Zie Scherp

C# voor beginners, enthousiastelingen en geeks.
(zwartwit editie)

Tim Dams

Zie Scherp

C# voor beginners, enthousiastelingen en geeks.

(zwartwit editie)

Tim Dams

Voor m'n studenten.

Inhoudsopgave

Welkom

Zo, je hebt besloten om de edele taal C# te leren? Je bent hier aan het juiste adres. Dit boek wordt gebruikt als handboek binnen de opleidingen professionele bachelor elektronica-ict en toegepaste informatica van de AP Hogeschool. Het is gedurende vele jaren gegroeid tot een tweedelige reeks (1 per semester) waarvan je nu het eerste deel vast hebt. In dit deel ga je de basisbeginselen van C# en programmeren in het algemeen ontdekken.

Dit boek zal de fundering leggen en zaken behandelen zoals variabelen, loops methoden en arrays. Het tweede boek gaat hier mee verder en zal (hopelijk) de mystieke, maar oh zo belangrijke, wereld van het Object georiënteerd programmeren uit de doeken doen.

Je vraagt je misschien af hoe up-to-date dit boek is? Wel, het is uitgekomen in 2020, dus… (mmm, het jaar 2020 als kwaliteitslabel gebruiken is een beetje zoals zeggen dat je wijn maakt met rioolwater). Desalniettemin, het boek is gloednieuw en heeft voorlopig het jaar 2020 overleefd.

Net zoals onze spreektalen, evolueert ook de programmeertaal C# constant (terwijl ik dit schrijf zijn we aan versie 8.0). Bij iedere nieuwe C#-versie worden bepaalde code-constructies plots veel eenvoudiger of zelfs gewoon overbodig. Een goed programmeur moet natuurlijk zowel met de oude als de nieuwe constructies kunnen werken. Ik heb getracht een gezonde mix tussen oud en nieuw te zoeken, waarbij de nadruk ligt op bruikbaarheid in je verdere professionele carrière. In dit boek zal je dus geen stoere, state-of-the-art C# innovaties terugvinden die enkel in heel specifieke projecten bruikbaar zijn. Integendeel, ik hoop dat, als je door dit boekdeel en het volgende bent, je een zodanige basis hebt, dat je ook zonder problemen in andere 'zustertalen' durft te duiken (zoals Java, C en C++, maar ook zelfs verre familieleden zoals Python of Javascript).

Dit boek ambieert niet om de volledige C#-taal en alles dat daar rond hangt aan te leren, daarvoor is dit boek veel te kort. Het boek daarentegen is gericht op eender wie die interesse heeft in de wondere wereld van programmeren, maar mogelijk nog nooit één letter code effectief heeft geprogrammeerd. Bepaalde concepten die ik te gecompliceerd acht voor een beginnende programmeur werden dan ook uit deze cursus gelaten. Beschouw dit boek dus maar als een gatewaydrug naar meer C#, meer programmeertalen en vooral meer (programmeer)plezier! U weze gewaarschuwd.

Veel lees-en programmeerplezier,

Tim Dams
Zomer 2020

Over de bronnen

Dit boek is het resultaat van bijna een decennia C# doceren aan de AP Hogeschool (eerst nog Hogeschool Antwerpen, dan Artesis Hogeschool, dan Artesis Plantijn Hogeschool…). De eerste schrijfsels verschenen op een eigen gehoste blog ("Code van 1001 Nacht", die ondertussen ter ziele is gegaan) en vervolgens kreeg deze een iets strakker, eenduidige vorm als gitbook cursus. Deze cursus, alsook een hele resem oefeningen en andere nuttige extra's kan je terugvinden op (**ziescherp.be**). De inhoud van die cursus loopt integraal gelijk aan die van dit boek. Uiteraard is de kans bestaande dat er in de online versie ondertussen weer wat minder schrijffoutjes staan.

Waarom deze korte historiek? Wel, de kans is bestaande dat er hier en daar in dit boek flarden tekst, code voorbeelden, of oefeningen niet origineel de mijne zijn. Ik heb getracht zo goed mogelijk aan te geven wat van waar komt, maar als ik toch iets vergeten ben, aarzel dan niet om me er op te wijzen.

Benodigdheden

Alle codevoorbeelden in deze cursus kan je zelf (na)maken met de gratis **Visual Studio 2019 Community** editie die je kan downloaden op visualstudio.microsoft.com[1].

[1] https://visualstudio.microsoft.com/vs/

Dankwoord

Aardig wat mensen, grotendeels studenten van 1EA (Professionele Bachelor Elektronica-ICT van de AP Hogeschool, academiejaar 2019-2020) hebben me met deze cursus geholpen. Hen allemaal afzonderlijk bedanken zou me een extra pagina kosten, toch wil ik graag volgende mensen uitdrukkelijk in de bloemetjes zetten voor hun inbreng: Ruben Simons, Ailko Claeys, Kevin Van Driel en Lennert Van Riel . Zonder hen zou dit boek veel meer fouten bevatten en niet de opbouw hebben die hij nu heeft. Indien er toch nog foutjes instaan komt dat uiteraard volledig door mezelf! Een speciale dank ook aan Maarten Wachters die de originele pixel-art van me maakte waar ik vervolgens enkele varianten op heb gemaakt!

