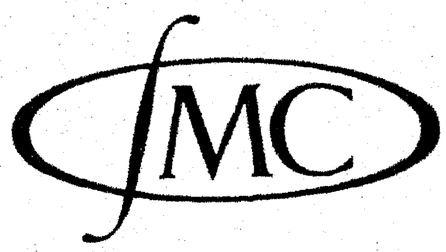
STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

DR 12$_r$

Het gebruik van automatische rekenmachines.

A. van Wijngaarden.



1953

Overdruk uit
Faraday

HET GEBRUIK VAN AUTOMATISCHE REKENMACHINES

DOOR PROF. DR. IR. A. VAN WIJNGAARDEN

Er is een zekere overeenkomst tussen de taak van de leraar voor de klas en de gebruiker van een automatische rekenmachine. Beiden bevinden zich voor een collectief, in staat en bereid om zekere prestaties te leveren mits — en hier schuilt de kunst — men het op de juiste manier toespreekt.

Natuurlijk moet men een weinig weten van de opbouw van een automatische rekenmachine om de taal waarop zij zal reageren te verstaan. Zo'n rekenmachine bestaat in hoofdzaak uit vijf min of meer duidelijk te onderscheiden organen, nl. het rekenorgaan, de besturing, het geheugen, de invoer en de uitvoer. Het rekenorgaan is in staat een aantal, heel eenvoudige, rekenkundige bewerkingen uit te oefenen, zoals optellen, aftrekken, vermenigvuldigen, eventueel ook delen. De besturing is het orgaan dat het rekenorgaan dwingt de juiste bewerkingen op de juiste getallen uit te oefenen. Daarbij maakt zij gebruik van het geheugen, waaruit zij de getallen haalt, waarmede gerekend moet worden en waarin zij de resultaten ook weer opbergt. Bovendien moet de besturing zelf ook te weten komen wat het moet doen. Daartoe haalt zij ook uit het geheugen zg. opdrachten, die haar vertellen welke bewerkingen op welke getallen moeten worden uitgevoerd. De invoer en uitvoer tenslotte zijn twee organen ter communicatie met de buitenwereld, bijv. een bandlezer, in staat om rijen gaatjes in een geperforeerde band te „lezen" en een electrische schrijfmachine in staat om door de machine toegevoerde cijfers te typen.

Het geheugen kan allerhande vormen hebben, afhankelijk van de gewenste snelheid van de machine. Een eenvoudige en in haar werking gemakkelijk te begrijpen uitvoering is de zg. magnetische trommel. Dit is een cylinder bedekt met een laagje magnetiseerbaar materiaal, bijv. ijzeroxyde. De trommel draait snel om haar as. Op korte afstand van de trommel bevinden zich een aantal zg. „kopjes", precies zoals gebruikelijk bij de magnetophoon in de geluidsweergavetechniek. Zo'n kopje bestaat uit een klein ijzeren ringetje, voorzien van een doorsnijding in radiale richting juist daar waar het vlak bij de cylinder is. Om de kern bevindt zich een aantal windingen. De besturing kan nu door deze windingen een zeer korte stroomstoot sturen, enkele microseconden lang.Er ontstaat dan een magnetische flux in de kern, maar deze heeft geen neiging om door het luchtspleetje te gaan, maar prefereert i. p. daarvan de gemakkelijker omweg via het buitenlaagje van de trommel, dat zich vlak bij het kopje bevindt. Het resultaat is, dat een gemagnetiseerd gebiedje op de trommel achterblijft. Aangezien de stroomstoot in twee richtingen door de wikkeling kan worden gestuurd, heeft ook de flux twee mogelijke richtingen en daarmede het magneetveldje op de trommel. Aan deze twee mogelijkheden kennen wij de cijferwaarde nul resp. één toe. Omgekeerd veroorzaakt het magneetveldje, wanneer het de kop bij het omwentelen passeert op de uiteinden van de wikkeling een zwakke spanning, die na versterking verraadt of er een nul dan wel een één voorbij is geraasd. Op deze wijze is de besturing in staat in het geheugen te schrijven en te lezen. Natuurlijk moet de besturing ook weten, waar zij schrijft, en de plaats waar zij wil schrijven is a.h.w.

niet direct beschikbaar. De machine moet dan even wachten totdat de gewenste plaats passeert voorbij een kopje. Als de trommel 50 omwentelingen per seconde maakt, dan is de minimum wachttijd 0 en de maximum wachttijd 20 msec., dus gemiddeld 10 msec. In moderne ogen is dat lang. Op het ogenblik is het mogelijk geheugens te maken die ongeveer duizend maal zo snel zijn.

