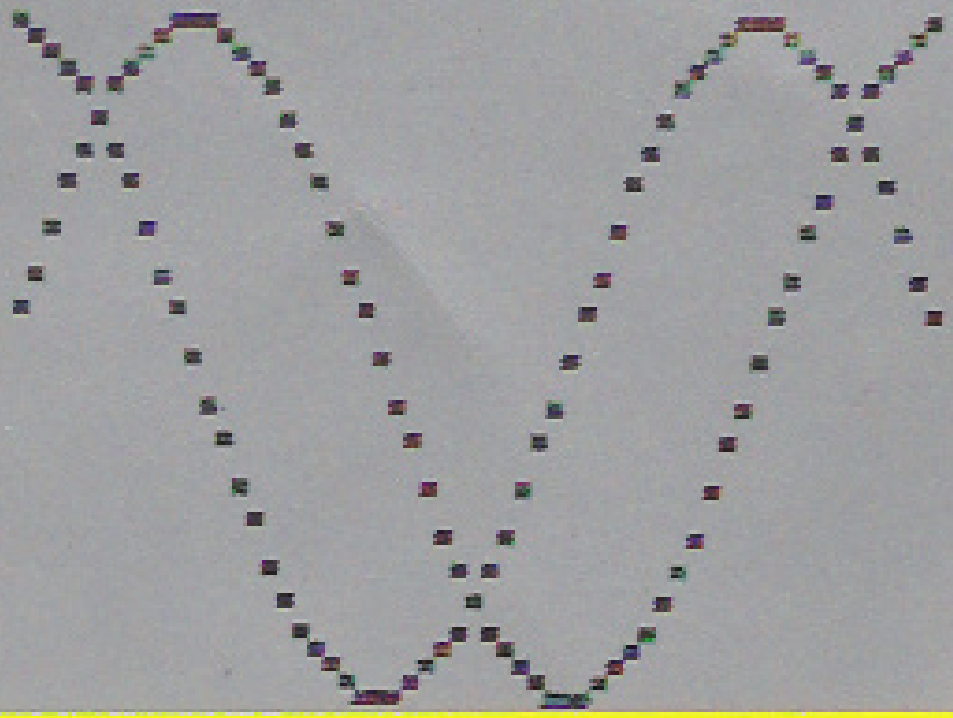
Max Voorburg

Toepassingen en spellen voor de ZX81

```
 10 REM  ************************
 20 REM  *** SINUS/COSINUS  ***
 30 REM  ************************
 40 REM
 50 FOR X=1 TO 63
 60 LET Y=X+PI/24
 70 PLOT X,SIN (Y)*21+21
 80 PLOT X,COS (Y)*21+21
 90 NEXT X
100 STOP
110 REM  ************************
```



Kluwer Technische Boeken

TOEPASSINGEN EN SPELLEN

VOOR DE ZX81

Max Voorburg

Toepassingen en spelen voor de ZX81

KLUWER TECHNISCHE BOEKEN B.V. DEVENTER - ANTWERPEN

Omslagprint: Fa. Putto, Apeldoorn
Illustraties: Wim Niessink

ISBN 90 201 1604 5
D/1983/0108/169

1e druk 1983

Inhoud

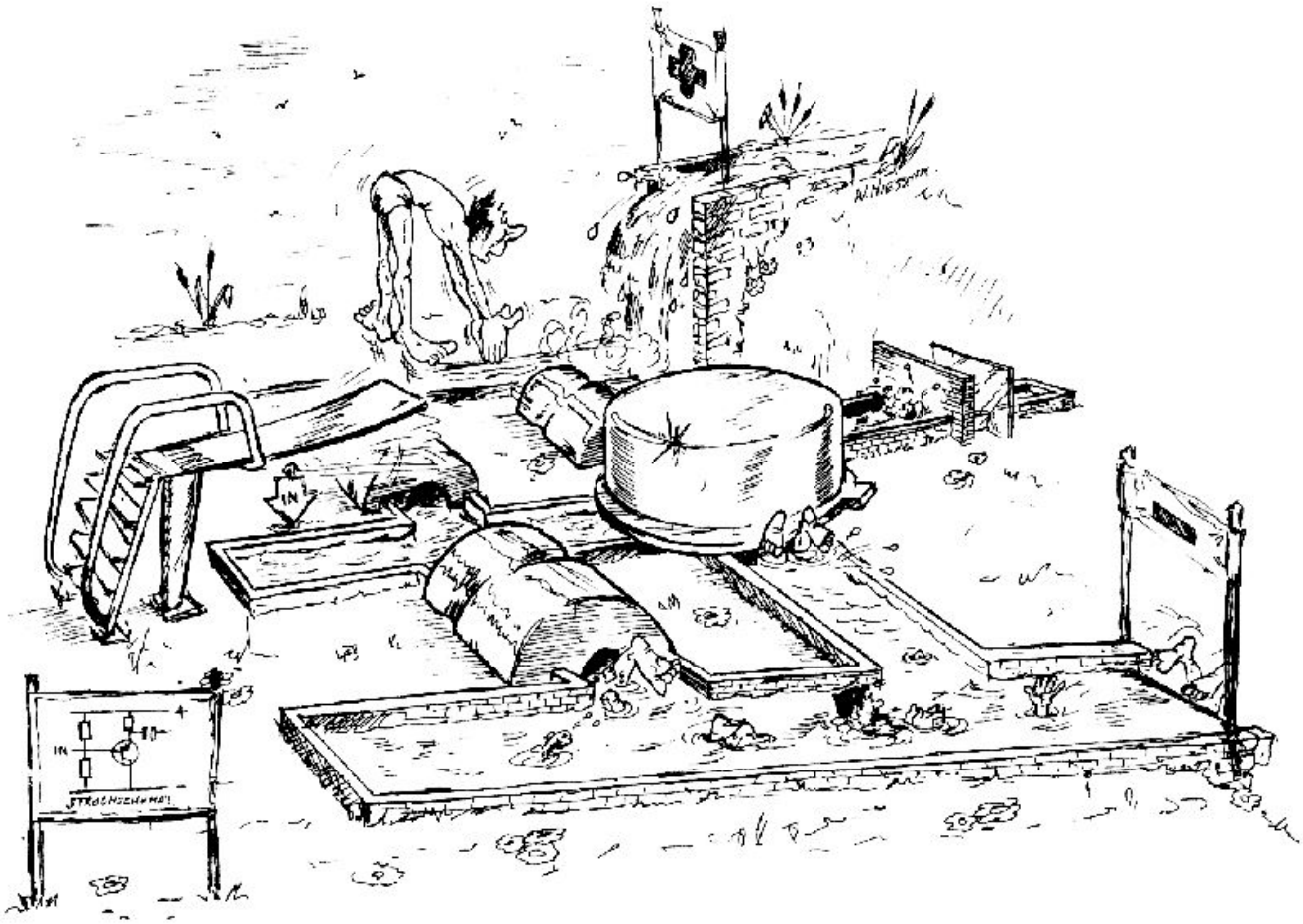
Inleiding

In Nederland is de ZX81 een enorm succes. Dit succes is niet alleen toe te schrijven aan de lage prijs voor een volwaardige computer, maar ook aan de compleetheid van de machine. Men krijgt een volwaardige microcomputer voor een paar honderd gulden. Daarbij moet men zich bedenken dat voor een machine met dezelfde capaciteit een aantal jaren geleden duizenden guldens moest worden neergeteld.
In het begin was de ZX81 alleen in de speciaalzaken te koop, zoals elektronicawinkels en computershops. Op dit moment kan de ZX81 bij de grote warenhuizen en zelfs bij postorderbedrijven worden gekocht. Deze machine wordt niet alleen meer gekocht door professionals, mensen die in hun dagelijks werk ook met computers omgaan, maar ook door mensen voor wie de computer iets nieuws is. Voor wie de computer niet zo vanzelfsprekend is. Voor hen is niet alleen de computer nieuw, maar ook het programmeren is nieuw!
Dit boek is vooral voor die laatste groep geschreven. Alle programma's in dit boek passen in de 1K-versie van de ZX81. Dit is de versie, zoals deze standaard in de handel is. Het blijkt dat slechts een gering aantal ZX81-computers is uitgebreid met de 16K-module.
Om de programma's in de 1K-versie te persen, is gebruik gemaakt van een aantal kunstgrepen. In een apart hoofdstuk zullen deze kunstgrepen uitgelegd worden. Men kan dan zelf in eigen programmatuur deze kunstgrepen overnemen en gebruiken.
De programma's beslaan een aantal terreinen. Er zijn een aantal spelletjes, eenvoudige educatieve programma's, financiële programma's, maar ook vrij complexe statistische en wiskundige programma's. Er is naar gestreefd een zo breed mogelijk scala van programmatuur voor de gebruiker te verzamelen, zodat hij een redelijk beeld krijgt waar deze ogenschijnlijk kleine machine toe in staat is.
Normaal wordt de handleiding in het programma zelf verwerkt, dat is gezien het kleine geheugen niet mogelijk. In plaats daarvan staat de handleiding in het boek. Daar waar het noodzakelijk is om de werking te demonstreren, zal een voorbeeld worden toegevoegd. Tevens wordt van een groot aantal programma's de werking uiteengezet. Om de werking nog beter te verduidelijken zijn een aantal programma's voorzien van stroomschema's.
De programma's in dit boek zijn niet gestructureerd modulair opgebouwd. Dit is niet mogelijk vanwege de zeer beperkte geheugenruimte. De meeste programma's in dit boek zijn niet op andere computers te gebruiken. De

programma's maken gebruik van de speciale mogelijkheden, die alleen op de ZX81 voorkomen. Men zal bij een flink aantal programma's PEEK- en POKE-statements tegenkomen. Verder zullen er constructies worden gebruikt, die op zijn zachtst gezegd vreemd voorkomen voor de geroutineerde programmeur. Al met al hoopt de auteur dat de gebruiker van dit ZX81-boek wordt geactiveerd om met dit boek ook zelf nieuwe programma's te schrijven.

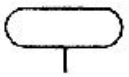
Den Haag, voorjaar 1983.

