PROGRAMMEERTALEN (Deel I)

door W. F. H. Alleijn

1 Algemeen

Volgens een woordenboek is een programma onder meer een geschrift waarin de werkzaamheden worden opgesomd die volgens een bepaald plan zullen worden verricht.

Zo kan het plan bestaan om door mensen een bepaald produkt te laten maken. Teneinde tot het gewenste resultaat te kunnen komen, zal van tevoren nauwkeurig worden vastgesteld:

- de konstruktie en de vormgeving van het te fabriceren produkt (bijvoorbeeld in een tekening);
- de benodigde materialen;
- de bewerkingen die het materiaal achtereenvolgens moet ondergaan en de daarbij te hanteren gereedschappen.

Wanneer het gewenste produkt langs mechanische weg moet worden vervaardigd - waarbij de mens zoveel als mogelijk wordt uitgeschakeld - zal de toe te passen apparatuur daartoe bestuurd moeten worden. Dit zal gebeuren in de vorm van een reeks opdrachten, waardoor de door de machine te verrichten funkties op de juiste tijdstippen en in de juiste volgorde worden geaktiveerd.

Alleen wanneer de opdrachten op een voor de machine „begrijpelijke" wijze worden verstrekt, kan het gewenste resultaat worden bereikt.

Bij conventionele ponskaarten-apparatuur kunnen bijvoorbeeld opdrachten aan de administratie-machine worden gegeven door middel van een besturingspaneel. Door het aanbrengen van een bedrading op zo'n paneel wordt onder meer aangegeven waarmee wat moet worden gedaan en waarheen het resultaat moet worden gebracht.

Zo kan een boekhoudmachine worden bestuurd door middel van een lineaal waarin de uit te voeren bewerkingen zodanig zijn vastgelegd - bijvoorbeeld in de vorm van een nokkenpatroon - dat de machine deze zelfstandig uitvoert.

Bij een geautomatiseerd proces moet eveneens aan de machine, in een voor deze begrijpelijke taal, een reeks uit te voeren opdrachten worden verstrekt.

Van een te automatiseren projekt dient eerst een analyse te worden gemaakt. Een uitgewerkte analyse wordt vastgelegd in een blokschema. Zo'n schema vormt met de specifikatie van de in te voeren gegevens en de uit te voeren informatie de basis voor de programmering.

De programmeur begint veelal met een nadere detaillering van het vorengenoemde schema. Voor elk onderdeel wordt dan een detailblokschema of programmablokschema verkregen.

In deze detailblokschema's worden de door de computer achtereenvolgens uit te voeren bewerkingen zo gedetailleerd opgenomen dat aan de hand hiervan de opdrachten voor de machine kunnen worden geschreven.

Het opstellen van de reeks opdrachten voor het oplossen van een gegeven probleem door een computer-systeem wordt programmeren genoemd.

Alhoewel computer-systemen wel worden aangeduid met electronische breinen, zijn het in wezen „domme" apparaten, die alleen iets kunnen doen als tot in het kleinste detail opdrachten beschikbaar zijn die aangeven hoe en waarmee moet worden gewerkt.

2 Werkgeheugen

Alvorens verder te gaan met de programmering, eerst iets over het werkgeheugen. In de centrale verwerkingseenheid van een computer-systeem is een geheugen aanwezig, waarin onder meer gebieden kunnen worden gereserveerd voor:
- het vasthouden van invoergegevens (inleesgebied),
- het tijdelijk vasthouden van uit te voeren informatie (bijvoorbeeld printgebied),
- één of meerdere programma's,
- eventueel te raadplegen tabellen.

Mede kunnen bepaalde geheugenplaatsen worden bestemd voor:
- het vasthouden van tussenresultaten tijdens bewerkingen,
- het vasthouden van eindresultaten,
- eventueel te hanteren constanten.

De geheugenplaatsen worden aangegeven met nummers. Deze worden adressen genoemd.

In een later te publiceren artikel zal nog nader worden ingegaan op het gebruik van het werkgeheugen ten behoeve van de zogenaamde besturingsprogramma's („operating systems").

