

Les 1 - Inleiding tot programmeren

» Volgende les

Stel een vraag

Programmeren voor Beginners - Les 1

Je hebt 1 vraagmoment. 1 moment over.

[Ga naar stel een vraag](#)

Inhoudsopgave

- 1.1** [Inleiding](#)
- 1.2** [Wat is programmeren?](#)
- 1.3** [Computer](#)
- 1.4** [Programma](#)
- 1.5** [Binaire systeem](#)
- 1.6** [Geschiedenis van programmeren](#)
- 1.7** [Samenvatting](#)
- 1.8** [Vraagmoment](#)
- 1.9** [Oefenopgaven](#)
- 1.10** [Huiswerkopgaven](#)

1.1 Inleiding

Voordat u aan de slag gaat met programmeren, is het goed om te weten wat het precies inhoudt. In deze les definiëren we 'programmeren' en zoomen we in op de belangrijkste onderdelen van dit begrip: de computer en een programma. Daarnaast bekijken we hoe het programmeren zich door de jaren heeft ontwikkeld.

Leerdoelen

Aan het einde van deze les weet u:

- wat programmeren is;
- hoe de computer zich heeft ontwikkeld;
- uit welke onderdelen een computer bestaat;
- wat een programma is;
- hoe het binaire systeem werkt op een computer;
- hoe u decimale getallen kunt omrekenen naar binaire getallen;
- hoe u binaire getallen kunt omrekenen naar decimale getallen.



Enkele afbeeldingen in deze les zijn moeilijk leesbaar in de printversie van deze les. Wilt u een afbeelding uitvergroten? Bekijk dan de digitale versie van de les op Plaza. Klik op de afbeeldingen om ze te vergroten.

1.2 Wat is programmeren?

Deze cursus leert u de beginselen van het programmeren. Maar wat is programmeren? De term is vandaag de dag behoorlijke ingeburgerd en de meeste mensen hebben er een globaal beeld bij, zoals 'iets met code' en 'iets met computers'.

Om deze cursus goed te starten, geven we u meteen een concrete definitie. Er zijn veel definities, maar in deze cursus gaan we uit van de volgende:

Programmeren is het schrijven van een programma dat een computer kan uitvoeren.

Iemand die een computerprogramma gebruikt, vertelt een computer wat hij moet doen. We geven daarvan een voorbeeld:

Het tekstverwerkingsprogramma Microsoft Word is geprogrammeerd. Daardoor kan de eindgebruiker de computer vertellen wat er moet gebeuren, zoals het typen van een brief, de brief opslaan, de brief printen en het programma afsluiten.



In de definitie zijn twee begrippen onderstreept, namelijk 'programma' en 'computer'. Om de definitie beter te kunnen begrijpen, is het nodig om deze begrippen nader te bekijken. Dat doen we in de volgende paragrafen.

1.3 Computer

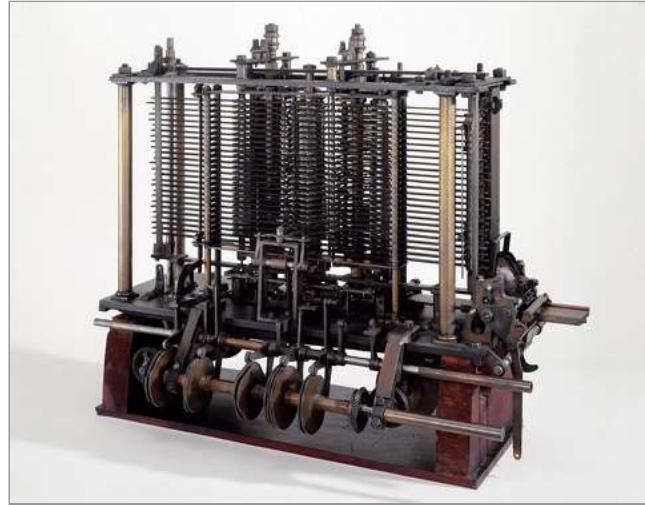
Het woord 'computer' is afgeleid van het Engelse *to compute*, dat weer afkomstig is van het Latijnse woord *computare*, dat '(be)rekenen' betekent. Men koos indertijd voor deze benaming, omdat de computer in de beginjaren vooral werd gebruikt als rekenapparaat. Tegenwoordig kunnen computers veel meer.