Ook een bos bloemen voor collega's Olga Coutrin en Walter Van Hoof om de ondankbare taak op zich te nemen mijn vele dt-fouten eruit te halen op nog geen week voor de deadline. Bedankt!

1. De eerste stappen

Wel, wel, wie we hier hebben?! Iemand die de edele kunst van het programmeren wil leren? Dan ben je op de juiste plaats gekomen. Je gelooft het misschien niet, maar reeds aan het einde van dit hoofdstuk zal je je eerste eigen computer-applicaties kunnen maken. De weg naar eeuwige roem, glorie, véél vloeken en code herbruiken ligt voor je. Ben je er klaar voor?

De eerste stappen zijn nooit eenvoudig. We gaan proberen het aantal dure woorden, vreemde afkortingen en ingewikkelde schema's tot een minimum te houden. Maar toch, als je een nieuwe kunst wil leren zal je je handen (én toetsenbord) vuil moeten maken. Wat er ook gebeurt de komende hoofdstukken: blijf volhouden. Leren programmeren is een beetje als een berg leren beklimmen waarvan je nooit de top lijkt te kunnen bereiken. Wat ook zo is. Er is geen "top", en dat is net het mooie van dit alles. Er valt altijd iets nieuws te leren! De zaken waar je de komende pagina's op gaat vloeken zullen over enkele hoofdstukken al kinderspel lijken. Hou dus vol, blijf oefenen, vloek gerust af en toe en vooral: geniet van nieuwe dingen ontdekken!

1.1 Wat is programmeren?

Je hoort de termen geregeld: softwareontwikkelaar, programmeur, app-developer, etc. Allen zijn beroepen die in essentie kunnen herleid worden tot hetzelfde: programmeren. Programmeurs hebben geleerd hoe ze computers opdrachten kunnen geven (**programmeren**) zodat deze hopelijk doen wat je ze vraagt.

In de 21e eeuw is de term *computer* erg breed. Quasi ieder apparaat dat op elektriciteit werkt tegenwoordig bevat een computertje. Gaande van slimme lampen, tot de servers die het Internet draaiende houden of de smartwatch aan je pols. Zelfs aardig wat ijskasten en wasmachines beginnen (kleine) computers te bevatten.

Het grote probleem van computers, ongeacht hun grootte of kracht, is dat het in essentie ongelooflijk domme dingen zijn. Ze zullen altijd **exact** doen wat jij hen vertelt dat ze moeten doen. Als je hen dus de opdracht geeft om te ontploffen, schrik dan niet dat je even later naar de 101 kunt bellen.

Programmeren houdt in dat je leert praten met die domme computers zodat ze doen wat jij wilt dat ze doen.

Het algoritme

> First, solve the problem. Then, write the code.

Deze quote van John Johnson wordt door veel beginnende programmeurs soms met een scheef hoofd aanhoort. “Ik wil gewoon code schrijven!” Het is een mythe dat programmeurs constant code schrijven. Integendeel, een goed programmeur zal veel meer tijd in de “voorbereiding” tot code schrijven steken: het maken van een goed **algoritme**.

Het algoritme is de essentie van een computerprogramma en kan je beschouwen als het recept dat je aan de computer gaat geven zodat deze jouw probleem op de juiste manier oplost. Het algoritme bestaat uit een reeks instructies die de computer moet uitvoeren telkens jouw programma wordt uitgevoerd.

Het algoritme van een programma moet je zelf verzinnen. De volgorde waarin de instructies worden uitgevoerd zijn echter zeer belangrijk. Dit is exact hetzelfde als in het echte leven: een algoritme om je fiets op te pompen kan zijn:

```
1 Haal dop van het ventiel
2 Plaats pomp op ventiel
3 Begin te pompen
```

Eender welke andere volgorde van bovenstaande algoritme zal vreemde (en soms fatale) fouten geven.

Wil je dus leren programmeren, dan zal je logisch moeten leren denken en een analytische geest hebben. Als je eerst tegen een bal trapt voor je kijkt waar de goal staat dan zal de edele kunst van het programmeren voor jou een…speciale aangelegenheid worden.

Vanaf nu ben je trouwens gemachtigd om naar de nieuwsdiensten te mailen telkens ze foutief het woord “logaritme” gebruiken in plaats van “algoritme”. Het woord logaritme is iets wat bij sommige nachtmerries uit de lessen wiskunde opwekt en heeft hoegenaamd niets met programmeren te maken. Uiteraard kan het wel zijn dat je ooit een algoritme moet schrijven om een logaritme te berekenen. Hopelijk moet een journalist nooit voorgaande zin in een nieuwsbericht gebruiken.