3 Instrukties

Een opdracht voor een computer bestaat uit:
- het opdracht-adres
- de opdracht-code
- één of meer operanden.

Het opdracht-adres geeft de plaats in het werkgeheugen aan waarin de opdracht moet worden geplaatst.

De opdracht-code, ook wel operatie-code genoemd, geeft aan wat er moet gebeuren; bijvoorbeeld optellen, transporteren, afdrukken e.d. De verschillende bewerkingen of operaties worden in vier groepen onderscheiden:
- invoer- en uitvoer-opdrachten
- transport-opdrachten
- rekenkundige opdrachten
- sprong-opdrachten.

Zoals de naam reeds aangeeft zijn de invoer-opdrachten gericht op het invoeren of „inlezen" van gegevens vanuit bijvoorbeeld ponskaarten, papieren ponsband, magneetband e.d. naar het daarvoor bestemde inleesgebied.

De uitvoer-opdrachten kunnen aangeven waar de informatie moet worden vastgelegd, bijvoorbeeld in ponskaarten, op een magneetband of afdrukken op papier vanuit een daarvoor bestemd uitleesgebied.

Met transport-opdrachten wordt aangegeven op welke wijze van welk adres, naar welk adres gegevens moeten worden gehaald respectievelijk worden gebracht; bijvoorbeeld van een adres in het inleesgebied naar een adres in het uitleesgebied.

Rekenkundige opdrachten geven aan wat met de gegevens moet gebeuren, zoals optellen, aftrekken, vermenigvuldigen en delen.

Sprong-opdrachten maken het mogelijk om op grond van bepaalde voorwaarden, in plaats van de opdrachten één voor één na elkaar te laten uitvoeren, bepaalde delen van het programma over te slaan en te springen naar een eerder of later in het programma voorkomende opdracht.

Bijvoorbeeld: als de uitkomst van een bewerking negatief is, ga dan niet verder met de volgende opdracht, maar met opdracht X.

Een operandum is het adresgedeelte van een opdracht, waarin wordt aangegeven op welke adressen in het werkgeheugen de uit te voeren opdracht betrekking heeft. Hierin staan namelijk de gegevens waarmede de opdracht moet worden uitgevoerd en/of naar welk adres het resultaat van een bewerking moet worden gebracht.

4 Interne machinecode

Het programmeren kon oorspronkelijk alleen gebeuren in de opdrachtencode van de machine waarmee het werk moest worden uitgevoerd.

Dit betekende dat het programma uitsluitend in de machinetaal aan de computer kon worden gegeven.

Deze taal is gebaseerd op het tweetallige stelsel, aangezien slechts gewerkt kan worden met het onderscheid tussen 0 en 1. Hieruit volgt, dat na het „inlezen" van een programma in het werkgeheugen, of het interne geheugen, de opdrachten in een tweetallige, of binaire, vorm zijn opgenomen. Iedere opdracht staat dan op een eigen adres (stored program).

Zo'n opdracht kan als volgt in een geheugen-adres staan:

1000111 0001010	0000101 0000011 1000100 0000101
opdrachtcode	adres (operandum)

Over het algemeen behoeft de programmeur zich niet te verdiepen in de interne machine-representatie van de opdracht. De programmeur schrijft namelijk de opdracht decimaal en/of alfabetisch. Bij de invoer van het programma worden de opdrachten automatisch in de machinecode omgezet, zoals dit ook met alle ingevoerde gegevens gebeurt.

De hiervoor als voorbeeld opgenomen opdracht wordt dan als volgt geschreven:

70	5345
opdrachtcode	adres

Aan het schrijven van een programma in de machinecode zijn nadelen verbonden:

- De programmeur dient bij voorkeur de opdrachtcode uit het hoofd te kennen. Dit is gewenst, omdat de gehanteerde getallen of symbolen - waaruit de code is samengesteld - over het algemeen te willekeurig zijn om aan te spreken.
- Hierbij komt, dat de geleerde code alleen kan worden gebruikt voor één type van een merk machine.
- De gebruikte adressen worden niet door het systeem vastgesteld en ook niet vastgelegd. Dit heeft tot gevolg dat de programmeur nauwkeurig de door hem gebruikte adressen moet aantekenen.
- De operanden (adressen) die bij een opdracht behoren worden aangegeven door codegetallen, die daardoor geen enkel verband met de inhoud van die adressen in het geheugen vertonen.