In deze paragraaf kijken we naar de opkomst, ontwikkeling en opbouw van de computer.

1.3.1 Geschiedenis

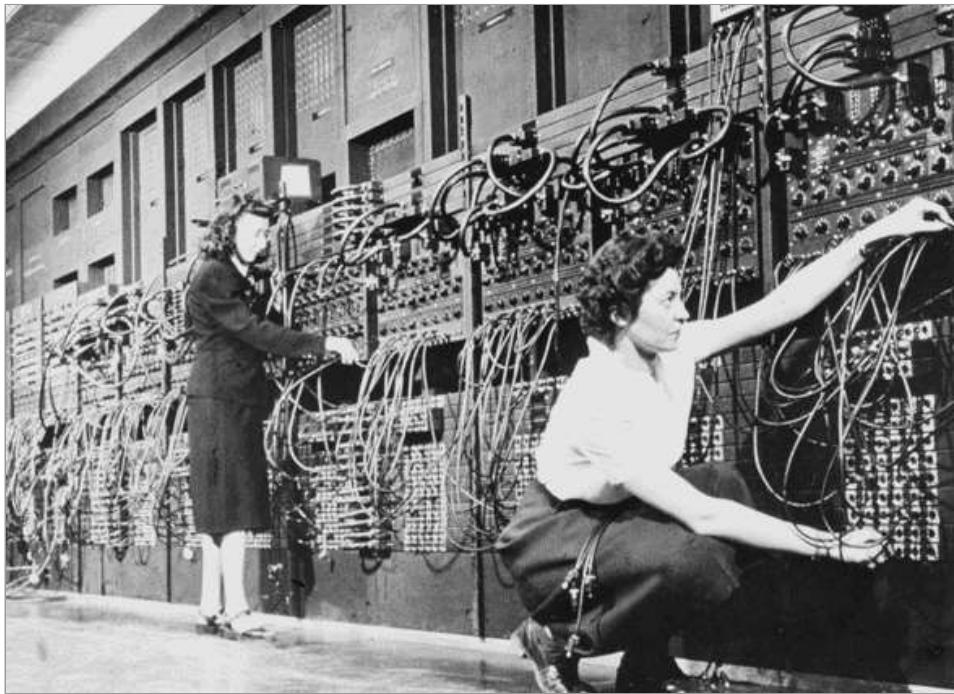
Op de vraag wie de computer uitvond, zijn meerdere antwoorden mogelijk:

- Een veelgehoord antwoord is **Charles Babbage**. Deze Engelse wiskundeprofessor vond halverwege de negentiende eeuw een computer uit. Zijn geautomatiseerde rekenmachine bestond echter alleen op papier: hij heeft de machine zelf nooit volledig gebouwd.
- Een ander mogelijk antwoord is **Konrad Zuse**. Hij bouwde in 1938 een computer, genaamd de Z1. Een paar jaar later bouwde Zuse de eerste volledige, functionele elektromechanische computer, de Z3.



De uiteindelijke computer van Babbage (links) en een replica van Zuse's Z3 (rechts).

- Vraag een Engelsman wat de eerste computer was, dan is de kans groot dat het antwoord de **Colossus** zal zijn. Deze computer dateert van 1943 en werd tijdens de Tweede Wereldoorlog gebruikt om geheime codes van de Duitse tegenmacht te kraken.
- Zou u diezelfde vraag stellen aan een Amerikaan, dan zal die waarschijnlijk de **ENIAC** als antwoord geven. Het is echter feitelijk onjuist dat dit de eerste computer was, de *Colossus* was er eerder.



De ENIAC-computer uit 1946.

Van 1950 tot 1980 werden vooral zogenoemde **mainframes** gebouwd. Dit waren zeer grote computers, waar honderden tot duizenden gebruikers tegelijkertijd op konden werken. Dit soort computers werd vooral gebruikt door banken en verzekерingsmaatschappijen. Ze zijn te vergelijken met de computers die wij vandaag de dag gebruiken, maar dan vele malen groter. De bekendste bouwer van mainframes is **IBM**.