Programmeertaal

Om een algoritme te schrijven dat onze computer begrijpt dienen we een programmeertaal te gebruiken. Computers hebben hun eigen taaltje dat programmeurs moeten kennen voor ze hun algoritme aan de computer kunnen *voeden*. Er zijn tal van computertalen, de ene al wat obscuurder dan de andere. Maar wat al deze talen gelijk hebben is dat ze meestal:

- **ondubbelzinnig** zijn: iedere opdracht of woord kan door de computer maar op exact één manier geïnterpreteerd worden. Dit in tegenstelling tot bijvoorbeeld het Nederlands waar "wat een koele kikker" zowel een letterlijke, als een figuurlijke betekenis heeft die niets met elkaar te maken heeft.
- bestaan uit **woordenschat**: net zoals het Nederlands heeft ook iedere programmeertaal een , meestal beperkte, lijst woorden die je kan gebruiken. Je gaat ook niet in het Nederlands zelf woorden verzinnen in de hoop dat je partner je kan begrijpen.
- bestaan uit **grammaticaregels**: Enkel Yoda mag Engels in een verkeerde volgorde gebruiken. Iedereen anders houdt zich best aan de grammatica-afspraken die een taal heeft. "bal rood is" lijkt nog begrijpbaar, maar als we zeggen "bal rood jongen is gooit veel"?

De C# taal

Net zoals er ontelbare spreektalen in de wereld zijn, zijn er ook vele programmeertalen. **C#** (spreek uit *'siesjarp'*) is er één van de vele. C# is een taal die deel uitmaakt van de .NET (spreek uit *'dotnet'*) omgeving die 20 jaar geleden door Microsoft werd ontwikkeld (juli 2000). Het fijne van C# is dat deze een zogenaamde **hogere programmeertaal** is. Hoe "hoger" de programmeertaal, hoe leesbaarder deze wordt voor leken omdat hogere programmeertalen dichter bij onze eigen taal aanleunen.

De geschiedenis van de hele .NET-wereld vertellen zou een boek op zich betekenen en gaan we hier niet doen. Het is nuttig om weten dat er een gigantische bron aan informatie over .NET en C# online te vinden is, beginnende met docs.microsoft.com[1].

Het fijne van leren programmeren is dat je binnenkort op een bepaald punt gaat komen waarbij de keuze van programmeertaal er minder toe doet. Vergelijk het met het leren van het Frans. Van zodra je Frans onder knie hebt is het veel eenvoudiger om vervolgens Italiaans of Spaans te leren. Zo ook met programmeertalen. De C# taal bijvoorbeeld lijkt bijvoorbeeld als twee druppels water op Java, alsook op de talen waar ze van afstamt, C en C++.

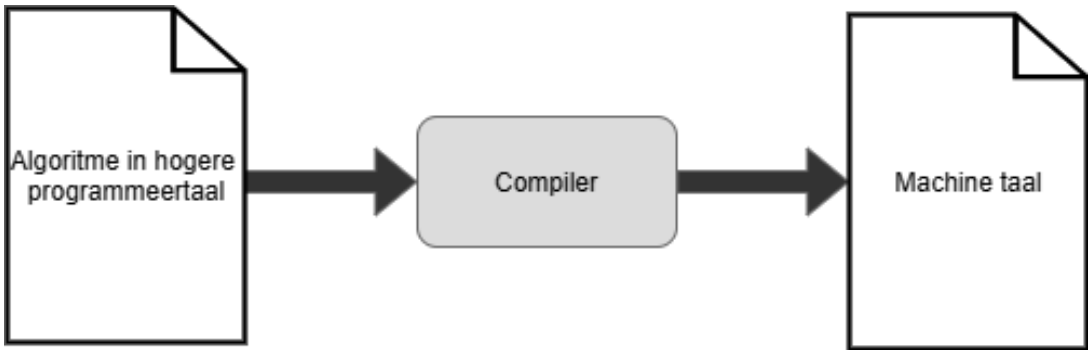
Zelfs JavaScript, Python en veel andere moderne talen zullen weinig geheimen voor jou hebben wanneer je aan het einde van dit boek bent.

[1] https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/

De compiler

Rechtstreeks onze algoritmen tegen de computer vertellen vereist dat we machinetaal kunnen. Deze is echter zo complex dat we tientallen lijnen machinetaal nodig hebben om nog maar gewoon 1 letter op het scherm te krijgen. Er werden daarom dus hogere programmeertalen ontwikkeld die aangenamer zijn dan deze zogenaamde machinetalen om met computers te praten.

Uiteraard hebben we een vertaler nodig die onze code zal vertalen naar de machinetaal van het apparaat waarop ons programma moet draaien. Deze vertaler is de **compiler** die aardig wat complex werk op zich neemt, maar dus in essentie onze code gebruiksklaar maakt voor de computer.