1. Stroomschema's

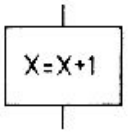


In de inleiding is reeds opgemerkt dat een aantal programmabeschrijvingen voorzien zijn van een stroomschema. Voor veel ZX81-bezitters is dit de eerste kennismaking met de automatisering en voor hen zullen deze stroomschema's zeker onbekend voorkomen. Daarom op deze plaats een globale uitleg. Niet een diepgaand verhaal van het hoe en het waarom, maar een zodanige uitleg dat daarmee de werking van de programma's verduidelijkt kan worden.
Stroomschema's worden in de automatisering gebruikt om de werking van een programma te verduidelijken. Het wordt voornamelijk gedaan om onoverzichtelijke of complexe zaken te vereenvoudigen. Stroomschema's zijn eigenlijk het beste te vergelijken met de principeschema's die in de elektronica worden gehanteerd.
Er zijn overigens meerdere methoden in de automatisering in gebruik, hier wordt de meest bekende uiteengezet. Dat wil niet zeggen dat dit nu de beste methode is. Alleen wordt deze methode in de wereld veel gebruikt. De symbolen, die in dit boek worden gehanteerd, zal men in tal van boeken of tijdschriften eveneens tegenkomen.
Bij deze stroomschema's worden in principe maar een viertal symbolen gebruikt. Het volgende symbool geeft een begin of een stop aan. De

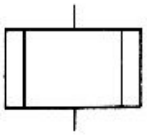
betreffende tekst wordt in de ellips geplaatst.



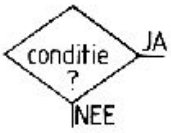
Een rechthoek geeft een bewerking aan of wel een proces, bijvoorbeeld:



BASIC maakt gebruik van subroutines. Dit zijn stukken programma, die meerdere malen kunnen worden herhaald en die vanuit meerdere plaatsen in het programma kunnen worden aangeroepen. In de stroomschematechniek heeft een subroutine een apart symbool. Het stroomschema van de subroutine zelf wordt dan op een apart blad getekend.

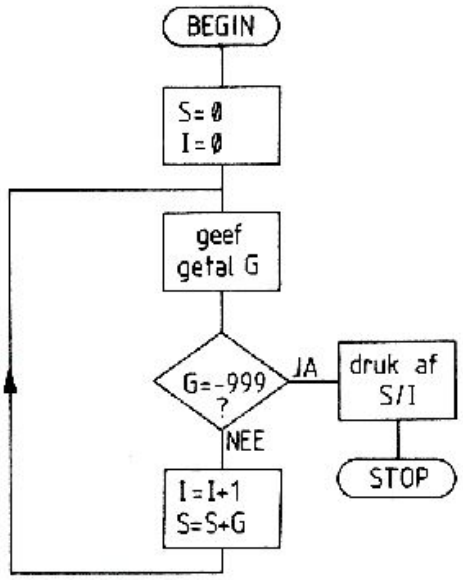


Het enige symbool dat nog overblijft, is het beslissingssymbool, de zogenaamde 'zoute drop'. Hierbij wordt een conditie getest en afhankelijk van het antwoord wordt een bepaalde tak van het programma uitgevoerd. Het antwoord op het testen van de conditie is altijd 'ja of nee' of 'waar of niet-waar'.



Verder geldt bij deze schematechniek dat er alleen pijltjes worden gebruikt, indien de stroomrichting niet meer van boven naar beneden verloopt.

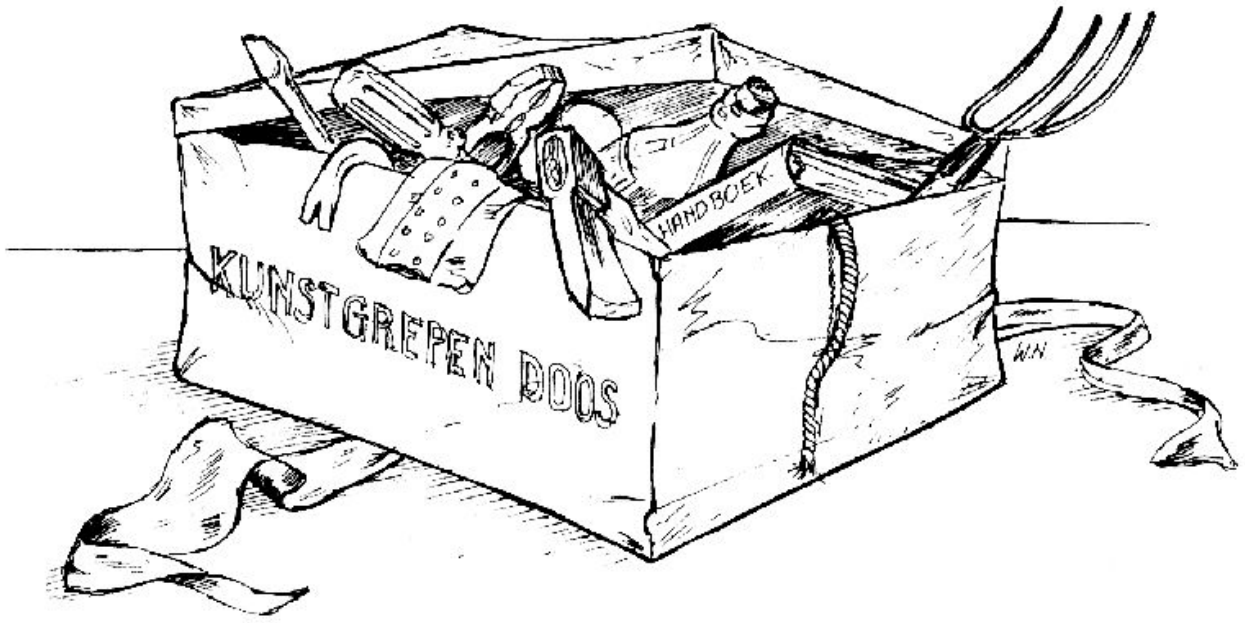
Als voorbeeld van deze stroomschema's zal de berekening van het rekenkundig gemiddelde worden genomen. Hiertoe worden een aantal getallen bij elkaar opgeteld en daarna gedeeld door het aantal getallen. Wanneer het ingevoerde getal gelijk is aan -999 dan geeft de gebruiker daarmee te kennen dat hij geen invoer meer heeft. Nadat de invoer gestopt is, kan de berekening worden afgemaakt en het resultaat worden uitgeprint. Zonder verder op de theorie in te gaan volgt hier het stroomschema met daarachter het bijbehorende BASIC-programma.



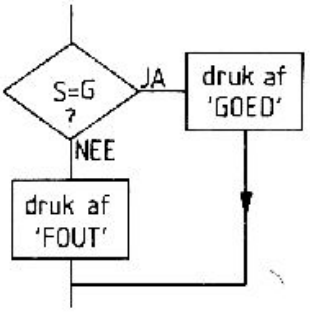
BASIC-programma

```
10 LET S=0
20 LET I=0
30 PRINT "GEEF GETAL: ";I+1;
40 INPUT G
50 PRINT G
60 IF G=-999 THEN GOTO 100
70 LET I=I+1
80 LET S=S+G
90 GOTO 30
100 PRINT "GEMIDDELDE:";S/I
110 STOP
```

2. De doos met kunstgrepen



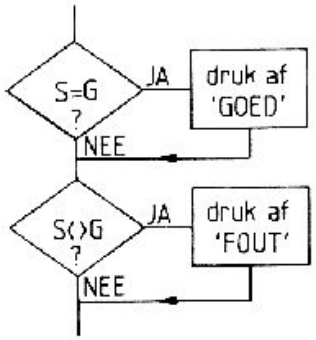
Als men programma's maakt voor de ZX81 in BASIC en zeker voor de 1K-versie, moet men opletten dat men binnen de beperkte geheugenruimte blijft. Meestal moet het programma er als het ware 'ingeperst' worden. Dit eist dat men zijn toevlucht zal moeten nemen tot bepaalde onorthodoxe oplossingen. In dit hoofdstuk worden een aantal van deze onorthodoxe methoden besproken. Men kan ze in de eigen programmatuur misschien goed gebruiken, omdat ze vaak erg ruimtebesparend zijn.
Het blijkt dat het bij de ZX81 is toegestaan meerdere BASIC-statements op een regel te zetten. Verder is de werking van bepaalde BASIC-statements ruimer dan in de standaard-BASIC waarin de meeste boeken over BASIC geschreven zijn.
In veel toepassingen komt het voor dat men afhankelijk van een bepaald resultaat van een bewerking of vergelijking een tekst wil afdrukken. Men heeft bijvoorbeeld de variabelen S en G. Wanneer deze twee variabelen met elkaar overeenstemmen, wordt de tekst 'GOED' afgedrukt. Is de variabele G ongelijk aan S dan moet de tekst 'FOUT' worden afgedrukt. In een stroomschema ziet dit er als volgt uit:



Het voorgaande stroomschema kan als volgt in de ZX81 worden gecodeerd:

```
70 IF S=G THEN GOTO 100
80 PRINT "FOUT"
90 GOTO 110
100 PRINT "GOED"
110 vervolg programma
```

Het stroomschema kan ook op een andere manier worden getekend:



Het bijbehorende stukje programma is al een stuk korter en het scheelt een tweetal GOTO-statements met bijbehorende regelnummers:

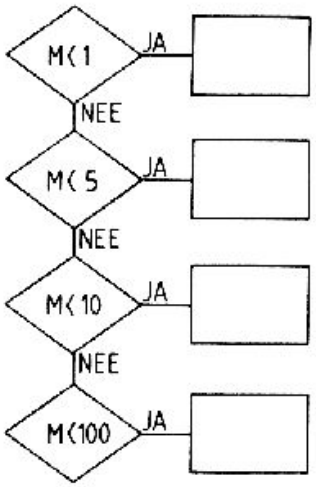
```
70 IF S=G THEN PRINT "GOED"
80 IF S<>G THEN PRINT "FOUT"
90 vervolg programma
```

Er is nog een kortere methode, die slechts één programmaregel beslaat. Deze methode zal in de hierna volgende programma's herhaaldelijk worden gebruikt:

```
70 PRINT "GOED" AND S=G;"FOUT" AND S<>G
```

Het woord AND wordt een Booleaanse operator genoemd. Er zijn er nog meer, zoals OR en NOT. Met deze operators zijn als het ware verkapte IF...THEN-situaties op te bouwen. Een voorbeeld ziet men hierboven. Wanneer S gelijk is aan G dan is het antwoord 'JA'. Dit betekent dat 'GOED' wordt afgedrukt. Is het antwoord 'NEE' dan wordt 'GOED' niet afgedrukt, maar de tweede vergelijking voldoet en er wordt 'FOUT' afgedrukt. De koppeling tussen 'GOED' en S=G en 'FOUT' en S<>G wordt tot stand gebracht door AND.
Een gelijksoortige kunstgreep is te realiseren met het GOTO-statement. Hiermee is men dan in staat een soort conditionele GOTO te maken.
Stel, de variabele M kan een beperkt aantal waarden aannemen en afhankelijk van een bepaalde waarde wordt er naar een bepaald regelnummer gesprongen. In een stroomschema ziet dat er uit zoals aangegeven op pagina 14.
Uit het voorbeeld blijkt dat het probleem niet met eenvoudige statements is op te lossen. De enige oplossing die dan nog over schijnt te blijven is:

```
20 IF M<1 THEN GOTO 100
30 IF M<5 THEN GOTO 200
40 IF M<10 THEN GOTO 300
50 IF M<100 THEN GOTO 400
```



Een nadeel dat direct opvalt is de enorme hoeveelheid ruimte die nodig is. Er blijkt ook nog een kortere oplossing te zijn en die luidt:

```
20 GOTO (100 AND M<1)+(200 AND M<5)+(300 AND M<10)+
        (400 AND M<100)
```

Uit deze oplossing komt duidelijk naar voren dat deze is afgeleid van het allereerste voorbeeld.
Er volgen nu nog twee kleine voorbeelden, die het uiteengezette demonstreren.