Door de snelle ontwikkeling van de elektronica, vooral door de komst van 'halfgeleiders' (transistoren), konden computers van een steeds kleinere omvang geproduceerd worden, zonder functionaliteit te verliezen. Zo werden eind jaren zeventig de **Apple II** en de **Commodore PET** uitgebracht: kleinschalige computers, die zelfstandig op kantoor of in de huiskamer gebruikt konden worden. **IBM** bracht in 1981 voor het eerst een computer uit met de benaming **Personal Computer** (kortweg pc).

Sindsdien werd het produceren van dit soort computers steeds goedkoper en gemakkelijker, waardoor steeds meer bedrijven en huishoudens er een kochten. Een paar jaar later werd de **IBM Portable Personal Computer** op de markt gebracht, die we met enige moeite als de eerste laptop mogen beschouwen.



De Apple II (links) en de IBM Portable PC 5155 model 68 (rechts).

Vanaf hier is de geschiedenis van de computer tamelijk bekend. De concurrentie van *IBM* met *Apple* en later met *Compaq* is uit een veel recentere geschiedenis en heeft ertoe geleid dat computers steeds goedkoper werden en nu in vrijwel elk huishouden te vinden zijn.



De door ons geschatte geschiedenis is vrij globaal en beperkt. Een vollediger overzicht van alle belangrijke momenten in de geschiedenis van de computer, compleet met jaartallen, vindt u op www.livescience.com/20718-computer-history.html.

1.3.2 Opbouw

In het dagelijks leven is het de gewoonte om te spreken van 'de computer'. Meestal bedoelen we daarmee niet alleen de computer zelf (de centrale machine), maar ook alle daarop aangesloten apparaten (de **randapparatuur**). De samenstelling van alle apparaten in en aan de computer noemen we de **computerconfiguratie**.

We lichten de verschillende onderdelen van de computer kort toe:

- **Centrale machine**

De centrale machine is de eigenlijke computer. Deze vormt het hart van de computerconfiguratie. In de centrale machine bevinden zich twee onderdelen die voor elke computer onmisbaar zijn: de centrale verwerkingseenheid (ook wel de **processor** genoemd) en het interne geheugen.

Programma's en gegevens moeten in het intern geheugen worden gezet voordat ermee gewerkt kan worden. De centrale verwerkingseenheid is als het ware het brein van de machine. Alle handelingen die een computer uitvoert, worden gestuurd door de centrale verwerkingseenheid.

- **Invoerapparaten**

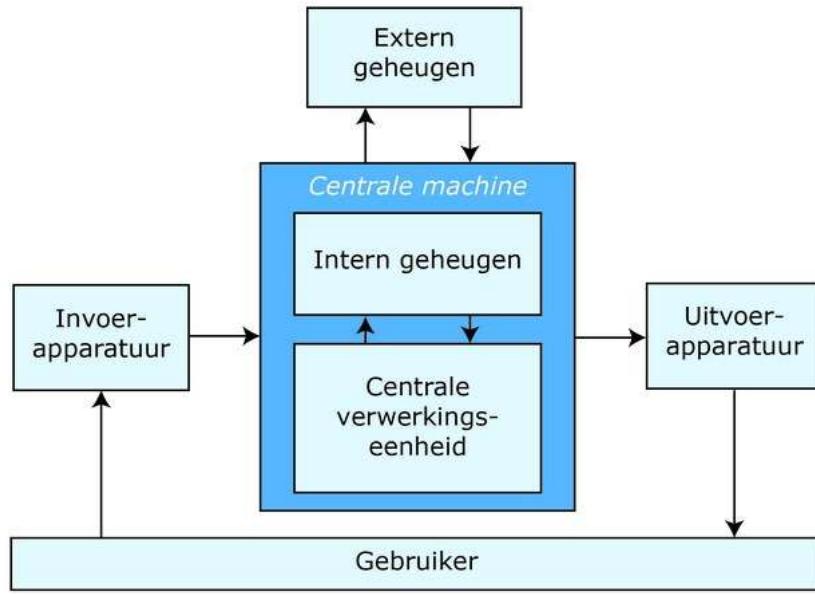
Deze apparaten dienen om gegevens en opdrachten door te geven aan de centrale machine. De belangrijkste zijn niet alleen het toetsenbord en de muis, maar ook een scanner, touchscreen en microfoon.

