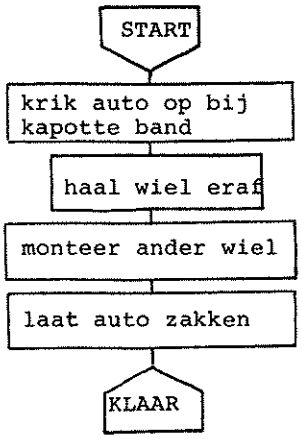
2.3 OPGAVEN

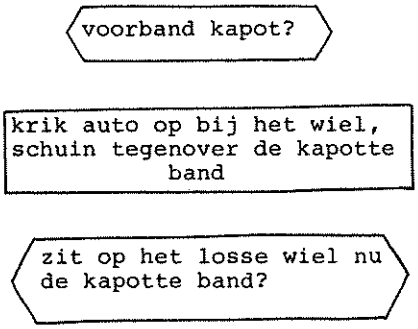
In de voorgaande jaren zijn door de experimenterende docenten opgaven uitgewisseld, die gebruikt werden bij proefwerken e.d. Deze vraagstukken hebben in de meeste gevallen de moeilijkheidsgraad van die uit Compk. Ze zijn hier globaal naar onderwerp gerangschikt en gewijzigd in die zin, dat getallenband gewijzigd is in getalkaarten. Dit laatste in verband met het gebruik van schrapkaarten. Er is afgezien van het vermelden van de herkomst van elk vraagstuk, beslist zonder hiermee ondankbaarheid te willen tonen aan alle trouwe inzenders.

Blokschema's

1) Een automobilist krijgt pech,hij moet een wiel verwisselen want hij heeft een lekke band. Dan gaat hij als volgt te werk:



Er is nu aan automobilist die het ingewikkelder wil doen. Hij wil graag vóór goede banden hebben, en zijn reserveband is niet meer zo best. Als de kapotte band dus vóór is, dan wil hij dat wiel verwisselen met een achterwiel en het reservewiel moet dan achter komen te zitten. Als de kapotte band achter is, dan wil hij die gewoon verwisselen met het reservewiel. Maak voor deze automobilist een blokschema. In dat blokschema mag je gebruik maken van de hokjes, die ook in het bovenstaande schema voorkomen, en verder mag je nog gebruik maken van de hokjes:



2) Een zeker iemand moet een aantal monsters frankeren. Op een monster van 100 g of minder moet 20 ct; op monsters die zwaarder zijn dan 100 g, maar lichter dan 200 g of juist 200 g, moet 50 ct; voor monsters die zwaarder zijn dan 200 g moet het porto eerst worden berekend. Schrijf een blokschema voor dit frankeren, waarin behalve het beginblok en het eindblok, alleen blokken met de volgende vragen en opdrachten voorkomen :

gewicht ≤ 100 g?
gewicht ≤ 200 g?
zijn er nog meer monsters?
frankeer met 20 ct
frankeer met 40 ct
bereken porto
frankeer met berekend porto
neem een monster

3) Een apotheker neemt een hulpje om zijn flessenvoorraad te sorteren. Vuile flessen moet hij weggooien. In bak 1 moeten de flessen met inh ≤ 25 cc. In bak 2 met inh ≤ 100 cc, maar meer dan 25 cc. In bak 3 met inh ≤ 250 cc, maar meer dan 100 cc. In bak 4 met inhoud groter dan 250 cc. Gebruik alleen de volgende opdrachten en vragen:

fles vuil?
gooi weg
pak fles
lees inhoud op flesbodem
inh > 25 cc
inh ≤ 100 cc
inh > 250 cc
zijn er nog flessen
in bak 1
in bak 2
in bak 3
in bak 4

4) Een zekere automobilist laat telkens na 3000 km rijden olie verversen (oude olie er uit en nieuwe er in). Als het langer duurt dan twee maanden voor hij 3000 km gereden heeft, dan laat hij al na twee maanden olie verversen. Elke dag controleert hij het oliepeil en als de stand te laag is, laat hij olie bijvullen. Als de olie ververst moet worden, laat hij op die dag geen olie bijvullen.

Maak een blokschema met de vragen en opdrachten:

meer dan 3000 km gereden?
oliepeil te laag?
laat olie verversen
meer dan twee maanden gereden?
laat olie bijvullen

5) Een zeker iemand laat een partij appels keuren op kleur, smaak en gewicht. Zijn smaak, kleur en gewicht goed, dan is de partij eerste soort. Zijn er twee van de drie goed, dan is de partij tweede soort. Alleen smaak goed, alleen kleur goed, alleen gewicht goed of niets goed, dan is de partij derde soort.
Schrijf een blokschema met de vragen en opdrachten:

bepaal gewicht, kleur en smaak
gewicht goed?
kleur goed?
smaak goed?
partij is eerste soort
partij is tweede soort
partij is derde soort

6) Een zekere leerling moet voor een proefwerk een aantal stellingen leren. Telkens als hij een stelling geleerd heeft, zegt hij die op. Gaat het goed, dan gaat hij de volgende stelling leren. Gaat het fout, dan leert hij dezelfde stelling nog eens. Schrijf een blokschema met de vragen en opdrachten:

leer de eerste stelling
leer deze stelling nog eens
zeg de stelling op
stelling goed opgezegd?
leer de volgende stelling
zijn er nog stellingen te leren?

7) Maak een blokschema voor een automaat, die koffie levert als je er een kwartje in doet en voor voor een dubbeltje een beker melk geeft. Gebruik de vragen en opdrachten:

zijn er nog bekertjes ?
is het muntstuk een kwartje?
zet een beker klaar
geef muntstuk terug
zet koffiekraan even open
is er nog melk?
is er nog koffie?
zet melkkraan even open

8) Iemand oefent het hoogspringen door de lat op 1 meter hoogte te stellen. Na elke goede sprong wordt dan de lat 5 centimeter hoger gezet. Een foutsprong mag altijd éénmaal over.
Schrijf een blokschema met de vragen en opdrachten

zet lat op 1 meter hoogte
spring
sprong fout?
zet lat 5 cm hoger
oefening is klaar
eerste fout op deze hoogte?

9) Een zekere leraar geeft op het kerstrapport g (goed), v (voldoende), t (twijfelachtig) of o (onvoldoende). Hij heeft van elke leerling één *proefwerk*-cijfer, dat cijfer is x. Hij heeft verder van elke leerling een *indruk*, die goed of slecht is. Hij gaat nu als volgt te werk:
Er komt een g op het rapport als $x > 7$ (dus $x = 8$ of 9 of 10, ongeacht de indruk).
Er komt een v op het rapport als:
$x = 7$, ongeacht de indruk
$x = 6$, en de indruk is goed
Er komt een t op het rapport als:
$x = 6$, en de indruk is slecht
$x = 5$, ongeacht de indruk
$x = 4$, en de indruk is goed.
Er komt een o op het rapport als:
$x = 4$, en de indruk is slecht
$x < 4$ (dus $x = 1$ of 2 of 3, ongeacht de indruk)

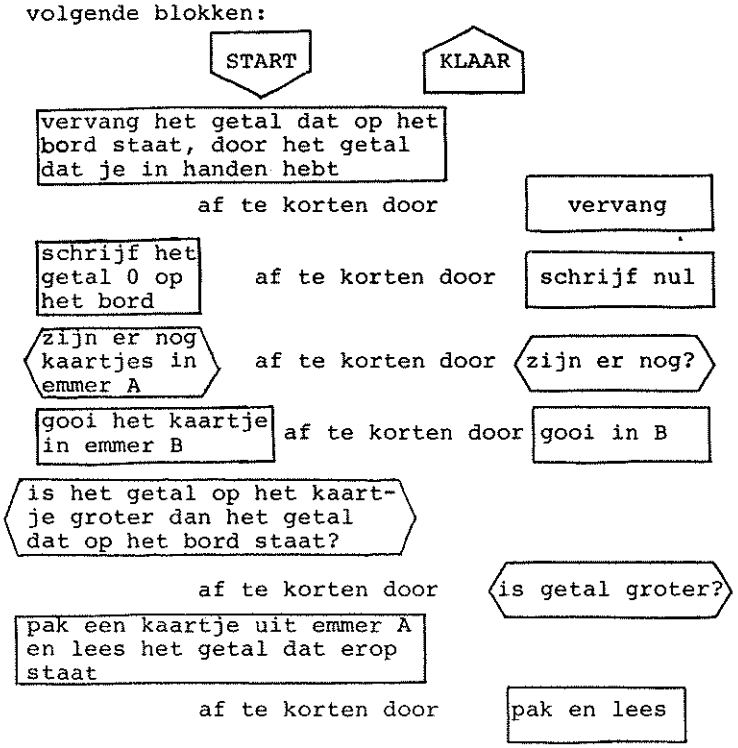
Maak nu een blokschema voor die leraar. Volgens dat blokschema moet hij alle leerlingen hun rapportbeoordeling geven In de hokjes mogen staan:

$x <$ cijfer?	neem eerste leerling
$x =$ cijfer?	neem volgende leerling
$x >$ cijfer?	alle leerlingen gehad?
indruk goed?	g op rapport
START	v op rapport
KLAAR	t op rapport
	o op rapport

10) Een partij bloemkolen moet over vier kisten worden verdeeld. In kist 1 moeten bloemkolen lichter dan 1 kg. In kist 2 moeten bloemkolen lichter dan 1½ kg, maar zwaarder dan 1 kg of precies 1 kg. In kist 3 moeten bloemkolen lichter dan 2 kg, maar zwaarder dan 1½ kg, of precies 1½ kg. In kist 4 moeten bloemkolen van 2 kg, of zwaarder
Maak een blokschema van het sorteerwerk en schrijf daarbij in de hokjes:

pak een bloemkool
weeg de bloemkool
gewicht ⩾ 2 kg?
gewicht < 1 kg?
gewicht < 1½ kg?
nog meer bloemkolen te sorteren?
in kist 1
in kist 2
in kist 3
in kist 4

11) In emmer A zitten kaartjes; op elk kaartje staat een *natuurlijk* getal. Alle getallen zijn *verschillend* en *groter dan nul*. In emmer B zit niets. Iemand moet het *grootste* getal, dat zich in emmer A bevindt, op een bord schrijven. Maak een blokschema, door gebruik te maken van de volgende blokken:



Eenvoudige rekenschema's of programma's

12) Een klas heeft 6 leerlingen. Hun leeftijden staan op getalkaarten. Maak een blokschema dat die getalkaarten leest, en dat de som èn het gemiddelde van alle leeftijden schrijft.

13) Schrijf een blokschema dat de getallen a, b en c van een getalkaart laat lezen en berekent:

$$x = \frac{a^2}{b\ c} + \frac{b^2}{a\ c} + \frac{c^2}{a\ b}$$

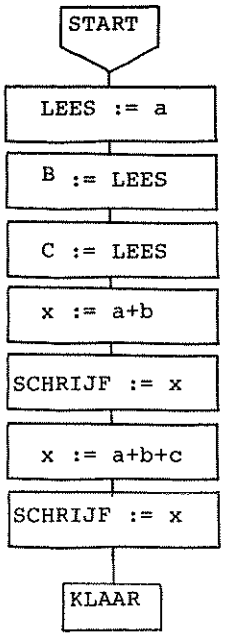
$$y = \frac{3}{\frac{1}{a^3} + \frac{1}{b^3} + \frac{1}{c^3}}$$

Achtereenvolgens moeten a, b, c, x en y worden afdrukt.

14) Iemand heeft een programma geschreven en daarin onder andere de volgende instructies gebruikt. Welke ervan zijn goed en welke zullen niet door de computer worden aanvaard.

a := b : c	a := -15 × jantje
a := b/c	a := jantje × -15
start	17 := 18.5 - 1.5
cruyff := gevloerd	SCHRIJF := schrijf
klas 4 := kampioen	SCHRIJF := 16 + 84
a := 3,5 + 26	k := lees
KLAAR	LEES := x
x := y/z - a	b := schrijf

15) Iemand heeft een blokschema gemaakt voor het lezen van 3 getallen, te weten a, b en c. Hij wil 2 getallen schrijven nl. de uitkomsten van de optellingen a+b en a+b+c. Het blokschema ziet er zo uit:



Je ziet natuurlijk wel dat er fouten in dit schema zitten. Schrijf het blokschema en verbeter daarbij de fouten. Zet erbij hoeveel fouten erin zitten.

16) Maak een blokschema voor een computer. De computer moet twee getallen, a en b, lezen; dan x uitrekenen en tenslotte x schrijven; de formule voor x is:

$$x = \frac{a^2 + 2ab + b^2}{a^2 - 2ab - b^2}$$

Controleer het programma voor een getalkaart, waarop de getallen 3 en 5 staan.
Extra vraag: als a en b nu eens breuken zijn, bijv. a = 2.6387 en b = 0.0193, moet het blokschema daarvoor dan anders worden geschreven?

17) Als een onderstaande bewering een opdracht is door de computer wordt geaccepteerd, zet er dan 'ja' achter, anders 'nee'.

ab := a × b	15 := 4 × 5
a := 5 + a	das := das × das
15 := 3 × 5	jan := ziek
15 = 3 × 5	

18) Schrijf een blokschema, dat (door jou zelf bedachte) getallen in de GP's a en b stopt en daarna berekent:

$$\frac{a^3 - b^3}{a+b}$$

Schrijf onder het blokschema in welke GP het antwoord zit.

