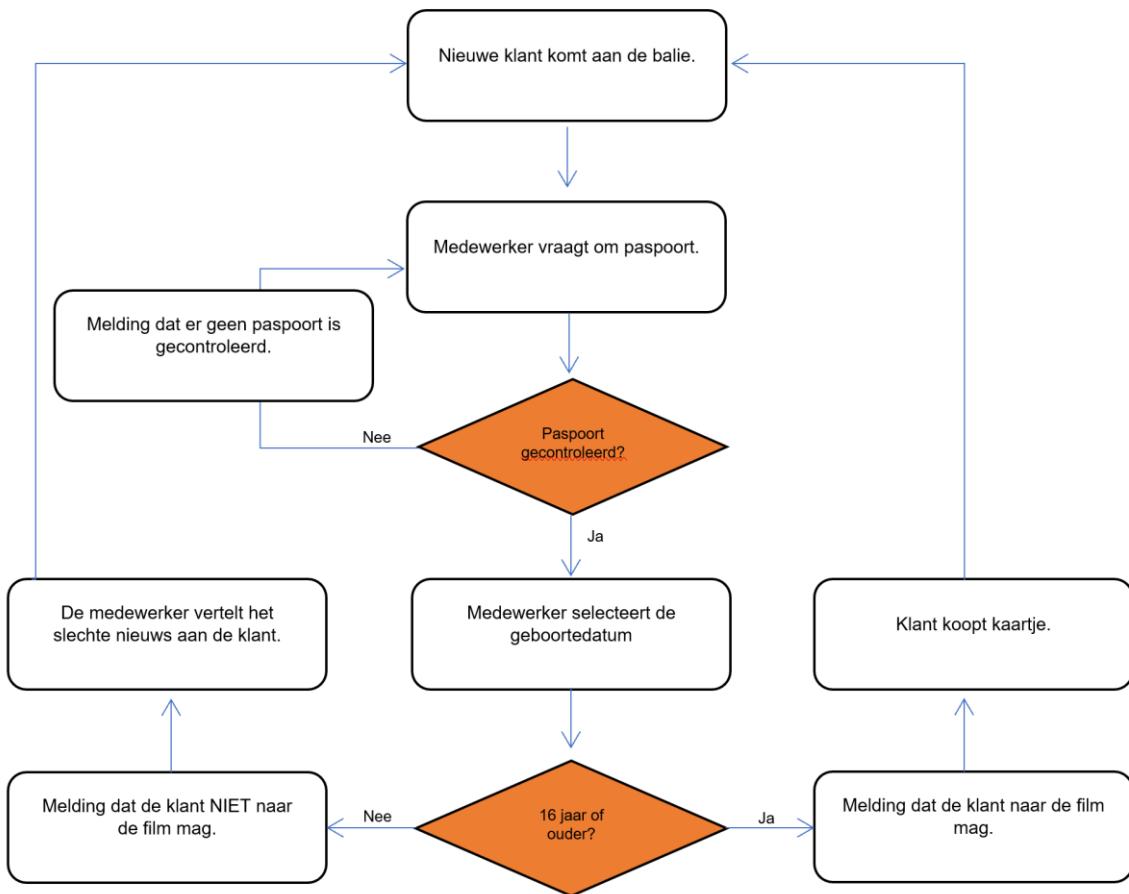
## 15.5 Wat heb je geleerd?

- Je kunt werken met If Else in je code om keuzes te maken.
- Je weet wat een vergelijking is.
- Je weet hoe het If Else statement wordt doorlopen.

## 16 Geneste If Statements

Indien je meerdere voorwaarden wilt opnemen zoals bijvoorbeeld twee controles dan kun je gebruik maken van 'een *IF statement* in een *IF statement*'. Dit wordt een geneste if statement genoemd. Zodra je hier gebruik van gaat maken wordt je applicatie al snel een stuk complexer van structuur en is het raadzaam om van tevoren eerst de 'flow' van je programma op papier te zetten om te voorkomen dat je fouten gaat maken.

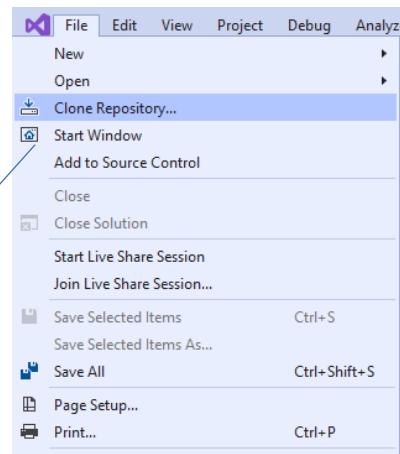
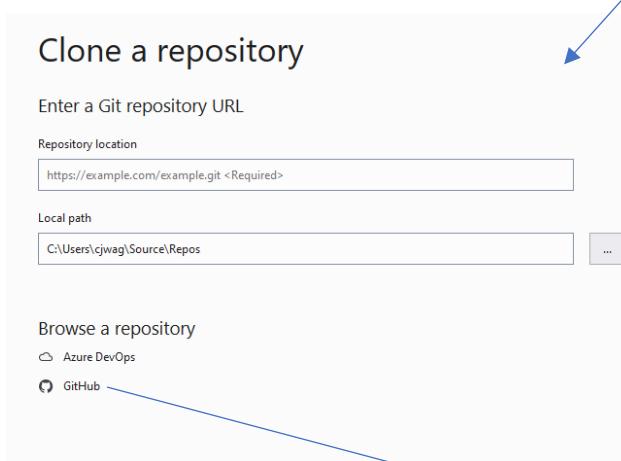
Je gaat je applicatie zo meteen uitbreiden zodat de medewerker van de bioscoop moet aanvinken dat de leeftijd is gecontroleerd op een paspoort. Indien er geen paspoort is gecontroleerd krijgt de medewerker een melding dat dit niet is uitgevoerd. De 'flow' van je programma gaat dan als volgt:



Door middel van bovenstaande flow ga je een nieuwe LeeftijdControle applicatie bouwen welke gebruikt kan worden in een bioscoop.

## 16.1 Opdracht – Geneste IF Statements

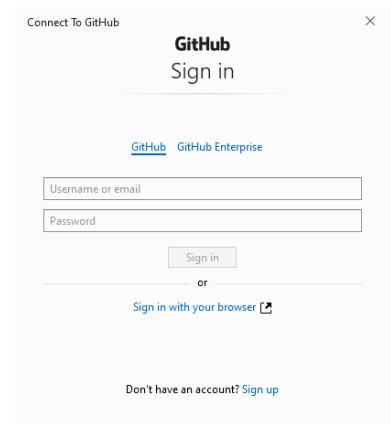
1. Clone de LeeftijdControle applicatie welke je eerder op Github hebt gezet door naar de optie File → Clone Repository te gaan.
2. Je kunt ook kiezen om een new project aan te maken. Je krijgt dan ook de optie voor het aanmaken van een clone.
3. Het scherm van Afbeelding 45 zal verschijnen.



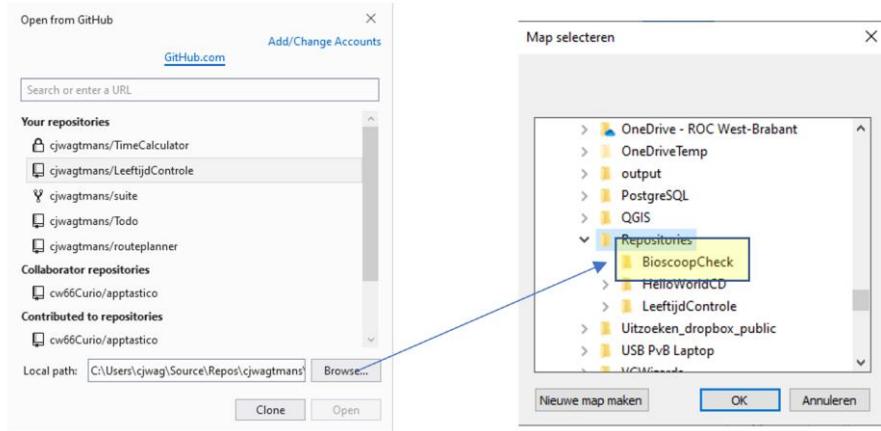
Afbeelding 44 - Clone Repository

Afbeelding 45 - Repository gegevens invullen

4. Klik onderin bij **Browse a Repository** op **GitHub** en de GitHub Sign in zal verschijnen.
5. Voer je gegevens en log in.
6. Je krijgt een lijst te zien met repositories van je GitHub account.
7. Klik op **Browse** en maak een nieuwe map aan in je lokale Repository map met de naam **BioscoopCheck**. Maak een nieuwe map aan waarin je de **clone** wilt plaatsen.

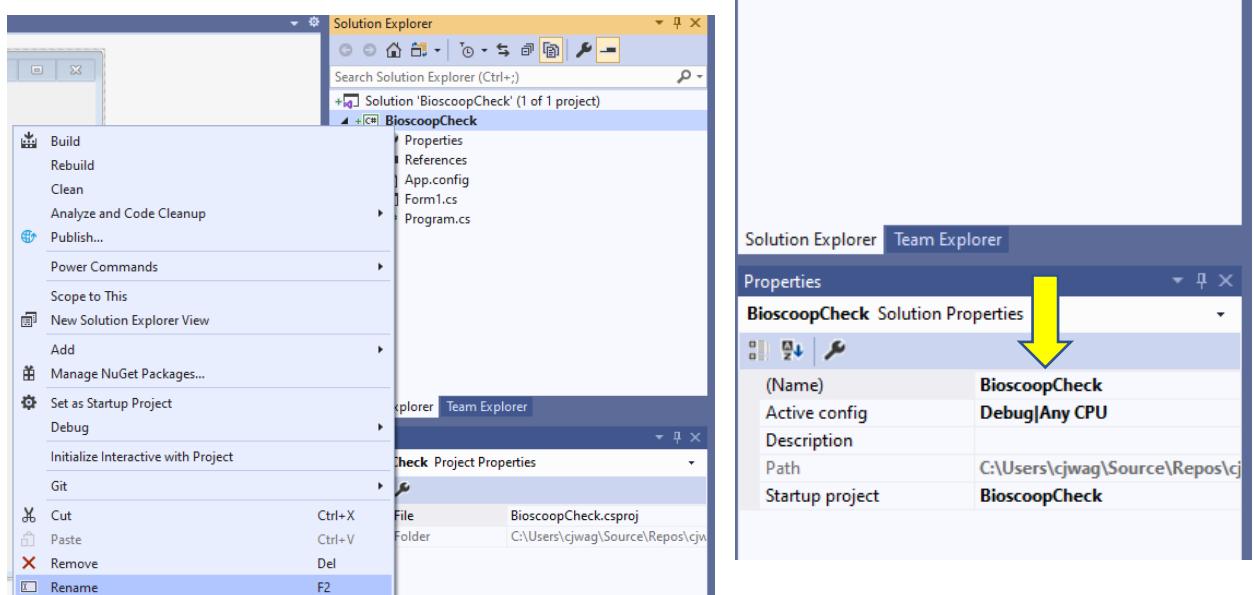


Afbeelding 46 - GitHub Sign in



Afbeelding 47 - Repository nieuwe map

8. Klik rechts op het tabje Solution Explorer en verander de naam van de Solution naar BioscoopCheck. Deze naam kun je veranderen door in het Solution venster te gaan staan op de Solution en bij de properties de naam te wijzigen.
9. Verander ook de naam van de applicatie naar **BioscoopCheck**.



Afbeelding 48 - Naam van Solution en applicatie aanpassen

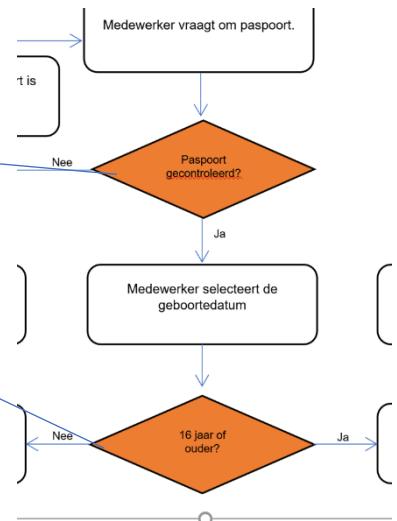
10. Je gaat nu dus gebruik maken van een geneste IF statement.

11. Voeg in de code die je nu hebt een IF statement toe op geel gemarkeerde plek. (*Let op dat in onderstaande code jouw aanpassingen uiteraard niet staan. Onderstaande code is alleen om duidelijk te maken waar het moet komen te staan.*)

```

if (true)
{
 if (true)
 {
 }
 else
 {
 Popup - Geen paspoortcontrole
 }
}

```



12. In de eerste `if (true)` moet de applicatie controleren of er een vinkje staat bij de paspoortcontrole. (Zoals aangegeven met de pijlen.)
13. Pas de `if (true)` aan zodat de voorwaarde wordt dat het vinkje aan staat.. (*Dit heb je al eens eerder gedaan!*)
14. In je programma kunnen zich nu twee situaties voordoen op het moment dat er op de knop wordt geklikt.
- Er staat een vinkje → de leeftijd kan gecontroleerd gaan worden.
  - Er staat GEEN vinkje → de medewerker krijgt een melding dat het paspoort niet is gecontroleerd.
15. Voor beide opties ga je nu de code aanpassen

#### A. Er staat een vinkje

- Indien het vinkje aan staat kan de leeftijd gecontroleerd worden.
  - Je applicatie gaat de code in het eerste IF statement uitvoeren.
  - Voor het controleren van de leeftijd staat de code al in de applicatie. Je hebt namelijk een kloon van de LeeftijdControle applicatie gemaakt. Dat is handig want dan hoeft je niet alles opnieuw te typen. Knip de code voor het controleren van de leeftijd welke je al eerder hebt gemaakt en plak deze in de scope van het eerste IF statement zoals hieronder is gemarkeerd.
- Let op dat onderstaande alleen dient om aan te geven waar jouw code geplakt moet worden. De code welke niet geel is gemarkeerd zal bij jou als het goed is anders zijn want deze heb je al aangepast.*

```

if (true)
{
 if (leeftijd >= 18)
 {
 MessageBox.Show("De klant mag alcohol drinken");
 }
 else if (leeftijd >= 17)
 {
}

```

```

 MessageBox.Show("U mag over " + dagenResterend + " dagen drinken!");
 }

 else
 {
 MessageBox.Show("De klant mag GEEN alcohol drinken!");
 }
}
else
{
 //melding dat er geen paspoortcontrole is uitgevoerd
}

```

4. De door jou geplakte code wordt dus uitgevoerd op het moment dat de voorwaarde het eerste IF statement (controle of de CheckBox is aangevinkt) WAAR is.
5. Je hebt nu binnen dit IF statement een nieuw IF statement geplaatst dus de applicatie gaat naar deze voorwaarden kijken en dat zijn de voorwaarden waarbij wordt gekeken naar de leeftijd en de bijbehorende pop-up vensters
6. Nadat deze code is doorlopen wacht het programma tot er weer een keer op de knop wordt geklikt want je hebt geen code meer onder het IF statement staan. De onderste ELSE wordt uiteraard overgeslagen want je eerste IF statement was waar en de LeeftijdControle is uitgevoerd.

## B. Er staat GEEN vinkje

1. Indien er geen vinkje staat wordt er niet voldaan aan het eerste IF statement. Dit betekent dat het paspoort niet is gecontroleerd.
2. De code in het eerste IF statement wordt dus overgeslagen en je applicatie gaat naar het ELSE statement en voert de code uit welke tussen de accolades staat welke bij dit ELSE statement horen.
3. Je hebt daar op dit moment nog geen code staan dus je applicatie zal weinig doen.
4. Schrijf de code welke ervoor zorgt dat er een pop-up venster verschijnt waarin de tekst "**Geen alcohol schenken als het paspoort niet is gecontroleerd!**" staat.
5. Als je alles goed hebt gedaan verschijnt er nu een pop-up venster. Na het sluiten van dit venster zal je applicatie wachten tot er weer op de knop wordt geklikt.
6. Test je applicatie en kijk of je de juiste meldingen krijgt in de volgende scenario's;
  - a. Vinkje bij paspoortcontrole staat uit.
  - b. Vinkje bij paspoortcontrole staat aan - De persoon is 15 jaar.
  - c. Vinkje bij paspoortcontrole staat aan - De persoon is 17 jaar.
  - d. Vinkje bij paspoortcontrole staat aan - De persoon is 19 jaar.

## 16.2 Eindopdracht – Foutafhandeling RekenMachine

In de opdracht van paragraaf 13.6 heb je een methode aangemaakt welke ervoor zorgt dat een gebruiker een melding krijgt als er geen getallen worden gebruikt in de applicatie. Deze fout afhandeling was zoals aangegeven eigenlijk niet de juiste manier om met foutafhandeling om te gaan. De hoofdreden was dat er toch een waarde wordt teruggegeven wat eigenlijk niet correct is want er moet helemaal geen waarde teruggegeven worden. Daarnaast wil je eigenlijk dat er ook een duidelijke melding komt waar de fout zich voordoet.

Je hebt in dit hoofdstuk kennis gemaakt met het if...else statement en weet nu ook hoe je geneste if statements kunt gebruiken. Je hebt nu meerdere ‘tools’ tot je beschikking en bent in staat om zelf een nette methode te schrijven voor de foutafhandeling.

- Je gaat gebruik maken van **IF...ELSE** om duidelijk aan te kunnen geven welke fout er door de gebruiker wordt gemaakt.
- Je gaat gebruik maken van **null** waarden zodat er bij een fout geen waarde 0 hoeft te worden teruggegeven en er dus ook geen resultaat verschijnt in de rekenmachine.
- Je gaat gebruik maken van de **TryParse()** methode. Deze methode kan:
  - Aangeven of een waarde omgezet kan worden in een getal met een **TRUE** of **FALSE**.
  - Als de waarde **TRUE** is wordt de waarde omgezet in een getal zodat je deze kunt gebruiken net als bij een reguliere **Parse()** methode.

Voer onderstaande stappen uit

1. Maak een kopie van je RekenMachineFoutAfhandeling uit paragraaf 13.6 , open deze en hernoem de solution naar RekenMachineFoutAfhandelingCorrect.