Vereenvoudigd compiler overzicht

Merk op dat we hier veel details van de compiler achterwege laten. De compiler is een uitermate complex element, maar in deze fase van je (prille) programmeursleven hoeven we enkel de kern van de compiler te begrijpen: **het omzetten van C# code naar een uitvoerbaar bestand geschreven in IL code**.

Microsoft .NET

Bij de geboorte van .NET in 2000 zat ook de taal C#.

.NET is een zogenaamd **framework**. Dit framework bestaat uit een grote groep van bibliotheken (*class libraries*) en een *virtual execution system* genaamd de **Common Language Runtime (CLR)**. De CLR zal ervoor zorgen dat C#, of andere .NET talen (F#, VB.NET, etc.), kunnen samenwerken met de vele bibliotheken.

Om een uitvoerbaar bestand te maken (**executable**, vandaar de extensie .exe bij uitvoerbare programma's in windows) zal de broncode die je hebt geschreven in C# worden omgezet naar **Intermediate Language** (IL) code. Op zich is deze IL code nog niet uitvoerbaar, maar dat is niet ons probleem. Wanneer een gebruiker een in IL geschreven bestand wil uitvoeren dan zal, achter de schermen, de CLR deze code ogenblikkelijk naar machine code omzetten (**Just-In-Time** of JIT compilatie) en uitvoeren. De gebruiker zal dus nooit dit proces opmerken (tenzij er geen .NET framework werd geïnstalleerd op het systeem).

1.2 Kennismaken met C# en Visual Studio

We gaan in dit boek leren programmeren met Microsoft Visual Studio 2019, een softwarepakket waar ook een gratis community versie voor bestaat. Microsoft Visual Studio (vanaf nu **VS**) is een pakket dat een groot deel van de tools samenvoegt die een programmeur nodig heeft (debugger, code editor, compiler, etc).

Het Visual Studio 2019 Logo

VS is een zogenaamde IDE ("Integrated Development Environment") en is op maat gemaakt om in C# geschreven applicaties te ontwikkelen. Je bent echter verre van verplicht om enkel C# applicaties in VS te ontwikkelen, je kan gerust VB.NET, TypeScript, Python en andere talen gebruiken. Ook vice versa ben je niet verplicht om VS te gebruiken om te ontwikkelen. Je kan zelfs in notepad code schrijven en vervolgens compileren (zie hierna). Er bestaan zelfs online C# programmeer omgevingen, zoals dotnetfiddle.net[2].

In dit boek zullen we steeds werken met Visual Studio. Niet met Visual Studio Code. Visual Studio code is een zogenaamde lightweight versie van VS die echter zeker ook z'n voordelen heeft (makkelijk uitbreidbaar, snel, compact, etc). Visual Studio vindt dankzij VS Code eindelijk ook z'n weg op andere platformen dan enkel die van Microsoft. Zoek je een lightweight versie dan moet je zeker Visual Studio Code[3] eens proberen.

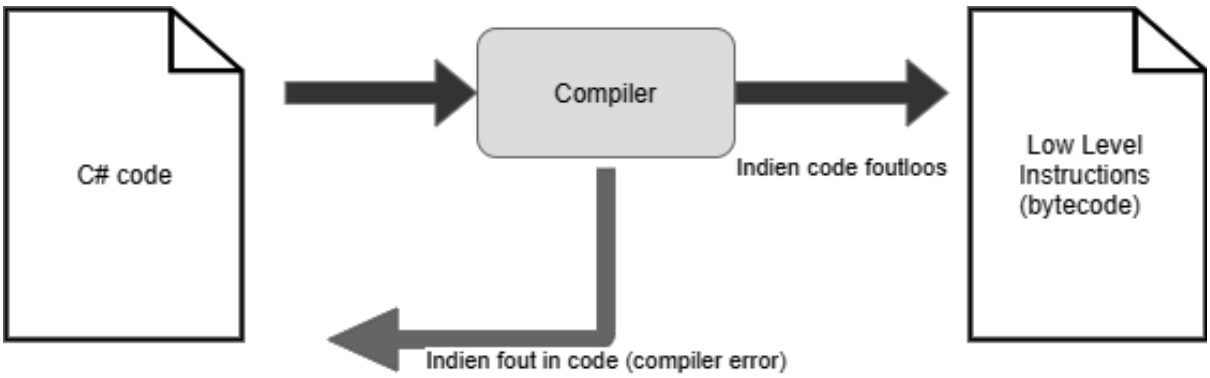
[2] https://dotnetfiddle.net/
[3] https://code.visualstudio.com/

De compiler en Visual Studio

Zoals gezegd: jouw taak als programmeur is algoritmes in C# taal uitschrijven. We zouden dit in een eenvoudige tekstverwerker kunnen doen, maar dan maken we het onszelf lastig. Net zoals je tekst in notepad kunt schrijven, is het handiger dit in bijvoorbeeld Word te doen: je krijgt een spellingchecker en allerlei handige extra's.