- **Uitvoerapparaten**

Deze apparaten dienen om gegevens vanuit de centrale machine uit te voeren en deze aan de gebruiker te laten zien, of om ze op te slaan op geheugens waar ze langdurig kunnen worden bewaard om ze later weer te gebruiken. Het beeldscherm, de printer en geluidsboxen zijn voorbeelden van uitvoerapparaten.

- **Externe geheugens**

Dit zijn apparaten waarmee de centrale verwerkingseenheid gegevens en opdrachten vanuit het intern geheugen op gegevensdragers kan opslaan en deze (later) vanaf deze gegevensdragers kan inlezen in het intern geheugen. De belangrijkste is de vaste schijf die in de systeemkast zit. Ook een USB-stick is een voorbeeld van een extern geheugen.

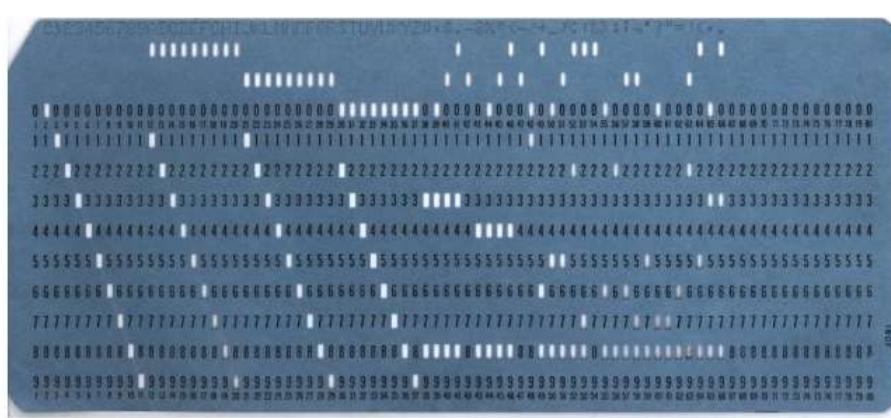


Schematische weergave van de computerconfiguratie.

1.4 Programma

Programmeren is dus het schrijven van een computerprogramma. Omschrijven wat een programma is, is een stuk lastiger dan omschrijven wat een computer is. Dat komt doordat de gebruiker van het programma vaak invloed kan uitoefenen op wat een computer zal doen.

De eerste computerprogramma's waren **ponskaarten**: kaarten van dik papier met gaten erin. Deze gaten vormden de code die de computer een opdracht gaf, zoals *Bereken de uitkomst van 29871 gedeeld door 299 en onthoud dat in geheugenbank 1*.



Voorbeeld van een ponskaart.

Ongeveer tot halverwege de jaren zeventig maakten, bewerkten en bewaarden programmeurs hun programma's op ponskaarten. Tegenwoordig zijn de ponskaarten vervangen door moderne middelen, maar de codering met nullen en enen (gaatje in de ponskaart of niet) verloopt nog altijd op dezelfde manier. Daarover leert u verderop in deze les meer.

Deze code is tegenwoordig voor u als gebruiker van het programma niet te zien. Alles wat u op uw beeldscherm ziet, bestaat echter aan de achterkant nog steeds uit deze code. Dankzij deze code weet de computer wat hij moet laten zien. Daardoor zijn programma's tegenwoordig een stuk gebruiksvriendelijker dan ponskaarten.

1.5 Binaire systeem

Een (gebruiker van een) programma vertelt een computer dus wat hij moet doen. Omdat computers werken op elektriciteit (netstroom of batterij), moeten al die gegevens op de een of andere manier worden omgezet in een elektrische stroom.



Hoe weet de computer wat er moet gebeuren als u de letter 'N' indrukt?

Iedere toets die wordt ingedrukt, genereert een combinatie van acht stroomstootjes. Zo'n stroomstootje kan van een hoge spanning zijn (voor het gemak geven we dat aan met een 1) of van een lage spanning (dat geven we aan met een 0). Zo'n stroomstootje noemen we een **bit**. Deze informatie-eenheid is een samentrekking van de Engelse woorden *binary digit*, in het Nederlands 'binair getal'. Een binair getal kan maar twee waarden aannemen: 0 en 1.

Een computer bevat enorm veel kleine schakelaars. Ook iedere schakelaar heeft maar twee mogelijkheden: stroom doorlaten of juist niet. Welke positie een schakelaar moet aannemen, bepaalt de binaire code.