2. Pas de methode **FoutAfhandeling()** aan en gebruik onderstaande code.

```
(double?, double?) FoutAfhandeling(string getal1String, string getal2String)
```

3. Met **(double?, double?)** geef je aan dat er twee waarden van het datatype double worden teruggegeven door de methode. Met het **vraagteken** achter double geef je aan dat de waarde ook **null** mag zijn. (dus dat de waarde leeg mag zijn.)

4. Verwijder de code welke staat in de scope van de methode **FoutAfhandeling()** zodat je de code opnieuw kunt schrijven.

5. Je start met het schrijven van de code waarin de **TryParse()** methode staat die gaat proberen de ingevoerde waarden om te zetten naar een double.

6. Er moeten twee getallen gecontroleerd worden dus de code komt twee keer terug. Neem onderstaande **twee regels** over.

```
static (double?, double?) FoutAfhandeling(string getal1String, string getal2String)
{
 bool getal1Check = double.TryParse(getal1String, out double result1);
 bool getal2Check = double.TryParse(getal2String, out double result2);
}
```

7. Je ziet dat de code voor beide waarden één keer wordt uitgevoerd. De opbouw van de code is als volgt.

```
bool getal1Check = double.TryParse(getal1String, out double result1);
```

8. Met `bool getal1Check` declareer je een variabele van het type bool. In dit type variabele kun je alleen de waarden **TRUE** of **FALSE** opslaan.

- De variabele **getal1Check** krijgt de waarde **TRUE** als het de **TryParse()** methode is gelukt om de waarde om te zetten.
- De variabele **getal1Check** krijgt de waarde **FALSE** als het de **TryParse()** methode NIET is gelukt om de waarde om te zetten. (De waarde zal dus FALSE worden op het moment dat de gebruiker een tekst (tien) in plaats van een getal (10) invult in de rekenmachine.)

9. De **TryParse()** methode krijgt twee argumenten mee namelijk **getal1String** en **out double result1**

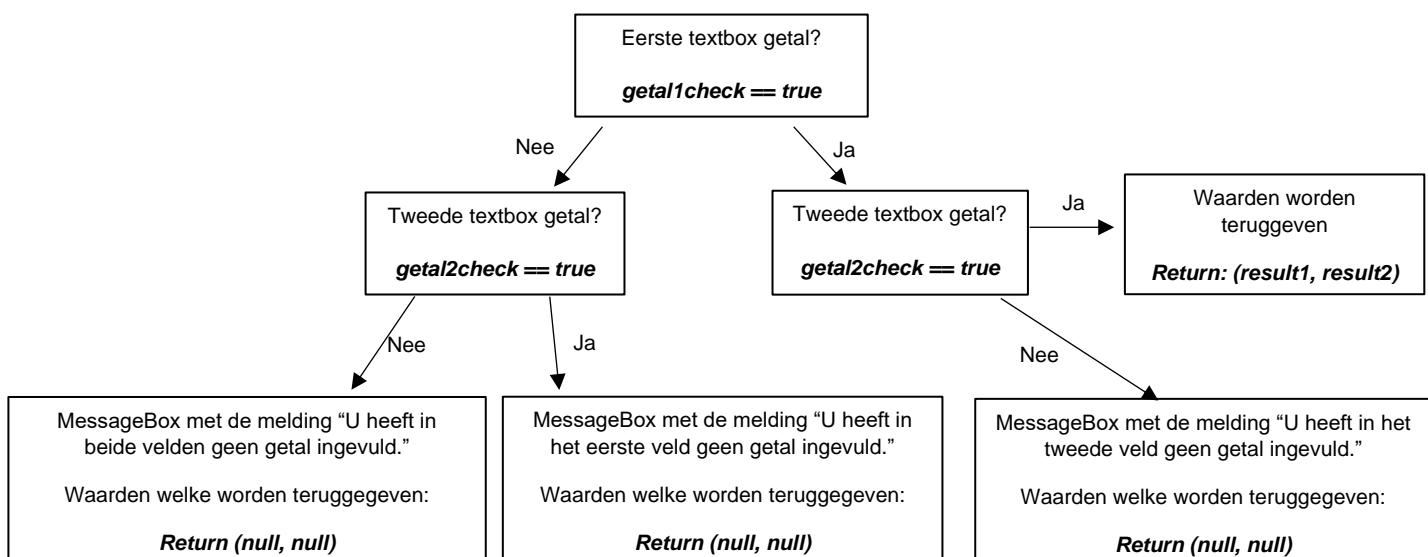
```
double.TryParse(getal1String, out double result1);
```

- **getal1String** is de waarde die wordt opgehaald uit het event wat je zo meteen nog gaat aanpassen.
- **double result1** is de variabele waarin de omgezette waarde wordt opgeslagen.
- **out** geeft aan dat de waarde 'vrij' moet worden gegeven om te gebruiken. Deze waarde wordt opgeslagen in **result1**. Dit is dus de waarde welke de **TryParse()** methode heeft omgezet en welke je kunt gebruiken in je **FoutAfhandeling()** methode.

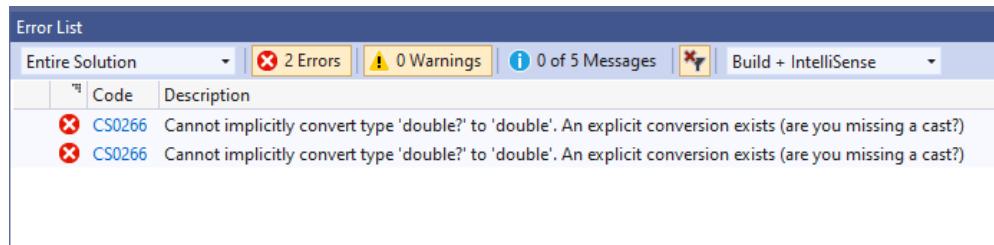
10. In de code heb je nu dus **vier** variabelen waarin een waarde wordt opgeslagen.

| variabele   | waarden       | Uitleg                                                                                                                                |
|-------------|---------------|---------------------------------------------------------------------------------------------------------------------------------------|
| getal1Check | TRUE of FALSE | Kun je gebruiken als voorwaarde in je IF statement.<br><b>IF (getal1Check == true)</b>                                                |
| getal2Check | TRUE of FALSE | Kun je gebruiken als voorwaarde in je IF statement.<br><b>IF (getal2Check == true)</b>                                                |
| result1     | getal of null | De omgezette waarde of geen waarde (null). Deze waarde kan teruggegeven worden aan je applicatie.<br><b>return (result1, result2)</b> |
| result2     | getal of null | De omgezette waarde of geen waarde (null). Deze waarde kan teruggegeven worden aan je applicatie.<br><b>return (result1, result2)</b> |

11. Maak nu gebruik van (geneste) IF..ELSE statement om de code voor de foutafhandeling af te maken. De IF..ELSE statements komen in de scope van **FoutAfhandeling()** onder de code die hierboven besproken is. Gebruik onderstaande schema om je hierbij te helpen.



12. Als je klaar bent met je IF statements zul je een aantal foutmeldingen zien zoals in onderstaande afbeelding.



Afbeelding 49 - Foutmelding

13. Dit komt omdat je in een deel van je code werkt met variabelen waarin een null waarde kan worden opgeslagen. Deze foutmelding geeft aan dat C# niet in staat is zelf deze waarde om te zetten. In de foutmelding kun je zien op welke regel er iets fout gaat. Ga naar deze regel en zet een vraagteken achter double zodat er null in kan worden opgeslagen. Doe dit voor alle meldingen in de Error List.

14. Als je klaar bent dan kun je de applicatie gaan testen. Voer onderstaande testen uit:

| Textbox1 | Textbox2 | Berekening | Gewenste Resultaat                                           | Werkt het correct? |
|----------|----------|------------|--------------------------------------------------------------|--------------------|
| 100      | 9,5      | +          | In het resultaat staat 109.5                                 | Ja / Nee           |
| 100      | 9,5      | -          | In het resultaat staat 90.5                                  | Ja / Nee           |
| 100      | 9,5      | /          | In het resultaat staat 10,5263157894737                      | Ja / Nee           |
| 100      | 9,5      | *          | In het resultaat staat 950                                   | Ja / Nee           |
| Honderd  | 9,5      | +          | Er komt een melding dat in het eerste veld geen getal staat. | Ja / Nee           |
| Honderd  | 9,5      | -          | Er komt een melding dat in het eerste veld geen getal staat. | Ja / Nee           |
| Honderd  | 9,5      | /          | Er komt een melding dat in het eerste veld geen getal staat. | Ja / Nee           |
| Honderd  | 9,5      | *          | Er komt een melding dat in het eerste veld geen getal staat. | Ja / Nee           |
| 100      | Negen    | +          | Er komt een melding dat in het tweede veld geen getal staat. | Ja / Nee           |
| 100      | Negen    | -          | Er komt een melding dat in het tweede veld geen getal staat. | Ja / Nee           |
| 100      | Negen    | /          | Er komt een melding dat in het tweede veld geen getal staat. | Ja / Nee           |
| 100      | Negen    | *          | Er komt een melding dat in het tweede veld geen getal staat. | Ja / Nee           |
| Honderd  | Negen    | +          | Er komt een melding dat in beide velden geen getal staat.    | Ja / Nee           |
| Honderd  | Negen    | -          | Er komt een melding dat in beide velden geen getal staat.    | Ja / Nee           |
| Honderd  | Negen    | /          | Er komt een melding dat in beide velden geen getal staat.    | Ja / Nee           |
| Honderd  | Negen    | *          | Er komt een melding dat in beide velden geen getal staat.    | Ja / Nee           |

15. Als alle tests goed zijn dan ben je klaar en heb je een nette foutafhandeling in je rekenmachine.  
 16. Maak een screenshot van jouw code en plaats deze op It's Learning.

### **16.3 Wat heb je geleerd?**

- Je weet wat een geneste IF Else statement is.
- Je kunt geneste IF Else statements inzetten in applicaties waarbij meer keuzemogelijkheden zijn.
- Je kunt een eenvoudige foutafhandeling module schrijven voor je applicatie met het gebruik van IF Else statements.
- Je weet dat je met een (?) achter het datatype aan kunt geven dat het datatype een NULL waarde mag bevatten.

## 17 Selecties - Switch

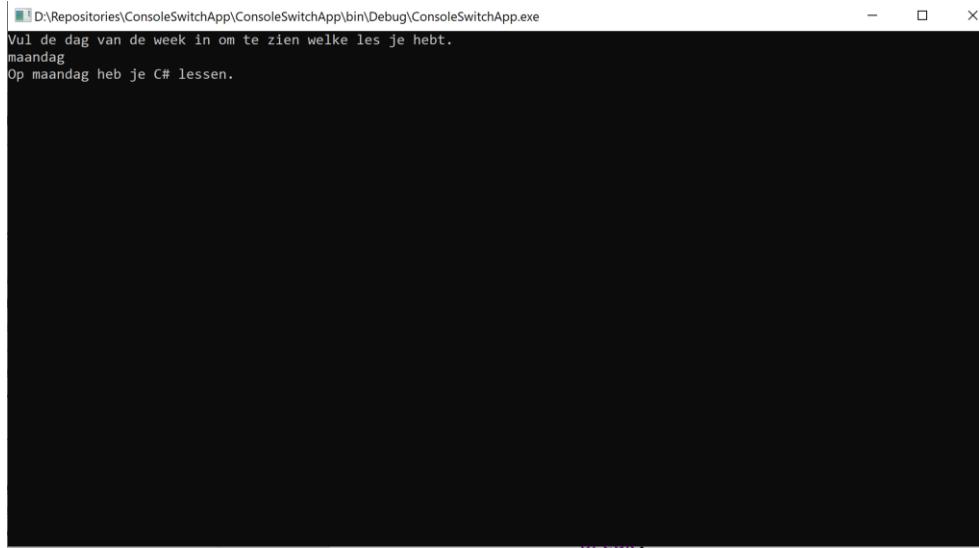
---

Een andere mogelijkheid om gebruik te maken van selecties is door gebruik te maken van het Switch-commando. Je bent bij het Switch commando beperkter in je keuzes want je kunt geen vergelijkingen gebruiken zoals kleiner dan of groter dan. Het Switch commando kun je gebruiken als er meerdere mogelijkheden zijn maar dat de antwoorden vast staan net als bij een meerkeuzetoets. Je geeft in je applicatie dus aan....als 'A' wordt ingevoerd, dan is 'B' het resultaat en als 'C' wordt ingevoerd dan is 'D' het resultaat.

De structuur voor het Switch-commando kun je hieronder zien aan de hand van een voorbeeld.

```
Console.WriteLine("Vul de dag van de week in om te zien welke les je hebt.");
string invoer = Console.ReadLine();
string uitvoer = "";

switch (invoer)
{
 case "maandag":
 {
 uitvoer = "C# lessen";
 break;
 }
 case "dinsdag":
 {
 uitvoer = "Project";
 break;
 }
 case "woensdag":
 {
 uitvoer = "Keuzedeel";
 break;
 }
 case "donderdag":
 {
 uitvoer = "HTML-CSS";
 break;
 }
 case "vrijdag":
 {
 uitvoer = "Python";
 break;
 }
}
Console.WriteLine("Op " + invoer + " heb je " + uitvoer + ".");
Console.ReadKey();
```



Afbeelding 50 - Console Switch App

- De console applicatie laat de vraag zien op het scherm.

```
Console.WriteLine("Vul de dag van de week in om te zien welke les je hebt.");
```

- De gebruiker wordt iets gevraagd en deze waarde wordt opgeslagen in de variabele **invoer**.

```
string invoer = Console.ReadLine();
```

- De waarde van de eerste variabele wordt gevuld met datgene wat de gebruiker gaat typen.
- De tweede variabele van het type string wordt gevuld met een lege string aangezien je de variabele anders niet aan kunt maken.

```
string uitvoer = "";
```

- In het Switch commando gaan in de gaten houden wat de waarde is van de variabele **invoer** (dus wat de gebruiker heeft getypt).

```
switch (invoer)
```

- Het programma gaat vervolgens van boven naar beneden één voor één voor alle mogelijkheden na.

```
case "maandag":
```

- Zodra het programma een 'match' heeft gevonden wordt de waarde in variabele **uitvoer** vervangen door de opgegeven waarde.

```
uitvoer = "C# lessen";
```

- Hierna moet worden aangegeven dat er niet verder gekeken hoeft te worden naar andere mogelijkheden (Case).

```
break;
```

- De applicatie gaat naar de code welke onder het Switch blok staat. Hierin staat aangegeven dat er een tekst op het scherm moet komen te staan waarin de invoer en uitvoer variabelen zijn opgenomen.

```
Console.WriteLine("Op " + invoer + " heb je " + uitvoer + ".");
```

- Om ervoor te zorgen dat het scherm niet direct verdwijnt en het resultaat niet is te zien krijgt de applicatie de opdracht om te wachten tot de gebruiker een toets indrukt.

```
Console.ReadKey();
```

## 17.1 Opdracht – Kassa

1. Maak een applicatie in Console of Windows Forms waarin gebruik wordt gemaakt van het Switch commando. Deze applicatie moet de prijs weergeven van de volgende items zodra deze worden ingegeven.

| Product    | Prijs |
|------------|-------|
| Tosti      | 6,50  |
| Uitsmijter | 7,95  |
| Koffie     | 2,25  |
| Melk       | 2,00  |
| Frisdrank  | 2,50  |

2. Bedenk een passende naam voor je applicatie.
3. Zorg ervoor dat de juiste naamgevingsconventies worden gebruikt.
4. Zorg ervoor dat er een foutmelding waarin staat ‘Dit item hebben we niet’ als de gebruiker een item ingeeft welke niet op de lijst staat. (*Dit is in een ConsoleApp wat lastiger dan in een Windows Forms App.*)
5. Test je applicatie.

## 17.2 Opdracht – Welke les heb ik vandaag?

1. Maak een nieuwe console applicatie aan met de titel LesVandaag.
2. Gebruik de code uit het voorbeeld van paragraaf 17.
3. Pas de code aan zodat de gebruiker NIETS hoeft in te vullen maar direct krijgt te zien welke les er vandaag op de planning staat.
4. Test je applicatie.

## 17.3 Eindopdracht – Welke les heb ik vandaag?

In de vorige opdracht heb je waarschijnlijk wat problemen gehad met de juiste benaming voor de dag van de week. Standaard wordt hier de Engelse benaming gebruikt. Je kunt dit natuurlijk aanpassen door je cases ook in het Engels weer te geven maar dat is eigenlijk een minder nette optie hier in Nederland. Er is ook een optie om het wel in het Nederlands te gebruiken door gebruik te maken van formatting waarbij de taal wordt gebruikt welke is ingesteld op de computer waarop de applicatie is gestart. Het probleem blijft echter dat je applicatie dan alleen maar werkt op machines waar Nederlands als standaard taal staat geïnstalleerd de cases zijn namelijk geschreven in het Nederlands.

Dit kun je natuurlijk voorkomen door gebruik te maken van nummers die aan een dag gekoppeld worden. Standaard is in C# de zondag de eerste dag van de week en zoals je weet beginnen we in de informatica bij 0. Zondag is dus de 0, Maandag wordt 1 enzovoort. Deze waarde kun je met DateTime ophalen.

- Pas je applicatie aan zodat onafhankelijk van de taal je het juiste resultaat krijgt.
- Maak gebruik van het internet om op te zoeken hoe je de juiste **type cast** kan gebruiken.
- Maak een screenshot van jouw code en plaats deze op It’s Learning.

## 17.4 Wat heb je geleerd?

- Je weet wat het switch statement doet en hoe je dit in je code kunt opnemen.
- Je weet wanneer je het Switch statement in kunt zetten.
- Je kunt bij het gebruik van datums ervoor zorgen dat het switchstatement ook werkt op een Engelse versie van Windows.

## 18 Comments

---

Bij het programmeren is het gebruikelijk dat je jouw code voorziet van zogenaamde comments. Een comment is een stukje uitleg over de code waarmee je bijvoorbeeld aangeeft wat er op een bepaalde regel gebeurt of waarom je als programmeur hebt gekozen voor een bepaalde oplossing. Comments gebruik je niet op elke regel want dan zou de code erg onoverzichtelijk worden. Daarnaast zijn veel stukken zo gebruikelijk dat extra uitleg niet nodig is. Bij het declareren van een variabele hoef je dan ook niet met een comment aan te geven dat je daar een variabele declareert want dat ziet iedere ontwikkelaar.