Voorbeeld 1:

```
10 PRINT "GEEF S-WAARDE ";
20 INPUT S
30 PRINT S
40 LET G=10
50 PRINT "GOED" AND S=G;"FOUT" AND S<>G
60 GOTO 10
```

Wanneer men het bovenstaande programmaatje intoetst en daarna runt, wordt er om een S-waarde gevraagd. Geeft men voor S een tien dan moet er 'GOED' worden afgedrukt. Geeft men daarentegen een waarde aan S die ongelijk is aan tien dan dient er 'FOUT' te worden afgedrukt.

Voorbeeld 2:

```
10 PRINT "GEEF M ";
20 INPUT M
30 PRINT M
40 PRINT "DIT IS REGEL ";
50 GOTO (60 AND M<1)+(80 AND M<5)+(100 AND M=>5)
60 PRINT "DIT IS REGEL 60"
70 GOTO 10
80 PRINT "80"
90 GOTO 10
100 PRINT "100"
110 GOTO 10
```

Men kan nu met behulp van dit programma het bovenstaande mechanisme uitproberen. Het is wel handig het programmaatje naast de machine te hebben liggen, zodat men van tevoren kan bepalen op welk regelnummer het programma moet uitkomen. Is bijvoorbeeld M gelijk aan 4 dan dient er naar regel 80 te worden gesprongen. Wordt daarentegen M gelijk gemaakt aan 40 dan moet worden gesprongen naar regel 100.
De ZX81 is uitgerust met een dynamische GOTO en GOSUB. Hoewel dit mechanisme in de Engelse handleiding besproken wordt, wordt de kunstgreep hier voor de volledigheid nogmaals uitgelegd aan de hand van een tweetal programma's.
In de meeste BASIC's dient achter het GOTO-statement een vast regelnummer te staan, dit geldt eveneens voor het GOSUB-statement. Het gevolg is dat altijd naar een vaste plaats in het programma wordt gesprongen. Bij de ZX81 is het mogelijk om aan de hand van een variabele of van een uitdrukking een regelnummer te bepalen, waar dan door het programma naar toe wordt gesprongen.

Voorbeeld 3:

```
10 PRINT "WAARDE K ";
20 INPUT K
30 PRINT K
40 GOTO K*2
60 PRINT "REGEL 60 "
70 GOTO 10
80 PRINT "REGEL 80"
90 GOTO 10
100 PRINT "REGEL 100"
110 GOTO 10
```

De in te voeren K-waarden zijn 30, 40 en 50. De ingevoerde K-waarde wordt met twee vermenigvuldigd en het resultaat bepaalt het regelnummer waar naar toe gesprongen wordt. In het geval dat voor K een waarde 50 wordt ingevoerd, zal naar regel 100 worden gesprongen.
Hetzelfde mechanisme werkt ook bij subroutines. Zonder verdere uitleg wordt alleen een voorbeeld gegeven.

Voorbeeld 4:

```
10 PRINT "WAARDE K ";
20 INPUT K
30 PRINT K
40 GOSUB 2*K
50 GOTO 10
60 PRINT "REGEL 60 "
70 RETURN
80 PRINT "REGEL 80"
90 RETURN
100 PRINT "REGEL 100 "
110 RETURN
```

Men dient bij het gebruik van de GOTO- en GOSUB-statements op deze manier zeer goed op te letten. Ten eerste moet men zorgen dat de uitkomst van de bewerking of de variabele, die naar het regelnummer verwijst altijd op een geheel getal uitkomt. Ten tweede dient het regelnummer waar naar toe wordt gesprongen ook inderdaad in het programma aanwezig te zijn.

Wanneer men eens naar programma's gekeken heeft, die in verschillende tijdschriften of in de software-reeks van Kluwer zijn gepubliceerd, zal het zijn opgevallen dat er statements worden gebruikt die in de ZX81 niet voorhanden zijn. Het zijn de statements DATA en READ. Met deze statements kan een intern bestand worden opgebouwd. Een bestand dat als het ware in het programma zit ingebakken.
Het DATA-statement stelt de gebruiker in staat een bestand op te bouwen. Met het READ-statement kan het bestand sequentieel worden uitgelezen. Sequentieel uitlezen betekent dat men het bestand alleen van voor naar achter kan lezen. Men kan dus niet ergens willekeurig in het bestand beginnen te lezen.
Toch is het jammer dat de ZX81 dergelijke statements niet kent, want het kan in een aantal gevallen best handig zijn. Toch is het mogelijk, zij het op zeer beperkte schaal, iets dergelijks in de ZX81 op te bouwen. Deze mogelijkheid zal aan de hand van een aantal voorbeelden worden uitgelegd. Een van de voorgestelde mogelijkheden wordt in de programma's een aantal keren gebruikt, zodat men daar ook de werking kan terugvinden. Stel dat men een bestand wil opbouwen met de namen van de dagen. Hiertoe worden de namen van de dagen in een string opgeborgen, maar dan wel zodanig dat alle namen evenveel ruimte innemen. Omdat de ZX81 maar een beperkte geheugencapaciteit heeft, moet men alleen maar het relevante gedeelte van de naam opslaan. In het geval van de namen van de dagen van de week kan het stukje 'dag' worden weggelaten. In de string, bijvoorbeeld A$, zal dan komen te staan:

```
A$="ZONMAANDINSWOENSDONDERVRIJZATERZON"
```

Een nadeel is dat de dagen niet zonder meer zijn terug te lezen. Elke dagaanduiding bestaat uit een verschillend aantal karakters. Bijvoorbeeld ZON heeft drie karakters, terwijl DONDER zes karakters heeft.
Nu zijn een aantal oplossingen mogelijk, waarvan er twee worden besproken, maar er zijn er meer!

Oplossing 1
Men kan er voor zorgen dat alle namen van de dagen uit een gelijk aantal karakters gaan bestaan. Dit is te bereiken door er extra spaties tussen te zetten. Men bepaalt hiertoe de naam met de meeste karakters. Dat is in dit geval DONDER met zes karakters. De spaties worden aan de voorzijde van de naam bijgevoegd. Om dit te verduidelijken worden alleen ter illustratie in plaats van spaties streepjes geplaatst.

```
A$="---ZON--MAAN--DINS-WOENSDONDER--VRIJ-ZATER"
```

Het is nu mogelijk om met substrings de gewenste dagnaam te selecteren. Bij de ZX81 wordt dit ook wel slicing genoemd. Hierna volgt een programma waarmee de werking wordt gedemonstreerd.

Voorbeeld 5:

```
10 LET A$="---ZON--MAAN--DINS-WOENSDONDER--VRIJ-ZATER"
20 PRINT "WELK DAGNR. ";
30 INPUT P
40 PRINT P
50 LET B$=A$(P*6-5 TO P*6)
60 PRINT B$+"DAG"
70 GOTO 20
```

Let er op dat de liggende streepjes in A$ vervangen moeten worden door spaties.
Door het dagnummer op te geven wordt een bepaald gedeelte uit A$ geselecteerd en in B$ geplaatst. Kiest men het dagnummer 1, dan wordt het eerste tot en met het zesde karakter uit A$ gepakt. Kiest men dagnummer 4, dan wordt het negentiende tot en met het vierentwintigste karakter geselecteerd. Deze methode heeft boven de hierna te bespreken methode het voordeel dat het mogelijk is een substring willekeurig te benaderen.

Oplossing 2
De vorige methode had het nadeel dat er vrij veel loze ruimte moest worden toegevoegd. In totaal zijn het 11 spaties. De tweede methode plaatst geen extra spaties bij, maar plaatst voor elk stuk naam een getal. Dit getal geeft aan uit hoeveel karakters dat stuk naam bestaat. Dit gaat er dan als volgt uit zien:

```
A$="3ZON4MAAN4DINS5WOENS6DONDER4VRIJ5ZATER"
```

Om de string te kunnen uitlezen is wel een groter programma nodig.

Voorbeeld 6:

```
5 LET A$="3ZON4MAAN4DINS5WOENS6DONDER4VRIJ5ZATER"
10 LET P=1
20 FOR I=1 TO 7
30 LET A=VAL A$(P)
40 LET B$=A$(P+1 TO A+P)
50 LET P=P+A+1
60 PRINT B$+"DAG"
70 NEXT I
```

Het verschil tussen beide methodes is duidelijk. Aan de ene kant wordt ruimte gewonnen, maar aan de andere kant moet weer programmaruimte worden ingeleverd. Als laatste zal nog een voorbeeld worden gegeven met numerieke gegevens.

