Bouw en gebruik van automatische rekenmachines op de Rekenafdeling van het Mathematisch Centrum

HT de Beer
huub@heerdebeer.org
https://heerdebeer.org

Amsterdam, 26 februari 2008

Inhoudsopgave

1 Inleiding

In de jaren '50 van de vorige eeuw werden op de Rekenafdeling van het Mathematisch Centrum in Amsterdam verschillende grote automatische rekenmachines gebouwd. Deze apparaten waren bedoeld voor geavanceerd technisch-wetenschappelijk rekenwerk dat door de Rekenafdeling werd verricht. In het begin van de jaren '50 werd het grootste deel van de rekenopdrachten verricht door meisjes met tafelrekenmachines.

In 1950 werd de eerste automatische rekenmachine gebouwd en in gebruik genomen, de ARRA. Deze machine had, buiten het testen van de machine, weinig bijgedragen aan het rekenwerk op de Rekenafdeling. Gaandeweg de jaren '50 groeide het aantal rekenopdrachten dat met een automatische rekenmachine werd uitgevoerd gestaag, ongeveer de helft van de rekenopdrachten werd echter door de rekenaarsters uitgevoerd. Rond 1960 nam de automatische rekenmachine definitief het werk over van de rekenaarsters en veranderde de Rekenafdeling van aanzicht.

Dit had twee oorzaken. Aan de ene kant verdween de machinebouwactiviteit van de Rekenafdeling naar de nieuwe firma Electrologica die een zeer volwassen computer, de X1, op de markt bracht. Deze machine was het sluitstuk van de ontwikkeling van de eigen computerbouw op de Rekenafdeling en de X1 was

dan ook uitermate geschikt voor het werk op de Rekenafdeling. De machine voorzag in de rekenbehoefte van de Rekenafdeling. Tegelijkertijd verlieten de meeste rekenaarsters de Rekenafdeling en het Mathematisch Centrum, vaak na hun huwelijk, en de functie van rekenaarster verdween op de Rekenafdeling.

Over de bouw van computers op de Rekenafdeling is veel geschreven, deze grote intrigerende machines zijn nu eenmaal monumenten in de geschiedenis van de informatica in Nederland. Maar juist door deze monumentale status zijn de verhalen over deze machines voornamelijk verhalen over monumenten. Wat ontbreekt zijn verhalen over het gebruik van de machines en over de wisselwerking van de machines met hun omgeving.

Sporen van computergebruik zijn niet eenvoudig te vinden. De machines bestaan niet meer en de meeste programma's die er voor geschreven waren, zijn verdwenen. Wat rest zijn technische rapporten van de Rekenafdeling over de verschillende computers en het gebruik ervan. Er waren vier verschillende soorten rapporten. Allereerst werd van elke machine een of enkele technische beschrijvingen gemaakt over hoe de machine werkte of zou moeten werken. Daarnaast waren er een aantal rapporten geschreven over speciale "superprogramma's", bijvoorbeeld over programma's die de invoer en uitvoer regelden of programma's die het rekenen met een drijvende komma mogelijk maakten. Verder werden er twee cursussen *Programmeren voor automatische rekenmachines* gehouden, een in 1955/56 en een in november 1957. Tenslotte werden over alle rekenopdrachten die door de Rekenafdeling zijn verricht een rapport geschreven. Naast deze technische rapporten werd ook in de jaarverlagen van het Mathematisch Centrum over de Rekenafdeling bericht.

Met behulp van deze rapporten en jaarverslagen is het niet mogelijk de dagelijkse gang van zaken op de Rekenafdeling te beschrijven, daarvoor zijn deze documenten te formeel: het waren handboeken, naslagwerken of jaarlijkse besprekingen over het reilen en zeilen van het Mathematisch Centrum. Met behulp van de jaarverslagen is het echter mogelijk om enig inzicht te verkrijgen in de ontwikkeling van het computergebruik op de Rekenafdeling. Deze ontwikkeling wordt in de volgende paragraaf besproken.

Omdat er van elke machine documentatie is overgebleven, kan de ontwikkeling in computerbouw en computergebruik die de Rekenafdeling doormaakte gekarakteriseerd worden. Het is mogelijk om in algemene termen te spreken over hoe een machine gebruikt kon worden en hoe er geprogrammeerd werd. Deze karakterisatie gebeurt op twee niveaus. Allereerst komt de ontwikkeling van de machines en het gebruik ervan aan bod. Daarna is de beurt aan de ontwikkeling in het denken over programma's en het programmeren. Tenslotte, in de conclusie, wordt een synthese gemaakt over de bouw en het gebruik van automatische rekenmachines op de Rekenafdeling.

2 De ontwikkeling van het computergebruik op de Rekenafdeling

Aan de hand van de jaarverslagen van het Mathematisch Centrum kan voor een deel de ontwikkeling van het computergebruik op de Rekenafdeling geschetst worden. Naast een algemene beschrijving van de werkzaamheden van de Rekenafdeling van dat jaar werd van alle uitgevoerde opdrachten ook een beschrij-

ving gegeven. In deze korte opdrachtbeschrijving werd ook vermeld of er een automatische rekenmachine gebruikt was om de opdracht uit te voeren. Met andere woorden, het totale aantal rekenopdrachten per jaar is bekend en ook het aantal opdrachten waarbij een computer is gebruikt is bekend. In Tabel 1 zijn deze gegevens weergegeven.

Het aantal rekenopdrachten dat door de Rekenafdeling werd uitgevoerd in de jaren 1946 tot en met 1948 is onbekend, het is onduidelijk of er berhaupt opdrachten uitgevoerd zijn. Vanaf 1949 werden de opdrachten netjes opgesomd met opdrachtnummer en een korte beschrijving. Vanaf 1961 veranderde de verslaglegging van de uitgevoerde opdrachten: ze werden nog wel opgesomd, maar er werd geen extra informatie gegeven naast het opdrachtnummer. De cijfers in de tabel moeten vanaf dat jaar dan ook anders geïnterpreteerd worden.

In de jaren voor 1961 vond een fundamentele verandering plaats op de Rekenafdeling: de bouw van computers werd in 1958 volledig afgestoten en in 1959 werd de nieuwe computer, de Electrologica X1, in gebruik genomen. Het aantal rekenopdrachten dat de Rekenafdeling uitvoerde bleef lang schommelen rond de 55 à 60 opdrachten per jaar. Pas na de ingebruikneming van de X1 verdubbelde dat aantal.

Steeds minder rekenopdrachten werden met behulp van de rekenaarsters uitgevoerd, de computer nam dat werk grotendeels over. De rol van de rekenafdeling veranderde: van een rekenafdeling werd het een computerservice afdeling met daarbij een wetenschappelijke poot. Waar eerst de aandacht lag op de bouw van computers en manieren van berekenen, verschoof die (wetenschappelijke) aandacht naar programeren en programmeertalen, de aandacht verschoof onder andere naar de ontwikkeling van ALGOL.

Uit de tabel wordt het verschil in het gebruik van de opeenvolgende machines duidelijk. De eerste ARRA werd amper gebruikt, zo lijkt het. In de opdrachtbeschrijvingen werd het gebruik van deze machine niet genoemd, behalve voor twee testopdrachten. De ARRA II werd wel ingezet voor rekenwerk op de Rekenafdeling. Zodra echter de nieuwe en snellere ARMAC beschikbaar kwam, was de rol van de ARRA snel uitgespeeld. De ARMAC bleef vijf of zes jaar in gebruik op de Rekenafdeling. Vier jaar daarvan werd de machine intensief ingezet voor rekenwerk, daarna werd het nog maar sporadisch gebruikt.

Na de ARMAC kwam de Electrologica X1 en daarmee was ook de computerpioniersperiode op de Rekenafdeling voorbij: de X1 werd voor bijna elke rekenopdracht ingezet, de Rekenafdeling veranderde compleet van aanzien, de rekenaarsters verdwenen, de computerbouw was al afgestoten en de oude unieke automatische rekenmachines waren verdwenen. De X1 van het Mathematisch Centrum was een uit een serie van vele tientallen machines.