Als een voordeel kan worden genoemd dat het invoerprogramma, dat veelal reeds in het geheugen is opgenomen, zeer eenvoudig kan zijn; het behoeft slechts de opdrachten op de juiste adressen te plaatsen.

Deze computertaal staat ver van de menselijke taal af. Het gevolg is dan ook, dat bij een programma van enige omvang het samenstellen van opdrachten onoverzichtelijk wordt.

Dit vormde de aanleiding om te gaan zoeken naar programmeertalen die dichter bij de mens staan en daardoor niet alleen eenvoudiger, maar mede sneller en beter te gebruiken zijn.

5 Ontwikkeling van het programmeren

Een eerste belangrijke stap tot de vereenvoudiging van het schrijven van een programma vormde het hanteren van gemakkelijk in het geheugen liggende afkortingen en lettercombinaties voor de opdrachtcode.

Voorbeelden:

ADD	- *add*	PSK	- *p*ons een *k*aart
MPL	- *m*ulti*pl*y	VGL	- *verge*lijk
DIV	- *div*ide	OPA	- tel *op* in register *A*

Dit worden mnemotechnische of zelfverklarende opdrachtcodes genoemd. Van deze codes mag niet worden afgeweken.

Een volgende belangrijke stap vormde de mogelijkheid om symbolische adressen te gebruiken. Hierdoor konden in plaats van de nietszeggende absolute adressen begrijpbare afkortingen door de programmeur worden geschreven. Deze benamingen kunnen vrij worden gekozen.

Voorbeelden:

LOBEL	- loonbelasting
BRULO	- brutoloon
WERKN	- werknemer
FAKTU	- faktuur

De machine houdt zelf aantekening van het gebruik van deze adressen.

Hierdoor verviel de noodzaak om de gebruikte geheugenplaatsen te noteren, waarop de opdrachten werden geplaatst. De programmeur kon voortaan de opdrachten normale volgnummers geven.

Een verdergaande vereenvoudiging in de programmering ontstond door de toepassing van de relatieve adressering. Hierbij wordt het absolute adres verkregen door bij het adresgedeelte van de opdracht een constante waarde te tellen. Deze waarde is veelal gelijk aan het absolute adres van de eerste programma-opdracht. Dit biedt onder meer het voordeel dat meerdere programmeurs delen van een programma kunnen schrijven, die later samengevoegd worden. Het vertaalprogramma draagt zorg om de adressen onder een noemer te brengen.

De programmeertalen die na deze veranderingen ontstonden, worden assembleertalen genoemd.

Deze talen staan dicht bij de mens en ver van de machinetaal af. De machine kan de assembleertaal niet „begrijpen"; daarom dient de code vertaald te worden in de interne machinecode.

Daartoe werd het invoerprogramma uitgebreid, waardoor het mogelijk werd om de vereiste omzetting te bewerkstelligen. Zo'n omzettings- of vertaalprogramma wordt assembler genoemd.

Alhoewel hiermede in de wijze van programmeren aantrekkelijke verbeteringen waren verkregen, bleven toch nog problemen bestaan.

De assembleertalen waren machine-gericht. Elk type van een merk machine had een eigen assembleertaal.

De computerleveranciers probeerden ook hierin vereenvoudiging te brengen door de ontwikkeling van talen die gericht waren op een lijn of familie van machines. Hierdoor ontstonden de leveranciersgerichte codes.

De ontwikkeling ging verder, aangezien de assembleertalen het nodig bleven maken om iedere elementaire opdracht te schrijven, die dan moest worden omgezet in één opdracht in de machinecode.

Het was daardoor een zogenaamde één-om-één-vertaler.