Een comment kun je eenvoudig aan je code toevoegen door een tekst vooraf te laten gaan door een dubbele slashforward ( // ). Het maakt niet uit waar dit in je code gebeurt en het is ook mogelijk om de comment achter bestaande code te plaatsen zoals hieronder te zien is.

```
string naamGebruiker = 'Jan'; //Hier maak in een variabele aan.
```

In bovenstaande code kun je zien dat de comment groen wordt weergegeven en achter de code staat. Visual Studio slaat dus alle tekst op die specifieke regel over zodra deze achter // staat. Dit geld dus niet meer voor de volgende regel. Wil je tekst op meerdere regels als commentaar gebruiken dan kun je gebruik maken van /\* en \*/. Alles wat hiertussen staat (dus over meerdere regels) wordt als commentaar gezien. Dit is handig want dan hoef je niet op iedere regel // te gebruiken. Dit is ook handig om tijdens het programmeren tijdelijk een stukje code 'uit' te zetten door het te markeren als comment.

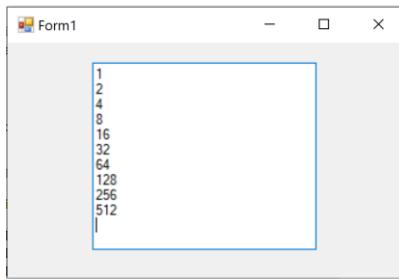
```
/*
Hiermee kun je hele blokken tekst als comment markeren.
Dit is handig.
*/
```

Vanaf nu is het de bedoeling dat je je code gaat voorzien van comments. In de opdrachten zal in het begin staan waar je dit moet doen. Later dien je zelf te bepalen waar comments moeten komen te staan.

## 19 Loops (herhalingen)

Bij het programmeren ga je vaak gebruik maken van loops (herhalingen). Door gebruik te maken van loops kun je als programmeur ervoor zorgen dat een bepaald stuk code meerdere keren achter elkaar wordt uitgevoerd. Dit voorkomt dat je dezelfde code meerdere keren opnieuw moet schrijven. Dit is handig want sommige opdrachten moeten misschien wel duizend keer worden uitgevoerd.

Een voorbeeld waarbij meteen duidelijk wordt dat het gebruik van loops handig is, is een lijst maken waarbij onderstaande getallen onder elkaar staan in een TextBox met als naam txbGetallen. De property Multiline van deze textbox zijn aangezet (met de property Multiline aan zodat meerdere regels mogelijk zijn).



Afbeelding 51 - TextBox Getallen

Zonder het gebruik van loops kan de code op de volgende manier geschreven worden.

```
txbGetallen.Text = "2" + "\r\n";
txbGetallen.Text = txbGetallen.Text + "4" + "\r\n";
txbGetallen.Text = txbGetallen.Text + "8" + "\r\n";
txbGetallen.Text = txbGetallen.Text + "16" + "\r\n";
txbGetallen.Text = txbGetallen.Text + "32" + "\r\n";
txbGetallen.Text = txbGetallen.Text + "64" + "\r\n";
txbGetallen.Text = txbGetallen.Text + "128" + "\r\n";
txbGetallen.Text = txbGetallen.Text + "256" + "\r\n";
txbGetallen.Text = txbGetallen.Text + "512" + "\r\n";
```

In deze code wordt aan de string in de property txbGetallen.Text telkens het volgende getal toegevoegd. Met de optie "`\r\n`" zorg je ervoor dat de cursor de volgende keer op de volgende regel staat.

### Opdracht

1. Maak een Windows Forms applicatie aan met als naam BinaryListApplicatie.
2. Maak de applicatie zoals hierboven beschreven.
3. Test je applicatie.

Je ziet dat je in deze code elke keer hetzelfde doet en dat is natuurlijk niet efficiënt. Nu kost het je nog niet veel tijd om deze code te schrijven maar stel je voor dat in de lijst door moet gaan tot 562.949.953.421.312, dan ben je wel even bezig. Dit is relatief eenvoudig op te lossen door in je code gebruik te maken van een loop. In de volgende paragrafen ga je kennis maken met een aantal verschillende loops welke je in kunt zetten.

## Itereren / Iteratie

Het gebruik van loops is onlosmakelijk verbonden met de termen itereren en iteratie dus ook in deze module ga je deze termen vaak terug zien. De term **iteratie** staat letterlijk voor herhaling en **itereren** dus voor herhalen. Pas je het concept loops of herhaling toe dan ben je dus aan het itereren en iedere herhaling die wordt uitgevoerd is een iteratie. Een stuk code wat negen keer wordt herhaald bestaat dus uit negen iteraties.

## 19.1 While loop

De **While** loop is een herhaling waarbij het programma in de gaten houdt of een bepaalde vergelijking ‘waar’ is. Dit kan bijvoorbeeld een waarde zijn die je hebt opgegeven. In het geval van de lijst uit voorgaande paragraaf zou de vergelijking kunnen zijn ....’doe iets tot de waarde 512 is’....want het laatste getal in de lijst is 512. Aan de hand van onderstaande stappen ga je de **BinaryListApplicatie** aanpassen zodat je gebruik maakt van de loop.

1. Maak van de bestaande code comment zodat deze niet wordt uitgevoerd.
2. In de tekst staat al aangegeven dat je iets moet gaan doen dus moet er een variabele gedeclareerd worden. Aangezien het om gehele getallen gaat kun je hier een integer voor gebruiken. Declareer een variabele met als naam **getal** van het type **integer** en zorg dat de waarde 1 is bij het start van de applicatie.
3. Maak een **While** loop aan door while te typen en tweemaal op de tabtoets te drukken.
4. De laatste waarde in de lijst is 512 dus er moet iets gebeuren tot het getal 512 is. In de vergelijking van de **While** loop kun je dus aangeven dat er iets moet gebeuren zolang het getal onder de 512 is. Je kunt hier onderstaande code voor gebruiken.

```
while (getal < 512)
{
}
```

5. Neem onderstaande code, die ervoor gaat zorgen dat het lijstje wordt gemaakt, over en zorg dat deze in de scope van de **While** loop komt te staan.

```
getal = getal * 2;
txbGetallen.Text = txbGetallen.Text + getal.ToString() + "\r\n";
```

6. **De code wordt als volgt uitgevoerd.**

7. Je start je programma en onderstaande code wordt uitgevoerd.

```
while (getal < 512)
{
 getal = getal * 2;
 txbGetallen.Text = txbGetallen.Text + getal.ToString() + "\r\n";
}
```

8. De waarde in variabele **getal** is 1 want dit heb je opgegeven in je code. De vergelijking **getal < 512** is dus WAAR en de code in de scope van de **While** wordt uitgevoerd.

```
while (getal < 512)
{
 getal = getal * 2;
 txbGetallen.Text = txbGetallen.Text + getal.ToString() + "\r\n";
}
```

9. Met **getal =** geef je aan dat je een nieuwe waarde wilt toekennen aan de variabele **getal**. Deze waarde moet worden: **getal \* 2;** Dit betekent dat je de huidige waarde van **getal** vermenigvuldigt met 2.  
Aangezien op dit moment de waarde van **getal** 1 is wordt de nieuwe waarde van **getal** dus 2.
10. In de volgende regel wordt deze nieuwe waarde op de lijst gezet zoals je dat ook in het voorbeeld hebt gezien.
11. De waarde van **getal** is nu 4 dus
  - a. Er wordt nog steeds voldaan aan de voorwaarde **getal < 512**.
  - b. De waarde van **getal** wordt 8 en wordt op de lijst gezet.
12. Dit proces wordt herhaald (LOOP) tot de vergelijking **getal < 512** niet meer waar is.
13. Op dat moment stopt het programma want er staat verder geen code meer welke uitgevoerd kan worden.
14. Je ziet nu dat de hoeveelheid code enorm meevalt. Wil je nu toch tot het getal 562.949.953.421.312 gaan dan hoef dit maar op één plek aan te geven.

## Schematisch ziet dat er als volgt uit.

Hieronder zie je drie keer de code van de **While** loop in je applicatie. De variabelen zijn vervangen door de waarden welke in de **variabelen** staat en zijn met **rood** aangegeven.

### \*\*\*\* Start applicatie \*\*\*\*

```
int getal = 1;

while (1 < 512)
{
 2 = 1 * 2;
 txbGetallen.Text = txbGetallen.Text + 2.ToString() + "\r\n";
}
```

2

### \*\*\*\* Tweede Iteratie \*\*\*\*

```
int getal = 2;

while (2 < 512)
{
 4 = 2 * 2;
 txbGetallen.Text = txbGetallen.Text + 4.ToString() + "\r\n";
}
```

2  
4

### \*\*\*\* Derde Iteratie \*\*\*\*

```
int getal = 4;

while (4 < 512)
{
 8 = 4 * 2;
 txbGetallen.Text = txbGetallen.Text + 8.ToString() + "\r\n";
}
```

2  
4  
8

## 19.2 Opdracht - While Loop

Je weet nu dat je met een **While** loop een opdracht kunt herhalen tot er is voldaan aan een voorwaarde die jij hebt gesteld in de code. Gebruik een **While** loop om onderstaande opdracht uit te voeren.

1. In Developer Land is er nog geringe overlast van ongedierte zoals muizen, ratten en kakkerlakken. De totale populatie is geschat op zo'n 1000 beestjes en dat lijkt veel maar valt mee voor zo'n groot park. Deze populatie van deze dieren neemt ieder jaar met 15 procent toe. Hier moet uiteraard actie op worden ondernomen want anders komen straks de bezoekers niet meer naar het park.
  - a. Maak een Windows Forms Applicatie met als naam **BerekenenOngediertePopulatie** waarmee inzichtelijk wordt over hoeveel jaar de populatie is gegroeid naar 10000.
  - b. Zorg ervoor dat te zien is wat de populatie is per jaar dus zorg ervoor dat op iedere regel staat



Afbeelding 52 - Ongedierte in het park

**Voorbeeld:**

Populatie in 2020: 1000

- c. Zorg ervoor dat dit jaar (dus het aantal 1000) ook in de lijst staat.
- d. Voorzie je code van Comments waarin je uitlegt wat er gebeurd in je code.

### 19.3 Do ... While

De **Do ... While** loop lijkt erg op de While loop welke je in de vorige paragrafen hebt gebruikt. Het grote voordeel van een **Do ... While** loop is dat de code in de loop minimaal één keer wordt uitgevoerd. Dit kun je zien aan onderstaande code waarin wederom het lijstje met getallen wordt gegenereerd.

```
private void btnStart_Click(object sender, EventArgs e)
{
 int getal = 1;
 do
 {
 getal = getal * 2;
 txbGetallen.Text = txbGetallen.Text + getal.ToString() + "\r\n";

 } while (getal < 512);
```

Voordat de controle `while (getal < 512)` wordt uitgevoerd is de code ervoor al een keer uitgevoerd. Dit kan handig zijn in sommige situaties. In Afbeelding 53 kun je het verschil zien. Roadrunner heeft de rand gedetecteerd en stopt met rennen. Wile E. Coyote detecteert de rand pas nadat hij al te ver is gegaan.



Afbeelding 53 - While E. Coyote

Op de volgende pagina wordt de code doorlopen waarbij in iedere iteratie de waarde van de variabele staat ingevuld.

```

private void btnStart_Click(object sender, EventArgs e)
{
 int getal = 1;
 do
 {
 getal = getal * 2;
 txbGetallen.Text = txbGetallen.Text + getal.ToString() + "\r\n";
 } while (getal < 512);

```

#### \*\*\*\* Start applicatie \*\*\*\*

```

private void btnStart_Click(object sender, EventArgs e)
{
 int getal = 1;
 do
 {
 2 = 1 * 2;
 txbGetallen.Text = txbGetallen.Text + 2.ToString() + "\r\n";
 } while (2 < 512); //TRUE dus doorgaan met itereren

```

A screenshot of a Windows application window titled 'Start applicatie'. Inside the window, the text '2' is displayed, indicating the initial value of the variable 'getal'.

#### \*\*\*\* Tweede Iteratie \*\*\*\*

```

private void btnStart_Click(object sender, EventArgs e)
{
 int getal = 2;
 do
 {
 4 = 2 * 2;
 txbGetallen.Text = txbGetallen.Text + 4.ToString() + "\r\n";
 } while (4 < 512); //TRUE dus doorgaan met itereren

```

A screenshot of a Windows application window titled 'Start applicatie'. Inside the window, the text '2' and '4' are displayed, indicating the values of 'getal' after the first iteration.

#### \*\*\*\* Derde Iteratie \*\*\*\*

```

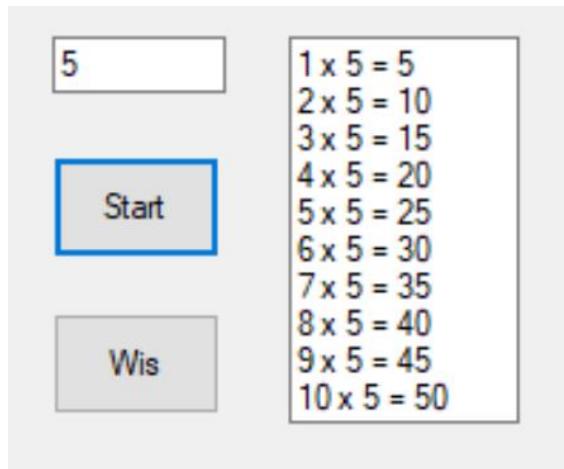
private void btnStart_Click(object sender, EventArgs e)
{
 int getal = 4;
 do
 {
 8 = 4 * 2;
 txbGetallen.Text = txbGetallen.Text + 8.ToString() + "\r\n";
 } while (8 < 512); //TRUE dus doorgaan met itereren

```

A screenshot of a Windows application window titled 'Start applicatie'. Inside the window, the text '2', '4', and '8' are displayed, indicating the values of 'getal' after the second iteration.

## 19.4 Opdracht – Do...While Loop

In onderstaande applicatie kan een gebruiker een getal ingeven en dan wordt de tafel van dit getal weergegeven in de lijst.



Afbeelding 54 - Tafels Applicatie

1. Maak een nieuwe Windows Forms applicatie en noem deze TafelsGenerator.
2. Voorzie je code van Comments waarin je uitlegt wat er gebeurd in je code.
3. Gebruik bovenstaande lay-out maar voeg labels toe zodat een gebruiker informatie krijgt over de knoppen en velden.
4. De gebruiker moet een willekeurig getal in kunnen voeren.
5. De vermenigvuldiging loopt van 1 x t/m 10 x het opgegeven getal.
6. De Wis knop zorgt ervoor dat er geen tekst meer staat in beide TextBoxen zodat er een nieuwe tafel gegenereerd kan worden. (Hiervoor moet je dus een nieuw event aanmaken want dit is een andere knop).
7. Als je klaar bent test je de applicatie met verschillende getallen.

## 19.5 For Loop

De For loop is een andere manier om code te herhalen waarbij het vooral handig is aangezien de hele code voor de herhaling in één regel staat beschreven. In een for loop moet wel vooraf bepaald worden hoe vaak de code doorlopen moet worden aangezien in dit type loop wordt gewerkt met een teller waarbij de teller iedere 'ronde' met één wordt verhoogd of verlaagd.

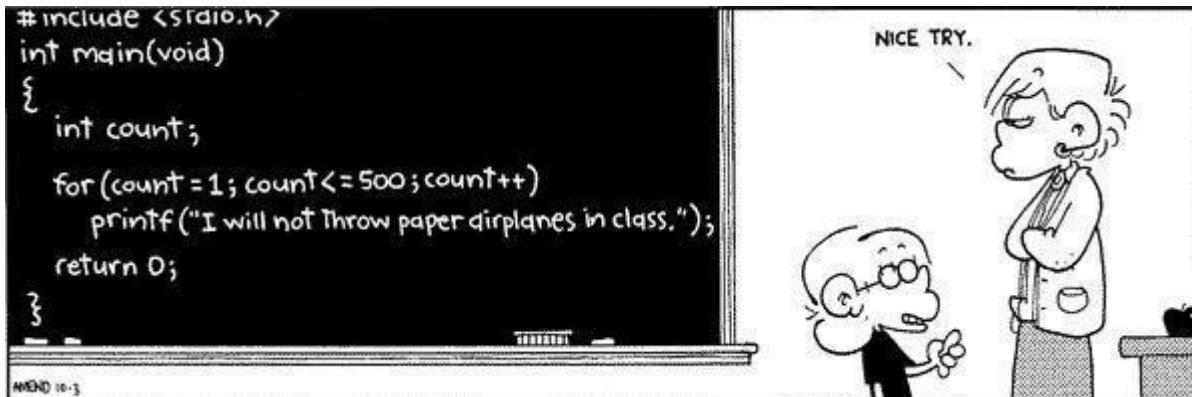
Gebruik je de For loop om hetzelfde lijstje met getallen te maken zoals in voorgaande paragrafen dan ziet het er als volgt uit.

```
int getal = 1;

for (int teller = 0; teller < 9; teller++)
{
 getal = getal * 2;
 txbGetallen.Text = txbGetallen.Text + getal.ToString() + "\r\n";
}
```

- In de For loop wordt de variabele **teller** van het type integer gedeclareerd en de waarde wordt op 0 gezet.
- De voorwaarde staat op **teller < 9** dus de code wordt herhaald tot de waarde van de variabele teller 9 is.
- **Teller++** zorgt ervoor dat de variabele teller iedere 'ronde' wordt opgehoogd met 1.

Op de volgende pagina wordt de For loop doorlopen waarbij de waarden van de variabelen zijn ingevuld in iedere iteratie.