De computers waren een belangrijk onderdeel van de Rekenafdeling, maar gekeken naar het effectieve gebruik van deze machines voor het rekenwerk van de afdeling, dan was het beeld van computergebruik minder positief. De ARRA machines, pakweg de eerste tien jaar van het bestaan van het Mathematisch Centrum, waren duidelijk niet geschikt om hun rol als rekenapparaat uit te voeren. Het gebruik van een automatische rekenmachine kan als reden gegolden hebben om dergelijke apparaten te bouwen, maar in de praktijk lag de waarde van deze vroege rekenmachines niet in hun rekenkracht. De waarde zat in de ontwikkeling van ervaring en kennis over de bouw en het gebruik van computers.

Wie waren de opdrachtgevers? Een klein deel van de opdrachten kwam vanuit de eigen afdeling of van andere afdelingen van het Mathematisch Centrum.

jaar	R	ΔR	O	MC	+MC	eigen	ARRA I	ARRA II	ARMAC	X1
1946										
1947										
1948										
1949	64	64	39							
1950	114	50	52	27		6	2			
1951	166	52	59	21		8	1			
1952	206	40	48	17		13	1			
1953	249	43	52	13		9				
1954	297	48	59	8		7		8		
1955	340	43	53	13		13		20		
1956	394	54	60	9	3	7		5	13	
1957	467	73	73	9	3	5			38	
1958	514	47	57	5	3	5			28	
1959	557	43	55	6	2	4			17	2
1960	605	48	69	7		5			1	42
1961	725	120	122	11	32	1				122
1962	852	127	179							179

R = het laatste R-nummer in het jaarverslag van dit jaar.
Δ R = het verschil tussen het laatste R-nummer van dit jaar en dat van het vorige jaar.
O = het aantal opdrachten vermeld in het jaarverslag van het MC bij de rekenafdeling.
MC = het aantal opdrachten waarvan de opdrachtgever uit het MC kwam, dus of de rekenafdeling zelf of een der andere afdelingen.
+MC = het aantal opdrachten van externe partijen die samenwerkten met een andere afdeling van het MC. Dit gegeven werd vanaf 1956 vermeld.
eigen = het aantal opdrachten uitgevoerd als onderdeel van eigen onderzoek. Dit aantal is ook onderdeel van het aantal in de kolom MC.
Hierna volgen het aantal opdrachten dat met een der verschillende computers is uitgevoerd. Hierbij zijn enkel die opdrachten geteld waarbij de computer in dat jaar ook daadwerkelijk gebruikt is om de opdracht uit te rekenen. Er zijn namelijk ook opdrachten waarvan het onduidelijk is of de computer gebruikt is, of dat er enkel een programma is voorbereid. Verder zijn er een aantal opdrachten die wel met computergebruik te maken hebben, zoals het programmeren van een bibliotheek, maar die niet worden geteld als een rekenopdracht.

Tabel 1: Tabel van het aantal rekenopdrachten van de Rekenafdeling van het MC per jaar, per computer.

Het overgrote deel kwam van buiten het Centrum. Opdrachtgevers waren bedrijven, onderzoeksinstituten, universiteiten, laboratoria, overheidsinstellingen, nutsbedrijven en particulieren, voornamelijk uit Nederland, maar er een aantal kwam uit het buitenland.

De opdrachtgevers vormden een gevarieerd gezelschap, de uitgevoerde opdrachten waren daardoor net zo gevarieerd, alhoewel de meeste problemen wel numeriek van aard waren. de opdrachten kunnen verdeeld worden in twee groepen. Ten eerste waren er opdrachten waarbij personeel van de Rekenafdeling een rekenopdracht uitvoerde en de opdrachtgever de resultaten van het rekenwerk deden toekomen. Daarnaast waren er opdrachtgevers die zelf, al dan niet met assistentie van medewerkers van de Rekenafdeling, programma's schreven en die op de computer van het Mathematisch Centrum uitvoerden, of lieten uitvoeren. Naarmate de computers betrouwbaarder werden, groeide ook het aantal opdrachten waarbij de opdrachtgever zelf programma's schreef en deze op de computer uitvoerde.

3 De bouw van computers

De computerbouw op de Rekenafdeling begon met de ARRA, de Automatische Relais Rekenmachine Amsterdam die in twee fasen werd gebouwd door Loopstra en Scholten. In de eerste fase werd een machine gebouwd zonder geheugen maar met een bedieningsmogelijkheid om de rest van de machine te kunnen testen en gebruiken[1]. In 1951 schreef Van Wijngaarden, het hoofd van de rekenafdeling, het rapport *Programmeren voor de A.R.R.A.* waarin hij 'de theorie van de programmering behandeld voor het stadium, waarin de machine het eerst gebracht zal worden na de ombouw volgend op het eerste stadium van gebruik in 1950.'[2]

Dit rapport was een technische beschrijving over de werking van de ARRA en over hoe met de ARRA te werken. De ARRA was een één-adrescode machine met woorden van 30 bits. Het trommelgeheugen was 1024 woorden groot en de machine kende 16 verschillende opdrachten (zie Figuur 1), inclusief deling en vermenigvuldiging. De machine beschikte over een ponsbandlezer als invoer en een typemachine als uitvoer. Het invoerprogramma vertaalde de code ingevoerd door de programmeur, de programmeurscode, naar machinecode. Deze twee codes verschilden maar weinig van elkaar.

In 1952 werd besloten een elektronische, en daardoor meer betrouwbare opvolger van de ARRA te ontwikkelen[3], de ARRA (II). Voor het gebruik van deze

[1] 'Jaarverslag Mathematisch Centrum' (1950), 18–19: 'Om toch de machine in staat te stellen berekeningen uit te voeren en tevens om in staat te zijn de arithmetische eenheden van de machine te kunnen testen op een langdurige belasting werd een noodbesturing van de machine ontwikkeld. Deze bestaat uit een rek, waarop aangebracht zijn een aantal stapschakelaars en een schakelpaneel, met behulp waarvan cycli van circa 200 opdrachten kunnen worden uitgevoerd en een beperkt geheugen, n.l. voor 3 variabele grootheden en 11 constanten.'

Met deze versie van de ARRA werden twee tabellen geproduceerd: R 98 en R 100. Respectievelijk de tabel van x^{-2} van 180 pagina's lang en een tabel van trinomiale vergelijkingen van 8125 waarden groot. Overigens werd in het jaarverslag van 1950 gesproken over de tabellen R 79 en R 100. In het rapport over R 79 werd geen melding gemaakt van het gebruik van de ARRA, waarschijnlijk werd hier R 98 in plaats van R 79 bedoeld.

[2] A. van Wijngaarden, 'Programmeren voor de ARRA', Technisch rapport MR-7 (Amsterdam: Mathematisch Centrum 1951), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9282A.pdf), 1

[3] 'Jaarverslag Mathematisch Centrum' (1952), 44

0,n	Telt (n) op bij (A).
1,n	Plaatst (n) in S.
2,n	Schrijft (A) op adres n.
3,n	Schrijft (S) op adres n en in A.
4,n	Vormt (n) $\times$ (S) + p $\times$ (A) en plaatst het resultaat in A en S.
5,n	Vormt (n) $\times$ (S) $+\frac{1}{2}$ p en plaatst het resultaat in A en S.
6,n	Vermenigvuldigt (A) met 2^n.
7,n	Verplaatst besturing naar n als (A) ≥ 0.
8,n	Trekt (n) af van (A).
9,n	Communicatieopdracht.
10,n	Schrijft (A) op adres n en maakt A schoon.
11,n	Schrijft (S) op adres n en maakt A schoon.
12,n	Deelt (A) + p $\times$ (S) door (n), plaatst rest in A en quotient in S.
13,n	Deelt (A) door (n), plaatst afgerond quotient in S en maakt A schoon.
14,n	Vermenigvuldigt (A) met 2^{-n}.
15,n	Verplaatst besturing naar n als (A) < 0.