19) Op een zekere school wordt het overgangsrapportcijfer berekend door op te tellen: het cijfer van het eerste rapport, tweemaal het cijfer van het tweede rapport en tweemaal het cijfer van het derde rapport en deze som te delen door 5.
Schrijf een blokschema, dat (door jou zelf bedachte) cijfers van de eerste drie rapporten in GP's stopt en het overgangsrapportcijfer berekent.
Schrijf onder het blokschema in welke GP dit overgangsrapport zit.

20) Als er bij het huiswerk geen repetitie is, dan zijn er 2 gevallen:
1. Er is huiswerk voor meer dan 4 vakken.
   Dan wordt aan elk vak 40 min. gewerkt.
2. Er is huiswerk voor 4 of minder vakken.
   Dan wordt aan elk vak 50 min. gewerkt.

Als er bij het huiswerk wel een repetitie is, dan wordt aan elk ander vak 40 min. gewerkt, en aan de repetitie 90 min.
Schrijf een blokschema en een programma, dat gegevens van de getalkaart leest en berekent, hoeveel minuten aan het huiswerk is gewerkt, en afdrukt: 'Aan het huiswerk is ... minuten gewerkt'.
In plaats van de puntjes moet dan het door de computer berekende getal worden afgedrukt.
Op de getalkaart staat:
1. Een gegeven dat uitdrukt of er wel of geen repetitie is. Kies hiervoor zelf iets en schrijf onder de oplossing wat je hebt gekozen.
2. Het aantal niet-repetitievakken.

Schrijf zinvolle getallen onder je programma.

21) Iemand (die een rijbewijs heeft) huurt een dag een auto. Voor 25 gulden mag men 100 km rijden. Rijdt men meer, dan moet men voor elke km. die men meer rijdt dan 100 km, 15 *cent* extra betalen. Is men jonger dan 23 jaar, dan moet men 5 gulden meer betalen.
Schrijf een blokschema en een programma, dat het aantal gereden km's en de leeftijd van de huurder van de getalkaarten leest en afdrukt:
'De kosten bedragen ..... gulden".
Schrijf zinvolle getallen onder aan het programma.

22) Voor ambtenaren geldt ongeveer de volgende pensioenregeling. De pensioengrondslag is het grootste van: (het salaris over het laatste dienstjaar) en (het gemiddelde van de salarissen over de laatste drie dienstjaren).
Voor elk dienstjaar krijgt men als pensioen $1\frac{3}{4}$ % van de pensioengrondslag (dus: als iemand 10 dienstjaren heeft dan krijgt hij: $10 \times \frac{1\frac{3}{4}}{100}$ de pensioengrondslag), maar nooit meer dan 70% van de pensioengrondslag.
Schrijf een blokschema en een programma, dat de salarissen over de laatste 3 dienstjaren en het aantal dienstjaren van de getalkaarten leest (het salaris over het laatste dienstjaar moet op de derde plaats staan) en het pensioen berekent en afdrukt.
Schrijf weer zinvolle getallen onder aan het programma.

23)

a := 5	mag wel/niet
kees := a	mag wel/niet
JAN := p	mag wel/niet
ab := 8	mag wel/niet
c := 1,4	mag wel/niet
p := a:b	mag wel/niet
a×b := 6	mag wel/niet
a := a×8	mag wel/niet.

24) Welke waarden hebben de variabelen a en b na afwerking van onderstaand programma? Maak een geheugenschema.

```
START
a := 3
b := a+a
a := b/a
a := a×a
b := a-b
b := a/b
KLAAR
```

25) Schrijf een blokschema voor het berekenen van:

$$\frac{347^3 - 49^3}{347 - 49}$$

(niet zelf de uitkomst berekenen !)

26)

```
START
TEKST := "....."
a := LEES
b := LEES
b := b×b
a := 5
a := a+b
NR
SCHRIJF := a
KLAAR
```

a. Wat wordt er afgedrukt door het programma hierboven als er op de getalkaarten staat:

+3+6 ?

b. En wat als er staat:

+13+6 ?

c. Wat wordt er afgedrukt als er +13+6 op de getalkaarten staat en we laten de derde regel uit het programma (a := LEES) weg?

27) Op een zekere getalkaart staan drie getallen, achtereenvolgens a, b en c. Schrijf een programma dat deze getallen leest en het volgende verricht:

indien $a \geqslant b+c$, dan het getal 1 schrijven
indien $a < b \times c$, dan het getal 2 schrijven
indien $b > c$, dan het getal 3 schrijven
indien $a^2 \geqslant b^2 - c^2$, dan het getal 4 schrijven.

Indien aan geen der 4 voorwaarden wordt voldaan, dan het getal 0 schrijven.

Voorbeeld:
als a = 15 b = 5 c = 12
wordt er voldaan aan de 2^e en de 4^e voorwaarden en schrijft de computer dus op papier:

2 4

28) Schrijf een programma dat twee getallen (a en b) van de band laat lezen en berekent:

$$x = \frac{a^2 + 2ab + b^2}{a^2 - 2ab + b^2}$$

en deze x laat afdrukken. Doe dit in zo weinig mogelijk opdrachten.

29) Op een zekere getalkaart staan twee repetitie cijfers. Schrijf een programma dat deze twee cijfers laat lezen en laat afdrukken:

goed (als het gemiddelde van de repetitiecijfers of meer is) of
onvoldoende (als dit gemiddelde minder dan 5 is)
matig (voor het geval dat niet "goed" of "onvoldoende" wordt geschreven).

Cyclus

30) In deze les zullen we proberen het verwerken van een computerprogramma na te bootsen. We gaan in groepjes van zes bij elkaar zitten. In elk groepje wordt een leider/leidster gekozen, die het programma leest en de opdrachten doorgeeft aan de andere leden van de groep. Er is een rekenaar(ster) die alle berekeningen uitvoert en door de leider worden de resultaten doorgegeven aan de betrokken personen. Er is in een 'dubbelrol' een lezer en een schrijver, die de LEES-, resp. de SCHRIJF-opdrachten uitvoert. Verder zijn er in elke groep drie geheugenplaatsen, die de getallen onthouden (noteren) zoals die in het programma voorkomen.
Verwerk op deze manier het volgende programma:

```
     START
     TEKST := "imitatiecomputer"
     NR
     som := 0
     teller := 1
3    getal := LEES
     som := som + getal
     teller := teller + 1
     ALS teller > 10 DAN 7 ANDERS 3
7    TEKST := "de som van de getallen is"
     SCHRIJF := som
     KLAAR
     + 1123 + 11 + 23.22 + 27.15 + 21 + 30 + 21
     + 38.4 + 1  + 200
```

31)
Op zekere getalkaarten staat een aantal positieve getallen afgesloten door een nul. Schrijf een programma dat deze band leest en de positieve getallen onder elkaar afdrukt met naast elk getal het omgekeerde van dat getal. (Men lette bij de correctie op het feit dat niet het omgekeerde van nul berekent wordt).

32) Als men 8 halveert krijgt men 4, halveert men het resultaat weer dan krijgen we ?. Na 7 maal halveren hebben we $\frac{1}{16}$.

Na voldoend aantal malen halveren wordt het resultaat kleiner dan $\frac{1}{1000}$.

Nu de opgave.
Schrijf een programma waarbij een onbekend positief getal van een getalkaart wordt gelezen en waarbij als antwoord verschijnt hoe vaak, te beginnen met dit getal, het tenminste gehalveerd moet worden om een resultaat kleiner dan $\frac{1}{1000}$ te krijgen.
(Men lette bij correctie op het feit dat het te lezen getal kleiner dan 0.001 kan zijn).

33) Een zekere winkelier maakt de volgende getallen op getalkaarten:
het 1e getal = de prijs van artikel 1 per stuk
het 2e getal = het aantal stuks van artikel 1, dat verkocht is
het 3e getal = de prijs van artikel 2 per stuk
het 4e getal = het aantal stuks van artikel 2, dat verkocht is
enz.
De rij getallen wordt afgesloten door -1.
Schrijf een programma dat de omzet (dat is de totale hoeveelheid geld die hij door de verkoop heeft ontvangen), van deze winkelier berekent.

34) Op zekere getalkaarten staat een rij getallen $\neq 0$, gevolgd door een nul. Schrijf een programma, dat de som van al de getallen groter dan tien berekent en afdrukt.

35) Op zekere getalkaarten staat een rij getallen $\neq 0$, gevolgd door een nul. Lees elk getal en druk af PLUS als dat getal > 0 is en MIN als dat getal < 0 is.

36) Op zekere getalkaarten staat een rij getallen $\neq 0$, gevolgd door een nul. Schrijf een programma dat onderzoekt of er een positief getal bij is. Dus je drukt af JA als er *wel* een getal > 0 bij is en NEE als er *geen* getal > 0 bij is.

37) Weer dezelfde getalkaarten. Lees elk getal in, en druk af GROTER als dat getal > 1
en TUSSEN als dat getal a zó is dat $-1 \leq a \leq 1$
en KLEINER als dat getal < -1 is.

38) Op zekere getalkaarten staat een even aanta getallen gevolgd door een nul. Schrijf een programma, dat achtereenvolgens afdrukt:
de grootste van het 1e en 2e getal
de grootste van het 3e en 4e getal
de grootste van het 5e en 6e getal enz.

39) Een steen valt van een 100 meter hoge toren We willen de computer laten berekenen na hoevee tijd de steen op de grond is aangekomen. Dit ga als volgt:
De afstand a van de steen tot de grond is $100 - 5t^2$; hierin is t de tijd die de steen valt.
Laat in een programma t starten bij 0 seconden, laat t daarna steeds 0,1 sec. toenemen. Bereken bij elke t de afstand a. Laat het programma stopen als a negatief wordt en laat de tijd t op dat moment afdrukken.

40) Op getalkaarten staat een aantal gehele getallen die groter zijn dan nul en kleiner dan 1 Deze rij getallen wordt op de band afgesloten m het getal nul. Schrijf een programma voor het volgende.
Lees de getallen en tel hoeveel getallen er liggen in de intervallen 1 t/m 25, 26 t/m 50, 51 t/m 75 en 76 t/m 100. Wanneer er een 0 wordt gelezen moet de telling stoppen en de vier antwoorden (nl. de vier getelde hoeveelheden), worden geschreven.

RIJ

41) Op een band staan 20 getallen (gehele getallen van hooguit zes cijfers). Schrijf een progr ma dat deze band laat lezen en de getallen op d volgende wijze laat afdrukken:

1e getal	6e getal	11e getal	16e getal
2e getal	7e getal	12e getal	17e getal
3e getal	8e getal	13e getal	18e getal
4e getal	9e getal	14e getal	19e getal
5e getal	10e getal	15e getal	20e getal.

42) Op zekere getalkaarten staan 1000 rapportcifers (gehele getallen 1 tot en met 10).
Schrijf een programma dat telt hoe vaak het cijfer 1 voorkomt, ook hoe vaak het cijfer 2 voorkomt, enz. tot en met de tienen. Laat deze tien antwoorden afdrukken.

43) Op getalkaarten staan 100 getallen. Het zijn gehele getallen van hoogstens zes cijfers. Deze getallen moeten worden gelezen en in een afwijkende volgorde worden geschreven, namelijk tien getallen op één regel, in omgekeerde volgorde. Dus zo:

10e getal 9e getal 8e getal ..... 1e getal
20e getal 19e getal ..... 11e getal
.....
1000e getal 999e getal 991e getal

44) Op getalkaarten staan de leeftijden van de leerlingen van een zekere school, die variëren van 11 tot 21 jaar. De rij getallen wordt afgesloten door een nul. Schrijf een programma, waarbij de kaarten éénmaal gelezen worden en waarin geteld wordt hoeveel leerlingen van 11 jaar er zijn, hoeveel leerlingen van 12 jaar er zijn, hoeveel van 13 jaar enz. tot en met 21 jaar.

45) Iemand wil (we weten niet waarom) de volgende getallen in een rij geheugenplaatsen hebben.

a(1)	a(2)	a(3)	a(4)	a(5)	a(6)	a(7)	a(8)	a(9)
1	2	3	4	5	4	3	2	1

Schrijf hiervoor een zo kort mogelijk programma.

46) Op zekere getalkaarten staan 20 getallen $a_1, a_2, \ldots a_{10}, b_1, b_2, \ldots b_{10}$. Schrijf een programma dat de waarde van $a_1 \times b_{10} + a_2 \times b_9 + \ldots + a_{10} \times b_1$ berekent en afdrukt.

47) Op zekere getalkaarten staan 100 getallen $g_1, g_2, \ldots, g_{100}$. Schrijf een programma voor het berekenen en afdrukken van g_1+g_{51}, en van g_2+g_{52}, en g_3+g_{53}, $\ldots g_{50}+g_{100}$.

Opdrachten G, SCHRIJF(n,m), GOK

48) Op een zekere getalkaart staat een positief geheel getal. Schrijf een programma, zodat de computer afdrukt

JA indien het getal op de kaart wel door 3 of 4
NEE indien het getal op de kaart noch door 3 noch door 4 deelbaar is.

49) Op zekere getalkaarten staat een rij positieve gehele getallen, afgesloten door een nul.
Schrijf een programma, zodat de computer afdrukt

NEE indien geen enkel getal door 3 deelbaar is
JA indien er wel een getal onder is, dat deelbaar is door 3.

50)
Geef een blokschema van het onderstaand programma. Geef met je eigen woorden weer, wat 't programma doet.

```
      START
      TEKST := "....."
      NR
 1    a := LEES
 2    ALS a = 0 DAN 11 ANDERS 4
 3
 4    a := a/5
      b := G(a)
      ALS a = b DAN 1 ANDERS 8
 8    NR
      TEKST := "NEE"
      NAAR 13
 11   NR
 12   TEKST := "JA"
 13   KLAAR
```

Als de getalkaart er als volgt uitziet:
+ 15 + 20 + 25 + 0. Geef dan weer, wat er afgedrukt wordt. Als er op regel 3 NR wordt ingevuld, wordt er wel/niet iets anders afgedrukt. Als in regel 2 de 11 door een 12 vervangen wordt, wordt er wel/niet iets anders afgedrukt.