Ook voor het schrijven van computer code is het handiger om een zogenaamde IDE te gebruiken, een omgeving die ons zal helpen foutloze C# code te schrijven.

Het hart van Visual Studio bestaat uit de compiler die we hiervoor besproken hebben. De compiler zal je C# code omzetten naar de IL-code zodat jij (of anderen) je applicatie op een computer (of ander apparaat) kunnen gebruiken. Zolang de C# niet exact voldoet aan de C# syntax en grammatica zal de compiler het vertikken een uitvoerbaar bestand voor je te genereren.



Vereenvoudigd compiler overzicht

Visual Studio Installeren

In dit boek zullen de voorbeelden steeds met de **Community** editie van VS gemaakt zijn. Je kan deze gratis downloaden en installeren via visualstudio.microsoft.com/vs[4].

Het is belangrijk bij de installatie dat je minimaal volgende zaken selecteert:

- de **.NET desktop development** en **.NET Core cross-platform development** workload.

Uiteraard ben je vrij om meerdere zaken te installeren.

[4]https://visualstudio.microsoft.com/vs/

Visual studio opstarten

Als alles goed is geïnstalleerd kan je Visual Studio starten zoals je gewend bent in je operating system.

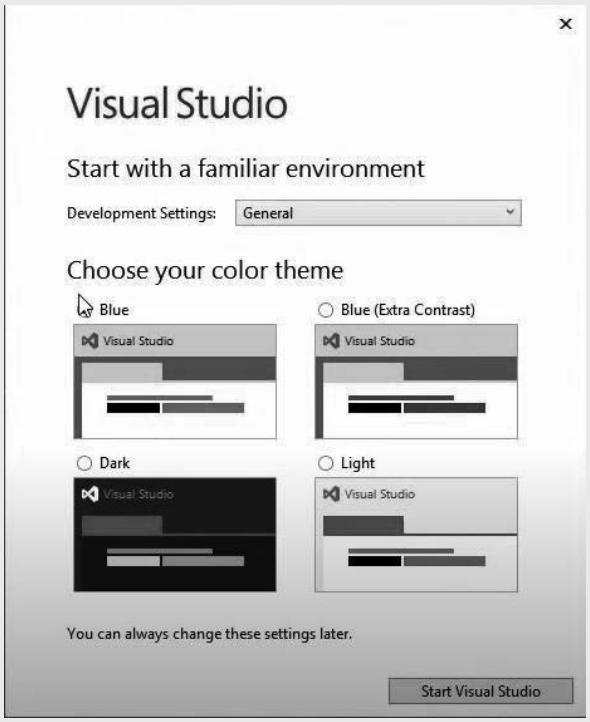
Allereerste keer opstarten

De allereerste keer dat je VS opstart krijg je 2 extra schermen te zien:

- Het “sign in” scherm mag je overslaan (kies “Not now, maybe later”)
- Vervolgens kan je je kleurenthema kiezen. Dit heeft geen invloed op de manier van werken.

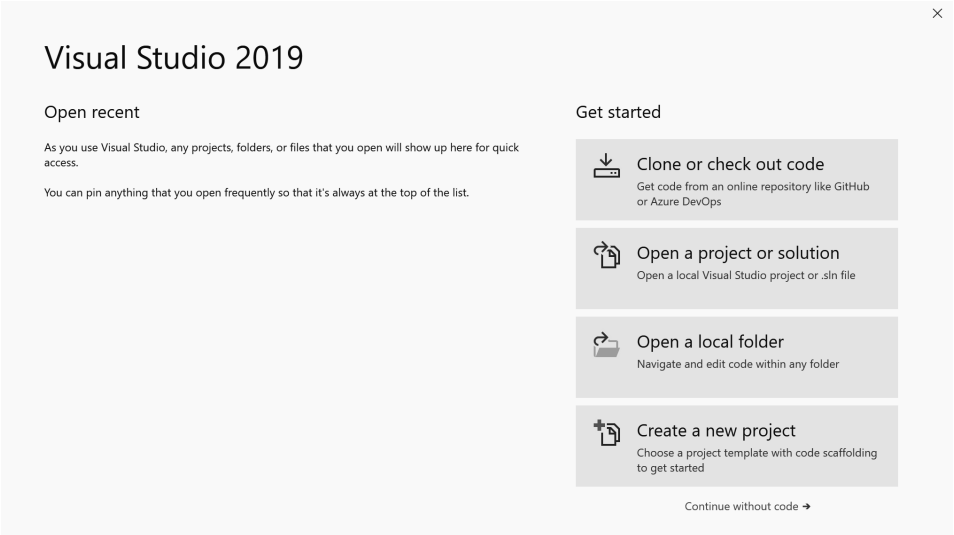
Dark is uiteraard het coolste thema om in te coderen. Je voelt je ogenblikkelijk Neo uit The Matrix. Het nadeel van dit thema is dat het veel meer inkt verbruikt indien je screenshots in een boek zoals dit wilt plaatsen. De keuze voor Development Setting kan je naar “Visual C#” veranderen, maar General is even goed (je zal geen verschil merken in eerste instantie).