Figuur 1: De opdrachten van de ARRA (Ib) uit: A. van Wijngaarden, *Programmeren voor de A.R.R.A.* (1951), 34.

machine zijn vier rapporten geschreven door Edsger Dijkstra die in 1952 als programmeur was aangesteld bij de Rekenafdeling. Naast deze technische rapporten werd de ARRA ook behandeld in de eerste cursus *Programmeren voor automatische rekenmachines*.

Het eerste rapport over de ARRA II was de *Functionele beschrijving van de ARRA*[4] uit 1953 waarin uitgelegd werd waartoe de ARRA in staat was en hoe die functionaliteit kon worden aangeroepen. De tweede ARRA was ook een één-adrescode machine met een trommelgeheugen van 1024 woorden van 30 bits. In een woord gingen twee instructies, de "a" en de "b" opdracht. In vergelijking met de opdrachten van de eerste ARRA waren de opdrachten van de ARRA II (zie Figuur 2) beter uitgebalanceerd. Een aantal eenvoudige rekenkundige opdrachten waren zowel in register A als in register S uitgevoerd. Daarnaast was er de onconditionele sprongopdracht bijgekomen, die speciaal bedoeld was voor de aanroep van subroutines. Ook de schuifopdrachten waren nieuw.

Een belangrijke ontwikkeling bij de ARRA II was de aandacht die de communicatie met de buitenwereld kreeg. Niet alleen werden de communicatieprogramma's complexer en kregen meer mogelijkheden, ze werden tijdens de levensduur van de ARRA II ook onderhouden. In 1954 schreef Dijkstra *In- en uitvoer van de ARRA*[5] waarmee de oude beschrijving van de communicatie met de ARRA kwam te vervallen.

Communicatie met een automatische rekenmachine ging handmatig via de bedieningstafel of automatisch via de ponsbandlezer en typmachine[6]. De bedie-

[4] E.W. Dijkstra, 'Functionele beschrijving van de ARRA', Technisch rapport MR-12 (Amsterdam: Mathematisch Centrum 1953), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9277A.pdf)

[5] E.W. Dijkstra, "'Drijvende komma" rekentechniek (ARRA subroutines RD1 en RD2)', Technisch rapport MR-16 (Amsterdam: Mathematisch Centrum 1954), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9273A.pdf)

[6] Door het gebruik van de typemachine waren deze automatische rekenmachines overigens verre van automatisch: vel voor vel werd getypt en zodra een vel vol was moest de computer stoppen, moest de operateur een nieuw vel in de typemachine stoppen en de computer instrueren door te gaan.

0/n	vervang (A) door (A)+(n)
1/n	vervang (A) door (A)−(n)
2/n	vervang (A) door (n)
3/n	vervang (A) door −(n)
4/n	vervang (n) door (A)
5/n	vervang (n) door −(A)
6/n	conditionele besturingsverplaatsing naar n a
7/n	besturingsverplaatsing naar n a
8/n	vervang (S) door (S)+(n)
9/n	vervang (S) door (S)−(n)
10/n	vervang (S) door (n)
11/n	vervang (S) door −(n)
12/n	vervang (n) door (S)
13/n	vervang (n) door −(S)
14/n	conditionele besturingsverplaatsing naar n b
15/n	besturingsverplaatsing naar n b
16/n	vervang [AS] door [n].[s] + [A]
17/n	vervang [AS] door -[n].[s] + [A]
18/n	vervang [AS] door [n].[s]
19/n	vervang [AS] door -[n].[s]
20/n	deel [AS] door [n], plaats quotient in S, rest in A
21/n	deel [AS] door -[n], plaats quotient in S, rest in A
22/n	"Schuif A→S", d.w.z. vervang [AS] door [A].2^{29-n}
23/n	"Schuif S→A", d.w.z. vervang [SA] door [S].2^{30-n}
24/n	communicatie opdracht

Figuur 2: De opdrachten van de ARRA (II) uit: E.W. Dijkstra, *Functionele beschrijving van de ARRA* (1953), 3.

ningstafel kon worden gebruikt om getallen in de registers te zetten, om bepaalde opdrachten uit te voeren of te herhalen en om opdrachten in het geheugen te zetten.

Het bedienen met de hand deed het automatische karakter van de machine teniet en werd daarom gebruikt voor die situaties waar directe controle over de machine gewenst was: bij het testen van apparatuur en programma's, bij problemen met de machine en bij bepaalde berekeningen waarbij de operateur invloed kon of moest oefenen om het gewenste resultaat te bereiken[7]. In hoeverre werden deze automatische rekenmachines eigenlijk automatisch gebruikt? Of algemener: hoe werden deze machines eigenlijk gebruikt, wat was de dagelijkse gang van zaken voor een operateur van zo'n machine? Met de huidige gegevens kunnen deze vragen niet beantwoord worden, maar het is wel duidelijk dat het continue in gebruik zijn van dergelijke machines niet betekende dat de machine continue draaide.

Een jaar later schreef Dijkstra wederom een nieuw rapport over de communicatieprogramma's omdat de invoer- en uitvoerapparaten van de ARRA uitgebreid of verbeterd werden[8]. Er werd nadrukkelijk op gelet om de bestaande ponsconventies niet te veranderen, bestaande programma's moesten gewoon blijven werken. Het laatste rapport over de ARRA II was *"Drijvende-komma"-rekentechniek (ARRA-subroutine Rd1 en Rd2*[9] waarin de subroutines werden beschreven die het rekenen en werken met een drijvende komma op de ARRA gemakkelijker maakten.

Nadat de ARRA II gebouwd en in gebruik genomen was, werd begonnen aan de bouw van de FERTA, een met de ARRA vergelijkbare computer voor de vliegtuigbouwer Fokker. Deze computer was niet, zoals vaak wordt verondersteld, een kopie van de ARRA, het was een volgende stap in de ontwikkeling van computers op de Rekenafdeling. Over de FERTA zijn twee rapporten verschenen, wederom van de hand van Dijkstra. Het eerste rapport[10] behandelde de machine, de opdrachten, het geheugen en subroutines. Het tweede rapport[11]

[7] In H.J. Dannenburg, 'De berekening van eigenwaarden en -vectoren van matrices op de FERTA', in: *Nederlands rekenmachine genootschap 1959 – 1969. Colloquium Moderne Rekenmachine II* (Amsterdam 1969), 90–97 werd een voorbeeld gegeven van zo'n berekening waarbij de operateur actief het rekenproces moest monitoren en ingrijpen om het gewenste resultaat te krijgen. De operateur werd als het ware onderdeel gemaakt van het programma: 'Het is niet eenvoudig om deze berekening vol-automatisch uit te laten voeren door de machine. De machine zou zelf moeten beslissen of een eigenvector nauwkeurig genoeg is bepaald door de iteratievector. Een criterium is wel aan te geven, doch vooraf is moeilijk te overzien of er, met de afrondingsfouten, die de machine maakt, aan kan worden voldaan. Een criterium, aan de hand waarvan besloten wordt tot het al dan niet oplossen van de vierkantsvergelijking zou al zeer gecompliceerd worden. Dergelijke beslissingen worden dan ook overgelaten een de operateur van de machine. Deze kan het iteratieproces volgen door van tijd tot tijd de iteratievector uit te laten typen. Hij kan de machine besturen met behulp van de getalschakelaar (GS).' (p. 94).