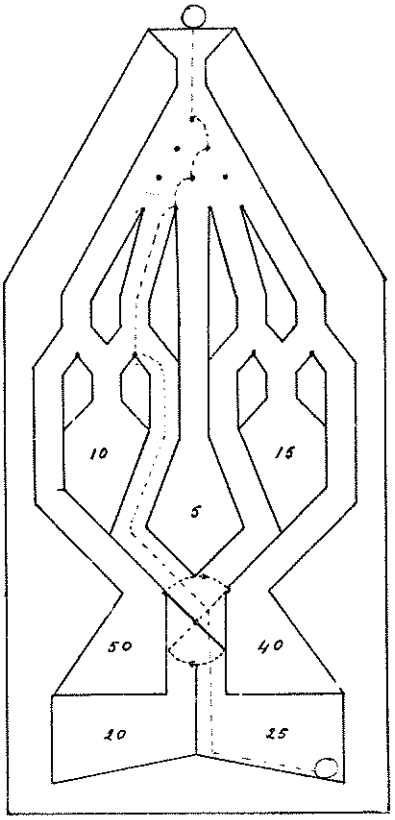
51) Op een zekere getalkaart staat een prijs in guldens, dus bijvoorbeeld 8.35 voor ƒ 8,35. Schrijf een programma dat de nieuwe prijs laat berekenen bij een prijsverhoging van 8%. Alleen wanneer deze nieuwe prijs niet op gehele centen uitkomt, moet naar boven op hele centen worden afgerond. De nieuwe prijs moet geschreven worden in guldens.

52) Op zekere getalkaarten staat een onbekend aantal positieve getallen, afgesloten door een nul. Schrijf een programma dat de band laat lezen en deze getallen onder elkaar laat afdrukken, met naast elk getal het omgekeerde van dat getal, beide in drie decimalen nauwkeurig.

53) Op zekere getalkaart staat één getal, namelijk de rente die een bank geeft (bijvoorbeeld 5 voor 5% of 4.5 voor 4½%). Een bedrag van 100 gulden wordt op deze bank gezet. Na één jaar wordt de rente, afgerond op hele centen, erbij geteld. Het jaar daarop wordt van dit nieuwe bedrag de rente berekend en erbij geteld. Evenzo de volgende jaren. Laat de computer de rente van de kaart lezen en een lijst maken, waarin staat het te goed (het bedrag dat op de bank staat) na één jaar, na twee jaar, na drie jaar, enz. De lijst moet stoppen als het te goed boven 200 gulden gekomen is.

54) Uit een vat met een inhoud van x liter worden potten met een inhoud van y liter getapt. Deze potten worden verpakt in kisten van n potten. Schrijf een programma dat van een getalkaart de 3 getallen x, y en n leest en vervolgens afdrukt:
a) het aantal kisten (ak) dat verpakt kan worden;
b) het aantal potten (ap) dat overblijft;
c) de hoeveelheid (hv) die in het vat achterblijft, nadat de laatste volle pot eruit getapt is.

55) Een aantal balletjes wordt boven in het apparaat (zie tekening) één voor één losgelaten. Elk balletje vindt zijn weg naar één van de genummerde hokjes via pennetjes, kanaaltjes en soms via een klep. De balletjes vallen boven op de pennetjes en hebben daarna 50% kans om naar rechts en 50% kans om naar links te gaan. De klep verandert van positie, wanneer er een balletje is gepasseerd. In de vakken 20 en 25 komen dus of evenveel balletjes, of het verschil is 1.
Schrijf een programma voor de simulatie van dit toestel, wanneer 100 balletjes gegooid worden. Laat de vulling van elk van de zeven vakjes afdrukken en bovendien de totaalscore, wanneer elk balletje de waarde krijgt die op het vakje staat aangegeven.



56) Schrijf een programma dat een aantal toevalsgetallen produceert. Het begingetal van 5 cijfers wordt van een kaart gelezen. Het toevalsgetal krijgt men door dit getal met 101 te vermenigvuldigen, bij het product 33333 op te tellen en dan de laatste vijf cijfers van het antwoord te nemen. Uit dit getal wordt het volgende toevalsgetal verkregen.

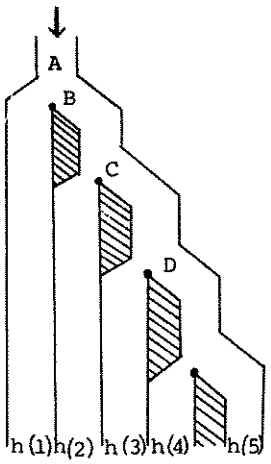
57) Schrijf een programma dat het getal n van een getalkaart leest en vervolgens n keer met een dobbelsteen gooit en afdrukt hoe vaak er 1,2,3 enz. is gegooid.

58) Schrijf een programma dat via toevalsgetallen een kolom van 13 wedstrijden invult van de voetbaltoto.

59) Schrijf een programma dat een frequentieverdeling maakt van de som van het aantal ogen bij worpe met twee dobbelstenen. Vergelijk het resultaat met een theoretisch berekende frequentieverdeling.

60) Het is mogelijk de computer een melodie te late componeren. We gebruiken de acht tonen c,d,e,f,g,a, b,c^1. Kies de eerste toon met behulp van de opdrach GOK(,). Laat ook de lengte daarvan gokken (2,1 of ½ tel). Schrijf een programma voor een melodie van 16 tellen. Laat het resultaat afdrukken in de vorm van a2,c1,$c^1$1,e½,.....

61)



Bij A vallen 100 knikkers bi nen (zie figuur). Bij elk va de splitsingen B,C,D enz. heeft een knikker evenveel k op naar links of naar rechts vallen. Schrijf een programm voor de simulatie van dit to stel en laat de vulling bepa len van de hokjes h(1),h(2), t/m h(5). Neem in dit progra ma ook op het maken van een liggend histogram in de vorm van

```
h(1)  x x x x x x
h(2)  x x x x
h(3)  x x x x
h(4)  x x
h(5)  x x
```

Bestanden

62) In een winkel worden 100 verschillende artikelen verkocht. Deze artikelen zijn genummerd van 1 t/m 100. De winkel wordt bevoorraad vanuit een aangebouwd magazijn. In dit magazijn bevindt zich van elk der 100 artikelen een aantal stuks in voorraad. De voorraadadministratie van dit magazijn wordt gevoerd met behulp van een computer. Telkens wanneer iemand van een bepaald artikel een aantal stuks uit het magazijn naar de winkel brengt, schrapt de magazijnbediende twee getallen aan op een getalkaart, namelijk eerst het nummer van het artikel en daarna het aantal stuks dat hij heeft afgegeven. Aan het eind van elke dag schrapt hij aan het eind van de getallenrij het getal -1 aan.

Opmerkingen:

1. Van een bepaald artikel kan men meer dan eens per dag een aantal stuks uit het magazijn halen.
2. Het is niet zeker dat per dag van elk artikel een aantal stuks uit het magazijn wordt gehaald.

In het achtergrondgeheugen van de computer zijn onder andere de volgende bestanden aanwezig:
BESTAND(1) bestaat uit 100 getallen die aangeven: het aantal stuks dat van elk artikel in voorraad is (dus in het magazijn). Dus: het vijfde getal van BESTAND(1) is het aantal stuks dat van artikel nummer 5 in voorraad is.
BESTAND(2) bestaat uit 100 getallen die aangeven: de inkoopsprijs van elk artikel. Dus: het vijfde getal van BESTAND(2) is de prijs van artikel nummer 5.

Opgaven hierover:

a) Op zekere dag worden de inkoopsprijzen van alle artikelen met 2% verhoogd. Schrijf een programma dat ervoor zorgt, dat de nieuwe prijzen van de artikelen in BESTAND(2) komen.
b) Schrijf een programma dat aan het eind van een dag het voorraadbestand bijwerkt met behulp van de gegevens van de getalkaarten (dit programma moet er dus voor zorgen dat het oude BESTAND(1), zoals dat aan het begin van de dag was, wordt vervangen door een nieuw BESTAND(1) waarin voor elk artikel het aantal stuks staat dat aan het eind van de dag in het magazijn is).
c) Ook moet het programma de nummers laten afdrukken van alle artikelen, waarvan aan het eind van de dag minder dan 30 stuks in het magazijn aanwezig zijn (dan worden deze artikelen die avond bij de grossiers besteld en de volgende dag vóór openingstijd aan het magazijn bezorgd).

63) In BESTAND(11) staan 1000 saldi (geldbedragen) van rekeninghouders in de volgorde van de gironummers. Men wil onder elkaar de gironummers 1 t/m 1000 afgedrukt hebben, met naast elk nummer het saldo. Schrijf hiervoor een programma.

64) In BESTAND(3) staan de leeftijden van 1000 leerlingen van onze school. Schrijf een programma dat de gemiddelde leeftijd uitrekent.

65) In BESTAND(11) staan 1000 saldi (geldbedragen) van rekeninghouders in de volgorde van de gironummers (1 t/m 1000). Men wil onder elkaar afgedrukt hebben de saldi die groter zijn dan 30 gulden, met naast elk saldo het bijbehorende gironummer. Schrijf hiervoor het programma.

66) In de bestanden 101 t/m 107 van het politiearchief staan de 200 bekendste Nederlandse misdadigers als volgt geregistreerd:

Bestand 101 het geslacht (0=man, 1=vrouw)
Bestand 102 geboortejaar in vier cijfers
Bestand 103 lengte in centimeters
Bestand 104 in hechtenis op dit moment? (0=nee, 1=ja)
Bestand 105 aantal veroordelingen
Bestand 106 aard van het laatste misdrijf (1=inbraak, 2=geweldpleging, 3=vermogensdelict, 4=overige misdaden)

Er is op elk politiebureau een genummerde lijst met de 200 namen, in dezelfde volgorde als in de bestanden is aangehouden.

a) Er is vannacht een inbraak gepleegd door een (volgens getuigeverklaringen) man van minstens 1 meter 70 lang van in de dertig jaar (d.w.z. $30 \leq$ leeftijd < 40).
Schrijf een programma voor het afdrukken van de nummers van de inbrekers (= mensen die de laatste keer voor inbraak veroordeeld zijn) die het misdrijf gepleegd kunnen hebben, rekening houdend met de getuigeverklaringen.

b) Uit de gegevens van het politiearchief wil men een statistiekje samenstellen en wel hoeveel er éénmaal zijn veroordeeld, hoeveel er tweemaal zijn veroordeeld, hoeveel driemaal, enz. Geen enkele

geregistreerde heeft meer dan 10 veroordelingen achter de rug. Schrijf een programma dat deze tien getallen laat schrijven.

c) Bedenk een manier om in één bestand van 200 getallen van alle misdadigers te registreren de aard van het misdrijf bij elke veroordeling (dus niet alleen de laatste). Je mag aannemen dat niemand meer dan 10 veroordelingen achter de rug heeft. Geef hierbij expliciet aan hoe iemand met vier veroordelingen wegens misdrijven +1,+1,+4,+3 in dit bestand geregistreerd staat.
Geef in een blokschema aan hoe deze registratie ontstaat als deze vier getallen van een getalkaart worden gelezen. Is de door jou bedachte manier geschikt om alle nummers te vinden van de personen, die wel eens vanwege misdaad + 2 veroordeeld zijn? Zo ja, hoe moet dat worden geprogrammeerd?

67) De bestanden 201 t/m 206 bevatten het bevolkingsregister van een klein dorp. Alle 1000 bewoners van de laatste tweehonderd jaar staan hierin vermeld. We nemen voor het gemak aan dat er geen bewoners van buitenaf in die tijd aan het register zijn toegevoegd. De registratie is als volgt:

Bestand 201 geslacht (0=man, 1=vrouw)
Bestand 202 burgelijke staat (0=ongehuwd, 1=gehuwd)
Bestand 203 geboortejaar (4 cijfers)
Bestand 204 in leven (0=nee, 1=ja)
Bestand 205 het nummer van de vader
Bestand 206 nummer van de echtgenoot (echtgenote)
(0 bij ongehuwd zijn)

Er is een lijst waarop de volledige namen en adressen van de inwoners, genummerd in dezelfde volgorde als in de bestanden is aangeboden.

a) Aan het loket verschijnt een inwoner die een uittreksel uit het geboorteregister wil hebben, waarop zijn naam en de namen van zijn beide ouders staan vermeld. Het nummer van de inwoner wordt op een schrapkaart gezet. Schrijf een programma, waarin deze getalkaart wordt gelezen en dit nummer en de nummers van de beide ouders worden afgedrukt.

b) Schrijf een programma waarin een nummer van een getalkaart wordt gelezen, waarna het nummer van de grootmoeder aan moederszijde wordt gezocht en afgedrukt.

c) Schrijf een programma waarin wordt bepaald en geschreven hoeveel kinderen meneer nummer 65 heeft (gehad).

d) Voor een onderzoek (over de woonomstandigheden van vrijgezellen) heeft men de nummers nodig van de te enqueteren ongehuwde inwoners tussen de 30 en 50 jaar. Laat een lijst maken van de nummers met naast elk nummer de leeftijd.

e) Voor een statistiek wil men weten hoeveel inwoners er zijn van 0 tot 9 jaar, hoeveel van 10 tot 19 jaar, hoeveel van 20 tot 29 jaar, enz. enz., tot en met 99 jaar en hoeveel van 100 of ouder. Laat een lijst afdrukken van deze aantallen.