Zouden we werken met maar één binair getal, dan zijn er dus slechts twee mogelijkheden: 0 en 1. Omdat elke mogelijke actie gedefinieerd moet worden, hebben we veel meer acties nodig. Daarom moeten we met meer binaire getallen werken:

- Werken we met twee bits, dan zijn er al vier mogelijkheden:

00	01	10	11
----	----	----	----

- Gaan we uit van drie bits, dan zijn er acht mogelijkheden:

000	001	010	011
100	101	110	111

Elke keer als we één binair getal extra toevoegen, nemen de mogelijkheden toe volgens een patroon:

1 binair getal = 2^1 mogelijkheden = $2 \times 1 = 2$ mogelijkheden

2 binaire getallen = 2^2 mogelijkheden = $2 \times 2 = 4$ mogelijkheden

3 binaire getallen = 2^3 mogelijkheden = $2 \times 2 \times 2 = 8$ mogelijkheden

(enzovoorts)

Computers werken met een combinatie van acht bits, ook wel een **byte** genoemd. Alle tekens en commando's worden uitgedrukt in bytes. Ieder teken of commando is één byte. De code die wordt doorgegeven, ziet er dus bijvoorbeeld zo uit:

10010101 11101001 10010110 10010010 10010010 00001000 01011001

Eenvoudig weergegeven is dit dus de 'taal' van de computer. Dit is voor mensen erg lastig te lezen. Dat komt doordat wij gewend zijn om te rekenen met het tientallig (decimale) stelsel, dat bestaat de tien cijfers 0 tot en met 9.

In het tientallig stelsel beginnen we bij 0 en hogen we dit getal telkens op met 1:

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

Na 9 doen we eigenlijk hetzelfde, alleen krijgt het getal nu een voorvoegsel. In de eerste ronde was het voorvoegsel 0 (maar dat laten we voor het gemak meestal weg), in de tweede ronde wordt dit voorvoegsel 1:

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19

In de binaire wereld werkt het eigenlijk op dezelfde manier, alleen moeten we dan veel sneller de voorvoegsels veranderen.

In de binaire wereld zijn de decimale getallen 0 en 1 het gemakkelijkst, want dat blijft gewoon 0 en 1:

Decimaal	Binair
0	0
1	1

Daarna wordt het wat lastiger. We beginnen, net als bij het decimale stelsel, bij 0 en hogen dat op met 1. Daarnaast werken we ook hier met het veranderen van voorvoegsels. De binaire getallen 2 en 3 zien er dus zo uit:

Decimaal	Binair
0	00
1	01
2	10
3	11

Zo blijft u extra getallen toevoegen, waardoor zoals gezegd de mogelijkheden blijven toenemen. Werkt u met vier getallen, dan heeft u ($2^4 =$) 16 mogelijkheden, die er zo uit zien:

Decimaal	Binair
0	0000 (of 0)
1	0001 (of 1)
2	0010 (of 10)
3	0011 (of 11)
4	0100 (of 100)
5	0101 (of 101)
6	0110 (of 100)
7	0111 (of 111)
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Snapt u het systeem?

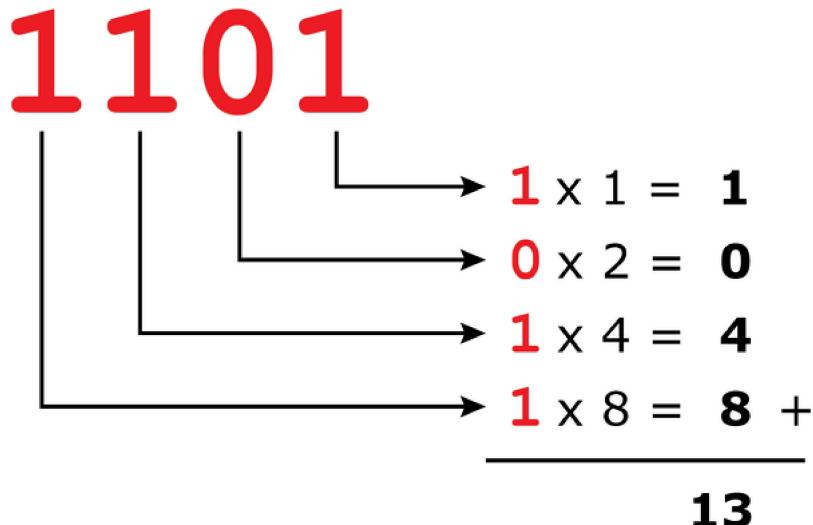
Wilt u een binair getal omrekenen naar een decimaal getal, dan zet u de volgende stappen:

1. Vermenigvuldig het meest rechtse cijfer met 1.

2. Vermenigvuldig het getal links daarvan met 2.
3. Vermenigvuldig het getal links daarvan met 4.
4. Vermenigvuldig het getal links daarvan met 8.

