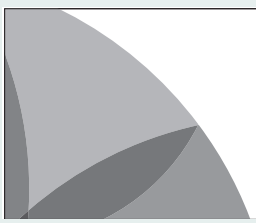
HOOFDSTUK 1
De achtergronden van C#

Dit hoofdstuk behandelt:

- hoe en waarom C# tot stand kwam;
- de Microsoft .NET-omgeving;
- de beginselen van het programmeren.



MyLab | Nederlandstalig

Op www.pearsonmylab.nl vind je studiemateriaal en de eText om je begrip en kennis van dit hoofdstuk uit te breiden en te oefenen.

1.1 De geschiedenis van C#

Een computerprogramma is een serie instructies die door een computer wordt opgevolgd. Het doel van deze instructies is om de computer een bepaalde taak uit te laten voeren (bijvoorbeeld een spelletje spelen, een e-mail verzenden enzovoort). De instructies zijn geschreven in een specifieke stijl en moeten voldoen aan de regels van de betreffende programmeertaal. Er bestaan honderden programmeertalen, maar slechts enkele daarvan hebben succes gehad en worden nu over de hele wereld gebruikt. Bij het programmeren is sprake van een interessante evolutie. Hier bekijken we het ontstaan van C# (uitgesproken als *see sharp*).

Rond 1960 werd de programmeertaal Algol gecreëerd. Algol staat voor Algoritmic Language. Een algoritme is een sequentie van stappen om een probleem op te lossen. Men streefde ernaar om deze sequenties uit te drukken in een taal die zo veel mogelijk leek op de gangbare wiskundige notatie, gecombineerd met gewoon Engels. Deze taal werd vooral populair in academische milieus, maar de kerngedachte ervan beïnvloedt nog altijd alle programmeertalen.

In dezelfde periode was de taal COBOL erg populair voor het verwerken van gegevens, en de taal FORTRAN voor wetenschappelijke berekeningen. In het Verenigd Koninkrijk evolueerde Algol naar CPL (Combined Programming Language) en verder naar BCPL (Basic Combined Programming Language). In de Verenigde Staten werkte Dennis Ritchie van Bell Laboratories USA verder aan BCPL om uiteindelijk te komen tot de programmeertaal

B, rond 1970 verfijnd tot C. C was enorm populair. Deze taal werd gebruikt om het UNIX-besturingssysteem te schrijven, en in de jaren negentig schreef Linus Torvalds een UNIX-variant voor pc's, Linux genaamd.
De volgende stap kwam toen Bjarne Stroustrup de taal C++ ('C plus plus') bedacht, ook voor Bell Labs. Hiermee konden programmeurs codefragmenten maken en hergebruiken via een methode die bekendstaat als 'objectgeoriënteerd programmeren' (OOP). De naam C++ verwijst naar C, waarin je een waarde met 1 kan ophogen door twee plustekens. C++ staat dus een trapje hoger dan C.
C++ is vandaag de dag nog altijd erg populair, maar het is geen eenvoudige taal. Daarom ontwikkelde James Gosling van Sun Microsystems in 1995 de taal Java. Objectoriëntatie staat bij Java centraal, maar de taal is veel eenvoudiger te gebruiken dan C++. Java-programma's hebben bovendien het voordeel dat ze op veel systemen kunnen draaien (Windows-pc's, Apple-computers, Linux en zelfs mobiele telefoons). Daarom is Java ook vandaag de dag nog erg in trek.
In 2002 kondigde Microsoft een nieuwe taal aan, C# genaamd. Deze taal was gelijkwaardig aan Java en dus ook aan C++, maar bevatte opnieuw enkele verbeteringen. In de muziekwereld betekent een hekje (#, spreek uit *sharp*) zoveel als een halve toon hoger. C# vormde een fundamenteel onderdeel van de .NET-beweging ('dot net'-beweging) die we hieronder beschrijven. Door de treffende gelijkenissen kun je een taal als C, C++ of Java gemakkelijk aanleren als je C# onder de knie hebt.
Natuurlijk is dit slechts een deel van de geschiedenis. Een andere belangrijke ontwikkeling was die van de taal BASIC rond 1964. Die taal zou uiteindelijk evolueren naar Visual Basic, nog altijd een onderdeel van .NET.