68) Vliegtuigreservering.
In bestand 41 staan de vrije plaatsen (=0) en gereserveerde plaatsen (=1) in de opstelling van Compk figuur 15.1. Schrijf een programma voor het reserveren voor 4 plaatsen naast elkaar. Indien dit niet mogelijk is, onderzoek dan of er wèl vier willekeurige vrije plaatsen zijn. Als resultaat moet verschijnen :

"gereserveerd op plaatsen nummers: .. .. .." of
"er zijn vier vrije plaatsen, maar niet naast elkaar" of
"er zijn geen vier vrije plaatsen".

Denk om het bijwerken van het bestand.

69) Iemand beweert dat men een kans $\frac{1}{10}$ heeft op een tweeletterig Nederlands woord, als men twee willekeurig gekozen letters die niet medeklinkers zijn achter elkaar zet.
a) Schrijf een programma dat 100 tweelettercombinaties 'gokt' en afdrukt, zonder dat hierin combinaties van twee medeklinkers voorkomen. Door te tellen hoeveel Nederlandse woorden hierin voorkomen, kan bepaald worden of de stelling aannemelijk is of niet.
b) Neem aan dat de 100 tweelettercombinaties uit a) op schrapkaarten zijn gezet en dat bestand 103 de 65 goed Nederlandse tweeletterwoorden bevat in alfabetische volgorde, achter elkaar en zonder spaties. Schrijf een programma dat uitzoekt en afdrukt hoeveel Nederlandse woorden er onder de 100 gegeven tweelettercombinaties zitten.
c) Vraagstukken a) en b) zijn ook te combineren tot één programma dat alleen het laatst gevraagde aantal afdrukt.

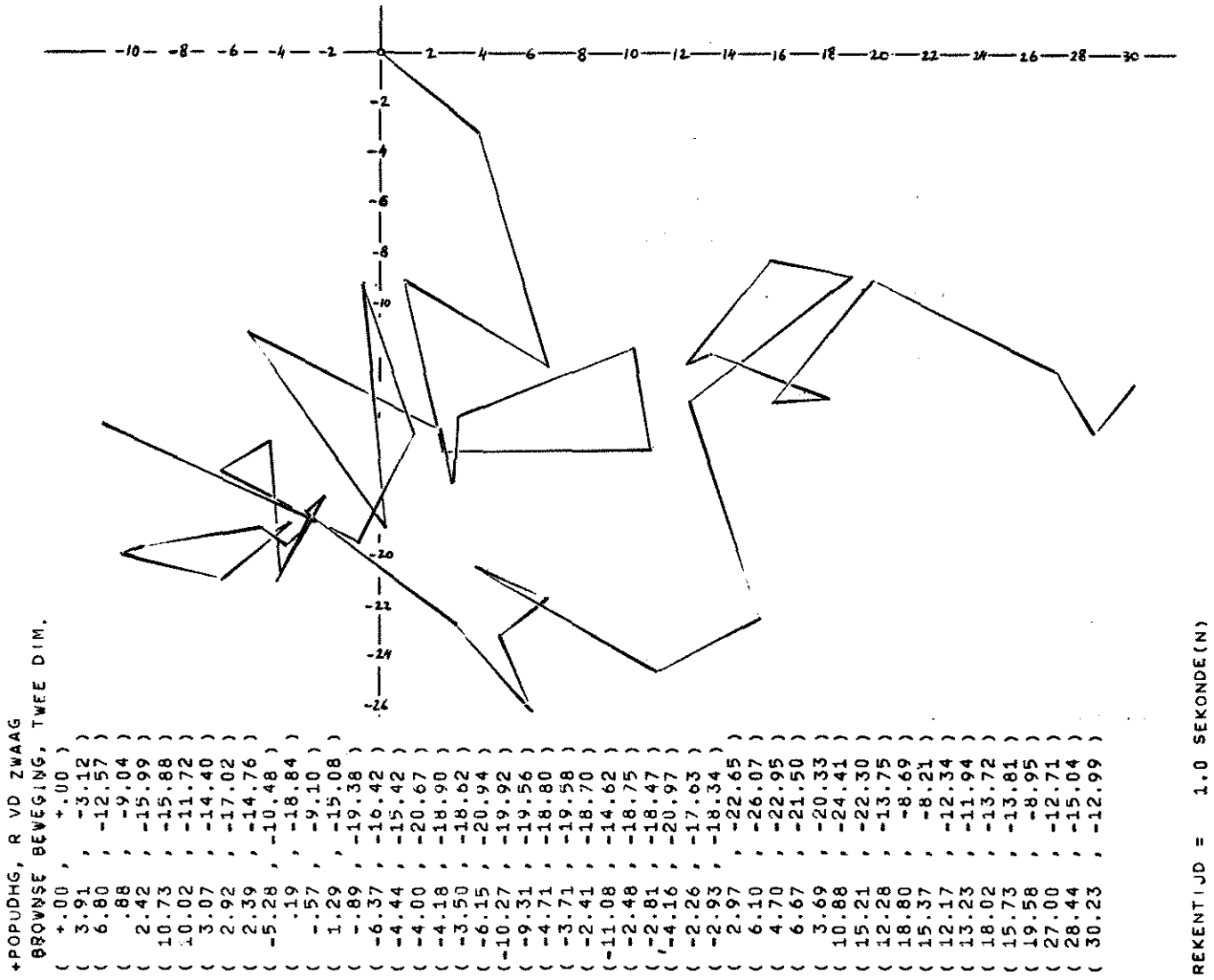
2.4 ENKELE PROGRAMMA'S

Behalve extra opgaven, opgesomd in de vorige paragraaf, werden door experimenterende docenten ook wel leerlingenprogramma's ingestuurd, die opvielen door hun oplossing. Hieruit is een vrij willekeurige keus gemaakt, die U hieronder kunt aantreffen.

2.4.1 Brownse beweging

Een programma voor de simulatie van een Brownse-beweging.
Het 'deeltje' beweegt zich rechtlijnig over een afstand r, waarbij r een toevalsgetal is met $0.00 \leq r \leq 10.00$. De richting fi (radialen) na een botsingsresultaat is een toevalsgetal $0.00 \leq fi \leq 6.3842$.
Onder het programma ziet U de posities in coördinaten en daarvan een met de hand getekende weergave.

```
        START
        TEKST:="+POPUDHG, R VD ZWAAG"
        NR
        TEKST:=" BROWNSE BEWEGING, TWEE DIM."
        NR
        T:=0
        X:=0
        Y:=0
1       T:=T+1
        SCHR|JESYM:=78
        SCHR|JE(2,2):=X
        SCHR|JESYM:=71
        SCHR|JE(3,2):=Y
        SCHR|JESYM:=79
        ALS T=50 DAN3 ANDERS2
2       R:=GOK(0,1000)/100
        FI:=GOK(0,62831)/10000
        X:=X+R*COS(FI)
        Y:=Y+R*SIN(FI)
        NR
        NAAR1
3       KLAAR
```



2.4.2. Iteratieve nulpuntbepaling

De functie $f:x \to x^2-x-1$ heeft een nulpunt tussen x=1 en x=2, want $f(1) < 0$ en $f(2) > 0$ en f is continu. Het volgende programma bepaalt het nulpunt, nauwkeurig tot het vierde cijfer achter de punt, met de zogenaamde halveringsmethode. Het nulpunt ligt steeds in het interval [a,b]. Bij iedere stap wordt de lengte van dit interval gehalveerd, door één van de grenzen van het interval te vervangen door $\frac{a+b}{2}$. De output bestaat uit een lijstje van deze intervalmiddens.

```
        START
        TEKST:="BENADERING VAN EEN NULPUNT VAN X2-X-1"
        WR
        A:=1
        B:=2
1       X:=(A+B)/2
        SCHR.JE(1,4):=X
        WR
        ALS B-A >LITEIL DAN 2 ANDERS 5
2       FX:=X*X-X-1
        ALS FX<0 DAN 3 ANDERS 4
3       A:=X
        JAAR 1
4       B:=X
        JAAR 1
5       KLAAR
```

```
BENADERING VAN EEN NULPUNT VAN X2-X-1
 1.5000
 1.7500
 1.6250
 1.5625
 1.5937
 1.6094
 1.6172
 1.6211
 1.6191
 1.6182
 1.6177
 1.6179
 1.6180
 1.6180
 1.6180

REKENTIJD =<15>SECONDE(N)
```

2.4.3. Een grafiek op de regeldrukker

Onderstaand programma geeft een "grafiek" van $f:x \to x^2-11x+24$, met vertikaal staande X-as. Neemt men één positie op een regel en de regelafstand als horizontale respectievelijk vertikale eenheid, dan ontstaat een vertekend beeld. Hier is voor de horizontale eenheid vijf posities en voor de vertikale drie regels genomen, wat een redelijk "vierkant" beeld geeft.
Het aangeven van de roosterpunten op de as gaat hier wat onhandig (en onjuist voor $x > 9$).

```
GRAFIEK VAN X2-11X+24

                                        1
                                        |
                                        |                                   *
                                        2                       *
                                        |           *
                                        |       *
                                        *
                                *       |
                            *           |
                        *               4
                    *                   |
                *                       |
            *                           5
        *                               |
        *                               |
            *                           6
                *                       |
                    *                   |
                        *               7
                            *           |
                                *       |
                                        *
                                        |       *
                                        |           *
                                        9                       *
                                        |                                   *
                                        |
```

```
   START
   TEKST:="GRAFIEK VAN X2-11X+24"
   NR
   N:=2
 2 N:=N+1
   NR
   X:=N/5
   F:=X*X-11*X+24
   F:=F*5
   ALS F<0 DAN 3 ANDERS 6
 3 SPATIE(F+49)
   SCHRIJF(8,0):=82
   SPATIE(-F-1)
   ALS X=G(X) DAN 4 ANDERS 5
 4 SCHRIJF(8,0):=X
   NAAR 2
 5 SCHRIJF(8,0):=13
   NAAR 2
 6 ALS F=0 DAN 7 ANDERS 8
 7 SPATIE(49)
   SCHRIJF(8,0):=82
   NAAR 2
 8 SPATIE(49)
   ALS X=G(X) DAN 9 ANDERS 10
 9 SCHRIJF(8,0):=X
   NAAR 11
10 SCHRIJF(8,0):=17
11 SPATIE(F-1)
   SCHRIJF(8,0):=82
   ALS X>0.5 DAN 12 ANDERS 2
12 KLAAR
```

2.4.4

Een programma voor de berekening van $\lim\limits_{n\to\infty} (1 + \frac{1}{n})^n$

n neemt hierbij achtereenvolgens de waarden, 2,4,8,16,....2^q aan.

Het programma stopt op het criterium, dat twee achtereenvolgende antwoorden gelijk zijn tot 8 cijfers achter de komma.

```
      START
      oudea := 0
      n := 1
2     n := 2 × n
      a := 1 + 1/n
      m := 1
3     a := a × a
      m := 2 × m
      ALS m = n DAN 4 ANDERS 3
4     SCHRIJF(8,0) := n
      SCHRIJF(2,8) := a
      NR
      ALS a-oudea < 0.00000001 DAN 6 ANDERS 5
5     oudea := a
      NAAR 2
6     KLAAR
```

2	2.25000000
4	2.44140625
8	2.56578451
16	2.63792850
32	2.67699013
64	2.69734495
128	2.70773902
256	2.71299162
512	2.71563200
1024	2.71695573
2048	2.71761848
4096	2.71795008
8192	2.71811594
16384	2.71819889
32768	2.71824035
65536	2.71826109
131072	2.71827146
262144	2.71827664
524288	2.71827923
1048576	2.71828182
2097152	2.71828182

3 Het 'ECOL' _ rapport

In het nu volgende hoofdstuk vindt U een volledige beschrijving van de programmeertaal ECOL. Iets te volledig zelfs, want in enkele opzichten is men vooruit gelopen op de situatie. Met name het FUN-mechanisme en de MET-opdracht zijn op het moment van het verschijnen van dit boek nog niet in werkelijkheid toe te passen.

Dit hoofdstuk is ook onafhankelijk van dit docentenboek, als rapport, verschenen. Daarom wijkt de paragraafnummering af van die van overige hoofdstukken. Zelfs is dit hoofdstuk onderverdeeld in zes hoofdstukken, aangegeven met Romeinse cijfers. Hieronder vindt U een inhoudsopgave. Aan het eind van dit hoofdstuk vindt U ook een index om het zoeken te vergemakkelijken.