(enzovoorts)

Bijvoorbeeld:



Ga naar Plaza en bekijk een video met extra uitleg over het binaire systeem.

Als u programmeert, zult u gelukkig niet al te veel zien van het binaire systeem. Het speelt echter altijd op de achtergrond, dus het is goed om de basis ervan te begrijpen.

TIP! Het binaire stelsel is niet het enige stelsel dat een rol speelt bij computers. We kennen namelijk ook het hexadecimale stelsel, dat bestaat uit zestien cijfers. Een hexadecimaal getal ziet er bijvoorbeeld als volgt uit:
`#ff00aa`

Dit stelsel wordt vooral gebruikt bij de ontwikkeling van websites en grafische vormgeving. We laten dit stelsel in deze cursus verder buiten beschouwing.

1.6 Geschiedenis van programmeren

Omdat computers niet zonder programma's kunnen, zijn we al bezig met programmeren sinds we ons bezighouden met computers.

In deze paragraaf geven we een overzicht van de geschiedenis van het programmeren. Omdat dit een omvangrijke geschiedenis is met veel gebeurtenissen, beperken we ons tot de belangrijkste gebeurtenissen.

We beginnen uiteraard bij het begin. Het is interessant dat in een door mannen gedomineerde wereld van computers en programmeren, deskundigen het er wel over eens zijn dat het eerste computerprogramma geschreven werd door een vrouw, **Ada Lovelace**. Deze Britse wiskundige schreef een programma voor het rekenapparaat van Charles Babbage. Ze wordt daarom gezien als de eerste computerprogrammeur.

Ook **Conrad Zuse** schreef een taal voor zijn computers Z1 tot en met Z4. Die taal noemde hij *Plankalkül*.

P3.16

	$R(V)$	$\Rightarrow R$
V	0	0
S	$m \times \sigma$	$m \times \sigma$
	$W1(m)$	$\begin{bmatrix} V \Rightarrow R \\ 0 & 0 \\ i & m - 1 - i \\ \sigma & \sigma \end{bmatrix}$
V		
K		
S		

Voorbeeld van Plankalkül.

Omstreeks de jaren vijftig van de twintigste eeuw ontwikkelde een kleine groep experts Assembly Language. Deze taal staat nog altijd dichtbij de eerder behandelde nullen en enen.

```
C000          ORG      ROM+$0000 BEGIN MONITOR
C000 8E 00 70  START    LDS      #STACK
*****
* FUNCTION: INITA - Initialize ACIA
* INPUT: none
* OUTPUT: none
* CALLS: none
* DESTROYS: acc A
0013          RESETA   EQU      %00010011
0011          CTLREG   EQU      %00010001
C003 86 13    INITA    LDA A   #RESETA   RESET ACIA
C005 B7 80 04          STA A   ACIA
C008 86 11    LDA A   #CTLREG  SET 8 BITS AND 2 STOP
C00A B7 80 04          STA A   ACIA
C00D 7E C0 F1    JMP     SIGNON  GO TO START OF MONITOR
```

Voorbeeld van Assembly Language.

Vanaf het moment dat Assembly Language werd ontwikkeld, kunnen we computertalen onderverdelen in 'lagere' en 'hogere' talen:

- Met een **hogere taal** bedoelen we een taal die dichter bij de menselijk taal staat, waardoor deze gemakkelijker is voor de menselijke gebruiker. Het nadeel is dat programma's die zijn geschreven in zo'n taal, iets trager zijn, omdat ze eerst (via een of meer tussenstappen) vertaald moeten worden naar de nullen en enen, die voor de computer te begrijpen zijn.
- Een **lagere taal** staat juist 'dichter bij de computer'. Deze taal lijkt dus minder op de Engelse mensentaal en meer op iets wiskundigs of voor een leek onbegrijpelijs. Het vergt voor de programmeur dus meer kennis en moeite om de taal te beheersen en te gebruiken. Het grote voordeel is dat er minder vertaalslagen nodig zijn om het programma uiteindelijk door de computer te laten uitvoeren. Het programma is daardoor sneller uit te voeren. Games worden bijvoorbeeld vaak in een lagere programmeertaal geprogrammeerd.