Stel men wil een berekening uitvoeren met een aantal percentages, waarvan bekend is dat er maar twee cijfers achter de komma staan. De gebruikte percentages zijn 2,00%, 2,75%, 37,85%, 11,00% en 12,75%. Bij de oplossing van dit probleem zal gebruik worden gemaakt van de eerste oplossings-

methode. Alleen wordt er nu niet met spaties opgevuld, maar met nullen. De variabele A$ (het interne bestand) gaat er dan als volgt uitzien:

```
A$="020002753785110012 75"
```

Zoals men ziet, zijn er 'voorloopnullen' geplaatst in plaats van spaties. Er is geen decimale punt opgenomen omdat dit overbodige informatie is. Bij het uitlezen van het percentage wordt door middel van een deling door honderd, de decimale punt weer in het percentage geplaatst.

```
10 LET A$="020002753785110012 75"
20 FOR I=1 TO 5
30 LET A=VAL A$(I*4-3 TO I*4)/100
40 PRINT "PERCENTAGE ";I;"= ";A
50 NEXT I
```

In een aantal programma's wordt gebruik gemaakt van PEEK- en POKE-statements. Deze twee statements zijn voor wat hun werking betreft vrij uniek in de hogere programmeertalen. De gebruiker wordt met deze twee statements in staat gesteld afzonderlijke bytes te bekijken (PEEK) of te wijzigen (POKE).
De gebruiker daalt dus af naar het machinetaal-niveau en dat betekent dat hij ook een aantal zaken zal moeten weten. Ten eerste houdt de ZX81 een aantal één- en twee-bytes-woorden vast voor huishoudelijke gegevens waarmee erg leuk gemanipuleerd kan worden. In hoofdstuk 28 van de Engelse handleiding wordt een opsomming gegeven van een aantal van deze één- en twee-bytes-woorden, tegelijkertijd wordt er ook een summiere verklaring gegeven.
Voordat er verder wordt ingegaan op een aantal toepassingen van deze gegevens, eerst een kleine uitleg hoe twee-bytes-getallen staan opgeslagen. Dit geldt overigens niet alleen voor de ZX81, maar voor alle computers, waarin een Z80 als microprocessor wordt gebruikt. Bij een twee-bytes-woord of -getal staat eerst het minst significante byte en daarna het meest significante byte, of met andere woorden: het kleinste gedeelte van het getal staat voorop en het grootste gedeelte komt er achteraan. Dat is eigenlijk wel erg vreemd, omdat men het in het dagelijks leven net andersom gewend is.
In de ZX81 wordt gebruik gemaakt van een beeldteller, met behulp waarvan het beeldscherm vijftig keer per seconde wordt opgebouwd. Het adres waar deze beeldteller staat is 16436 en 16437. Deze beeldteller is te gebruiken om een vrij nauwkeurige klok in de ZX81 te bouwen. Het eerste dat men moet doen is de beeldteller op een bekende waarde zetten. In de programma's is gekozen voor de grootst mogelijke binaire waarde, namelijk $2^{16} - 1$, ofwel als decimaal getal 65.535. Binair uitgeschreven zijn dat 16 enen. Nu gaan er maar 8 bits in een byte. Dus in elke byte moeten 8 enen komen, omgerekend naar het tientallig stelsel is dit 255.
Uit de Engelse handleiding (PAUSE blz. 199) blijkt dat elke seconde de beeldteller met een waarde 50 zakt. Het beeldscherm wordt vijftig keer per seconde opgebouwd. Wanneer men de inhoud van de beeldschermteller van de startwaarde 65.535 aftrekt en door vijftig deelt, heeft men als resultaat een aantal seconden. Dit aantal seconden is het tijdsverloop vanaf het moment dat de beeldschermteller op een bekende waarde gesteld

werd tot het moment waarop de beeldteller werd uitgelezen. Een programma kan dit verduidelijken.

Voorbeeld 7:

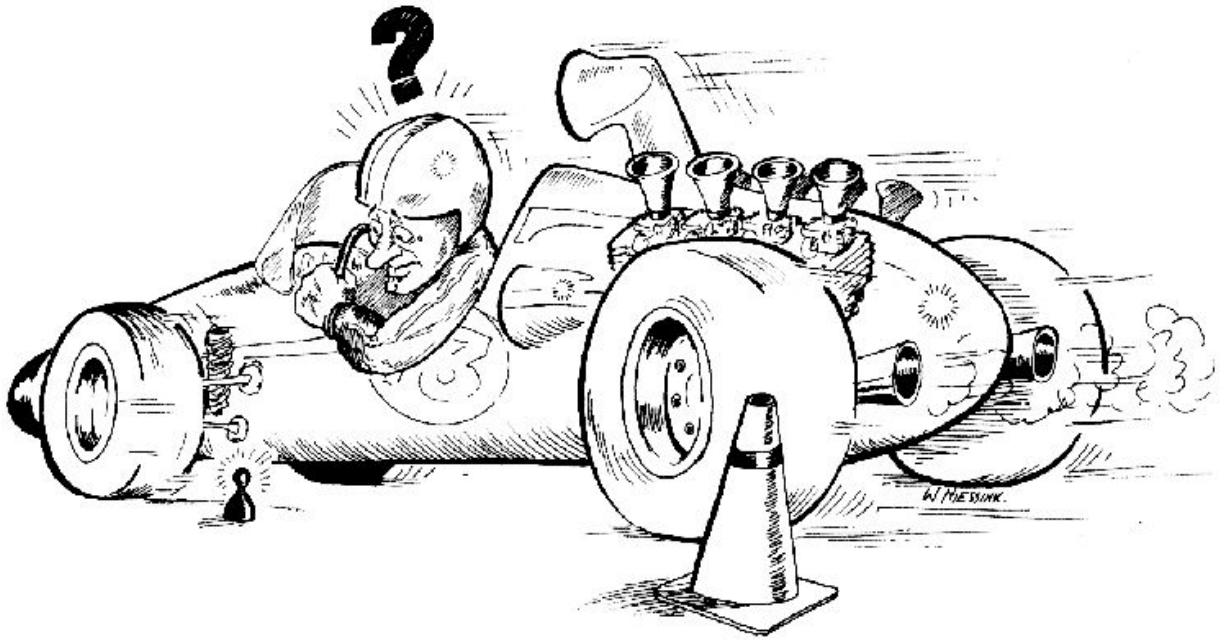
```
10 POKE 16436,255
20 POKE 16437,255
30 LET T=INT((65535-PEEK 16436-256*PEEK 16437)/50)
40 PRINT AT 10,0;"VERSTREKEN TIJD ";T
50 GOTO 30
```

Men dient er wel op bedacht te zijn, dat men in het programma geen gebruik maakt van het PAUSE-statement. Dit statement manipuleert eveneens de beeldteller.
De ZX81 heeft eigenlijk geen apart schermgeheugen zoals bij andere microcomputers het geval is. Het maakt eigenlijk gebruik van een dynamisch schermgeheugen. Het groeit en krimpt al naar gelang de behoefte. Het grote voordeel is dat op zo'n manier alle beschikbare ruimte ook optimaal wordt gebruikt. Een nadeel is wel dat men niet rechtstreeks naar het scherm kan PEEKen en POKEn, zodat het programmeren van een realtime-spel iets moeilijker wordt. Hier brengt hoofdstuk 29 van de Engelstalige handleiding de oplossing. Geheugenlocatie 16398 en 16399 levert het adres waar het volgende karakter staat dat zal worden afgedrukt. Met deze informatie is het mogelijk na te gaan, wat het volgende karakter zal zijn dat wordt afgedrukt. Afhankelijk van dat karakter kan dan een bepaalde actie worden ondernomen. Onder andere in het programma 'Spookrijder' wordt hiervan gebruik gemaakt.
Met PEEK 16399 * 256 + PEEK 16398 heeft men het decimale adres van het volgende te printen karakter. Om nu de inhoud te pakken te krijgen, zal men opnieuw een PEEK-opdracht moeten geven met het hierboven bepaalde adres. De programmaregel ziet er dan als volgt uit:

```
PEEK (PEEK 16399*256+PEEK 16398)
```

3. Spelprogramma's

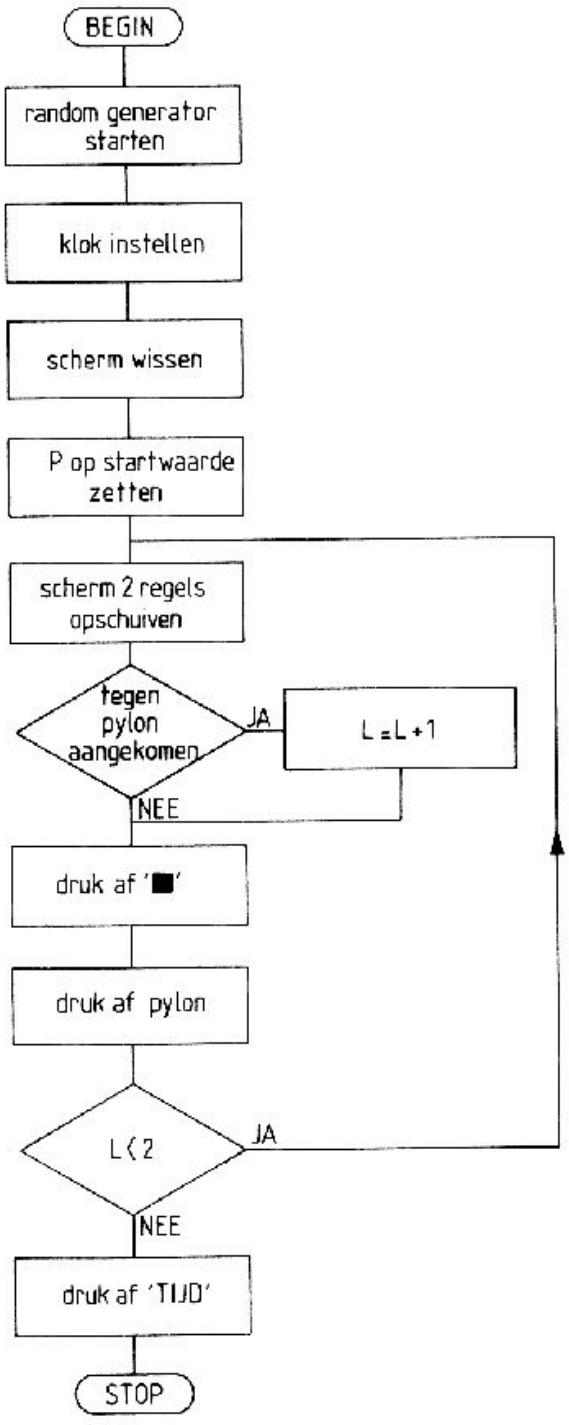
3.1 PYLONRACE



In dit spel moet de speler, die wordt voorgesteld door een zwart blokje, proberen zoveel mogelijk pylons met vlaggetjes te omzeilen. Om zijn pylonracer te besturen kan hij op de M- of N-toets drukken. Drukt hij op de N-toets dan gaat de pylonracer naar links, drukt hij op de M-toets dan gaat de racer naar rechts. Hij mag maar één keer tegen een pylon aankomen. Bij de tweede keer is hij af en stopt het spel. Wanneer het spel beëindigd is, wordt op het scherm aangegeven hoeveel seconden de speler in de baan is gebleven. Het zal zeker niet meevallen lang in de baan te blijven. Het record ligt nu op ongeveer twee minuten!

