Als "max" wordt aangeroepen bijvoorbeeld in:

```
y := max(i, rij)
```

wordt aan "y" de waarde van het maximale element van de aktuele "rij" toegekend, terwijl de index waarvoor dit maximum wordt bereikt in "i" is te vinden.
Deze "i" uit de aanroep (een GP uit het hoofdprogramma, of een andere FUN-beschrijving!) heeft n i e t s te maken met de "i" van de beschrijving "max".

Als "max" wordt aangeroepen als [opdracht]:

```
max(u,rij)
```

gooien we de waarde van het maximale getal van "rij" weg, maar die kunnen we altijd weer terugvinden door de opdracht:

```
grootste := rij(u)
```

Als "max" wordt aangeroepen bijvoorbeeld als in:

```
y := max(0,rij)
```

wordt de aktuele waarde 0 in een voor de programmeur onbereikbare GP gestopt. Deze onbereikbare GP gaat bij het doorlopen van "max" nu fungeren als hulp-GP voor de formele "index", maar de laatste waarde daarvan ben je wel kwijt als de funktie klaar is!

25. Recursiviteit

Funkties mogen ook "in recursie" gaan, dat wil zeggen: zichzelf (hetzij direkt of via een andere funktie) aanroepen. Bij recursief gebruik kan een funktie dus in verschillende "incarnaties" (of "diepten") worden geaktiveerd. Er komt dan nog een regel bij:

C25.1 Als een funktie in recursie wordt aangeroepen, worden voor alle lokale-, niet-formele [GP]'s, nieuwe geheugenplaatsen gereserveerd, die niets te maken hebben met de geheugenplaatsen die voor die [GP]'s in enige vorige incarnatie waren gereserveerd.

Dit recursie mechanisme is een afspiegeling van en kan gebruikt worden bij rekenmachine implementatie voor recursieve processen in de wiskunde. Een heel bekend voorbeeld van zo'n recursief wiskundig proces is de algoritme van Euclides voor het bepalen van de grootste gemene deler van twee (niet noodzakelijk positieve) gehele getallen m en n. In woorden luidt deze algoritme:

als $n = 0$ dan is de ggd van m en n het getal m
als $n \neq 0$ dan is de ggd van m en n hetzelfde als de ggd van n en de rest die ontstaat bij deling van m door n.

Men vindt dit vrijwel letterlijk terug in onderstaande FUN-beschrijving. Alleen, omdat men in een FUN-beschrijving niet tweemaal mag zeggen FUN :=--- vindt men op regel 2 een toekenning aan m, die op dat moment toch overbodig is en als hulpvariabele dienst doet.

```
  FUN ggd(m,n)
1 ALS n = 0 DAN 3
2 m := ggd(n,m-n × G(m/n))   ; m - n×G(m/n) is rest bij deling van m door n
3 FUN := ABS(m)              ; we maken de ggd positief
```

VI DE ECOL-SYNTAX

In de syntax wordt min of meer vastgelegd welke teksten wel, en welke niet door de compiler als correct ECOL worden geaccepteerd. Hierbij maken we gebruik van zg."produktie-regels", zoals [5], waarvan de betekenis geen toelichting zal behoeven, en waarbij een ieder zich wel kan voorstellen welke letters niet expliciet opgeschreven zijn.
In [8] is ruimte gelaten voor eventueel nog enkele leestekens die mogelijk niet op elk apparaat (of schrapkaart) produceerbaar zijn; de gegeven leestekens zijn de minimaal vereiste.
Vele produktieregels produceren een opeenvolging van syntactische eenheden, zoals [55]. Men moet dit zo lezen dat een [RIJ-reservering] bestaat uit het woordje RIJ, gevolgd door een openingshaakje, waarna iets komt dat door de produktieregel voor [uitdr] (zie [34]) geproduceerd wordt, waarna dubbelpunt etc.
In een aantal produktieregels komt een "exponent" n of n+1 voor. Bijvoorbeeld in [12], waar dit aanduidt dat een [naam] een opeenvolging is van n+1 [kleine letter]s . Steeds is n = 0,1,2,3,...
Een [naam] bestaat dus uit minstens één kleine letter.
Evenzo betekent [opdracht]^n (zie [86]) dat er n [opdracht]en na elkaar mogen komen, maar omdat n ook 0 mag zijn mag er ook helemaal niks staan. Als er meer dan één [opdracht] staat moeten deze vanzelf onder elkaar komen te staan, omdat krachtens [23] elke opdracht wordt beëindigd met □ (zie hiervoor [16]).
In enkele gevallen tenslotte is een produktieregel recursief, zoals bijv. [88]; men moet dit zo lezen:

[formelen] is [formele] of [formele],[formele]
of [formele],[formele],[formele] etc

Bij de recursie kunnen meerdere produktieregels betrokken zijn, zoals het geval is in [34] t/m [37].

Op vanzelfsprekende wijze worden in de hoofdstukken I t/m V van dit rapport verwijzingen naar syntactische begrippen ook in de meervoudsvorm gegeven, dus [namen],[labels], e.d.

Voor apparaten met een kleinere symbolen-voorraad dan minimaal voor ECOL vereist (bijv. geen onderscheid tussen [kleine letter] en [HOOFDLETTER]), kunnen eenvoudige conventies worden opgesteld, die het toch mogelijk maken ECOL-programma's via een dergelijk apparaat in te voeren. Deze conventies zijn echter veelal machine-, althans implementatie-gebonden en worden hier verder buiten beschouwing gelaten.

Spatieconventie

In de syntax is eenvoudigheidshalve niet tot uitdrukking gebracht dat overal in een [ECOL-programma] spaties mogen worden toegevoegd zonder dat daardoor de betekenis van het programma verandert, behalve in een [hoofdletterwoord] (= [HOOFDLETTER]$^{n+1}$) en tussen de aanhalingstekens achter TEKST:= . Er moet echter minstens één spatie staan wanneer twee [hoofdletterwoord]en direct op elkaar volgen.

[1] [sym]	is	[↩] of [symbool]
[2] [↩]	is	[overgang op nieuwe regel, of einde kaart]
[3] [symbool]	is	[cijfer] of [kleine letter] of [HOOFDLETTER]
		of [teken] of [spatie]
[4] [cijfer]	is	0 of 1 of 2 of 3 of 4 of 5 of 6 of 7 of 8 of 9
[5] [kleine letter]	is	a of b of c of of x of y of z
[6] [HOOFDLETTER]	is	A of B of C of of X of Y of Z
[7] [teken]	is	[leesteken] of [operator] of [vergelijker]
[8] [leesteken]	is	. of , of : of ; of ? of ! of (of) of of "
[9] [operator]	is	+ of - of × of / of M
[10] [vergelijker]	is	= of < of > of ≤ of ≥ of ≠

```
[11] [niks]                 is
[12] [naam]                 is   [kleine letter]^(n+1)
[13] [NAAM]                 is   [HOOFDLETTER]^(n+1)
[14] [label]                is   [cijfer]^(n+1)
[15] [nummer]               is   [label] of [label]:   of  [niks]
[16] ▯                      is   [↩]  of  ; [symbool]^n [↩]
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
[20] [ECOL-programma]       is   [FUN-beschrijving]^n
                                 [nummer] START ▯
                                 [programmadeel]^n
                                 [nummer] KLAAR ▯
[21] [programmadeel]        is   [aktie] of [SUB-beschrijving]
[22] [aktie]                is   [opdracht]
                               of [RIJ-reservering]
[23] [opdracht]             is   [nummer][toekenning] ▯  of [nummer][invoer/uitvoer] ▯
                               of [nummer][sprong] ▯      of [nummer][cyclus] ▯
                               of [nummer][SUB-roep] ▯    of [nummer][FUN-roep] ▯
                               of [nummer] ▯
[30] [toekenning]           is   [variabele] := [uitdr]
[31] [variabele]            is   [GP] of [GP-rij]([uitdr])
[32] [GP]                   is   [naam]
[33] [GP-rij]               is   [naam]
[34] [uitdr]                is   [voorteken][formule]
[35] [voorteken]            is   [niks] of + of -
[36] [formule]              is   [operand] of [formule][operator][operand]
[37] [operand]              is   [getal] of [variabele] of [FUN-roep] of ([uitdr])
[38] [getal]                is   [cijfer]^(n+1) of [cijfer]^(n+1) , of [cijfer]^(n+1) . [cijfer]^(n+1)
[40] [invoer/uitvoer]       is   [variabele] := LEES
                               of [variabele] := LEESSYM
                               of [GP-rij] := BESTAND([uitdr])
                               of [GP-rij] := BESTAND([uitdr],[uitdr],[uitdr])
                               of  SPATIE of SPATIE([uitdr])
                               of  NR     of     NR([uitdr])  of  NP
                               of  TEKST := "[symbool]^(n+1)"
                               of  SCHRIJF    := [uitdr] of  SCHRIJM := [uitdr]
                               of  SCHRIJFSYM := [uitdr]
                               of  SCHRIJF([uitdr],[uitdr]) := [uitdr]
                               of  BESTAND([uitdr])                 := [GP-rij]
                               of  BESTAND([uitdr],[uitdr],[uitdr]) := [GP-rij]
```

```
[45] [sprong]                 is   NAAR [label]
                                 of ALS[vergelijking] DAN [label] ANDERS [label]
                                 of ALS[vergelijking] DAN [label]
[50] [vergelijking]           is   [uitdr][vergelijker][uitdr]
                                 of [vergelijking] EN [vergelijking]
                                 of [vergelijking] OF [vergelijking]
                                 of ([vergelijking])
[55] [RIJ-reservering]        is   RIJ   ([uitdr] : [uitdr])[GP-rij] []
                                 of RIJSYM([uitdr] : [uitdr])[GP-rij] []
[60] [cyclus]                 is   MET[uitdr], [GP] := [uitdr], [uitdr]
                                   [programmadeel]^n
                                   [nummer]HERHAAL
[70] [SUB-roep]               is   [SUB-naam]
[71] [SUB-beschrijving]       is   [nummer]SUB[SUB-naam] []
                                   [opdracht]^n
                                   [nummer]END []
[72] [SUB-naam]               is   [naam]
[80] [FUN-roep]               is   [FUN-NAAM][lijst aktuelen] of [FUN-NAAM]
                                 of [FUN-naam][lijst aktuelen]
[81] [FUN-NAAM]               is   [NAAM]
[82] [FUN-naam]               is   [naam]
[83] [lijst-aktuelen]         is   ([aktuelen])
[84] [aktuelen]               is   [aktuele] of [aktuelen] ,[aktuele]
[85] [aktuele]                is   [uitdr] of [GP-rij]
[86] [FUN-beschrijving]       is   [nummer]FUN[FUN-naam][lijst formelen] []
                                   [opdracht]^n
                                   [nummer]FUN := [uitdr] []
                                 of [nummer] []
[87] [lijst formelen]         is   ([formelen])
[88] [formelen]               is   [formele] of [formelen],[formele]
[89] [formele]                is   [GP] of RIJ[GP-rij]
```

INDEX

adres 6 10.2 24
afronden 9
aktuele 10,3 22
binnencyclus 14
compiler 2 6 7
context regels 1 7 8 20 23
cyclus 14 20
descriptor 6 13 24
doeladres 11
EN 12
executie 3 9
formele 22 23
funktiedeel 4 23
FUN 19 22
FUN-beschrijving 4 7 8 12 14 22 23
FUN-roep 7 8 10.3 22 23 24
geheugenplaats, GP 6
getal 10.1
hoofdprogramma 4 7 8 20 23
index 10.2
index expressie 10.2
inhoud van GP 6
invoer 16
kommentaar 5.2
label 5.1 5.2 8 22 23
lokaal 22
naam 7 22 23
OF 12
ongereserveerd 13 21
ONWAAR 12
parameter 23
primitiva 10.5
prioriteit 10.4
produktie-regel VI
recursie 25 VI
RIJ,RIJSYM,RIJ-reserv. 3 6 7 13 14 23
spatie 5.3 VI
sprong 8 12
standaardfunkties 18 19
SUB 19
SUB-beschrijving 3 7 8 12 14 20 23
SUB-roep
subroutine 20
syntax 1 VI
tabel van symbolen 16
tekstuele volgorde 2 7 23
toekenning 11
uitdrukking 10
uitvoer 17
variabele 6 10.2
vertaalfase 2
WAAR 12
waarde 6 9 10.2 10.4

In deze index zijn de nummers paragraafnummers. Als naar meerdere nummers verwezen wordt zijn de belangrijkste onderstreept.

4 Wetenswaardigheden

4.1 LITERATUURBESPREKING

In het volgende worden in het kort enkele boeken besproken, die tot de redelijk eenvoudige leesstof gerekend moeten worden.
Tot en met 4.1.13 is Drs. C.A.Ch. Görts Uw redakteur.

4.1.1 *'Computerwiskunde'* f 3,50

van Prof.Dr.J.J. Seidel
uitgave Het Spectrum, Utrecht - Antwerpen
Aula boeken 407 - Nederlands - 155 blz.

Algemene inleiding over wiskundige hulpmiddelen die bij de computer van belang zijn. Het boek is ontstaan als heroriënteringscursus van de CMLW in 1967-1968.

Inhoud:
1) Algoritmen (Prof.Dr. A. v.d. Sluis)
2) Numerieke toepassingen (Prof.Dr.A.v.d.Sluis)
3) Non numerieke computertoepassingen (Prof.Dr. N.G. de Bruijn)
4) Over de structuur van rekenautomaten (Prof.Dr. E.W. Dijkstra).

4.1.2 *'Assemblers and Loaders'*

van D.W. Barron
uitgave MacDonald, Londen
Engels - 62 blz. - prijs f 14,25

Bij het programmeren in machinetaal schrijft de programmeur niet de bit-patronen op van de instructies, maar kan hij gebruik maken van lettercodes om het soort instructie aan te geven, van decimale getallen voor de adressen enz. Dit boek behandelt de vertaling van zulke machinetaal in de uiteindelijke bitpatronen. Enige kennis van de structuur van machinetalen wordt vereist.

4.1.3. *'Compiling Techniques'*

van F.R.A. Hopgood
Engels - 115 blz. - prijs f 19,60

Compiling is het vertalen van een programma geschreven in een probleemgeoriënteerde taal (b.v. ALGOL) naar machinetaal. Dit boek geeft in een compacte, en daarom niet altijd gemakkelijke stijl een inleiding op dit terrein. Het boek veronderstelt bij de lezer enige kennis op het gebied van programmeertalen.

4.1.4 *'Digitale automaten'*

van W. Kämmerer
uitgave Akademie Verlag Verlin 1969
Duits - 482 blz. - prijs f 40,15

Een degelijk studieboek, bedoeld om een leek een basis te geven voor een werkkring op computergebied.

Inhoud:
1) Theorie (theoretische begrippen als berekenbaarheid, Turing machine, Boolese algebra)
2) Struktur (voorstelling van informatie in de computer, arithmetiek, structuur van computer, opdrachten)
3) Technik (poorten, flip flops, kernen magneetbanden enz.)
4) Programmieren (programmeren in machinetaal; hogere programmeertalen en hun vertaling in machinetaal).

4.1.5 *'Inleiding tot computer en automatisering'*

van Prof.Dr. M. Euwe
uitgave Samson n.v., Alphen/Rijn, Brussel 1966
Nederlands - 126 blz. - prijs f 9,30

Algemene oriëntatie in computerwereld. Tevens bedoeld als leerboek bij het MEAO.

Inhoud:
1) Het nauwkeurig formuleren
2) Probleemanalyse
3) Schematechnieken (blokschema's)
4) De computer (opbouw van computer; rekenen in binair stelsel)
5) Het programma (beginselen van programmeren in de machinetaal van de denkbeeldige machine SERA. Deze taal wordt veel gebruikt bij cursussen van het Studie Centrum voor Informatica).
6) Invoermiddelen (ponskaart, ponsband, magneetband).

4.1.6. *'List processing'*

van J.M. Foster
uitgave MacDonald, Londen
Engels - 54 blz. prijs f 12,40

List processing is een programmeertechniek, waar-

INDEX

adres 6 10.2 24
afronden 9
aktuele 10,3 22
binnencyclus 14
compiler 2 6 7
context regels 1 7 8 20 23
cyclus 14 20
descriptor 6 13 24
doeladres 11
EN 12
executie 3 9
formele 22 23
funktiedeel 4 23
FUN 19 22
FUN-beschrijving 4 7 8 12 14 22 23
FUN-roep 7 8 10.3 22 23 24
geheugenplaats, GP 6
getal 10.1
hoofdprogramma 4 7 8 20 23
index 10.2
index expressie 10.2
inhoud van GP 6
invoer 16
kommentaar 5.2
label 5.1 5.2 8 22 23
lokaal 22
naam 7 22 23
OF 12
ongereserveerd 13 21
ONWAAR 12
parameter 23
primitiva 10.5
prioriteit 10.4
produktie-regel VI
recursie 25 VI
RIJ,RIJSYM,RIJ-reserv. 3 6 7 13 14 23
spatie 5.3 VI
sprong 8 12
standaardfunkties 18 19
SUB 19
SUB-beschrijving 3 7 8 12 14 20 23
SUB-roep
subroutine 20
syntax 1 VI
tabel van symbolen 16
tekstuele volgorde 2 7 23
toekenning 11
uitdrukking 10
uitvoer 17
variabele 6 10.2
vertaalfase 2
WAAR 12
waarde 6 9 10.2 10.4

In deze index zijn de nummers paragraafnummers. Als naar meerdere nummers verwezen wordt zijn de belangrijkste onderstreept.

4 Wetenswaardigheden

4.1 LITERATUURBESPREKING

In het volgende worden in het kort enkele boeken besproken, die tot de redelijk eenvoudige leesstof gerekend moeten worden.
Tot en met 4.1.13 is Drs. C.A.Ch. Görts Uw redakteur.

4.1.1 *'Computerwiskunde'* f 3,50

van Prof.Dr.J.J. Seidel
uitgave Het Spectrum, Utrecht - Antwerpen
Aula boeken 407 - Nederlands - 155 blz.

Algemene inleiding over wiskundige hulpmiddelen die bij de computer van belang zijn. Het boek is ontstaan als heroriënteringscursus van de CMLW in 1967-1968.

Inhoud:
1) Algoritmen (Prof.Dr. A. v.d. Sluis)
2) Numerieke toepassingen (Prof.Dr.A.v.d.Sluis)
3) Non numerieke computertoepassingen (Prof.Dr. N.G. de Bruijn)
4) Over de structuur van rekenautomaten (Prof.Dr. E.W. Dijkstra).

4.1.2 *'Assemblers and Loaders'*

van D.W. Barron
uitgave MacDonald, Londen
Engels - 62 blz. - prijs f 14,25

Bij het programmeren in machinetaal schrijft de programmeur niet de bit-patronen op van de instructies, maar kan hij gebruik maken van lettercodes om het soort instructie aan te geven, van decimale getallen voor de adressen enz. Dit boek behandelt de vertaling van zulke machinetaal in de uiteindelijke bitpatronen. Enige kennis van de structuur van machinetalen wordt vereist.

4.1.3. *'Compiling Techniques'*

van F.R.A. Hopgood
Engels - 115 blz. - prijs f 19,60

Compiling is het vertalen van een programma geschreven in een probleemgeoriënteerde taal (b.v. ALGOL) naar machinetaal. Dit boek geeft in een compacte, en daarom niet altijd gemakkelijke stijl een inleiding op dit terrein. Het boek veronderstelt bij de lezer enige kennis op het gebied van programmeertalen.

4.1.4 *'Digitale automaten'*

van W. Kämmerer
uitgave Akademie Verlag Verlin 1969
Duits - 482 blz. - prijs f 40,15

Een degelijk studieboek, bedoeld om een leek een basis te geven voor een werkkring op computergebied.

Inhoud:
1) Theorie (theoretische begrippen als berekenbaarheid, Turing machine, Boolese algebra)
2) Struktur (voorstelling van informatie in de computer, arithmetiek, structuur van computer, opdrachten)
3) Technik (poorten, flip flops, kernen magneetbanden enz.)
4) Programmieren (programmeren in machinetaal; hogere programmeertalen en hun vertaling in machinetaal).

4.1.5 *'Inleiding tot computer en automatisering'*

van Prof.Dr. M. Euwe
uitgave Samson n.v., Alphen/Rijn, Brussel 1966
Nederlands - 126 blz. - prijs f 9,30

Algemene oriëntatie in computerwereld. Tevens bedoeld als leerboek bij het MEAO.

Inhoud:
1) Het nauwkeurig formuleren
2) Probleemanalyse
3) Schematechnieken (blokschema's)
4) De computer (opbouw van computer; rekenen in binair stelsel)
5) Het programma (beginselen van programmeren in de machinetaal van de denkbeeldige machine SERA. Deze taal wordt veel gebruikt bij cursussen van het Studie Centrum voor Informatica).
6) Invoermiddelen (ponskaart, ponsband, magneetband).

4.1.6. *'List processing'*

van J.M. Foster
uitgave MacDonald, Londen
Engels - 54 blz. prijs f 12,40

List processing is een programmeertechniek, waar-

bij op objecten in het geheugen wordt geopereerd met inachtneming van de relaties, die er tussen de objecten bestaat. Voor deze techniek, die in niet-numerieke toepassingen veel gebruikt wordt, zijn speciale programmeertalen ontwikkeld. Ook verschillende algemene programmeertalen bieden faciliteiten op het gebied van list processing. Dit boekje veronderstelt enige basiskennis op het gebied van programmeren.

4.1.7. *'Primer of computing'*
van Donald Leach en Bob Bowker
uitgave Holmes Mc.Dougall, Edinburg
Engels - 127 blz. - prijs onbekend

Hierin wordt iets verteld over de geschiedenis van de computer, hardware, programmeren in een eenvoudige machinecode, programmeertalen, toepassingen.
Er wordt niet diep op de zaken ingegaan. Misschien aardig voor de leerlingenbibliotheek (als de taal hiervoor geen bezwaar is).

4.1.8 *'Inleiding tot IBM informatie verwerkende systemen'*
Uitgave IBM
Nederlands - 107 blz. - prijs ± *f* 40,--

Deze handleiding is bedoeld als hulpmiddel bij het zich eigen maken van een basiskennis van computers. Besproken worden o.a. computertalstelsels, geheugeneenheden en hun werking, in- en uitvoereenheden, schrijven van programma's, programmeersystemen.
Uit dit uiteraard hier en daar op IBM apparatuur gericht boek zal men heel wat achtergrondinformatie kunnen halen. Het is voor serieuze belangstellenden, waaronder ook de experimenterende leraren worden gerekend, gratis verkrijgbaar bij IBM Nederland N.V., Afd. Publiciteit (t.a.v. N.A.M. v.d. Broek), Joh.Huizingalaan 257, Amsterdam.

4.1.9 *'Elektronische Rekenmachines'*
van P.L. Latour
uitgave Marka Boeken, Utrecht/Antwerpen
Nederlands - 218 blz. - prijs *f* 4,--

Een heel bruikbaar boek om wat achtergrondinformatie te verkrijgen. Behandelt o.a. de opbouw van de computer, machine-opdrachten, elektronische bouwstenen, constructie van geheugens, in- en uitvoerorganen, algorithmische talen, analoge rekenmachines, kunstmatige intelligentie.

4.1.10 *'Vertalen met de computer'*
van Dr. F.G. Droste
uitgave Wolters Noordhoff N.V., Groningen - 1969
Nederlands - 169 blz. - prijs *f* 17,25

Een inleiding tot wat er in verschillende centra gedaan is aan machinale vertaling. Vakkennis op dit gebied noch op dat van computers is vereist.

4.1.11 *'Hoe werkt een computer'*
van R. Lohberg *f* 15,75
uitgave Kluwer, Deventer/Antwerpen
Nederlands (uit Duits vertaald) - 235 blz.

Behandelt o.a. onderdelen van computer en hun werking, opdrachten, toepassingen. Gaat niet diep, maar bevat wel nuttige informatie. Maar de poging de zaak lichtverteerbaar te maken door een gezocht humoristische stijl van schrijven werkt wel storend.

4.1.12 *'Computers and problem solving'*
van T.E. Hull en D.D.F. Day
uitgave Addison Wesley, Canada - 1970
Engels - 276 blz. - prijs *f* 24,35

Het doel van dit boek is een leek de grondslagen van computergebruik bij te brengen. In het eerste deel wordt men vertrouwd gemaakt met programmeren; de gekozen programmeertaal is FORTRAN. Het tweede deel bevat een aantal toepassingen; deze zijn niet altijd volledig uitgewerkt; men kan dan desgewenst aan de hand van opgaven dieper op de zaak ingaan. De onderwerpen in deel II zijn: numerieke berekeningen, statistiek, simulatie, data processing, niet-numerieke berekeningen, toepassingen in computer science. In appendices komen o.a. ter sprake programmeren in assembleertaal en voorstelling van getallen. Bestudering van dit prettig geschreven boek, met name van deel II en de appendices, is ongetwijfeld erg nuttig, vooral als men bereid is ook de opgaven goed te bekijken.

4.1.13 *'Inleiding tot de automatische informatieverwerking'*
van A.B. Frielink
uitgave Agon Elsevier, Amsterdam/Brussel 1968
Nederlands - 118 blz. - prijs *f* 17,50

Een belangrijk gedeelte van dit boek wordt besteed aan de opbouw van de computer, de voorstelling van data in het tweetallig stelsel, het rekenorgaan en het programmeren. Bij het programmeren wordt de taal SERA gebruikt, een machinetaal voor een fictieve computer, ontworpen ten behoeve van de programmeursopleiding. Daarnaast komen onderwerpen aan de orde als voorbereiding van een computertoepassing, typen van computers, invoerproblematiek, organisatie rekencentrum, historisch overzicht, theoretische grondslagen. Waarschijnlijk omdat de auteur in een kleine omvang veel heeft willen aansnijden, zijn de meeste onderwerpen wel erg beknopt behandeld. Het boekje kan van nut zijn als men het vak eens van een andere zijde (administratief gebruik) belicht wil zien.

4.1.14 *'Eenvoudige Computerkunde'*
van A. Goudvis
uitgave J. Rijkhoek, Den Haag
Nederlands - 110 blz. - prijs *f* 5,--.

De schrijver geeft 110 pastelkleurige bladzijden informatie over de computer en zijn toepassingen. Na een kort historisch overzicht en enige instructie over het tweetallig stelsel volgt een opsomming van functie en werking van veel toestellen, die men in de omgeving van een computer kan aantreffen. Daarna volgt de organisatie van de rekenautomaat en iets over de wijze, waarop de machine door de mens wordt geïnstrueerd. De schrijver kan in een zo kort bestek over de onderwerpen niet volledig zijn, maar hij doet - gezien het grote aantal ingevoerde begrippen wel een poging in die richting. Daarna spitst het betoog zich toe op het aanbrengen van een juist begrip van de toepasbaarheid van computers in het bedrijfsleven. Het boek is volgens de ondertitel bedoeld als leerboek in computerkunde voor de hoogste klassen van de middelbare school en voor de H.T.S. Als leerboek bevat het te veel gegevens in een kort bestek en te weinig (namelijk helemaal geen) oefenmateriaal om instructief te zijn. Het leervak 'computerkunde' zal meer moeten bevatten dan de door de schrijver gegeven feitenkennis van computers. Het zal in de eerste plaats de - ook buiten de automatisering belangrijke - vaardigheden moeten aanbrengen op operationeel (duidelijke, ondubbelzinnige instructies geven),algoritmisch (herkennen van routinematigheden) en organisatorisch gebied. Aan deze facetten is de schrijver helaas voorbijgegaan. Dit neemt niet weg, dat het boek aanbevolen kan worden als leesboek voor de leerlingenbibliotheek.

4.1.15 *'Elementaire computerkunde voor mavo en havo'* deel I en II
Drs. W. van den Camp, H.B. Emanuels, J.B. Lubke, P.Th.J. Ploeger
Uitg. Meulenhoff, Amsterdam.
f. 7.90 en *f*. 13.90

De inhoud van beide delen is in grote lijnen als volgt:
Het eerste deel begint met een elementaire uiteenzetting over wat een computer is en hoe deze werkt; vervolgens wordt het begrip algoritme besproken, waarna verder veel aandacht wordt besteed aan programmastroomschema's enerzijds en (een subset van) SERA-69 anderzijds.
In het tweede deel wordt begonnen met een behandeling van ALGOL. Deze behandeling draagt het karakter van een 'eerste' ronde cursus, waarbij de procedures in het geheel niet ter sprake komen. Verder worden verschillende voorbeelden uitgebreid besproken. Het tweede deel besluit met twee hoofdstukjes over de sociale aspekten van de automatisering.
Vooraan in het tweede deel doen de schrijvers een voorstel tot behandeling van de geboden leerstof in de vorm van een gemeenschappelijk basisstuk (met onderwerpen uit zowel het eerste als tweede deel), gevolgd door een verdere uitbouw in twee richtingen waarvan de een meer wiskundig en de ander meer administratief is gericht. Bij nadere beschouwing blijken de twee richtingen maar weinig van elkaar te verschillen. De splitsing doet dan ook wat geforceerd aan. Gezien de groepen, waarvoor de 'Elementaire computerkunde' is bedoeld lijkt deze splitsing ook niet erg zinvol.
Keuze van onderwerpen is wel zinvol en eventueel zelfs nodig, als deze dan maar geschiedt op basis van niveau en belangstelling van de groep in kwestie.
Afgezien van genoemd (detail-) bezwaar lijkt mij het voorstel alleszins de moeite van het overwegen waard. De schrijvers hebben in ieder geval de mogelijkheid er toe geschapen, door de lesstof in niet al te grote, duidelijk afgeronde,

eenheden te presenteren. Gelukkig wordt in het voorstel het SERA-gedeelte als een op zichzelf staande eenheid beschouwd. Het is namelijk zeer de vraag of SERA (zij het tot BABY of JR gereduceerd) in een cursus voor het secundaire onderwijs thuishoort.
Enig inzicht betreffende de werking in de computer, zo al nodig, kan m.i. op een meer eenvoudige manier worden bijgebracht.
Als opvallend goede onderdelen kunnen worden genoemd:

a. de behandeling van programmastroomschema's. Zowel in lesstof als vraagstukken is sprake van een didactisch goed doordachte opbouw. De schrijvers hebben blijkbaar bij het uittesten een schat van ervaring opgedaan. Verwacht kan worden dat leerlingen (en docenten!) aan de duidelijke, goed gedoseerde en rijkelijk genuanceerde weergave van deze ervaring een sterke motivatie tot het verwerven van eigen inzicht zullen ontlenen. Dit laatste geldt ook en zeker niet in mindere mate voor
b. de meer uitgebreide voorbeelden in het tweede deel. Zowel de afwisseling van onderwerpen als de manier van behandeling maken dat de kwalifikatie *boeiend* hier terecht van toepassing is. Vooral de manier waarop hierbij gebruik wordt gemaakt van stroomschema's, getuigt van de grote zorg waarmee de schrijvers deze voorbeelden en bijbehorende vraagstukken hebben samengesteld.