[8] E.W. Dijkstra, 'Het communicatieprogramma van de ARRA', Technisch rapport MR-21 (Amsterdam: Mathematisch Centrum 1955), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9268A.pdf)

[9] E.W. Dijkstra, 'In- en uitvoer van de ARRA', Technisch rapport MR-14 (Amsterdam: Mathematisch Centrum 1954), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9275A.pdf)

[10] E.W. Dijkstra, 'Handboek voor de programmeur (FERTA) I', Technisch rapport MR-17 (Amsterdam: Mathematisch Centrum 1955), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9272A.pdf)

[11] E.W. Dijkstra, 'Handboek voor de programmeur (FERTA) II', Technisch rapport MR-20 (Amsterdam: Mathematisch Centrum 1955), (URL:http://repos.project.cwi.nl:8888/

0,n	(A)' = (A) + (n)
1,n	(A)' = (A)−(n)
2,n	(A)' = (n)
3,n	(A)' = −(n)
4,n	(n)' = (A), (C)' = $(A)_{29}$'
5,n	(n)' = −(A), (C)' = 1 − $(A)_{29}$'
6,n	als (C) = 0, dan (T)' = n
7,n	(T)' = n
8,n	(S)' = (S) + (n)
9,n	(S)' = (S)−(n)
10,n	(S)' = (n)
11,n	(S)' = −(n)
12,n	(n)' = (S), (C)' = $(S)_{29}$'
13,n	(n)' = −(S), (C)' = 1 − $(S)_{29}$'
14,n	als (C) = 0, dan (T)' = n+$\frac{1}{2}$
15,n	(T)' = n+$\frac{1}{2}$
16,n	[AS]'= [n].[s] + [A]
17,n	[AS]'= -[n].[s] + [A]
18,n	[AS]'= [n].[s]
19,n	[AS]'= -[n].[s]
20,n	[n].[S]'+[A]'=[AS]
21,n	−[n].[S]'+[A]'=[AS]
22,n	als (C) = 0, stop
23,n	als (C) = 1, stop

Figuur 3: Een deel van de opdrachten van de FERTA, opdrachten 24 tot en met 29 zijn functieopdrachten. Uit: E.W. Dijkstra, *Handboek voor de programmeur. FERTA deel I* (1955), 18.

ging over de in- en uitvoerfaciliteiten en bijbehorende programmatuur. Deze rapporten waren uitgebreider dan de twee overeenkomstige rapporten over de ARRA, waarschijnlijk omdat de machine niet voor intern gebruik op de Rekenafdeling was bedoeld.

De technische specificaties van de FERTA waren wat betreft woordlengte en geheugengrootte gelijk aan die van de ARRA. De verschillen zaten vooral in de in- en uitvoer en de daar op betrekking hebbende opdrachten. In Figuur 3 zijn de opdrachten van de FERTA weergegeven. De eerste 21 opdrachten waren gelijk aan die van de ARRA. De schuifopdrachten van de ARRA zijn opgegaan in de functieopdrachten (opdracht 24 tot en met opdracht 29) die niet met een 10 bits adres werkten maar met twee 5 bits codes die de functie bepaalden. Verder werd de conditionele stop toegevoegd.

De functieopdrachten waren een doorontwikkeling van de communicatieopdrachten bij eerdere machines: een code op het adres of numerieke gedeelte van de opdracht bepaalde de uit te voeren handeling. Bij de ARRA werd nog maar een beperkt aantal functies mogelijk gemaakt door dergelijke functieopdrachten, bij de FERTA was dat aantal verveelvoudigd. Deze functieopdrachten hadden voor een deel betrekking op de in- en uitvoer, bijvoorbeeld het lezen van het handregister of het getalregister in A of S, of het schrijven naar de typemachine. Verder kon de opdrachtteller gelezen worden, kon het conditieregister worden geschreven, speelde het een rol bij de vorming van de koppelopdracht bij de aan-

`cwi_repository/docs/I/09/9269A.pdf`)

roep van subroutines, was het mogelijk om het register A te schuiven, konden enkele snelle optellingen en vermenigvuldigingen worden uitgevoerd, kon de nultest op A gedaan worden en kon de skipopdracht worden uitgevoerd. Kortom, veel functionaliteit werd bij de FERTA via deze functieopdrachten beschikbaar gesteld.

Ook de invoer- en uitvoerfaciliteiten zoals dat door de communicatieprogramma's werd aangeboden, was sterk uitgebreid ten opzicht van de ARRA. Naast de typemachine en bandlezer was er ook een bandponser. Dit maakte het onder meer mogelijk om de computer ponsbanden te laten kopiëren, om gegevens of tussenresultaten bij grote berekeningen uit te ponsen of om delen van het geheugen direct op band te ponsen door gebruik te maken van het serviceprogramma BIBAND.

Na de FERTA werd begonnen aan een nieuwe machine voor gebruik op de Rekenafdeling: de ARMAC. Over de ARMAC en een aantal superprogramma's voor de ARMAC verscheen een serie van zes rapporten: *Programmeren voor de ARMAC*[12]. Twee delen werden later aangepast[13]. Verder verschenen een tweetal korte rapportjes waarin de opdrachtencode[14] en het standaard typprogramma[15] van de ARMAC werden toegelicht, deze twee rapportjes kunnen beschouwd worden als "uittreksels" van het later verschenen eerste deel van de serie waarin de machine en het gebruik ervan uitgebreid werd beschreven.

De ARMAC leek in vele opzichten op de FERTA, maar de verbeteringen waren enorm. Door het gebruik van snel geheugen, zowel voor buffering van het langzame trommelgeheugen als voor vrij toegankelijk geheugen, was de ARMAC vele malen sneller dan voorgaande machines. Verder was het langzame geheugen

[12] E.W. Dijkstra, 'Programmering voor de ARMAC, 1; Algemeen', Technisch rapport MR-25 (Amsterdam: Mathematisch Centrum 1956), (URL:http://www.cs.utexas.edu/users/EWD/MCReps/MR25.PDF);
E.W. Dijkstra, 'Programmering voor de ARMAC, 2; De inhoud der geblokkeerde kanalen', Technisch rapport MR-26 (Amsterdam: Mathematisch Centrum 1956), (URL:http://www.cs.utexas.edu/users/EWD/MCReps/MR26.PDF);
E.W. Dijkstra, 'Programmering voor de ARMAC, 3; Interpretatief programma voor drijvende komma berekening', Technisch rapport MR-27 (Amsterdam: Mathematisch Centrum 1956), (URL:http://www.cs.utexas.edu/users/EWD/MCReps/MR27.PDF);
L. Vasmel-Kaarsemaker, 'Programmering voor de ARMAC, 4; Interpretatief programma voor het werken met 6-voudige lengte getallen', Technisch rapport MR-28 (Amsterdam: Mathematisch Centrum 1957), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9259A.pdf);
E.W. Dijkstra, 'Programmering voor de ARMAC, 5; Interpretatief programma voor breuken van dubbele lengte', Technisch rapport MR-29 (Amsterdam: Mathematisch Centrum 1957), (URL:http://www.cs.utexas.edu/users/EWD/MCReps/MR29.PDF);
T.J. Dekker, 'Programmering voor de ARMAC, 6; Matrix complex RAM', Technisch rapport MR-30 (Amsterdam: Mathematisch Centrum 1959), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9257A.pdf)

[13] E.W. Dijkstra, 'Programmering voor de ARMAC, 1a; Algemeen', Technisch rapport MR-25 a (Amsterdam: Mathematisch Centrum 1957), (URL:http://www.cs.utexas.edu/users/EWD/MCReps/MR25a.PDF);
N.C. Bakker, 'Programmering voor de ARMAC, 3a; Interpretatief programma voor complexe getallen met drijvende komma's', Technisch rapport MR-27a (Amsterdam: Mathematisch Centrum 1957), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9260A.pdf)

[14] E.W. Dijkstra, 'Korte beschrijving van de opdrachtencode etc. voor de ARMAC', Technisch rapport MR-23 (Amsterdam: Mathematisch Centrum 1956), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9266A.pdf)

[15] E.W. Dijkstra, 'Het standaard typprogramma van ARMAC', Technisch rapport MR-24 (Amsterdam: Mathematisch Centrum 1956), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9265A.pdf)