Werking van het programma

Op regel 7 en 8 wordt de klok op een bekende waarde gezet. Het decimale getal in de twee geheugenlocaties komt overeen met 256*255+255.
Om beweging in het beeld te krijgen wordt het scherm elke keer twee regels omhoog geduwd met het SCROLL-statement. Daarna wordt het



Principeschema van 'Pylonrace'.

eerste gedeelte van het PRINT-statement uitgevoerd. Net op het moment dat het zwarte blokje afgedrukt zal worden, wordt er gestopt. Op dat moment wordt er gekeken of het zwarte blokje niet toevallig op de plaats wordt afgedrukt waar een pylon staat. De pylons worden voorgesteld door een 'I'. Kijkt men in appendix A van de Engelstalige handleiding (blz. 182) dan ziet men dat de I een decimale waarde heeft van 46. Blijkt het af te drukken symbool die waarde 46 te hebben dan wordt de variabele L met één verhoogd.
Na deze actie wordt ■ afgedrukt en onderaan het beeldscherm wordt op een willekeurige plaats weer een pylon geplaatst. Overigens, de plaats waar de pylon wordt geplaatst is maar gedeeltelijk willekeurig. Hij ligt maximaal

maar vijf kolommen af van de plaats waar op dat moment de pylonracer staat.
Op regel 50 wordt door het programma naar het toetsenbord gekeken of de M- of N-toets is ingedrukt. Wordt aan beide voorwaarden voldaan die tussen de haakjes staan, dan levert dat een één op. Wordt er niet aan de voorwaarden voldaan, dan levert dat een nul op; daarbij zijn de voorwaarden wederzijds uitsluitend. Wordt er aan de ene voorwaarde voldaan, dan mag en kan er niet aan de andere voorwaarde worden voldaan. Is de N-toets ingedrukt en de variabele P is groter dan of gelijk aan 5 dan wordt er van de waarde van P één afgetrokken. Wordt bijvoorbeeld de M-toets ingedrukt en P is groter dan 24 dan levert dat als resultaat een nul op. De feitelijke inhoud van P wijzigt dan niet!
Het spel stopt als blijkt dat de pylonracer twee keer tegen een pylon is aangekomen. Op regel 75 wordt gekeken of dit laatste het geval is. Verandert men het getal 2 in een andere waarde, bijvoorbeeld 5 of 6, dan kan men dus meerdere keren tegen een pylon aankomen zonder dat het spel stopt. Ten slotte wordt op regel 80 de verstreken tijd afgedrukt.

Programma

```
1 LET L=0
2 RAND
7 POKE 16436,255
8 POKE 16437,255
9 CLS
10 LET P=15
15 SCROLL
17 SCROLL
20 PRINT AT 0,P;
30 IF PEEK (256*PEEK 16399+PEEK 16398)=46 THEN LET
   L=L+1
40 PRINT "■";AT 14,P+RND*10-5;"I"
50 LET P=P-(INKEY$="N" AND P>=5)+(INKEY$="M" AND
   P<=24)
75 IF L<2 THEN GOTO 15
80 PRINT AT 19,6;"TIJD ";INT((65535-PEEK 16437*256 -
   PEEK 16436)/50)
```

3.2 ONTSNAPPING

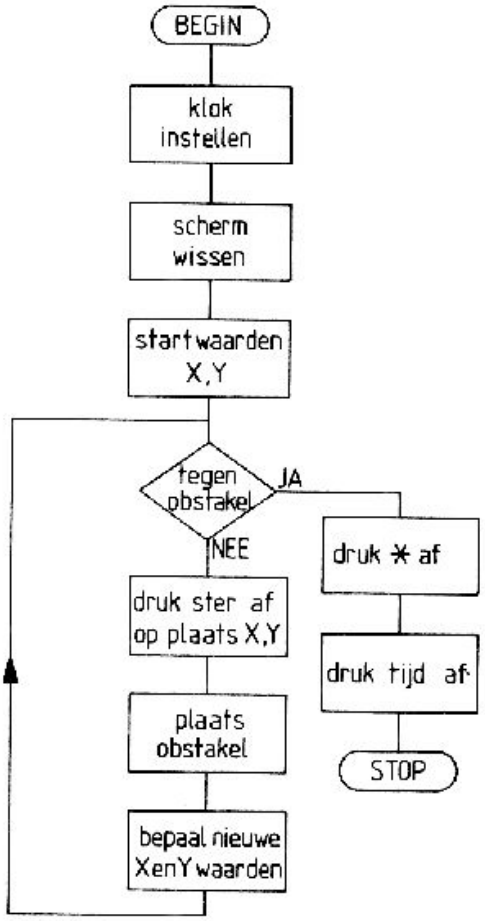
In dit spel is de gebruiker opgesloten in een onzichtbaar labyrint. Er is geen uitweg mogelijk, maar men moet proberen zolang mogelijk in leven te blijven. Het programma plaatst muren rond de gevangene, zodat de doorgang wordt geblokkeerd. De speler moet proberen te voorkomen dat hij ingesloten raakt.
Hoe meer obstakels hij weet te vermijden, hoe langer hij in het spel blijft. Zodra de speler tegen een obstakel oploopt, is het afgelopen. Het programma geeft dit aan met een zwart blokje met een sterretje er in.
Het sterretje dat de speler voorstelt, kan worden bestuurd. Drukt men de A-toets in dan gaat het sterretje omhoog, met de Z-toets gaat het sterretje

U heeft zeker toevallig ook geen vuur ???

omlaag, met de M-toets gaat het sterretje naar rechts en ten slotte met de N-toets gaat het sterretje naar links.
Aan het einde van het spel wordt aangegeven hoe lang de speler in leven is gebleven.

Werking van het programma

De werking van het programma vertoont overeenkomsten met het program-



Principeschema van 'Ontsnapping'.

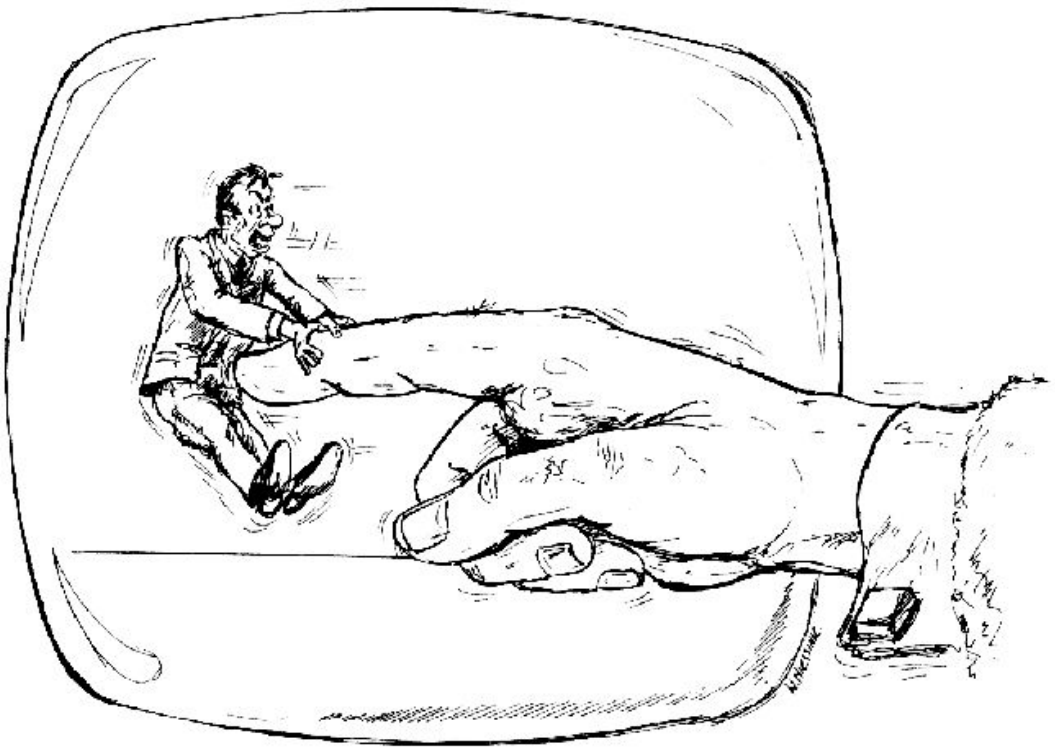
ma 'Pylonrace'. Er is één opmerkelijk verschil. De bewegingen in 'Pylonrace' werden veroorzaakt door de dubbele SCROLL-opdracht. Het beeld werd aan de onderzijde iedere keer twee regels omhoog geduwd. Deze twee opdrachten zorgden er voor dat het leek alsof de pylons zich naar de racer toe bewogen. In dit spel 'Ontsnapping' wordt het beeld stilgehouden en wordt de gevangene (*) bewogen.
De beweging wordt gesuggereerd door eerst de ster af te drukken en na enkele tienden seconden de plaats waar de ster stond weer te wissen. In werkelijkheid wordt een spatie afgedrukt op de plaats waar de ster stond. Op deze manier lijkt het alsof de ster zich over het beeldscherm beweegt. Van deze techniek wordt in meerdere programma's gebruik gemaakt en in het programma dat hierna komt wordt gedemonstreerd hoe men grote figuren over het beeldscherm kan verplaatsen. Het teken □ stelt een spatie voor.

Programma

```
1 POKE 16437,255
2 POKE 16436,255
9 CLS
10 RAND
11 LET X=10
12 LET Y=15
20 PRINT AT X,Y;
30 IF PEEK (256*PEEK 16399+PEEK 16398)=128 THEN GOTO 70
```

```
40 PRINT "*";AT X+RND*6-3,Y+RND*6-3;"■"
41 PRINT AT X,Y;"□"
50 LET Y=Y-(INKEY$="N" AND Y>=2)+(INKEY$="M" AND
   Y<=30)
55 LET X=X-(INKEY$="A" AND X>=2)+(INKEY$="Z" AND
   X<=20)
60 GOTO 20
70 POKE (PEEK 16399*256+PEEK 16398),151
80 PRINT AT 21,0;"TIJD ";INT((65535-PEEK 16436-256
   *PEEK 16437)/50)
```

3.3 VERPLAATSEN VAN EEN POPPETJE



Dat niet alléén een karakter over het beeldscherm verplaatst kan worden, bewijst dit programma. In dit programma worden een aantal technieken gebruikt om de beweging redelijk na te bootsen. Wanneer men in staat is in machinetaal te programmeren kunnen nog vloeiender beelden ontstaan zonder het hinderlijke geflikker. In principe blijft de techniek hetzelfde, of er nu in BASIC of in machinetaal wordt geprogrammeerd.
Een van de eerste problemen die men tegenkomt is dat bij het bewegen van figuren, die uit meerdere regels bestaan, men ook meerdere regels zal moeten wissen. Dit heeft tot gevolg dat er in BASIC een flikkerend beeld ontstaat, hetgeen door menigeen als storend wordt ervaren. Men kan dit flikkeren sterk beperken door het beeld alleen te wissen als het niet wordt verplaatst. Van deze techniek is in dit programma uitgegaan. Blijkt dat de X- en Y-coördinaten van het poppetje niet zijn gewijzigd, dan wordt het beeld niet gewist. Zijn de X- en de Y-waarden wel gewijzigd, dan wordt eerst het poppetje op de oude plaats gewist en daarna op de nieuwe plaats opgebouwd.