In het voorwoord van het tweede deel wordt gesteld dat beide delen een experimenteel karakter hebben. Mogelijk is het daaraan te wijten dat, onverlet de grote waardering voor het geheel en de meeste onderdelen, er toch ook enkele onderdelen zijn, die van de nodige kritische kanttekeningen moeten worden voorzien. De schrijvers verwachten (voorwoord deel I) dat de leerlingen grote delen van de stof zelfstandig kunnen doorwerken. Maar dan is het wel nodig dat, zeker waar het gaat om een eerste kennismaking, verwarring scheppende onduidelijkheden en niet waar te maken uitspraken worden vermeden. Het eerste heeft vooral betrekking op de hoofdstukken 3 (Wat is een computer) en 4 (Hoe werkt een computer) van het eerste deel, het laatste op de twee laatste hoofdstukken van het tweede deel; die handelen over de sociale aspekten en de automatisering.
De gesignaleerde tekortkomingen doet de overheersend goede kwaliteit van de 'Elementaire Computerkunde' nauwelijks geweld aan. De boekjes kunnen dan ook van harte worden aanbevolen, met name voor de in de titel genoemde schoolsoorten: Havo, Havo.

F. Remmen

4.1.16 '*Inleiding in de informatica*'
Drs. F. Remmen
Uitg. H.E. Stenfert Kroese N.V. Leiden
f. 10.--

Het hier besproken boek is ontstaan uit een syllabus voor eerstejaarsstudenten aan de H.E.A.O.-Arnhem.
Zoals het voorwoord aangeeft is het bedoeld voor het bijbrengen van de nodige basisbegrippen. Bijzonder prettig is dat naast de nederlandse benamingen van deze begrippen ook de engelse equivalenten worden gegeven.
Hoewel dit niet geheel in het voorwoord van het boek tot uitdrukking komt, is de behandelde stof in eerste instantie bestemd voor eerstejaars H.E.A.O.-studenten; daarnaast is het boek echter goed bruikbaar om basisbegrippen bij te brengen aan degenen die een eerste (theoretische) kennismaking wensen met de automatisering.
Het zelfstandig doorwerken van het boek acht ik niet goed mogelijk, ook vanwege het feit dat de grote verzameling opgaven, waarvan de antwoorden niet in het boek zijn opgenomen, een essentieel bestanddeel van het boek vormen.
Ik zou mijn waardering willen uitspreken over de korte en zakelijke opzet van het boek, geschreven in een prettige stijl. Het boek telt, inclusief register en literatuurlijst, 94 pagina's.
Er wordt in tegenstelling tot vele andere publikaties die beogen een inleiding te zijn, in deze 94 bladzijden een schat van gegevens overgedragen. Dat de compactheid somtijds bezwaren oplevert, zal echter ook duidelijk zijn.
Naar mijn mening komt het onderwerp systeemanalyse en -ontwerp nauwelijks aan de orde. Hoewel ik mij bewust ben dat het op het niveau van degenen voor wie dit boek bestemd is, een bijzonder moeilijk te behandelen onderwep is, vindt ik het een zo essentieel onderdeel van de informatica, dat het in een inleiding beslist niet mag ontbreken. Bovendien kan het dienen als een 'kader' waarin een groot aantal onderwerpen een plaats kunnen krijgen. Nu zijn onderwerpen als programmeren, systeemstroomschema's, e.d. wat moeilijk plaatsbaar. Bovendien krijgt de hardware een te zwaar accent. In het boek worden hier dan ook ruim 40 pagina's aan gewijd.
Aan de onderwerpen systeemstroomschema's en programmastroomschema's 14 pagina's.
Opvallend zijn de hoofdstukken Bestandsbeheer, als inleiding op de externe geheugens en het hoofdstuk Beslissingstabellen.
Programmeren komt slechts in een hoofdstukje van

6 pagina's aan de orde. De (goede) reden die voor deze beknoptheid wordt gegeven, is dat de schrijver meent dat de keuze van de te behandelen programmeertaal momenteel nog te veel afhankelijk is van plaatselijke omstandigheden. Het leek de schrijver daarom niet zinvol in deze algemeen bedoelde inleiding een keuze te maken die voor sommigen (nog) niet bruikbaar zou zijn. Hij merkt daarbij nog op dat er op het gebied van de programmeertalen al diverse goede handleidingen bestaan. Het zou overigens prettig zijn geweest als de schrijver in zijn literatuurlijst een aantal handleidingen op het gebied van de programmeertalen had opgenomen.

Een ander onderwerp dat ik graag iets uitgebreider behandeld had willen zien, is het bedrijfssysteem van de computer. Hier en daar komt het wel aan de orde, maar gezien de belangrijke plaats die het bedrijfssysteem van de computer zo langzamerhand inneemt, wat te summier.
Samenvattend zou ik willen stellen dat het gedeelte van het boek dat de apparatuur behandelt, het sterkste deel van het boek is. Bovendien is een leuk, en toch kort, hoofdstukje over beslissingstabellen opgenomen.

P.Th.J. Ploeger

4.2 TERMINALGEBRUIK BIJ DE COMPUTERKUNDE

Wat is een terminal?

Het maakt de indruk van een schrijfmachine op een metalen tafeltje, meestal als één geheel uitgevoerd. Van de meeste merken is het schrijfmachine gedeelte van dezelfde soort als een elektrische schrijfmachine. Men ziet de schrijfkop langs het papier bewegen en symbolen op het papier slaan (Compk 1, foto op blz. 89).
Elk door de gebruiker aangeslagen symbool verschijnt alzo op het papier, maar wordt tevens omgezet in een soort Morse- code. En deze code wordt via een draad, meestal een telefoonlijn, verzonden naar een rekencentrum. In het rekencentrum worden de binnenkomende signalen opgevangen, door een apparaat dat in wezen een kleine computer is. Hierin worden de signalen onderzocht op defecten, die door een storing op de lijn ontstaan kunnen zijn. De code heeft namelijk een ingebouwde 'check' tegen verminkingen. Iets soortgelijks heeft bijvoorbeeld ook de ponsbandcode van Compk 1 blz. 37, waarbij het aantal gaatjes van een symbool (boven elkaar) altijd oneven is. Het ontbreken van een gaatje wordt ogenblikkelijk ontdekt. Maar het ontbreken van twee gaatjes wordt dan toch maar niet ontdekt, zegt U nu en gelijk hebt U. Als de kans op het ontbreken van een gaatje 1 op 1000 is, is de kans op 2 verdwenen gaatjes in eenzelfde teken 1 op 1000^2, en dat risiko neemt men. Overigens bestaan er codes, die zo zijn opgebouwd, dat zij lichte verminkingen overleven. Bij aankomst kan het gezonde signaal er weer uit worden afgeleid. Maar wij zijn wat afgedwaald naar de ziekenhuissfeer. Terug naar het rekencentrum.
Het computertje is niet alleen verbonden met onze terminal, maar ook met andere eindstations, die gelijktijdig (time sharing) zich mogen aanmelden. Maar het apparaat is ook geprogrammeerd om de verschillende stations uit elkaar te houden.

Ook het verzenden van informatie in omgekeerde richting is mogelijk, waarbij dan echter geen fout-detectie bij het eindstation mogelijk is. Een kraakje op de lijn geeft al gauw een onleesbare regel.

Hoe gaat de bediening in zijn werk ?

Men belt een bepaald nummer en hoort aan een bepaald toontje dat men aangesloten is op het computersysteem. Men legt de hoorn naast het telefoontoestel (of bij sommige portable terminals in een daarvoor bestemde opening) en schakelt de terminal in. Het eerste wat men in moet tikken is een geheim deelnemersnummer of iets dergelijks. Dit om te voorkomen dat iedereen die een terminal in elkaar knutselt naar willekeur van allerlei computers gratis gebruik zou maken. Na deze legitimatie en mogelijke andere formaliteiten kunt U beginnen met het intikken van programma's. Er zijn systemen waarbij symbool voor symbool wordt doorgezonden, andere waarbij eerst een hele regel wordt opgespaard in een geheugentje en daarna in zijn geheel verzonden. Zoals gezegd: bij verminkte overkomst krijgt U onmiddellijk reaktie op Uw toestel door middel van een foutmelding. U kunt de regel opnieuw tikken.
Als U Uw programma hebt voltooid en U laat een startsein volgen, dan wordt het programma aangeboden aan de computer. De uitkomsten verschijnen onmiddellijk na berekening op Uw terminal, tenzij fouten in het programma dat verhinderen, maar dan ontvangt U de foutmelding. Onmiddellijk herstel v fouten is dus mogelijk.

Voorbeeld van een programma

Hieronder ziet U een programma in de programmeertaal FORTRAN voor de berekening van een kwadraten tafel. READY geeft aan dat het systeem klaar staa

om de regel te ontvangen, die er achter wordt ingetikt. Dit is dus informatie door ons weggezonden. De terug ontvangen regels zijn aangegeven met YIELD. Het programma zou in ECOL luiden:

```
        START
        k := 2
        i := 2
    5   j := i × i
        SPATIE(2)
        SCHRIJF(2,0) := k
        NR
        SCHRIJF(3,0) := i
        SPATIE(5)
        SCHRIJF(6,0) := j
        ALS i=50 DAN 20
        i := i + 1
        NAAR 5
   20   KLAAR
```

U ziet dat men in FORTRAN evanals in ECOL in de voorwaardelijke sprongopdracht het ANDERS mag weglaten. Voor het geval dat i≠ 50 gaat het programma bij de volgende regel verder. In regel 105 moet GOF"TO gelezen worden als GOTO. De leerling van de Prot.Chr. Mavo in Amsterdam-Noord, die abusievelijk F schreef, heeft dit hersteld.
Verder ziet U dat de SCHRIJF-opdracht veel korter geformuleerd is dan in ECOL; er staat zoiets als:

103. SCHRIJF (op terminal 6, volgens regel 15) k, i en j

104. 15 FORMAAT (2 spaties, geheel getal van 2 cijfers, nieuwe regel, geheel getal van 3 cijfers, 5 spaties, geheel getal van 6 cijfers)

```
LINE   STATUS        CONVERSATIONAL FORTRAN
100.   READY          K=2
101.   READY          I=2
102.   READY   5      J=I*I
103.   READY          WRITE(6,15)K,I,J
104.   READY   15     FORMAT (2X,I2,/,I3,5X,I6)
105.   READY          IF (I.EQ.50) GOF"TO 20
106.   READY          I=I+1
107.   READY          GOTO 5
108.   READY   20     STOP
109.   READY          END
110.   READY #BEGIN
103.   YIELD    2
103.   YIELD   2            4
103.   YIELD    2
103.   YIELD   3            9
103.   YIELD    2
103.   YIELD   4           16
103.   YIELD    2
103.   YIELD   5           25
103.   YIELD    2
103.   YIELD   6           36

       ..................
```

Gebruik van de terminal

Het grote voordeel van een terminal heeft U uit het voorgaande kunnen afleiden. De leerling heeft 'lijfelijk kontakt' met apparatuur en ziet onmiddelijk resultaten.
Als nadelen zijn aan te merken:

- het toetsenbord en de verdere bediening van het toestel kunnen voor leerlingen zonder ervaring met een typemachine een hoge drempel zijn
- één terminal per 30 leerlingen blijkt onvoldoende
- de bediening van de terminal moet in veel gevallen buiten het lesuur computerkunde plaatsvinden (per lesuur kunnen immers hooguit 4 leerlingen hun programma's laten verwerken)
- het zijn vaak alleen de leerlingen met grote belangstelling voor computers, die de terminal de meeste tijd bezetten. De meerderheid van onze leerlingen, die we toch een indruk van computers willen geven en waarop ons onderwijs zich juist richt, raken in dat geval in de verdrukking.
- een schatting van de totale kosten van ƒ 600,- per maand is aan de lage kant.

Het is zaak de genoemde bezwaren zoveel mogelijk op te heffen, want de terminal zal in de wat verdere toekomst beslist zijn weg vinden naar onze scholen.

5 Achtergrondinformatie

5.1 COMPUTERHARDWARE

Hieronder vindt U de syllabus van het college, zoals dit op de herorriënteringscursus voor leraren gegeven werd door Prof.Dr. A. van der Sluis.

1. Geheugen

Het geheugen van een computer bestaat meestal uit een werkgeheugen en een achtergrondgeheugen. In kantoortermen: het werkgeheugen is zoiets als de kaartenbak die de boekhouder voor zich heeft staan, het achtergrondgeheugen is zoiets als het archief op zolder. Het werkgeheugen is niet zo groot, maar zeer snel toegankelijk, het achtergrondgeheugen is zeer groot maar niet zo snel toegankelijk.

1.1. Werkgeheugen

Het werkgeheugen bestaat doorgaans uit een groot aantal (enkele duizenden tot ettelijke miljoenen) ferrietringetjes. Zo'n ringetje kan linksom of rechtsom gemagnetiseerd zijn en gedraagt zich dus als een binaire variabele. We zullen de ene magnetisatie richting 0 en de andere 1 noemen. We noemen zo'n binaire variabele ook een bit (binary digit).

Deze bits worden in groepjes ingedeeld. Een zo'n groepje heet een woord. We zullen in deze syllabus de bits van één woord steeds nummeren 0, 1, 2, ..., en als we zo'n woord op papier schrijven zullen we steeds bit 0 rechts schrijven, bit 1 links daarvan etc. We zullen ook bit 0 het minst significante bit noemen en het meest linkse bit het meest significante.

Het aantal bits waaruit een woord bestaat varieert van het ene machinetype tot het andere en heet de woordlengte. Voorkomende woordlengtes zijn 12, 16, 18, 24, 27, 32, 36, 48, 64 en misschien nog wel andere. Hiernaast kwamen vooral vroeger computers voor met variabele woordlengte, d.w.z. dat de programmeur zelf de indeling van al de geheugenbits in woorden kon vaststellen.

De woorden worden genummerd 0, 1, 2, ... Het nummer van zo'n woord noemt men zijn adres.

Gewoonlijk heeft een woord nog een extra bit, het zgn. pariteitsbit. Bij het opbergen van informatie in het geheugen wordt dit bit steeds 1 gemaakt als de op te bergen informatie een even aantal enen bevat, anders 0 (of steeds andersom). Dit gebeurt automatisch. Bij het uitlezen van informatie uit het geheugen controleert de computer of het totaal aantal enen de juiste pariteit heeft en slaat alarm als dit niet zo is.

Zo'n ringetjes geheugen is heel snel toegankelijk, des te sneller naarmate de ringetjes kleiner zijn. Men is tegenwoordig al onder de microseconde accestijd en bij ± 0,5 mm buitendiameter van de ringetjes.

1.2. Registers

Naast het ringetjes geheugen heeft vrijwel elke computer nog een aantal uit flip-flops bestaande geheugenplaatsen welke men gewoonlijk registers noemt. Deze zijn nog sneller toegankelijk. Men meet de accestijd in nanoseconden.

1.3. Achtergrondgeheugen. Zie hiervoor 9.

2. Getalrepresentatie

Wat de rij nullen en enen, die de inhoud van een geheugenwoord vormen, voorstelt hangt geheel af van de opdrachten die op zo'n geheugenwoord opereren. Maar zelfs als een geheugenwoord alleen maar een getal hoeft voor te stellen gebeurt dit nog op allerlei verschillende manieren (afhankelijk van het type computer); de meeste computers hebben zelf nog weer verschillende manieren om getallen voor te stellen.

2.1. Gehele getallen

Er zijn twee standaard manieren om gehele getallen voor te stellen:

(a) 2-complement representatie: het woord $p_n, p_{n-1}, \ldots, p_0$ stelt het getal $\sum_{i=0}^{n-1} p_i\, 2^i$ voor als $p_n=0$ en het getal $(\sum_{i=0}^{n-1} p_i\, 2^i)-2^n$ als $p_n=1$.

Voorbeeld: n=4 00101=5 10101= -11.

We zullen ons in deze syllabus voornamelijk aan deze representatie houden, alhoewel in computers de volgende representatie gebruikelijker is:

(b) 1-complement representatie: het woord $p_n, p_{n-1}, \ldots, p_0$ stelt het getal $\sum_{i=0}^{n} p_i 2^i$ voor als $p_n=0$ en het getal $-\sum_{i=0}^{n-1} (1-p_i)2^i$ als $p_n=1$. Dit komt er in feite op neer dat als $p_n=1$ alle overige bits "omklappen": 0 wordt 1 en 1 wordt 0.

Voorbeeld: n=4 00101=5 10101= -10.

Voor $p_n=0$ stemmen (a) en (b) overeen, voor $p_n=1$ geldt (a)=(b)-1. Bij (b) zijn er twee representanten van het getal 0: allemaal nullen en allemaal enen.

Bij (b) heeft men geen speciaal aftrekmechanisme nodig: als men van de aftrekker alle enen in nullen verandert en andersom mag men optellen. Een opteller voor (b) is evenwel iets ingewikkelder dan een voor (a) zoals we nu zien.

Voorbeeld: n=4, representatie (a), tel op 5 en -3, dus 00101 en 11101. De optelling wordt eenvoudig tot stand gebracht door de operanden gewoon als positieve binaire getallen te beschouwen en ze op te tellen. In feite ontstaat dan 100010, maar door voor de som maar weer 5 bits ter beschikking te stellen ontstaat de meest linkse 1 helemaal niet en krijgt men het juiste antwoord 00010=2. Analoog -5+3:11011+00011= 11110= -2.

Voorbeeld: n=4, representatie (b), tel op 5 en -3, dus 00101 en 11100. Gewoon optellen als zoëven geeft 00001 en dat is 1 te weinig. Evenwel geeft -5+3, dus 11010+00011= 11101= -2 het goede antwoord. Het geheim zit erin dat als bij het "gewoon optellen" een 6e bit 1 zou ontstaan, men op de meest rechtse plaats van het resultaat 1 moet bijtellen (end around carry).

Als het resultaat van optelling of aftrekking in absolute waarde te groot is ontstaan er moeilijkheden.

Voorbeeld: n=4, representatie (a) of (b), tel op 12 en 14, dus 01100 en 01110. Dat kan helemaal niet, want 26 is niet voorstelbaar. De uitkomst 11010 wordt als negatief getal geïnterpreteerd. Men zegt dat er overflow optreedt.

Er is analogie met de kilometerteller van een auto. Als de auto nieuw is en op 00000 staat en men rijdt 5 km achteruit dan staat hij op 99995 en dat zal men wel willen identificeren met -5. Maar als men dat doet kan men het getal 99995 niet meer representeren.

Hoe groter de woordlengte, hoe groter getallen men nog kan representeren, maar een grens blijft er. In vele machines is er een soort alarm als er overflow optreedt.

2.2. Niet gehele getallen

Men wil natuurlijk ook met niet gehele getallen kunnen rekenen. Daartoe beschikken vele computers over een drijvende komma voorstelling van getallen. Een onrealistisch voorbeeld hiervan: woordlengte 9 bits. De bits 0 t/m 4 stellen een geheel getal voor (de mantisse), de bits 5 t/m 8 stellen de exponent voor van de macht van 2 waarmee de mantisse moet worden vermenigvuldigd.

Dan stelt 0000 01111 gewoon 15 voor, maar 0001 01111 stelt 2x15= 30 voor, 0101 01011= 2^5x11= 352, 1101 01011= 2^{-5}x 11, 1101 11011= 2^{-3}x(-5) wanneer men mantisse en exponent steeds volgens 2.1(a) interpreteert.

Dit voorbeeld is zeer onrealistisch door het geringe aantal bits dat voor mantisse en exponent beschikbaar is. Realistischer is: woordlengte 48 bits; mantisse 0 t/m 37, exponent 38 t/m 47. Dan ligt de exponent in tussen -512 en +511, zodat de 2 macht waarmee de mantisse moet worden vermenigvuldigd in

ligt tussen circa 10^{-153} en 10^{153}. Dit betekent dat men zeer grote en zeer kleine getallen nog met zeer goede precisie kan voorstellen.

Merk nog op dat ook gehele getallen uitstekend in drijvende kommagedaante zijn voor te stellen.

.3. Waarom verschillende representaties

Dat men in een en dezelfde computer gehele getallen zowel in vaste komma (d.w.z. volgens 2.1)als in drijvende komma (d.w.z. volgens 2.2) voorstelling kan weergeven is een kwestie van tijdeconomie. Het ligt immers voor de hand dat de extra manipulaties met de exponent extra tijd kosten, vooral bij het optellen: de computer moet dan immers eerst de mantissen ten opzichte van elkaar verschuiven over een aantal posities dat met het exponentenverschil overeenkomt. Met name bij vele simpele telopdrachten (bijv. de teller in een cyclus) werkt men met vaste komma getallen veel sneller.

Hoe weet de computer nu of een getal dat hij uit het geheugen haalt in vaste of in drijvende komma voorstelling staat? Dat hangt af van de opdracht waarvoor dit getal uit het geheugen wordt gehaald; er zijn opdrachten die met drijvende komma getallen werken en opdrachten die met vaste komma getallen werken (zie 4.2).

Hoe kan men nu in een computer met woordlengte 12 of 16 (of zelfs 32) bits enigermate redelijke getallen voorstellen. (2^{16} is immers slechts ≈64000). In zo'n computer gebruikt men meerdere woorden (gewoonlijk in opeenvolgende adressen) om één getal voor te stellen en zo krijgt men toch voldoende veel bits voor één getal beschikbaar.

Sommige computers hebben ook opdrachten die dergelijke meervoudige lengte getallen "in een klap" uit het geheugen halen en verwerken, bij andere moet men dit geheel uitprogrammeren.

Bij het werken met ECOL laten we de computer uitsluitend met drijvende komma getallen werken. Zou men namelijk van de grotere efficiëncy van het werken met gehele getallen hebben willen profiteren, dan had men de leerling moeten verplichten aan het begin van zijn programma te verklaren (declareren) welke van zijn variabelen alleen gehele waarden hoeven te kunnen aannemen en welke niet (zie bijv. ALGOL). Dit hebben we hem willen besparen.

2.4. Decimale getallen

Sommige computers kunnen ook met decimale getallen werken en onder omstandigheden spreekt men ook van decimale computers. Dit betekent eenvoudig, dat men in de meest rechtse 4 bits (16 mogelijkheden!) de meest rechtse decimaal onderbrengt,in de volgende 4 bits de tweede decimaal etc.

Het nut hiervan is, dat met name bij administratief gebruik een computer met een getal vaak niet veel meer doet dan het aflezen en afdrukken en dan is het wel jammer dit getal, dat immers op een ponskaart decimaal wordt voorgesteld eerst naar de binaire vorm te converteren en dan weer terug. Het rekenen met decimale getallen gaat echter wel langzamer.

3. Andere soorten informatie

In het geheugen van een computer wil men niet alleen getallen kunnen opbergen, maar ook bijv. tekst en de opdrachten van het programma. Ook deze informatie moet men dus voorstellen door rijtjes bits. Ook deze voorstelling verschilt weer van machine tot machine. Een heel gebruikelijke situatie is dat men letters, cijfers, tekens door rijtjes van 8 bits voorstelt (zodat men dus 256 verschillende "karakters" kan voorstellen). Bij een woordlengte van 32 bits kan men dan 4 karakters in een woord onderbrengen. Zo'n woord dat met 4 karakters gevuld is is weer ononderscheidbaar van een woord met een getal erin. Het hangt weer af van de opdracht, die op zo'n woord opereert, hoe de informatie geïnterpreteerd wordt.

Omdat er zo vaak met karakters gemanipuleerd wordt is men er toe overgegaan machines te bouwen die niet alleen in woorden van (zeg) 32 bits adresseerbaar zijn, maar bovendien in "woorden" van 8 bits, zgn. bytes.

Zo zal het woord met adres 328 de bytes met adressen 1312 (=4x328), 1313,1314 en 1315 bevatten. Aldus is men in staat om met een opdracht 1 karakter uit een woord te halen. Zo kan de RIJSYM in ECOL door een rij bytes worden voorgesteld.

Maar ook kan een woord uit "losse" bits bestaan.

Een andere mogelijkheid is dat men in een programma met zeer vele kleine getalletjes, bijv. <31, moet werken. Hiervan kan men er 6 achter elkaar in een woord van 32 bits stoppen.

4. Opdrachten van een computer

Het opdrachten-arsenaal van een computer is enerzijds veel beperkter en anderzijds veel uitgebreider dan de opdrachten die we in ECOL tegenkwamen.

De meeste computers hebben geen opdrachten van het type a:= b+c, waarin a,b en c willekeurige geheugenplaatsen zijn, om nog maar te zwijgen van opdrachten als a:= (b+c)x(d+e) of a:= sin (x) of a(i):=b(3xj+k)+c(r).

Wanneer we nu een schets geven van opdrachttypes die computers wel kennen, bedenke men dat de opdrachtensets van computers zeer verschillend zijn en er waarschijnlijk geen computer is die alle volgende opdrachten heeft, terwijl wij anderzijds niet volledig zijn.

In het volgende zullen we het geheugen beschouwen als een rij M(0:n), n een of ander groot getal, zodat bijv. M (8) de geheugenplaats met adres 8 is.

De registers (zie 1.2) zullen we aanduiden met R_1,....., R_m, waarin m gewoonlijk een vrij laag getal is, bijv. 7 of 15.

4.1. Transportopdrachten

Een computer kan gewoonlijk alleen maar informatie heen en weer transporteren tussen een geheugenplaats en een register, niet rechtstreeks van geheugenplaats naar geheugenplaats.

Dergelijke opdrachten hebben het type

(1) $R_i := M(j+R_k)$ (2) $R_i := -M(j+R_k)$

(3) $M(j+R_k) := R_i$ (4) $M(j+R_k) := -R_i$

Hierin zijn i,j en k getallen. Dezelfde opdrachten, maar dan zonder de R_k zijn ook toegestaan. Om dit in het volgende niet telkens expliciet te hoeven zeggen voeren we een fictief register R_0 in, dat altijd het getal 0 bevat.

Kennelijk kan men dan deze opdrachten kenmerken door een viertal getallen, namelijk opdrachttype (we zullen dit getal verder f noemen) en de getallen i,k,j. Door voor elk van deze getallen een vast aantal bits beschikbaar te stellen wordt de hele opdracht eenduidig door een rij bits voorgesteld. Zo zouden deze opdrachten bijv. het volgende formaat kunnen hebben

f	i	k	j
31 24	23 21	20 18	17 0

met 8 bits (dus 256 mogelijkheden) voor het opdrachttype f, 3 voor i en k en 18 voor j.

Gesteld dat de opdracht (1) de f-bits 00101001 heeft, zal het patroon

$$\underbrace{00101001}_{f}\ \underbrace{101}_{i}\ \underbrace{011}_{k}\ \underbrace{000000100010001101}_{j}$$

de opdracht

$$R_5 := M(2189+R_3)$$

voorstellen.

Het is nu ook duidelijk dat er nog weer verschillende opdrachten van bijv. gedaante (1) kunnen zijn:

- een die slechts 8 bits uit het aangeroepen woord ophaalt (omdat men bijv. met karakters wil werken en deze bits juist een karakter kenmerken), welke 8 bits nog weer op een aangegeven plaats in dit woord staan;
- een die een heel woord ophaalt;
- een die twee opeenvolgende woorden ophaalt (en deze ook in twee opeenvolgende registers deponeert.

Deze opdrachten onderscheiden zich dan namelijk in de eerste 8 bits.

De mogelijkheid geheugenplaatsen aan te wijzen door $j+R_k$ dient voor de rij organisatie. Zij a een rij zodat a(0) op adres 80 staat a(1) op 81 etc. en zij b een rij zodat b(0)

op adres 216 staat, b(1) op 217 etc. Laat de variabele p op geheugenplaats 12 staan en q op 27. Dan wordt een simpele ECOL opdracht als a(p):= b(q) voorgesteld door de machine opdrachten R_1:= M(27); R_2:=M(216+R_1); R_1:=M(12); M(80+R_1):= R_2.

4.2. Optel- en aftrekopdrachten

Deze plegen van het type a:= b±c te zijn, waarbij b en/of c een register is en waarbij a hetzelfde is als b of c. Geheugenadressen zijn weer als onder 4.1: getal + R_k.

Dus bijv.:

R_i:= R_i-M(j+R_k)

M(j+R_k):=M(j+R_k)-R_i

Er zijn weer verschillende opdrachten van dezelfde gedaante al naar gelang de betrokken bitrijen als vaste- of als drijvende komma getal moeten worden geïnterpreteerd of misschien zelfs de informatie in twee opeenvolgende geheugenplaatsen of registers als een geheel moet worden beschouwd.