- INHOUD -

I Taaldefinitie, vertaling, executie
1 Taaldefinitie
2 Vertaling
3 Executie
II Globale vorm van ECOL programma's
4 Structuur van een ECOL programma
5 Labels, kommentaar, spaties
5.1 Labels
5.2 Kommentaar
5.3 Spaties
III Context-regels en compiler aktie
6 Geheugenplaatsen, waarde, adres, descriptor
7 Namen
8 Labels en sprongopdrachten
IV Executie
9 Inleidende opmerkingen
10 Uitdrukkingen
10.1 Getal
10.2 Variabele
10.3 Fun-roep
10.4 Waarden van uitdrukkingen; prioriteiten
10.5 Volgorde bepaling primitiva
10.6 Waarde van aMb
11 Toekenning
12 Sprongopdrachten
13 RIJ-reservering
14 Cyclus
15 Lege opdracht
16 Invoer
16.1 LEES, LEESSYM
16.2 BESTAND (x,x,x)
16.3 BESTAND (x)
17 Uitvoer
17.1 SCHRIJF (x,x)
17.2 SCHRIJF
17.3 SCHRIJM
17.4 SCHRIJFSYM
17.5 TEKST
17.6 SPATIE(x)
17.7 SPATIE
17.8 NR(x)
17.9 NR
17.10 NP
17.11 BESTAND (x,x,x)
17.12 BESTAND (x)
17.13 Opmerkingen uitvoer
18 Standaardfunkties
V SUB en FUN
19 Inleiding
20 SUB-beschrijving
21 SUB-roep
22 Informele inleiding FUN mechanisme
23 FUN-beschrijving
24 FUN-roep
25 Recursiviteit
VI De ECOL syntax

DEFINIEREND RAPPORT VAN E C O L, EEN PROGRAMMEERTAAL VOOR HET VOORTGEZET ONDERWIJS

ECOL, samentrekking van Educational COmputer Language, is de programmeertaal die o.a. wordt gebruikt in de leergang "Computerkunde 1 en 2 voor a.v.o. en v.w.o." door C.A.Ch.Görts, S.G.van der Meulen, A.van der Sluis en J.R.Zweerus, uitgave Wolters-Noordhoff, Groningen 1970-1971.

Aan de opstelling van dit rapport werd, behalve door de vier auteurs van de genoemde leergang, meegewerkt door E.Lopez Cardozo, J.H.J.M.Truijens, G.A.Vonk.

I TAALDEFINITIE, VERTALING, EXECUTIE

1. Taaldefinitie

ECOL is een taal om programma's voor een computer te formuleren. Bij het beschrijven van zo'n taal moet men enerzijds zeggen hoe correcte teksten in deze taal er uit zien, anderzijds wat voor computer-akties deze teksten beschrijven.
Hoe een correcte ECOL-tekst er uitziet wordt voor een belangrijk deel vastgelegd in een syntax (zie VI). Omdat deze, ter wille van de eenvoud, weinig formeel is gehouden, is hij in feite wat te ruim, en zijn verdere inperkingen nodig. Deze worden in dit rapport gegeven in zgn. context-regels.
Verwijzingen naar syntax-regels worden gegeven door [ ], verwijzingen naar context-regels door C...(C 8.1 bijv. staat in paragraaf 8).

In het rapport wordt voortdurend naar de syntax verwezen, en de lezer wordt dan ook uitgenodigd te beginnen met de inleiding tot de syntax te lezen.

2. Vertaling

Bij de computerverwerking van ECOL-teksten zijn twee fasen te onderscheiden, nl. de vertaalfase en de executiefase.
In de vertaalfase wordt de ECOL tekst door een voor ECOL gemaakt vertaalprogramma ("compiler") regel na regel, van boven naar beneden, en per regel van links naar rechts, gelezen en vertaald naar interne machine-opdrachten. We duiden deze leesvolgorde verder aan als tekstuele volgorde.
Tijdens deze actie ziet de compiler voortdurend uit naar overtredingen van syntax en context-regels. Wanneer hij zo'n fout ontdekt wordt deze gesignaleerd, zoveel mogelijk met regelnummer en aanduiding wat er fout is. Na het ontdekken van een fout gaat de compiler rustig verder met lezen om te trachten nog meer fouten te ontdekken, zodat de auteur van het programma in één keer zoveel mogelijk fouten aan de weet komt. Evenwel kan een fout tot gevolg hebben dat foutmeldingen gegeven worden betreffende andere regels ECOL-tekst die in feite goed zijn.
Het is echter duidelijk dat als er eenmaal een fout is geconstateerd, niet tot executie van het programma kan worden overgegaan. Ook in de beschrijving van ECOL op de volgende pagina's duikt de compiler af en toe op. Strikt genomen is dit overbodig. Het kan echter bijdragen tot goed begrip van wat er gebeurt.

3. Executie

Als het programma bij vertaling "correct" bevonden is, wordt tot uitvoering (executie) overgegaan, d.w.z.: de computer gaat nu doen, wat de programmeur in zijn (taalkundig correct) programma voorschreef. Dit hoeft evenwel niet te betekenen, dat er nu ook gebeurt wat de programmeur beoogde.
Er kan bijvoorbeeld iets uit een GP worden gehaald, voordat er iets in is gestopt; er kunnen RIJ-grenzen worden overschreden; er kan gevraagd worden een getal te lezen terwijl er geen getallen meer zijn, enz. Allemaal fouten die een vertaal-programma onmogelijk kan voorzien. We noemen dit executie-fouten; ook hiervan worden er vele gesignaleerd, weer met vermelding van regelnummer.
Meestal heeft een executie-fout tot gevolg dat het programma wordt afgevoerd, d.w.z.: niet verder wordt geëxecuteerd.

II GLOBALE VORM VAN ECOL-PROGRAMMA'S

4. Structuur van een ECOL-programma

Uit de syntax (zie [20]) blijkt dat een ECOL-programma er als volgt uitziet.

```
FUN---
-----      }---- [opdrachten] (zie [23]), dus
-----               geen RIJ-reserv., geen
FUN:=---            SUB-beschr., maar ook geen
                    SUB-roep (zie 23)
FUN---
-----
-----
FUN:=---
START
-----      }---- [programmadelen] dus
-----               [opdrachten] en
-----               [RIJ-reserveringen] en
-----               [SUB-beschrijvingen]
KLAAR
```

FUN-beschrijving
FUN-beschrijving
} funktiedeel (mag ook ontbreken)

} hoofdprogramma

Het funktiedeel, waarover in de leergang Computerkunde niet gesproken wordt,maakt het mogelijk, onafhankelijk van het hoofdprogramma een of meer rekenschema's (bestaande uit gewone ECOL-opdrachten "klaar te zetten", die dan door het hoofdprogramma kunnen worden gebruikt. Voor nadere verduidelijking zie 22 en volgende.

De indeling van een programmatekst in regels wordt in hoge mate door de syntax voorgeschreven. Overgang op een nieuwe regel programmatekst is in ECOL dan ook zeer betekenis-dragend. Zo mag men niet twee korte opdrachten op één regel zetten, of een lange formule op een volgende regel voortzetten.

5. Labels, kommentaar, spaties

5.1. Labels

Uit de syntax blijkt (zij het op een nogal ingewikkelde wijze), dat elke regel programma-tekst mag beginnen met hoogstens één [label], dat is een geheel getal zonder teken (zie [14]), eventueel gevolgd door dubbelpunt (zie [15]).

5.2. Kommentaar

Uit de syntax blijkt dat achteraan elke regel tekst een puntkomma, eventueel gevolgd door willekeurige symbolen mogen komen. Wat er achter de puntkomma staat is het zgn. kommentaar. De enige functie ervan is dat dit bij het afdrukken van de programmatekst mee wordt afgedrukt. Verder wordt er niets mee gedaan. Het dient om de tekst voor de menselijke lezer wat leesbaarder te maken.

In een programmatekst mogen op willekeurige plaatsen regels voorkomen die hetzij helemaal leeg zijn of alleen een label en/of kommentaar bevatten. Dit mag omdat [86] een "lege" FUN-beschrijving toelaat en [23] een "lege" opdracht. Wegens C 8.1 zijn daarom de labels op dergelijke regels tussen twee "echte" FUN-beschrijvingen geheel willekeurig, terwijl op dergelijke regels binnen een FUN-beschrijving of binnen het hoofdprogramma de labels moeten verschillen van de overige labels in de betreffende FUN-beschrijving resp. het hoofdprogramma.

5.3. Spaties

In ECOL-teksten mogen bijna overal spaties worden ingelast of weggelaten, zonder dat daardoor de betekenis, dat wil zeggen: de bij executie geproduceerde uitvoer, of de afgegeven foutmeldingen, verandert. De uitzonderingen staan in de inleiding tot de syntax:

a) g e e n spaties mogen worden ingelast in hoofdletter-woorden (zoals START, LEES, ALS etc.);
b) g e e n spaties mogen worden toegevoegd of weggelaten in een rijtje symbolen tussen aanhalingstekens, zoals achter "TEKST :=" kan voorkomen (dat is niet fout, maar daardoor verandert de geproduceerde uitvoer en dus de betekenis van het programma);
c) tenminste 1 spatie m o e t er zijn wanneer hoofdletter-woorden op elkaar volgen.

Voorbeeld 5.1

```
     7                          ; demonstratietekst voor labels, kommentaar en spaties.
     8   START
     7:                         ; deze nieuwe label 7 is toegestaan (zie 5.2 en C 8.1).
15   2                          ; dit is label 152.
     5 : x:   = LEES            ; lelijk "wordt-teken" maar wel goed.
         y := LEE S             ; fout, spaties in hoofdletterwoord.
     9   ALS SIN(x) > 0.5 DAN 5  ; goed; spatie tussen ALS en SIN. In plaats van SIN(x) had er ook
                                ; mogen staan SIN (   x ).
     5   TEKST := "jan hein"    ; als men de spatie tussen jan en hein weglaat wordt er afgedrukt
                                ; janhein.
     8   KLAAR                  ; fout, label 8 was al uitgegeven in het hoofdprogramma; verder
                                ; mag op de regels volgend op KLAAR geen kommentaar meer komen.
```

III CONTEXT-REGELS EN COMPILER-AKTIE

6. Geheugen-plaatsen, waarde, adres en descriptor

Zoals in "Computerkunde I" staat, is een computer een apparaat dat getallen snel kan optellen, aftrekken, vermenigvuldigen, delen en vergelijken. We weten dat die getallen in het geheugen zijn opgeslagen; hoe dat technisch is gerealiseerd, doet hier niet ter zake. Wel van belang voor ons is:

> Een plaats in het geheugen waar een getal kan worden opgeborgen, noemen we een Geheugen-Plaats (afgekort: GP).
> Alle GP's in het geheugen zijn genummerd, het nummer van een GP heet zijn adres.

Het onderscheid tussen het getal in de GP (ook wel de inhoud of de waarde van de GP genaamd) en het adres van de GP speelt op verschillende plaatsen een rol in deze beschrijving van ECOL.

Aan elke [variabele] van de vorm [GP] (zie syntax [32]) in het hoofdprogramma wordt door de compiler een GP (dus een adres) toegekend.
Bij RIJ-en ligt de zaak ingewikkelder. Als men schrijft

RIJ(m:n)r of RIJSYM(m:n)r

dan krijgt men bij executie n-m+1 GP's ter beschikking:

r(m), r(m+1),----------r(n-1), r(n)

en elk van deze GP's heeft natuurlijk een adres. Maar de compiler weet niet welke adressen dit zijn omdat hij niet weet welke waarden m en n tijdens executie zullen hebben. Door sprongopdrachten kan het voorkomen dat een RIJ-reservering meerdere keren geëxecuteerd wordt met telkens andere onder- en bovengrens. Daarom zet de compiler voor elke rij een descriptor klaar, waarin tijdens executie geadministreerd kunnen worden
a) het begin-adres van de RIJ (dat is het adres van het eerste element)
b) de waarde van de onder-grens en de boven-grens,
c) gegevens die verder van belang kunnen zijn (zoals het al of niet gereserveerd zijn van de RIJ)
en hij vertaalt de RIJ-reservering in een aantal opdrachten die bij executie de nodige gegevens in de descriptor noteren. Een RIJ-reservering is dus wel degelijk een aktie.
Uit de syntax (zie [55]) blijkt nog dat onder- en bovengrens van een RIJ-reservering willekeurige [uitdr]'s (= uitdrukkingen, zie [34]) mogen zijn, zodat dus bijv. RIJ (p+1 : rxs)q toegestaan is.

7. Namen

Uit de syntax-regels [32], [33], [72] en [82] blijkt dat in een ECOL-programma verschillende soorten namen kunnen voorkomen:

[GP], [GP-rij], [SUB-naam] en [FUN-naam]

De schrijver van het programma weet natuurlijk wat er op elke plaats met een zekere [naam] wordt bedoeld, maar de lezer (in casu de compiler) weet dit niet. Vandaar de volgende context-regels

waaruit blijkt dat bij het in tekstuele volgorde lezen van een ECOL-tekst van elke [naam] die men ontmoet, direkt bekend is wat voor soort [naam] het is, ook als die [naam] nog niet eerder is voorgekomen. We volstaan hieronder met de context-regels voor het hoofdprogramma; de overeenkomstige regels voor het funktie-deel vindt men in 23.

Context-regels voor het hoofdprogramma

C7.1 Een en dezelfde [naam] heeft overal in het hoofdprogramma dezelfde betekenis.

Gevolgen

a) Een [naam] mag niet gebruikt worden in meer dan één der betekenissen [GP], [GP-rij], [SUB-naam], [FUN-naam].

b) Als een [naam] een
[GP] is stelt hij overal dezelfde GP voor
[GP-rij] is stelt hij overal dezelfde RIJ voor
[SUB-naam] is verwijst hij overal naar dezelfde [SUB-beschrijving]
[FUN-naam] is verwijst hij overal naar dezelfde [FUN-beschrijving].

c) Een [naam] mag niet meer dan eens op de plaats van [GP-rij], [SUB-naam] of [FUN-naam] in een [RIJ-reservering], [SUB-beschrijving] of [FUN-beschrijving] optreden.

C7.2 Een [naam] is een [FUN-naam] als hij eerder in de programmatekst op de plaats van [FUN-naam] in een [FUN-beschrijving] is opgetreden (zie [86]).

C7.3 Een [naam] is een [GP-rij] als hij eerder in de programmatekst op de plaats van [GP-rij] in een [RIJ-reservering] is opgetreden (zie [55]).