Na de komst van Assembly Language volgden nog enkele talen:

- Fortran (1957):** Deze naam is een samenvoeging van het Engelse *formula translating*. De taal werd vooral gebruikt voor het oplossen van technische, natuurwetenschappelijke en econometrische problemen. Het was de eerste hogere programmeertaal die in algemeen gebruik kwam en is ook vandaag de dag nog van belang.
- Lisp (1958):** Deze hoge programmeertaal volgde snel na Fortran. Deze taal werd onder meer toegepast bij kunstmatige intelligentie (AI).

Als u de volgende voorbeeldcodes bekijkt, zult u zien dat deze talen nog altijd tamelijk cryptisch zijn voor niet-experts.

```

1  READ(5,501) IA,IB,IC
2  501 FORMAT(3I5)
3      IF(IA.EQ.0 .OR. IB.EQ.0 .OR. IC.EQ.0) STOP 1
4      S = (IA + IB + IC) / 2.0
5      AREA = SQRT( S * (S - IA) * (S - IB) * (S - IC) )
6      WRITE(6,601) IA,IB,IC,AREA
7  601 FORMAT(4H A= ,I5,5H B= ,I5,5H C= ,I5,8H AREA= ,F10.2,
8      $13H SQUARE UNITS)
9      STOP
10     END

```

Voorbeeld van Fortran.

```

1  (defvar *test-name* nil)
2
3  (defmacro deftest (name parameters &body body)
4      "Define a test function. Within a test function we can call
5      other test functions or use 'check' to run individual test
6      cases."
7      `(defun ,name ,parameters
8          (let ((*test-name* (append *test-name* (list ',name))))
9              ,@body)))
10
11 (defmacro check (&body forms)
12     "Run each expression in 'forms' as a test case."
13     `(combine-results
14         ,(loop for f in forms collect `(report-result ,f ',f))))
15

```

Voorbeeld van Lisp.

Toch zal iemand die de Engelse taal beheerst wel woorden in de code zien die enigszins hinten op wat dat specifieke commando zou kunnen doen.

Veel computertalen die daarna werden ontwikkeld, leken steeds meer op de reguliere Engelse taal:

- **COBOL** (1959): Deze naam is een samenvoeging van *Common Business Oriented Language*. Deze taal moest ten tijde van de ontwikkeling de gemakkelijkst te lezen, schrijven en onderhouden taal zijn. Vandaag de dag zijn er nog altijd computers die draaien op COBOL (ook op LISP en FORTRAN overigens).
- **BASIC** (1964): Dit was een taal die ook voor niet-wiskundigen te begrijpen was en op verschillende manieren ingezet kon worden. Dat is ook terug te zien aan de naam: *Beginners All-purpose Symbolic Instruction Code*.
- **PASCAL** (1970): Deze taal werd vernoemd naar de Franse wiskundige en natuurkundige Blaise Pascal. Hierbij ligt de nadruk op onder meer eenvoud en gestructureerd programmeren. Een bekende hedendaagse toepassing van PASCAL is de communicatiesoftware *Skype*.
- **C** (1972): Een taal die ook nu nog veel wordt gebruikt, vooral door gameontwikkelaars, is de taal C. De invloed van deze taal is dusdanig groot, dat de taal al verschillende afgeleiden heeft, zoals C++, C#, Java en Perl. Ook **Python**, de taal die wij in deze cursus vaak zullen gebruiken als praktisch voorbeeld, is afgeleid van C.

```

1  #include <stdio.h>
2  int main() {
3      // printf() displays the string inside quotation
4      printf("Hello, World!");
5      return 0;
6  }

```

Voorbeeld van C.

```
1  def Hello_10():
2      for i in range(10):
3          print("Hello World", i)
4
5  Hello_10()
```

Voorbeeld van Python.