Geheel getal optel- en aftrekopdrachten duren nauwelijks langer dan de geheugenaccestijd, in heel snelle computers zelfs aanmerkelijk korter. Drijvende komma optel- en aftrekopdrachten kunnen wel duidelijk langer duren: 30 μsec. in een middelsnelle computer, 75 nanoseconde in de allersnelste.

4.3. Vermenigvuldig- en deelopdrachten

Voor de drijvende komma opdrachten geldt veelszins hetzelfde als bij 4.2, zij het dat van de royale verzameling opdrachten die aldus in het vooruitzicht wordt gesteld veelal slechts een paar gerealiseerd zijn, bijv. alleen

R_1:=R_1× M(j+R_k)

R_1:=R_1/M(j+R_k)

en het verkregen resultaat is afgerond.

Bij vaste komma vermenigvuldigen heeft men meestal een opdracht als

$\overline{R_1R_2}$:= R_1× M(j+R_k)

waarin $\overline{R_1R_2}$ aanduidt dat de inhouden van R_1 en R_2 achter elkaar geschreven moeten worden gedacht.

Bij vaste komma delen heeft men veelal een opdracht als

R_2:= $\overline{R_1R_2}$/M(j+R_k), R_1:= rest

zodat men met één opdracht quotiënt en rest ter beschikking krijgt.

Vermenigvuldigopdrachten variëren van 30 μsec. in middelsnelle tot 225 nanoseconde in heel snelle computers. Delen is wat langzamer.

4.4. Logische opdrachten

De logische optelling geeft voor twee binaire woorden als resultaat de "bitsgewijze" som met de afspraak 1+1=1. Dus 101101'+'001010= 101111.

De logische vermenigvuldiging geeft voor twee binaire woorden als resultaat het "bitsgewijze" product. Dus 101101'×'001010= 001000.

Bij de logische vermenigvuldiging kan men de ene operand interpreteren als een masker waardoor men de andere bekijkt. Wanneer de bits van het laatstgenoemde patroon bijv. weergeven dat bepaalde condities al of niet vervuld zijn (1 voor wel en 0 voor niet), bijv. in een register van persoonsgegevens bij de politie kan men de computer aldus als het ware in een oogopslag laten constateren of minstens één van de door het masker zichtbare subset der condities vervuld is: het resultaat van de logische vermenigvuldiging is dan niet 0. En men kan ook heel snel constateren of ze allemaal vervuld zijn door van het resultaat het masker-patroon af te trekken en op 0 te testen. Etc.

Ook is de logische vermenigvuldiging heel geschikt om uit een woord van 32 bits dat 4 karakters van 8 bits bevat een bepaald karakter te lichten.

4.5. Schuifopdrachten

Ook deze opdrachten vindt men in ECOL niet en we zullen ook niet trachten ze in ECOL te omschrijven. Er zijn weer vele typen. Ze verschuiven het hele bitpatroon in een aangegeven register een aangegeven aantal plaatsen naar links of naar rechts en vullen de leeggekomen plaatsen met nullen of enen of met het achterste bit van de oorspronkelijke inhoud; of wat er links uitschuift komt er rechts weer in (rondschuiven); of het zojuist geleveerde vindt plaats met de hele bittrein die 2 met name genoemde registers tezamen vormen. Etc.

Deze opdrachten vervullen een duidelijke functie wanneer men een binair getal met een macht van 2 wenst te vermenigvuldigen. Of wanneer bijv. de bits 8-15 een karakter voorstellen dat men op de plaatsen 0-7 wenst te krijgen. Ook bij het in- en uitpakken van meerdere kleine getalletjes in één woord (zie eind van 3).

Een voorbeeld van samengaan van logische en schuifopdrachten levert het vliegtuigreserveringsprobleem. Stel dat de raamplaatsen genummerd zijn 0-19, de eraan grenzende plaatsen 20-39 en dat het bezet zijn van plaats i gekenmerkt wordt door het 1 zijn van de i-de bit in een bepaald woord.

Dan kan men met slechts een paar opdrachten constateren of er 2 plaatsen naast elkaar vrij zijn: tel het betreffende woord logisch op bijzijn 20 plaatsen naar rechts verschovene. Iedere 0 in het resultaat geeft een paar vrije plaatsen aan. Zijn er geen vrije paren dan bestaat het resultaat uit uitsluitend enen en dat kan men met één opdracht testen. Na een dergelijke test weet men dus of er vrije paren zijn. Om te weten welk paar vrij is is het handig als men over een zgn. normeerschuif-opdracht beschikt d ze schuift het woord zoveel plaatsen op dat de eerste 0 (als het woord met een 1 begint) of de eerste 1 (als het woord met een 0 begint) op kop komt en deponeert het aa. tal plaatsen waarover geschoven is in een register.

4.6. Sprongopdracht

De ECOL opdracht "NAAR 27 " vindt hier zijn analogon in "NAAR j+R_k" : ga verder met de opdracht die in geheugenplaats j+R_k zit. Dit herinnert ons eraan dat we wel hebben gezegd dat de opdrachten ook in het geheugen zitten, maar niet in welke volgorde ze worden afgewerkt. Het ligt voor de hand, dat de normale volgorde is de volgorde waarin ze in het geheugen zitten. Een opdracht als "NAAR j+R_k" doorbreekt dit.

Hiernaast heeft men dan veelal nog opdrachten als NAAR M(j+R_k) (strooisprong), waarmee men aangeeft dat in geheugenplaats j+R_k het adres te vinden is waar moet worden verder gegaan. D.w.z. NAAR M(10) is hetzelfde als NAAR 88 wanneer geheugenplaats 10 het getal 88 bevat.

De ECOL opdracht ALS....DAN....ANDERS...., waarmee men dus twee plaatsen moet aangeven waar kan worden verder gegaan, treft men bij computers niet aan. In plaats daarvan heeft men een opdracht ALS....DAN.... Wanneer de conditie vervuld is wordt verder gegaan op de aangegeven plaats. Wanneer de conditie niet vervuld is wordt gewoon verder gegaan op de geheugenplaats volgende op die waar de opdracht ALS....DAN... zelf staat.

Achter ALS... mag natuurlijk geen ingewikkelde toestand staan zoals in ECOL. De opdrachten zijn veelal van het type

ALS R_i>0 DAN j+R_k

ALS R_i=0 DAN j+R_k

e.d.

Deze opdrachten passen weer in het standaardformaat (zie 4.1), waarbij het typenummer in 24-31 tot uitdrukking brengt of de conditie >0 of = 0 o.i.d. luidt.

4.7. Subroutine sprong

Dit is een sprongopdracht met hetzelfde gevolg als de NAAR opdracht, met nog iets extra's. Hij bergt namelijk op een aan te geven plaats het adres (of het adres + 1) op waar hij zelf staat.

Als de opdracht bijv. luidt

SUB j+R_k, R_i

en hij staat zelf op adres 723, dan vindt een sprong plaats naar adres j+R_k en het getal 723 (resp. 724) wordt opgeborgen in R_i. De opdracht past blijkbaar nog steeds in het standaardformaat.

We illustreren het belang van de subroutinesprong aan een voorbeeld. Stel dat men in een programma op vele plaatsen een sinus wenst uit te rekenen. Men zal het dan op prijs stellen slechts eenmaal een stuk programma te hoeven schrijven dat de sinus bepaalt van het argument dat bijv. in R_3 zit; telkens als men in het eigenlijke programma een sinus wil laten uitrekenen stopt men het argument in R_3 en schrijft een sprongopdracht naar het stuk sinus programma. Het probleem zit nu hierin, dat men na het sinusprogramma doorlopen te hebben weer wenst terug te keren naar de plaats waar men vandaan gekomen is, maar dit is telkens ver-

schillend (zie figuur 1)!

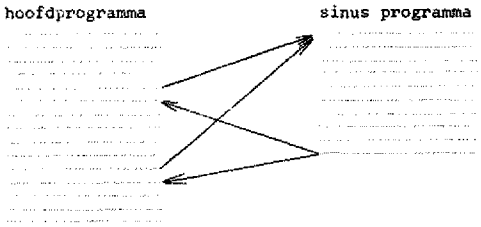


fig. 1

Door naar het sinusprogramma te springen met een subroutinesprong die het adres waar hij zelf staat (of dit adres + 1) opbergt bijv in R_5 en de sinusroutine te laten eindigen met NAAR 1+R_5 (resp. NAAR R_5) is men uit de moeilijkheden. Een zo eindigend stuk programma noemt men een subroutine.

Het is ook fijn wanneer naast de opdrachten SUB j+R_k, R_i en SUB M(j+R_k), R_i nog subroutinesprongen bestaan die het terugkeeradres niet in R_i opbergen, maar in het adres aangewezen door R_i. We zullen deze opdrachten aanduiden met SUBS j+R_k, R_i, resp. SUBS M(j+R_k), R_i.

.8. Stapelprogrammering

Een faciliteit waarvan vooral de geavanceerde programmeur plezier beleeft is dat een bepaald register, bijv. R_7 de eigenschap heeft dat telkens wanneer in een opdracht de uitdrukking M (j+R_7) voorkomt de volgende extra actie plaatsvindt: als er iets uit de geheugenplaats j+R_7 wordt afgelezen wordt R_7 voor dat aflezen met 1 verlaagd, als er iets wordt opgeborgen wordt R_7 na dat opbergen met 1 verhoogd.

Voorbeelden

(a) R_7:=81; R_3:= R_3+M(R_7). Effect: R_7:= 81; R_7:= 80; R_3:= R_3+M(80)

(b) R_7:=81; M(R_7):=R_3. Effect: R_7:= 81; M(81):= R_3; R_7:= 82

(c) R_7:=81; M(R_7):= M(R_7)+R_3. Effect: voor de bepaling van het rechterlid wordt eerst R_7 met 1 verlaagd, zodat het rechterlid dus het effect M(80)+R_3 heeft; dit wordt opgeborgen in de geheugenplaats met adres 80 waarna R_7 weer met 1 verhoogd wordt. Dus totaal effect: M(80):= M(80)+R_3; R_7:= 81.

Deze faciliteit maakt het mogelijk het geheugen als een zgn. stapel te beschouwen, waarbij R_7 als de stapelwijzer fungeert. Dit is van belang als men een hoeveelheid informatie wenst op te slaan die men in omgekeerde volgorde wenst te gebruiken. (Voorbeelden hiervan worden elders in deze cursus gegeven.) Men legt dan de op te slane informatie als het ware op een stapel, elke nieuwe informatie-eenheid er boven op en bij het terughalen van de informatie neemt men telkens de bovenste informatie-eenheid eraf (denk aan een stapel kaarten). In dit licht bezien wijst R_7 dus steeds naar de eerste vrije plaats van de stapel.

Heel prettig is het terugkeeradressen van subroutines op een stapel te kunnen leggen. Dit kan blijkbaar door de subroutine aan te roepen met SUBS ..., R_7, waarbij analoog geldt dat R_7 dan na dat opbergen met 1 verhoogd wordt.

5. Werking van een computer in principe

Het gebeuren in de computer speelt zich af in twee fasen, die elkaar voortdurend afwisselen.

Fase 1: de aan de beurt zijnde opdracht wordt uit het geheugen afgelezen.

Fase 2: de afgelezen opdracht wordt uitgevoerd.

Deze afwisseling van fase 1 en 2 noemt men de basiscyclus.

Het besturingsorgaan van de computer bevat daartoe twee registers: de opdrachtwijzer (OW) houdt het adres bij waar de volgende opdracht vandaan gehaald moet worden; het opdrachtregister (OR) bevat de opdracht die uitgevoerd moet worden.

Het hele gebeuren in de computer berust er nu op, dat op het juiste tijdstip de juiste electrische verbindingswegen bestaan.

In fase 1 moet OW verbonden zijn met de zgn. selectie-ingang van het geheugen, d.i. die ingang die bepaalt, welk adres wordt uitgelezen en de uitgang van het geheugen moet verbonden zijn met OR (zie figuur 2).

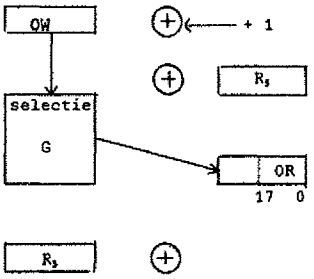


fig. 2

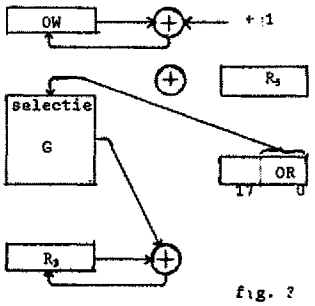


fig. 3

We zullen de selectie-ingang steeds bovenaan tekenen, de ingang waar men de op te bergen informatie aanbiedt links en de uitgang van het geheugen rechts.

In fase 2 wordt OW met 1 verhoogd. Stel dat we de eenvoudige opdracht R_3:= R_3+M(k) hebben. Dan moeten de bits 0-17 van OR met de selectie-ingang van het geheugen verbonden worden. De uitgang van het geheugen wordt, samen met de uitgang van R_3 aangeboden aan een opteller. De uitgang van de opteller moet verbonden zijn met de ingang van R_3 (zie figuur 3).

Voor een opdracht als R_3:= R_3+M(80+R_5) wordt fase 2 afgebeeld in figuur 4.

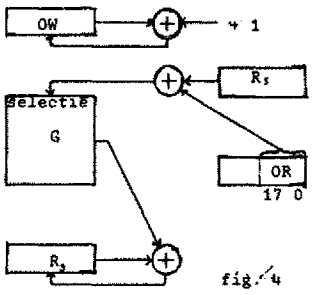


fig. 4

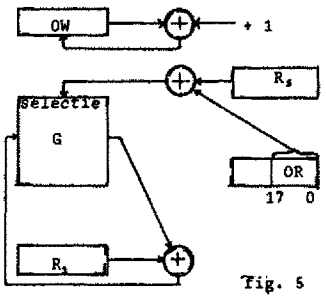


fig. 5

Figuur 5 geeft fase 2 weer voor M(80+R_5):= M(80+R_5)+R_3.

Sprongopdrachten van het type NAAR 80 worden eenvoudig gerealiseerd door 0-17 van OR over te nemen in OW (zie figuur 6).

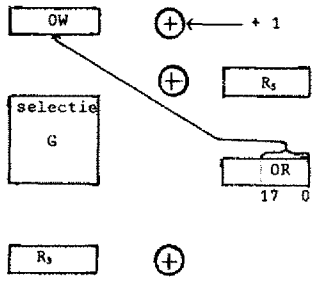


fig. 6

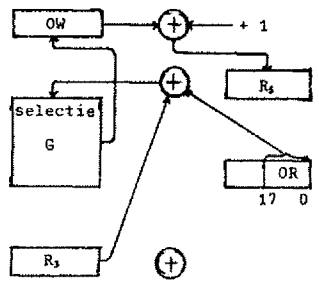


fig. 7

In figuur 7 ten slotte ziet men de opdracht SUB M(80+R_5), R_5.

Het is duidelijk dat er in dit betoog wat vereenvoudigingen zijn aangebracht. Elk der fasen zal uit meerdere delen moeten bestaan. Bijv. in figuur 3 zal men moeten voorkomen dat nadat de met 1 vermeerderde OW is teruggeschreven in OW deze nieuwe waarde nog eensmet 1 vermeerderd zou worden en nog eens. Analoog voor het bij elkaar tellen van geheugeninhoud en R_3.

6. Practische realisatie

6.1. Geheugen

Door elk ferrietringetje lopen 3 draden (zie figuur 8), een "horizontale" draad, een "verticale" draad en een leesdraad.

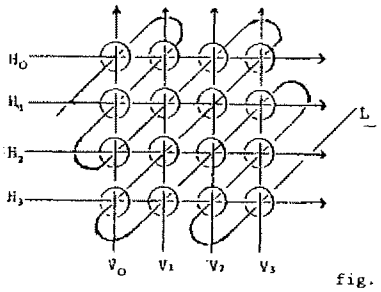


fig. 8

De vierkante hystereselus van de ringetjes (figuur 9) maakt dat er een stroomsterkte I_0 bestaat zodanig dat een stroom van deze sterkte niet bij machte is een ringetje van magnetische richting te doen omslaan, terwijl een stroom van dubbele sterkte dit wel kan. Stuurt men bijv. door H_i en V_j stromen met sterkte I_0 in de richting van de pijlen of juist in tegengestelde richting, dan neemt het ringetje op hun kruispunt een magnetisatie richting aan passend bij de stroomrichting en alle andere ringetjes blijven staan.

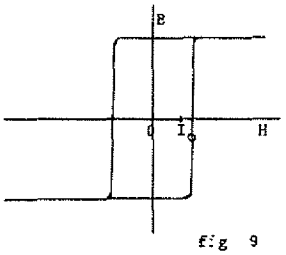


fig 9

Wenst men later te weten hoe dit ringetje staat dan stuurt men een stroom door H_i en V_j in de richting der pijlen. Past de magnetisatierichting van het ringetje bij de stroomrichting, dan blijft het staan en wordt er een andere spanning in de leesdraad geïnduceerd dan wanneer het niet paste en dus omklapte.

Het uitlezen is blijkbaar destructief: na afloop past het ringetje beslist bij stromen in de richting van de pijlen. Er moet derhalve een voorziening getroffen worden om zo'n uitgelezen ringetje meteen weer goed te zetten.

Op de vraag hoe nu juist H_i en V_j stroom gaan voeren als wij dat wensen gaan we straks in.

6.2. Flip-flop

Ook een flip-flop is een geheugen element, maar dan een electronisch. Het is een kastje met 2 ingangen en 2 uitgangen (fig. 10). Van deze uitgangen heeft er steeds één een"hoge" spanning (die we verder als 1 zullen aanduiden) en één een "lage" spanning (verder aan te duiden als 0). Als beide ingangen 0 zijn verandert er niets. Als een der beide ingangen 1 gemaakt wordt, wordt de bijbehorende uitgang 1 (en dus de andere 0). Het is verboden beide ingangen 1 te maken. Een flip-flop is dus net een wip: die "onthoudt" ook aan welke kant hij het laatst naar beneden gedrukt is, ook als die druk wordt opgeheven. Maar als men aan beide kanten tegelijk drukt gebeurt er iets onvoorspelbaars.

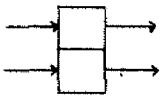


fig. 10

We zullen bovendien afspreken dat we zeggen dat de flip-flop een 1 voorstelt als de onderste uitgang de waarde van 1 heeft.

6.3. Poorten

We hebben verder nog electronische schakelaars nodig om op commando verbindingswegen te openen of af te sluiten. Deze functie wordt door poorten verricht.

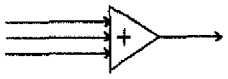


fig. 1

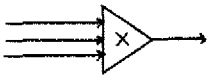


fig. 12

Poorten hebben een aantal ingangen en één uitgang. Bij de plus poort (or gate) (fig. 11) is de uitgang slechts dan 0 als alle ingangen 0 zijn. Bij de maal poort (and gate) (fig12) is de uitgang slechts dan 1 als alle ingangen 1 zijn.

De principe schema's van poorten zijn simpel (fig. 13 en 14).

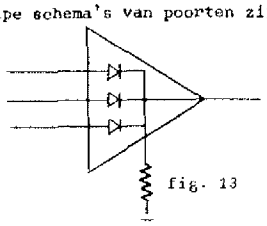


fig. 13

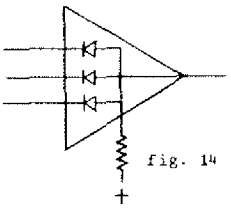


fig. 14

De werking wordt verklaard door op te merken dat een diode links nooit een hogere spanning kan hebben dan rechts.

6.4. Verbindingswegen

Een verbindingsweg tussen bijv.de uitgang vanG en de ingang van OR die alleen actief is in fase 1 (zie fig. 2) komt nu als volgt tot stand. Laat fase 1 daardoor gekenmerkt zijn dat op één heel bepaalde draad, verder te noemen F1, het signaal 1 staat idem fase 2 een draad F2.

Het geheugen heeft voor elke bit van het op te leveren woord een uitgang en deze uitgang is doorgaans dubbel uitgevoerd in die zin dat voor elk bit ook de tegengestelde waarde (d.w.z. 0 wordt 1 en andersom) is af te lezen.

Dan bouwt men tussen de geheugenuitgang en de ingangen van d flip-flops waaruit OR bestaat het volgende circuit, even vaak herhaald als er bits in een woord zijn (fig. 15).

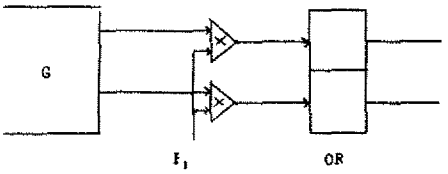


fig. 15

In fase 2 is de zaak duidelijk ingewikkelder, want dan wil men dat een bepaalde verbindingsweg precies dan actief is als er bij een heel bepaald bitpatroon in 24-31 van OR zit.
H Het aanwezig zijn van dat heel bepaalde bitpatroon kan men als volgt constateren met een poortschakeling. Ter vermindering van het tekenwerk doen we evenwel maar net alsof we slechts met bits 24-27 van OR te maken hebben. Dan geeft de maal poort in figuur 16 precies dan het signaal 1 af als 27-24 van OR het patr 0101 voorstellen en bovendien F2 de waarde 1 heeft.

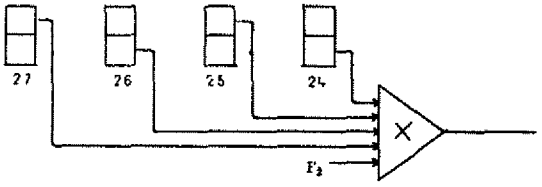


fig. 16

Met het uitgangssignaal van deze poortschakeling kan men nu verbindingswegen activeren analoog aan fig.15.

Op soortgelijke wijze activeert ook het geheugen de juiste draden H_i en V_j (zie fig. 8).

6.5. In- en uitvoer

Het is nu ook wel duidelijk hoe de informatie uit een ponskaart in de computer zou kunnen doordringen: men voert eenvoudig een opdracht in die een verbindingsweg tot stand brengt tussen de uitgangen van de kaartlezer (waar in de vorm van spanningen op draden het gaatjesbeeld vast ligt) en bijv. een bepaalde geheugen plaats. Analoog voor uitvoer.

7. Programmaverwerking

Wat gebeurt er nu als men een programma ter verwerking aanbiedt (bijv. een ALGOL 60 programma). In de computer is "permanent" een groot programma aanwezig, ALGOL compiler geheten. Met een druk op de knop forceert de operateur (= bediener van de machine) een sprongopdracht naar dit programma, zodat het aan de gang gaat. Dit programma leest nu de programmatekst karakter voor karakter in (zie 6.5) en slaat het als tekst op in het geheugen. Deze teks is nu vanzelfsprekend voor de computer nog geen programma (vgl. 4 hoe computeropdrachten er in het geheugen uitzien). De ALGOL compiler gaat nu (op een hoogst ingewikkelde) manier met deze tekst manipuleren en "berekent" een voor een de bitpatronen van d opdrachten die de computer straks zal moeten gaan uitvoeren om he door de auteur van het programma gewenste effect te bewerkstellig Zo zal de compiler moeten uitmaken welke geheugenplaats hij aan de variabele piet zal toekennen en hij zal expressies als a:= b+cx(d+exsin(x)) moeten reduceren tot de simpele opdrachten van 4 (incl. een subroutine aanroep voor de functie sin (x), welk subroutine vast in de ALGOL compiler is opgenomen).

De bedoelde bitpatronen schrijft hij netjes in volgorde in het geheugen en als hij daarmee helemaal klaar is eindigt hij met een sprongopdracht naar het eerste van deze bitpatronen, waarmee dan het aldus vertaalde programma geactiveerd (we zeggen ook geëxecuteerd) wordt.

Hiermee hebben we en passant een zeer belangrijke eigenschap van een computer leren kennen: omdat de bitpatronen van opdrachten niet "anders" zijn dan die van getallen en teksten kunnen bitpatronen die op een zeker moment uitkomst zijn van een berekening later als opdrachten optreden.

Dit is ook van belang bij het beantwoorden van de vraag hoe de ALGOL compiler zelf in de machine is gekomen en algemener hoe het "eerste" programma in de machine komt. Gewoonlijk wordt van fabriekswege op onuitwisbare wijze een uiterst simpel programmaatje in het geheugen geplaatst van maar een paar opdrachten dat bijv. alleen in staat is zgn. binaire programma's te lezen, dat zijn programma's die helemaal in enen en nullen uitgeschreven zijn precies zoals ze in het geheugen moeten komen. Om een programma zo uit te schrijven is natuurlijk niet leuk, vandaar dat men dat maar met één programma doet, nl. een programma dat in staat is programma's in een prettiger code in te lezen. In die prettiger code schrijft men dan wellicht een inleesprogramma voor een nog hoger geordende code.

8. In- en uitvoer; verkeer met achtergrondgeheugen

Het is vrijwel nooit doelmatig een gebruikersprogramma zelf direct met in- en output organen of achtergrondgeheugen te laten communiceren. Daarvoor is een hedendaagse computer eenvoudig te snel. Als men in ECOL schrijft a:= LEES, en pas op dat moment zou een ponsbandlezer of ponskaartlezer karakter voor karakter van band of kaart gaan lezen dan zou deze opdracht vele millisecunden in beslag nemen en dat terwijl de computer slechts microsecunden nodig heeft om een reken operatie uit te voeren. Bij verkeer met magneetband, -trommel of -schijf is het al niet anders. Men staat daar eerst een tijd te wachten (misschien minuten bij magneetband, tientallen millisecunden bij trommel of schijf), voordat de juiste geheugenplaats langs lees- of schrijfkop komt, maar ook als deze plaats gearriveerd is en ook als men dan een aantal opeenvolgende geheugenplaatsen op dat medium wenst te bespelen, dan zijn deze apparaten nog steeds "langzaam": ergens tussen 10000 en 100000 woorden per seconde.

Moderne, snelle computers beschikken daarom over een of meer zgn. communicatiesatelleten. Dat zijn eigenlijk kleine computertjes met een eigen besturingsorgaan en soms ook nog met een eigen klein geheugentje; soms herbergen deze één vast (door de fabriek erin geplaatst) programma, soms zijn ze nog programmeerbaar ook. De opdrachtenset is evenwel geheel gericht op communicatie met de randapparatuur en de eigenlijke computer (verder te noemen de centrale computer).

Het programma in de centrale computer kan nu aan de communicatiesatelliet opdracht geven om bijv. 1000 woorden uit het trommelgeheugen af te lezen, op een bepaalde plaats te beginnen en deze in het werkgeheugen van de centrale computer te plaatsen, op een bepaalde plaats te beginnen en na afloop te melden dat het gebeurd is.

Dit werpt het boeiende probleem op twee computers, die elk in hun eigen ritme werken, met elkaar te laten samenwerken. Omdat deze samenwerking o.a. de bedoeling heeft dat de ene computer in het geheugen van de andere leest of schrijft zal er in elk geval een soort voorrangsregeling moeten zijn die uitmaakt wie toegang krijgt tot het geheugen als ze allebei tegelijk komen.

Fundamenteler is evenwel dat de centrale computer aan de satelliet een seintje moet kunnen geven dat deze een karweitje moet opknappen en dat de satelliet computer aan de centrale computer moet kunnen berichten dat het karweitje klaar is.

Dat ze in elkaars geheugen kunnen lezen en schrijven is daarvoor niet voldoende. Het zou immers te tijdrovend zijn (en te vervelend) het computerprogramma voortdurend naar bepaalde geheugenplaatsen te laten kijken om te zien of daar al wat staat. Dat zou net zo iets zijn als een telefoon zonder bel: men zou elke keer naar de telefoon moeten lopen om te horen of er soms iemand iets wil zeggen. De bel dient ervoor om aan te kondigen dat iemand contact wenst.

De computers moeten dus elkaars aandacht kunnen trekken. We schetsen hoe de satelliet een seintje geeft aan de centrale computer (andersom gaat net zo of nog eenvoudiger, omdat de satelliet mogelijk wel tijd heeft om voortdurend naar bepaalde geheugenplaatsen te kijken).

Als de satelliet een seintje wil geven aan de centrale computer zet hij op een bepaalde draad, de interrupt-draad, het signaal 1. De centrale computer let in fase 1 van de basiscyclus op de interruptdraad. Als daar het signaal 0 op staat gaat alles zoals in 6 beschreven. Als er evenwel het signaal 1 op staat wordt de selectie ingang van het geheugen niet verbonden met OT, maar wordt er een bepaald bitpatroon opgezet. Stel dat dit het bitpatroon 00.....011000 is. Dan wordt dus geheugenplaats 24 uitgelezen en de inhoud daarvan wordt geplaatst in OR. Wat er nu verder gebeurt hangt er natuurlijk van af wat dit voor een opdracht is, maar het is wel duidelijk dat het op dat moment werkende programma ruwelijk onderbroken is. Dit is op zichzelf genomen hoogst bedenkelijk, want de interrupts van de satelliet komen op volstrekt onvoorspelbare momenten en het is doorgaans heel ongewenst dat een werkend programma zo maar ergens onderbroken wordt.

Gewoonlijk evenwel zal de programmeur van het zgn. bedrijfsysteem (dat is een groot vast programma dat altijd in de computer zit, net als de ALGOL compiler) op de betreffende geheugenplaats 24 een subroutinesprong geplaatst hebben naar een stuk programma waar verder kennis genomen wordt van de mededelingen die de satelliet computer in het geheugen van de centrale computer heeft geplaatst. Omdat we met een subroutinesprong naar dit stuk programma zijn gesprongen, is het adres waar het onderbroken programma gebleven was, bewaard gebleven en kan daarheen later worden teruggekeerd.