In een volgende fase werd de mogelijkheid ontwikkeld om te werken met macro- en pseudo-opdrachten. Uit één opdracht worden na de vertaling meerdere machine-opdrachten verkregen. Dit gebeurt onder meer bij worteltrekken en vermenigvuldigen met drijvende komma bij berekeningen met decimalen.

De uitbreidingen en verfijningen van de assembleertalen maakten het programma eenvoudiger. Hierdoor konden sneller „goede" programma's worden gemaakt. Het bezwaar bleef dat de autocodes nog machinegebonden waren.

De ontwikkeling is verder in de richting gegaan van programmeertalen, waarbij zowel uit één geschreven opdracht meerdere machineopdrachten worden samengesteld, en waarbij tevens de taal voor diverse typen en merken computers kan worden gebruikt.

6 Universele programmeertalen

De universele programmeertalen - ook wel algemene programmeertalen genoemd - zijn niet langer machine-gerichte talen, maar probleem-gerichte en procedure-gerichte talen.

Ook hierbij dient het geschreven programma te worden vertaald in de interne machinecode.

Deze talen kunnen worden gebruikt voor ieder type computer, mits een vertaalprogramma daarbij ter beschikking staat. Zo'n vertaalprogramma wordt compiler genoemd.

Er zijn reeds vele algemene programmeertalen ontwikkeld.

Als probleem-gerichte talen zijn onder meer te noemen:

LISP	- *LIS*t *P*rocessing
SIMSCRIPT	- een taal gericht op omvangrijke simulatieproblematiek
SIMULA	- *SIMU*lation *LA*nguage

terwijl procedure-gerichte talen onder meer zijn:

COBOL	- *CO*mmon *B*usiness *O*riented *L*anguage
ALGOL	- *ALGO*rithmic *L*anguage
FORTRAN	- *FOR*mula *TRAN*slation
PL/1	- *P*rogram *L*anguage One

COBOL is hoofdzakelijk bestemd voor administratief-technische toepassingen, terwijl ALGOL en FORTRAN bestemd zijn voor technisch-wetenschappelijke toepassingen. PL/1 kan zowel voor administratief-technische als voor technisch-wetenschappelijke toepassingen worden gehanteerd.

Om programma's in deze universele programmeertalen te kunnen schrijven dient de programmeur een aantal regels en voorschriften te leren. Deze regels zijn uitsluitend gericht op de uit te voeren bewerkingen en de daarbij te gebruiken gegevens.

Kennis van de specifieke eigenschappen van de toe te passen apparatuur is slechts in beperkte mate nodig.

De universele programmeertalen vereisen veelal een minimale samenstelling en grootte van het computer-systeem.

Een nauwkeurige en gedetailleerde analyse van het probleem blijft echter steeds de basis voor het programma vormen. Soms is het computergebruik bij de toepassing van universele programmeertalen minder efficiënt dan bij het hanteren van machine-gerichte talen, aangezien de verwerkingstijden langer kunnen zijn.

Ook kan een groter geheugenbeslag het gevolg zijn.

COBOL

Teneinde een beeld te krijgen van een algemene programmeertaal, is het COBOL-programmeersysteem als model genomen.

De COBOL-taal ligt dicht bij de normale Engelse taal. Zo'n geschreven programma kan veelal worden gelezen alsof het een beschrijving van de werkwijze is.

Het programma bestaat uit vier gedeelten - divisions -, te weten:

- *de Identification Division*
  Hierin worden onder meer vermeld de naam van het programma, de naam van de programmeur en de datum waarop het programma is gereed gekomen.
- *de Environment Division*
  Deze geeft onder meer de te gebruiken apparatuur aan, waarmede het geschreven programma - source program - moet worden omgezet in de interne machinecode.
  Tevens wordt de samenstelling genoteerd van de machine waarmede het verkregen programma in de machinecode - object program - moet worden uitgevoerd.
- *de Data Division*
  Deze bevat een gedetailleerde beschrijving en indeling van de benodigde bestanden.
- *de Procedure Division*
  Hierin wordt beschreven welke bewerkingen de gegevens moeten ondergaan.