Het is geen bezwaar, dat er slechts twee cijfers, 0 en 1, kunnen worden geschreven, want vrijwel alle machines bedienen zich, al dan niet verkapt, van het tweetallig stelsel. Natuurlijk bedient de machine zich niet van losse cijfers, maar van getallen van bijv. 30 cijfers lang (dat is dus ongeveer vergelijkbaar met getallen van 9 cijfers in het tientallig stelsel). Daartoe is het geheugen verdeeld in een aantal, bijv. 1024 „adressen", op elk waarvan een heel „woord" van 30 cijfers geschreven kan worden. Zo'n adres wordt met een nummer aangegeven, dus bijv. adres 0, adres 1, . . . adres 1023. Met opzet hebben wij het neutrale woord „woord" gebruikt i.p.v. „getal", omdat immers ook de opdrachten in het geheugen moeten worden geborgen. Deze worden nu ook op adressen geborgen, dus een woord behoeft niet een getal te zijn, het kan ook een opdracht zijn. Overigens is ook een opdracht slechts een rij van nullen en enen en is dus evengoed als een getal te interpreteren als men dat wil, en ook omgekeerd is een getal (veelal) als opdracht te interpreteren. Hoe vindt de machine nu uit, wat onze bedoeling is? Om dit in te zien, moeten we het begrip „programma" invoeren en nader bezien hoe de besturing in grote trekken werkt.

Iedere opdracht dan bestaat uit twee gedeelten, nl. een functiegedeelte en een adresgedeelte, twee getallen achter elkaar geschreven, die tezamen een woord vormen. Het functiegedeelte is een getal dat een bewerking specificeert die uitgevoerd moet worden op het getal dat zich bevindt op het adres, gespecificeerd in het adresgedeelte.

Schematisch kan men dus een opdracht zien als een getallenpaar *F*, *n*, te interpreteren als: „Doe operatie *F* op het getal op adres *n*". Bijv. zouden enkele opdrachten kunnen luiden:

0, *n* Tel het getal op adres *n* op bij wat zich reeds in het rekenorgaan bevindt.

1, *n* Trek het getal op adres *n* af van wat zich reeds in het rekenorgaan bevindt.

2, *n* Vermenigvuldig het getal op adres *n* met dat wat zich reeds in het rekenorgaan bevindt en plaats het product in het rekenorgaan.

3, *n* Lees het cijfer, dat voorgesteld wordt door het rijtje gaatjes in de band, dat op het ogenblik onder de bandlezer ligt en schrijf het op adres *n*. Schuif de band daarna een plaats verder op.

4, *n* Lees het „karakter" dat voorgesteld wordt door de laatste vier cijfers van de inhoud van adres *n* (bijv. 0000 = 0, 0001 = 1, 0010 = 2, 0011 = 3, ..., 1001 = 9, 1010 = +, 1011 = —, 1100 = ., 1101 = spatie, 1110 = tabuleer, 1111 = wagen terug, nieuwe regel) en typ dat uit op de schrijfmachine.

5, *n* Schrijf het getal, dat zich in het rekenorgaan bevindt, neer op adres *n* en maak vervolgens het rekenorgaan schoon.

De besturing werkt nu steeds in twee stappen.

Stap 1: Zij leest het woord op adres *m* en interpreteert dit als een opdracht. Deze vertelt haar, dat zij operatie *F* moet uitvoeren op het getal op adres *n*. Dan doet zij stap 2.
Stap 2: Zij leest het woord op adres *n* en voert er operatie *F* op uit. Tevens telt zij bij getal *m* één op. Dan doet zij stap 1.

Men ziet, dat stap 1 nu niet dezelfde opdracht leest als zoeven, maar die, welke zich op het volgende adres bevindt, enz. De machine werkt dus een aantal opdrachten na elkaar af, d.w.z. zij voert een programma uit. Overigens zou het, zo het hier bij bleef, niet veel zin hebben bij zijn berekeningen een automatische rekenmachine te gebruiken. Immers zo men voor iedere elementaire berekening een opdracht uit moest schrijven, kon men sneller zelf de berekening direct uitvoeren en bovendien zou de machine in zeer korte tijd zijn gehele geheugenvoorraad aan opdrachten uitgeput hebben. Er moet een mogelijkheid zijn om cycli opdrachten te herhalen. Dit kan geschieden met een opdracht van een ander type, bijv.

6, *n* De volgende opdracht staat op adres *n*.