Gewoonlijk zijn er nog voorzieningen om interrupts, al naar hun herkomst, met verschillende prioriteit af te werken en ook om al naar deze herkomst een andere geheugenplaats dan 24 te laten aflezen, alles ter verhoging van de snelheid.

Het interrupt-mechanisme maakt het mogelijk om een nieuw programma met zijn gegevens alvast in het computergeheugen te laten opnemen, terwijl een ander programma nog bezig is. En ook om de output van een programma, dat veel meer regeldruktijd nodig heeft dan rekentijd (een zgn. output intensief programma) op de regeldrukker te produceren terwijl een nieuw programma al weer bezit van de centrale computer genomen heeft.

Het interrupt-mechanisme is ook de grondslag van de mogelijkheid tot multiprogramming, d.w.z. dat de computer schijnbaar aan meerdere programma's tegelijk rekent, maar in werkelijkheid het ene programma laat werken in de tijden dat het andere niet verder kan omdat het bijv. op informatie van een achtergrondgeheugen staat te wachten.

Bij grote computersystemen komt ook steeds meer multiprocessing in zwang, d.w.z. dat het computersysteem meerdere centrale computers bevat die elkaar ook weer via een interruptsysteem signaleren.

Tenslotte maakt het interruptsysteem timesharing mogelijk, d.w.z. dat vele gebruikers om de beurt een kort stukje rekentijd toegemeten krijgen. Deze mensen zijn dan via een terminal (doorgaans een schrijfmachine-achtig apparaat, soms op grote afstand van de computer geplaatst en daarmee via telefoonlijnen verbonden) met de computer verbonden en geven over deze terminal hun opdrachten aan de computer en krijgen daarop ook hun antwoorden.

Een heel belangrijk voordeel van timesharing is de mogelijkheid die het geeft programma's in samenspraak met een computer te testen en te verwerken. Bij de normale wijze van programmaverwerking (in batch, zoals dat heet) kost het dagen om een programma een aantal keren, in een steeds wat andere versie, aan te bieden. Met timesharing kost hetzelfde slechts uren of kwartieren.

Dat al het geïnterrumpeer, dat op deze bladzijden beschreven werd, goed verloopt is dankzij het bedrijfsysteem, dat boekhoudt waar de informatie betreffende elk programma zich bevindt, uitzoekt welk programma aan de beurt is en nog veel meer. Hierover wordt elders in deze cursus gerapporteerd.

9. Allerlei randapparatuur

9.1. Ponsbandlezer

200 - 2000 tekens per seconde. Werkt foto-electrisch of capacitief. 5, 7 of 8 sporen.

9.2. Ponsbandponser

100 - 200 tekens per seconde. 5, 7 of 8 sporen.

9.3. Ponskaartlezer

10 - 20 kaarten per seconde. Werkt met contactborstels of foto-electrisch.

9.4. Ponskaartponser
100 - 500 kaarten per minuut.

9.5. Magneetbandapparaat
Rollen magneetband met maximale lengte 720 meter. Snelheid ca. 3 m/sec., start/stoptijd enkele millisecunden. 10 - 60 tekens van 6 of 8 bits (+ parity) per mm. Dus per rol enkele tientallen miljoenen tekens. Overdrachtsnelheid in de orde van 100.000 tekens per seconde.

9.6. Magneettrommel
Ronddraaiende magnetiseerbare cilinder met enkele honderdduizenden tot enkele miljoenen woorden. Rotatiesnelheid 20 - 100 maal per seconde, dus wachttijd in de orde van tientallen millisecunden. Overdrachtsnelheid plm. 100.000 woorden per seconde. Terugdenkend aan de vergelijking in het begin van 1 (kaartenbak op tafel versus archief op zolder) is de zolder wel ver weg: om iets uit het archief te halen duurt 10000 maal zo lang als om iets uit de kaartenbak te halen, d.i. een etmaal versus 8 seconden.

9.7. Magneetschijven
Stapel ronddraaiende magnetiseerbare schijven. Werkwijze vergelijkbaar met die van trommel. Elke schijfkant heeft zijn eigen lees- schrijfkop die in radiale richting instelbaar is. Dit kost wat extra tijd (± 100 millisecunden). Voordelen boven magneettrommel: schijven zijn uitwisselbaar; bij gelijke geheugenomvang minder duur.

9.8. Magneetkaarten
De informatie staat op magnetiseerbare kaartjes (maten ca. 35 x 9 cm). Per kaart ca. 400.000 tekens. In de tijd van ± 0,2 sec. wordt de gewenste kaart uit het magazijn gelicht, langs een draaiende trommel gevoerd en aldus afgelezen.

9.9. Regeldrukker
10 - 20 regels van maximaal ± 150 tekens per seconde. Twee uitvoeringen (in beide wordt het papier horizontaal bedrukt).
(a) horizontale cilinder met op elke drukpositie een hele tekenset (dus max. 150 keer) draait 10 of 20 maal per seconde in het rond en als het juiste teken voordraait slaat een hamer het papier met carbonlint en al tegen de cilinder.
(b) ketting met 5 tekensets draait horizontaal voor het papier langs en telkens als het juiste teken op de juiste plaats staat slaat een hamer het papier met carbonlint en al tegen de ketting (allebei fraaie staaltjes van techniek: het tijdstip van afslaan van de hamer mag niet meer dan 50 μsec. variëren op straffe van dansende of wiebelende letters).

9.10. Plotter
Dit is een tekenapparaat. Het papier zit met pennen vast op een rol die heen en weer kan draaien in stapjes van 0,1 mm. In axiale richting loopt hierboven een wagentje met een schrijfpen, welk wagentje kan bewegen in stapjes van 0,1 mm. Het apparaat kent als opdrachten:
(a) haal pen van papier
(b) zet pen op papier
(c) draai rol 1 stapje vooruit
(d) draai rol 1 stapje achteruit
(e) pen 1 stapje naar links
(f) pen 1 stapje naar rechts.
De opdrachten (c of d) en (e of f) kunnen ook tegelijk worden gegeven, zodat de pen in feite in de 8 hoofdwindrichtingen over het papier kan bewegen. Het apparaat doet 300 stapjes per seconde (moderne uitvoering is nog sneller).

9.11. Visual display
Een soort televisiescherm waarop de computer een beeld kan projecteren. Heel interessant is dat men dit apparaat ook als input apparaat kan gebruiken. Bij het apparaat hoort een potloodachtig apparaatje aan een snoer, lichtpen geheten, voorzien van een lichtgevoelige cel.

De computer projecteert op een bepaalde plaats op het scherm een punt. Houdt men de lichtpen bij deze punt en beweegt men hem er vervolgens kalmpjes vandaan dan constateert de computer de vermindering van lichtsterkte en verschuift de punt net zo lang totdat de lichtsterkte weer maximaal is. Zo kan de computer dus altijd weten waar de lichtpen zich bevindt, met andere woorden de beweging van de pen kan gebruikt worden om gegevens in de computer te voeren.

5.2 DATASTRUCTUREN

Hieronder volgt de tekst van de lezing over datastructuren, zoals die gehouden is op een heroriënteringscursus computerkunde door Drs. C.A.Ch. Görts.

1. Inleiding

De informatie waarop een programma moet werken bezit vaak een zekere structuur. Voordat men het programma gaat schrijven moet men zich van deze structuur bewust zijn en zich afvragen op welke manieren men deze structuur in het geheugen van de computer kan representeren. Bij het maken van een keuze uit deze manieren moet men zich goed voor ogen houden wat men met de informatie wil doen. In enkele voorbeelden zal dit worden toegelicht.

2. De woordenlijst

Een in de litteraire wetenschap veel voorkomende bezigheid, waarvoor tegenwoordig vaak computers worden gebruikt, is het opstellen van een woordenlijst van een tekst (een boek, of het hele werk van een auteur):deze woordenlijst bestaat in zijn eenvoudigste vorm uit een alfabetische lijst van alle woorden die in de tekst voorkomen.

De computer leest de tekst woord voor woord. Van ieder woord wordt nagegaan of het al in de woordenlijst voorkomt. Zo niet, dan wordt het eraan toegevoegd.

Het gaat er nu dus om de woordenlijst in een handige vorm in het geheugen op te bouwen, uitgaande van de volgende drie werkzaamheden die we op de woordenlijst willen uitvoeren: het opzoeken of een woord voorkomt, het toevoegen van een woord en het (na afloop) in alfabetische volgorde afdrukken van de woordenlijst.

Eenvoudigheidshalve zullen we aannemen dat de woorden uit niet meer dan 10 letters bestaan. Voor een woord reserveren we 10 plaatsen in een rij a. Als het woord minder dan 10 letters heeft vullen we de resterende plaatsen op met nullen. We nemen aan, dat bekend is hoe men van twee woorden kan uitmaken welk het eerst komt in alfabetische volgorde.

2.1. Sequentieel

Uit § 20 van computerkunde is duidelijk dat het niet verstandig is de woorden van de woordenlijst in volgorde van binnenkomst in een rij te zetten: om op te zoeken of een woord voorkomt moet men dan de hele woordenlijst langslopen.

Het opzoeken of een woord voorkomt gaat veel sneller als de woordenlijst alfabetisch geordend is: dan kan de halveringsmethode worden toegepast. (Computerkunde, § 20, 4.)

RIJ a

1	e	n	0	0	0	0	0	0	0	0
11	h	e	t	0	0	0	0	0	0	0
21	h	o	e	0	0	0	0	0	0	0
31	v	e	r	0	0	0	0	0	0	0
41	w	a	r	m	0	0	0	0	0	0
51	w	a	s	0	0	0	0	0	0	0
61										

figuur 1

Als we echter de woorden sequentieel, d.i. in alfabetische volgorde achter elkaar (zie figuur 1, het begin van de tekst is: "hoe warm het was en hoe ver") opbergen, hebben we het bezwaar dat het toevoegen van een woord aan de woordenlijst te veel tijd kost: er moet een plaats worden vrijgemaakt door een deel van de rij op te schuiven.

2.2. De keten

We willen dus de woordenlijst met behoud van de alfabetische volgorde in het geheugen refresenteren, maar zo, dat het mogelijk is snel een woord toe te voegen. Dit kan als we het doen in de vorm van een keten (zie figuur 2).

RIJ a

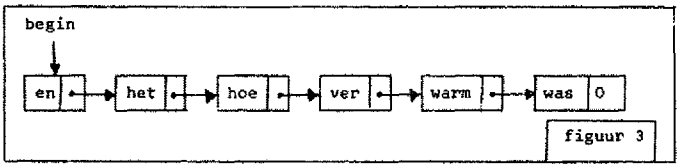
1	h	o	e	0	0	0	0	0	0	0	56
12	w	a	r	m	0	0	0	0	0	0	34
23	h	e	t	0	0	0	0	0	0	0	1
34	w	a	s	0	0	0	0	0	0	0	0
45	e	n	0	0	0	0	0	0	0	0	23
56	v	e	r	0	0	0	0	0	0	0	12
67											

begin=45
vrij =67

figuur 2.

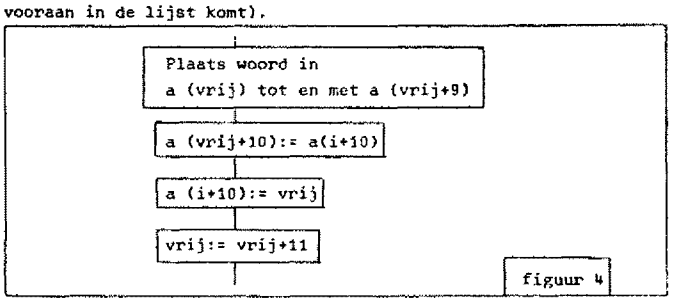
De woorden staan nu niet in alfabetische volgorde in de rij, maar aan ieder woord is een getal toegevoegd (de link) dat aangeeft waar men het volgende woord kan vinden. Bovendien zijn er twee variabelen, die we begin en vrij noemen, die aangeven waar het eerste woord staat en welke de eerste vrije plaats in de rij is. Het alfabetisch laatste getal heeft de link 0.

Vaak gebruikt men voor een keten (omdat het niet relevant is op welke index een woord te vinden is) de notatie van figuur 3, met links aangeduid door pijlen.



figuur 3

Het toevoegen van een woord aan de lijst is nu eenvoudig een kwestie van het bijwerken van enkele links. Het woord zelf plaatsen we achteraan op de vrije plaatsen (zie figuur 4, het woord wordt ingevoegd na het woord op index i; er moet natuurlijk een extra voorziening bij voor het geval het nieuwe woord alfabetisch vooraan in de lijst komt).



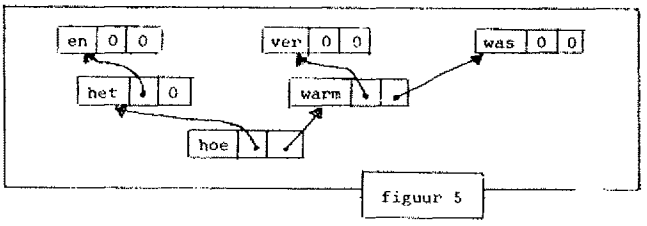
figuur 4

Het toevoegen van een nieuw woord is bij de keten dus eenvoudig. Het opzoeken van een woord is nu echter weer moeilijker: we krijgen immers het middelste woord niet snel te pakken. We zoeken daarom een betere oplossing van ons probleem.

We hebben de keten besproken omdat hij in veel andere toepassingen van nut kan zijn.

2.3. De boom

Om zowel snel te kunnen opzoeken als snel te kunnen toevoegen gebruiken we een boom: vanuit een woord wordt verwezen naar een linker en een rechter tak (deze kunnen leeg zijn). In de linker tak van een woord staan alle woorden die alfabetisch eerder komen,in de rechter tak de woorden die alfabetisch later komen (zie figuur 5).



figuur 5

Algemeen gedefinieerd is een (binaire) boom een verzameling elementen, die leeg is, of bestaat uit een element en twee takken, een linker en een rechter tak, die beide ook weer bomen zijn.

RIJ a

1	h	o	e	0	0	0	0	0	0	0	25	13
13	w	a	r	m	0	0	0	0	0	0	61	37
25	h	e	t	0	0	0	0	0	0	0	49	0
37	w	a	s	0	0	0	0	0	0	0	0	0
49	e	n	0	0	0	0	0	0	0	0	0	0
61	v	e	r	0	0	0	0	0	0	0	0	0
73												

vrij=73

figuur 6

Ieder woord is nu dus voorzien van twee links (zie figuur 6)

De tijd, nodig voor het opzoeken of een woord voorkomt, zal afhangen van de vorm van de boom en dus van de volgorde waarin de woorden binnenkomen. Over het algemeen gaat het sneller dan bij de keten.

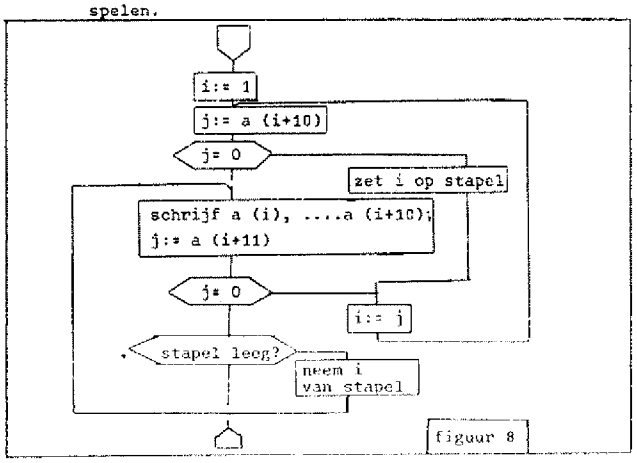
Ook het toevoegen van een nieuw woord gaat snel: zet het maar op de vrije plaatsen en werk enkele links bij.

Een speciale moeilijkheid bij het gebruik van een boom is hoe we er voor kunnen zorgen dat de woorden na afloop in alfabetische volgorde worden afgedrukt. (Bij de andere methoden, sequentieel en keten,vormde dit geen probleem.) We moeten de woorden in goede volgorde van de boom plukken: een woord moet worden geplukt na alle woorden in zijn linker tak en voor alle woorden in zijn rechter tak. Vanwege de recursieve definitie van een boom ligt een recursieve procedure voor de hand (zie figuur 7). Door de aanroep plukboom (1) wordt dan de woordenlijst in alfabetische volgorde afgedrukt.

```
procedure plukboom (i); integer i;
begin integer j;
      j:= a [i+10];
      if j ≠ 0  then  plukboom (j);
      for j:= 0 step 1 until 9 do PRINT (a [i+j]);
      j:=  a [i+11];
      if j ≠ 0 then plukboom (j)
end;
```

figuur 7

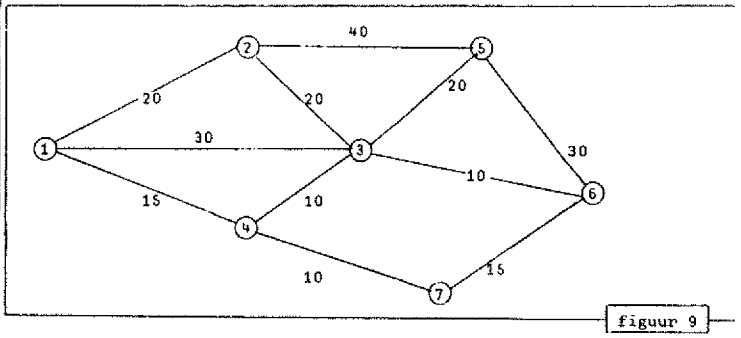
Als we het niet recursief doen moeten we zelf een stapel organiseren, waarin we onthouden langs welke weg we in de boom omhoog klimmen (figuur 8). Voor een goed begrip van dit in volgorde plukken van de woorden van de boom is het wel gewenst deze programma's op een eenvoudig voorbeeld na te spelen.



figuur 8

3. Kortstewegprobleem

Gegeven is een netwerk met punten (genummerd 1,2,....n) en lijnstukken tussen punten, de lijnstukken hebben gegeven lengten (zie figuur 9). Als punten i en j door een lijnstuk verbonden zijn, zullen we de lengte van dit lijnstuk aangeven met d (i, j).



figuur 9

Gevraagd de kortste weg langs deze lijnen van een gegeven punt b naar een gegeven punt e.

Voordat we ons afvragen hoe we het netwerk in het geheugen zullen representeren gaan we eerst na volgens welke methode we het probleem kunnen oplossen. We doen het zo:

We bouwen een verzameling S op van punten en onthouden van ieder punt $i \in S$ de afstand a(i) tot b. We beginnen met S= {b} en a(b)= 0.

Om een punt aan S toe te voegen bepalen we m= min{a(i)+d(i,j)}, het minimum genomen over alle paren (i,j) met $i \in S$, $j \notin S$, i en j door een lijnstuk verbonden. Als (i_o, j_o) het paar is waarvoor dit minimum wordt aangenomen, dan wordt j_o toegevoegd aan S, met $a(j_o)$= m.

We gaan hiermee door totdat e aan S wordt toegevoegd. Dan is a(e) de afstand van b tot e. (De kortste weg vindt men dan door van e uit terug te rekenen.)

Nu we de methode kennen weten we wat voor eisen we aan de wijze van representatie in het geheugen moeten stellen: We moeten bij een gegeven punt snel alle punten kunnen vinden die ermee verbonden zijn.

3.1. De matrix

Men kan het netwerk in het geheugen vastleggen in een tweedimensionaal array, g[1:n, 1:n] met

$$g[i,j] = \begin{cases} d(i,j) & \text{als i en j door lijnstuk verbonden} \\ -1 & \text{als i en j niet door lijnstuk verbonden.} \end{cases}$$

Dit is voor ons doel minder geschikt. De matrix neemt veel geheugenruimte in beslag en men kan niet snel alle met een punt verbonden punten vinden.

3.2. Sequentieel

Aannemend dat in een punt niet meer dan 5 lijnen samenkomen, reserveren we in een rij g voor ieder punt 10 plaatsen (voor de hoogstens 5 lijnstukken en hun lengten). Overtollige plaatsen worden gevuld met nullen (figuur 10).

RIJ g

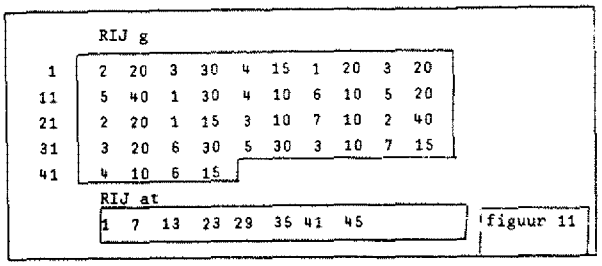
1	2	20	3	30	4	15	0	0	0	0
11	1	20	3	20	5	40	0	0	0	0
21	1	30	4	10	6	10	5	20	2	20
31	1	15	3	10	7	10	0	0	0	0
	enz.									

figuur 10

De punten bereikbaar vanuit punt i vindt men als g((i-1)x10+1), g((i-1)x10+3),....

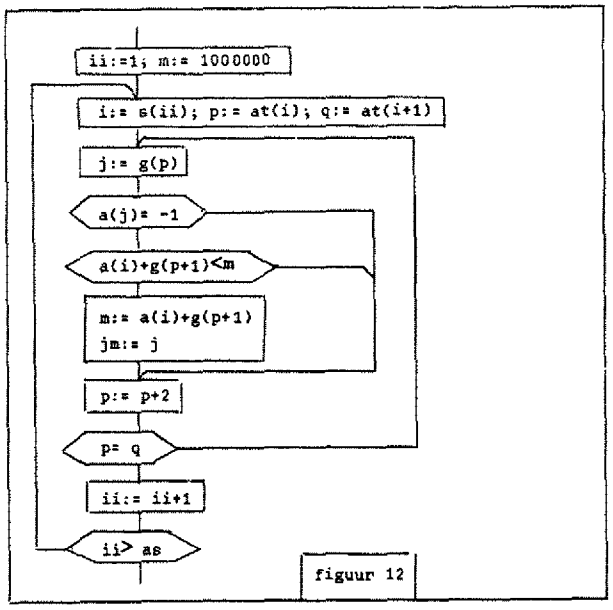
3.3. Sequentieel met acces-tabel

Het ruimteverlies van figuur 9, veroorzaak doordat niet alle rijen even lang zijn, kunnen we opheffen door alles direct tegen elkaar te schuiven.



figuur 11

Welk deel van g bij een punt i behoort vindt men met behulp van de accestabel at (figuur 11). Als at(i) = p en at(i+1) =q staan de punten bereikbaar vanuit punt i in g(p) tot en met g(q)-1.



figuur 12

In figuur 12 vindt men, voor het geval dat de gegevens van het netwerk zich in rijen g en at bevinden, de kern van de kortste weg algoritme, namelijk het bepalen van

$$m = \min_{i \in S,\ j \notin S} \{a(i)+d(i,j)\}$$

Er wordt aangenomen dat een rij s de punten van S bevat (aantal as) en een rij a, de afstanden en wel:

als $i \in S$ is a(i) de afstand van b tot i en

als $i \notin S$ is a(i)= -1.

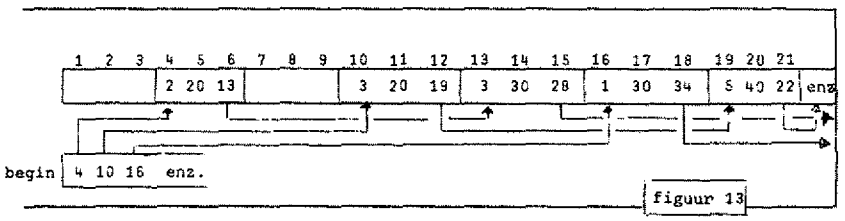
Bijv. als S bestaat uit de punten 1, 4 en 2 (zie figuur 9), dan is as= 3, s(1)= 1, s(2)= 4, s(3)= 2, a(1)= 0, a(2)= 20, a(4)= 15, a(3)= a(5)= a(6)= a(7)= -1.

In figuur 12 doorloopt ii de waarden van 1 tot as en dus i= s(ii) de punten van S. Voor ieder punt i worden de met i verbonden punten j bekeken. Als $j \in S$, dus a(j)≠ -1, komt deze j niet in aanmerking, als $j \notin S$ wordt nagegaan of a(i)+d(i,j) kleiner is dan het tot dan toe gevonden minimum m. Na afloop van het stuk programma in figuur 12 is m het gezochte minimum en jm de waarde van j waarvoor het wordt aangenomen.

3.4. Ketening

Extra moeilijkheden doen zich voor als het netwerk voortdurend aan wijzigingen onderhevig is. We willen dan niet alleen efficiënt de kortste weg bepalen, maar ook lijnstukken aan het netwerk toevoegen of uit het netwerk verwijderen. Daarvoor is de aanpak met een acces-tabel niet geschikt.

Hier kan men weer gebruik maken van ketens, een keten voor ieder punt. In een rij begin kan men dan vinden waar iedere keten begint (figuur 13).



figuur 13

Het toevoegen en verwijderen van lijnstukken gaat weer door het wijzigen van enkele links. Als men de bijkomende lijnstukken achteraan in de rij g laat aansluiten, loopt men echter de kans dat men achter in de rij vastloopt, terwijl er meer naar voren inmiddels vele lege plekken ontstaan zijn.

Daarom is het beter de lege plekken ook te ketenen. Als een lijnstuk wordt verwijderd, worden de plaatsen die het innam aan deze keten toegevoegd, heeft men nieuwe plaatsen nodig, dan neemt men ze van deze keten.

4. Slot

Het was niet de bedoeling in het voorgaande kant en klare oplossingen te bieden. Bij nadere beschouwing zal men ontdekken, dat er vele verfijningen kunnen worden aangebracht. Er is hier slechts een idee gegeven van hoe men in het geheugen structuren kan verwezenlijken.

.3 IETS OVER SOFTWARE

.3.1 Inleiding

onder hardware (apparatuur) wordt verstaan de omputer in de toestand, waarin deze verkeert op et moment dat hij in een krat door de fabrikant ordt opgeleverd. Het toestel is op dat moment og tot niets in staat. Als U de stekker in het topkontakt steekt, zullen er enige onvoorspelare chaotische reacties ontstaan binnen de kast, n daarmee uit. Om van deze nare 'ijzerwinkel' en bruikbare machine te maken, moeten er programma's in het computergeheugen worden opgenomen die vrijwel alle 'handelingen' van het apparaat regelen.

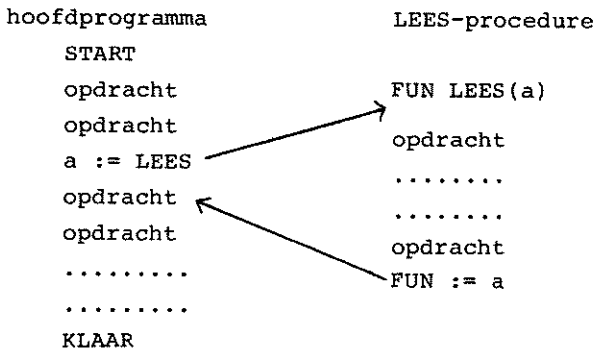
oor de meeste vakmensen wordt onder 'software' (programmatuur) alles verstaan wat de computer laat werken, uitgezonderd de hardware zelf. Onder deze software kan men dus ook verstaan de programma's, die door (of voor) de gebruiker (ook U en Uw leerlingen) zijn geschreven. Dit wordt wel de 'zachte programmatuur' (applicatie software) genoemd.
De 'harde programmatuur' (basis software) bestaat uit de programma's, die de elementaire verrichtingen van de machine en zijn bijbehorende apparaten regelen. Deze programma's worden door specialisten (al of niet in dienst van de fabrikant) geschreven. Veel van deze programma's zijn permanent in het geheugen van de computer aanwezig en vaak zodanig beveiligd dat ze niet door gebruikers kunnen worden 'beschadigd'.
Over deze harde programmatuur gaat de nu volgende beschouwing.

Om U een indruk te geven van harde programmatuur nu een voorbeeld. U bent gewend om U te kunnen bedienen van de opdracht a := LEES. Daarbij moet het getal -851.37 in de geheugenplaats a worden opgenomen. Hoe weet de machine waar precies het getal ophoudt? Een hoe wordt dit getal, met toestandsteken en decimale punt, omgezet in de interne notatie bestaande uit alleen nullen en enen van een geheugenplaats?