Nu het tweede probleem. Het poppetje is drie karakters breed en drie karakters hoog. Dat betekent dat de eerste regel van het poppetje op regel X wordt afgedrukt, de tweede regel op regel X+1 en ten slotte de derde regel op regel X+2. Door er nu voor te zorgen dat elke regel van het poppetje uit een gelijk aantal karakters bestaat, is de relatieve kolom voor alle regels hetzelfde. Door nu X en Y te specificeren, is daarmee de plaats op het beeldscherm vastgelegd en kan worden afgedrukt.
Stel X=10. Regel 1 van het poppetje wordt dan op regel 10 afgedrukt, de tweede regel op regel 10+1 en de derde regel op regel 10+2.

Regel X " □○□ "
Regel X+1 " ▬■▬ "
Regel X+2 " ◨□◧ "

Het wissen gaat feitelijk op dezelfde manier. Er wordt een string afgedrukt met drie spaties op de oude plaats van het poppetje. Dus op de oude regel X, regel X+1 en regel X+2. Als dat is gebeurd, is het poppetje verdwenen van het beeldscherm.
De oude X- en Y-waarden zijn belangrijk om te onthouden, omdat deze de plaats aangeven waar er moet worden gewist. Nadat men de oude X- en Y-waarden veilig heeft gesteld in de variabelen A en B, kunnen X en Y weer worden gewijzigd. Dit wijzigen van de X- en Y-waarden geschiedt op de bekende manier. Met de A- en de Z-toets wordt het plaatje in verticale richting bewogen en met de N- en M-toets wordt het plaatje in horizontale richting bewogen. Wanneer men de BREAK-toets indrukt stopt het programma.

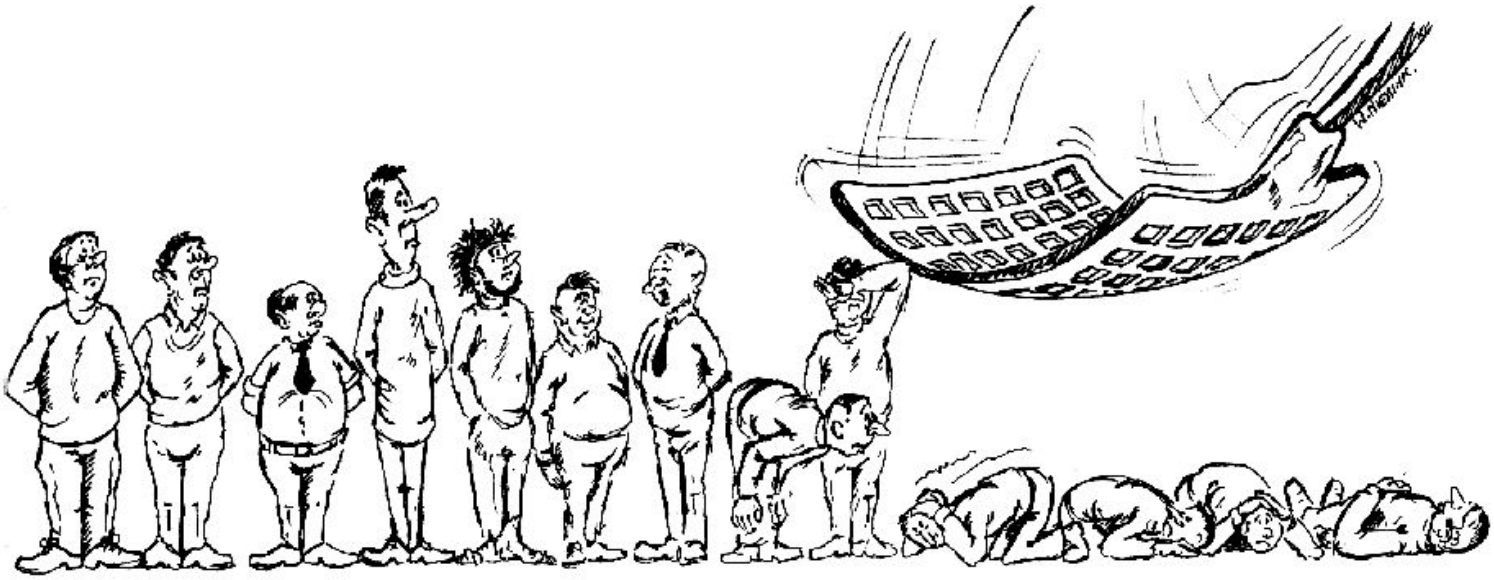
Programma

□ = spatie

```
1 CLS
3 LET X=15
4 LET Y=15
5 GOTO 31
6 IF A=X AND B=Y THEN GOTO 40
15 PRINT AT A,B;" □□□ ";AT A+1,B;" □□□ ";
   AT A+2,B;" □□□ "
20 PRINT AT X,Y;"□ ○ □";AT X+1,Y;" ▬■▬ ";
   AT X+2,Y;" ◨□◧ "
31 LET A=X
32 LET B=Y
40 LET X=X+(INKEY$="Z" AND X<=17)-(INKEY$="A" AND
   X>=1)
50 LET Y=Y+(INKEY$="M" AND Y<=26)-(INKEY$="N" AND
   Y>=1)
60 GOTO 6
```

Resultaat op het scherm

3.4 21 MANNETJES



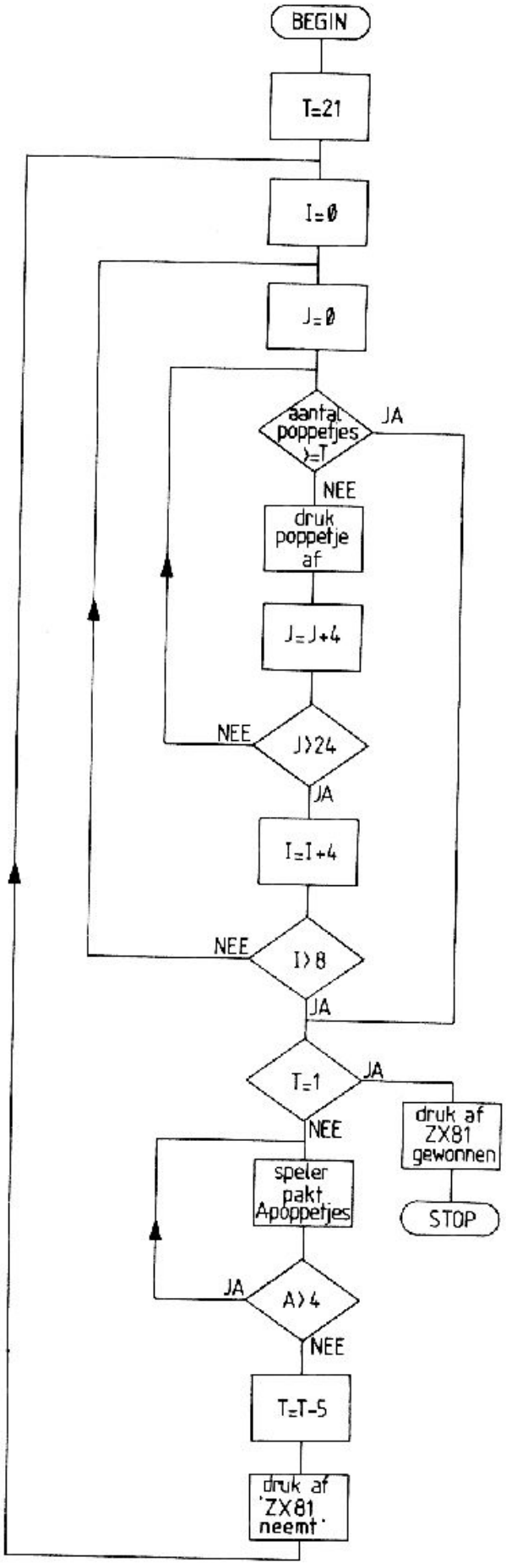
In dit spel speelt men tegen de computer, de ZX81. Het is de bedoeling dat de speler en de computer ieder een aantal mannetjes wegnemen. Dit mogen er maximaal vier zijn en minimaal één. Degene, die het laatste mannetje moet pakken heeft verloren! Het is dus uitkijken geblazen. De machine is beleefd en laat, zoals het hoort, de speler beginnen! Dit eenvoudige spel is gebaseerd op het bekende NIM-spel, dat in allerlei vormen bekend is.

Werking van het programma

Wanneer men het stroomschema op de volgende pagina bekijkt zal de werking wel volledig duidelijk zijn.
Heeft men het spel een aantal keren gespeeld dan zal men in de gaten krijgen dat de computer altijd wint. Dat heeft twee oorzaken. Ten eerste is het aantal weg te nemen mannetjes zo gekozen dat er altijd één moet overblijven. Ten tweede, het aantal weg te nemen mannetjes is beperkt tot zeven mannetjes per speler per keer. Het programma is zo gemaakt dat er per spelbeurt, speler en computer samen, in totaal vijf mannetjes verdwijnen. Pakt bijvoorbeeld de speler er drie, dan zal de computer er maar twee pakken. Pakt de speler er maar één, dan pakt de computer er vier. Wanneer men nu gaat rekenen zijn er in totaal vier beurten voordat er één mannetje overblijft. De speler begint, dus de computer wint altijd, omdat na zijn tweede beurt er één mannetje overblijft. Namelijk, vier beurten van ieder vijf mannetjes is twintig mannetjes en er is begonnen met eenentwintig mannetjes. Het teken ☐ is een spatie.