Afbeelding 55 - For Int - Zoek de fout...

```

int getal = 1;

for (int teller = 0; teller < 9; teller++)
{
 getal = getal * 2;
 txbGetallen.Text = txbGetallen.Text + getal.ToString() + "\r\n";
}

```

**\*\*\*\* Start applicatie \*\*\*\***

*De waarde van getal = 1 en de teller staat nog op 0*

```

int getal = 1;

for (int teller = 0; teller < 9; teller++)
{
 2 = 1 * 2;
 txbGetallen.Text = txbGetallen.Text + 2.ToString() + "\r\n";
}

```



**\*\*\*\* teller < 9 dus → Tweede iteratie \*\*\*\***

*De waarde van getal = 2 geworden in de vorige ronde en de teller is opgehoogd naar 1*

```

int getal = 2;

for (int teller = 1 ; teller < 9; teller++)
{
 4 = 2 * 2;
 txbGetallen.Text = txbGetallen.Text + 4.ToString() + "\r\n";
}

```



**\*\*\*\* teller < 9 dus → Derde iteratie \*\*\*\***

*De waarde van getal = 4 geworden in de vorige ronde en de teller is opgehoogd naar 2*

```

int getal = 4;

for (int teller = 2; teller < 9; teller++)
{
 8 = 4 * 2;
 txbGetallen.Text = txbGetallen.Text + 8.ToString() + "\r\n";
}

```

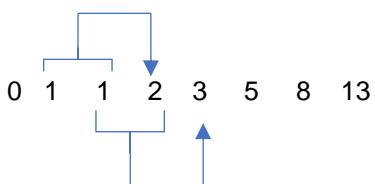


**\*\*\*\* Dit gaat door tot teller < 9 niet meer waar is. \*\*\*\***

## 19.6 Eindopdracht - For Loop

De rij van Fibonacci is een rij met getallen waarbij het opvolgende getal een opsomming is van de twee getallen ervoor. Deze ‘vermenigvuldiging’ of groei zie je overal in de natuur en de rij wordt dan ook vaak gevonden in onderzoeken of juist toegepast bij ontwerpen. Ook bij het programmeren kom je de rij van Fibonacci regelmatig tegen.

Het begin van de rij van Fibonacci zie er als volgt uit.



- Het tweede getal is 1 omdat de twee voorgaande getallen bij elkaar opgeteld ( $0 + 1$ ) 1 is.
- Het derde getal is 2 omdat de twee voorgaande getallen bij elkaar opgeteld ( $1 + 1$ ) 2 is.
- Het vierde getal is 3 omdat de twee voorgaande getallen bij elkaar opgeteld ( $1 + 2$ ) 3 is.
- Het vijfde getal is 5 omdat de twee voorgaande getallen bij elkaar opgeteld ( $2 + 3$ ) 5 is.

1. Maak een applicatie met een For Loop die de eerste 25 getallen laat zien van de rij van Fibonacci.
2. Voorzie je code van Comments waarin je uitlegt wat er gebeurd in je code.
3. Maak een screenshot van jouw code en plaats deze op It's Learning,

## 19.7 Wat heb je geleerd?

- Je weet wat een loop is.
- Je kent de While loop, Do While loop en de For loop.
- Je weet het verschil tussen een While loop en een Do While loop.
- Je kunt bepalen wanneer je welke loop in kunt zetten.
- Je weet hoe de code wordt doorlopen bij het gebruik van een loop.

# 20 Gegevens, dataset en Informatie

---

Applicaties draaien altijd om het verwerken van of het werken met gegevens. In sommige applicaties zoals je rekenmachine of de console applicaties die je hebt gebouwd gaat het om een beperkt aantal gegevens maar meestal werk je met grotere hoeveelheden aan gegevens. De gegevens waar je mee hebt gewerkt beperken zich nu nog tot datgene wat een gebruiker invoert in de applicatie zoals in een textbox of wanneer hier om wordt gevraagd met een **Console.ReadLine()**. In het hoofdstuk met loops heb je al met meerdere gegevens en waarden gewerkt maar ook dit is nog redelijk beperkt.

Bij het programmeren kun je twee verschillende gegevenstypen onderscheiden en dat zijn interne- en externe gegevens. Interne gegevens zijn of worden letterlijk opgeslagen in je code. Externe gegevens staan in bestanden of in een database buiten je applicatie. In deze module ligt de nadruk op het gebruik van interne gegevens en ga je gebruik maken van arrays en lists om met grotere hoeveelheden data te werken. In de module PRC-II die je na deze module gaat volgen ga je aan de slag met het gebruik van externe gegevens.

Voor je aan de slag gaat is het handig om te weten dat je eerst snapt wat gegevens en informatie zijn. Dit helpt je namelijk later bij het ontwikkelen om de juiste keuzes te maken. Indien een klant vraagt om gegevens op te halen met de applicatie dan is dit een stuk eenvoudiger dan wanneer hij vraagt dat hij informatie wil zien.

## 1. Gegevens

Gegevens of data zijn woorden, getallen, tekens of symbolen.

Voorbeelden

Auto  
1998  
20012

## 2. Dataset

Een dataset is een verzameling van gegevens welke bij elkaar horen. Vaak is dit in een tabelvorm maar dit kan ook een andere structuur hebben.

Voorbeeld:

|        |       |              |                |      |
|--------|-------|--------------|----------------|------|
| Nissan | Rood  | Hatchback    | Automaat       | 2012 |
| Toyota | Grijs | Stationwagon | Handgeschakeld | 2005 |

## 3. Informatie

Data die in een bepaalde context geplaatst wordt zodat de gegevens geïnterpreteerd kunnen worden. Een set met gegevens wordt dus pas informatie als duidelijk is wat er staat en de lezer iets met deze data kan.

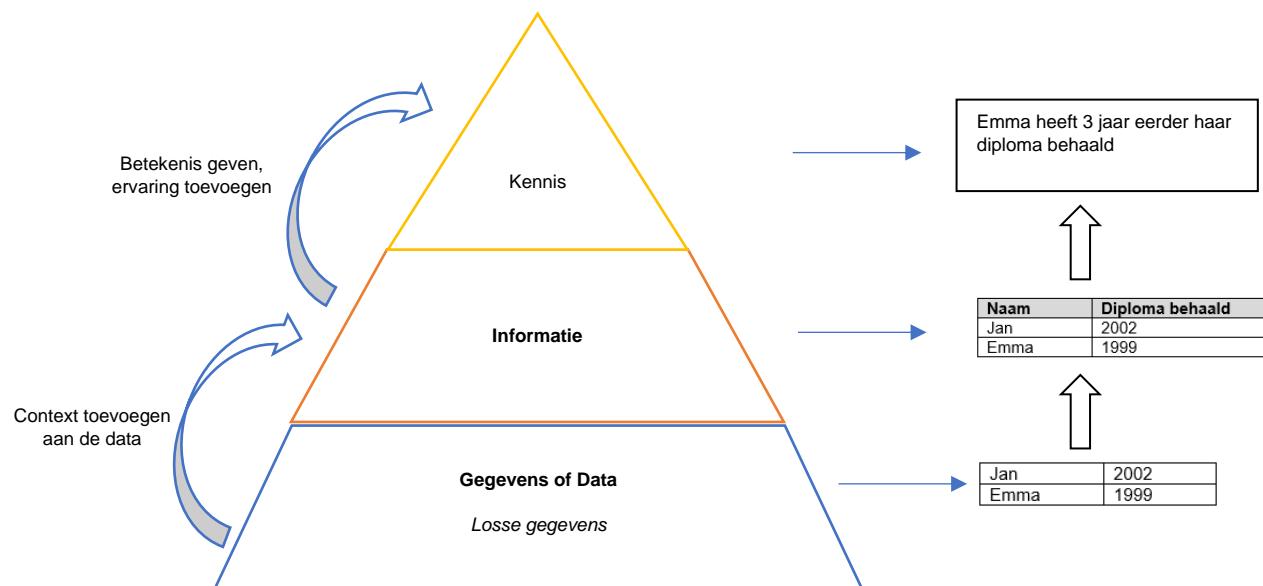
Voorbeeld:

|      |      |
|------|------|
| Jan  | 2002 |
| Emma | 1999 |

Bovenstaande gegevens zijn een dataset maar nog GEEN informatie want je weet niet wat de gegevens inhouden. Wat is je eerste gedachte als je deze gegevens ziet? Waarschijnlijk dat het om de namen van personen gaat met de geboortedatum. Laten we nu van deze gegevens informatie maken:

| Naam | Diploma behaald |
|------|-----------------|
| Jan  | 2002            |
| Emma | 1999            |

Nu zie je dat het duiden van de gegevens erg belangrijk is. In onderstaande driehoek staat schematisch weergegeven.



Een groot bedrijf als Google werkt op dezelfde manier. Google verzamelt bergen met gegevens van mensen. Aan deze gegevens wordt een context toegevoegd zodat er structuur ontstaat in de gegevens. De beschikbare informatie wordt onderzocht en er wordt betekenis gegeven. Google heeft op dat moment de kennis waarmee ze iets kan doen zoals bijvoorbeeld het bedenken van nieuwe oplossingen of het aanpassen van bestaande oplossingen.

Niet alleen bij het ophalen of werken met gegevens is het dus belangrijk om duidelijk aan te geven wat de gebruiker ziet. Dit kan zijn een tekst op een knop, het toevoegen van een kolomkop of eenvoudig een label voor een textbox plaatsen waarin staat aangegeven wat er in de textbox komt, of moet komen te staan.

## 21 Array

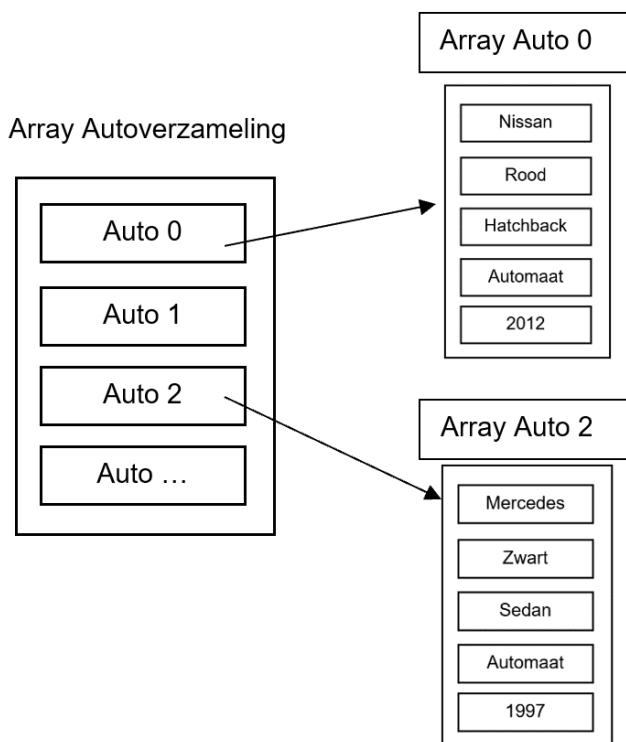
Een [Array](#) is een verzameling van gegevens van hetzelfde typen welke gerangschikt staan in een tabel. Is er maar één kolom aanwezig in deze tabel dan wordt er gesproken over een één dimensionale tabel. Hieronder is een weergave van een één dimensionale array te zien.

|            |
|------------|
| Nissan     |
| Toyota     |
| Mercedes   |
| Volkswagen |
| Opel       |
| Opel       |

Het is ook mogelijk om meerdere gegevens achter elkaar te plaatsen in een array, dan wordt er gesproken over een multi dimensionale array. Dit is dus eigenlijk een verzameling van arrays. Hieronder is voorbeeld van een multi dimensionale array te zien.

| index | 0          | 1     | 2            | 3              | 4    |
|-------|------------|-------|--------------|----------------|------|
| 0     | Nissan     | Rood  | Hatchback    | Automaat       | 2012 |
| 1     | Toyota     | Grijs | Stationwagon | Handgeschakeld | 2005 |
| 2     | Mercedes   | Zwart | Sedan        | Automaat       | 1997 |
| 3     | Volkswagen | Wit   | Stationwagon | Automaat       | 2016 |
| 4     | Opel       | Blauw | Hatchback    | Handgeschakeld | 1990 |
| 5     | Opel       | Grijs | Hatchback    | Automaat       | 2003 |

De inhoud van deze tabel kan worden opgeslagen in de array ‘autoverzameling’ wat een verzameling is van arrays waarin verschillende auto’s en de kenmerken van deze auto’s zijn opgeslagen.



Je kunt een [Array](#) in je applicatie gebruiken om een verzameling met gegevens op te slaan zonder voor alle gegevens een afzonderlijke variabele aan te maken. De gegevens in een [Array](#) zijn net als Excel te benaderen via een adres wat gewoon het index nummer (rij nummer) en kolomnummer is. In een [Array](#) wordt één gegeven op een adres een **element** genoemd.

**In dit hoofdstuk ga je aan de slag met de één dimensionale array. Wordt er in de komende paragrafen gesproken over een array dan wordt hier dus een één dimensionale array bedoelt.**

## 21.1 Het maken van een Array

Een [Array](#) dien je net als een variabele te declareren waarbij je ook direct aangeeft wat voor gegevens er in de [Array](#) worden opgeslagen. Naast het declareren wordt een [Array](#) gevuld met gegevens en dit wordt initialiseren genoemd.

Het declareren van een [Array](#) gaat hetzelfde als bij het declareren van een variabele. Je geeft aan wat het datatype is en wat de naam wordt van de [Array](#). Achter het datatype gebruik je twee rechte haken [ en ] waarmee je aangeeft dat het om een array gaat en voor het initiëren gebruik je twee accolades { en }. Een ééndimensionale [Array](#) waarbij er maar één kolom beschikbaar is kun je dat op de volgende manier doen.



Het resultaat van de Array is (de linker kolom geeft de locatie van het gegeven (index) aan:

| index | 0          |
|-------|------------|
| 0     | Nissan     |
| 1     | Toyota     |
| 2     | Mercedes   |
| 3     | Volkswagen |
| 4     | Opel       |
| 5     | Opel       |

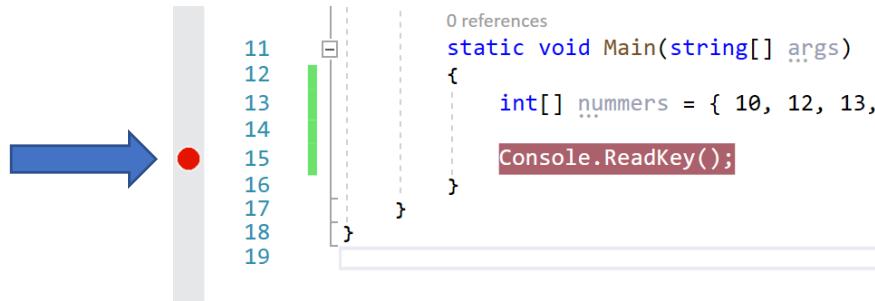
Het is ook mogelijk om een lege array aan te maken welke je later in je applicatie pas gaat vullen. Dit gebruik je bijvoorbeeld bij het verzamelen van gegevens welke door een gebruiker worden ingevoerd. Een lege array maak je op dezelfde manier aan als het voorbeeld hierboven alleen worden uiteraard de gegevens niet ingevuld. In plaats daarvan geef je aan hoe groot de Array moet worden.

```
string[] merkenLeeg = new string[6];
```

In plaats dat je tussen { } aangeeft wat er in de Array moet komen te staan geef je dus aan dat er 6 nieuwe strings komen in de array (welke nog niet gevuld zijn).

## 21.2 Opdracht – Aanmaken van een Array

1. Maak een console applicatie aan en voeg een Array toe met als naam **numbers**.
2. Vul deze array met minimaal 5 nummers.
3. Gebruik de ReadKey() methode zodat de console niet direct wordt afgesloten na het starten van je applicatie.
4. Klik met je muis in de grijze balk links van de regel waarin je de ReadKey() methode hebt gebruikt. Dit wordt een breekpunt genoemd waarmee je aangeeft dat de applicatie de code moet uitvoeren tot dit specifieke punt. Dit ga je later nog heel vaak gebruiken om fouten uit je code te halen.



Afbeelding 56 - Breekpunt

5. Start je applicatie en je zult zien dat de console wordt geopend maar dat je direct weer naar Visual Studio gaat.
6. Links onderin Visual Studio zie je een venster met de naam **Locals**. In dit venster kun je de gegevens terugvinden welke in je applicatie gebruikt worden. De Array **numbers** welke je hebt aangemaakt staat hier ook.
7. Aangezien je programma nog loopt is deze Array op dit moment gevuld en kun je hier zien wat de inhoud is.
8. Klik in het venster **Locals** op het driehoekje voor **numbers** en je zult de inhoud zien van je Array inclusief het datatype.

| Name    | Value       | Type     |
|---------|-------------|----------|
| args    | {string[0]} | string[] |
| numbers | {int[8]}    | int[]    |
| [0]     | 10          | int      |
| [1]     | 12          | int      |
| [2]     | 13          | int      |
| [3]     | 14          | int      |
| [4]     | 15          | int      |
| [5]     | 16          | int      |
| [6]     | 17          | int      |
| [7]     | 18          | int      |

Afbeelding 57 - Array Nummer in het Locals vensters

9. Voeg een nieuwe Array toe met de naam **numbersTwee** daarin een lijst met daarin de volgende waarden: -12, -10, -5, 0, 1, 6, 8 , 10.
10. Voeg een Array toe met de naam **gebrokenGetallen** met daarin de waarden 16,45 , 18,98 , 897,78 , 12,11 , 78976,78.

## 11. Start je applicatie en controleer de inhoud van je Arrays.

Bij het gebruik van het datatype float voor het aanmaken van een array dan krijg je een foutmelding. Dit komt doordat Microsoft standaard een gebroken getal ziet als een Double. Plaats je een gebroken getal dus in een array van het type float dan gaat het fout omdat dit niet past. (Een double is namelijk groter dan een float). Je kunt dit voorkomen door achter het getal een f te typen waarmee je aangeeft dat het bereffende getal een float is en niet een double. Dus in plaats van 1.1 , 12.3, 56.25 typ je 1.1f , 12.3f, 56.25f en je zult zien dat het dan wel werkt.

### 21.3 Opdracht - Ophalen van gegevens uit een Array

Wil je gebruik maken van gegevens uit een Array dan kun je deze gegevens net als een variabele eenvoudig ophalen. De code welke je hiervoor gebruikt is de naam van de variabele met daarachter tussen rechte haken het nummer van de rij waarin de gegevens staan welke je wilt gebruiken. Als voorbeeld gebruiken we weer de lijst met automerken.