In de eerste drie divisions worden alle begrippen en grootheden nauwkeurig gedefinieerd en benoemd, waarmede in de vierde division moet worden gewerkt.

In de procedure division worden alle stappen genoemd om een gegeven probleem op te lossen. Deze stappen - procedures - worden in zinnen geschreven, waarbij de bewerkingen in werkwoorden - verbs - worden genoteerd. Een zin wordt afgesloten met een punt.

Voorbeeld:

MULTIPLY uren *BY* tarief *GIVING* bedrag.

Multiply is een verb dat door het vertaalprogramma - compiler - wordt herkend en wordt omgezet in het codegetal van de desbetreffende opdrachtcode.

Er zijn ook keuze-mogelijkheden.

Voorbeelden:

IF . . . . . *.IS*	Less than	. . . . .
IF . . . . . *.IS*	Equal to	. . . . .
↑	↑	↑
naam van het eerste te vergelijken gegeven	naam van het tweede te vergelijken gegeven	verwijzing naar de volgende uit te voeren stap (next statement)

De programmeur heeft de beschikking over een paar honderd woorden. Deze zogenaamde gereserveerde woorden worden veelal onderstreept gebruikt op plaatsen in de zinnen, volgens bepaalde regels. Dit mag dus niet willekeurig gebeuren.

7 Report program generator

Een Report Program Generator - veelal afgekort tot R.P.G. - is een hulpmiddel om op eenvoudige wijze computerprogramma's te vervaardigen. Aanvankelijk was R.P.G. gericht op het snel schrijven van programma's voor het afdrukken van lijsten. Thans biedt R.P.G. veel verdergaande mogelijkheden.

De gegevens worden genoteerd op een set formulieren, te weten:
1 - de omschrijving van te hanteren bestand(en)
2 - de specifikatie van de in te voeren gegevens
3 - de specifikatie van de bewerkingen
4 - de specifikatie van de te verstrekken informatie
5 - de specifikatie van eventueel te gebruiken tabellen.

Wanneer de gegevens van de genoemde formulieren bijvoorbeeld zijn verponst, worden de verkregen kaarten ingelezen. In het werkgeheugen van het computersysteem is de R.P.G. - als het ware een vertaalprogramma - opgenomen, waarmede automatisch het machine-programma wordt samengesteld.

R.P.G. vereenvoudigt niet alleen het schrijven van programma's, maar biedt tevens het voordeel dat het verkregen programma de volledige capaciteit van het computer-systeem benut. Hierbij komt dat wijzigingen op eenvoudige wijze kunnen worden aangebracht.

8 Applicatie-pakketten e.d.

Zowel door de leveranciers als door instituten die onder meer programma's vervaardigen - software houses - worden programma-pakkettten aangeboden die gericht zijn op bepaalde toepassingen.

Zo bestaan er standaardprogramma's voor onder meer:
- fakturering
- debiteuren-administratie
- voorraadbeheersingssystemen.

Veelal worden dergelijke programma's gecombineerd tot pakketten. Verder zijn er programma's voor het verwerken van technieken, zoals:
- operational research technieken
- netwerk-planning.

Voor computer-verwerkingen zijn eveneens programma's beschikbaar voor:
- sorteren
- converteren.

Indien deze programma's kunnen worden gebruikt in de toepassingen van het desbetreffende bedrijf, biedt dit het voordeel dat geen eigen programma's behoeven te worden ontwikkeld. Dit kan uiteraard beperkingen inhouden.

Soms worden daarom standaardprogramma's aangepast aan de eigen behoeften.

Thans zijn programma's ontwikkeld die door de accountant kunnen worden gehanteerd ten behoeve van zijn controlerende funktie, de z.g. audit

systems. Hieraan zal op een later tijdstip nog afzonderlijk aandacht worden besteed.

In een volgend artikel zullen nog andere vormen van programmeren worden behandeld, terwijl tevens een programma-voorbeeld in verschillende talen wordt opgenomen.