Je kan dit achteraf nog aanpassen in VS via “Tools” in de menubalk, dan “Import and Export Settings” en kiezen voor “Import and Export Settings Wizard”.



Project keuze

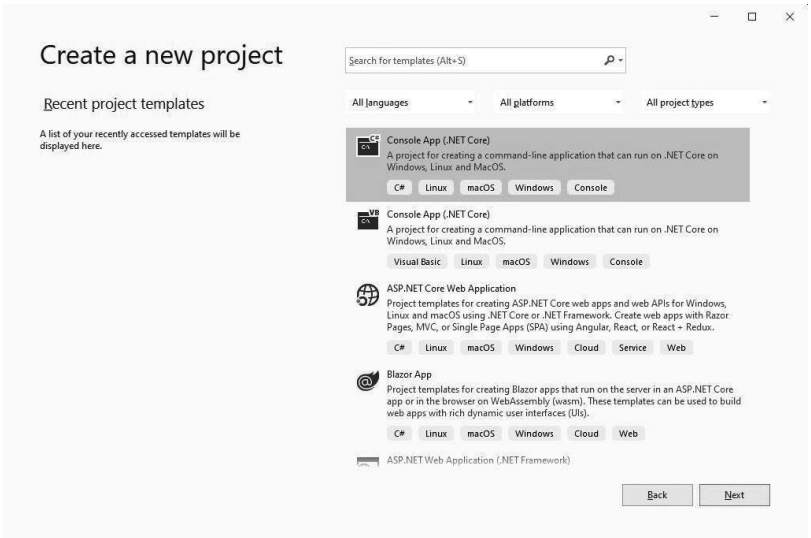
Na het opstarten van VS krijg je het startvenster te zien van waaruit je verschillende dingen kan doen. Van zodra je projecten gaat aanmaken zullen deze in de toekomst ook op dit scherm getoond worden zodat je snel naar een voorgaand project kunt gaan.



Het startscherm

Een nieuw project aanmaken

We zullen nu een nieuw project aanmaken, kies hiervoor “Create a new project”.

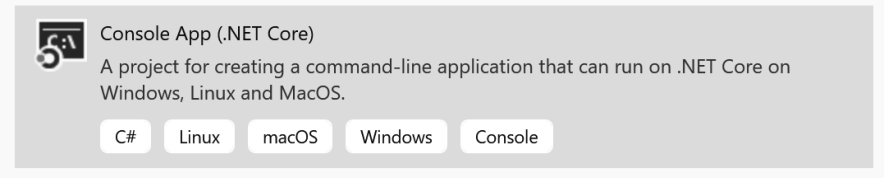


Kies je projecttype

Het “New Project” venster dat nu verschijnt geeft je hopelijk al een glimp van de veelzijdigheid van VS. In het rechterdeel zie je bijvoorbeeld alle Project Types staan. M.a.w. dit zijn alle soorten programma’s die je kan maken in VS. Naargelang de geïnstalleerde opties en bibliotheken zal deze lijst groter of kleiner zijn.

In dit boek zullen we altijd het Project Type **Console App (.NET Core)** gebruiken. Een console applicatie is een programma dat alle uitvoer naar een zogenaamde *console* stuurt, een shell. M.a.w., je kan enkel tekst (Unicode) als uitvoer genereren en dus geen multimedia elementen zoals afbeeldingen, geluid, etc.

Kies dit type en klik 'Next'.



Een VS project aanmaken. Zoals je ziet zal dit soort applicatie op alle gekende computer besturingssystemen werken (Windows, Linux, Mac)

Op het volgende scherm kan je een naam ingeven voor je project alsook de locatie op de harde schijf waar het project dient opgeslagen te worden. **Onthoud waar je je project aanmaakt zodat je dit later terugvindt**.

De solution name blijf je af (deze moet momenteel dezelfde naam zijn als je project).

Geef je projectnamen ogenblikkelijk duidelijke namen zodat je niet opgezadeld geraakt met projecten zoals Project201, etc. waarvan je niet meer weet welke belangrijk zijn en welke niet.