De code om het 1<sup>e</sup> merk uit de array te gebruiken is:

```
merken[0]
```

Om het merk in de console te laten verschijnen wordt dus:

```
Console.WriteLine(merken[0]);
```

1. Pas je applicatie uit paragraaf 0 aan
2. Zorg ervoor dat in de console de volgende drie teksten verschijnen bij het starten van de applicatie.
  - a. 'Het tweede nummer uit de Array nummers is <het opgehaalde getal>.'
  - b. 'Het nummer <het opgehaalde getal> is het eerste nummer uit de Array gebrokenGetallen.'
  - c. '<het opgehaalde getal> is het vierde nummer uit de Array nummersTwee.'
3. Test je applicatie en controleer of je de juiste getallen op je scherm krijgt.
4. Voorzie je code van Comments waarin je uitlegt wat er gebeurd in je code.

### 21.4 Opdracht - Arrays manipuleren

In de vorige opdrachten ben je aan de slag gegaan met het aanmaken van een array en het ophalen van de gegevens. Je zult echter ook regelmatig de array aan willen passen om er bijvoorbeeld meer gegevens in te zetten of de array leeg te gooien. Dit doe je door gebruik te maken van de class array welke onderdeel is van de system library.

De class array heeft verschillende methodes welke je kunt gebruiken voor het werken met jouw array.

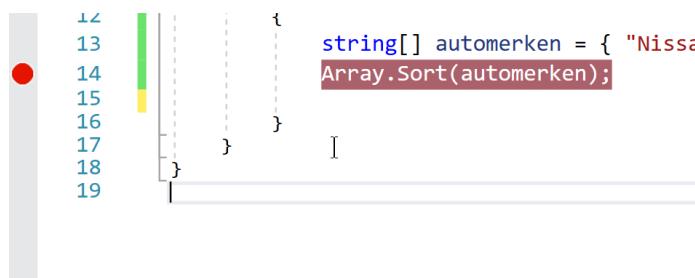
1. Maak een nieuwe console applicatie aan met als naam ArrayManipuleren en maak een array aan automerken zoals je dit voorgaande paragrafen hebt gedaan.
2. In de volgende delen ga je een aantal bewerkingen uitvoeren op deze array.

#### 21.4.1 Sorteren van de gegevens

De gegevens in jouw array staan niet netjes in alfabetische volgorde en dit ziet er eigenlijk niet echt netjes uit. Wordt de array groter en toon je de inhoud in je applicatie dan wil je dit eigenlijk wel want dan kan de gebruiker sneller zoeken in de lijst op zijn scherm. Om deze gegevens op je scherm toch netjes op alfabetische volgorde te krijgen kun je gebruik maken van de methode `sort()` welke onderdeel is van de class `Array` welke je standaard tot je beschikking hebt.

```
string[] automerken = { "Nissan", "Toyota", "Mercedes", "Volkswagen", "Opel", "Opel" };
Array.Sort(automerken);
```

3. Neem de code over en controleer de inhoud van de array als je de applicatie uitvoert door een breakpoint te plaatsen in je code.



Afbeelding 58 - Breakpoint

4. Start je code en kijk onderin bij locals wat de inhoud is van je array `automerken`.
5. Als het goed is staan de merken nog niet in alfabetische volgorde want je hebt aangegeven dat je applicatie moet stoppen zodra deze regel bereikt is want bij het zetten van een breakpoint wordt alleen de code vóór het breakpoint uitgevoerd. Je kunt nu wel controleren wat de inhoud is van de `automerken` want de array is al wel aangemaakt.
6. Plaats het breakpoint nu op de regel onder de code welke ervoor zorgt dat de gegevens gesorteerd worden en controleer nogmaals de inhoud van de array.
7. Als het goed is staan de gegevens nu in alfabetische volgorde in de array.
8. Gebruik nu in plaats van de `sort()` methode de `reverse()` methode en kijk wat die methode doet.

#### 21.4.2 Gegevens toevoegen aan een array

Toevoegen van gegevens aan een array gaat in C# niet zo makkelijk. Dit komt omdat een array meer bedoeld is voor statische gegevens, dus gegevens die je als programmeur in je applicatie plaatst welke niet meer gewijzigd of aangevuld worden. Je zult op het internet dan ook al snel lezen dat je beter gebruik kunt maken van **lists** op het moment dat je gegevens dynamischer zijn. Het gebruik van **lists** komt later in dit hoofdstuk nog aan bod. Het is echter wel degelijk mogelijk om items aan een array toe te voegen maar je zult zien dat dit best omslachtig is.

Bij het aanmaken van de array is bepaald hoe groot de array is. De array `automerken` is 6 rijen groot. Wil je nu een nieuw automerk toevoegen dan kun je dit niet zomaar uitvoeren omdat er geen ruimte meer is in de array. Je zult dus eerst de grootte van de array moeten passen voordat je een nieuw item toe kunt voegen.

Hiervoor kun je de methode **Array.Resize()**

```
string[] automerken = { "Nissan", "Toyota", "Mercedes", "Volkswagen", "Opel", "Opel" };
```

```
Array.Sort(automerken);
Array.Resize(ref automerken, 7);
```

9. Voeg de code toe en controleer met een breakpoint of je array groter is geworden.

|     | automerken   |
|-----|--------------|
| [0] | "Mercedes"   |
| [1] | "Nissan"     |
| [2] | "Opel"       |
| [3] | "Opel"       |
| [4] | "Toyota"     |
| [5] | "Volkswagen" |
| [6] | null         |

Afbeelding 59 - Nieuw item in array

10. Je ziet dat er nu een leeg item (null) is toegevoegd aan je array en hierin kan een nieuw merk worden geplaatst.

Met de methode `SetValue()` kun je een waarde in je array wijzigen. Door nu de waarde `null` te wijzigen in een merk heb je nieuwe waarde toegevoegd.

11. Gebruik onderstaande code om de waarde aan te passen.

```
string[] automerken = { "Nissan", "Toyota", "Mercedes", "Volkswagen", "Opel", "Opel" };
Array.Sort(automerken);
Array.Resize(ref automerken, 7);
automerken.SetValue("Audi", 6);
```

12. In de code geef je dus aan dat de waarde bij index 6 moet wijzigen naar Audi. Je kunt op deze manier uiteraard ook andere waarden wijzigen.
13. Voeg de code toe en controleer of de Audi in de Array staat bij het uitvoeren van het programma.
14. Voorzie je code van Comments waarin je uitlegt wat er gebeurd in je code.

## 21.5 Loops en arrays

Het grote voordeel van het gebruik van arrays is dat je in één variabele meerdere waarden op kunt slaan. Om de waarden uit deze variabele te halen maak je gebruik van loops. Zoals je eerder hebt gezien zijn er meerdere loops en ook bij het werken met arrays kun je verschillende loops gebruiken. Voor een aantal van de loops dien je echter te weten hoe vaak de code herhaald moet worden of onder welke waarde een bepaalde variabele moet blijven.

Bij het werken met arrays is dat vaak het aantal rijen waaruit de array bestaat. Je wilt immers dat je loop iedere rij in de array bekijkt en hier iets mee doet. Het aantal rijen waaruit een array bestaat kun je eenvoudig ophalen uit de property **Length**.

Kijken we weer naar de array met automerken dan zal **merken.Length** een waarde van 6 teruggeven want er staan 6 rijen in de array.

|   |            |
|---|------------|
| 0 | Nissan     |
| 1 | Toyota     |
| 2 | Mercedes   |
| 3 | Volkswagen |
| 4 | Opel       |
| 5 | Opel       |

Wil je nu de waarden uit deze array op het scherm tonen dan kun je verschillende loops gebruiken.

### Optie 1 - While loop

```
string[] automerken = { "Nissan", "Toyota", "Mercedes", "Volkswagen", "Opel", "Opel" };
int teller = 0;
int aantalRijen = automerken.Length;

while (teller < aantalRijen)
{
 Console.WriteLine(automerken[teller]);
 teller++;
}

Console.ReadKey();
```

Voor de **while** loop wordt met **automerken.Length** het totale aantal rijen opgehaald in de array en dit aantal wordt in de variabele **aantalRijen** gezet. Deze variabele wordt gebruikt om te bepalen hoe vaak de code moet worden herhaald.

De variabele **teller** in deze code wordt **twee** keer gebruikt.

1. Teller wordt gebruikt om te kijken of de code nog moet worden uitgevoerd.  
`while (teller < aantalRijen)`
2. Teller wordt gebruikt om aan te geven uit welke rij (index) het automerk moet worden opgehaald.  
`Console.WriteLine(automerken[teller]);`

Let op dat in de code `teller++;` wordt gebruikt om iedere ronde de waarde van teller op te hogen met 1.

## Optie 2 – for loop

```
string[] automerken = { "Nissan", "Toyota", "Mercedes", "Volkswagen", "Opel", "Opel" };
int aantalRijen = automerken.Length;
for (int teller = 0; teller < aantalRijen; teller++)
{
 Console.WriteLine(automerken[teller]);
}
Console.ReadKey();
```

Net als bij de **while** loop wordt met **automerken.Length** het totaal aantal rijen opgehaald en ook hier wordt de variabele **teller** twee keer gebruikt

1. Teller wordt gebruikt om te kijken of de code nog moet worden uitgevoerd.  
`for (int teller = 0; teller < aantalRijen; teller++)`
2. Teller wordt gebruikt om aan te geven uit welke rij (index) het automerk moet worden opgehaald.  
`Console.WriteLine(automerken[teller]);`

## Optie 3 – foreach loop

```
string[] automerken = { "Nissan", "Toyota", "Mercedes", "Volkswagen", "Opel", "Opel" };
foreach (var merk in automerken)
{
 Console.WriteLine(merk);
}
Console.ReadKey();
```

De foreach loop ben je in het vorige hoofdstuk nog niet tegengekomen. Het is een loop die erg handig is als je alle rijen in een array ‘langs wilt lopen’ en iets met de inhoud wilt doen. Je hebt in deze loop dus ook geen teller nodig aangezien de loop door gaat tot alle rijen zijn doorlopen.

In deze loop wordt de waarde uit de array **automerken** iedere ronde in de variabele **merk** geplaatst.

```
foreach (var merk in automerken)
```

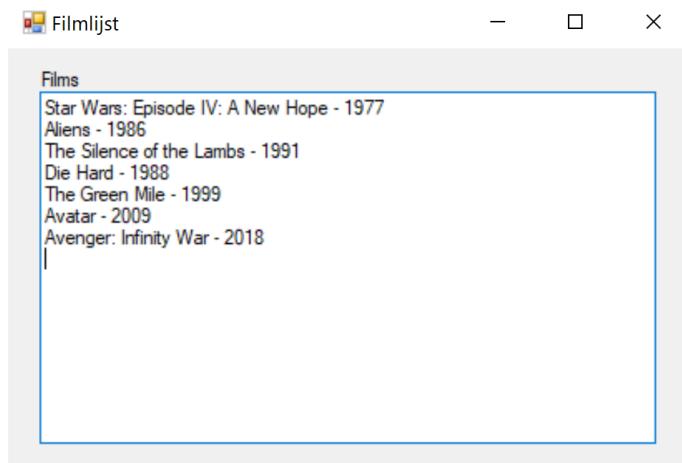
Deze variabele kun je dan weer gebruiken voor in de code binnen de loop.

## 21.6 Opdracht - Loops en Arrays

1. Maak een Windows Forms applicatie aan en noem deze `filmsArray`.
2. Maak twee arrays aan van het type string en noem deze ***filmTitels*** en ***jaartal***
3. Vul de arrays met de volgende waarden. (*Let op hieronder staat 1 tabel maar je maakt dus 2 arrays.*)

| filmtitels                        | jaartal |
|-----------------------------------|---------|
| Star Wars: Episode IV: A New Hope | 1977    |
| Aliens                            | 1986    |
| The Silence of the Lambs          | 1991    |
| Die Hard                          | 1988    |
| The Green Mile                    | 1999    |
| Avatar                            | 2009    |
| Avenger: Infinity War             | 2018    |

4. Zorg dat de lijst films en het jaartal wordt getoond in een textbox zoals in Afbeelding 60 - Filmlijst. Waarbij je gebruik maakt van een While loop.



Afbeelding 60 - Filmlijst

5. Voorzie je code van Comments waarin je uitlegt wat er gebeurd in je code.
6. Test of je applicatie werkt.
7. Als het werkt pas je jouw applicatie aan en maak je twee textboxen aan zoals in Afbeelding 61. Zorg er nu voor dat in de rechter textbox alleen de titels van films komen te staan met behulp van de ***While loop*** die je al hebt.
8. Zorg dat in de linker textbox de waarde van de index komen te staan. Maak hierbij gebruik van een ***For loop***.

| nr | Films                             |
|----|-----------------------------------|
| 0  | Star Wars: Episode IV: A New Hope |
| 1  | Aliens                            |
| 2  | The Silence of the Lambs          |
| 3  | Die Hard                          |
| 4  | The Green Mile                    |
| 5  | Avatar                            |
| 6  | Avenger: Infinity War             |

Afbeelding 61 - Filmlijst 2

9. Test of je aanpassingen werken.
10. Als alles werkt pas je de applicatie aan en voeg je nogmaals een textbox toe waarin je de jaartallen in een losse textbox plaatst zoals in Afbeelding 62. Gebruik voor het vullen van de textbox een **foreach** loop.

| nr | Films                             | Jaartal |
|----|-----------------------------------|---------|
| 0  | Star Wars: Episode IV: A New Hope | 1977    |
| 1  | Aliens                            | 1986    |
| 2  | The Silence of the Lambs          | 1991    |
| 3  | Die Hard                          | 1988    |
| 4  | The Green Mile                    | 1999    |
| 5  | Avatar                            | 2009    |
| 6  | Avenger: Infinity War             | 2018    |

Afbeelding 62 - Filmlijst 3

11. Test je applicatie en als alles werkt.

De applicatie is natuurlijk niet op de juiste manier opgebouwd. Je hebt gegevens die bij elkaar horen in losse arrays geplaatst waardoor er allerlei zaken mis kunnen gaan. Zo kun je bijvoorbeeld de films niet sorteren omdat anders de jaartallen en nummers niet meer kloppen. Dit ga je later oplossen door te werken met multi dimensionale arrays. Voor nu is het vooral belangrijk dat je hebt gezien hoe arrays werken en hoe je de gegevens op het scherm kunt krijgen in combinatie met een loop.

## 21.7 Opdracht - Multi Dimensional Array

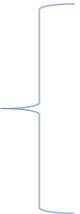
Een Multi dimensionale array is een array met daarin verschillende arrays. Dat klinkt best ingewikkeld maar het valt reuze mee. Voor deze oefening maak je gebruik van onderstaand tabel **films**.

| filmtitels                        | jaartal |
|-----------------------------------|---------|
| Star Wars: Episode IV: A New Hope | 1977    |
| Aliens                            | 1986    |
| The Silence of the Lambs          | 1991    |
| Die Hard                          | 1988    |
| The Green Mile                    | 1999    |
| Avatar                            | 2009    |
| Avenger: Infinity War             | 2018    |

In deze tabel **Films** zie je een kolom met de titel van de film en een kolom waarin het jaartal staat. Stel je zet al de gegevens (buiten de kolomkoppen uiteraard) in een array met de naam films dan heb je een array met daarin 7 arrays waarin de gegevens staan die bij de betreffende film horen, namelijk de titel en het jaartal waarin de film is uitgebracht. Iedere rij in deze tabel is dus eigenlijk een eendimensionale array en om deze reden wordt het een multi dimensionale array genoemd.

Array films

Array 1  
Array 2  
Array 3  
Array 4  
Array 5  
Array 6  
Array 7



|                                   |      |
|-----------------------------------|------|
| Star Wars: Episode IV: A New Hope | 1977 |
| Aliens                            | 1986 |
| The Silence of the Lambs          | 1991 |
| Die Hard                          | 1988 |
| The Green Mile                    | 1999 |
| Avatar                            | 2009 |
| Avenger: Infinity War             | 2018 |

In bovenstaande tabel zie je dat een twee dimensionale array bestaat uit meerdere arrays. Je kunt dit nog complexer maken door een drie dimensionale array te maken waarbij Array 1 t/m 7 ook weer bestaan uit arrays. Voor nu beperken we ons tot een twee dimensionale array zodat we de films kunnen koppelen aan het jaartal waarin de film is uitgebracht.

1. Maak een nieuwe Windows Forms applicatie aan en noem deze FilmMultiArray.
2. Voeg een textbox toe aan het formulier waarin de films getoond gaan worden.
3. Dubbelklik op je formulier om naar het frmload event te gaan.
4. Maak een twee dimensionale array films aan van het type string door gebruik te maken van onderstaande syntax. Zorg dat de films uit de lijst in de inleiding van dit hoofdstuk in deze array komen te staan.

```
string[,] films = { { <filmtitel1> , <jaartal> } , { <filmtitel2> , <jaaral2> } , enz. }
```

- Door het gebruik van de **komma** bij het declareren van de variabele geef je aan dat het om een multi dimensionale array gaat.
- De **gele** accolades geven aan wat het bereik is van de array films.
- De **blauwe** accolades geven het bereik aan van de array waarin de gegevens van de films staan. Deze krijgen geen eigen naam omdat ze onderdeel zijn van de array films.