Het antwoord op beide vragen is, dat dit geregeld moet worden door een programma. Aangenomen dat de LEESSYM-opdracht bekend is, zodat symbool voor symbool gelezen kan worden, ziet dit programma er bijvoorbeeld als volgt uit:

```
     FUN LEES(a)
     teken := LEESSYM
     a := 0
1    b := LEESSYM
     ALS b > 9 DAN 3 ;hier wordt gekontroleerd of we
                     ;iets anders dan een cijfer te-
                     ;genkomen
2    a := 10 × a + b ;hier ziet U het 'effect' van
                     ;het tientallig stelsel
     NAAR    1
3    ALS b = 70 DAN 4 ANDERS 7  ;hier wordt gezocht
                                ;naar decimale punt
4    d := 1
5    d := d/10       ;na de decimale punt wordt het
                     ;gelezen symbool door de juiste
                     ;macht van 10 gedeeld
     c := LEESSYM
     ALS c > 9 DAN 7
6    a := a + c × d
     NAAR 5
7    ALS teken = 81 DAN 8 ANDERS 9  ;negatief zijn
                                    ;wordt verwerkt
8    a := -a
9    FUN := a
```

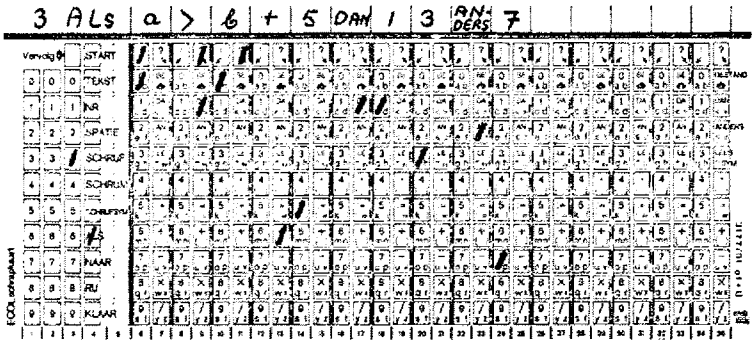
Dit programma is niet in overeenstemming met de ECOL-werkelijkheid. In hoofdstuk 3 kunt U lezen, dat er in het getal spaties mogen voorkomen. Dit programma zou het lezen van het getal beëindigen bij een spatie, want een tweede punt en elk ander niet-cijfer symbool geldt hier als stopcriterium. Als dit symbool het toestandsteken van het volgende getal is, zou dit bewaard moeten worden tot de volgende keer. Voor de duidelijkheid zien we hier maar van af.
Dat dit programma niet wordt ingesloten door START en KLAAR vindt zijn oorzaak in het feit, dat dit programma dient als onderdeel van een langer programma, zie onderstaande figuur .



Hoofdstuk 3 licht U in over de preciese werking van het FUNctie-mechanisme. Enkele aspekten die in dit voorbeeld niet gemist kunnen worden zijn:

- Men kan bij het aanroepen van de LEES-procedure een parameter meegeven, te weten: de geheugenplaats waarin het te lezen getal terecht moet komen.
- De in de LEES-procedure genoemde namen en regelnummers mogen niet verward worden met eventueel identieke namen en nummers in het hoofdprogramma. U begrijpt dat hier nog wel enige software aan te pas komt.

Ook de opdracht LEESSYM is overigens niet zonder meer duidelijk aan de machine. Zolang het gelezen symbool een cijfer is, is het probleem minder groot als bij het lezen van een letter of een leesteken. In het geheugen kunnen deze laatsten nu eenmaal niet worden opgeslagen. Daarom worden de laatstgenoemde symbolen omgezet in getallen. We zullen dit de interne code noemen. Ook dit omzetten gebeurt met een programma. Dit programma kan signalen ontvangen van een ponsbandlezer of een schrapkaartlezer en deze 'omrekenen' tot interne code. Bijvoorbeeld de schrapkaart



veroorzaakt de volgende 'signalen':

```
0000010010100000000000000000000000000
0000010001000000000000000000000000000
0000000010000000110000000000000000000
0000000000000000000000100000000000000
0010000000000000000100000000000000000
0000000000000000000000000000000000000
0000000000000100000000000000000000000
0001000000001000000000000000000000000
0000000000000000000000001000000000000
0000000000000000000000000000000000000
0000000000000000000000000000000000000
```

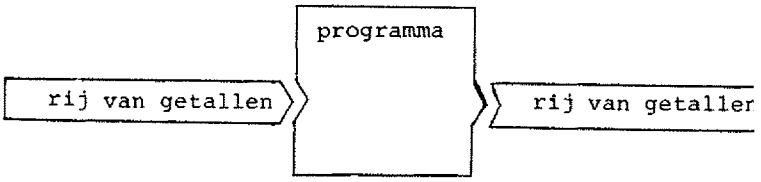
Deze signalen worden omgezet in de volgende getallen:

```
90 90 3 90 90 90 40 51 58 90 10 86 11
 *  * 3  *  *  *  A  L  S  *  a  >  b

80 5 90 43 40 53 90 1 3 90 40 53 43 44 57 58 90
 + 5  *  D  A  N  * 1 3  *  A  N  D  E  R  S  *

7 90 90 90 90 90
7  *  *  *  *  *
```

Deze interne code is overigens afhankelijk van de machine en zal bovendien in het algemeen in binaire code worden geschreven. Hier is de ECOL-code uit Compk § 16.1 gebruikt.

5.3.2 Schema van een programma: maverwerking

Zoals bij het laatste voorbeeld heel duidelijk te zien is, zal een softwareprogramma vaak een rij van getallen laten omwerken tot een rij van andere getallen. Symbolisch zullen we dit alsvolgt aangeven:

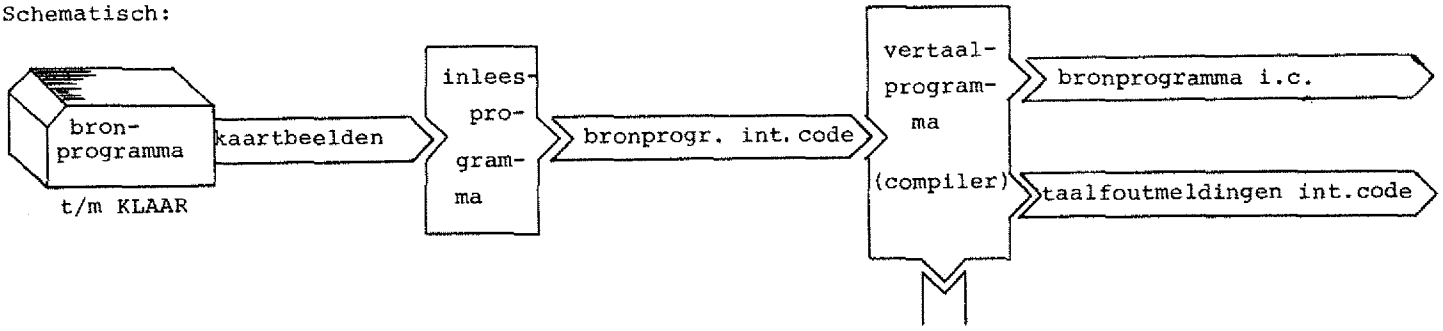


Met opzet staat hier 'laten omwerken', omdat het in wezen niet het programma zelf is, waar de getallen ingaan en uitkomen. Het is het rekengedeelte van de computer, waarin dit gebeurt onder invloed van het programma.
Het inlezen van schrapkaarten kan nu als volgt worden geschematiseerd.

Wanneer U als gebruiker een programma indient op schrapkaarten, dan wordt dit op dezelfde wijze in interne code omgezet. Maar hiermee moet nog wel iets gebeuren, want dat 40 51 58.... enzovoort een voorwaardelijke sprongopdracht voorstelt is aan de computer niet duidelijk.

Wat wèl door een machine wordt verstaan, kunt U vinden in de paragraaf over hardware, namelijk machinecode-opdrachten van een eenvoudiger soort, binair weergegeven.
Uw ingediende programma (het zogenaamde bronprogramma) moet worden omgezet in machinecode. Het wordt na deze behandeling objectprogramma genoemd. En dat gebeurt niet automatisch (hardware), maar weer onder invloed van een software-programma.

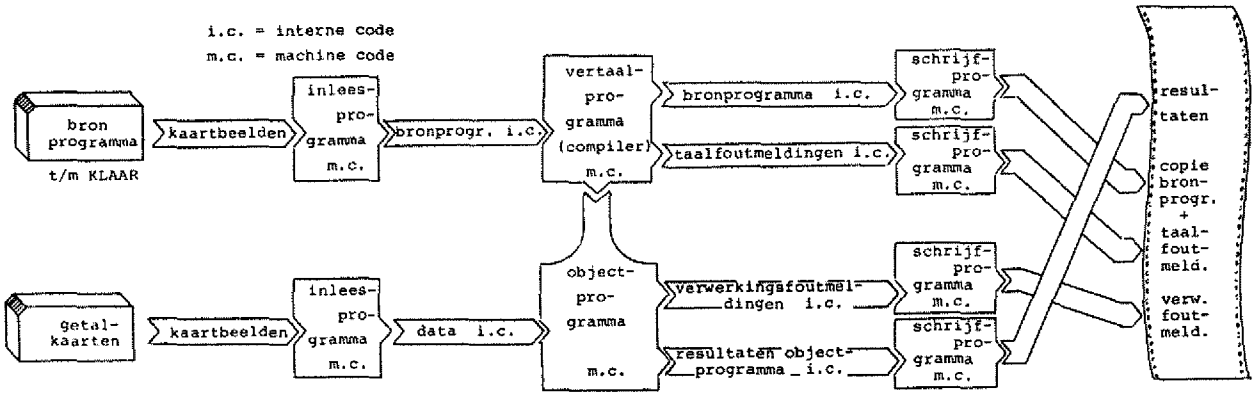
Schematisch:



Uit dit vertaalprogramma komen weer rijen getallen. Het eerst komt een herhaling van het bronprogramma. Als de interne code hiervan is omgezet in leesbaar schrift, kan de schrijver van het bronprogramma controleren of alles goed is doorgekomen. Daarna verschijnen eventueel de taalfoutmeldingen. Als een bepaalde regel van het programma luidde:

ALS 7 LEES a ANDERS

dan kan het vertaalprogramma daar met de beste wil van de wereld niets mee beginnen. Dit programma produceert dan een tekst, waaruit de fout blijkt. Dit is dan een fout tegen de ECOL-taal.
Tenslotte komt er een rij binaire getallen, dat de vertaling is van het bronprogramma in het objectprogramma (machinecode). Deze rij onderscheidt zich niet van de andere programma's in machinecode (inleesprogramma, vertaalprogramma, schrijfprogramma), zodat het in het schema hieronder rechthoekig is weergegeven. Het bijzondere van het vertaalprogramma is dus, dat één van zijn 'afscheidingsprodukten' weer een programma (het objectprogramma) is.



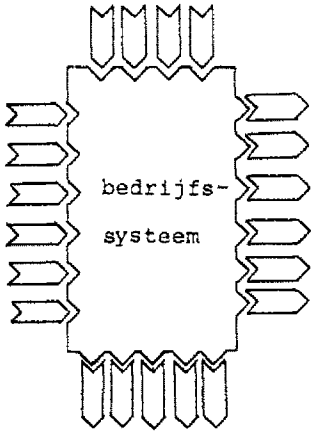
Het objectprogramma in machinecode kan dan aan het werk, waarbij eventuele getallen van getalkaarten als voedsel worden toegediend. Uitvoer van het objectprogramma bestaat natuurlijk uit de resultaten, maar zo mogelijk ook uit meldingen van fouten die bij de verwerking van het programma blijken (executiefouten). Als het bronprogramma tien keer een LEES-opdracht bevat, terwijl maar zes getallen op de getalkaarten aanwezig zijn, is dat niet een fout tegen de ECOL-taal die door de vertaler wordt ontdekt, maar een fout die blijkt als het objectprogramma werkelijk draait. Een andere veel voorkomende executiefout is de eindeloze cyclus.

Er is een derde type fout, die niet in dit schema (figuur 1) is aangegeven: de schrapfout. Het is mogelijk dat een leerling een niet geoorloofde combinatie van hokjes aanstreept (bijvoorbeeld START en KLAAR tegelijk in kolom 4). Het inleesprogramma signaleert dit, maar geeft de foutmelding niet direkt naar buiten, maar geeft in interne code een "niet bestaand symbool" door, waarna de compiler voor foutmelding zorgdraagt.

U ziet aan alle software die aan de programmaverwerking te pas komt, dat zonder deze hardeprogrammatuur de computer een onhandelbaar ding is. De hardware is slechts het zichtbare fundament, waarop de software de ruime toepasbaarheid bouwt.
In de inleiding werd al gezegd: als een computerinstallatie bij U zou worden binnengebracht en U

steekt de stekker in het stopkontakt, dan is het toestel nog niet in staat iets nuttigs te doen. Er is nog geen compiler, er zijn nog geen inlees- of schrijfprogramma's. Gelukkig is er door de fabrikant wel een kleine toegang tot het computergeheugen ingebouwd, de zogenaamde 'hardware loader'. Met een speciale code kunnen hierdoor kleine programma's worden ingevoerd, als het ware met een theelepeltje. Onder de software, die als eerste naar binnen wordt gelepeld, zal men een redelijk inleesprogramma aantreffen, zodat men tenminste met de paplepel kan verdergaan.

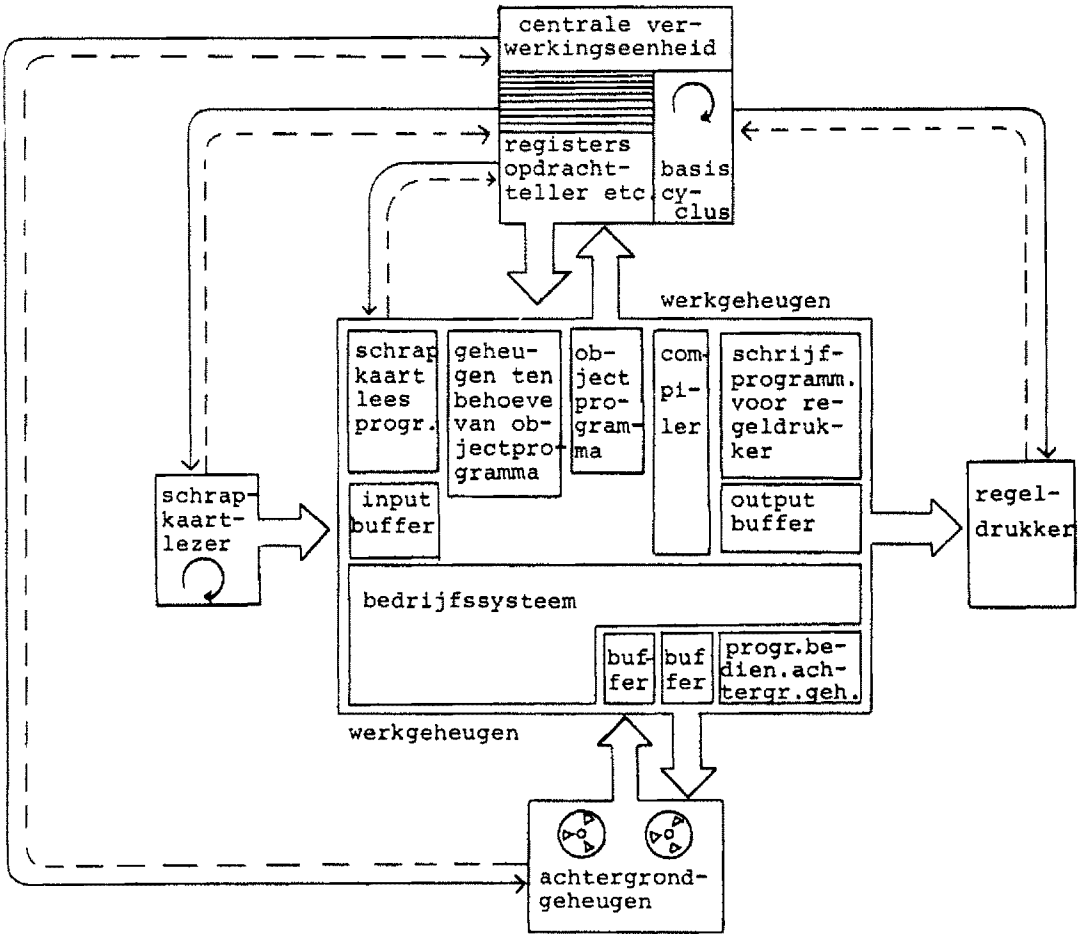
Het bovenstaande schema (figuur 1) is trouwens nog maar een gedeelte van de werkelijkheid. Een moderne centrale verwerkingseenheid van een computer, waarvoor een miljoen geheugencontacten per seconde geen uitzondering is, zal aan de hier geschetste verwerking maar een gering deel van zijn tijd besteden. De meeste tijd gaat zitten in het inlezen en schrijven. Bij moderne configuraties zijn meerdere objectprogramma's tegelijk in verwerking. 'Tegelijk' in dit geval op te vatten als: het eerste programma rekent, terwijl het tweede leest, het derde schrijft en het vierde even staat te wachten. Deze wirwar mag niet ontaarden in een chaos. De organisatie van dit geheel berust bij....., U raadt het al, een programma. Dit programma onderhoudt kontakt met zoveel onderdelen en zoveel andere software, dat de volgende symbolisering niet overdreven is.



Dit programma wordt ook wel bedrijfssysteem genoemd. Het samenstellen van bedrijfssystemen is een hooggekwalificeerde arbeid, waarvoor nog steeds te weinig mensen zijn opgeleid. Onder andere hieraan ontleent het Hoger Informatica Onderwijs in Nederland zijn bestaansrecht. De ontwikkeling van bedrijfssystemen is nog ver achter bij de mogelijkheden, die de hardware biedt. Voor het "bouwen" van compilers geldt overigens hetzelfde.

5.3.3 Geheugengebruik

Figuur 1 geeft geen overzicht naar plaats maar naar tijd aan. De vier schrijfprogramma's in dit schema geven bijvoorbeeld in wezen één programma aan, op vier verschillende tijdstippen aangeroepen. In figuur 2 wordt een poging gedaan een sterk vereenvoudigde en geschematiseerde plaatsaanduiding te geven.



figuur 2

We gaan er weer vanuit, dat sprake is van één objectprogramma.

In figuur 2 geeft ⇨ een kanaal aan waardoor informatie kan stromen (de rijen getallen uit de vorige paragraaf), ⟶ een commandolijn voor de besturing vanuit de centrale verwerkingseenheid en - - → een verbinding voor berichten aan deze eenheid.

Het voornaamste nieuwe element is het begrip buffer. Een regeldrukker is in staat een regel van meer dan 100 symbolen in één keer af te drukken. Als het objectprogramma allemaal getallen naast elkaar laat schrijven, worden deze niet stuk voor stuk geschreven, maar verzameld tot een hele regel gevuld is. De verzamelplaats voor zo'n regel heet de output buffer. In de meeste gevallen zal het schrijfprogramma zorgen dat bij overlopen van de buffer het laatste getal niet wordt afgebroken, maar bewaard tot de volgende regel. De regel kan dan, of bij eerder ontvangen van de opdracht "nieuwe regel", in één klap worden afgedrukt.
De inputbuffer speelt een soortgelijke rol in omgekeerde richting. Wanneer een schrapkaart gelezen wordt, zal deze niet schrapje voor schrapje worden doorgegeven, maar er wordt een kompleet kaartbeeld in een buffer opgeslagen. Het schrapkaartleesprogramma kan nu netjes de kolommen tot superkolommen samenvoegen, detecteren op onjuiste aanstrepingen en dergelijke. Omdat het verkeer met het achtergrondgeheugen (werken met bestanden) in beide richtingen gaat, zijn hiervoor twee buffers.

Het programma voor de bediening van het achtergrondgeheugen regelt zaken als het opzoeken van het juiste stuk magneetband en het onderhouden van de buffers.

De dunne pijlen geven in het schema de commando- en berichtenlijnen aan. Als het objectprogramma een leesopdracht bevat, wordt de schrapkaartlezer volgens de getrokken lijn geaktiveerd. De lezer op zijn beurt signaleert langs de gestippelde lijn het gereedstaan voor een volgende opdracht. De schrapkaartlezer houdt er een eigen (vrij langzaam) tempo op na, vandaar de eigen aanduiding van een basiscyclus.
Bij de meeste computerfiguraties moet U denken aan meer lees- en schrijfapparatuur en meer achtergrondgeheugen, de zogenaamde randapparatuur of perifere apparatuur, elk met hun eigen buffers in het werkgeheugen.

Bovendien zijn, zoals in de vorige paragraaf gezegd, meestal meerdere gebruikersprogramma's tegelijk in het geheugen opgenomen. Tenslotte zijn bij echte grote computers soms meerdere centrale verwerkingseenheden ingeschakeld. Het is overigens vrijwel nooit zo, dat deze aan eenzelfde taak (eenzelfde objectprogramma) werken. Dus in dat geval kunt U de computer denken te bestaan uit meerdere rekenmachines, die één werkgeheugen gebruiken.
Het bedrijfssysteem dat bij zo'n computer orde houdt, is van indrukwekkende omvang.

5.4 RECURSIVITEIT

Aan de hand van een aantal voorbeelden zal getracht worden het begrip recursiviteit duidelijk te maken.

Voorbeeld 1) In een syntax staat:

onder [rij kleine letters] wordt verstaan: óf [kleine letter] óf [rij kleine letters] [kleine letter]

Hier staat een definitie van het begrip "rij kleine letters". Rechts staat [kleine letter]. De tekst-haken geven aan, dat het om het begrip gaat en niet om de 12 letters en één spatie die tussen deze haken staan. Het begrip [kleine letter] moet elders in de syntax zijn gedefinieerd, bijvoorbeeld door opsomming van alle symbolen die men onder een [kleine letter] wenst te verstaan. De recursie bestaat nu uit het gebruik van het te definiëren begrip in de definitie. Men ziet immers aan de rechterzijde weer [rij kleine letters] voorkomen. Met [rij kleine letters] [kleine letter]wordt bedoeld een [rij kleine letters] gevolgd door een [kleine letter].

Is pbda een rij kleine letters? De vraag beantwoorden we door op te bouwen:

p is een [kleine letter] dus (volgens bovenste regel van de definitie) en [rij kleine letters]

pb is de [rij kleine letters]p, gevolgd door de [kleine letter]b, en dus (onderste definitieregel) en [rij kleine letters]

pbd is de [rij kleine letters]pb, gevolgd door de [kleine letter]d en dus een [rij kleine letters]

pbda is [rij kleine letters] [kleine letter] dus [rij kleine letters].

Hieruit blijkt dat het gebruik van het te definiëren begrip in de definitie zinvol kan zijn, maar dan alleen als de constructie zodanig is, dat er iets mee kan worden 'opgebouwd' of, als U achterstevoren zou redeneren, kan worden 'afgebroken'.

Constructies als

onder [rij lettertjes] wordt verstaan: óf [lettertje] óf [rij lettertjes]

of nog erger

onder [rij lettertjes] wordt verstaan: [rij lettertjes]

definiëren niets.

Voorbeeld 2)

onder a^n $n \in \{1,2,3,...\}$	wordt verstaan:	a	als $n=1$
		$a \times a^{n-1}$	als $n>1$

Volgens deze definitie wordt onder a^5 verstaan:

$a \times a^4$ en daaronder wordt verstaan
$a \times a \times a^3$ en dat is
$a \times a \times a \times a^2 =$
$a \times a \times a \times a \times a^1 =$
$a \times a \times a \times a \times a$

U ziet hier de exponent met behulp van de definitie 'afbreken'. In voorbeeld 1 was het nut van de recursieve definitie niet erg duidelijk, omdat een omschrijving in normaal Nederlands niet veel langer zou zijn. Het laatste voorbeeld laat zien, dat een recursieve definitie niet alleen overzichtelijker, maar ook korter kan zijn dan een andere.

Voorbeeld 3) Zij R(n,m) de rest bij deling van n door m, dan wordt onder

GGD(n,m) $n \in \{1,2,3,...\}$ $m \in \{1,2,3,...\}$ $n \geq m$	verstaan:	m	als R(n,m) = 0
		GGD(m,R(n,m))	als R(n,m)>0

Wat is GGD(231,60) ?

R(231,60)	= 51	dus GGD(231,60)	= GGD(60,51)
R(60,51)	= 9	dus GGD(60,51)	= GGD(51,9)
R(51,9)	= 6	dus GGD(51,9)	= GGD(9,6)
R(9,6)	= 3	dus GGD(9,6)	= GGD(6,3)
R(6,3)	= 0	dus GGD(6,3)	= 3

Dit is een weergave van de algoritme van Euclides, meestal als volgt uitgevoerd:

```
60/231/3
   180
    51/60/1
       51
        9/51/5
          45
           6/9/1
             6
           → 3/6/2
               6
               0
```

Voorbeeld 4)

onder n!	wordt verstaan:	1	als n=0
n∈ℕ		n×(n-1)!	als n>0

Dit voorbeeld zal na de vorige voorbeelden zonder meer duidelijk zijn. Dergelijke recursies kunnen bij het programmeren worden overgenomen.

```
      FUN fac(n)
      ALS G(n)=n EN n≥0 DAN 1
      NR
      TEKST := "ongeoorloofde parameter"
      a := -1
      NAAR 3
1     ALS n>0 DAN 2
      a := 1
      NAAR 3
2     a := n×fac(n-1)
3     FUN := a
```

} controle op het juiste karakter van n

In het programma dat hierop zou volgen kan van faculteiten van natuurlijke getallen gebruik gemaakt worden. Echter fac(3.5) = fac(-5) = -1 met een tekst als een soort foutmelding.

Voorbeeld 5) De Ackermann-functie

onder A(m,n)	wordt verstaan:	n+1	als m=0
m∈ℕ		A(m-1,1)	als n=0
n∈ℕ		A(m-1,A(m,n-1))	als m×n>0

Deze zo schijnbaar eenvoudige definitie geeft al snel aanleiding tot onverkwikkelijke berekeningen, die zonder computer uiterst tijdrovend zijn. Als voorbeeld A(2,1).

```
A(2,1) =                          A(1,3) =
A(1,A(2,0)) =                     A(0,A(1,2)) =
A(1,A(1,1)) =                     A(0,A(0,A(1,1))) =
A(1,A(0,A(1,0))) =                A(0,A(0,A(0,A(1,0)))) =
A(1,A(0,A(0,1))) =                A(0,A(0,A(0,A(0,1)))) =
A(1,A(0,2)) =                     A(0,A(0,A(0,2))) =
A(1,3) =                          A(0,A(0,3)) =
                                  A(0,4) = 5
```

U ziet dat bij het uitwerken van de definitie de problemen zich opstapelen, totdat een vorm bereikt wordt waarvan de waarde bekend is (A(0,1)=2). Daarna verlagen we de stapel weer tot we vastlopen bij A(1,3). De problemen stapelen zich weer op tot een waarde die zonder meer bekend is. Daarna kunnen we terugkeren naar de bodem van de stapel. Dit werken met een stapel is een veelvoorkomend principe bij computers. In de meeste gevallen zijn zelfs hardware voorzieningen voor het werken met een stapel ingebouwd (zie 4.6 in § 5.1).

Voorbeeld 6) Het partitieprobleem.

Onder een partitie van een positief geheel getal verstaan we dit getal zelf of een wijze om dit getal

te schrijven als som van een aantal positieve gehele getallen. Zo is 2+2+1 een partitie van 5.
Alle verschillende partities van 5 zijn: 5, 4+1, 3+2, 3+1+1, 2+2+1, 2+1+1+1, 1+1+1+1+1.
U ziet hieruit dat 4+1 en 1+4 niet als verschillende partities zullen worden aangemerkt. Partities schrijven we voortaan in niet stijgende termen. Een partitie van 5 kan dus beginnen met een 5, met een 4, met een 3, met een 2 of met een 1. Een partitie van 5 beginnend met een 2 heeft de gedaante 2 + [partitie van (5-2)]. Maar niet elke [partitie van 3] is hierin toegestaan, want 2+3 is tegen de zojuist gemaakte afspraak. Daarom is het volgende wèl juist.
Een partitie van 5 beginnend met 2 heeft de gedaante 2 + [partitie van (5-2) met termen $\leqslant$ 2].
In het algemeen heeft een partitie van a de gedaante:

n + [partitie van (a-n) met termen $\leqslant$ n]

$n \leqslant a$, a en n positief geheel.

De probleemstelling luidt nu: bepaal het <u>aantal</u> partities van een positief geheel getal a. Voor de berekening hiervan voeren we de grootheid P(m,n) in, zijnde het <u>aantal</u> partities van m met termen $\leqslant$ n. Het aantal partities van 5, dat is P(5,5) kan bepaald worden uit:

$$P(5,5)= 1+ P(1,4) + P(2,3) + P(3,2) + P(4,1).$$

De eerste 1 geeft de triviale partitie van 5 aan, namelijk 5 zelf. P(4,1) is het aantal partities van 4 met termen $\leqslant$ 1, dus uitsluitend enen. Er is maar één zo'n partitie, namelijk 1+1+1+1.
In het algemeen is P(m,1) = 1. Ook P(1,n) = 1, want het getal 1 heeft slechts de triviale partitie.
Verder volgt uit de definitie dat bijvoorbeeld P(3,3) = P(3,4) = P(3,5) enz.
Terwijl P(3,3) = P(3,2) + de triviale partitie.
In het algemeen: P(m,n) = P(m,m-1) + 1 als $m \leqslant n$.
Nu nog het geval dat $m>n>1$. De recursieve relatie van P(m,n) moet bestaan met P-functies waarvan de m of de n lager is. Bijvoorbeeld:

$$\overbrace{3+2\ ,\ 3+1+1\ ,\ \underbrace{2+2+1\ ,\ 2+1+1+1\ ,\ 1+1+1+1+1}_{P(5,2)}}^{P(5,3)}$$

P(5,3) kan verknipt worden in iets+P(5,3-1). Dat iets is het aantal partities van 5, die met een 3 beginnen, dat is het aantal partities van 5-3, met termen kleiner dan 3, ofwel P(5-3,3).
Kortom P(5,3) = P(5-3,3) + P(5,3-1).
In het algemeen P(m,n) = P(m-n,n) + P(m,n-1) als $m>n>1$.