0/n	(A)+(n) ⇒ (A)
1/n	(A)−(n) ⇒ (A)
2/n	+ (n) ⇒ (A)
3/n	− (n) ⇒ (A)
4/n	+ (A) ⇒ (n); (n) ≥ + 0? of (A) ≥ + 0?
5/n	− (A) ⇒ (n); (n) ≥ + 0? of (A) ≤ − 0?
6/n	n ⇒ (T)
7/n	n $+\frac{1}{2}$ ⇒ (T)
8/n	(S)+(n) ⇒ (S)
9/n	(S)−(n) ⇒ (S)
10/n	+ (n) ⇒ (S)
11/n	− (n) ⇒ (S)
12/n	+ (S) ⇒ (n); (n) ≥ + 0? of (S) ≥ + 0?
13/n	− (S) ⇒ (n); (n) ≥ + 0? of (S) ≤ − 0?
14/n	n ⇒ (T) als (C) = 0 en (T) $+\frac{1}{2}$ ⇒ (T) als (C) = 1
15/n	n $+\frac{1}{2}$ ⇒ (T) als (C) = 0 en (T) $+\frac{1}{2}$ ⇒ (T) als (C) = 1
16/n	[A] + [n].[s] ⇒ [AS]
17/n	[A] − [n].[s] ⇒ [AS]
18/n	+ [n].[s] ⇒ [AS]
19/n	− [n].[s] ⇒ [AS]
20/n	transporteer track van langzaam geheugen naar snel.
21/n	transporteer track van snel geheugen naar langzaam.
22/n	plaats link in A en spring naar de a-opdracht van adres n
23/n	plaats link in A en spring naar de b-opdracht van adres n
24/···	Schuif- en communicatieopdrachten
25/···	“ “ ”
26/···	“ “ ”
27/···	“ “ ”
28/···	“ “ ”
29/···	“ “ ”

Figuur 4: De opdrachten van de ARMAC uit: E.W. Dijkstra, *Programmering voor de ARMAC. Deel I* (1956), 23.

vier keer zo groot geworden waardoor ook de woordlengte groter werd: 34 bits; nog steeds twee opdrachten in een woord.

De opdrachten (zie Figuur 4) leken op die van de FERTA, maar er waren ook een aantal verschillen. Ten eerste was de deelopdracht verdwenen: de deling moest op de ARMAC via een standaard subroutine uitgevoerd worden. Verder was de conditionele stop weer een functieopdracht geworden en waren er twee opdrachten toegevoegd om tracks van langzaam naar snel geheugen te verplaatsen en omgekeerd. De functieopdrachten waren grotendeels hetzelfde, maar er werden een aantal functies toegevoegd voor het handmatig werken met de buffer; normaal gesproken vulde de ARMAC automatisch de buffer met het huidig in gebruik zijnde geheugenkanaal uit het langzame geheugen.

Verder waren er een aantal subroutines standaard in het geblokkeerde geheugen aanwezig zoals worteltrekken, sinus en cosinus, exacte deling en breuken deling. In het geblokkeerde deel van het geheugen huisden ook de communicatieprogramma's. Net als bij de FERTA was er een ponsbandlezer, een bandponser, een typemachine en een bedieningspaneel. De functionaliteit van de typemachine was echter wel sterk vergroot: er was nu ook de mogelijkheid om lettertekens te typen.

Net als voor de ARRA werd ook voor de ARMAC een superprogramma gemaakt voor het werken met een drijvende komma. Door middel van dit programma werd de ARMAC als het ware een machine die over drijvende komma beschikte en werd het werk van de programmeur enorm vergemakkelijkt. Daarnaast werden er nog een aantal andere interpretatieve programma's voor de ARMAC geschreven die de machine van extra functionaliteit en gebruiksgemak konden voorzien: een programma voor het werken met zesvoudig lange getallen, een programma voor het werken met breuken van dubbele lengte en een complex van programma's voor het werken met matrices.

De opeenvolgende machines die de Rekenafdeling bouwde, werden steeds complexer, kregen meer functionaliteit, werden sneller en kregen een groter geheugen. Ook werd meer en betere randapparatuur aangesloten. Niettemin bleef de basis van de machine constant gedurende deze ontwikkeling, de machines zijn duidelijk opvolgers van vorige machines. Naast deze kwantitatieve verbeteringen vond ook een kwalitatieve verbetering plaats: de machines werden steeds betrouwbaarder.

Aan de andere kant groeide het gebruiksgemak van de machines door verbetering en uitbreiding van de opdrachten. Er werd functionaliteit ingebouwd voor het eenvoudig uitvoeren van subroutines, de functieopdrachten werden ingevoerd en de mogelijkheden ervan vergroot. Door het gebruik van superprogramma's zoals het invoerprogramma en verschillende interpretatieve programma's kreeg de gebruiker een machine die extra functionaliteit had en die eenvoudiger te gebruiken was.

Deze dubbele ontwikkeling van machine en gebruiksgemak was een gelijktijdige ontwikkeling: doordat de machines beter werden, kon meer functionaliteit door programma's geregeld worden. Doordat meer functionaliteit en gebruiksgemak gewenst was doordat de computers steeds meer gebruikt werden, was een ontwikkeling en verbetering van de computer ook noodzakelijk.

4 Programmeren van automatische rekenmachines

Naast een ontwikkeling in computerbouw en functionaliteit was er een tweede ontwikkeling gaande: de ontwikkeling van het programmeren. In de verschillende rapporten werden aanwijzingen gegeven hoe de machines geprogrammeerd konden worden en werden opmerkingen gemaakt over het programmeren in het algemeen. Hierbij werden verschillende notaties gebruikt die in de loop van de jaren ook veranderden. Deze notaties en opmerkingen geven een beeld over hoe op de Rekenafdeling werd gedacht over programmeren.

Daarnaast werden er twee cursussen programmeren voor automatische rekenmachines gegeven door de Rekenafdeling waarin uitgelegd werd wat het programmeren inhield en welke technieken bij het programmeren komen kijken. Ook deze informatie geeft een beeld van het programmeren en het gebruik van de machines op de Rekenafdeling.

De Rekenafdeling stond in het teken van het rekenen, van het uitvoeren van geavanceerd technisch-wetenschappelijk rekenwerk voor de andere afdelingen van het Mathematisch Centrum en voor derden uit industrie, wetenschap en overheid. De bouw van automatische rekenapparaten, maar bovenal het gebruik ervan stond dan ook in het teken van geavanceerd wetenschappelijk rekenwerk en bouwde voort op de al bestaande traditie van handmatig rekenwerk. De computer en het programmeren werden dan ook vanuit het perspectief van de rekenaar met een handrekenmachine geïntroduceerd.

In het eerste rapport, *Programmeren voor de A.R.R.A.* uit 1951 werd uitgelegd wat een woord was, dat het op verschillende manieren geïnterpreteerd kon worden: als een geheel getal, als een breuk, als een opdracht en als een code. De notatie waarin programma's in machinecode werden opgeschreven is eenvoudig: een adres gevolgd door de combinatie opdracht/numeriek gedeelte of een getal. In de rechter kantlijn was plaats voor uitleg.

Sprongen werden met pijltjes verduidelijkt, in de linker kantlijn werden inkomende pijlen geannoteerd met het adres vanwaar gesprongen werd en in de rechter kantlijn werd enkel met een pijl aangegeven dat dit een sprongopdracht betrof. Een dubbele pijl in de rechter kantlijn betekende een niet-conditionele sprong. Een adres tussen haakjes betekende dat de inhoud van dat adres bedoeld werd en niet het adres zelf. In veel voorbeeldprogramma's werd een relatieve adressering gebruikt van $c + x$ met x oplopend van 0 tot het aantal instructies van het programma.

Deze machinecode verschilde van de zogenaamde programmeurscode die door de programmeur geponst werd. Een geponste regel had de vorm: beginletter, getal en eindletter. Een reeks geponste instructies kon tussen twee verticale lijnen geplaatst worden waar aan beide kanten het programma geannoteerd kon worden met regelnummers, sprongpijlen en uitleg.

Als de geponste regel een opdracht was, dan was de beginletter eenvoudigweg het opdrachtnummer. Als echter een getal geponst werd dan was het een teken, hetzij voor een breuk, hetzij voor een geheel getal. Tenslotte konden nog opbergadressen geponst worden, die begonnen met een "A". Het getal was natuurlijk het numerieke gedeelte van een opdracht, de absolute waarde van een getal of een adres. De eindletter was bedoeld voor relatieve adressering, de verschillende eindletters kregen voor gebruik een waarde toegewezen dat als het

beginadres van de reeks opdrachten afgesloten met die eindletter werd gebruikt.