Nu kan men inderdaad een gedeelte van het programma laten herhalen door onderaan een reeks opdrachten een 6-opdracht te plaatsen, die de besturing terugverwijst naar het begin van die rij. Maar als men dat zonder meer doet heeft men een vicieuse cirkel gesloten, want de machine zal deze rij dan steeds opnieuw herhalen! Blijkbaar moet nog een element van onderscheiding aanwezig zijn, bijv. tot uiting gebracht in

7, *n* Als aan de conditie voldaan is, dan staat de volgende opdracht op adres *n*; zo niet dan als normaal op het adres volgend op dat van deze opdracht.

Wat die conditie precies is, is een zaak van minder belang. Meestal is dat het teken van een of ander getal, bijv. van het getal, dat zich op dat moment in het rekenorgaan bevindt. De opdracht 7, *n* wordt dan „gehoorzaamd" als het rekenorgaan een positief getal of nul bevat, dan wel „genegeerd" als het een negatief betal bevat. Wil men dan een bepaalde reeks opdrachten vijf maal uitvoeren alvorens met een nieuw gedeelte van het programma te beginnen, dan lasse men aan het eerste programmagedeelte een telling vast, waarvan het resultaat negatief is, zolang nog herhaald moet worden, terwijl het nul is als het vereiste aantal herhalingen is geschied. Een voorbeeld van zo'n programma luidt:

```
100   1   300                          300   5
101   7   200                          301   1
102   0   301                          302   ...
103   5   302
104       ...        Programma,
...       ...        dat een aantal
197       ...        keren herhaald moet worden
198   0   302
199   6   101
200   Volgend programma.
```

Men neme aan, dat het rekenorgaan bij de aanvang van dit gedeelte van het programma, d. i. bij de opdracht op adres 100, een nul bevat. Opdracht 100 plaatst dan — 5 in het rekenorgaan. Opdracht 101 wordt genegeerd, want — 5 is negatief. Opdracht 102 telt bij — 5 een 1 op, resultaat — 4 en opdracht 103 bergt deze — 4 op het adres 302 weg. De opdrachten 104 tot en met 197 mogen dan het vijf maal uit te voeren programmagedeelte zijn. Dit late weer een nul achter in het rekenorgaan. Opdracht 198 plaatst nu de opgeborgen — 4 in het rekenorgaan en de opdracht 199 zorgt dat de besturing „terugspringt" naar adres 101. Deze opdracht wordt weer genegeerd want ook — 4 is negatief. Opdracht 102 maakt er van — 3 en dit wordt opgeborgen. Dit spelletje herhaalt zich tot het programma vijf maal is doorlopen. Dan was immers nul weggeborgen op 302 en nu wordt 101 wel gehoorzaamd, d.w.z. de besturing springt naar 200 en een ander gedeelte van het programma komt aan bod. Wij merken hierbij op, dat de 5 op adres 300 bij dit programma niet bedorven is. Als dus verder op in de berekening de besturing weer eens bij opdracht 100 arriveert, gaat het spelletje precies eender gebeuren, behalve natuurlijk als een ander gedeelte van het programma intussen het getal 5 op adres 300 door een ander getal heeft vervangen. Men ziet ook, dat wanneer zich op adres 300 een nul had bevonden, het programma inderdaad nul maal uitgevoerd zou zijn, want dan werd opdracht 101 direct gehoorzaamd! In het algemeen zal het getal op adres 300 zelf weer de uitkomst van een ander deel van de berekening zijn, zodat men als men niet precies alles zelf na wenst te gaan rekenen van te voren wenig idee heeft wat de machine precies zal uitvoeren.

Trouwens, als men het zelf niet narekent, heeft men ook geen idee, wat de getallen zelf zijn, waarmede de machine te maken krijgt. Het genoemde stelsel van opdrachten is het best te vergelijken met algebra. In plaat van letters x, y, z enz. gebruikt men x_{300}, x_{301}, x_{302} en laat nu omdat de x zelf overbodig is haar maar weg en spreekt over 300, 301, 302, daarbij niet die getallen zelf bedoelend, maar de onbekenden met dat rangnummer, te vinden op adressen 300, 301, 302. Natuurlijk hebben die variabelen gedurende het echte rekenproces slechts een numerieke waarde, maar het programma zelf is evengoed algebra als ieder vraagstuk met x, y en z.