Het aantal partities P(a,a) van het positieve gehele getal a kan berekend worden met de recurente betrekking:

P(m,n) =	1	als m of n = 1
	P(m,m-1)+1	als $m \leqslant n$
	P(m-n,n)+P(m,n-1)	als $m>n>1$

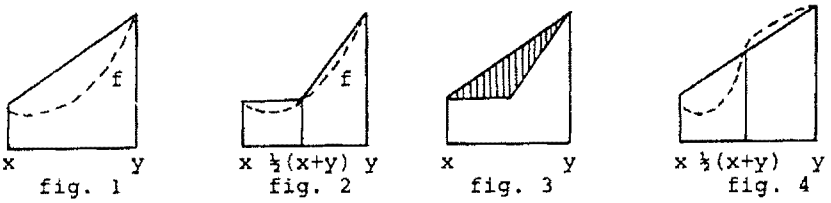
Een programma hiervoor, waarin de waarden van m en n niet in de FUN-deklaratie worden gekontroleerd is het volgende:

```
        FUN p(m,n)
        ALS m=1 OF n=1 DAN 1 ANDERS 2
1       b := 1
        NAAR 4
2       ALS m>n DAN 3
        b := p(m,m-1) + 1
        NAAR 4
3       b := p(m-n,n)+p(m,n-1)
4       FUN := b
        START
        MET 1, a:=1, 20
        NR
        TEKST := "P("
        SCHRIJF(2,0) := a
        SCHRIJF SYM := 71
        SCHRIJF(2,0) := a
        TEKST := ")="
        SCHRIJF(6,0) := P(a,a)
        HERHAAL
        KLAAR
```

Uitwerkingen:

```
P(  1 ,  1 ) =     1
P(  2 ,  2 ) =     2
P(  3 ,  3 ) =     3
P(  4 ,  4 ) =     5
P(  5 ,  5 ) =     7
P(  6 ,  6 ) =    11
P(  7 ,  7 ) =    15
P(  8 ,  8 ) =    22
P(  9 ,  9 ) =    30
P( 10 , 10 ) =    42
P( 11 , 11 ) =    56
P( 12 , 12 ) =    77
P( 13 , 13 ) =   101
P( 14 , 14 ) =   135
P( 15 , 15 ) =   176
P( 16 , 16 ) =   231
P( 17 , 17 ) =   297
P( 18 , 18 ) =   385
P( 19 , 19 ) =   490
P( 20 , 20 ) =   627
```

Voorbeeld 7) De trapeziumregel.



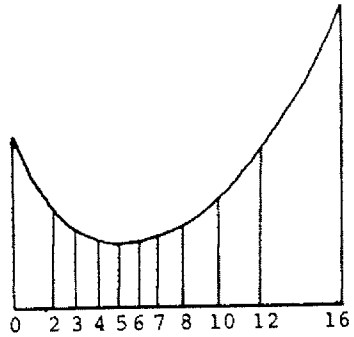
De oppervlakte $\frac{1}{2}(y-x)(f(x)+f(y))$ van het trapezium is een benadering voor $\int_x^y f(t)dt$ (fig. 1). We zullen in het vervolg deze oppervlakte aangeven met $T(x,y)$.
In het algemeen zal $T(x,\frac{1}{2}(x+y))+T(\frac{1}{2}(x+y),y)$ een betere benadering geven (fig. 2). Een criterium voor de precisie kan zijn: $V(x,y) = T(x,y) - T(x,\frac{1}{2}(x+y)) - T(\frac{1}{2}(x+y),y)$ (fig. 3).
Op dit laatste zijn wel uitzonderingen te bedenken, zoals in fig. 4, waarin $V(x,y)=0$, zonder dat er van een goede benadering sprake is. We zonderen dit soort gevallen uit door te eisen dat $f''(t)\neq 0$ op (x,y), zodat de funktie f op dit interval uitsluitend convex of uitsluitend concaaf is.

De benadering $I(a,b)$ van $\int_a^b f(t)dt$, met $f''(t)\neq 0$ op (a,b) kan berekend worden met een tolerantie ε met de volgende recursie:

$$\boxed{\begin{array}{lll} I(x,y) = & T(x,\frac{1}{2}(x+y)) + T(\frac{1}{2}(x+y),y) & \text{als } |V(x,y)| < \varepsilon.\frac{y-x}{b-a} \\ \text{anders} & I(x,\frac{1}{2}(x+y)) + I(\frac{1}{2}(x+y),y) & \end{array}}$$

Hoe kleiner het interval $[x,y]$, hoe kleiner de "fout" $V(x,y)$ die we mogen toestaan. De som van alle foutjes op de intervallen mag immers ε niet te boven gaan, zodat per interval slechts het $\frac{y-x}{b-a}$ - deel van ε getolereerd kan worden.
De recursieve toepassing van de trapeziumregel heeft het voordeel, dat niet meer gerekend wordt dan strikt noodzakelijk. Dit wil zeggen, er wordt alleen tot halvering van een interval overgegaan, als het nauwkeurigheidscriterium op dat interval niet gehaald wordt.

fig. 5



Stel $I(0,16)$ van een zeker funktie moet bepaald worden met een tolerantie 0,32.
Men schrijft een programma dat werkt volgens de aangegeven recursie (fig. 5).
$V(0,16)>0,32 \Rightarrow I(0,16) = I(0,8) + I(8,16)$.
De computer legt $I(8,16)$ even apart en gaat verder met $I(0,8)$.

$V(0,8) \geq 0,16 \Rightarrow I(0,8) = I(0,4) + I(4,8)$ $\quad$ $I(4,8)$ apart leggen, bovenop $I(8,16)$
$V(0,4) \geq 0,8 \Rightarrow I(0,4) = I(0,2) + I(2,4)$
$V(0,2) < 0,4 \Rightarrow I(0,2)$ wordt berekend. Er ligt nu een stapel te wachten van de inhoud: → $\boxed{\begin{array}{l} I(2,4) \\ I(4,8) \\ I(8,16) \end{array}}$
De bovenste van de stapel komt nu het eerst aan bod (het principe LIFO, Last In - First Out)
$V(2,4) \geq 0,4 \Rightarrow I(2,4) = I(2,3) + I(3,4)$ → $\boxed{\begin{array}{l} I(3,4) \\ I(4,8) \\ I(8,16) \end{array}}$
$V(2,3) < 0,2 \Rightarrow I(2,3)$ berekend
$V(3,4) < 0,2 \Rightarrow I(3,4)$ berekend $\quad$ $I(2,4)$ berekend
$I(0,4)$ berekend
$V(4,8) \geq 0,8 \Rightarrow I(4,8) = I(4,6) + I(6,8)$ → $\boxed{\begin{array}{l} I(6,8) \\ I(8,16) \end{array}}$
$V(4,6) \geq 0,4 \Rightarrow I(4,6) = I(4,5) + I(5,6)$ → $\boxed{\begin{array}{l} I(5,6) \\ I(6,8) \\ I(8,16) \end{array}}$
enz.

U ziet hierbij weer het principe van de stapel.

Slotopmerking:

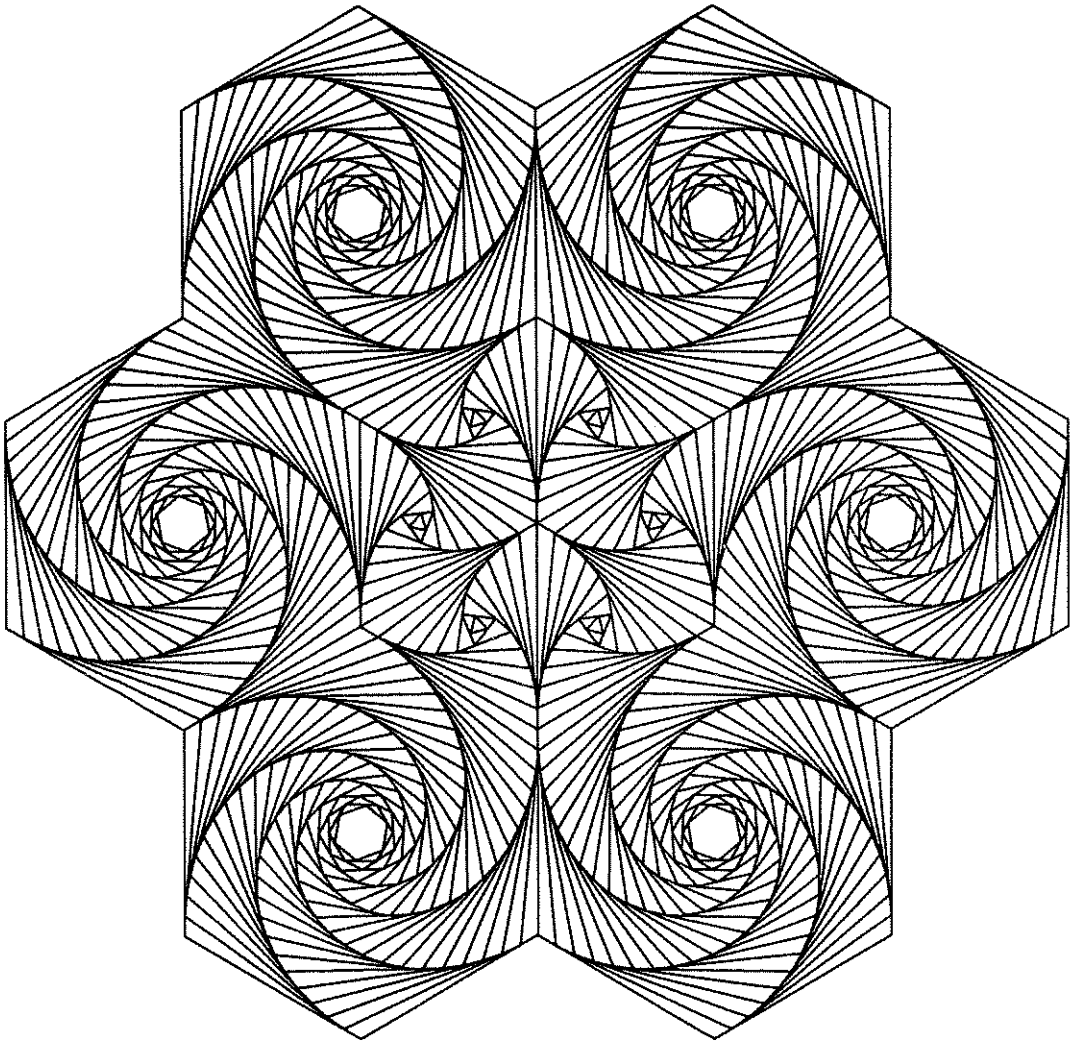
Enkele van de gegeven voorbeelden hebben een minder uitgesproken recursief karakter. Ze zijn ook te definiëren (of te berekenen) door een herhaling van stappen, die omschreven kunnen worden in termen van reeds gedefinieerde begrippen (iteratie).

In voorbeeld 3 kan de definitie luiden: deel het kleinste op het grootste getal. Als de rest niet nul is, neem dan de deler als deeltal en de rest als deler en voer deze deling uit. Herhaal de laatste regel, totdat de rest nul is. De GGD is gelijk aan de laatste deler. Zo kan ook de berekening van n! als een iteratief proces worden aangegeven:

```
      prod := 1
      m := 1
1     prod := prod × m
      ALS m=n DAN 2
      m := m+1
      NAAR 1
2     nfac := prod
```

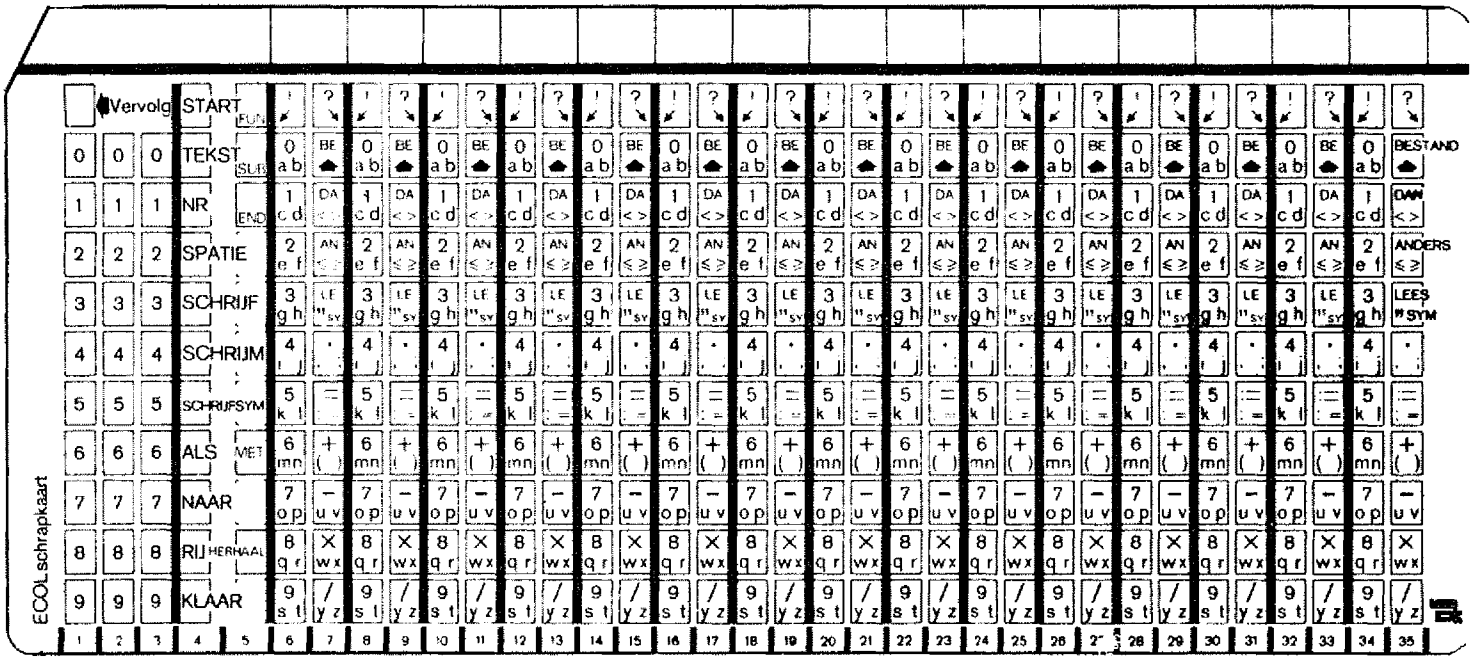
Tenslotte kan ook de integraal van voorbeeld 7 volgens een iteratie benaderd worden, maar dan door telkenmale halvering van <u>alle</u> intervallen. $I(a,b) = I(a,b,n)$ als $|I(a,b,n) - I(a,b,n-1)| < \varepsilon$ waarbij voor

$$\Delta t = \frac{b-a}{2^n}, \quad I(a,b) = T(a,a+\Delta t) + T(a+\Delta t, a+2\Delta t) + \ldots\ldots\ldots\ldots T(b-\Delta t, b)$$



6 Programmaverwerking

6.1 DE ECOL-SCHRAPKAART



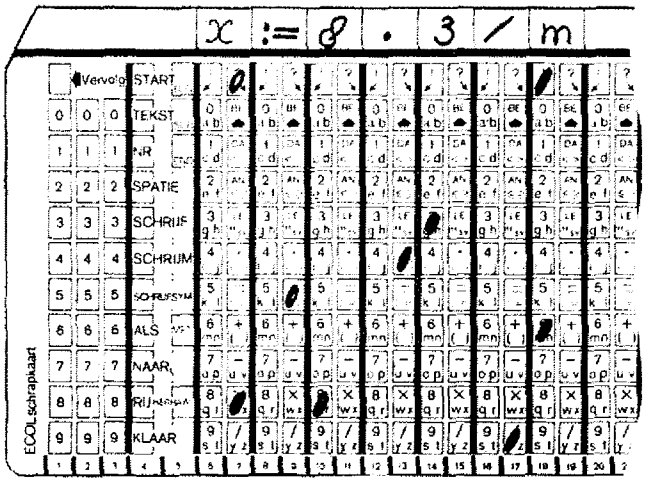
De schrapkaart is U bekend van de volkstelling, maar dan met andere opdruk. Het gebruik is als volgt:

1) Eén regel van het programma wordt op één of meer kaarten aangegeven
2) De regel wordt geschreven aan de bovenzijde van de kaart, één symbool per hokje (afgezien van regelnummers en hoofdletterwoorden)
3) De regel wordt daarna gecodeerd door potloodstreepjes in de aangegeven hokjes
4) De kolommen 1, 2 en 3 dienen voor het aangeven van een regelnummer
5) Kolom 4 voor de aangegeven hoofdletterwoorden, die alle aan het begin van een opdracht staan
6) Kolom 5 evenzo
7) Kolom 6 en 7, verder alle volgende tweetallen kolommen tot en met 34 en 35, beschouwen als telkens één superkolom voor het aangeven voor één symbool of hoofdletterwoord
8) Met één streepje (in het desbetreffende hokje) worden aangegeven:

!	0	1	2	3	4	5	6	7	8	9
?	BESTAND	DAN	ANDERS	LEES	.	:=	+	-	×	/

9) Een symbool dat linksonder in het hokje staat wordt aangeduid door een streep in dat hokje plus een streep in het hokje links boven in de superkolom
10) Analoog voor een symbool rechtsonder. Met SY wordt zowel SYM als LEESSYM bedoeld
11) Is één kaart onvoldoende om een opdracht te bevatten dan mag worden doorgeschreven op de volgende(n), mits deze wordt(worden) gemarkeerd door een streep in het bovenste hokje van kolom 3
12) Hoofdletters geeft men aan als de overeenkomstige kleine letter plus een streep in hokje met de omhooggerichte pijl (in totaal drie streepjes in die superkolom)
13) De kaarten mogen na beschrijving niet van volgorde verwisseld worden
14) Aantekeningen van de gebruikers op de achterzijde van de kaart.

Hieronder volgen een aantal voorbeelden van het gebruik van de schrapkaart.



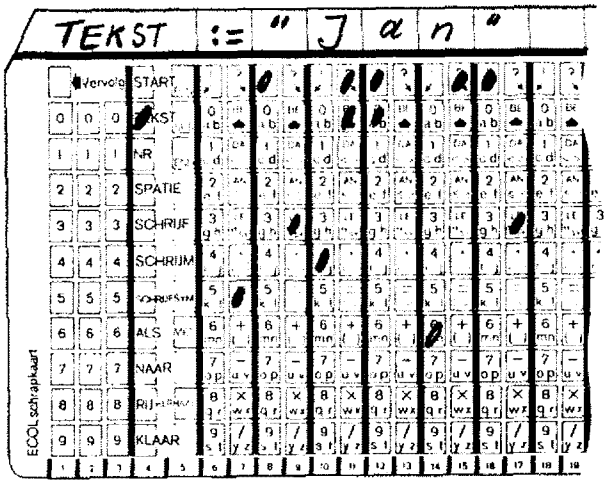
Het wordt-teken staat midden in een hokje en wordt daarom met één streepje aangegeven.

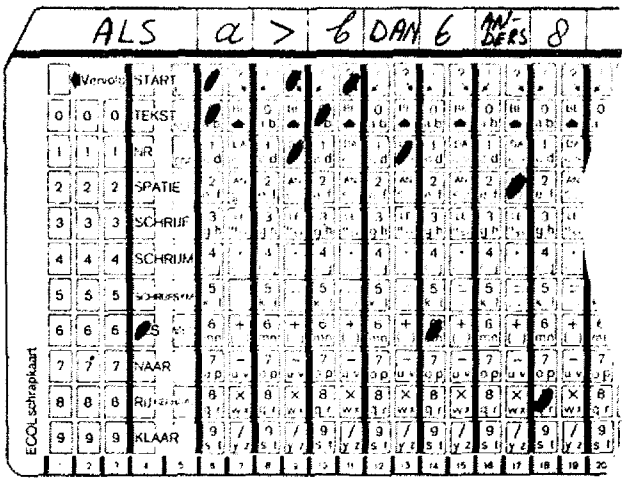
Elke letter wordt er onder met twee streepjes aangegeven.

Hieronder is de opdracht x := 8.3/m op de kaart aangestreept. De punt kunt U vinden in het hokje naast de 4.

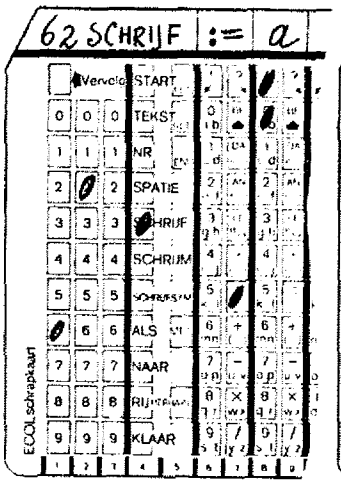
De opdracht TEKST := "Jan"
Let vooral op de aanhalingstekens. Die zijn te vinden linksonder (dus twee streepjes) in het hokje naast de 3.

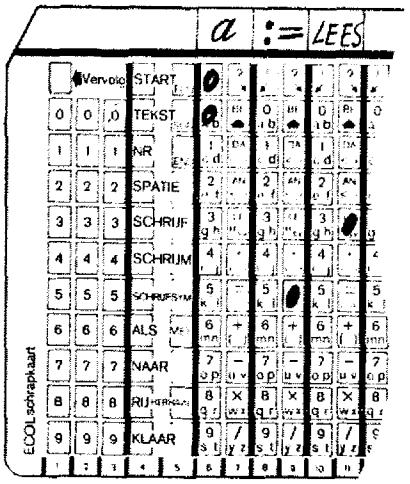
Het is ook mogelijk hoofdletters op de schrapkaart aan te strepen. U doet hetzelfde als met een kleine letter, dat is dus twee streepjes. Maar bovendien moet er een streepje in het hokje met de naar boven gerichte pijl. Dus in het totaal drie streepjes. Hieronder ziet U de hoofdletter J.

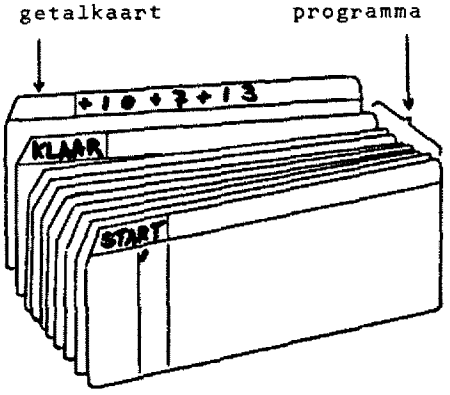




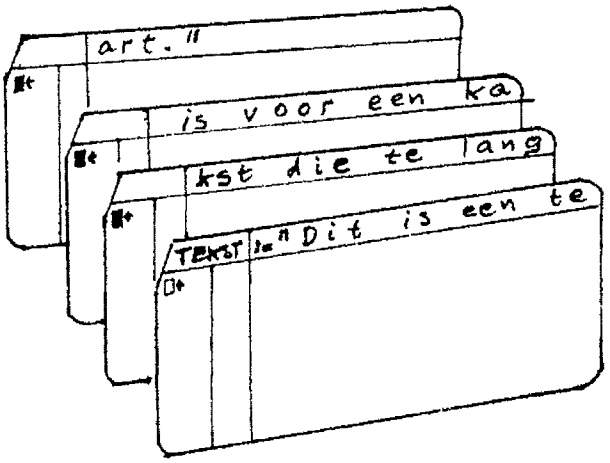
We zien hoe de opdracht ALS .. DAN .. ANDERS .. op de schrapkaart moet worden aangestreept. Elk van deze drie woorden kan worden aangegeven met één streepje, voor DAN in het hokje met DA en voor ANDERS in het hokje met AN.
Bovendien wordt getoond hoe een regelnummer voor de opdracht wordt aangegeven.
Op de schrapkaart kan LEES worden aangegeven door één streepje in het hokje met LE.
Vergeet U bij deze opdracht niet de getalkaart(en) achter het programma.



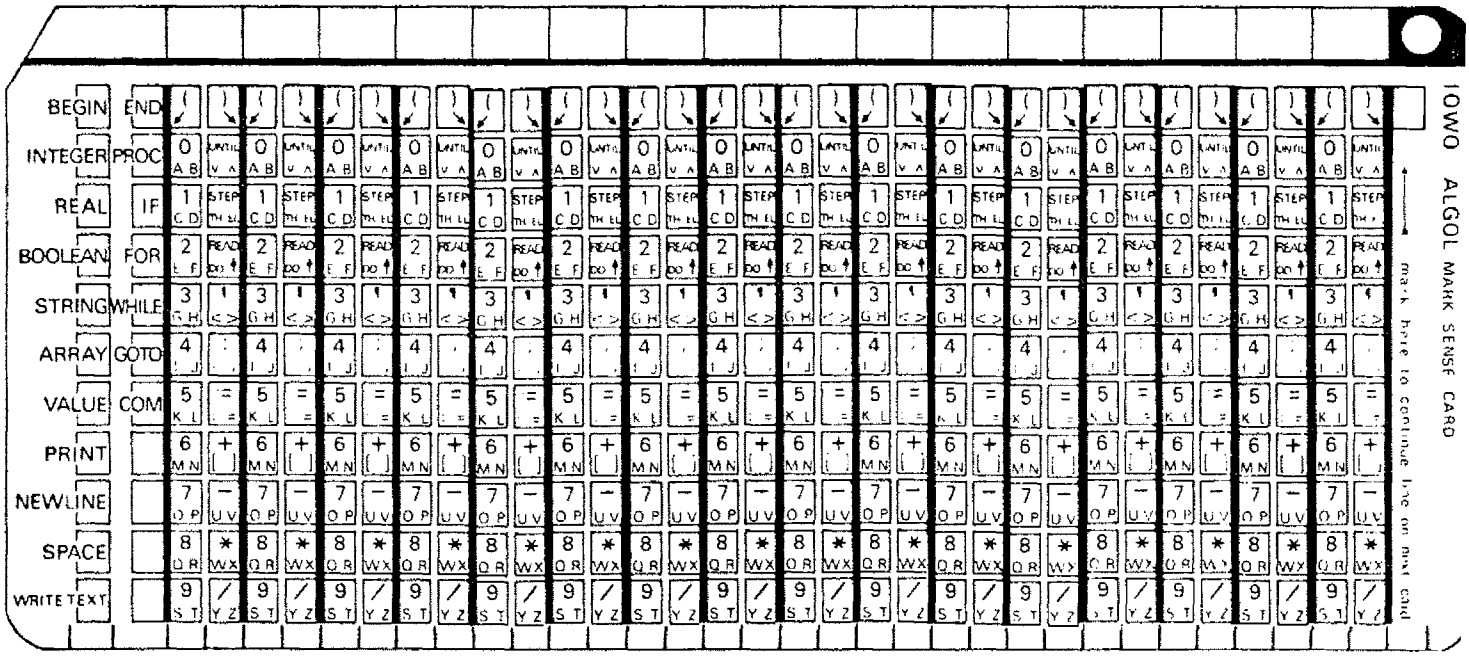




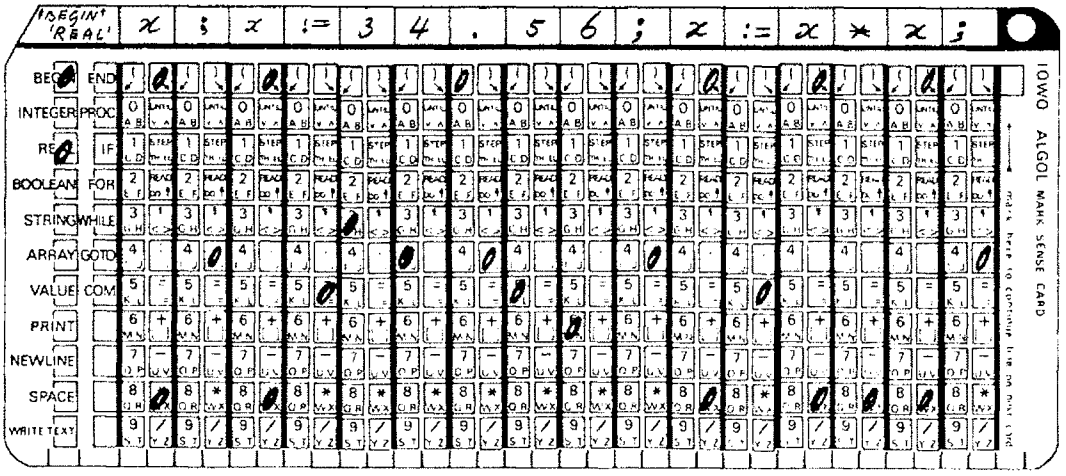
Wanneer een opdracht, b.v. een TEKST-opdracht te lang is voor één kaart, kan zo'n kaart "verlengd" worden door de volgende kaart te merken in het hokje "vervolg". Ook bij getalkaarten kan zich dat voordoen.
Men moet rekening houden met de beperking dat sommige machines niet meer dan 80 symbolen op een regel accepteren. Eén TEKST-opdracht zal in het algemeen daarom niet meer dan 4 kaarten mogen beslaan.



6.2 DE ALGOL-SCHRAPKAART

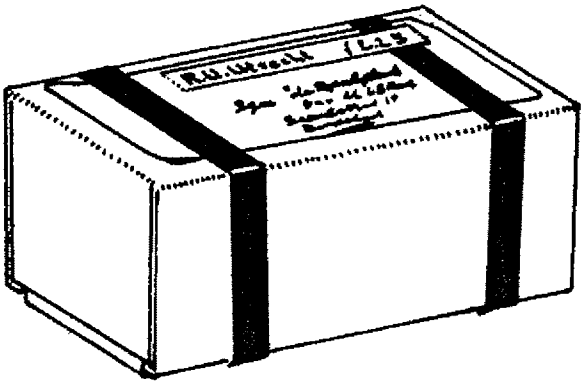


Voor de Algolgebruikers heeft het I.O.W.O. een schrapkaart ontworpen die op soortgelijke wijze als de Ecol-schrapkaart kan worden gebruikt. Voor gebruikers van de Algol-schrapkaart is een brochure verkrijgbaar waarin een beknopte toelichting op het gebruik van deze kaart wordt gegeven.