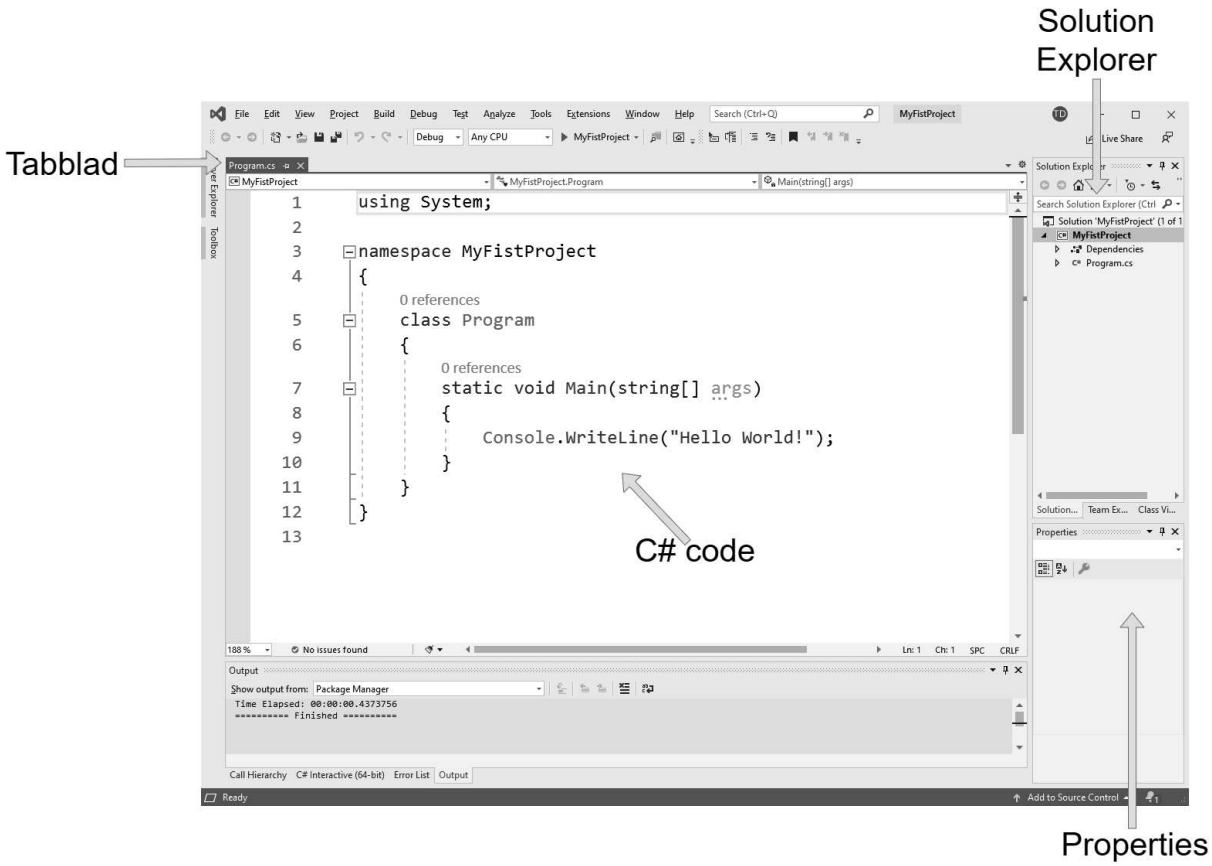
Geef je project de naam "MyFirstProject" en kies een goede locatie (ik raad je aan dit steeds in Dropbox of Onedrive te doen). We raden af om de checkbox onderaan aan te vinken. In de toekomst zal het nuttig zijn dat je meer dan 1 project per solution zal kunnen hebben. Lig er nog niet van wakker. **Klik nu op create**.

VS heeft nu reeds een aantal bestanden aangemaakt die je nodig hebt om een 'Console Applicatie' te maken.

IDE Layout

Wanneer je VS opstart zal je mogelijk overweldigd worden door de hoeveelheid menu's, knopjes, schermen, etc. Dit is normaal voor een IDE: deze wil zoveel mogelijk mogelijkheden aanbieden aan de gebruiker. Vergelijk dit met Word: afhankelijk van wat je gaat doen gebruikt iedere gebruiker andere zaken van Word. De makers van Word kunnen dus niet bepaalde zaken weglaten, ze moeten net zoveel mogelijk aanbieden.

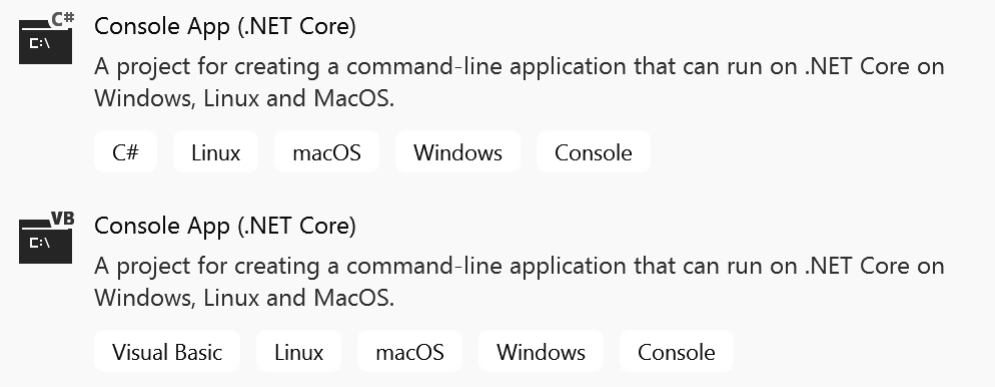
We zullen nu eerst eens bekijken wat we allemaal zien in VS na het aanmaken van een nieuw programma.