Gevolg

In de programmatekst mag een element van een zekere RIJ pas voorkomen nadat de reservering van deze RIJ verschenen is.

C7.4 Een [naam] is een [SUB-naam] als hij optreedt als [SUB-roep] of op de plaats van [SUB-naam] in een [SUB-beschrijving] (zie [70] en [71]). Een [SUB-naam] moet ook eenmaal op de plaats van [SUB-naam] in een [SUB-beschrijving] optreden.

Gevolg

In de programmatekst mag een [SUB-roep] voorkomen voordat de betreffende [SUB-beschrijving] is verschenen. N.B. Een [SUB-roep] is in de programmatekst gemakkelijk te herkennen: het is een [naam] die als zelfstandige [opdracht] optreedt (zie [70] en voorbeeld 7, regels 3 en 7).

C7.5 Een [naam] is een [GP] in alle overige gevallen.

Deze regels stellen bij een correct programma de compiler in staat alle optredende [namen] thuis te brengen. Wanneer tegen deze regels gezondigd wordt volgt een foutmelding, maar het kan nu verder voor de compiler wel moeilijk worden. Als een [naam] bijv. eenmaal verschenen is als [GP-rij] in een [RIJ-reservering] en vervolgens ook nog eens verschijnt zonder gevolgd te worden door een openingshaakje (dus in feite als [GP] of als [SUB-roep]), dan weet de compiler niet meer wat er nu eigenlijk bedoeld wordt (dit is een der oorzaken van het reeds in 2 gememoreerde verschijnsel dat een fout tot gevolg kan hebben dat "goede" stukjes tekst fout gemeld worden).

Voorbeeld 7.1

```
        FUN f(x)    ; hiermee is "f" een [FUN-naam]
        -----
        -----
        FUN :=w

        START       ; hier begint het hoofdprogramma
        a := LEES   ; hiermee is "a" een [GP]
        RIJ(1:9)a   ; FOUT! "a" is al in gebruik genomen als [GP]
        RIJ(1:a)r   ; in orde: "r" is hiermee een [GP-rij]
        i :=a*3     ; hiermee is "i" een [GP]
        y :=r(i)    ; hiermee is "y" een [GP], "r" wordt correct gebruikt als [GP-rij]
        y :=u(i)    ; FOUT! "u" is nog niet verschenen in een [RIJ-reservering] en ook niet als
                    ; [FUN-naam]
        SUB klus    ; hiermee is "klus" een [SUB-naam]
        -----
        -----
        END
        y := klus   ; FOUT! "klus" wordt hier gebruikt als [GP], maar is al verschenen als
                    ; [SUB-naam]
    3   klus        ; in orde, een reeds beschreven SUB wordt geroepen
    7   klos        ; GOED! "klos" verschijnt als een [SUB-roep], merk op dat het onmogelijk een
                    ; [GP] kan zijn, "klos" is nu een [SUB-naam] en de compiler verwacht v ó ó r
                    ; KLAAR de beschrijving van "klos"
        y :=klos    ; dat is dus FOUT
        SUB klos    ; daar is hij dan
        -----
        END
        y :=f       ; FOUT! "f" is al verschenen als [FUN-naam] van een functie met een parameter
        y :=f(a)    ; in orde, "f" wordt gebruikt in een [FUN-roep]
        RIJ(1:a)f   ; FOUT! "f" is al verschenen als [FUN-naam] en staat niet meer ter beschikking
                    ; als [GP-rij]
        f           ; FOUT! "f" is al verschenen als [FUN-naam] van een functie met een parameter
        KLAAR       ; dit programma komt nooit aan executie toe
```

8. Labels en sprongopdrachten

Zoals gezegd mag men elke programma-regel een label geven. Men is echter niet geheel vrij in de keuze van deze labels. Er geldt:

C8.1 Meerdere regels mogen hetzelfde getal als [label] hebben, mits maar alle labels in het hoofdprogramma verschillend zijn, evenals alle labels van een FUN-beschrijving (de labels in een SUB-beschrijving moeten dus ook verschillen van de labels in andere SUB-beschrijvingen en die in de rest van het hoofdprogramma).

Op deze regel is al geanticipeerd in 5.2 en voorbeeld 5.1.

Sprongopdrachten moeten verwijzen naar labels vooraan programmaregels. Ook hier is men weer niet geheel vrij. Er geldt:

C8.2 Een sprongopdracht die in het hoofdprogramma staat maar niet in een [SUB-beschrijving] mag alleen verwijzen naar een [label] in het hoofdprogramma maar niet in een [SUB-beschrijving].

C8.3 Een sprongopdracht die staat in een SUB- of FUN-beschrijving mag alleen verwijzen naar een label in dezelfde SUB- of FUN-beschrijving.

C8.4 Een sprongopdracht mag nooit verwijzen naar een [label] voor START, of voor een SUB- of FUN-beschrijving .

Als gevolg hiervan kan men een SUB of FUN alleen binnenkomen door de voordeur met een SUB- of FUN-roep, en alleen verlaten door de achterdeur met END resp. FUN:=---

IV EXECUTIE

9. Inleidende opmerkingen

Wanneer de compiler een aangeboden ECOL-tekst correct heeft bevonden en heeft vertaald, kan het vertaalde programma in executie worden genomen, dat wil zeggen: worden uitgevoerd.

Bij executie van een ECOL-programma wordt begonnen met de [aktie] die tekstueel volgt op START. Na elke [aktie] wordt de tekstueel volgende [aktie] uitgevoerd, behalve bij sprongopdrachten, SUB- en FUN-roepen. Aangezien [SUB-beschrijvingen] geen [akties] zijn vallen zij buiten de executie-volgorde. Men mag dus rustig tussen twee [akties] een [SUB-beschrijving] zetten: die wordt bij executie overgeslagen. Men ziet dat de syntactische term [aktie] juist gekozen is: een [aktie] doet inderdaad iets, terwijl een [SUB-beschrijving] een rekenschema is dat door een [SUB-roep] geaktiveerd moet worden.

In dit hoofdstuk wordt in meer detail besproken hoe een correct ECOL-programma wordt geëxecuteerd. We beginnen daarbij met te bespreken hoe waarden worden berekend uit formules. Daarna komen de verschillende akties ter sprake.

Afronden

In het volgende wordt nogal eens gesproken over afronding. Wanneer voor een willekeurig getal a en een geheel getal g geldt $g-\frac{1}{2} \leq a < g+\frac{1}{2}$ dan verstaan we onder het afronden van a het vervangen van a door g.

10. Uitdrukkingen

In [34] t/m [38] wordt recursief (voor de betekenis van dit woord zie inleiding syntax) vastgelegd wat een [uitdr] allemaal kan zijn. Zinvolle samenstellingen van getallen, variabelen en FUN-roepen (de zgn. primitiva) met de operatoren + - × / en M (machtsverheffing) en haakjesparen zijn toegestaan, mits geen twee van deze operatoren onmiddellijk op elkaar volgen. Dus niet a × -b, ook niet a × -6, maar wel a × (-b). Enige uitzondering: men mag i.p.v. M ook ×× schrijven.

Ingewikkeld voorbeeld:

r(i+j×k) := ((a+b)/(a-b) - u×vM2) - SIN(u+v)×COS(u-v) + r(i-j×k)

Hier volgt eerst informatie over de primitiva en hun waarden.

10.1 Getal

Volgens [38] mag een getal genoteerd worden als geheel getal of als geheel getal gevolgd door punt (deze punt heeft dan geen betekenis) of als geheel getal gevolgd door punt gevolgd door decimale fractie.

Voorbeelden: 23 123. 3.47 Niet toegestaan is .47

10.2 Variabele

Er zijn twee soorten variabelen: [GP] of [GP-rij]([uitdr]), de laatste noemen we ook wel rij-element en de [uitdr] achter de [GP-rij] de index-expressie.
De waarde van een variabele is altijd de waarde die er bij executie het laatst aan is toegekend en die waarde is altijd een getal. Zoals bekend heeft een variabele behalve een waarde ook een adres.
Een GP-rij heeft evenveel adressen (en dus waarden) als is gegeven door ondergrens en bovengrens (zie 6). Het adres van een rij-element volgt uit de index, dat is de afgeronde waarde van de index-expressie. Er wordt gecontroleerd of deze index niet onder de ondergrens, of boven de bovengrens ligt; in dat geval volgt een foutmelding en wordt het programma afgevoerd. De waarde van een rij-element is het getal op het aldus aangewezen adres.

Voorbeeld

```
a := 2.3
b := 2.6
c := WRTL(2)          ; WRTL betekent vierkantswortel
r(a) := r(b) + r(c)   ; dat is nu hetzelfde als :  r(2) := r(3) + r(1)
```

Aan de schrijvers van compilers wordt aanbevolen ervoor te zorgen dat bij executie een foutmelding gegeven wordt zodra de waarde van een variabele gebruikt wordt voordat deze variabele in de loop van dit programma een waarde heeft gekregen.

10.3 FUN-roep

Bij een FUN-roep (zie ook hoofdstuk V) wordt het programma van de FUN-beschrijving doorlopen met de meegegeven aktuele(n). Een FUN-beschrijving eindigt altijd met een opdracht "FUN :=[uitdr]". De waarde van [uitdr] is tevens de waarde die de FUN -roep aflevert.

10.4 Waarden van uitdrukkingen; prioriteiten

Wanneer de waarden der primitiva bekend zijn volgt de waarde van de uitdrukking door de operatoren + - × / M in de goede volgorde uit te voeren. Wat die goede volgorde is volgt uit onderstaande prioriteitsregels (die behalve ten aanzien van de deling overeenstemmen met Meneer Van Dalen Wacht Op Antwoord).

Omdat uitdrukkingen in een [vergelijking] (zie [50]) ook nog voorkomen in combinatie met vergelijkers, EN en OF geven we hier meteen een overzicht van al deze tekens in volgorde van hoogste naar laagste prioriteit; wat op dezelfde regel staat heeft gelijke prioriteit.

M
× /
+ -
< ≤ = ≠ ≥ >
EN
OF

Gelijke prioriteit betekent dat de operatoren in tekstuele volgorde aan de beurt komen.

Voorbeelden van gelijke prioriteiten

a - b + c	betekent	(a - b) + c	
a / b × c	betekent	(a / b) × c	N.B. Wijkt af van Nederlands gebruik.
a M b M c	betekent	(a M b) M c	

Hogere prioriteit betekent dat een operator die tekstueel later komt dan een van lagere prioriteit, toch eerder wordt uitgevoerd.

Voorbeeld van verschillende prioriteiten

ALS a+b×c < pMq OF x = y-z EN u ≥ v/w DAN 12 ANDERS 13

betekent

ALS ((a+(b×c)) < (pMq)) OF ((x = (y-z)) EN (u ≥ (v/w))) DAN 12 ANDERS 13

Haakjesparen doorbreken uiteraard de prioriteitsregels: wat tussen haakjes staat moet eerst worden gedaan. Zodoende betekent a + b × c iets anders dan (a + b) × c.

10.5 Volgorde bepaling primitiva

Met de prioriteitsregels alleen is men er nog niet. Zoals het voorbeeld u := LEES / LEES laat zien kan de uitkomst nog afhangen van de volgorde waarin men de waarden van de primitiva bepaalt: als men eerst de eerste LEES uitvoert en daarna de tweede dan gebeurt er wat anders dan wanneer men eerst de tweede en daarna pas de eerste LEES uitvoert. Iets soortgelijks kan zich voordoen wanneer men FUN-roepen hanteert die de waarden der meegegeven [aktuele]n veranderen.

Afspraak

De primitiva in een [uitdr] worden van links naar rechts geëvalueerd. Bij een RIJ-reservering wordt eerst de [uitdr] voor de ondergrens en daarna die voor de bovengrens geëvalueerd.

Bij een [vergelijking] wordt eerst de eerste [uitdr] geëvalueerd, dan de tweede, dan (als er nog meer zijn, nl. bij het gebruik van EN en OF) de derde, etc.

10.6 Waarde van aMb

De waarde van aMb zal over het algemeen niet exact zijn. Voor niet te grote gehele waarden van a en niet-negatieve gehele waarden van b zal de uitkomst echter wel exact zijn.

11. Toekenning

Een toekenning (zie [30]) heeft de gedaante [variabele] :=[uitdr]. Bij executie wordt eerst het adres van de variabele in het linkerlid bepaald (het doeladres), daarna wordt de waarde van de uitdrukking in het rechterlid bepaald. Deze laatste waarde wordt opgeborgen in de door het doeladres aangewezen GP.

Voorbeeld

```
een := twee + drie
```

Het adres van de GP "een" wordt klaargezet als doeladres. De waarde van "twee" wordt opgehaald, de waarde van "drie" wordt daarbij opgeteld. Deze waarde wordt de nieuwe waarde van "een".

Voorbeeld

```
r(i+j×k) := a + b × WRTL(xM2 + yM2)
```

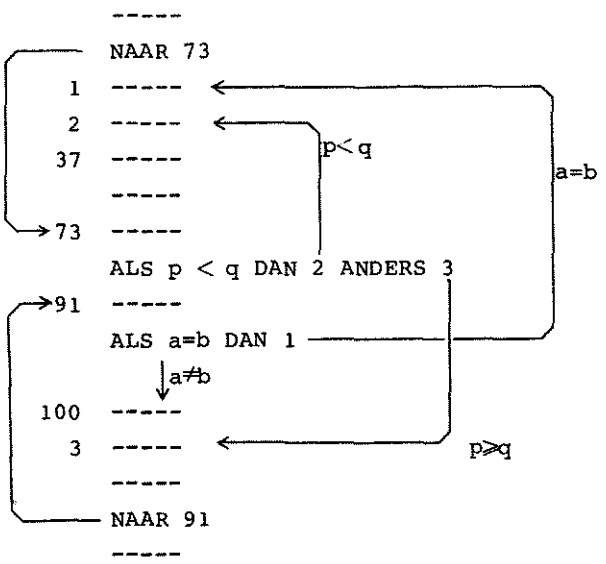
Eerst wordt de index-expressie i+j×k uitgerekend, de opgeleverde waarde wordt afgerond, dit is de index;
de index wordt op de grenzen uit de descriptor van "r" gecontroleerd, uit de index en het begin-adres van de rij wordt nu het adres berekend, het aldus berekende doeladres wordt klaargezet.