Programma

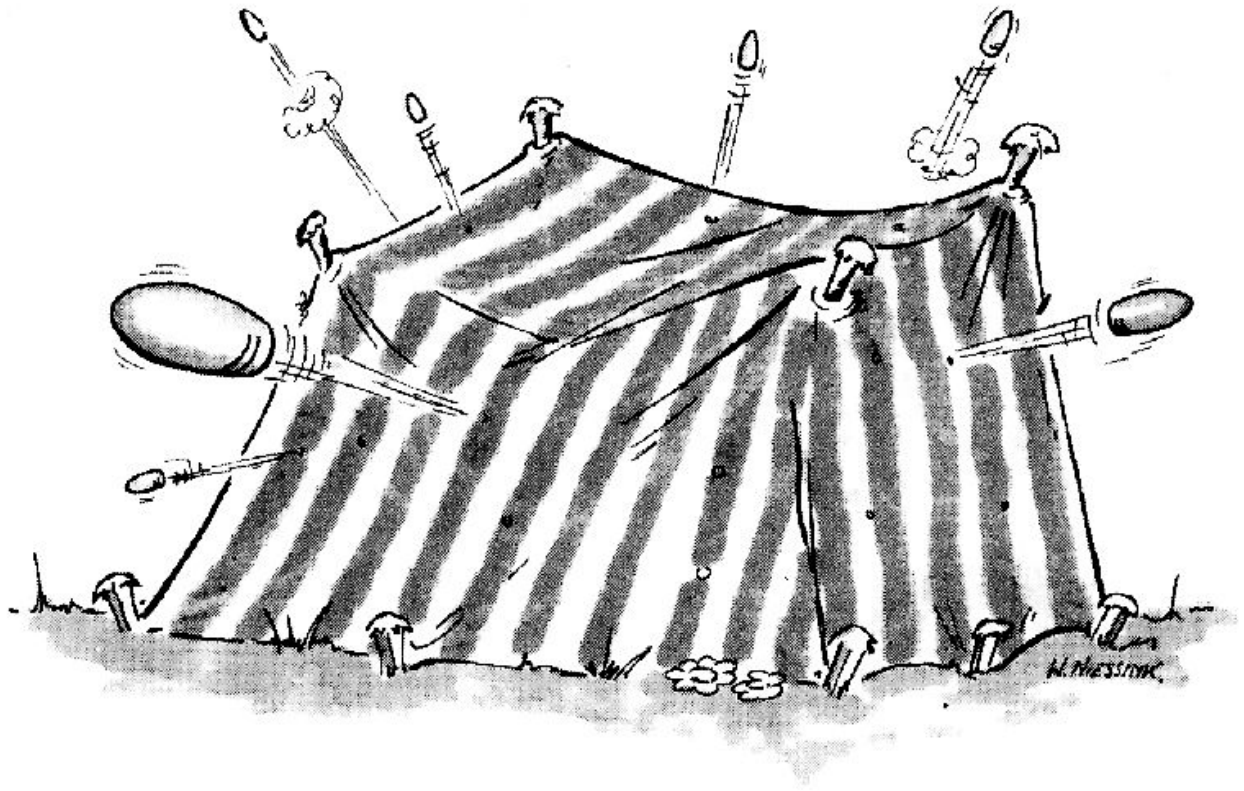
```
2 LET T=21
3 CLS
10 FOR I=0 TO 8 STEP 4
```



Principeschema van '21 mannetjes'.

```
12 FOR J=0 TO 24 STEP 4
13 IF I/4*7+J/4>=T THEN GOTO 31
20 PRINT AT I+2,J;"□○□";AT I+3,J;" ▄▄▄ ";
   AT I+4,J;" ▌□▐ "
25 NEXT J
30 NEXT I
31 IF T=1 THEN GOTO 70
35 PRINT "HOEVEEL PAK JIJ ";
40 INPUT A
41 PRINT A
45 IF A>4 THEN GOTO 35
50 LET T=T-5
51 CLS
55 PRINT "ZX81 PAKT ";5-A;"ER ZIJN ER NOG ";T
60 GOTO 10
70 PRINT "ZX81 HEEFT GEWONNEN "
```

3.5 SCHIETTENT



Op het beeldscherm wordt een schiettent opgebouwd, zoals deze vroeger op de kermis stond. Het is de bedoeling om de pluimpjes boven de letters te raken. Linksboven in het beeld loopt de tijd. Men heeft maximaal 60 seconden om een zo hoog mogelijke score te bereiken. De score wordt rechtsboven in het beeld bijgehouden. In het midden van het beeldscherm zijn de letters A tot en met I afgebeeld. Er boven loopt een zwart blokje. Het is de bedoeling om de toets in te drukken, die wordt aangegeven door het zwarte blokje.
Het lijkt vrij eenvoudig, maar of het meevalt staat te bezien. Heeft men

raak geschoten, dus de betreffende toets op tijd ingedrukt, dan wordt de score met één verhoogd. In het zwarte blokje verschijnt dan eventjes een klein wit sterretje.

Werking van het programma

Het programma begint met het instellen van de bekende klok op de waarde 65535 oftewel 255*256+255 (regel 1 en 2). Daarna wordt de scoreteller S op nul gesteld en de variabele G wordt op één gesteld. Deze variabele G geeft de relatieve plaats van de zwarte stip aan. Dat wil zeggen dat er niet rechtstreeks naar de kolom wordt verwezen waar de stip wordt afgedrukt, maar naar de letter waarboven de stip staat. Wanneer de variabele G de waarde één heeft, wordt er gewezen naar de letter A, die als eerste in de regel staat. Heeft de variabele G de waarde drie, dan wordt er gewezen naar de letter C.
Op regel 60 wordt de zwarte stip boven de letter afgedrukt. Dit wordt gerealiseerd met behulp van een rekentruc waarbij wordt uitgegaan van de derde kolom. Tussen alle karakters zitten twee spaties plus de letter zelf. Dit geeft drie posities (kolommen). Kent men de plaats van de letter A dan kan men door een aantal malen drie kolommen op te schuiven precies boven de letter uitkomen, die op dat moment aan de beurt is. De eerder genoemde variabele G neemt de waarde 1 t/m 9 aan. De door de speler in te voeren karakters zijn A t/m I; dit komt volgens appendix A op blz. 181 (Engelse handleiding) overeen met de decimale waarden 38 t/m 46. Wanneer men van het ingevoerde karakter de decimale waarde bepaalt en daar 37 van aftrekt, moet, indien het goede karakter is ingedrukt, dit overeenkomen met de waarde van variabele G.
Blijkt dit laatste het geval te zijn, dan wordt in het zwarte blokje een sterretje geplaatst en de score wordt met één verhoogd.
In alle gevallen wordt na deze actie, waarin al of niet de score moet worden verhoogd, de zwarte stip gewist en de variabele G met één verhoogd. Is G gelijk aan 10 geworden, dan wordt de variabele teruggesteld naar één. Een nieuwe ronde kan beginnen op voorwaarde dat de tijd niet verstreken is.

Programma

□ = spatie

```
1 POKE 16437,255
2 POKE 16436,255
3 LET G=1
5 LET S=0
6 PRINT AT 4,4;"A□□B□□C□□D□□E□□F□□G□□H□□I"
7 GOTO 91
10 PRINT AT 1,4;"TIJD ";T;AT 1,20;"SCORE ";S
60 PRINT AT 3,3+G*3-2;"■"
66 IF CODE INKEY$-37<>G THEN GOTO 80
70 PRINT AT 3,3+G*3-2;"▣"
75 LET S=S+1
80 PRINT AT 3,3+G*3-2;" □ "
81 LET G=G+1
```

```
90 IF G=10 THEN LET G=1
91 LET T=INT((65535-PEEK 16436-256*PEEK 16437)/50)
96 IF T<61 THEN GOTO 10
100 PRINT AT 10,6;"SPEL VOORBIJ "
```

3.6 SCHETSBLOK

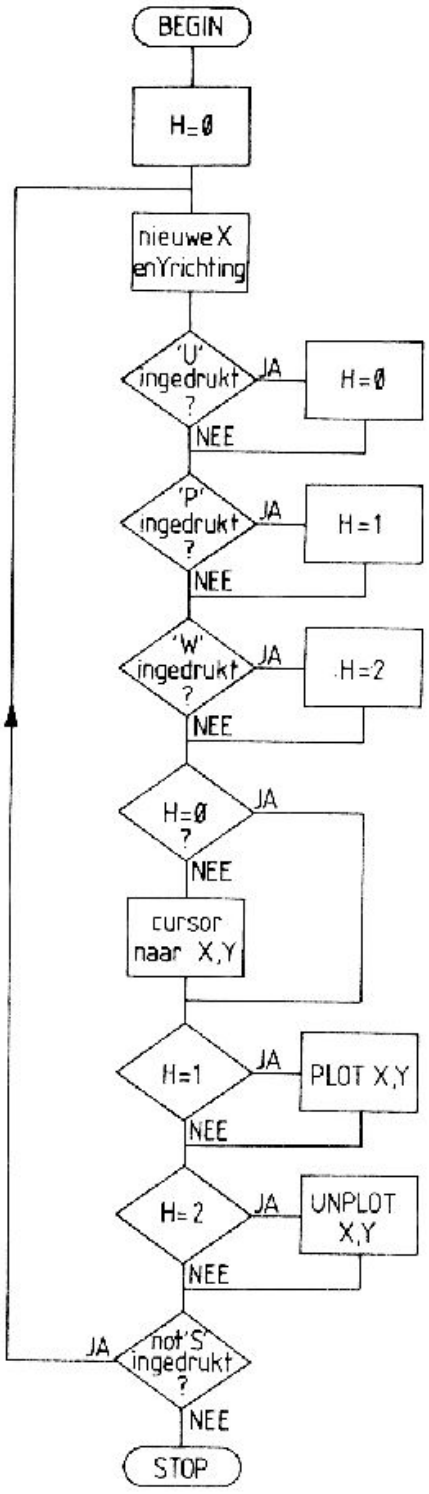
Met dit kleine programma kan men op het beeldscherm een tekening maken. Jammer genoeg kan de tekening niet op band worden opgeslagen. Heeft men een 16 K-uitvoering dan is die mogelijkheid wel aanwezig.
Het tekenen geschiedt met de M-, N-, Z- en A-toets. Drukt men de M-toets in dan beweegt de cursor naar rechts, met de N-toets gaat de cursor naar links, met de A-toets beweegt de cursor omhoog en ten slotte met de Z-toets gaat de cursor omlaag.
Door het indrukken van de P-toets gaat het programma een lijn op het beeldscherm tekenen en volgt daarbij de bewegingen van de cursor. Wil men het plaatje wissen, dan moet men de W-toets indrukken en wordt de plaats gewist waar de cursor staat. Drukt men de U-toets in dan wordt de cursor uitgeschakeld, maar blijven alle gedefinieerde toetsen gewoon werken. Dit uitschakelen van de cursor is ingebouwd om een rustig plaatje te krijgen zonder een flikkerende cursor. Drukt men ten slotte de S-toets in dan heeft men daarmee het programma beëindigd.