- Jaren later werden scripttalen als **PHP** (1994), **JavaScript** (1995) en **Ruby** (1995) ontwikkeld vanwege (onder meer) de behoefte om webpagina's meer functionaliteit te geven dan gewoon statische pagina's met tekst en afbeeldingen. Wat een scripttaal is, leert u in de volgende les.
- **Scratch** (2002): Veel kinderen die willen programmeren, starten hiermee. De taal is geschikt voor het maken van visualisaties, zoals animaties en spellen.
- **Swift** (2014): Deze programmeertaal is ontwikkeld door *Apple* en wordt op de meeste *Apple*-producten gebruikt.

1.7 Samenvatting

Omdat wetenschappers en wiskundigen behoefte hadden aan een apparaat dat berekeningen voor hen uitvoerde, ontstond de computer. Dit apparaat werd in de negentiende eeuw voor het eerst ontwikkeld en wordt tot op de dag van vandaag gebruikt voor bijvoorbeeld het encoderen van geheime berichten, het bijhouden van administratie, communicatie en het streamen van muziek.

Een computer weet pas wat hij moet doen als hij daarvoor instructies krijgt. Deze instructies komen van een programma (al dan niet onder invloed van een gebruiker) en zijn geschreven in binaire code. Dit is een code die uitsluitend bestaat uit nullen en enen, bijvoorbeeld:

10010101 11101001 10010110 10010010 10010010 00001000 01011001

Omdat het binaire systeem weinig gebruiksvriendelijk is, zijn er veel verschillende programmeertalen ontwikkeld. Wat begon met de talen van Ada Lovelace en Conrad Zuse, leidde tot de programmeertalen die momenteel populair zijn, zoals C, JavaScript en Python.

1.8 Vraagmoment

Heeft u vragen over deze les? Dan kunt u die stellen via een vraagmoment.

U herkent een vraagmoment aan het spreekballonnetje dat op Plaza in het overzicht "Lessen" bij een les staat. Door op deze spreekballoon te klikken, komt u op een nieuwe pagina, waar u uw vraag in het teksvak kunt typen.



Het teksvak mag maximaal vijfhonderd tekens bevatten. Zorg er dus voor, dat u uw vraag/vragen bondig formuleert.

Nadat u op "Verstuur vraag" heeft geklikt, wordt uw vraag automatisch naar uw docent verstuurd. Zodra uw docent de vraag heeft beantwoord, verschijnt er op uw dashboard bij de opleiding een spreekballoon.

U kunt per les één keer vragen stellen. Verzamel dus uw vragen over de les, voordat u deze instuurt. Als u gebruik heeft gemaakt van het vraagmoment, verdwijnt het spreekballonnetje bij de les.

1.9 Oefenopgaven

De volgende opgaven werkt u als oefenopgaven voor uzelf uit. De uitwerkingen van deze opgaven kunt u meteen zelf controleren.

Wat is in deze cursus de definitie van het begrip 'programmeren'?

→ Bekijk antwoord

Verklaar de herkomst van het woord 'computer' en geef aan waarom voor deze benaming werd gekozen.

→ Bekijk antwoord

Welk bedrijf bracht de eerste computer uit onder de naam 'pc'?

→ Bekijk antwoord

Uit welke vier onderdelen bestaat het concept 'computer'?

→ Bekijk antwoord

Hoe zagen de eerste computerprogramma's eruit?

→ Bekijk antwoord

Wat is een bit?

→ Bekijk antwoord

Computers werken met een combinatie van acht bits, ook wel een **byte** genoemd. Hoeveel mogelijke combinaties zijn er, als er gewerkt wordt met acht getallen?

→ Bekijk antwoord

Schrijf het decimale getal '13' in binaire code.

→ Bekijk antwoord

Vertaal het binaire getal **1110** in een decimaal getal.

→ Bekijk antwoord

Wie wordt gezien als 's werelds eerste programmeur?

→ Bekijk antwoord

Naar wie is de programmeertaal Pascal vernoemd?

1.10 Huiswerkopgaven

Een groot deel van de lessen in deze cursus bevatten huiswerkopgaven. Deze opgaven testen niet alleen of u de lesstof goed begrepen heeft, maar ook of u de lesstof kunt toepassen. Uw uitwerkingen van deze opgaven kunt u via Plaza insturen naar uw docent. Deze docent beoordeelt uw werk en beantwoordt eventuele korte, gerichte vragen.

Deze les bevat geen huiswerkopgaven.