5. De gegevens in je array zijn individueel op te halen door gebruik te maken van het adres op basis van de index. In onderstaande afbeelding kun je zien wat de 'adressen' zijn van de individuele elementen binnen de array films.

| Name       | Value                               |
|------------|-------------------------------------|
| filmtitels | [string[7, 2]]                      |
| [0, 0]     | "Star Wars: Episode IV: A New Hope" |
| [0, 1]     | "1977"                              |
| [1, 0]     | "Aliens"                            |
| [1, 1]     | "1986"                              |
| [2, 0]     | "The Silence of the Lambs"          |
| [2, 1]     | "1991"                              |
| [3, 0]     | "Die Hard"                          |
| [3, 1]     | "1988"                              |
| [4, 0]     | "The Green Mile"                    |
| [4, 1]     | "1999"                              |
| [5, 0]     | "Avatar"                            |
| [5, 1]     | "2009"                              |
| [6, 0]     | "Avenger: Infinity War"             |
| [6, 1]     | "2018"                              |

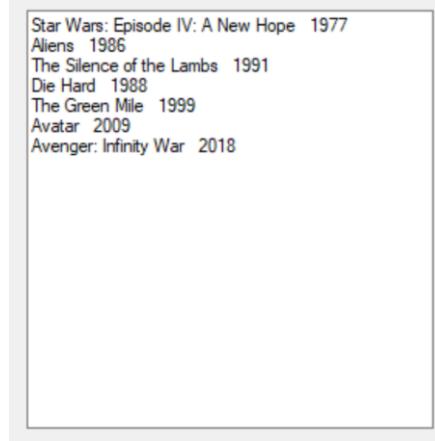
Afbeelding 63 - Elementen in de array films

- Haal je nu de lengte van de array op om te gebruiken in een loop dan zul je zien dat de lengte 14 is.

| Locals      |                                     |
|-------------|-------------------------------------|
| Name        | Value                               |
| this        | {FilmsMultiArray.frmMultiArray, Te} |
| sender      | {FilmsMultiArray.frmMultiArray, Te} |
| e           | {System.EventArgs}                  |
| filmtitels  | {string[7, 2]}                      |
| lengteArray | 14                                  |
| row         | 0                                   |

Afbeelding 64 - Lengte van array

- Dit is dus niet het aantal rijen uit de tabel omdat in een array alle elementen onder elkaar worden gezet. De index [ 0 , 1 ] geeft aan om welk veld het gaat waarbij het eerste getal het rijnummer is en het tweede getal het kolomnummer.
- Wil je nu met een loop de inhoud van de array laten zien in een textbox dan heb je het aantal rijen nodig waaruit de tabel bestaat. Je kunt dit aantal niet ophalen met de Length() methode zoals je hiervoor hebt gezien want dan krijg je teveel rijen terug. Met de methode getLength() kun je dit wel ophalen want je kunt dan tussen de haakjes het nummer van de kolom opgeven en wordt het aantal items in deze kolom geteld wat resulteert in het correct aantal rijen. De code **filmtitels.getLength(0)** telt dus eigenlijk het aantal filmtitels in de tabel.
- Maak nu de applicatie zelf af waarmee je de textbox vult met alle gegevens uit de array en test je applicatie. Zorg ervoor dat de gegevens in de textbox staan zoals in onderstaande afbeelding.



Afbeelding 65 - Films in textbox

10. Voorzie je code van Comments waarin je uitlegt wat er gebeurd in je code.

Lukt het niet om deze opdracht te maken kijk dan in Bijlage 2 op pagina 144 voor een complete uitleg en uitwerking.

## 21.8 Opdracht - Control DataGridView

Het tonen van de data uit de Array in een textbox is niet echt de beste manier. De items zijn niet mooi uitgelijnd en je hebt ook geen kolomkoppen. Een betere manier om de data in je applicatie te tonen is door gebruik te maken van de control **DataGridView**. Dit is een control die specifiek is bedoeld voor het tonen van data in de vorm van rijen en kolommen. In C# kun je deze DataGridView koppelen aan een gegevensbron zoals een database of bijvoorbeeld een array zoals je in onderstaande afbeelding kunt zien.

| Titel                             | Jaar |
|-----------------------------------|------|
| Star Wars: Episode IV: A New Hope | 1977 |
| Aliens                            | 1986 |
| The Silence of the Lambs          | 1991 |
| Die Hard                          | 1988 |
| The Green Mile                    | 1999 |
| Avatar                            | 2009 |
| Avenger: Infinity War             | 2018 |

Afbeelding 66 - Films Array in DataGridView

1. Maak een nieuwe Windows Forms Applicatie aan met als naam **FilmApplicatie**.
2. Voeg de control **DataGridView** toe aan het formulier met als naam **dgvFilms**.
3. Maak de array **films** en vul deze met de gegevens uit onderstaande tabel.

| filmtitels                        | jaartal | Genre    |
|-----------------------------------|---------|----------|
| Star Wars: Episode IV: A New Hope | 1977    | SciFi    |
| Aliens                            | 1986    | Horror   |
| The Silence of the Lambs          | 1991    | Thriller |
| Die Hard                          | 1988    | Actie    |
| The Green Mile                    | 1999    | Drama    |
| Avatar                            | 2009    | SciFi    |
| Avenger: Infinity War             | 2018    | Actie    |

4. Voor je in de code kunt werken met de DataGridView dien je eerst op te geven uit hoeveel kolommen deze bestaat. Je kunt de waarde direct opgeven want je weet dat er drie kolommen worden gebruikt maar het is netter om de code dit zelf te laten bepalen. Met de methode **GetLength()** kun je eenvoudig het aantal kolommen en rijen ophalen. Voeg onderstaande code toe zodat je twee variabelen hebt waarin het aantal kolommen en het aantal rijen is opgeslagen. Het aantal rijen ga je straks ook nog gebruiken voor het maken van de loop.

```
int aantalRijen = films.GetLength(0);
int aantalKolommen = films.GetLength(1);
```

5. Met de property **ColumnCount** van de dgvFilms geef je op uit hoeveel kolommen de DataGridView moet bestaan. Neem onderstaande code over.

```
dgvFilms.ColumnCount = aantalKolommen;
```

6. Nu het aantal kolommen is opgegeven kun je met een Loop de gegevens uit je array gaan plaatsen in de DataGridView. Bij het vullen wordt er altijd eerst een rij aangemaakt en daarna wordt de rij gevuld met gegevens (de kolommen) en daarna gaat de code naar de volgende rij.
7. Om dit voor elkaar te krijgen maak je gebruik van een For Loop aan waarin de rijen worden aangemaakt. Gebruik hiervoor onderstaande code.

```
for (int r = 0; r < aantalRijen; r++) //Toevoegen van rijen.
{
}
```

8. Iedere iteratie moet er een nieuwe rij met gegevens uit je array worden toegevoegd aan de **DataGridView**. De gegevens worden eerst opgeslagen in een variabele van het type **DataGridViewRow**. Neem onderstaande code over en kijk naar de comments wat de code doet.

```
//Maak een variabele dataRij aan waarin een DataGridViewRow wordt opgeslagen
DataGridViewRow dataRij = new DataGridViewRow();

//Maak lege Velden(kolommen) aan in dataRij op basis van de inhoud van de array
dataRij.CreateCells(dgvFilms);
```

9. Iedere iteratie moet de variabele dataRij gevuld worden met gegevens uit de Array. Hiervoor ga je gebruik maken van een geneste For Loop. Deze Loop moet uiteraard uitgevoerd worden NADAT de variabele dataRij is aangemaakt en voorzien is van velden. Je plaatst deze Loop dan ook onder deze code zoals je hieronder kunt zien. Neem deze code over.

```
for (int r = 0; r < aantalRijken; r++) //Voor het toevoegen van rijen.
{
 DataGridViewRow dataRij = new DataGridViewRow();
 dataRij.CreateCells(dgvFilms);

 for (int k = 0; k < aantalKolommen; k++)
 {

 }

}
```

10. De property **Cells[ ].Value** Tussen de rechte haken [ ] komt het kolomnummer te staan kun je nu gebruiken om de waarden uit de array op te slaan. De code komt er als volgt uit te zien. (voeg deze code toe aan de scope van de geneste loop)

```
dataRij.Cells[k].Value = films[r, k];
```

11. Hoe bovenstaande code werkt wordt zo meteen verder uitgelegd als heel de code wordt doorlopen.
12. Nadat de variabele **dataRij** gevuld is met waarden moeten deze waarden worden weggeschreven in de DataGridView. Hiervoor gebruik je de methode **dgvFilms.Rows.Add()**. Aangezien deze code pas mag worden uitgevoerd NADAT de variabele dataRij is gevuld dien je deze code ONDER de geneste For Loop te plaatsen maar uiteraard wel in de eerste Loop. Neem de onderstaande **gemarkeerde** code over.

```
for (int r = 0; r < aantalRijken; r++) //Voor het toevoegen van rijen.
{
 DataGridViewRow dataRij = new DataGridViewRow();
 rij.CreateCells(dgvFilms);

 for (int k = 0; k < aantalKolommen; k++)
 {

 dataRij.Cells[k].Value = films[r, k];

 }

 dgvFilms.Rows.Add(dataRij);
}
```

13. Test nu je applicatie en kijk naar het resultaat. Als je alles goed hebt overgenomen is je resultaat gelijk aan Afbeelding 67.



|   |                       |      |
|---|-----------------------|------|
| ► | Star Wars: Episo ...  | 1977 |
|   | Aliens                | 1986 |
|   | The Silence of th...  | 1991 |
|   | Die Hard              | 1988 |
|   | The Green Mile        | 1999 |
|   | Avatar                | 2009 |
|   | Avenger: Infinity ... | 2018 |

Afbeelding 67 - DataGridView

14. Je applicatie ziet er niet hetzelfde uit zoals aan het begin van de opdracht staat aangegeven. Dit komt omdat je nog een aantal properties van de DataGridView aan moet passen. Pas dit aan in je code dus **niet** met het properties venster in de design.

- a. De eerste property is het verwijderen van de eerste kolom met het driehoekje. De velden in deze kolom worden de **RowHeaders** genoemd. Zorg ervoor dat deze RowHeaders niet meer zichtbaar zijn en de kolom dus ook niet meer zichtbaar is.
- b. Er staan ook nog geen titels boven de kolommen. Maak gebruik van de property `Columns[ ].HeaderText` om titels toe te voegen aan de kolommen. (Tussen de [ ] komt het index getal van de kolom.)
- c. Met de property **AutoSizeColumnsMode** kun je bepalen hoe de kolommen binnen je DataGridView op het scherm worden getoond. De waarde **DataGridViewAutoSizeColumnsMode.Fill** zorgt ervoor dat de kolommen worden verdeeld over het DataGridView venster zoals je deze hebt getekend.

15. Test je applicatie. Als alles goed is gegaan ziet deze er net zo uit als het scherm in Afbeelding 66. Sla je applicatie op.

## 21.8.1 Werking van de loop in de code

Hieronder kun je zien hoe de loops in je applicatie nu werken. De Loop waarin de rij wordt aangemaakt en weggeschreven in de DataGridView noemen we Loop 1. De Geneste Loop waarin de cellen worden gevuld noemen we loop 2.

In Afbeelding 68 kun je de Array zien welke wordt gebruikt inclusief de index. Afbeelding 68 – Index van Array

| Index | 0                                 | 1       | 2        |
|-------|-----------------------------------|---------|----------|
|       | filmtitels                        | jaartal | Genre    |
| 0     | Star Wars: Episode IV: A New Hope | 1977    | SciFi    |
| 1     | Aliens                            | 1986    | Horror   |
| 2     | The Silence of the Lambs          | 1991    | Thriller |
| 3     | Die Hard                          | 1988    | Actie    |
| 4     | The Green Mile                    | 1999    | Drama    |
| 5     | Avatar                            | 2009    | SciFi    |
| 6     | Avenger: Infinity War             | 2018    | Actie    |

Afbeelding 68 – Index van Array

```
for (int r = 0; r < aantalRijen; r++) //Voor het toevoegen van rijen.
{
 DataGridViewRow dataRij = new DataGridViewRow();
 dataRij.CreateCells(dgvFilms); //Voor het toevoegen van cellen aan de rij.

 for (int k = 0; k < aantalKolommen; k++)
 {

 dataRij.Cells[k].Value = films[r, k]; //Vullen van cellen met gegevens uit de array.

 }

 dgvFilms.Rows.Add(dataRij); //Vullen van de rij aan de DataGridView.
}
```

### Eerste Iteratie (r = 0 en k = 0)

```
for (int r = 0; r < aantalRijen; r++) //Voor het toevoegen van rijen.
{
 DataGridViewRow dataRij = new DataGridViewRow();
 dataRij.CreateCells(dgvFilms); ←
 ← Loop 1: De variabele dataRij wordt aangemaakt met een lege rij en 3 cellen.

 for (int k = 0; k < aantalKolommen; k++)
 {
 ← Loop 2: Star Wars: Episode IV: A New Hope wordt in de eerste cel geplaatst.
 dataRij.Cells[0].Value = films[0, 0];
 }
 ← Loop 1: Er gebeurt niets.
 dgvFilms.Rows.Add(dataRij);
}
```

Loop 1: De variabele dataRij wordt aangemaakt met een lege rij en 3 cellen.

Loop 2: *Star Wars: Episode IV: A New Hope* wordt in de eerste cel geplaatst.

Loop 1: Er gebeurt niets.

### Tweede Iteratie (r = 0 en k = 1)

```
for (int r = 0; r < aantalRijen; r++) //Voor het toevoegen van rijen.
{
 DataGridViewRow dataRij = new DataGridViewRow();
 dataRij.CreateCells(dgvFilms); ←
 ← Loop 1: Deze code wordt niet uitgevoerd want je applicatie zit nog in loop 2.

 for (int k = 1; k < aantalKolommen; k++)
 {
 ← Loop 2: 1977 wordt in de tweede cel van de rij ingevoegd.
 dataRij.Cells[1].Value = films[0, 1];
 }
 ← Loop 1: Deze code wordt niet uitgevoerd want je applicatie zit nog in loop 2.
 dgvFilms.Rows.Add(dataRij);
}
```

Loop 1: Deze code wordt niet uitgevoerd want je applicatie zit nog in loop 2.

Loop 2: 1977 wordt in de tweede cel van de rij ingevoegd.

Loop 1: Deze code wordt niet uitgevoerd want je applicatie zit nog in loop 2.

### Derde Iteratie (r = 0 en k = 2)

```
for (int r = 0; r < aantalRijen; r++) //Voor het toevoegen van rijen.
{
 DataGridViewRow dataRij = new DataGridViewRow(); ← Loop 1: Deze code wordt niet uitgevoerd want je applicatie zit nog in loop 2.
 DataRij.CreateCells(dgvFilms);

 for (int k = 2; k < aantalKolommen; k++) ← Loop 2: SciFi wordt in de derde cel van de rij ingevoegd.
 {
 dataRij.Cells[2].Value = films[0, 2]; ←
 }
 dgvFilms.Rows.Add(dataRij); ← Loop 1: Deze code wordt niet uitgevoerd want je applicatie zit nog in loop 2.
}
```

### Vierde Iteratie (r = 0 en k = 3)

```
for (int r = 0; r < aantalRijen; r++) //Voor het toevoegen van rijen.
{
 DataGridViewRow dataRij = new DataGridViewRow(); ← Loop 1: Deze code wordt niet uitgevoerd want je applicatie zit nog in loop 2.
 DataRij.CreateCells(dgvFilms);

 for (int k = 3; k < aantalKolommen; k++) ← Loop 2: Deze code wordt niet uitgevoerd want k < aantalKolommen is niet meer waar. Je applicatie gaat uit Loop 2.
 {
 dataRij.Cells[2].Value = films[0, 2]; ←
 }
 dgvFilms.Rows.Add(dataRij); ← Loop 1: De rij wordt toegevoegd aan de DataGridView.
}
```

### Vijfde Iteratie (r = 1 en k = 0)

```
for (int r = 1; r < aantalRijen; r++) //Voor het toevoegen van rijen.
{
 DataGridViewRow dataRij = new DataGridViewRow(); ← Loop 1: De variabele dataRij wordt opnieuw aangemaakt en is weer leeg.
 DataRij.CreateCells(dgvFilms);

 for (int k = 0; k < aantalKolommen; k++) ← Loop 2: Aliens wordt in de eerste cel geplaatst.
 {
 dataRij.Cells[0].Value = films[1, 0]; ←
 }
 dgvFilms.Rows.Add(dataRij); ← Loop 1: Deze code wordt niet uitgevoerd want je applicatie zit nog in loop 2.
}
```

## 21.9 Eindopdracht – Dessert Applicatie

In DeveloperLand zijn verschillende plekken waar je een hapje kunt eten. Ze hebben uiteraard geen standaard menukaart maar een applicatie waarin de menu's staan. Aan jou de taak om een applicatie te maken waarin de desserts te zien zijn maar waar ook nieuwe desserts in kunnen worden opgeslagen.

| Naam       | IJssoort  | Ingrediënt_1 | Ingrediënt_2   |
|------------|-----------|--------------|----------------|
| CodeMonkey | Vanille   | Banaan       | Chocoladesaus  |
| Perl       | Aardbeien | Aardbei      | Aardbeiensaus  |
| Noob       | Vanille   | TumTum       | Sprinkles      |
| Java       | Chocolade | KoffieSaus   | Koffieboontjes |
|            |           |              |                |

1. Maak een nieuwe applicatie met als naam DessertsApp.
2. Zorg dat je met een DataGridView de bovenstaande desserts kunt zien.
3. Zorg met invoervelden en een knop dat je nieuwe desserts kunt toevoegen.
4. Zorg ervoor dat een dessert ook weer van de kaart verwijderd kan worden.
5. Maak een screenshot van de code van je applicatie en lever deze in op It's Learning!

## 21.10 Wat heb je geleerd?

- Je weet het verschil tussen een Array en multidimensionale Array.
- Je weet hoe een Array is opgebouwd en kent de term index.
- Je kunt gegevens ophalen uit een Array met behulp van een Loop.
- Je kunt gegevens toevoegen of verwijderen uit een Array.
- Je kent de ForEach loop.
- Je kunt werken met de DataGridView control.

## **22 Extra Opdrachten**

---

### **22.1 Hoofdstuk 10 - Met je gezin naar Developer Land in december.**

In december wil de directeur van Developer Land een wat feestelijker uitstraling van zijn applicatie waarmee mensen begroet worden. In deze periode komen de bezoekers ook meestal met hun gezin.

- Maak een console applicatie waarbij je gebruik maakt van kerst kleuren zoals groen, rood en bruin/goud.
- Aangezien de bezoekers met hun gezin komen moeten er drie namen ingegeven kunnen worden.
- De gezinnen bestaan uit twee ouders en één kind.
- Het gezin geeft eerst de achternaam in.
- Daarna geven de gezinsleden ieder hun eigen naam in.
- Iedere persoon krijgt zijn eigen vraag (op een nieuwe regel) en kan daarna de naam invullen.
- Er bestaan natuurlijk ook gezinnen met twee vaders of twee moeders. Houd hierbij rekening in je vraagstelling.
- Na het invullen van alle namen wordt in de eerste regel het gezin begroet met "Welkom familie ...."
- Ieder gezinslid krijgt daarna zijn eigen begroeting:
  - "Hallo ..... , welkom in Developer Land. Ik hoop dat het een leuke dag wordt."
  - "Welkom ..... , gezellig dat je mee bent gekomen naar Developer Land."
  - "Super ..... dat je mee bent gekomen. Er zijn veel leuke dingen te doen voor kinderen."
- Houd rekening met de juiste naamgevingsconventies.