6.3 HET VERZENDEN VAN SCHRAPKAARTEN

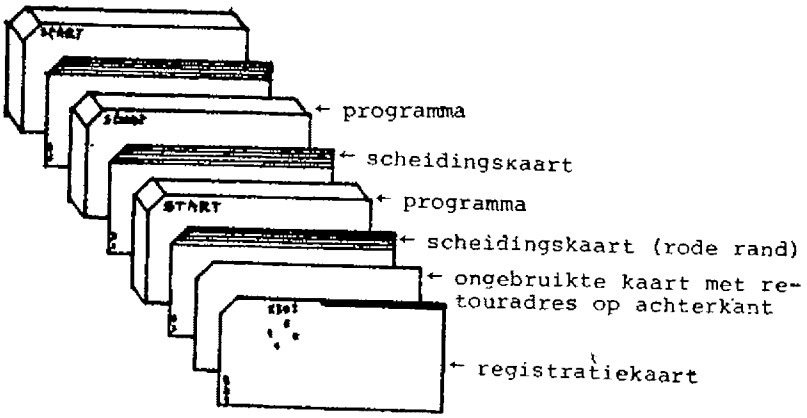
Voor het insturen van schrapkaarten voor de programmaverwerking stelt het I.O.W.O. standaardverpakking ter beschikking.



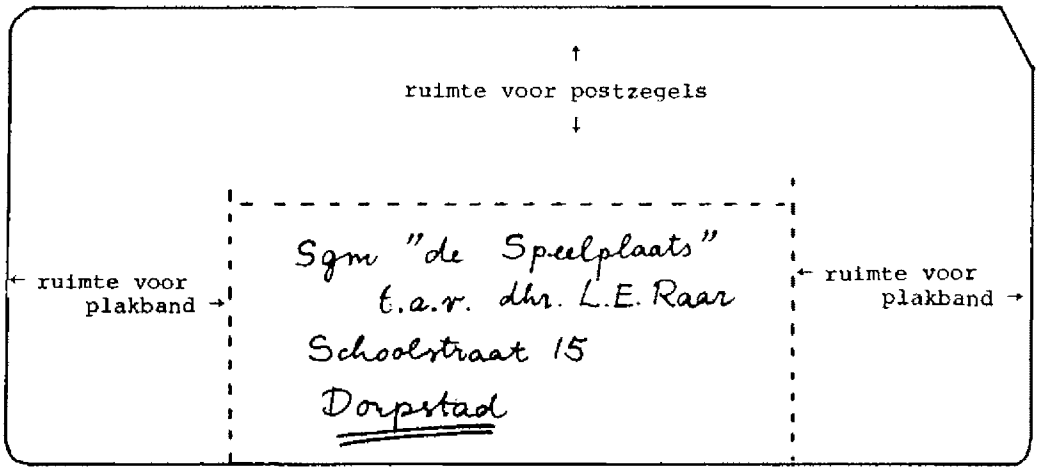
Standaardverpakking bij retourzending

In elk op te sturen doosje moeten de voorste twee kaarten zorg dragen voor een correcte retourzending:

1. De registratiekaart, een met een nummer geponste kaart, met een gedeeltelijk zwarte bovenrand. Elke docent heeft een eigen viercijferig nummer. Deze kaart zorgt ervoor, dat de papieren die uit de regeldrukker komen (de "output") automatisch van een retouradres worden voorzien.
2. Een gewone schrapkaart, waarvan aan de voorkant niets geschreven mag zijn. Op de achterkant vermeldt U het adres waar U de schrapkaarten terug wenst te ontvangen (zie figuur). Na de verwerking wordt deze kaart gebruikt als etiket voor het verzenddoosje.



Behalve door de twee genoemde kaarten moet elk programma afzonderlijk worden voorafgegaan door een scheidingskaart. Dit is een kaart met een rode bovenrand.



6.4 KOSTEN PROGRAMMAVERWERKING

Bij het volgende wordt een schatting gegeven van de officiële kosten per programma. Let wel: niet het bedrag dat de school volgend jaar in rekening wordt gebracht, maar een berekening van de totale kosten, waarvan het Rijk een groot gedeelte voor haar rekening kan nemen. De genoemde bedragen blijven voor verantwoording van de schrijver.

We gaan uit van 10 programma's per leerling per jaar. Dit is een gemiddelde. Sommige leerlingen schrijven meer. Werken in groepsverband drukt het gemiddelde weer. Een aantal programma's wordt door schrapfouten meermalen aangeboden enz. Behalve bij punt 7 nemen we 50 leerlingen per school, die computerkunde volgen.

Kosten per programma

1. Programma's worden centraal verponst (zoals in 1973 nog voor scholen gebeurt bij het CRI te Leiden). Voordelen:
   a) eenvoudigste werkwijze voor de leerlingen
   b) geen investeringskosten voor de school
   Nadelen:
   d) 'turn around'-tijd vaak meer dan 1 week
   e) leerlingen hebben geen enkel kontakt met apparatuur
   f) onleesbaar handschrift geeft fouten
   g) lange programma's of getalbanden zijn minder welkom
   h) niet herstelde ponsfouten
   i) toepassing op grote schaal onmogelijk door personeelstekort
   j) verbetering van een programmafout kan alleen door programma te herhalen.

2 sec computertijd	(á *f* 600,-/uur)	*f* 0,30
10 min ponswerk	(à *f* 12,-/uur)	" 2,--
porti heen en terug		" 0,10
		f 2,40

2. Schrapkaart (zie ook het artikel hierover)
   Voordelen:
   a) lengte en inhoud van het programma en lengte van de getalband zijn vrij
   b) programma is eenvoudig te verbeteren (kaarten vervangen)
   c) verkeerd schrijven van hoofdletterwoorden komt niet voor (staan kant en klaar op de kaart)
   d) geen investeringskosten voor de school
   e) 'turn around'-tijd minder dan 1 week
   Nadelen:
   f) aanstrepen programma kost 5 à 10 min
   g) kaarten mogen niet door elkaar raken
   h) zwaar in het versturen
   i) geen kontakt met apparatuur

2 sec computertijd	*f* 0,30
30 kaarten	" 0,30
porti	" 0,20
	f 0,80

3. Telexverbinding met computercentrum (1 toestel per 50 leerlingen)
   Voordelen:
   a) geen andere fouten dan door de leerling zelf gemaakt
   b) leerling heeft enig kontakt met apparatuur, zij het nauwelijks verband houdend met computers
   c) 'turn around'-tijd van 1 dag
   Nadelen:
   d) investering van *f* 2500,- per toestel per jaar
   e) leerling moet ervaring krijgen in machineschrijven
   f) intypen van een programma kost 20 min
   g) intypen moet in de meeste gevallen buiten de lesuren

2 sec computertijd	*f* 0,30
huur en aanleg telextoestel (*f* 2500,-/500)	" 5,--
huur en aanleg ontvangtoestel op rekencentrum (*f* 2500,-/5000)	" 0,50
telefoonkosten	" 2,40
	f 8,20

4. Ponsapparaat op school, ponsband opsturen.
   Voordelen:
   a) geen andere fouten dan door de leerling zelf gemaakt
   b) zie 3.b
   c) 'turn around'-tijd van 4 dagen
   Nadelen:
   d) investering van *f* 8000,- per toestel
   zie verder 3.e, 3.f en 3.g.

2 sec computertijd		ƒ 0,30
afschrijving, renteverlies, onderhoud	(ƒ 1600,-/500)	" 3,20
porti		" 0,05
		ƒ 3,55

Op dit bedrag kan bezuinigd worden in het geval dat het apparaat bediend wordt door (niet) onderwijzend personeel en bovendien het aantal leerlingen, dat met dit ene apparaat bediend wordt groot is. Bij 500 leerlingen zakt de prijs naar ƒ 2,40 per programma.

. De Terminal (in directe verbinding met de computer = on line)
Voordelen:
a) fouten worden onmiddellijk gecorrigeerd
b) leerling heeft kontakt met apparatuur
c) onmiddellijke verwerking
Nadelen: zie 3e, 3f en 3g.

2 sec computertijd		ƒ 0,30
huur en aanleg terminal	(ƒ 3500,-/500)	" 7,--
telefoonkosten		" 4,80
		ƒ 12,10

Op de telefoonkosten kan worden bespaard als in ieder telefoondistrict een time-sharing systeem beschikbaar is.

. De Terminal (off line)
Voordelen:
a) geen andere fouten dan door de leerling zelf gemaakt
b) leerling heeft kontakt met apparatuur
c) 'turn around'-tijd van 1 dag
Nadelen: zie 3e, 3f en 3g.

2 sec computertijd		ƒ 0,30
huur en aanleg terminal	(ƒ 3500,-/500)	" 7,--
telefoonkosten		" 1,--
		ƒ 8,30

. Computer op school (van minstens 100 leerlingen computerkunde)
Voordelen:
a) direct kontakt met de computer
b) zie verder bij 5 en 6

Nadelen: zie 3e, 3f, 3g en hoge investering van ƒ 150.000,-.

ponskosten of terminalgebruik bij 1000 programma's/jaar	ƒ 1,60
renteverlies, afschrijving en onderhoud computer (ƒ 15.000,-/1000)	" 15,--
	ƒ 16,60

Het gebruik van formulierenlezers wordt nog onderzocht. Zo'n lezer werkt in principe hetzelfde als de optische lezer voor schrapkaarten. De formulieren (velletjes papier iets groter dan de schrapkaart) zijn lichter dan de schrapkaart en daarom goedkoper en gemakkelijker te versturen.
Het nadeel is dat de lezer iets beperkter is dan de schrapkaartlezer. Bij de schrapkaart kan gebruik gemaakt worden van 12 hokjes per kolom, waarvan er overigens maar 11 in gebruik zijn vanwege de schrijfruimte die bovenaan de kaart is vrijgehouden.
Bij een formulier zijn er 10 hokjes per kolom. Hierdoor moet worden overgegaan op een andere codering, wat op zichzelf niet zo'n bezwaar is, als er maar een aanvaardbare indeling kan worden gevonden.
De kosten zijn nog onbekend, maar liggen waarschijnlijk lager dan die van een schrapkaartenprogramma.

N.B. De voorgaande cijfers zijn gebaseerd op de de situatie in 1971. Sommige prijzen van apparatuur zijn inmiddels verlaagd, arbeidsloon en posttarieven zijn niet onaanzienlijk gestegen.

6.5 LIJST VAN BESTANDEN

BESTAND-NUMMER	*OPGAVE UIT COMPK*	*AANTAL GP'S*	*OMSCHRIJVING*
10	13.7 17.2	1000	333 overschrijvingen: nraf bedr nrbij, afgesloten door -1
11	§13	1000	1000 saldi
12 t/m 14		1000	onbezet
15	personeels-administratie	200	geb, laatste twee cijfers van de geboortejaren van 200 werknemers
16		200	bs, burgelijke staat van 200 werknemers: -2, -1, +1, +2
17		200	ak, aantallen kinderen van 200 werknemers
18		200	jid, laatste twee cijfers van jaar van indiensttreding
19		200	sal, salarissen van 200 werknemers
20		150	belastingtabel van Compk II bladzijde 27
21	bevolkingsregister	300	geb, geboortedatum gecodeerd
22		300	overl, datum van overlijden gecodeerd
23		300	mv, man (1) of vrouw (-1)
24		300	va, nummer van de vader
25		300	egen, nummer van de echtgenoot of echtgenote
26		300	oudk, nummer van het oudste kind
27		300	brzus, nummer van broer of zuster die in leeftijd volgt
28 t/m 30		1000	onbezet
31 t/m 40	§15	20	elk 20 getallen ≤ 100: aantal vrije vliegtuigplaatsen
41 t/m 50	§15	40	plaatsbezetting genoteerd in enen en nullen
51	blz. 44 en 17.4	300	100 woorden van drie letters, niet alfabetisch N.B. In Compk II, blz. 44 is dit aangegeven met BESTAND(80)
52	16.5	1000	tekst
53	16.7	1000	100 woorden van 10 of minder letters
54	16.8	1000	tekst in geheimschrift, verschuiving over n, n is eerste getal
55	16.11	1000	tekst zonder overgangen op nieuwe regel
56	16.17	1000	Tom Poes verhaal met Professor Prwytzkofsky
57	16.18	1000	tekst met opmerkingen tussen haakjes
58	16.20	500	geldbedragen in oude Engelse notatie
59 t/m 63	§16	500	teksten in ECOL
64	17.1	251	getallen, voorafgegaan door hun aantal
65	17.5	500	50 woorden van 20 of minder letters gescheiden door spaties
66	17.7	63	63 telefoonnummers in opklimmende volgorde
67	17.7	1000	63 bijbehorende abonnees
68	17.8	630	63 woorden ≤ 10 letters in alfabetische volgorde

Op de volgende bladzijden vindt U de gehele of gedeeltelijke inhoud van een groot aantal bestanden.

BESTAND 10

2	0.51	24
908	106.12	754
361	298.60	455
817	488.98	607
293	23.92	648
28	527.95	725
609	16.36	534
310	107.13	159
968	26.04	668
56	41.68	554
749	63.31	81
743	18.92	354
431	145.01	716
593	632.54	660
25	8.60	282
55	193.67	648
815	127.56	990
546	20.36	198
478	101.10	623
598	27.73	146
311	18.30	208
266	59.75	316
155	145.87	216
615	98.99	236
246	47.83	305
436	43.07	183
58	399.74	376
562	204.93	989
907	42.04	856
718	45.71	743
847	143.45	951
146	56.84	296
131	54.21	473
140	8.48	240
296	251.54	903
927	114.68	754
615	65.38	350
233	30.27	477
493	27.94	28
33	56.76	314
19	161.96	403
930	22.34	834
795	33.39	540
315	158.80	666
760	79.54	457
261	36.55	38
534	76.35	402
397	479.54	384
624	21.58	551
442	93.26	667
460	140.74	396
425	11.11	812
76	141.12	864
622	69.95	429
956	89.80	957
595	19.49	710
338	52.68	420
167	150.61	171
984	56.96	749
415	25.90	638
226	10.04	543
606	72.41	641
792	73.57	995
718	30.17	105
720	29.65	60
957	31.69	14
365	36.36	553
431	80.85	449
297	171.60	252
873	69.32	147
625	6.21	742
90	24.31	490
5	56.90	564
526	24.95	600
392	135.28	929
104	358.60	910
295	297.97	50
252	287.20	402
80	9.78	845
770	24.91	399
596	16.96	578
941	182.25	566
154	0.94	674
45	39.00	544
646	145.74	798
173	17.35	351
332	15.39	872
52	16.82	471
570	105.77	790
135	246.19	283
537	144.44	754
353	231.08	235
703	39.21	620
203	155.99	916
602	205.70	815
962	10.78	957
176	110.79	439
395	38.35	358
723	80.35	810
112	306.25	724
237	276.16	492
480	36.42	515
659	88.18	585
764	189.32	219
334	345.06	807
872	68.54	131
691	37.21	652
886	20.05	117
941	115.82	650
277	70.07	537
318	130.51	518
455	48.79	224
137	174.39	726
75	364.38	176
716	201.87	763
230	27.34	371
931	251.36	135
468	1.21	862
939	48.82	869
265	51.95	51
348	76.12	72
369	191.90	802
871	57.96	797
177	26.51	809
246	138.14	284
40	153.44	355
932	153.39	317
160	162.26	387
903	229.85	270
483	37.97	553
528	14.82	83
748	25.22	611
344	80.66	234
585	92.59	364
254	155.90	462
338	19.55	32
412	138.21	790
3	12.93	707
850	61.65	117
447	128.16	317
606	64.94	416
807	86.99	222
904	52.71	326
734	40.38	390
653	54.54	652
874	11.28	777
302	159.71	70
763	22.02	325
836	266.66	60
22	106.40	744
434	34.56	852
518	147.85	979
78	31.30	920
902	116.32	8
145	266.44	285
669	83.97	393
760	9.15	690
647	9.73	733
439	25.62	408
590	22.99	923
310	18.01	206
252	44.25	881
934	75.53	775
123	122.55	140
523	49.34	832
718	456.23	474
63	102.19	279
90	4.50	459
645	524.56	168
947	20.90	612
26	107.54	728
566	278.29	542
196	4.40	503
376	146.45	238
495	11.02	176
888	232.92	430
550	17.48	13
371	40.12	651
69	31.35	1000
425	78.12	436
268	73.47	719
363	114.54	778
472	176.79	729
91	10.43	788
166	8.24	991
765	71.60	182
512	121.84	622
610	28.81	18
152	6.11	996
558	36.89	833
778	18.15	943
537	16.76	99
800	115.27	904
734	77.03	623
100	22.08	295
17	137.75	342
682	17.80	843
413	5.79	627
744	11.21	942
303	34.61	33
439	258.15	598
737	157.15	127
374	47.04	892
25	13.96	823
615	10.63	67
511	108.16	368
747	152.23	973
193	283.08	872
236	30.63	464
595	134.33	81
172	138.94	960
996	35.80	736
131	52.31	271
43	116.05	749
711	338.87	406
264	25.14	10
18	20.17	943
989	54.13	606
131	26.15	216
781	97.60	714
327	48.25	358
302	2.58	438
607	85.09	980
280	67.96	443
782	112.86	23
959	296.30	60
180	94.31	48
211	117.37	249
726	89.92	24
193	50.67	604
953	16.98	359
258	145.66	284
201	141.28	735
403	2.64	531
51	8.15	18
603	146.88	193
773	46.18	264
747	190.53	392
316	52.86	626
944	34.42	252
115	208.79	662
580	55.59	346
768	124.19	358
259	144.41	258
333	37.13	870
573	30.51	428
805	111.41	738
321	0.68	174
41	160.05	430
606	69.78	561
156	379.30	473
963	3.09	514
193	150.02	930
412	181.66	327
581	53.75	274
102	6.79	489
661	44.70	215
960	115.52	473
331	26.93	440
488	211.30	892
563	58.82	331
607	93.46	187
348	142.40	433
237	37.24	744
337	123.81	236
977	411.94	108
213	27.00	508
85	7.60	681
574	84.11	252
613	270.75	91
858	329.83	35
410	159.41	97
596	57.34	263
355	66.54	727
14	36.51	597
994	39.89	31
779	294.99	673
492	0.46	605
412	246.12	732
541	32.70	812
640	15.60	110
646	194.55	329
743	53.48	798
939	169.24	449
655	322.71	871
417	41.26	282
356	196.80	741
281	125.70	766
844	309.67	139
767	183.80	144
710	78.43	878
627	106.88	303

Het restant van bestand 10 staat op bladzijde 110.

97		35.34		919													
170		32.73		156													
582		215.51		74													
519		146.20		944													
252		0.37		757													
491		142.32		137													
17		38.42		771													
250		55.62		320													
924		53.85		135													
640		69.52		247													
31		50.56		118													
875		35.34		939													
83		235.44		923													
551		1.10		507													
64		286.63		35													
983		106.39		87													
378		3.92		836													
331		70.36		61													
188		108.96		295													
206		215.59		457													
223		58.01		637													
142		65.89		198													
52		0.95		600													
488		39.05		549													
620		105.50		337													
843		12.93		85													
589		73.37		409													
102		35.76		807													
736		113.41		449													
412		71.27		353													
472		0.10		768													
411		45.71		512													
236		0.95		945													
417		0.89		292													
313		59.26		186													
387		405.18		416													
354		193.30		955													
972		59.31		699													
617		158.39		212													
874		19.10		140													
730		96.81		221													
-1																	

Hierboven staat het restant van bestand 10.

In bestand 11 hieronder zijn de rekeningnummers en de streepjes in werkelijkheid niet opgenomen. Ze staan hier uitsluitend voor de duidelijkheid.

BESTAND 11

1	-	305.80	2	-	48.07	3	-	133.51	4	-	215.45	5	-	354.30
6	-	78.64	7	-	142.99	8	-	314.83	9	-	350.13	10	-	5.72
11	-	349.50	12	-	362.70	13	-	273.58	14	-	30.60	15	-	10.36
16	-	771.91	17	-	49.59	18	-	444.18	19	-	1518.28	20	-	26.83
21	-	235.43	22	-	1027.33	23	-	107.62	24	-	198.77	25	-	143.50
26	-	133.22	27	-	166.97	28	-	71.42	29	-	655.45	30	-	438.66
31	-	154.66	32	-	331.35	33	-	135.25	34	-	490.38	35	-	703.20
36	-	120.55	37	-	631.06	38	-	420.59	39	-	256.62	40	-	435.33
41	-	79.14	42	-	379.35	43	-	280.57	44	-	680.00	45	-	50.30
46	-	2.04	47	-	253.26	48	-	219.16	49	-	426.94	50	-	455.08
51	-	199.03	52	-	20.12	53	-	84.88	54	-	832.35	55	-	51.06
56	-	292.18	57	-	583.55	58	-	88.78	59	-	415.51	60	-	65.58
61	-	23.25	62	-	461.35	63	-	288.55	64	-	77.16	65	-	24.71
66	-	198.84	67	-	143.80	68	-	131.00	69	-	179.29	70	-	37.09
71	-	87.64	72	-	170.50	73	-	38.00	74	-	145.53	75	-	292.56
76	-	487.89	77	-	255.98	78	-	118.37	79	-	631.08	80	-	479.57
81	-	111.87	82	-	911.39	83	-	199.78	84	-	216.57	85	-	232.41
86	-	44.39	87	-	116.74	88	-	528.00	89	-	42.12	90	-	431.14
91	-	84.96	92	-	322.47	93	-	528.63	94	-	32.30	95	-	809.36
96	-	199.29	97	-	355.17	98	-	189.44	99	-	873.82	100	-	35.74
101	-	506.23	102	-	31.29	103	-	21.99	104	-	346.18	105	-	88.86
106	-	107.17	107	-	663.43	108	-	922.11	109	-	91.71	110	-	542.28
111	-	285.46	112	-	473.01	113	-	376.89	114	-	251.87	115	-	50.08
116	-	1003.06	117	-	528.47	118	-	128.52	119	-	182.23	120	-	42.60
121	-	34.40	122	-	51.78	123	-	23.03	124	-	41.78	125	-	242.87
126	-	208.13	127	-	1.08	128	-	193.68	129	-	36.21	130	-	11.30
131	-	421.87	132	-	44.44	133	-	143.21	134	-	198.15	135	-	344.83
136	-	239.68	137	-	239.70	138	-	412.22	139	-	250.74	140	-	633.13
141	-	297.31	142	-	114.73	143	-	170.72	144	-	230.49	145	-	629.61
146	-	722.24	147	-	309.67	148	-	885.09	149	-	220.12	150	-	129.97
151	-	71.27	152	-	251.07	153	-	356.52	154	-	388.14	155	-	147.18
156	-	45.81	157	-	324.29	158	-	429.49	159	-	200.19	160	-	77.97
161	-	84.17	162	-	191.72	163	-	1804.24	164	-	974.12	165	-	239.48
166	-	15.59	167	-	412.44	168	-	19.05	169	-	711.99	170	-	1.57
171	-	493.82	172	-	34.89	173	-	23.88	174	-	241.43	175	-	1561.55
176	-	17.80	177	-	365.00	178	-	89.88	179	-	35.83	180	-	1554.70
181	-	6.18	182	-	790.27	183	-	85.18	184	-	53.34	185	-	480.01
186	-	410.46	187	-	126.66	188	-	76.37	189	-	240.83	190	-	328.67
191	-	11.87	192	-	449.21	193	-	1490.54	194	-	17.02	195	-	394.59
196	-	36.28	197	-	395.20	198	-	261.68	199	-	752.89	200	-	86.25
201	-	79.36	202	-	132.27	203	-	439.42	204	-	58.78	205	-	3.85
206	-	116.70	207	-	909.79	208	-	147.71	209	-	31.85	210	-	679.49
211	-	378.06	212	-	303.23	213	-	399.51	214	-	588.97	215	-	74.04
216	-	907.47	217	-	117.05	218	-	207.18	219	-	31.65	220	-	724.69
221	-	302.15	222	-	1606.53	223	-	113.75	224	-	48.45	225	-	92.41
226	-	20.83	227	-	71.29	228	-	40.84	229	-	121.32	230	-	587.05
231	-	577.68	232	-	47.51	233	-	170.36	234	-	25.71	235	-	280.73
236	-	285.94	237	-	556.88	238	-	156.96	239	-	981.77	240	-	247.06
241	-	1100.61	242	-	32.62	243	-	283.62	244	-	155.24	245	-	22.71
246	-	784.98	247	-	457.34	248	-	996.24	249	-	58.21	250	-	93.15
251	-	580.08	252	-	393.53	253	-	113.71	254	-	227.27	255	-	1088.34
256	-	142.77	257	-	1024.52	258	-	467.02	259	-	246.85	260	-	228.12

261	-	425.10	262	-	325.42	263	-	70.67	264	-	457.76	265	-	161.16
266	-	186.66	267	-	196.19	268	-	902.72	269	-	1116.06	270	-	267.05
271	-	343.33	272	-	279.21	273	-	78.92	274	-	52.56	275	-	924.15
276	-	530.42	277	-	172.56	278	-	727.56	279	-	270.51	280	-	78.97
281	-	92.08	282	-	120.71	283	-	301.38	284	-	72.97	285	-	0.11
286	-	16.87	287	-	331.37	288	-	205.65	289	-	1.60	290	-	209.59
291	-	823.49	292	-	591.64	293	-	159.93	294	-	445.09	295	-	308.15
296	-	63.00	297	-	26.88	298	-	455.76	299	-	234.29	300	-	379.51
301	-	506.34	302	-	314.62	303	-	436.48	304	-	242.15	305	-	450.75
306	-	129.45	307	-	692.00	308	-	175.79	309	-	80.66	310	-	270.88
311	-	111.54	312	-	461.67	313	-	203.83	314	-	1083.34	315	-	386.48
316	-	683.89	317	-	30.16	318	-	127.82	319	-	17.75	320	-	228.31
321	-	101.03	322	-	201.42	323	-	203.52	324	-	309.43	325	-	367.07
326	-	1088.93	327	-	493.12	328	-	152.38	329	-	129.13	330	-	114.38
331	-	898.64	332	-	177.74	333	-	438.18	334	-	138.26	335	-	619.78
336	-	172.27	337	-	213.63	338	-	920.80	339	-	280.00	340	-	450.84
341	-	621.90	342	-	424.94	343	-	265.80	344	-	320.96	345	-	439.62
346	-	192.36	347	-	81.85	348	-	110.12	349	-	2.29	350	-	62.94
351	-	477.08	352	-	220.15	353	-	376.82	354	-	8.90	355	-	147.07
356	-	159.19	357	-	12.76	358	-	13.54	359	-	134.08	360	-	174.14
361	-	36.34	362	-	130.23	363	-	590.13	364	-	332.33	365	-	82.53
366	-	226.23	367	-	321.88	368	-	101.03	369	-	135.10	370	-	86.79
371	-	78.17	372	-	135.19	373	-	426.85	374	-	59.99	375	-	25.64
376	-	629.64	377	-	1913.89	378	-	36.18	379	-	121.77	380	-	1116.90
381	-	46.10	382	-	135.23	383	-	233.55	384	-	201.26	385	-	66.74
386	-	155.59	387	-	449.99	388	-	358.10	389	-	526.23	390	-	436.23
391	-	960.13	392	-	127.84	393	-	100.61	394	-	55.12	395	-	167.97
396	-	746.82	397	-	305.78	398	-	728.46	399	-	450.06	400	-	42.56
401	-	72.29	402	-	223.15	403	-	775.09	404	-	393.39	405	-	105.43
406	-	312.61	407	-	179.05	408	-	195.01	409	-	27.38	410	-	57.72
411	-	54.11	412	-	171.13	413	-	710.67	414	-	314.12	415	-	887.16
416	-	190.73	417	-	310.59	418	-	680.59	419	-	509.20	420	-	9.17
421	-	189.30	422	-	763.44	423	-	90.13	424	-	48.52	425	-	438.92
426	-	13.90	427	-	501.44	428	-	382.19	429	-	138.63	430	-	105.81
431	-	652.29	432	-	2.17	433	-	1173.31	434	-	17.07	435	-	565.92
436	-	159.31	437	-	669.17	438	-	828.37	439	-	127.62	440	-	201.96
441	-	854.13	442	-	838.12	443	-	660.19	444	-	72.67	445	-	367.05
446	-	529.83	447	-	336.05	448	-	205.79	449	-	949.05	450	-	462.72
451	-	326.91	452	-	857.58	453	-	139.04	454	-	103.06	455	-	748.25
456	-	632.79	457	-	191.05	458	-	92.67	459	-	54.55	460	-	283.00
461	-	548.79	462	-	188.80	463	-	308.54	464	-	854.90	465	-	250.68
466	-	39.79	467	-	51.33	468	-	836.36	469	-	922.56	470	-	533.44
471	-	168.29	472	-	867.38	473	-	189.83	474	-	347.74	475	-	325.93
476	-	565.81	477	-	880.34	478	-	345.62	479	-	140.69	480	-	273.21
481	-	13.58	482	-	131.40	483	-	192.55	484	-	1012.67	485	-	854.58
486	-	319.56	487	-	266.95	488	-	250.61	489	-	106.50	490	-	35.12
491	-	312.99	492	-	38.17	493	-	181.37	494	-	67.90	495	-	362.74
496	-	1030.24	497	-	635.07	498	-	3.83	499	-	2.24	500	-	551.87
501	-	652.64	502	-	35.40	503	-	131.20	504	-	618.95	505	-	256.10
506	-	359.77	507	-	307.51	508	-	601.98	509	-	10.93	510	-	1609.58
511	-	367.30	512	-	491.68	513	-	499.34	514	-	644.78	515	-	154.78
516	-	298.92	517	-	265.81	518	-	477.70	519	-	803.31	520	-	83.67
521	-	51.64	522	-	591.37	523	-	159.61	524	-	451.45	525	-	289.42
526	-	100.63	527	-	381.76	528	-	36.94	529	-	524.38	530	-	0.61
531	-	242.97	532	-	316.02	533	-	30.50	534	-	245.46	535	-	450.74
536	-	109.81	537	-	53.75	538	-	89.14	539	-	1195.60	540	-	479.95
541	-	142.86	542	-	235.70	543	-	117.95	544	-	49.39	545	-	53.50
546	-	318.08	547	-	281.00	548	-	189.83	549	-	375.60	550	-	205.33
551	-	274.66	552	-	850.38	553	-	931.39	554	-	479.45	555	-	288.06
556	-	37.44	557	-	405.77	558	-	205.31	559	-	130.95	560	-	162.15
561	-	111.95	562	-	140.24	563	-	940.60	564	-	691.10	565	-	1403.77
566	-	928.59	567	-	131.62	568	-	97.09	569	-	46.92	570	-	308.94
571	-	548.73	572	-	58.48	573	-	144.55	574	-	35.58	575	-	656.07
576	-	329.88	577	-	2.54	578	-	878.67	579	-	84.30	580	-	3.42
581	-	389.91	582	-	228.94	583	-	86.62	584	-	352.44	585	-	272.04
586	-	67.98	587	-	162.24	588	-	331.56	589	-	77.42	590	-	286.83
591	-	90.02	592	-	703.17	593	-	83.35	594	-	157.35	595	-	137.63
596	-	37.99	597	-	1104.92	598	-	25.16	599	-	408.37	600	-	504.72
601	-	164.80	602	-	44.35	603	-	139.34	604	-	229.03	605	-	180.71
606	-	195.92	607	-	15.46	608	-	144.68	609	-	87.18	610	-	13.31
611	-	190.46	612	-	467.28	613	-	10.35	614	-	353.62	615	-	233.47
616	-	222.85	617	-	1313.79	618	-	132.51	619	-	1056.44	620	-	995.53
621	-	198.07	622	-	166.70	623	-	29.20	624	-	279.89	625	-	223.65
626	-	147.04	627	-	873.63	628	-	547.88	629	-	182.27	630	-	378.83
631	-	274.30	632	-	901.93	633	-	702.70	634	-	6.62	635	-	397.23
636	-	153.15	637	-	1623.38	638	-	167.88	639	-	189.81	640	-	120.11
641	-	484.12	642	-	76.65	643	-	173.63	644	-	122.04	645	-	12.22
646	-	418.89	647	-	40.39	648	-	191.52	649	-	10.76	650	-	234.19