Een belangrijk onderdeel van het programmeren was het gebruik en het maken van subroutines. Deze programma's waren van hoge kwaliteit en konden door hergebruik de programmeur veel werk uit handen nemen. Een subroutine kon gewoonweg overgenomen worden in een programma wanneer het nodig was, dit werden "open subroutines" genoemd. "Gesloten subroutines" daarentegen stonden op zichzelf en maakten geen deel uit van het programma, ze werden enkel aangeroepen door het programma. Bij het gebruik van dergelijke subroutines was het gebruik van sluitletters onontbeerlijk. Bij gebruik van een subroutine diende de programmeur een voorponsing te doen waarin aangegeven werd waar de subroutine in het geheugen kwam te staan en wat de parameters waren. Hierna kopieerde hij de subroutine automatisch met behulp van de reproduceerpons. Een subroutine werd aangeroepen met een speciale combinatie van opdrachten die de adresinformatie in A plaatste dat door de subroutine gebruikt werd om het terugkeeradres achteraan de subroutine te plaatsen, dit was de zogenaamde koppelopdracht.

In dit rapport uit 1951 werd dus wel uitgelegd hoe een programma er uit zag en hoe subroutines gebruikt konden worden. Over het programmeren, het omzetten van een wiskundig probleem in een programma dat dat probleem oploste, werd echter niets gezegd. Bij de rapporten over de ARRA II werd daar wel iets over gezegd: in *Functionele beschrijving van de ARRA* werd, ten opzichte van het eerdere rapport uit 1951, het hoofdstuk 'De taak van de programmeur' toegevoegd. Het programmeren werd in vijf stappen opgedeeld:

> '1^e. mathematische formulering van het probleem,
> 2^e. mathematische oplossing van het probleem,
> 3^e. keuze of constructie van numerieke processen, die (in het licht van 2^e) tot het gewenste antwoord leiden,
> 4^e. *programmering*: gedetailleerde opbouw van de onder 3^e genoemde processen uit de elementaire bewerkingen, waartoe de machine in staat is,
> 5^e. *codering*: uitschrijven van het programma in de code der machine, zodat hierna de band onmiddellijk geponst kan worden.'[16]

Verder noemde Dijkstra een zestal idealen die nagestreefd werden bij het programmeren: maximale snelheid, minimale geheugenruimte, maximale veiligheid, maximale accuratesse, maximale souplesse en maximale overzichtelijkheid.[17] De kwaliteit van het programma, zowel voor de machine als voor de programmeur, was dus een belangrijk aspect van het programmeren. Daarna werd een typisch programma gekarakteriseerd: het bestaat uit een aantal opdrachten die herhaaldelijk uitgevoerd worden. Eigenlijk was dit ook een karakterisatie van een automatische rekenmachine: het automatisch uitvoeren van veel (deel en tussen) berekeningen.

Naast algemene opmerkingen over het programmeren werd ook de notatie van programma's in dit rapport uitgebreid. De verschillende interpretaties van woorden werden met verschillende haakjes aangegeven. De inhoud van het adres x was (x); (x) geïnterpreteerd als een geheel getal werd geschreven als $[x]$; (x)

[16] Dijkstra, 'Functionele beschrijving van de ARRA', 33
[17] Ibidem

geïnterpreteerd als een breuk werd geschreven als $\{x\}$ en met $< x >$ werd aangegeven dat (x) als twee opdrachten werd geïnterpreteerd. Met behulp van deze notatie konden programma's korter en overzichtelijker worden uitgelegd. Een andere manier om programma's overzichtelijk weer te geven was het blokschema dat in dit rapport wel gebruikt werd, maar niet geïntroduceerd.

Ook de notatie voor de programmeurscode was iets veranderd. De sluitletters waren nu geen getallen meer maar letters X en A tot en met F. Daarnaast was er een vierde element op een regel bijgekomen: de kanaalcorrectie, die alleen bij het invoeren van een adres gebruikt mocht worden. Met de kanaalcorrectie kon het kanaal op het trommelgeheugen uitgekozen worden waar de geponste regel opgeslagen moest worden.

Opvallend is dat in de twee handboeken voor de FERTA weer niets gezegd werd over het programmeren in het algemeen, enkel de machine zelf werd beschreven. Ook bij de beschrijving van de ARMAC was dit het geval. Dit betekende niet dat het programmeren geen belangrijk onderwerp meer was, integendeel, in 1955 werd de eerste cursus programmeren voor automatische rekenmachines gehouden en eind 1957 de tweede. In deze cursussen werd het hoe en wat van het programmeren zeer uitgebreid behandeld. Het programmeren werd dus losgemaakt van het bedienen van een speciale machine, de cursussen zijn dan ook niet bedoeld om specifiek te leren werken met de machine van de Rekenafdeling, alhoewel die machines wel gebruikt worden als de voorbeeldmachine: programmeren zat immers nog dicht bij de machine.

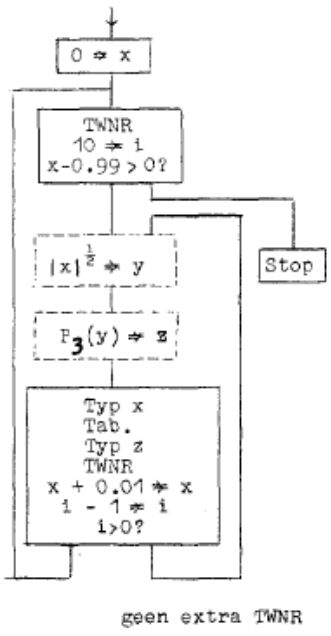
De cursus begon met een kenschets van de kracht van de automatische rekenmachine: 'tegenwoordig kan men amper spreken van "werk uit handen nemen": de problemen, waarbij de machtigste der moderne machines pas goed tot hun recht komen, zijn dusdanig omvangrijk, dat men er zonder deze rekenapparatuur nooit aan zou zijnbegonnen! Er worden problemen mee aangepakt, die vroeger de meest drieste niet eens als "numeriek probleem" zou durven te beschouwen; en inderdaad, naarmate de methoden, waarop de machines hun resultaten afleveren, geraffineerder worden, raakt het numeriek karakter althans voor de naïeve bezoeker, ernstig op de achtergrond.'[18]

Na een viertal algemene introducerende hoofdstukken waarin achtereenvolgens automatische rekenmachines, woorden, getallen en opdrachten besproken werden, ging de cursus verder met blokschema's (flow diagrams). Het belang van blokschema's was het vergroten van de leesbaarheid of begrijpelijkheid van programma's: 'Voor de geïnteresseerde, die het programma inkijkt, zelfs voor de programmeur, die het programma opgesteld heeft, houdt deze "verstaanbaarheid" echter niet over: het lezen van een programma van de opdrachten alleen, zonder een blik te slaan in de explicatie, die er gelukkig doorgaans wel naast staat, vergt erkend veel geduld en doorzettingsvermogen.'[19]

De leesbaarheidsproblemen werden veroorzaakt doordat er veel opdrachten nodig waren om duidelijk afgebakende eenheden van functionaliteit te beschrijven en doordat er veel administratieve operaties uitgevoerd moesten worden: het overzicht op de structuur van het programma raakte daardoor verborgen. Blokschema's maakten het juist mogelijk die structuur en die blokken functionaliteit duidelijk aan te geven. Het grote nadeel van blokschema's was de

[18] T.J. Dekker, E.W. Dijkstra en A. van Wijngaarden, 'Cursus programmeren voor automatische rekenmachines', Technisch rapport CR-9 (Amsterdam: Mathematisch Centrum 1957), (URL:http://www.cs.utexas.edu/users/EWD/MCReps/CR1957-009.PDF), 1

[19] Ibidem, 20



Stel, dat gevraagd is een 3^{de}-graads polynoom te tabelleren, met als argument $x^{\frac{1}{2}}$, voor x = 0 (0,01)0.99. Om de 10 regels is een extra regel blank gewenst. We maken gebruik van de typ-signalen TWNR (= Terug Wagen, Nieuwe Regel) en Tab = (Tabuleert) en een typprogramma. We kunnen de beide behandelde blokschema's incorporeren. De index i telt de regels in een "blokje" van 10. In fig. 17 is het blokschema weergegeven. De gecomprimeerde blokken zijn door stippellijnen aangegeven. Men realisere zich, dat (b.v. als de machine in het tweetallig stelsel werkt), voor de operatie Typ mogelijkerwijs meer dan één opdracht - een stuk programma dus - nodig is.