1.2 De Microsoft .NET-omgeving

In 2002 introduceerde Microsoft een belangrijk nieuw product, het .NET-raamwerk (*framework*) genaamd. De voornaamste pluspunten van .NET zijn:

- Het wordt geleverd met de programmeertalen C#, Visual Basic, C++ en F#.
- Het heeft faciliteiten die het voor programmeurs eenvoudiger maken om nagenoeg alle soorten applicaties te bouwen: traditionele venstergebaseerde toepassingen, mobiele applicaties (apps), webapplicaties, games en nog veel meer! De naam .NET geeft aan dat Microsoft de rol van het internet als cruciaal beschouwt.
- De .NET-omgeving is niet noodzakelijk gebonden aan Microsoft Windows, ook voor andere besturingssystemen zijn er mogelijkheden.[1]
- Het maakt het mogelijk software te maken met behulp van bouwstenen (objecten) die over een netwerk verspreid kunnen zijn.

In 2016 heeft Microsoft het .NET Core-raamwerk uitgebracht. Dit is een nieuwe, herschreven versie van het klassieke .NET-raamwerk specifiek toegespitst op server-toepassingen en apps die op meerdere besturingssystemen kunnen draaien (Windows,

1 Surf maar eens naar http://dot.net, daar vind je ontwikkeltools om C# te schrijven voor bijna elk platform.

Linux en macos). Daarnaast bestaat nog steeds het 'klassieke' .NET-raamwerk. Alle voorbeelden uit het boek zijn met dit klassieke raamwerk gebouwd, tenzij anders vermeld.

1.3 Wat is een programma?

We proberen nu eerst je een indruk te geven van wat een programma is. Een manier om dat te doen, is door te wijzen op de overeenkomsten met recepten, muziekpartituren en breipatronen. Zelfs de gebruiksaanwijzing op een fles shampoo is een eenvoudig programma:

```
Maak het haar nat.
Doe er shampoo op.
Wrijf de shampoo door het haar.
Spoel het uit.
```

Dit programma is weliswaar een lijst van instructies voor een mens, maar het verduidelijkt een belangrijk aspect van een computerprogramma: een *programma* is een opeenvolging van instructies die in een bepaalde volgorde worden uitgevoerd, beginnend met de eerste opdracht en eindigend met de laatste. We kunnen dit vergelijken met een recept, een muziekpartituur of een breipatroon: ook deze 'programma's' bestaan uit een lijst van instructies die in de goede volgorde moeten worden afgewerkt. Neem het breipatroon: er bestaan breimachines die gevoed worden met een programma van instructies en die deze instructies vervolgens automatisch uitvoeren (*executeren*). Dit nu is de essentie van een computer: een machine die automatisch een serie instructies, een programma, uitvoert. Dit houdt echter wel in dat als wij zelf een fout maken in deze instructies, de computer de verkeerde taak zal uitvoeren. Op de keper beschouwd mag je dus gerust stellen dat een computer een 'dom ding' is dat uit zichzelf niets intelligents doet. Een programmeur moet expliciete instructies intikken. Standaardvoorbeelden van opdrachten waarop een computer reageert zijn:

- Lees een getal in dat de gebruiker heeft ingevoerd.
- Lees een aantal tekens in (letters en cijfers).
- Geef een paar tekens als uitvoer.
- Maak een berekening.
- Geef een getal als uitvoer.
- Zet een plaatje op het scherm.
- Reageer als een knop op het scherm met de muis wordt aangeklikt.