651	-	169.49	652	-	228.68	653	-	739.86	654	-	370.27	655	-	214.32
656	-	243.91	657	-	20.65	658	-	269.39	659	-	560.93	660	-	268.58
661	-	546.79	662	-	311.59	663	-	258.87	664	-	378.82	665	-	204.82
666	-	258.38	667	-	25.98	668	-	109.56	669	-	162.24	670	-	404.25
671	-	482.53	672	-	233.08	673	-	246.79	674	-	446.17	675	-	111.08
676	-	58.52	677	-	49.32	678	-	205.26	679	-	325.90	680	-	249.88
681	-	311.32	682	-	11.77	683	-	947.42	684	-	718.72	685	-	794.38
686	-	84.21	687	-	55.44	688	-	461.03	689	-	443.27	690	-	109.08
691	-	76.39	692	-	714.96	693	-	226.70	694	-	313.64	695	-	123.04
696	-	47.03	697	-	32.10	698	-	26.75	699	-	256.54	700	-	459.23
701	-	40.17	702	-	65.25	703	-	360.55	704	-	13.70	705	-	172.89
706	-	1054.30	707	-	80.91	708	-	568.19	709	-	447.86	710	-	363.83
711	-	493.20	712	-	541.64	713	-	387.58	714	-	41.11	715	-	365.54
716	-	291.51	717	-	805.76	718	-	506.16	719	-	385.87	720	-	117.11
721	-	238.00	722	-	491.01	723	-	81.85	724	-	115.24	725	-	830.79
726	-	211.34	727	-	226.90	728	-	92.64	729	-	298.55	730	-	526.14
731	-	136.55	732	-	135.81	733	-	223.05	734	-	257.32	735	-	520.79
736	-	622.65	737	-	124.20	738	-	30.08	739	-	141.71	740	-	642.25
741	-	114.44	742	-	50.57	743	-	72.61	744	-	113.16	745	-	59.54
746	-	28.93	747	-	528.66	748	-	67.93	749	-	119.81	750	-	67.27
751	-	534.47	752	-	67.88	753	-	107.14	754	-	141.25	755	-	13.50
756	-	132.20	757	-	188.00	758	-	2006.94	759	-	509.22	760	-	35.01
761	-	1315.92	762	-	26.11	763	-	283.92	764	-	259.24	765	-	6.58
766	-	325.92	767	-	530.34	768	-	18.91	769	-	55.87	770	-	169.06
771	-	651.26	772	-	493.83	773	-	60.82	774	-	301.71	775	-	235.11
776	-	9.87	777	-	743.99	778	-	71.22	779	-	199.54	780	-	60.77
781	-	225.03	782	-	60.88	783	-	48.33	784	-	181.62	785	-	824.89
786	-	105.49	787	-	477.99	788	-	240.41	789	-	221.16	790	-	719.49
791	-	571.85	792	-	371.10	793	-	24.82	794	-	43.84	795	-	190.27
796	-	953.62	797	-	6.98	798	-	81.08	799	-	139.64	800	-	302.84
801	-	544.22	802	-	89.38	803	-	761.53	804	-	50.55	805	-	454.99
806	-	762.13	807	-	178.29	808	-	6.44	809	-	550.23	810	-	437.62
811	-	536.85	812	-	29.95	813	-	812.68	814	-	236.02	815	-	120.79
816	-	34.74	817	-	134.39	818	-	568.16	819	-	317.74	820	-	171.23
821	-	214.80	822	-	5.45	823	-	102.92	824	-	30.18	825	-	501.55
826	-	92.76	827	-	30.03	828	-	10.59	829	-	818.53	830	-	24.97
831	-	34.50	832	-	297.70	833	-	84.92	834	-	583.90	835	-	304.82
836	-	183.34	837	-	216.81	838	-	478.72	839	-	264.17	840	-	112.24
841	-	586.06	842	-	12.76	843	-	7.92	844	-	189.78	845	-	914.91
846	-	20.19	847	-	10.94	848	-	66.70	849	-	197.14	850	-	113.89
851	-	440.59	852	-	231.99	853	-	551.92	854	-	33.81	855	-	468.54
856	-	106.71	857	-	28.93	858	-	421.49	859	-	310.63	860	-	124.09
861	-	53.39	862	-	1083.30	863	-	356.20	864	-	22.88	865	-	109.04
866	-	214.44	867	-	64.36	868	-	84.33	869	-	470.78	870	-	13.40
871	-	283.86	872	-	825.79	873	-	627.35	874	-	113.54	875	-	75.07
876	-	542.36	877	-	1435.23	878	-	224.08	879	-	750.87	880	-	422.47
881	-	68.47	882	-	1129.53	883	-	108.09	884	-	149.93	885	-	747.84
886	-	938.17	887	-	2.15	888	-	130.06	889	-	523.09	890	-	644.11
891	-	96.45	892	-	154.20	893	-	36.40	894	-	358.58	895	-	464.50
896	-	2161.56	897	-	749.86	898	-	207.60	899	-	407.12	900	-	15.29
901	-	5.60	902	-	174.86	903	-	60.74	904	-	776.58	905	-	484.81
906	-	609.86	907	-	256.79	908	-	390.43	909	-	183.01	910	-	370.54
911	-	302.14	912	-	496.36	913	-	171.01	914	-	36.18	915	-	157.18
916	-	226.89	917	-	47.38	918	-	128.30	919	-	272.87	920	-	100.39
921	-	750.06	922	-	810.27	923	-	45.77	924	-	422.32	925	-	64.95
926	-	10.48	927	-	12.93	928	-	172.89	929	-	1455.99	930	-	33.14
931	-	175.40	932	-	154.47	933	-	275.56	934	-	444.34	935	-	938.43
936	-	111.00	937	-	232.08	938	-	1255.26	939	-	83.75	940	-	337.76
941	-	420.71	942	-	150.95	943	-	553.14	944	-	148.67	945	-	24.77
946	-	2.25	947	-	83.06	948	-	197.11	949	-	814.07	950	-	19.61
951	-	73.39	952	-	41.22	953	-	97.14	954	-	61.34	955	-	153.70
956	-	798.85	957	-	498.93	958	-	434.53	959	-	36.84	960	-	163.93
961	-	170.72	962	-	280.53	963	-	179.20	964	-	539.68	965	-	871.60
966	-	510.52	967	-	265.13	968	-	38.57	969	-	1596.08	970	-	862.09
971	-	144.86	972	-	70.93	973	-	102.67	974	-	136.71	975	-	65.29
976	-	228.14	977	-	13.68	978	-	842.38	979	-	972.46	980	-	450.97
981	-	347.89	982	-	644.50	983	-	221.77	984	-	739.48	985	-	494.17
986	-	760.17	987	-	70.47	988	-	183.73	989	-	137.61	990	-	333.38
991	-	327.03	992	-	1164.46	993	-	48.54	994	-	102.44	995	-	757.26
996	-	629.18	997	-	183.67	998	-	132.95	999	-	394.22	1000	-	148.43

BESTANDEN

Persoonsnummer	15	16	17	18	en 19
1	39	2	4	58	1966.00
2	36	2	2	58	1523.00
3	11	2	1	54	3214.00
4	41-	1	0	65	1541.00
5	39	2	3	57	2146.00
	47	2	0	66	1267.00
	23-	1	0	57	2040.00
	44	2	2	60	1571.00
	47	2	3	68	1055.00
10	43	1	0	59	1537.00
	45	2	3	61	1327.00
	27-	1	0	61	2546.00
	41-	1	0	67	1694.00
	49-	2	1	66	1661.00
15	44	2	1	61	1320.00
	19	2	4	43	1811.00
	43	1	0	66	2192.00
	38	2	6	55	1451.00
	56	1	0	72	1033.00
20	36	2	3	61	1999.00
	23	2	0	45	2532.00
	37	2	3	61	2171.00
	43	1	0	68	1679.00
	9	2	1	39	2299.00
25	9	2	1	37	2765.00
	45	2	3	66	1531.00
	44	2	1	67	2030.00
	11	2	1	52	3668.00
	30-	1	0	47	1592.00
30	38	2	4	59	1513.00
	22	2	2	46	2148.00
	38	2	4	65	2211.00
	49-	2	2	67	1381.00
	40-	2	4	64	1376.00
35	53-	1	0	70	1629.00
	27-	2	1	55	2636.00
	49-	2	1	67	1442.00
	55	1	0	72	999.00
	20	2	2	43	2725.00
40	35	2	3	64	2496.00
	29-	1	0	61	2555.00
	47	2	2	55	1416.00
	43	1	0	65	1624.00
	49-	2	1	69	1123.00
45	54-	1	0	70	1032.00
	9	2	1	44	3331.00
	42-	1	0	64	1622.00
	45	2	3	64	1623.00
	16	2	3	56	3036.00
50	42-	1	0	62	1393.00
	51-	1	0	63	1363.00
	10	2	1	42	1974.00
	41-	1	[illegible]	61	1577.00
	41-	1	1	66	1763.00
55	41-	1	0	63	[illegible]
	43	2	4	61	1326.00
	33	2	3	57	1413.00
	50-	1	0	66	1277.00
	42-	1	0	62	1223.00
60	44	2	2	63	1517.00
	43	1	[illegible]	60	1166.00
	42-	1	0	65	1352.00
	12	2	0	37	2555.00
	17	2	1	53	2219.00
65	32-	1	0	54	1430.00
	43	2	4	60	1253.00
	46	2	3	64	1194.00
	49-	2	1	67	1222.00
	31-	1	0	54	1713.00
70	22	2	1	41	1927.00
	10	2	1	54	3552.00
	56	1	0	72	1096.00
	42	1	0	67	1697.00
	55	1	0	72	1114.00
75	41-	1	0	60	1945.00
	40-	2	5	62	1926.00
	36	2	3	55	2427.00
	25	2	2	61	2503.00
	40	2	2	59	1946.00
80	37	2	2	55	1270.00
	39	2	3	55	1725.00
	45	2	3	67	1137.00
	47	2	2	69	1747.00
	44	2	4	63	1524.00
85	55	1	0	71	995.00
	44	2	0	62	1604.00
	39	2	3	57	2348.00
	34	1	0	52	1476.00
	48	2	1	67	1817.00
90	11	2	0	38	2566.00
	48-	2	2	56	1534.00
	35	2	4	58	2431.00
	15	2	3	39	3174.00
	45	2	2	62	1240.00
95	54-	1	0	71	1464.00
	45	2	5	65	1332.00
	44	2	4	65	1923.00
	43	2	1	66	1392.00
	45	2	3	67	1431.00
100	40	2	4	59	2044.00
	43	1	0	63	2137.00
	32-	1	0	55	1596.00
	52-	1	0	69	953.00
	43	1	0	63	2046.00
105	11	2	1	31	2306.00
	20	2	2	37	2241.00
	39	2	3	65	2237.00
	25	2	3	50	1711.00
	46	2	6	64	1671.00
110	34	1	0	54	2137.00
	37	2	4	65	2339.00
	15	2	3	53	2649.00
	50-	1	0	66	1385.00
	51-	1	0	67	993.00
115	44	2	0	64	2107.00
	35	2	3	54	2562.00
	36	2	3	60	2299.00
	49-	2	1	63	1567.00
	36	2	2	62	1507.00
120	56	1	0	72	1247.00
	29-	1	0	57	2291.00
	27-	2	1	55	1933.00
	33	2	4	62	1362.00
	8	2	1	35	2631.00
125	39	2	2	59	1735.00
	51-	1	0	63	1043.00
	49-	2	1	66	1331.00
	37	2	3	59	1521.00
	17	2	1	51	2636.00
130	56	1	0	72	956.00
	43	1	0	61	1734.00
	24	2	2	43	1977.00
	16	2	3	43	2945.00
	38	2	6	64	1396.00
135	48	2	2	64	1269.00
	34	1	0	55	1763.00
	38	2	5	56	2375.00
	32-	1	0	54	1824.00
	36	2	2	55	1443.00

BESTAND	31	4	5	18	22	11	16	30	17	54	39	40	61	68	38	63	27	65	75	93	100
BESTAND	32	6	7	12	9	20	11	44	44	47	59	34	46	31	54	69	43	63	100	56	67
BESTAND	33	0	2	15	13	19	23	48	40	29	35	36	27	79	72	50	40	63	45	65	45
BESTAND	34	3	3	13	10	13	32	28	14	32	43	63	66	54	92	69	70	84	98	82	65
BESTAND	35	7	7	8	10	15	21	38	32	37	38	43	36	51	48	98	67	79	100	100	65
BESTAND	36	2	0	9	14	22	24	26	52	59	30	44	62	59	58	67	51	70	91	60	100
BESTAND	37	0	8	12	13	18	26	41	35	35	43	58	49	57	54	81	67	84	57	94	100
BESTAND	38	0	3	13	18	12	25	31	31	37	56	68	25	27	52	77	94	62	75	83	100
BESTAND	39	3	10	18	5	7	21	35	29	24	51	22	53	45	82	30	27	85	78	100	97
BESTAND	40	5	10	8	7	16	18	38	53	56	54	40	55	24	51	34	69	56	100	69	88

BESTAND 41

1	0	0	1
1	1	0	1
1	1	0	1
1	1	1	1
1	1	0	0
1	0	1	1
1	1	1	0
1	0	0	1
1	1	1	0
0	1	1	1

BESTAND 42

1	1	1	1
1	0	0	1
1	0	1	1
1	0	0	1
1	1	1	1
0	1	1	1
1	0	1	1
0	0	1	1
1	0	0	1
0	0	1	1

BESTAND 43

1	1	1	0
1	1	1	1
0	1	1	1
1	0	0	0
1	1	1	1
1	1	0	1
0	1	0	1
0	1	0	0
1	1	1	1
1	0	0	1

BESTAND 44

1	0	0	1
1	1	0	1
1	1	0	0
1	0	0	1
1	0	0	1
1	1	0	0
0	0	1	1
1	0	1	1
1	0	0	1
1	1	1	1

BESTAND 45

1	1	1	0
1	0	0	1
1	1	0	1
1	0	1	1
1	1	1	1
0	1	1	1
1	1	0	1
1	0	1	1
0	0	0	1
1	1	0	1

BESTAND 46

1	1	1	1
1	1	0	1
1	1	1	1
1	0	0	1
1	1	0	0
1	1	1	0
1	0	1	1
0	0	0	1
0	1	1	0
1	1	1	1

BESTAND 47

0	0	0	1
1	0	1	1
1	0	1	0
1	1	1	1
1	1	0	1
1	0	1	1
0	1	0	1
1	1	0	0
0	0	0	1
1	0	0	1

BESTAND 48

1	1	0	0
1	1	1	1
0	0	1	1
0	0	1	0
1	0	0	1
1	0	1	1
1	0	1	0
1	1	0	0
1	1	0	1
1	0	0	1

BESTAND 49

1	1	1	1
1	0	1	1
1	1	1	1
1	0	0	1
1	1	0	1
1	0	0	1
1	0	1	0
0	1	0	1
1	1	1	1
1	1	1	1

BESTAND 50

0	0	1	1
1	1	0	1
0	1	0	0
0	0	0	1
1	0	1	1
1	0	0	1
0	1	1	1
1	1	1	1
0	0	1	1
1	0	0	1

BESTAND 51

WEG	
DIK	
FAT	BEK
MIS	LOS
KOP	DOE
OOI	PIL
ERG	ZAK
EER	LES
BEL	OUD
ZON	REL
EGO	DAT
VAT	RIT
NEP	DIA
VAN	DOL
DUN	HEL
DOM	SLA
DEK	ZES
MUI	BUI
PEP	SIP
ZEE	DOT
WAT	KIP
OOR	SPA
DAM	DOR
ALT	BOT
DEN	KAT
POP	ZET
ZOT	BOF
VAL	JUK
MAN	WEI
LIP	BOL
VET	BAS
LOT	IEP
ENT	TAS
CEL	ABT
GAS	HUT
EED	UIL
UIT	AKS
MAF	TOP
GAT	ARM
DOG	TOT
AAI	WOL
PAP	BAR
BOA	BED
EEN	LUW
ZUS	ELK
BIG	VEL
EIS	UUR
WEE	BIT
LAP	ION
DOK	URN
AAN	
PIT	

BESTAND 52

HIJ HAD ZICH NET ALS ADVOCAAT GEVESTIGD, EN OP ZIJN KANTOOR DRAAIDE HIJ DOELLOOS IN ZIJN BUREAUSTOEL HEEN EN WEER.

PLOTSELING HOORDE HIJ STAPPEN OP DE GANG.

ALSOF HIJ HET HEEL DRUK HAD, NAM HIJ DE TELEFOON OP EN SPRAK:"NATUURLIJK MEVROUW, DEZE ZAAK WIL IK GAARNE VOOR U BEHANDELEN..."

NA EEN BESCHEIDEN KLOPJE WAS DE MAN BINNENGEKOMEN, EN IN EEN AFWACHTENDE HOUDING BIJ DE DEUR BLIJVEN STAAN. "..NEE MEVROUW, DAT KOST U... PARDON? ALS HET NIET CONVENIEERT, KUNT U GERUST NAAR EEN CONFRATER GAAN... ALS U LIEVER HEBT DAT IK HET DOE... GOED MEVROUW STORT U DAN ALVAST 275 GULDEN, DAN REGEL IK DE ZAAK VOOR U... "

MET EEN BLASE GEBAAR LEGDE HIJ DE TELEFOON NEER EN WENDDE ZICH TOT DE MAN DIE NOG STEEDS BESCHEIDEN BIJ DE DEUR STOND.

"WAT KAN IK VOOR U DOEN?"

DE MAN ZETTE ZIJN TAS OP DE GROND EN ZEI: "ALS HET U SCHIKT, WOU IK DE TELEFOON EVEN AANSLUITEN..."

MAX TAILLEUR

In bestand 52 komen geen hoofdletters voor. Bestand 54 is ontstaan uit bestand 52 door "verschuiving over" 17. Het eerste stukje staat hieronder.

BESTAND 54
H YZA YPU QZTY EVK RCJ RUMFTRRK XW

BESTAND 53

VERSCHIJN	BEFREIN	LURVEN	JAT	COURTISANE
IN	LEF	PEUK	SOUVEREIN	PENOSE
KONIJN	GOZEP	BELABBERD	GEIN	KIT
MOLLEN	SCHAVUIT	MIETERS	KEET	FIJN
BALEIN	KLEUNEN	OMKOPEN	SCHURK	KRAAK
HEITJE	CHANTAGE	LINK	POEN	TRONIE
MAJEM	KASSIAN	EVERZWIJN	KASTELEIN	MAZZEL
ROES	VERDWIJN	TREIN	PLEIN	LIJN
HOMMELES	CHAGRIJN	GRIEBUS	KOTER	OPROER
SCHOREM	OBSCUUR	KLEIN	GABBER	BAK
FILIGREIN	KEDIN	MIJN	MARSEPEIN	MUITERIJ
GIS	BINK	GORDIJN	PETOET	STUK
PIEREMENT	JAJEM	CRIMINEEL	JOFEL	MAFKIKKER
RIJKSHOTELMEP		DRIJFSIJS	KAPSONES	TRAMMELANT
PIMPELEN	BREIN	GREIN	SCHOFT	NEUTJE
BAJES	FIELT	ASTRANT	KEUTEL	SATIJN
SCHOBBEJAKREGIJN		SCHOELJE	KLOFFIE	BONJE
SJOECHE	BRANIE	DRIVE IN	PUS	KAMELENRUG
DOL	PONUM	DOKKEN	LUMMEL	CHARLATAN
TERREIN	KUL	ZWIJN	TEUT	WIJN

De onderstreepte letters zijn hoofdletters.

BESTAND 56

TOEN DE ROOKWOLKEN WAREN OPGETROKKEN, KON MEN DE GELEERDE PRWYTZKOFSKY KLAGEND BIJ DE OVERBLIJFSELEN ZIEN ZITTEN.
"PRAW", ZO RIEP PRWYTZKOFSKY UIT. "DAAR IS MIJNE GANSE KOSTELIJKE BUKKEL IN DE LUCHT GESPRINGT. HET IS KOMMERVOL."
"BENT U IEMAND?" VROEG EEN STEM NAAST HEM.
DE HOOGLERAAR PRWYTZKOFSKY KEEK GETROFFEN OP EN ZAG IN HET MOERAS EEN KLEINE GESTALTE STAAN.
"PRAW" SPRAK PRWYTZKOFSKY, ZICH HERSTELLEND, "IK BEN DER DOCTOR PRWYTZKOFSKY; HOOGLERAAR TE ROMMELDAM; EEN MAN DER WETENSCHAP."
"GOED," HERNAM HET VREEMDE VENTJE, "ZEG DAN TEGEN UW STADGENOTEN, DAT DE DROON WIL ONDERHANDELEN."
"ONDERHANDELEN?" HERHAALDE PROFESSOR PRWYTZKOFSKY VERBAST.
"DAT IS JA SCHOON. VOER MIJ NAAR HEM TOE."
"NEE," ZEI DE VREEMDELING, "UW KUKEL IS GEMETEN: HET WAS EEN MINKUKEL."
EEN OGENBLIK WAS PRWYTZKOFSKY VERSLAGEN.
"PRAW" MOMPELDE HIJ. "WELK EEN VERSCHRIKKELIJKER ONVERSCHAAMDHEID.. MAAR IK WEET WANNEER IK NIET ERWENST BEN.."
EN PRWYTZKOFSKY STAPTE HEEN.

```
BESTAND  59
     START
     AANT:=0
   1 A:=LEES
     ALSA=0DAN3
     ALSA=1DAN ANDERS1
   2 AANT:=AANT+1
     NAAR
   3 SCHRIJF:=AANT
     KLAAR
     5+1+1+6+7+11+1+10+4+1+0
```

```
BESTAND  60
     START
     TEKST:="JAN FICTIEF"
     GROOTSTE:=LEES
   1 A:=LEES
     ALSA=0DAN4ANDERS2
   2 ALSA>GROOTSTEDAN3ANDERS1
   3 GROOTSTE:=A
     NAAR1
   4 TEKST:="HET GROOTS?E GETAL
     NR
     TEKST:="UIT DEZE RIJ IS    "
     SCHRIJF:=GROOTSTE
     KLAAR
     5+12+3+14 +9+0
```

```
BESTAND  61
     START
     SOM:=0
   1 A:=LEES
     ALSA<1000DAN2ANDERS3
   2 SOM:=SOM+A
     NAAR1
   2 SCHRIJF:=SOM
     KLAAR
     5+7+3-10+4+8-47+53+42-73+1000
```

```
BESTAND  62
     START
     SOMPOS:=0
     SOMNEG:=0
   1 A:=LEES
     ALSA<1000DAN2ANDERS4
   2 ALSA<0DAN3
     SOMPOS:=SOMPOS+A
     NAAR1
     SOMNEG:=SOMNEG+A
     NAAR1
   4 SCHRIJF:=SOMPOS
     SCHRIJF:=SOMNEG
     KLAAR
     +4+5-3+7-3+4-10+13+23+1000
```

Hoofdletterwoorden zijn onderstreept.

Mogelijk zullen later in de programmateksten nog spaties worden ingelast.

```
BESTAND  63
     START
   1 A:=LEESSYM
     ALSA:=99DAN2
     ALSA:=86DAN3ANDERS4
   3 TEKST:="ARCHIBALD"
   4 SCHRIJFSYM:=A
     NAAR1
   2 KLAAR
     DE GROTE > ZEI TOT DE KLEINE
     >, > IS EEN MOOIE NAAM.
```

```
        BESTAND  68

AAMBEELD   AANSLAG    AGENDA     AMBIVALENTARABESK    AUTOMOBIEL
BANKBILJETBEJEGENINGBESCHIMPENBLAFFER    BOEZEM     BOUVIER
CONTINU    DAMSLAPER DOCENT     DOPING     DUIKELAAR EEUW
GELAG      GENOT     GEVLIJ     HEETHOOFD HETERDAAD HUMOR
JUWEEL     KANNIBAAL KIESKEURIGLA          LEEP       LOVER
MAGNEET    MAST      MENHIR     NAVRANT    NOODLOT    OLIFANT
OVERAL     PAARD     PARAFRASE  PEGEL      PIETLUT    PLATJE
POND       RUNNER    SCHERTS    SCHORT     SKI        SLOPEN
STOFFER    STRALEN   SUBLIEM    TRAP       TREND      TRIMBAAN
U          UITSCHOT  VALS       VREEMD     WAFFEL     WATERVERF
WIELRENNERXYLOFOON   ZENUWEN
```

	755	236	900	881
	510	421	-270	294
	129	845	506	294
BESTAND 64	128	-365	385	-418
	564	78	129	853
250	952	808	-242	-361
	697	-64	-212	-16
	235	98	-53	-157
639	802	983	219	582
558	60	212	151	411
890	911	-128	-387	779
-374	983	-146	459	112
-281	382	25	228	804
48	344	657	-8	-338
238	-14	-179	485	283
633	982	648	-347	836
-446	-272	218	207	925
972	-213	-146	-122	
671	225	-89	-135	
209	64	-153	-93	
-302	91	-288	-229	
924	-122	-399	211	
-334	909	-13	292	
679	-366	166	345	
-75	-498	111	795	
-170	-221	334	613	
-312	805	994	650	
-103	-348	-480	566	
-75	336	921	738	
869	613	-213	466	
949	290	-14	550	
969	-272	0	992	
126	10	-380	581	
510	381	276	571	
-201	119	-244	-166	
611	-147	-26	-152	
637	803	351	403	
673	494	623	-57	
219	-200	95	707	
-383	748	815	259	
-180	-198	-255	573	
-475	-55	-314	-278	
967	107	678	503	
967	843	-246	481	
-16	968	863	-424	
-115	-405	-474	709	
329	711	568	559	
53	374	351	252	
707	-160	-228	447	
770	83	699	540	
505	438	349	-273	
28	616	103	766	
780	844	233	355	
559	872	168	965	
-386	151	886	-495	
-217	-270	165	-223	
-221	-248	-471	789	
-224	412	839	-234	
-173	757	206	-203	
-450	240	-293	-177	

BESTAND 65

CLUB TAAK INTRODUCTIE PAGINA VOORAL MOGELIJKHEID GEDAANTEVERWISSELING LIJN REFREIN
OP ORGANISATIE U VERENIGING BEHENDIGHEIDSPROEVEN PARAGRAAF MIJN TELEFOONNUMMER ZWIJN
UI ZADEL SPOORWEGMAATSCHAPPIJ OPENING EN LIDMAATSCHAP VERZAMELING UIT IJDELTUIT AARDBEI
ZWOEL BLOEDKORAAL OPEN PLEIN SPOORWEGEMPLACEMENT GEDAANTEVERANDERING TELEFOONBOEK
AF PA IJSBAAN KASTELEIN CITROEN UITGANG IJZERVIJLSEL HOEFSLAG AARDEWERKFABRIEK HARNACHEMENT
IJSSCHOTS WATERVERF QUOTUM NIEUWSGIERIGHEID FEEST

Universiteitsbibliotheek te Utrecht
Centrum Uithof