Figuur 5: Het blokschema en uitleg van de tabellatie van een derdegraads polynoom, uit: Dekker, Dijkstra en Van Wijngaarden, *Cursus programmeren voor automatische rekenmachines* (1957), 29.

inefficiëntie van de programma's; door 'geraffineerd van de - vaak onbedoelde! - speciale eigenschappen van de machine gebruik te maken'[20] was het mogelijk 'uitgekookte "getructe" programma's'[21] te maken. Op een dergelijke manier optimum programmeren was geschikt voor het maken van standaard subroutines.

Blokschema's werden vereenvoudigd door bepaalde conventies in acht te houden. Een blokschema bestond uit blokken, functionele eenheden, verbonden door lijnen. Een lijn kwam bovenaan een blok binnen en verliet het blok onderaan, hierdoor was het gebruik van pijlen overbodig. Een keuzemogelijkheid werd aangegeven door een vraag onderaan in een blok met twee uitgangen. De linkeruitgang was het nee-geval en de rechteruitgang het ja-geval. Verder werd in deze cursus ook een symbool gebruikt voor toekenning van waarden aan variabelen (adressen). Aanroepen van subroutines, die in zichzelf al functionele eenheden waren, werden met een gestippeld blok aangegeven. In Figuur 5 is een voorbeeld gegeven van blokschema dat een programma voorstelt dat de 3^de-graads polynoom tabelleert.

In deze tweede cursus werd ook een zogenaamd verloopschema geïntroduceerd dat in de eerste cursus nog niet voorkwam. In een verloopschema werd aangegeven in welke volgorde de aritmetische operaties worden uitgevoerd. Veelal was zo'n verloopschema bedoeld om een groot aantal vergelijkbare operaties weer te geven. Een dergelijk verloopschema kon omgezet worden in een blokschema door de introductie van een teller en een iteratie. Zo'n programma was

[20] Ibidem, 21
[21] Ibidem

vaak langzamer dan het gestrekte programma waar alle operaties uitgeschreven waren. Maar het gestrekte programma nam weer meer geheugenruimte in beslag.

Het grootste deel van de rest van de cursus werd besteed aan subroutines van verschillende aard. Het nut van standaard subroutines, de herbruikbaarheid en daardoor het verminderen van fouten en programmeertijd, werd uitgelegd. Een aantal standaard subroutines werd uitgebreid besproken door verschillende manieren om dergelijke functies, zoals sinus, worteltrekken en machtsverheffen, uit te werken tot blokschema's.

Voordat administratieve subroutines en superprogramma's besproken werden, verschoof de aandacht naar de kwaliteit van programma's en subroutines, eigenschappen zoals snelheid, schaling, controle en flexibiliteit werden besproken. De snelheidsbeperkende factor van een machine was over het algemeen het geheugen. Door de introductie van snel geheugen en een buffer was dit bij de ARMAC eigenlijk geen probleem meer. Schaling was het vermenigvuldigen van constanten en variabelen met geschikt gekozen schalingsfactoren waardoor ze een hanteerbare orde van grootte kregen. Als geen goed idee bestond over de orde van grootte dan moesten andere getalrepresentaties, zoals de drijvende komma, gebruikt worden.

Tot controle behoorden al die maatregelen die de programmeur trof om zeker te zijn dat de door het programma verkregen resultaten overeenkwamen met de bedoelde resultaten. Ook hierbij gold dat deze controle steeds meer in handen van de machine kwam, bij de ARMAC was er bijvoorbeeld een paritycheck in het geheugen ingebouwd, een extra bit zodat het aantal enen in een half woord plus een altijd oneven was, bleek bij controle dat dit niet het geval was, dan stopte de machine. Maar ook de programmeur kon en moest controles inbouwen in zijn programma: 'In tegenstelling tot het verleden is nu het stadium bereikt, dat de zwakste schakel in het proces niet meer de machine, maar – de programmeur is! En de programmeur doet er goed aan met behulp van het programma zichzelf te controleren.'[22] Bijvoorbeeld om het gebrek aan accuratesse bij het ponsen te verhelpen.

Flexibiliteit van een programma hield in dat het programma robuust was: dat na een machinefout of tijdens het testen het programma herstart kon worden. Maar ook het eenvoudig kunnen wijzigen van een programma was een onderdeel van de flexibiliteit. Sterker nog, zodra een programma hergebruikt kon worden was het goed de flexibiliteit ervan in het oog te houden: 'de moeilijkheid is dat veelal de wens naar voren zal komen een bestaand programma te gebruiken op een manier, die niet precies bedoeld was, voor een probleem, waarvoor het programme niet precies gemaakt was. De programmeur maakt het programma, d.w.z. een precieze formulering van het rekenschema: aan hem wordt overgelaten, rekening te houden met allerlei mogelijke nog niet geformuleerde eisen.'[23]

Administratieve subroutines waren subroutines met een coördinerende taak, ze riepen andere subroutines aan en konden op verschillende manieren aangeroepen worden om de gewenste, iets verschillende, functionaliteit te verkrijgen. Als voorbeeld werden subroutines voor tabellatie en integratie genoemd.

Tenslotte werden zogenaamde superprogramma's behandeld. De term "su-

[22] Ibidem, 79
[23] Ibidem, 82

perprogramma" werd in de cursus gebruikt om die programma's aan te duiden die de taak hadden om andere programma's te onderzoeken, te interpreteren of op een andere manier met of op programma's werken. Het doel van deze programma's, zoals een invoerprogramma of een drijvende komma programma, was om het leven van de programmeur gemakkelijker te maken. Een ander voorbeeld was een zogeheten ladderprogramma dat iteraties automatisch uitschreef.

Overigens werd de term "superprogramma" niet gebruikt in de andere besproken rapporten, het was bedoeld om tijdens de cursus een bepaald soort programma te karakteriseren. In het hoofdstuk over superprogramma's werd dan ook niet gesproken over hoe dergelijke programma's gemaakt kunnen worden, enkel de mogelijkheden van superprogramma's werden besproken. Superprogramma's waren als het ware voorlopers van besturingssystemen, compilers, interpreters en andere tools. Deze superprogramma's maakten, vanuit het perspectief van de gebruiker, onderdeel uit van het computersysteem, het waren systeemprogramma's.

Een belangrijke klasse superprogramma's was de klasse van het interpreterende programma: programma's die 'het objectprogramma wezenlijk kunnen interpreteren, d.w.z. met inachtneming van de werking van het objectprogramma.'[24] In de cursus werden drie verschillende soorten interpreterende programma's onderscheiden: interpreters die machinecode in het geheugen interpreteerden, interpreters die objectcode (anders dan machinecode) in het geheugen interpreteerden en interpreters die objectcode vanuit de invoer omzette in machinecode instructies.

De eerste soort interpreterende programma's vormden samen met de machine waarop het draaide een nieuwe machine die vergelijkbaar was met de originele machine. '[E]en pseudo-machine, die handelt als de echte machine, maar daar[b]ij nog voortdurend verslag uitbrengt van zijn handelingen als een neurotische patiënt aan de psychiater (nl. de programmeur). (...) Het interpreterende programma moet zelf een pseudo-machine bijhouden in de vorm van een aantal adressen gebruikt als pseudo-opdrachtteller, pseudo-rekenregister, pseudo-condities, enz.'[25]

De tweede soort interpretatieve programma's interpreteerde programma's die al in het geheugen van de machine aanwezig waren maar niet in de machine code gesteld waren. Deze programma's draaiend op een machine vormden een pseudo-machine met nieuwe functionaliteit, bijvoorbeeld het rekenen met drijvende komma. Omdat deze programma's traag waren, was het van belang om te kunnen wisselen van interpretatieve modus naar niet-interpretatieve modus: een groot deel van een programma bestond uit administratieve handelingen, en die konden net zo goed op de echte machine uitgevoerd worden.