De waarde van het rechterlid wordt berekend uit de gegeven formule.

Deze waarde wordt opgeborgen in de GP waarvan het adres was klaargezet.

12. Sprong-opdrachten

Sprong-opdrachten dienen om bij executie een andere dan de tekstueel volgende aktie aan te wijzen als de vervolgens te executeren aktie. Een en ander blijkt voldoende uit onderstaand voorbeeld:



Men ziet dat de opdracht "ALS a=b DAN 1" hier hetzelfde doet als de opdracht "ALS a=b DAN 1 ANDERS 100" gedaan zou hebben.

Derhalve: bij een sprongopdracht wordt eerst de vergelijking achter ALS uitgewerkt. Zo'n vergelijking is WAAR of ONWAAR. Zie voor de betekenis van EN en OF onderstaand tabelletje.

De DAN-sprong wordt gedaan als de vergelijking WAAR is. Als de vergelijking ONWAAR is en er staat een ANDERS-sprong geschreven, dan wordt deze uitgevoerd, anders is de tekstueel volgende aktie de vervolgens geëxecuteerde.

We brengen nog in herinnering (zie C7.4) dat wel naar allerlei akties (dus o.a. naar een RIJ-reservering) gesprongen mag worden maar niet naar START of een SUB- of FUN-beschrijving.

p	q	p EN q	p OF q
W	W	W	W
W	O	O	W
O	W	O	W
O	O	O	O

"W" is "WAAR"
"O" is "ONWAAR"

Voor een belangrijk neveneffect van sprongopdrachten zie 13 (het ongereserveerd maken van rijen).

13. RIJ-reservering

De executie van een RIJ-reservering betekent dat de waarden van onder en bovengrens worden uitgerekend, afgerond, en in de descriptor (zie 6) worden ingevuld. De aldus verkregen waarde van de bovengrens mag niet kleiner zijn dan die van de ondergrens, anders volgt een foutmelding. Vervolgens wordt in het geheugen van de computer het verlangde aantal GP's gereserveerd:

```
m := LEES
n := LEES
RIJ(m:n)r      ; in orde als m ≤ n, anders FOUT-melding, er wordt ruimte gereserveerd voor
               ; n - m + 1  GP's
```

Men zie in dit verband nog de opmerkingen over volgorde in 10.6.

In plaats van RIJ(m:n)r mag men ook schrijven RIJSYM(m:n)r. Het enige verschil is dat men dan in de elementen van de rij uitsluitend gehele getallen tussen 0 en 99 (0 en 99 inbegrepen) mag onderbrengen. Deze faciliteit kan nuttig zijn bij het werken met zeer lange rijen kleine getallen, omdat de computer dergelijke RIJSYM's zuiniger in het geheugen kan opbergen dan gewone RIJen.
Als men een waarde wil toekennen aan een element van een RIJSYM dan wordt deze waarde eerst afgerond. Als de afgeronde waarde niet een der getallen 0 t/m 99 is volgt een foutmelding, en wordt de executie gestaakt.

In verband met sprong-opdrachten is het volgende van veel belang:

a) Voordat een programma wordt geëxecuteerd wordt ervoor gezorgd dat in de descriptoren van a l l e rijen is aangetekend, dat deze "ongereserveerd" zijn.

b) Wanneer men door een of andere aktie een waarde wil toekennen aan of tevoorschijn halen uit een "ongereserveerde" rij, volgt o p d a t m o m e n t tijdens executie een fout-melding.

c) Bij een "terugspringende" sprongopdracht (dat is een sprongopdracht die verwijst naar een plaats eerder in de programmatekst dan waar de sprongopdracht zelf staat) worden alle rijen, waarvan de reservering voorkomt in het stuk programmatekst waarover werd teruggesprongen, "ongereserveerd" verklaard en opgeruimd, d.w.z. de waarden van de elementen gaan verloren, en komen bij een later opnieuw reserveren van deze RIJ niet weer terug. (N.B. Als men "vooruit" springt over een RIJ-reservering dan was deze rij op grond van regels a en c al ongereserveerd vóór de sprong, en is dat nog steeds na de sprong omdat de betreffende reservering niet is uitgevoerd).

Voorbeeld

```
        -----      ; de rijen p en q zijn nog ongereserveerd
   37 -----
        NAAR 15    ; deze sprong laat rij p ongereserveerd
        RIJ(m:n)p  ; nu wordt rij p gereserveerd
   73 -----
   15   RIJ(a:b)q  ; nu wordt rij q gereserveerd
   97 -----
        -----      ; bij deze sprong wordt q opgeruimd
        -----
        -----      ; bij deze sprong worden p en q opgeruimd
        -----
        -----      ; bij deze sprong wordt niets opgeruimd
        -----
```

14. Cyclus

Volgens de syntax-regel [60] heeft een cyclus de gedaante

MET [stap], i := [start], [stop]
[programmadeel]^n
[nummer] HERHAAL

(uiteraard mag bij het gebruik van de cyclus in een programmatekst ook voor MET een label staan, terwijl in [programmadeel] het mogen plaatsen van een label al besloten ligt, zie [21], [23] en [71]).

Hierin is [stap], [start] en [stop] ieder voor zich een [uitdr] , dat wil dus zeggen: mogelijk een gecompliceerde uitdrukking met veel operanden, maar in de praktijk waarschijnlijk meestal niet gecompliceerder dan een [getal] of een heel simpele [formule].
Verder is "i" een [GP].

Losjes gezegd is de betekenis van de [cyclus] dat [programmadeel]^n een aantal malen wordt uitgevoerd met telkens een andere waarde van i, te beginnen met de waarde van [start], oplopend met de waarde van [stap] en eindigend bij de waarde van [stop].

Voorbeelden

```
MET 2, i := 1 , 7 }
MET 2, i := 1 , 8 }  [programmadeel]^n  wordt uitgevoerd met achtereenvolgens i = 1,3,5,7
MET 2, i := 1 , 1     [programmadeel]^n  wordt uitgevoerd met i = 1
MET 2, i := 1 , 0     [programmadeel]^n  wordt helemaal niet uitgevoerd
MET-2, i := 3 ,-3 }
MET-2, i := 3 ,-4 }  [programmadeel]^n  wordt uitgevoerd met achtereenvolgens i = 3,1,-1,-3
MET-2, i := 3 , 4     [programmadeel]^n  wordt helemaal niet uitgevoerd
```

Als de cyclus verlaten wordt heeft i de waarde waarmee de cyclus voor het laatst doorlopen is, resp. de waarde van [start] als [programmadeel]^n helemaal niet uitgevoerd is.

Zelfs bij deze eenvoudige voorbeelden moet nog een voorbehoud worden gemaakt: als men namelijk binnen [programmadeel]^n de waarde van i verandert kan er best iets heel anders gebeuren.
Daarom zullen we nu precies zeggen wat er in een cyclus gebeurt. De cyclus is gelijkwaardig met het volgende stuk ECOL-programma. Hierin zijn de onderstreepte namen teller, stap en stop hulpvariabelen die door de compiler worden toegevoegd en waar de programmeur niet bij kan komen, en evenzo zijn label1, label2 en label3 labels die de programmeur niet kan bereiken.

```
        stap    := [stap]        ; deze opdrachten worden door de compiler gemaakt voor de
        teller  := [start]       ; regel die met "MET" begint
             i  := teller
        NAAR label 1
label2  -----
        -----                    ; dit is [programmadeel]^n :
        -----                    ; het stuk programma dat cyclisch wordt doorlopen
        -----
        stap    := [stap]
        teller  := i + stap      ; deze opdrachten worden door de compiler gemaakt voor de
label1  stop    := [stop]        ; regel waarin "HERHAAL" staat
        ALS stap < 0 EN teller < stop OF stap > 0 EN teller > stop DAN label3
            i   := teller
        NAAR label2
label3  -----                    ; dit zijn de opdrachten na "HERHAAL"
        -----
```

Men ziet hieruit dat de uitdrukkingen [start] en [stop] telkens nadat [programmadeel]^n is doorlopen maar één keer worden uitgerekend (in totaal dus juist één keer vaker dan de cyclus doorlopen wordt); vandaar ook de "hulpvariabelen" stap en stop. De "hulpvariabele" teller bewerkstelligt dat i de cyclus verlaat met de laatste waarde waarmee hij de cyclus doorlopen heeft.

Men ziet ook dat de waarden van [stap],[start] en [stop] niet geheel hoeven te zijn, noch dat de waarde van ([stop] - [start])/[stap] geheel hoeft te zijn. Bij het werken met niet gehele waarde van [stap] loopt men wel enig risico. Als men bijv. schrijft "MET 0.1, i:=0, 1", en men werkt op een binaire computer (vrijwel alle computers zijn tegenwoordig binair) dan is 0.1 niet exact in de computer voorstelbaar; wanneer nu de computer 0.1 voorstelt door een iets groter getal, dan krijgt i na 10 doorgangen ook niet precies de waarde 1 maar een iets grotere, en deze wordt verworpen. Gevolg: de laatste waarde waarmee i de cyclus doorloopt is omstreeks 0.9.

Met een cyclus doet men over het algemeen eenvoudige dingen zoals:

Voorbeeld

```
        MET 1 , i := 1 , 100
        r(i) := LEES                ; een rij wordt ingelezen
        HERHAAL
```

Voorbeeld

```
        max := r(1)
        MET 1 , i := 2 , 100        ; deze cyclus berekent de waarde van het grootste element
        ALS r(i) < max DAN 37       ; van de rij
        max := r(i)
     37 HERHAAL
```

Men moet wel heel goed weten wat men doet, wanneer men onderweg de waarden van i, [stap] en/of [stop] verandert; men kan er wel aardige dingen mee doen, maar voorzichtigheid is hier geboden.

In het cyclisch programma mogen weer andere cycli voorkomen (zgn. "binnen-cycli").

Voorbeeld

```
     1  MET 10, i := 0 , 90        ; i doorloopt nu de waarden 0, 10, 20, ...
     2  NR
     3  MET 1 , j := 0 , 9         ; voor elke waarde van i doorloopt j de waarden 0 t/m 9
     4  SCHRIJF := (i+j)M2
     5  HERHAAL                    ; deze HERHAAL hoort bij de MET op 3
     6  HERHAAL                    ; deze HERHAAL hoort bij de MET op 1
```

Met dit stukje programma worden de kwadraten van de getallen 0 t/m 99 afgedrukt, telkens met tien op een regel. De vraag welke HERHAAL bij welke MET hoort is eenvoudig te beantwoorden: als twee MET's op elkaar volgen zonder HERHAAL ertussen komt de HERHAAL van de tweede MET vóór de HERHAAL van de eerste MET (vergelijk de haakjes structuur bij [uitdr]ukkingen).

In een cyclus mogen ook RIJ-reserveringen en (alhoewel dit geen bijzondere effecten heeft) ook SUB-beschrijvingen voorkomen. Dit is echter verboden bij het gebruik in een FUN-beschrijving, omdat daar geen RIJ-reserveringen en SUB-beschrijvingen mogen voorkomen.

15. Lege opdracht

Volgens [23] mogen in een ECOL tekst ook regels voorkomen waarop behalve eventueel een label en kommentaar verder niets staat, zulks ter bevordering van de leesbaarheid voor de menselijke lezer. Een dergelijke regel wordt bij executie gewoon overgeslagen, ook als men er via een sprong op terecht komt. Het is dus net of er op zo'n regel een opdracht staat die niets doet, een lege opdracht.

16. Invoer

16.1 [variabele] := LEES
[variabele] := LEESSYM

Met LEES wordt het eerstvolgende getal van een sequentieel extern medium gelezen, bijv. een ponsband of schrapkaart, en dit getal wordt als waarde toegekend aan de variabele in het linkerlid (de term sequentieel staat tegenover adresseerbaar; men kan niet zeggen: "ik wil de zoveelste hebben" maar slechts "geef mij de volgende"). Een getal op het externe medium moet een gedaante hebben als in syntax [38], eventueel voorafgegaan door + of - en ook hier geldt dat spaties geen betekenis hebben. LEES leest dan ook op het externe medium symbool na symbool totdat een cijfer verschijnt of een + of - gevolgd door een cijfer; hiermee is dan een getal begonnen, en dit getal wordt geacht op te houden zodra een symbool verschijnt dat niet meer tot het getal kan behoren, dus een symbool dat geen punt of cijfer is, of ook het symbool punt zelf als er in het getal al een punt is voorgekomen. Bij dit alles worden spaties steeds veronachtzaamd.
Het eerste symbool dat niet meer tot het getal behoort is tevens het eerste symbool dat door een volgende LEES of LEESSYM gelezen wordt.
Als LEES tijdens het lezen van een getal bij het einde van het medium komt, is hiermee het getal ook afgelopen. Als LEES op het externe medium geen getal meer aantreft volgt een foutmelding, alsmede beëindiging der executie van het programma.