Programmeren is eigenlijk niets anders dan het selecteren van instructies uit een dergelijke lijst, zodat de gewenste taak wordt uitgevoerd. Deze instructies zijn geschreven in een speciaal jargon: de *programmeertaal*. C# is een van de vele programmeertalen die er bestaan. Leren programmeren houdt in dat je de verschillende mogelijkheden van de programmeertaal leert kennen en gebruiken om de computer iets te laten doen wat jij wilt. Het voorbeeld van de muziekpartituur laat een ander aspect van programma's zien. In de muziek is het gebruikelijk om sommige stukken te herhalen, bijvoorbeeld het

refrein. Notenschrift bespaart de componist de moeite de te herhalen stukken muziek steeds opnieuw op te moeten schrijven; in plaats daarvan wordt een aanduiding gebruikt die aangeeft dat een bepaald stuk moet worden herhaald. Hetzelfde gebeurt in een programma. Het komt heel vaak voor dat een bepaalde handeling moet worden herhaald, bijvoorbeeld wanneer een stuk tekst doorzocht moet worden op het voorkomen van een bepaald woord. Herhaling (of *iteratie*) is een standaardopdracht in programma's en C# heeft hiervoor speciale instructies.

In recepten staat vaak zoiets als: 'Mocht je geen verse erwten hebben, neem dan diepvries-erwten'. Dit illustreert weer een ander aspect van programma's: ze voeren vaak een test uit en op basis van de uitkomst van die test maken ze een keuze tussen twee opties. Dit proces heet *selectie* en ook hier heeft C# speciale instructies voor.

Als je ooit een recept uit een kookboek hebt gebruikt om een maaltijd te bereiden, dan weet je vast dat er in een recept regelmatig wordt verwezen naar een ander recept. Bijvoorbeeld wanneer je eerst naar een andere pagina moet bladeren om te lezen hoe je rijst kookt voor je met het recept voor jouw rijstmaaltijd verder kunt gaan: van de bereiding van de rijst is een subtaak gemaakt. Deze manier van opdrachten schrijven heeft een belangrijke tegenhanger in programmeren. In C# en andere objectgeoriënteerde talen heten die 'recepten' *methoden*. Methoden worden in feite in alle programmeertalen gebruikt, maar soms heten ze anders, bijvoorbeeld functies, procedures, subroutines of subprogramma's. Methoden zijn subtaken en worden zo genoemd, omdat ze aangeven hoe iets gedaan moet worden. Het gebruik van methoden geeft overzicht als je door de bomen het bos niet meer ziet.

We keren nog even terug naar onze analogie met recepten. Neem nu het bereiden van een kerriesaus. Een paar jaar geleden zou in het recept nog vermeld staan dat je verse kruiden moest kopen, deze moest malen en vervolgens moest bakken. Maar tegenwoordig kun je gewoon kant-en-klare sauzen kopen. Dat maakt het voor ons een stuk eenvoudiger. De analogie met programmeren ligt erin dat het schrijven van een programma veel gemakkelijker wordt als je gebruik kunt maken van een verzameling kant-en-klare *objecten*, zoals knoppen, schuifbalken en databases. C# beschikt over een grote verzameling objecten (verzameld in bibliotheken) die we eenvoudig kunnen opnemen in ons programma, in plaats van dat we het wiel zelf opnieuw uit moeten vinden.

Kort samengevat is een programma een lijst van instructies die automatisch worden opgevolgd door een computer. Een programma bestaat uit combinaties van:

- opeenvolgingen;
- herhalingen;
- selecties;
- methoden;
- kant-en-klare objecten;
- objecten die je zelf schrijft.

Deze eigenschappen worden gedeeld door alle moderne programmeertalen.

TESTVRAGEN

1.1 Hieronder staan enkele instructies om het salaris van een medewerker te berekenen:

```
Bepaal het aantal gewerkte uren.
Bereken het salaris.
Druk de salarisstrook af.
Verminder het salaris met de aftrek voor ziekteverzuim.
```

Wordt hier een fout gemaakt?