VS IDE Overzicht

- Je kan meerdere bestanden tegelijkertijd openen in VS. Ieder bestand zal z'n eigen **tab** krijgen. De actieve tab is het bestand wiens inhoud je in het hoofdgedeelte eronder te zien krijgt. Merk op dat enkel open bestanden een tab krijgen. Je kan deze tabbladen ook "lostrekken" om bijvoorbeeld enkel dat tabblad op een ander scherm te plaatsen.
- De "**solution explorer**" toont alle bestanden en elementen die tot het huidige project behoren. Als we dus later nieuwe bestanden toevoegen, dan kan je die hier zien (en openen). Verwijder hier géén bestanden zonder dat je zeker weet wat je aan het doen bent.
- Het **properties** venster (eigenschappen) is een belangrijk venster. Hier komen alle eigenschappen van het huidige geselecteerde element. Selecteer bijvoorbeeld maar eens Program.cs in de solution explorer en merk op dat er allerlei eigenschappen getoond worden. Onderaan het Properties venster wordt steeds meer informatie getoond over de huidig geselecteerde eigenschap.

Indien je een nieuw project hebt aangemaakt en de code die je te zien krijgt lijkt in de verste verte niet op de code die je hierboven ziet dan heb je vermoedelijk een verkeerd projecttype aangemaakt. De meest gemaakte fout in deze fase is dat je een Visual Basic (VB) Console applicatie hebt gekozen en dus niet Visual C# versie. Deze staan vlak onder elkaar bij het keuzemenu en je ziet het verschil amper. (Je ziet het verschil onder andere aan het kleine VB logo in het icoontje).



VB en C#: vrienden voor het leven, met een lang, kleurrijk, bewogen verleden

Layout kapot/kwijt/vreemd

De layout van VS kan je volledig naar je hand zetten. Je kan ieder (deel-)venster en tab verzetten, verankeren en zelfs verplaatsen naar een ander bureaublad. Experimenteer hier gerust mee en besef dat je steeds alles kan herstellen. Het gebeurt namelijk al eens dat je layout een beetje om zeep is:

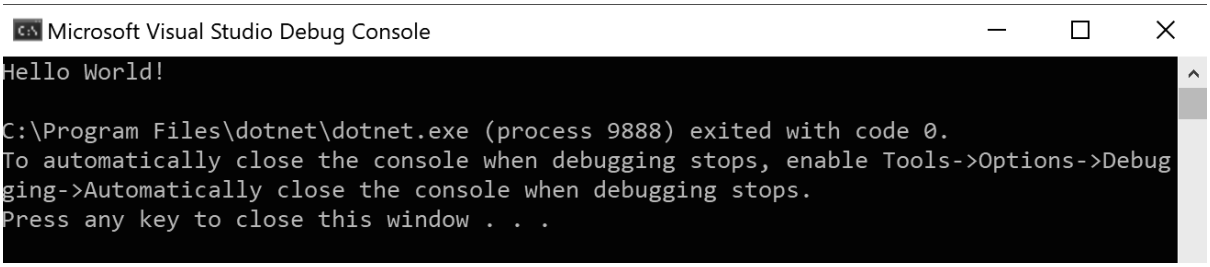
- Om eenvoudig een venster terug te krijgen, bijvoorbeeld het properties window of de solution explorer: klik bovenaan in de menubalk op “View” en kies dan het gewenste venster (soms staat dit in een submenu).
- Je kan ook altijd je layout in z’n geheel **resetten**: ga naar “Window” en kies “Reset window layout”.

Je programma starten

De code die VS voor je heeft gemaakt is reeds een werkend, maar weinig nuttig, programma. Je kan de code compileren en uitvoeren door op de groene driehoek bovenaan te klikken:

Het programma uitvoeren

Als alles goed gaat krijg je nu "Hello World!" te zien en wat extra informatie omtrent het programma dat net werd uitgevoerd:



Uitvoer van het programma

Veel doet je programma nog niet natuurlijk, dus sluit dit venster maar terug af door een willekeurige toets in te drukken.

Is dit alles?

Nee hoor. Visual Studio is lekker groot, maar laat je dat niet afschrikken. Net zoals voor het eerst op een nieuwe reisbbestemming komen, kan deze in het begin overweldigend zijn, tot je weet waar het zwembad en de pingpongtafel staat en je van daaruit je weg stilletjes aan leert kennen.