Met LEESSYM wordt het eerstvolgende symbool van een sequentieel extern medium gelezen, en de interne waarde van dit symbool wordt aan de variabele in het linkerlid toegekend. De interne waarde van een extern symbool is een geheel getal. De relatie is in onderstaande tabel vastgelegd. Als LEESSYM een ander symbool ontmoet dan de onder 0 t/m 91 genoemde, wordt de waarde 98 afgeleverd (zie os = onbekend symbool). Als het externe medium uitgeput is levert LEESSYM de waarde 99 af (zie et = einde tekst).

Als na LEESSYM een LEES wordt uitgevoerd is het eerste symbool dat LEES leest het laatste dat LEESSYM gelezen heeft. Hierdoor is het mogelijk met LEESSYM te speuren naar het begin van een nieuw getal en dan toch nog met LEES het hele getal te lezen.

TABEL van symbolen en hun interne waarde:

cijfers		letters												overige symbolen					
sym	get	sym	get	sym	get	sym	get	sym	get	sym	get	sym	get	sym	get	sym	get	sym	get
0	0	a	10	k	20	u	30	A	40	K	50	U	60	.	70	+	80	sp	90
1	1	b	11	l	21	v	31	B	41	L	51	V	61	,	71	-	81	nr	91
2	2	c	12	m	22	w	32	C	42	M	52	W	62	:	72	×	82		
3	3	d	13	n	23	x	33	D	43	N	53	X	63	;	73	/	83		
4	4	e	14	o	24	y	34	E	44	O	54	Y	64	?	74	=	84		
5	5	f	15	p	25	z	35	F	45	P	55	Z	65	!	75	<	85		
6	6	g	16	q	26			G	46	Q	56			'	76	>	86		
7	7	h	17	r	27			H	47	R	57			"	77	⩽	87		
8	8	i	18	s	28			I	48	S	58			(	78	⩾	88	os	98
9	9	j	19	t	29			J	49	T	59			)	79	≠	89	et	99

Met sp en nr bedoelen we 'spatie' en 'overgang op nieuwe regel', met et bedoelen we 'einde tekst' (om aan te geven dat een tekst afgelopen is), met os een 'overig symbool' of 'onbekend symbool'.

Voorbeeld 16

Het externe medium bevat

```
+-.-   5   .   A+.5   7.23   .   -17
```

en wordt gelezen met

```
a:= LEES        ; leest tot A; -5 naar a
b:= LEES        ; begint bij A en leest tot vijfde punt; 57.23 naar b
c:= LEESSYM     ; leest vijfde punt; 70 naar c
d:= LEESSYM     ; leest spatie; 90 naar d
e:= LEESSYM     ; leest -; 81 naar e
f:= LEES        ; begint bij - en leest tot einde; -17 naar f
g:= LEESSYM     ; medium uitgeput; 99 naar g
```

Tenslotte: men mag LEES en LEESSYM ook in [uitdr]ukkingen gebruiken (zie 18).

16.2

```
                                    h        k
                                    ↓        ↓
[GP-rij] := BESTAND([uitdr] , [uitdr] , [uitdr])
```

De eerste parameter "[uitdr]" is altijd het bestand-nummer (mogelijk is op sommige computers hier alleen maar een [getal] toegestaan i.p.v. een willekeurige [uitdr]).
De opdracht kopieert het stuk h t/m k van het aangewezen bestand in de rij met de naam [GP-rij].
Het kopiëren eindigt zodra of de rij, of het aangegeven stuk bestand uitgeput is.
Alle [uitdr] worden weer eerst afgerond.

16.3 [GP-rij] := BESTAND([uitdr])
doet hetzelfde als:

```
[GP-rij] := BESTAND([uitdr], 1, totale lengte van het bestand)
```

17. Uitvoer

17.1

```
                m        n
                ↓        ↓
SCHRIJF([uitdr] , [uitdr]) := [uitdr]
```

Het uitvoerapparaat (in het algemeen een regeldrukker) drukt de waarde van de [uitdr] rechts van het wordt-teken in decimale voorstelling af met teken (een plusteken wordt mogelijk vervangen door spatie), m cijfers voor de decimale punt en n cijfers erachter, in totaal dus m+n+2 posities. Bij

het gedeelte voor de punt worden niet-significante nullen onderdrukt. Als n=0 worden evenwel geen decimale punt met cijfers erachter afgedrukt, dus slechts m+1 posities. De [uitdr] voor m en n worden weer afgerond.
Wanneer het gedeelte vóór de decimale punt meer cijfers nodig heeft dan door m is aangegeven, of wanneer m te groot is (varieert van machine tot machine) dan wordt het getal afgedrukt volgens SCHRIJM.

17.2 SCHRIJF := [uitdr]
Als de waarde van [uitdr] geheel is doet deze opdracht hetzelfde als SCHRIJF(6,0) := [uitdr], in het andere geval hetzelfde als SCHRIJF(6,4) := [uitdr].
In beide gevallen worden zoveel spaties toegevoegd dat op het uitvoerapparaat evenveel posities worden bezet als met SCHRIJM.

17.3 SCHRIJM := [uitdr]
Het uitvoer-apparaat drukt de afgeronde waarde rechts van het ":=" teken in decimale voorstelling af als een <u>mantisse</u> tussen 0 en 1 gevolgd door de exponent van de macht van 10 waarmee dit getal vermenigvuldigd moet worden om de waarde van de meegegeven [uitdr] voor te stellen. Zowel mantisse als exponent worden voorafgegaan door een voorteken.
Het aantal decimalen van mantisse en exponent is afhankelijk van de machine waarop het programma wordt verwerkt.

<u>Voorbeeld</u>

SCHRIJF := pi × 10 M5	⇒	314159.2654
SCHRIJF(3,5) := pi × 10 M2	⇒	314.15927
SCHRIJF := -pi × 10 M3	⇒	- 3141.5927
SCHRIJM := pi × 10 M5	⇒	+0.3141592653589E+6
SCHRIJF(5,5) := pi × 10 M5	⇒	+0.3141592653589E+6
SCHRIJF := -12	⇒	- 12

17.4 SCHRIJFSYM := [uitdr]
Er wordt gekeken of de op een geheel getal afgeronde waarde van [uitdr] hoogstens 91 is en voorkomt in de tabel van symbolen (zie 16.1). Zo ja, dan wordt het symbool met die waarde afgedrukt; zo nee, dan wordt een bijzonder symbool afgedrukt. Wat er op papier verschijnt als het uitvoerapparaat een symbool moet afdrukken waartoe het niet in staat is (bijv. een kleine letter als het alleen hoofdletters kent) kan van machine tot machine verschillen.

<u>Voorbeeld</u>

SCHRIJFSYM := 74 drukt af ?

17.5 TEKST := "[sym]$^{n+1}$"
Afgedrukt wordt de tekst tussen de aanhalingstekens; in deze tekst mag echter geen aanhalingsteken voorkomen en geen [↩].

17.6 SPATIE([uitdr])
Het uitvoer-apparaat geeft net zoveel spaties als door de afgeronde waarde van [uitdr] wordt aangegeven; als deze afgeronde waarde ≤ 0 is, gebeurt er niets.

17.7 SPATIE
doet hetzelfde als de opdracht:

SPATIE(1)

17.8 NR([uitdr])
Het uitvoer-apparaat schuift het papier evenveel regels omhoog als door de afgeronde waarde van [uitdr] wordt aangegeven; als deze afgeronde waarde ≤ 0 is, gebeurt er niets met het papier. Na het omhoogschuiven, maar ook bij "NR(0)" begint het afdrukken weer vooraan de regel (dit vooraan laten

beginnen bij NR(0) kan bij sommige machines op moeilijkheden stuiten).

17.9 NR
doet hetzelfde als de opdracht

NR(1)

17.10 NP
Deze opdracht laat op het uitvoerapparaat een nieuwe pagina verschijnen.

17.11

```
                          h         k
                          ↓         ↓
BESTAND([uitdr] , [uitdr] , [uitdr])  := [GP-rij]
```

De eerste parameter "[uitdr]" is altijd het bestand-nummer (zie opmerking in 16.2).
De opdracht kopieert de rij met de naam [GP-rij] in het stuk h t/m k van het aangewezen bestand.
Het kopiëren eindigt zodra <u>of</u> de rij, <u>of</u> het aangegeven stuk bestand uitgeput is.

17.12 BESTAND([uitdr]) := [GP-rij]
doet hetzelfde als de opdracht:

BESTAND([uitdr] , 1,totale lengte van het bestand) := [GP-rij]

17.13 <u>Opmerkingen uitvoer</u>
De wijze waarop de computer reageert op situaties zoals:

- men wil meer symbolen op een regel hebben dan mogelijk is bij een bepaald uitvoerapparaat,
- men wil een symbool afdrukken dat er niet is op het uitvoerapparaat

kan van machine tot machine verschillen, evenals de voorstelling van allerlei symbolen en tekens (bijv. "*" in plaats van "×", "<=" in plaats van "≤", "$_{10}$" in plaats van "E" in een SCHRIJM-getal).
Men bedenke hierbij ook, dat vele computers regeldrukkers van vaak uiteenlopend type hebben, en het lang niet zeker is dat de ECOL-uitvoer altijd over hetzelfde type regeldrukker loopt.

18. <u>Standaard-funkties</u>

De volgende standaard-funkties worden in iedere ECOL-implementatie verondersteld: ("u" staat hier steeds voor een [uitdr], tenzij anders vermeld):

G(u)	levert het gehele deel van u af:	G(2.7) is +2 G(-2.3) is -3
ABS(u)	levert de absolute waarde van u af	
WRTL(u)	levert $\sqrt{u}$ af als $u \geqslant 0$; anders foutmelding	
SQRT(u)	doet hetzelfde als WRTL(u)	
SIN(u)	levert de waarde van de sinus van u (in radialen) af	
COS(u)	levert de waarde van de cosinus van u (in radialen) af	
ARCTAN(u)	levert de waarde van de hoek (in radialen) af die gelegen is tussen $-\pi/2$ en $+\pi/2$ en waarvan de tangens u is	
LN(u)	levert ${}^{e}\log(u)$ af als $u > 0$; anders foutmelding	
LOG(u)	levert ${}^{10}\log(u)$ af als $u > 0$; anders foutmelding	
EXP(u)	levert de waarde van e tot-de-macht u af	
GOK(u1,u2)	levert (pseudo) aselecte getallen af, t.w. met ongeveer gelijke waarschijnlijkheid een der gehele getallen tussen de afgeronde waarden van u1 en u2, deze afgeronde waarden inbegrepen. Dus GOK(1,6) en GOK(6,1) werken als dobbelsteen	

GOKC	levert (pseudo) aselecte getallen af, t.w. met ongeveer gelijke waarschijnlijkheid een der getallen tussen 0 en 1, 0 wel en 1 niet inbegrepen
ONDIN([GP-rij])	levert de ONDerINdex af van de rij met naam [GP-rij]
BOVIN([GP-rij])	" " BOVenINdex af van " " " " [GP-rij]
LEES	leest het volgende getal van de band en levert daarvan de waarde af net zoals in de invoer-opdracht [variabele] := LEES. Men mag dus ook schrijven bijv. "y := LEES + 1"
LEESSYM	idem maar nu voor het volgende symbool op de band of kaart
SKIP	doet net zo vaak LEESSYM totdat [↵] is gepasseerd
PS	(positie schrijfkop) levert het aantal symbolen af dat door het uitvoerapparaat op de huidige regel is afgedrukt
PP	(positie papier) levert het aantal voltooide regels op de huidige pagina van het uitvoer-apparaat
STOP	de executie van dit programma wordt beëindigd

V. SUB EN FUN

19. Inleiding

Het is bij het schrijven van programma's niet ongebruikelijk dat men in verschillende delen van zijn programma gelijke of bijna gelijke reeksen opdrachten nodig heeft. Het SUB en FUN mechanisme maakt het nu mogelijk zo'n reeks opdrachten eenmaal op te schrijven, waarna ze van verschillende plaatsen door één simpele opdracht kunnen worden geaktiveerd. Men zie bijv. de standaardfuncties (zie 18), die eenmaal ergens in het geheugen van de computer uitgeprogrammeerd staan, en die men in zijn programma overal heel simpel kan aanroepen.
De SUB is een heel eenvoudig mechanisme, dat dan ook niet zo erg flexibel is, de FUN opent veel verder gaande mogelijkheden.

20. SUB-beschrijving

Volgens syntax-regel [71] is een [SUB-beschrijving] :

[nummer] SUB[SUB-naam] ▯
[opdracht]^n
[nummer] END ▯

In een SUB-beschrijving mogen dus geen RIJ-reserveringen en ook geen SUB-beschrijvingen voorkomen. Wel zijn alle [opdrachten] , inklusief de [cyclus] en de [SUB-roep], toegestaan.
Omdat een SUB-beschrijving deel uitmaakt van het hoofdprogramma, gelden in het bijzonder de in 7 en 8 behandelde context-regels.
Men noemt een SUB-beschrijving ook wel een subroutine.

Voorbeeld van een SUB-beschrijving

```
100  SUB skip tot nul        ; deze subroutine leest de getalband tot en met het eerst-
                             ; volgende getal 0
101  getal := LEES
102  ALS getal ≠ 0 DAN 101
103  END                     ; de subroutine wordt verlaten via de achterdeur, zoals het be-
                             ; hoort, zie 8.
```

Deze SUB-beschrijving mag overal in het hoofdprogramma staan mits buiten een SUB-beschrijving, en ook overal in het hoofdprogramma worden aangeroepen (dus wel in een SUB-beschrijving, niet in een FUN-beschrijving).