1.2 Detailleer de instructie:

```
Wrijf de shampoo door het haar.
```

Maak hierbij gebruik van herhalingen.

1.3 De volgende instructie staat vermeld op een bord bij een achtbaan:

```
Maak alleen gebruik van deze achtbaan als u ouder bent dan 8 of jonger
dan 70!
```

Wat is er mis met deze waarschuwing? Hoe kunnen we de formulering verbeteren?

Programmeerprincipes

- Programma's bevatten methoden voor opeenvolging, selectie, herhaling en het gebruik van subtaken.
- Het schrijven van programma's wordt aanzienlijk vereenvoudigd door de beschikbaarheid van kant-en-klare objecten.

Programmeervalkuilen

Fouten maken is menselijk, bijvoorbeeld het in de verkeerde volgorde zetten van instructies.

Samenvatting

- C# is een objectgeoriënteerde taal, afgeleid van Java en C++.
- De Microsoft .NET-omgeving is een zeer krachtig softwareproduct en bevat met C#, Visual Basic, C++ en F# enkele geavanceerde hedendaagse programmeertalen.
- Een programma is een lijst van instructies die automatisch worden opgevolgd door een computer.

- Objectgeoriënteerd programmeren (OOP) blijft erg belangrijk in het hedendaagse programmeren; C# maakt er volop gebruik van.

Opgaven

1.1

Denk eens aan de stappen die een student achtereenvolgens onderneemt vanaf het moment van ontwaken tot aan het moment dat hij naar college gaat. Hier is een suggestie voor de eerste stappen:

```
Word wakker.
Kleed je aan.
Eet je ontbijt.
Poets je tanden.
...
```

- Vul deze stappen aan. Bedenk dat er meerdere antwoorden mogelijk zijn, aangezien de stappen van persoon tot persoon kunnen variëren.
- De stap 'poets je tanden' bevat een herhaling, we borstelen immers meer dan eens. Wijs nog een andere stap aan die een herhaling inhoudt.
- Wijs een stap aan die een selectie(keuze) bevat.
- Kies een bepaalde stap en deel deze op in kleinere stappen.

1.2

Je krijgt een enorme stapel papier met 10.000 getallen erop, in een willekeurige volgorde. Beschrijf hoe (volgens welk proces) je de stapel zou doorzoeken om het grootste getal te vinden. Let erop dat het proces helder is en geen dubbelzinnige uitleg bevat. Wijs aan waar je selectie en herhaling gebruikt in je proces.

1.3

Probeer een reeks duidelijke instructies op te schrijven voor het spelletje boter-kaas-en-eieren[2] die ervoor zorgt dat een speler het spel zal winnen. Is dit niet mogelijk, probeer dan in elk geval te voorkomen dat hij verliest.

Antwoorden op de testvragen

1.1 De volgorde van de instructies is niet goed. De aftrek voor ziekteverzuim moet met het salaris verrekend worden voordat de salarisstrook afgedrukt wordt.

2 De regels van dit spelletje vind je terug op: http://nl.wikipedia.org/wiki/Boter-kaas-en-eieren.

1.2 Een mogelijkheid zou zijn:

```
Blijf wrijven door je haar tot alles goed gewassen is.
```
of
```
Blijf wrijven door je haar zolang niet alles goed gewassen is.
```

1.3 Het woordje ‘of’ is fout. Iemand van 73 is ouder dan 8 en zou dus volgens deze instructie in de achtbaan mogen. We zouden het woordje ‘of’ door ‘en’ kunnen vervangen om de instructie te corrigeren, maar dat zou nog steeds voor verwarring kunnen zorgen. We zouden het ook op de volgende manier kunnen zeggen:

```
Maak alleen gebruik van deze achtbaan als u tussen de 8 en 70 jaar oud bent.
```

Maar wees er dan wel op voorbereid dat je de waarschuwing nóg een keer moet aanpassen als er opeens hele hordes 8- en 70-jarigen in willen!