De laatste soort interpretatieve programma's werkte op programma's op ponsband. Deze programma's konden op twee wijzen verwerkt worden: het programma werd vertaald naar een equivalent machine code programma of het de programmaregels en voert die uit[26]. Het invoerprogramma was het voorbeeld van de eerste verwerkingswijze. De programmeurscode werd omgezet in machinecode en het programma werd in het geheugen gezet waarna het uitgevoerd kon worden.

[24] Ibidem, 103
[25] Ibidem, 104
[26] Dit is dus eigenlijk het latere onderscheid tussen compilers en interpreters.

Het hoofdstuk over superprogramma's werd besloten met: 'In het algemeen kan men zeggen, dat een schier eindeloze hoeveelheid "kennis" in de superprogramma's kan worden opgeslagen, zodat het converseren met de machine meer het karakter krijgt van het praten tot een collega in plaats van tegen een imbeciele slaaf.'[27]

Uit deze cursus blijkt wat men verwachtte van het programmeren van een machine, welke onderdelen belangrijk waren, welke methoden er waren om een programma op te stellen, wat voor soorten programma's er zijn, aanstippen wat er mogelijk was. Maar het geeft geen inzicht in hoe er nu eigenlijk geprogrammeerd werd, hoe verschillende mensen met een probleem hebben zitten worstelen, wat de ontwikkelingscyclus was, hoe vaak er getest werd, hoe lang gemiddeld een programma was, enzovoorts. Deze cursus en rapporten geven een geïdealiseerd beeld van het programmeren, de dagelijkse gang van zaken blijft verborgen.

5 Conclusie

Tussen 1951 en 1958 werd het programmeren duidelijk een aparte taak met zijn eigen moeilijkheden. Sterker nog, doordat de machines sterk verbeterden, denk aan snel geheugen, buffers, betere in- en uitvoer, ingebouwde controles, enzovoorts, werden de machines steeds betrouwbaarder en gebruiksvriendelijker en daardoor ook steeds bruikbaarder. Het aantal opdrachten dat met de opeenvolgende machines uitgevoerd werd groeit steeds sneller en daardoor werd ook het aandeel van het programmeren en het aantal programmeurs steeds groter. Veel van die programmeurs kwamen niet van de Rekenafdeling. Het programmeren en het denken over programmeren werd daardoor steeds belangrijker.

Het is een zichzelf versterkende dubbele ontwikkeling van machine en programmeren: door ontwikkeling van de machine werd het programmeren makkelijker en belangrijker; door de ontwikkeling van het programmeren, van het groeiende gebruik van de machines was het belangrijk om een nieuwe en betere machine te verkrijgen. Het programmeren als zodanig veranderde niet fundamenteel: de basis was de machine met een machinecode. Er werden allerhande hulpmiddelen in hardware en in programma's ontwikkeld, maar het programmeren kon zich niet niet boven het machineniveau verheffen waardoor de logische structuur van de oplossing verborgen bleef in het programma.

Referenties

'Jaarverslag Mathematisch Centrum' (1950).

'Jaarverslag Mathematisch Centrum' (1952).

Bakker, N.C., 'Programmering voor de ARMAC, 3a; Interpretatief programma voor complexe getallen met drijvende komma's', Technisch rapport MR-27a (Amsterdam: Mathematisch Centrum 1957), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9260A.pdf).

[27] Ibidem, 106

Dannenburg, H.J., 'De berekening van eigenwaarden en -vectoren van matrices op de FERTA', in: *Nederlands rekenmachine genootschap 1959 – 1969. Colloquium Moderne Rekenmachine II* (Amsterdam 1969), 24 november 1956, 90–97.

Dekker, T.J., 'Programmering voor de ARMAC, 6; Matrix complex RAM', Technisch rapport MR-30 (Amsterdam: Mathematisch Centrum 1959), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9257A.pdf).

Dekker, T.J., E.W. Dijkstra en A. van Wijngaarden, 'Cursus programmeren voor automatische rekenmachines', Technisch rapport CR-9 (Amsterdam: Mathematisch Centrum 1957), (URL:http://www.cs.utexas.edu/users/EWD/MCReps/CR1957-009.PDF).

Dijkstra, E.W., 'Functionele beschrijving van de ARRA', Technisch rapport MR-12 (Amsterdam: Mathematisch Centrum 1953), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9277A.pdf).

Dijkstra, E.W., "'Drijvende komma" rekentechniek (ARRA subroutines RD1 en RD2)', Technisch rapport MR-16 (Amsterdam: Mathematisch Centrum 1954), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9273A.pdf).

Dijkstra, E.W., 'In- en uitvoer van de ARRA', Technisch rapport MR-14 (Amsterdam: Mathematisch Centrum 1954), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9275A.pdf).

Dijkstra, E.W., 'Handboek voor de programmeur (FERTA) I', Technisch rapport MR-17 (Amsterdam: Mathematisch Centrum 1955), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9272A.pdf).

Dijkstra, E.W., 'Handboek voor de programmeur (FERTA) II', Technisch rapport MR-20 (Amsterdam: Mathematisch Centrum 1955), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9269A.pdf).

Dijkstra, E.W., 'Het communicatieprogramma van de ARRA', Technisch rapport MR-21 (Amsterdam: Mathematisch Centrum 1955), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9268A.pdf).

Dijkstra, E.W., 'Het standaard typprogramma van ARMAC', Technisch rapport MR-24 (Amsterdam: Mathematisch Centrum 1956), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9265A.pdf).

Dijkstra, E.W., 'Korte beschrijving van de opdrachtencode etc. voor de ARMAC', Technisch rapport MR-23 (Amsterdam: Mathematisch Centrum 1956), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9266A.pdf).

Dijkstra, E.W., 'Programmering voor de ARMAC, 1; Algemeen', Technisch rapport MR-25 (Amsterdam: Mathematisch Centrum 1956), (URL:http://www.cs.utexas.edu/users/EWD/MCReps/MR25.PDF).

Dijkstra, E.W., 'Programmering voor de ARMAC, 2; De inhoud der geblokkeerde kanalen', Technisch rapport MR-26 (Amsterdam: Mathematisch Centrum 1956), (URL:http://www.cs.utexas.edu/users/EWD/MCReps/MR26.PDF).

Dijkstra, E.W., 'Programmering voor de ARMAC, 3; Interpretatief programma voor drijvende komma berekening', Technisch rapport MR-27 (Amsterdam: Mathematisch Centrum 1956), (URL:http://www.cs.utexas.edu/users/EWD/MCReps/MR27.PDF).

Dijkstra, E.W., 'Programmering voor de ARMAC, 1a; Algemeen', Technisch rapport MR-25 a (Amsterdam: Mathematisch Centrum 1957), (URL:http://www.cs.utexas.edu/users/EWD/MCReps/MR25a.PDF).

Dijkstra, E.W., 'Programmering voor de ARMAC, 5; Interpretatief programma voor breuken van dubbele lengte', Technisch rapport MR-29 (Amsterdam: Mathematisch Centrum 1957), (URL:http://www.cs.utexas.edu/users/EWD/MCReps/MR29.PDF).

Loopstra, B.J. en C.S. Scholten, 'Constructie van automatische rekenmachines. Cursus 1955/56', Technisch rapport CR-6 (Amsterdam: Mathematisch Centrum 1956).

Vasmel-Kaarsemaker, L., 'Programmering voor de ARMAC, 4; Interpretatief programma voor het werken met 6-voudige lengte getallen', Technisch rapport MR-28 (Amsterdam: Mathematisch Centrum 1957), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9259A.pdf).

Wijngaarden, A. van, 'Programmeren voor de ARRA', Technisch rapport MR-7 (Amsterdam: Mathematisch Centrum 1951), (URL:http://repos.project.cwi.nl:8888/cwi_repository/docs/I/09/9282A.pdf).