### **22.2 Hoofdstuk 11 – Werken met operatoren**

#### **22.2.1 Applicatie – Temperatuur omrekenen**

Maak een applicatie waarbij de gebruiker de temperatuur van Celsius naar Fahrenheit kan omzetten. De gebruiker krijgt de vraag om de waarde (temperatuur in Celsius) in te vullen en krijgt daarna op de volgende regel de melding "De temperatuur in Fahrenheit is:

De formule om dit te berekenen is:  $\text{Graden Fahrenheit} = (9 / 5) * \text{Graden Celsius} + 32$

#### **22.2.2 Applicatie – Omrekenen TV grootte**

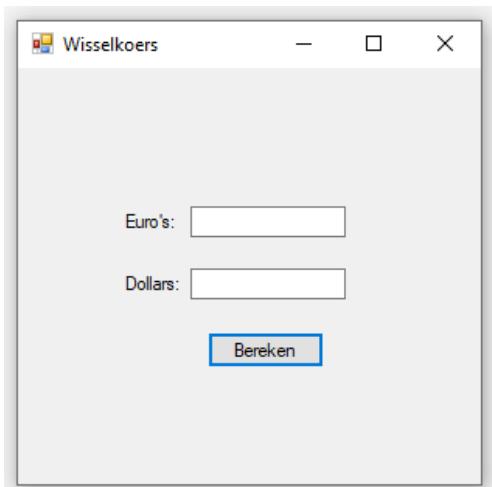
Bij het kopen van een tv staat het formaat altijd in aantal inch. Maak een console applicatie waarbij de gebruiker de grootte in inch kan ingeven en de applicatie de waarden in centimeters teruggeeft zodat je een beter beeld hebt van de grootte van de tv. De gebruiker krijgt de vraag om de waarde (in Inch) en krijgt daarna op de volgende regel de melding ...inch = ....cm waarbij op de puntjes de ingevulde en uitgerekende waarden staan. (voor het omrekenen kun je aanhouden: 1 inch = 2,54cm)

#### **22.2.3 Applicatie – Entree prijs berekenen**

Voor de balie van Developer Land is een systeem nodig om eenvoudig de entreeprijs voor een gezin te berekenen. Volwassenen betalen 15 euro en kinderen 7,50. Het is de bedoeling dat alleen het aantal volwassenen en kinderen wordt opgegeven waarna de totale entreeprijs te zien is welke de gasten af moeten rekenen. Voorzie de console applicatie van nette teksten en pas de kleuren aan naar een witte achtergrond en zwarte letters.

## 22.3 Hoofdstuk 12 – Controls, Events en foutafhandeling

### 22.3.1 Wisselkoers



Afbeelding 69 - Wisselkoers

Maak een Windows Forms applicatie waarmee euro's kunnen worden omgezet naar dollars.

Het formulier bestaat uit:

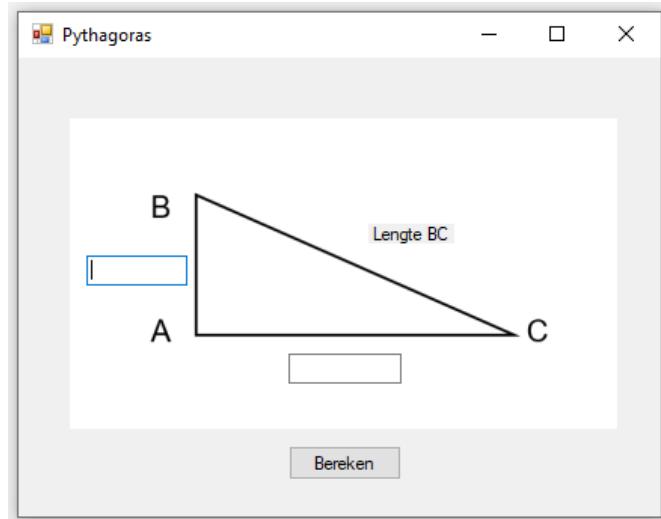
- Een textbox waarin het bedrag kan worden getypt in euro's.
- Een textbox waarin het bedrag in dollars wordt getoond.
- Een knop waarmee de berekening wordt gestart.
- Labels om aan te geven welke getallen in de textboxen staan
- Er wordt gebruik gemaakt van foutafhandeling zodat de gebruiker een foutmelding krijgt als er iets niet goed gaat.

Zoek zelf op wat de actuele wisselkoers is om te gebruiken in de berekening.

### 22.3.2 Pythagoras

Maak een applicatie zoals in onderstaand afbeelding waarin je gebruik maakt van een PictureBox, twee textboxen en een label waarin de uitkomst komt te staan (lengte BC).

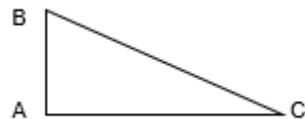
De lengte van BC wordt getoond in het label als de gebruiker de lengte van AB en AC in de textbox invult en op de knop berekenen klikt. Hiervoor gebruik je de stelling van Pythagoras.



Afbeelding 70 - Pythagoras Applicatie

Met de stelling van Pythagoras kun je de lengte van de schuine zijde van een rechthoekige driehoek berekenen. De berekening ziet er als volgt uit.

$$AB^2 + AC^2 = BC^2$$



$$AB = 3$$

$$AC = 4$$

$$BC = ?$$

$$BC^2 = (AB * AB) + (AC * AC)$$

$$BC^2 = (3 * 3) + (4 * 4)$$

$$BC^2 = 25$$

$$BC = \sqrt{25}$$

$$BC = 5$$

In C# heb je een aantal methodes tot je beschikking waarmee je eenvoudig de machten en wortels van getallen kunt uitrekenen.

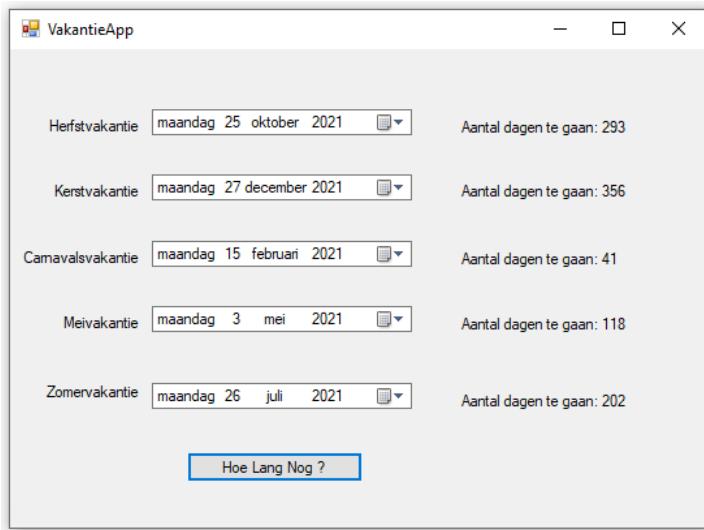
Machten:      `Math.Pow( <getal> , <macht> )`

Wortel:        `Math.Sqrt( <getal> )`

## 22.4 Hoofdstuk 14 – DateTime Variabele

### 22.4.1 Vakantie App

Maak een applicatie die toont hoe lang het nog duurt voor het vakantie is. Zorg ervoor dat de datums van de start van de vakanties ingevuld kunnen worden en dat er per vakantie te zien is hoe lang het nog duurt. Zorg ervoor dat alle vakanties in de app aanwezig zijn.

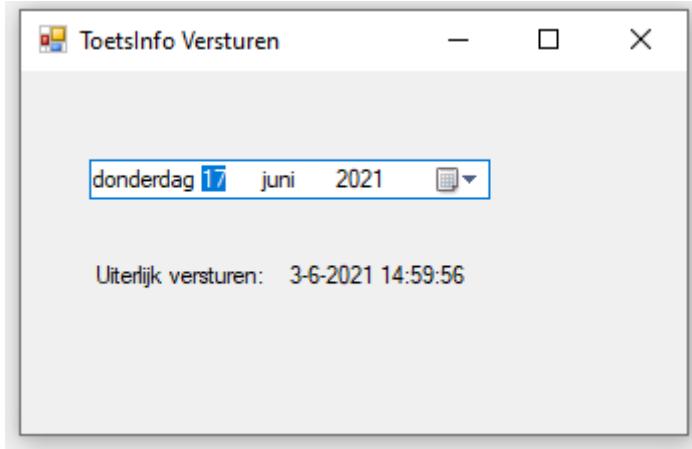


Afbeelding 71 - Vakantie App

### 22.4.2 ToetsInfo Applicatie

Het inplannen van alle proeven van bekwaamheid is altijd een lastige klus. Er kunnen maar een maximaal aantal studenten deelnemen en iedere student moet minimaal 2 weken voorafgaande aan de toets alle nodige informatie ontvangen.

Maak een eenvoudige applicatie waarin een datum kan worden ingegeven waarop een student staat ingepland voor een toets. Zodra de datum door de gebruiker wordt gewijzigd, krijgt de gebruiker een datum te zien waarop de student uiterlijk (dus twee weken ervoor) de informatie moet ontvangen.



Afbeelding 72 - ToetsInfo Versturen

## 22.5 Hoofdstuk 15 – Selecties

### 22.5.1 DatatypeApp

Een beginnende developer heeft moeite met het bepalen wat voor datatypen moeten worden gebruikt bij het declareren van variabelen. Maak een Windows Forms Applicatie waarbij de gebruiker kan aangeven met wat voor waarden de applicatie gaat werken waarna met een klik op de knop te zien is welke datatypen gebruikt kunnen worden met daarbij het bereik van deze datatypen zodat hij op basis daarvan een keuze kan maken. De gebruiker kan onderstaande tekst invullen.

- Gebroken getallen
- Gehele getallen
- Negatieve getallen
- Tekst
- Geld
- ja / nee

Na het invullen van de tekst klikt de gebruiker op een knop en krijgt in een label te zien wat het datatype moet worden.

The screenshot shows a Windows application window titled "DatatypeApp". Inside, there is a table titled "Bereik" (Range) with the following data:

| Datatype          | Minimum | Maximum                                       |
|-------------------|---------|-----------------------------------------------|
| Gehele Getallen   | Float   | $-3,4 \times 10^{-45}$                        |
| Gebroken Getallen | Double  | $3,4 \times 10^{38}$<br>$-5 \times 10^{-324}$ |
| Financieel        |         |                                               |
| Tekst             |         |                                               |
| Ja / Nee          |         |                                               |

At the bottom of the window is a button labeled "Welke Datatype?".

## 22.6 Hoofdstuk 16, 17 – Geneste Selecties

### 22.6.1 Applicatie – ToetsScore app

Developer Land heeft voor sollicitanten een toets ontwikkeld welke ze moeten halen om in aanmerking te komen voor een baan in het park. In deze toets krijgen ze verschillende opdrachten waarvoor ze een punt halen tussen 1 en 10. De test bestaat uit de volgende onderdelen

Python  
C#  
Laravel  
Unity  
JavaScript  
PHP  
MicroBit

**De sollicitant heeft de test behaald als aan de onderstaande eisen is voldaan:**

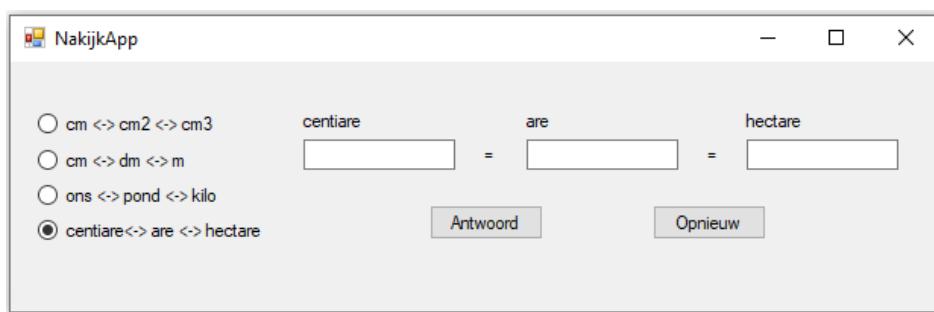
- Het totaal van alle onderdelen moet op minstens 40 punten uitslaan.
- Voor C# moet minimaal een 7 worden behaald.
- Er mag maximaal 1 onvoldoende worden behaald.

Maak een applicatie waarbij de behaalde punten kunnen worden ingegeven en er op een knop kan worden geklikt zodat de gebruiker een melding krijgt of de sollicitant geslaagd of gezakt is voor de test.

### 22.6.2 Applicatie – NakijkApp

Veel kinderen op de basisschool hebben moeite met bijvoorbeeld het omrekenen van cm naar  $\text{cm}^2$  en  $\text{cm}^3$ . Ze krijgen veel oefening hierin maar het is natuurlijk handig als ze dit zelf na kunnen kijken.

Maak een Windows Forms Applicatie waarbij je gebruik maakt van labels, textboxen, buttons en Radiobuttons.



- Met de radiobutton kunnen ze aangeven om welke omrekening het gaat.
- De labels boven de textboxen veranderen als een andere berekening wordt geselecteerd met de radiobuttons. (hier kun je een event voor gebruiken)
- De leerling kan in een willekeurige textbox een waarde invullen en op de knop antwoord klikken. In de twee andere textboxen verschijnen dan de juiste waarden welke overeenkomen met het label boven de textbox.

### 22.6.3 DatatypeApp 2.0 – (pittige opdracht)

Het nut van de DataTypeApp uit de vorige opgave is natuurlijk niet echt nuttig. Een tabel laat exact hetzelfde zien. Maak nu bovenstaande applicatie maar nu met een invoerveld waarin de gebruiker een getal kan typen. Op basis van dit getal wordt aangegeven welk datatype gebruikt kan worden.

De applicatie moet rekening houden met:

- Positieve en negatieve getallen
- Gehele of gebroken getallen
- tekst

Aangezien je niet met het getal 0 kunt bepalen om wat voor datatype het gaat dien de gebruiker een melding te krijgen dat het getal niet 0 mag zijn.

#### Hulp

Met onderstaande code kun je controleren of een tekst een string is of niet. De uitkomst (True / False) wordt opgeslagen in de variabele **uitkomst**. Kan de waarde worden omgezet, dan wordt tevens de omgezette waarde in de variabele **result** opgeslagen.

```
bool uitkomst = double.TryParse(waarde, out result);
```

Om te bepalen of een getal een komma getal is kun je gebruik maken van **modulo**. Met module bepaal je de restwaarde nadat je hebt bepaald hoe vaak een getal in een ander getal past (deling).

Voorbeeld: 12 gedeeld 5 is 2,4; maar 12 **modulo** 5 is 2, doordat 5 tweemaal in 12 past en dan houd je nog 2 over. Wiskundelocenten die vinden dat natuurlijk helemaal fantastisch om uit te rekenen maar wij gaan dit alleen ‘misbruiken’ door te kijken hoe vaak het getal 1 in een getal past. Bij een gebroken getal houdt je altijd iets over en bij een geheel getal niet.

$$\begin{aligned}5 \% 1 &= 0 \\5,5 \% 1 &= 0,5 \\6,1 \% 1 &= 0,1\end{aligned}$$

De modulo wordt, zoals je hierboven kunt zien, in C# aangeduid met het %- teken. Om nu te bepalen of een getal een kommagetal of gebroken getal is kun je in je code eenvoudig het volgende doen.

```
<variabele> % 1 == 0 (is het een geheel getal?)
of
<variabele> % 1 != 0 (is het een gebroken getal?)
```

In dit geval heb je een vergelijking en die kun je gebruiken in je IF statement.  
De code die je kunt gebruiken ziet er dan als volgt uit.

```
if (result % 1 != 0)
{
 //code welke wordt uitgevoerd als het om een gebroken getal gaat.
}
else
{
 // code welke wordt uitgevoerd als het om een geheel getal gaat
}
```

#### 22.6.4 Nieuwe Actiedagen bij Kubus.Com

Gill Bates van Kubus.Com was niet tevreden met de uitkomst van de actiedagen. Er is te weinig geld verdiend en ze heeft een nieuwe actie bedacht. Ze heeft gemerkt dat er vooral oudere mensen bij haar komen winkelen en hoopt met een nieuwe actie ook het jongere publiek aan te trekken. Haar plan is om jongere mensen extra korting te geven op basis van hun leeftijd.

Pas je applicatie Kortingsberekenen uit de opdracht in paragraaf 15.4 aan volgens onderstaande criteria.

- De huidige kortingen blijven bestaan.
- Personen jonger dan 25 jaar krijgen 7,5% extra korting.
- Personen jonger dan 30 jaar krijgen 5% extra korting.
- Personen jonger dan 40 jaar krijgen 3% extra korting.

Je code komt er misschien niet netjes uit te zien omdat je veel dubbele stukken code hebt staan. Dit is niet erg want je leert later hoe je de code netjes kunt maken.

#### 22.6.5 Selecties Switch - Palatia Village Cocktails

Maak een console applicatie waarbij je gebruik maakt van de Switch. Het is de bedoeling dat de gebruiker de lijst met namen van de cocktails krijgt te zien met daarvoor een nummer. Door het ingeven van het nummer krijgen ze de ingrediënten te zien zodat nieuwe medewerkers de cocktails ook kunnen maken.

**Tequila Sunrise** (45 ml Tequila, 90 ml sinaasappelsap, Grenadine)

**Sex on the Beach** (30ml Wodka, 30ml Perzik Likeur, 90ml Sinaasappelsap, 30ml cranberry sap)

**Margarita** (40ml Tequila, 25ml Triple sec, 75ml Limoensap, schijfje limoen)

**Daiquiri** (45ml Rum, Sap van 1 limoen, 10ml suikersiroop)

**Cosmopolitan** (45ml Wodka, 30ml Triple Sec, 30ml Cranberry sap, Sap van halve limoen)

**Blue Lagoon** (30ml Wodka, 90ml Sprite, 15ml Blue Curacao)

## 22.7 Hoofdstuk 19 – Loops

### 22.7.1 Getallen raden

Maak een Windows Forms applicatie waarbij een gebruiker een getal moet raden.