Het voorbeeld van de herhaalde berekening diende om aan te tonen, dat de programmeur zich het leven wat gemakkelijker kan maken door een bekend stuk programma meerdere malen te laten doorlopen. Men noemt zoiets een routine of subroutine, al naar gelang van de aard. Een volgend voorbeeld van zulke efficientie levert de echte of zg. gesloten subroutine. Stel men heeft een stukje programma gemaakt, dat van het getal op adres 1000 de wortel trekt en deze aflevert op adres 1001. Nu hebbe men in zijn programma meerdere malen een wortel te trekken. Men zou nu iedere maal dit stuk hulpprogramma in kunnen lassen, maar dat is natuurlijk niet sierlijk. Stel men heeft het eerst bij opdracht 150 behoefte aan die worteltrekking en stel, dat het hulpprogramma op adres 800 begint. Dan zou men de opdracht 6, 800 kunnen geven, waarna de besturing overging op het hulpprogramma, maar nu moet na afloop daarvan weer doorgegaan worden met opdracht 151. Dit zou kunnen door onder aan het hulpprogramma de opdracht 6, 151 te hebben. Wil men echter bij opdracht 200 weer het hulpprogramma aanroepen, dan kan dit natuurlijk met de opdracht

6, 800, maar nu willen we dit keer, dat de berekening doorgaat op 201 en niet, zoals automatisch zou geschieden, weer op 151! Een voorbeeld hoe men zo iets aanpakt is:

Hoofdprogramma:			Subroutine:			
149	...		800	0	824	
150	0	150	801	5	823	
151	6	800	802	...		
152	...		...	...		Eigenlijk
...	...		...	...		programma van
199	...		...	...		de subroutine
200	0	200	823	(		)
201	6	800	824	6	2	
202	...					

Op opdracht 150 wil men de worteltreksubroutine aanroepen. Het te behandelen getal is al op adres 1000 geplaatst en het rekenorgaan bevat nul. Opdracht 150 luidt nu 0,150, d.w.z. „tel in het rekenorgaan op dat wat zich bevindt op adres 150". Maar dat is de opdracht zelf! De besturing heeft hier helemaal geen weet van, zoekt weer in het geheugen, pakt het „getal" dat zich bevindt op adres 150 en telt het op in het rekenorgaan. Dit bevat nu dus 0,150 als getal gelezen. Als men aanneemt dat de opdracht F, *n* als getal gelezen er bijv. uitziet als 1024F + *n*, dan is onze opdracht als getal gelezen dus eenvoudig 150. Opdracht 141 zegt: „ga naar 800, het begin van de subroutine". Deze opdracht 800 telt in het rekenorgaan er bij op dat wat op adres 824 staat, dus de „opdracht" 6,2. Het resultaat is nu, dat het rekenorgaan een getal bevat, dat als opdracht gelezen luidt: 6,152. Opdracht 801 plaatst dit op 823. Nu volgt het eigenlijke worteltrekken en uiteindelijk komt opdracht 823 aan bod. Deze is juist ingevuld! Er staat nu 6,152 en de besturing springt terug naat 152. Heeft het hoofdprogramma de subroutine weer nodig, zoals bij opdracht 200, dan ziet men dat het zelfde spelletje de besturing nu terug laat keren bij opdracht 202.

Men ziet aan dit voorbeeld, dat naarmate men meer subroutines ter beschikking heeft, de taal waarin men tot de machine kan spreken steeds kernachtiger kan zijn. Bij iedere machine behoort dan ook een zich steeds uitbreidende bibliotheek van routines en subroutines.

Als men nu een volledig programma van een berekening op papier heeft gemaakt, hoe moet men de machine dan duidelijk maken, dat men dit wenst uitgevoerd te hebben? Aangezien de enige opdracht (in ons stelsel) die de machine in staat stelt informatie uit de buitenwereld op te nemen, de opdracht 3, *n* is, moet het programma blijkbaar volgens een of andere conventie op een band geponst worden. Men maakt de band en legt hem in de bandlezer. Maar dan gebeurt er nog niets. Waarom zou er? Er is niets, dat de machine nu plotseling zou gaan dwingen, de band te lezen, want de machine gehoorzaamt programma's in het geheugen. Blijkbaar moet er in het geheugen al een ander programma aanwezig zijn, een zg. invoerprogramma, dat een band met een programma erop in het geheugen binnenloodst. Er zijn dan nog twee logische moeilijkheden. In de eerste plaats, hoe kan men de machine dwingen dit invoerprogramma uit te gaan voeren? Nu, daarvoor zijn enkele knopjes op de machine aanwezig, die de besturing dwingen een programma te starten althans op een enkele plaats. Als dit dan

de eerste opdracht van het invoerprogramma is, dan is deze moeilijkheid opgeheven. De tweede vraag is natuurlijk: Hoe komt dat invoerprogramma dan in het geheugen? Nu, dat is een technische zaak. Men brengt het er met behulp van andere middelen in, dan de programmeur ten dienste staan en in principe behoeft dit slechts één maal te gebeuren.

Na afloop van het Congres maakte een vijftiental deelnemers (mathematici) een dankbaar gebruik van de door Prof. VAN WIJNGAARDEN geboden gelegenheid de grote rekenmachine van het Mathematisch Centrum te bezichtigen en in werking te zien.