- Tussen welke waarden er geraden moet worden moet opgegeven kunnen worden.
- De reeds geraden getallen moeten in een lijst zichtbaar zijn.
- De gebruiker krijgt te zien of het goed of fout is. Hoe je dit doet mag je zelf bepalen.
- Er is te zien, naast de lijst met reeds geraden getallen, hoeveel pogingen er zijn gedaan door de gebruiker.

Extra: De gebruiker kan een vinkje aanzetten zodat hij bij de melding dat het getal niet goed is ook de melding krijgt of het te raden getal hoger of lager is.

Extra: De gebruiker krijgt een maximaal aantal pogingen. Het maximaal aantal pogingen is vooraf in te geven door de gebruiker.

## 22.8 Hoofdstuk 21 – DataGridView en Arrays

### 22.8.1 Gemiddeld gebruik van een auto

Maak een applicatie in Windows Forms waarin een gebruiker zijn kilometerstand kan bijhouden en het aantal liter dat hij heeft getankt. In een DataGridView wordt getoond

1. De dag waarop getankt is.,
2. Het aantal kilometers.
3. Het aantal liters.
4. Het gemiddelde verbruik berekend op de dag van invoeren.

Zodra de gebruiker op de knop klikt wordt het gemiddelde verbruik getoond in een MessageBox. Dit gemiddeld verbruik over alle dagen. Dus hier worden alle ingevoerde kilometers en liters bij elkaar opgeteld en wordt het verbruik berekend.

### 22.8.2 Puppyhulp

Mensen die een pup in huis halen hebben hun handen vol aan de opvoeding van de pup. Het aan en afleren van bepaalde gedragingen is een vast onderdeel in de tijd dat de pup wakker is. Naast deze gedragingen is het heel belangrijk dat een pup op het juiste moment en de juiste hoeveelheid voer krijgt. Dit is belangrijk omdat een pup enorm snel groeit en de juiste hoeveelheid voedingsstoffen nog zijn voor deze groei. Teveel voeding kan niet alleen leiden tot een te dikke pup maar ook vergroeiingen omdat een pup te veel bouwstoffen binnen krijgt. Fabrikanten zetten dan ook altijd een voedingsschema op de verpakking zodat de hond de juiste hoeveelheid voeding binnen krijgt.

In onderstaande afbeelding staat een voedingsschema van het merk Royal Canin voor pups die als ze volwassen zijn tussen de 11 en 25 kilo gaan wegen. Je dient dus altijd de hoeveelheid te geven bij het gewicht wat de volwassen hond gaat wegen. Dit volwassen gewicht wordt altijd aangegeven door de fokker waar je de pup gaat halen.

| ADULT WEIGHT<br>grammes               | AGE month, mois, Monat, mese |                   |     |                    |     |       |                                 |        |                                                                                                                                                                                                                                            |       |
|---------------------------------------|------------------------------|-------------------|-----|--------------------|-----|-------|---------------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|                                       | 2 m                          | 3 m               | 4 m | 5 m                | 6 m | 8 m   | 10 m                            | 12 m   | 13 m                                                                                                                                                                                                                                       |       |
| 11 kg                                 | (g)                          | cup               | (g) | cup                | (g) | cup   | (g)                             | cup    | (g)                                                                                                                                                                                                                                        | cup   |
| 14 kg                                 | 155                          | 1+4/8             | 182 | 1+6/8              | 194 | 1+7/8 | 197                             | 2      | 197                                                                                                                                                                                                                                        | 2     |
| 18 kg                                 | 184                          | 1+7/8             | 216 | 2+1/8              | 232 | 2+2/8 | 236                             | 2+3/8  | 204                                                                                                                                                                                                                                        | 2     |
| 22 kg                                 | 217                          | 2+1/8             | 259 | 2+5/8              | 279 | 2+6/8 | 285                             | 2+7/8  | 247                                                                                                                                                                                                                                        | 2+4/8 |
| 25 kg                                 | 237                          | 2+3/8             | 288 | 2+7/8              | 311 | 3+1/8 | 324                             | 3+2/8  | 332                                                                                                                                                                                                                                        | 3+2/8 |
|                                       | 244                          | 2+3/8             | 300 | 3                  | 325 | 3+2/8 | 347                             | 3+4/8  | 366                                                                                                                                                                                                                                        | 3+5/8 |
| Metabolisable energy:<br>4109 kcal/kg | 2                            | ► 5 months / mois | 6   | ► 12 months / mois |     |       | Water<br>Eau<br>Wasser<br>Acqua | L.I.P. | • Protein selected for its very high digestibility<br>• Protéines sélectionnées pour leur très haute digestibilité<br>• Für Ihre sehr hohe Verdaulichkeit ausgewählte Proteine<br>• Proteine selezionate per la loro elevata digestibilità |       |
|                                       | cup = 240 ml = 100 g         |                   |     |                    |     |       |                                 |        |                                                                                                                                                                                                                                            |       |

In deze opdracht ga je uit van een pup die op de volwassen leeftijd **25 Kg** gaat wegen dus je gebruikt de onderste rij uit de tabel.

- In de tabel zie je veel getallen staan en de focus ligt op de kolommen waar (g) boven staan want dit is de hoeveelheid voeding (in gram) de pup per dag moet hebben.
- Op basis van de leeftijd van de pup wordt bepaald hoeveel eten het mag hebben. Een pup die bijvoorbeeld drie maanden is moet dus 300g eten krijgen en als deze 4 maanden is moet dit 325g zijn.
- Iedere week krijgt de pup wat extra eten wat je kunt uitrekenen door (globaal) de hoeveelheid die de pup mag eten af te trekken van de hoeveelheid die de pup mag krijgen in de volgende maand en dit te delen door 4 (voor het gemak gaan we uit van 4 weken in een maand)

Een pup van 3,6 maanden krijgt dan:

$$(\text{eten bij 3 maanden}) + ((\text{eten bij 4 maanden}) - (\text{eten bij 3 maanden})) * 0,6$$

$$\begin{array}{ccccccccc} 300 & & + & ( & 325 & ) - ( & 300 & ) & * 0,6 \\ 300 & & + & & & & 15 & & \end{array}$$

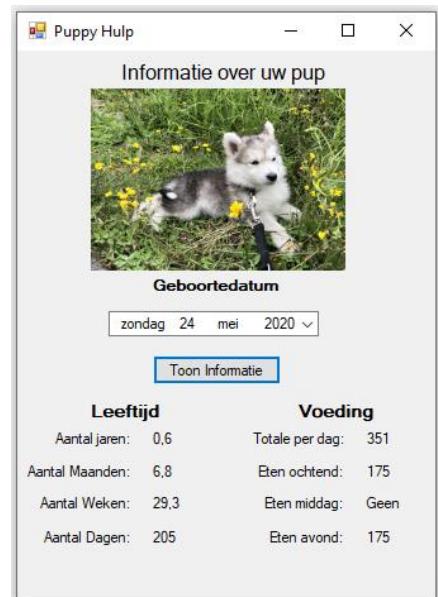
$$= 315 \text{ Gram}$$

- Naast de hoeveelheid dien je ook een pup te laten wennen aan het aantal keren eten per dag.
  - Een jonge pup moet drie keer per dag even dus wordt de hoeveelheid in de kolom verdeeld over 3 porties op een dag.
  - Een pup krijgt vanaf de 6<sup>e</sup> maand maar twee keer per dag eten dus wordt de hoeveelheid verdeeld over twee porties op een dag.

Zoals je kunt zien is dit een heel gedoe om telkens uit te rekenen hoeveel eten je jouw pup moet geven. De opdracht is dus om hier een applicatie voor te maken die nieuwe eigenaren kunnen gebruiken zodat ze nooit te veel of te weinig voer geven. De applicatie bouw je in eerste instantie voor een hond die op zijn volwassen leeftijd 25 gaat worden (dus de onderste regel in de voedingstabel).

## Maak een Windows Forms applicatie waarbij de volgende onderdelen aanwezig zijn:

- Een optie om de geboortedatum van de pup in te geven.
- De applicatie haalt automatisch de datum van vandaag op om zo de leeftijd te kunnen bepalen.
- Een overzicht van de actuele leeftijd van de pup in Jaren, Maanden, Weken en dagen. Dagen moet een geheel getal zijn de overige waarden maximaal één cijfer achter de komma zoals in de afbeelding hiernaast.
- Na het ingeven van de geboortedatum en het daarna klikken op de knop krijgt de gebruiker de informatie te zien.
- De hoeveelheid eten per maand wordt in een Array opgeslagen.
- Er wordt automatisch uitgerekend hoeveel eten de pup moet krijgen op de dag dat de applicatie bekeken wordt.
- Er is rekening gehouden met het aantal keren per dag dat de pup eten moet krijgen en de hoeveelheid wordt hierop aangepast.
- Er is een foto, tekening, logo etc. van een pup om de applicatie er wat leuker uit te laten zien.
- Als de pup ouder is dan 13 maanden moet er onderin een melding komen waarin staat aangegeven dat de pup geen puppyvoer meer moet krijgen maar voer voor volwassen honden.
- De afbeelding kun je gebruiken als inspiratie maar je hoeft dit niet exact na te maken als de functionaliteiten maar aanwezig zijn zoals ze hierboven beschreven staan.



## 23 Bijlage 1 – Naamgevingsconventies

---

- Gebruik zelfstandige naamwoorden of zinnen met een zelfstandig naamwoord voor classes.
- Gebruik **PascalCase** voor classes, properties en bestandnamen.  
(Bij PascalCase wordt ieder woord met een hoofdletter geschreven zoals **AutoSize**)
- Gebruik **camelCase** voor fields, variabelen en argumenten.  
(Bij camelCase begin je met een kleine letter en wordt verder ieder woord met een hoofdletter genoteerd)
- Gebruik zo min mogelijk afkortingen.
- Gebruik geen – (streepje of minteken) en liever ook geen \_ (underscore).
- Gebruik bij naamgeving van controls een voorloop.
  - Button – btn
  - TextBox – txb
  - Label – lbl
  - Form – frm

## 24 Bijlage 2 - Uitwerking - Filmlijst

---

Het is je niet direct gelukt om de opdracht zelf te maken. Dat is begrijpelijk want het is best wel complex om dit zomaar te bedenken. Hieronder leer je stap voor stap hoe je te werk moet gaan.

### Wat weten we?

- De gegevens in de array staan allemaal onder elkaar ook al zijn er rijen en kolommen.
- Elementen uit een array kun je benaderen met behulp van de index
- We moeten een loop gebruiken om de gegevens in de textbox te krijgen.
- Elementen uit de array films kun je met `textBox.Text = textBox.Text + films[ ]` in de textbox krijgen.
- Met “`\r\n`” kunnen we in een textbox naar een volgende regel gaan.

### Stap 1

Er moet een loop gebruikt worden dus we bepalen eerst wat voor loop we kunnen gebruiken. Aangezien het een multi dimensionale array is dienen we de lengte op te halen met de methode `getLength()` om het correcte aantal rijen (uit de tabel) te krijgen.

- **While** loop kunnen we gebruiken want daarin maken we gebruik van een vergelijking.
- **For** loop kunnen we gebruiken want ook hierin maken we gebruik van een teller.
- **Foreach** loop kunnen we NIET gebruiken want die houdt geen rekening met kolommen en pakt alle rijen.

We kunnen dus kiezen uit de **While** of **For** loop. Laten we een While loop gebruiken.

```
while (true)
{
}
```

### Stap 2

Om de while loop te laten werken hebben we nodig:

- Een variabele teller welke in iedere ronde wordt opgehoogd.
- Een variabele waarin de lengte van de array wordt opgeslagen.

```
int teller = 0;
int lengteArray = filmtitels.GetLength(0);

while (teller < lengteArray)
{
 teller++;
}
```

### Stap 3

Om de elementen in de array in de textbox op te slaan hebben we nodig:

- De inhoud van de eerste kolom (titel).

- De inhoud van de tweede kolom (jaartal).
- De index/adressen van de elementen in de array.
- Een aantal spaties om het jaartal een stukje achter de titel te plaatsen.
- “\r\n” om naar de volgende regel te gaan.

```

int teller = 0;
int lengteArray = films.GetLength(0);

while (teller < lengteArray)
{
 txbFilms.Text = txbFilms.Text + films[<rij>, <kolom>] + " ";
 txbFilms.Text = txbFilms.Text + films[<rij>, <kolom>] + "\r\n";
 teller++;
}

```

Je ziet in bovenstaande code dat er twee keer iets uit de array wordt opgehaald voordat er met een “\r\n” door wordt gegaan naar de volgende regel. In de volgende stap ga je kijken wat er moet gebeuren met <rij> en <kolom>.

## Stap 4

De code om iets in de textbox te krijgen is nu ver klaar maar je moet nog iets doen met die index getallen. We weten dat ieder element een unieke index heeft. In de lijst kun je zien dat het eerste getal (de rij) iedere ‘ronde’ wordt opgehoogd maar dat de kolom telkens 0 en 1 moet zijn.

| Name        | Value                               |  |
|-------------|-------------------------------------|--|
| (filmtitels | {string[7, 2]}                      |  |
| [0, 0]      | "Star Wars: Episode IV: A New Hope" |  |
| [0, 1]      | "1977"                              |  |
| [1, 0]      | "Aliens"                            |  |
| [1, 1]      | "1986"                              |  |
| [2, 0]      | "The Silence of the Lambs"          |  |
| [2, 1]      | "1991"                              |  |
| [3, 0]      | "Die Hard"                          |  |
| [3, 1]      | "1988"                              |  |
| [4, 0]      | "The Green Mile"                    |  |
| [4, 1]      | "1999"                              |  |
| [5, 0]      | "Avatar"                            |  |
| [5, 1]      | "2009"                              |  |
| [6, 0]      | "Avenger: Infinity War"             |  |
| [6, 1]      | "2018"                              |  |

Dat kun je oplossen door gebruik te maken van variabelen.

- De variabele teller kun je gebruiken voor het nummer van de rij.
- Voor het kolomnummer maak je een nieuwe variabele aan.
- Deze variabele moet 0 zijn als het eerste element wordt opgehaald.
- De variabele moet 1 zijn als het tweede element wordt opgehaald.

```
int teller = 0;
```

```

int lengteArray = films.GetLength(0);

while (teller < lengteArray)
{
 int kolom = 0;
 txbFilms.Text = txbFilms.Text + films[teller, kolom] + " ";

 kolom = 1;
 txbFilms.Text = txbFilms.Text + films[teller, kolom] + "\r\n";

 teller++;
}

```

In de code zie je dat de variabele **kolom** is gedeclareerd binnen de scope van de loop en niet bij de andere variabelen. Dit komt omdat de waarde van de variabele iedere ronde wisselt van 0 naar 1. Aan het begin van de volgende ronde moet de variabele weer op 0 worden gezet. Door de variabele aan het begin van de scope te declareren en de waarde op 0 te zetten zorg je er dus voor dat aan het begin van iedere ronde de waarde van **kolom** weer 0 is.

*(Je kunt natuurlijk ook de variabele wel declareren buiten de scope maar dan moet je aan het begin van de scope **kolom = 0;** opnemen om de waarde terug te zetten naar 0.)*

## Stap 5

De volledige code is nu klaar en je kunt je applicatie testen.

```

string[,] filmtitels = { { "Star Wars: Episode IV: A New Hope", "1977" },
{ "Aliens", "1986"}, { "The Silence of the Lambs", "1991" },
{ "Die Hard", "1988"}, { "The Green Mile", "1999" },
{ "Avatar", "2009"}, { "Avenger: Infinity War", "2018" } };

int teller = 0;
int lengteArray = filmtitels.GetLength(0);

while (teller < lengteArray)
{
 int kolom = 0;
 txbFilms.Text = txbFilms.Text + filmtitels[teller, kolom] + " ";

 kolom = 1;
 txbFilms.Text = txbFilms.Text + filmtitels[teller, kolom] + "\r\n";

 teller++;
}

```

## 25 Bijlage 2 - Index

---

### A

---

Array · 114  
Loops · 120  
Resize · 118  
Sorteren · 118  
Toevoegen Gegevens · 118

---

### B

---

Binair · 43  
Bits · 42  
bool · 44  
**Button** · 52  
byte · 44  
Bytes · 42

---

### C

---

char · 44  
Class · 30  
Controls · 52  
Conventies · 32  
Convert · 58  
    ToInt32() · 47  
**Convert.ToString** · 58

---

### D

---

**Data Sources** · 25  
DataGridView · 126  
Datatype · 44  
DateTime · 8, 75, 76, 80, 82, 99, 136  
decimal · 44  
double · 44

---

### E

---

Events · 57

---

### F

---

Financiële Getallen · 45  
float · 44  
For Loop · 121  
Foreach loop · 121

---

### G

---

Gebroken getallen · 45  
Gehele getallen · 45

---

### I

---

Integer · 44  
IntelliSense · 34, 35  
Interactie · 36, 39

---

### L

---

**Label** · 53  
long · 44  
Loop · 130

---

### M

---

MessageBox · 61  
methode · 28, 34, 35  
**Methode** · 31, 71  
    Aanmaken · 67  
    Aanroepen · 68  
**modulo** · 139  
Multidimensionale Array · 123

---

### O

---

Object · 30  
Operatoren · 48  
    Logisch · 48  
    Rekenkundig · 48  
    Relationeel · 48

---

### P

---

Parse · 46  
**Properties**  
    **Length** · 120  
Property · 30, 35  
    Length · 121

---

### R

---

**RadioButton** · 53

---

**S**

sbyte · 44  
**Server Explorer** · 25  
short · 44  
**Solution Explorer** · 24, 27  
string · 44

---

**T**

talstelsel · 43  
**TextBox** · 52  
**Toolbox** · 25

---

**U**

ushort · 44

---

**V**

Variabele  
Plaatsing · 37  
Reference type · 37  
Value Type · 37  
Variabelen · 36  
Visual Studio · 19  
Aanmaken Project · 21

---

**W**

While Loop · 120  
**WriteLine** · 28

c

icu