Werking van het programma

Met de uitleg hoe men het moet gebruiken en het stroomschema zoals afgebeeld op de volgende pagina moet duidelijk zijn hoe het programma in principe werkt. Het enige dat wel aardig is om bij wijze van experiment uit te voeren is regel 52 met regel 53 te verwisselen. Men ziet dan tegelijkertijd waarom de cursor op deze manier is opgebouwd.
Overigens laat het programma ook zien dat binnen de beperkte ruimte van de ZX81 1K-versie toch een volledig programma is op te bouwen. Alleen eist het enige transpiratie en denkwerk om dit te bereiken. Meestal komt het er op neer dat men zijn oorspronkelijk idee van het programma aardig in elkaar zal moeten drukken. Bij een ander type computer zou een gelijksoortig programma veel meer ruimte innemen.

Programma

```
5 LET X=15
10 LET Y=15
15 LET H=0
35 LET X=X-(INKEY$="N" AND X>0)+(INKEY$="M" AND X<63)
40 LET Y=Y-(INKEY$="Z" AND Y>0)+(INKEY$="A" AND Y<43)
42 IF INKEY$="U" THEN LET H=0
45 IF INKEY$="P" THEN LET H=1
50 IF INKEY$="W" THEN LET H=2
51 IF H=0 THEN GOTO 55
52 UNPLOT X,Y
```

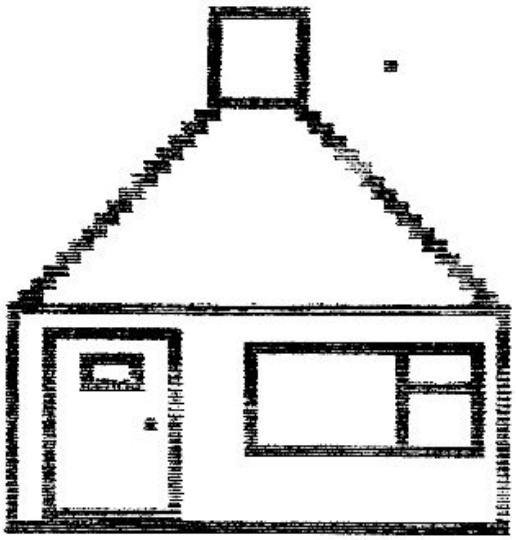


Principeschema van 'Schetsblok'.

Vervolg programma

```
53 PLOT X,Y
55 IF H=1 THEN PLOT X,Y
60 IF H=2 THEN UNPLOT X,Y
80 IF INKEY$<>"S" THEN GOTO 35
```

Voorbeeld



3.7 EENDENJACHT

Als eerste zou moeten worden opgemerkt dat dit programma in een 1 K-versie eigenlijk niet te realiseren is. Dit programma past maar net in de 1 K-versie en af en toe wil het er wel eens uitlopen. Dat het in dit boek toch is opgenomen is vooral te danken aan de aardige plaatjes, die het programma produceert. Overigens, de 16 K-bezitters kunnen het programma drastisch uitbreiden.
Wanneer men het programma heeft ingetoetst en gestart, verschijnt er ergens op het beeldscherm een gestileerde eend. Het is de bedoeling de eend te raken met een enorm ouderwets kanon. Het kanon staat zelf links buiten beeld, maar de kogelbaan is op het beeldscherm goed te volgen. De eend, die zo geschrokken is van het enorme kanon blijft stokstijf zitten.
Men geeft de vuurleiding van het kanon op, onder welke hoek men een schot wil lossen. De hoek moet liggen tussen 5 en 45 graden. In het programma is geen beveiliging ingebouwd tegen verkeerde invoer, dus er bestaat een kans dat bij foutieve invoer het programma stopt. Geeft men een hoek tussen de genoemde getallen dan ziet men de kogelbaan op het beeldscherm. Raakt men de eend, dan gaat deze als reactie daarop flikkeren. Om dit te stoppen moet men de BREAK-toets indrukken. Heeft men echter gemist dan kan men het nog eens proberen; de eend blijft rustig zitten. In het programma wordt niet expliciet om invoer gevraagd, maar als de '[L]' verschijnt, kan men de hoek ingeven.

Werking van het programma

Het ontwerp van de eend is ontstaan met het programma 'Schetsblok'. Mocht men het vervelend vinden om op een eend te schieten; niemand houdt u tegen om er een tank, een schip of iets anders van te maken. Het gaat in dit programma om een visuele weergave van de kogelbaan. Er moet bij worden vermeld, dat de tekening wel behoorlijk overtrokken is, maar dat is voor de duidelijkheid gedaan. Het berekenen en plotten van een

kogelbaan gaat niet zomaar en zeker niet op een ZX81 met maar 1K. Er zullen een aantal behoorlijke simplificaties en trucs gebruikt moeten worden. Heeft men echter de kunstgrepen door dan kan men dezelfde trucs ook in de eigen programmatuur toepassen.
Om het aantal regels rekenwerk in het programma drastisch te beperken, wordt er van uitgegaan dat elke kogelbaan een zelfde hoogte bereikt. In werkelijkheid is dit natuurlijk niet zo, maar het scheelt aanzienlijk wat ruimte in de ZX81 en het doet aan het plaatje niets af. Een kogel beschrijft altijd een parabolische baan. Om zo'n baan te kunnen plotten, houdt dit in dat de Y-waarden van de functie eerst kwadratisch oplopen en daarna weer dalen. Dit kan worden bereikt met een eenvoudige formule: J*(T-J). Waarbij T de maximale waarde voorstelt tot waar J kan oplopen.

Men moet er wel op letten dat J bij nul begint. In het programma is gekozen voor T=25, dit is voornamelijk gedaan uit het oogpunt van ruimtegebrek. Wanneer T de waarde 25 heeft, betekent dat dat de maximale waarde van Y wordt bereikt, als J gelijk is aan 13. De Y-waarde is dan 156 [(13*(25-13)]. De Y-waarde in een PLOT-statement kan maximaal 40 worden. Om ervoor te zorgen dat de Y-waarde de 40 nooit overschrijdt, dient de Y-waarde te worden vermenigvuldigd met een correctiefactor, zodat het resultaat op 40 of nagenoeg 40 uitkomt. De correctiefactor is 40/156 oftewel 0,256. Wanneer nogmaals de hele formule voor de Y-waarde wordt ingevuld, komt er te staan, indien J gelijk is aan 13:
Y=13*(25-13)*0.256=>Y=39.936. Zeg maar nagenoeg 40.

Voor de X-waarde geldt ongeveer iets dergelijks, zij het dat daar de sinusfunctie in de berekening sluipt met al zijn problemen.
De formule om de worpafstand te bepalen, is sterk vereenvoudigd. In dit programma wordt voor het gemak gerekend met sin 2α, waarbij α de werphoek voorstelt. De grootste worpafstand wordt bereikt bij een hoek van 45 graden, aannemende dat de werpkracht gelijk blijft en daar wordt in het programma van uitgegaan. De sinus van tweemaal 45 graden is gelijk aan 1.
Nu loopt de variabele J van 0 op naar 25. De maximale X-waarde die bereikt moet worden is 50. Het voorgaande houdt in dat de correctiefactor voor de X-waarde 2 moet zijn. Tot zover is het allemaal nog vrij eenvoudig, maar nu komt het. Het vervelende is dat bij de ZX81 de sinus en andere goniometrische functies worden berekend in radialen en niet in graden. Dit eist conversie van graden naar radialen.
Zonder verdere uitleg kan deze conversiefactor gelijk aan $\pi/180$ worden gesteld. Nu is π een irrationeel getal, maar voor het gemak wordt vaak met 3,14159 gerekend. Nu wordt de formule: SINA*3.14159/180*2 (de twee is van de formule worpafstand). Wanneer de getallen met elkaar vermenigvuldigd en gedeeld worden, ontstaat ten slotte de formule SINA*0.034906. Wanneer men nu alle gegevens over de X-waarde combineert, komt men tot de formule S=J*2*SIN(A*0.034906).
Met behulp van de in de vorige alinea's bepaalde formules is de kogelbaan tussen 5 en 45 graden te plotten. Het enige dat nu nog moet worden uitgezocht, is of de kogel wel of niet de eend heeft geraakt? De eend wordt in een bepaalde kolom geplaatst, terwijl de kogelbaan met het PLOT-statement wordt gerealiseerd. Op blz. 123 van de Engelse handleiding blijkt dat twee pixels overeenkomen met een printkolom. Dit is dan ook de reden dat de berekende S-waarde door twee wordt gedeeld, nadat de lus is verlaten. De eend zelf is drie karakters breed, maar er zit een spatie voor, daarom wordt getest of S/2 ligt tussen Z+1 en Z+4.

Programma

(□ = spatie)

```
25 RAND
30 LET T=25
35 LET Z=15+INT(RND*9)
40 PRINT AT 19,Z;" □▄▌□□□□ ";AT 20,Z;
   " □□▌▌▄□ "
45 INPUT A
50 FOR J=0 TO T
55 LET S=J*2*SIN(A*.034906)
60 PLOT S,J*(T-J)*.256
65 UNPLOT S,J*(T-J)*.256
70 NEXT J
75 IF S/2<Z+1 OR S/2>Z+4 THEN GOTO 45
80 PRINT AT 16,Z;"RAAK"
90 PRINT AT 19,Z;" □▄▌□□□□ ";AT 20,Z;
   " □□▌▌▄□ "
95 FOR J=1 TO 15
100 NEXT J
105 PRINT AT 19,Z;" ▌▄▌▌▌▌ ";AT 20,Z;" ▌▌□□▄▌ "
110 GOTO 90
```

3.8 MASTERMIND

Mastermind is zo'n bekend spel, dat de uitleg beperkt kan blijven tot hoe het spel met deze machine moet worden gespeeld.
De computer genereert een geheime code, die bestaat uit vier verschillende cijfers, die samen een getal vormen. Het is de bedoeling dat de speler deze code breekt. De speler geeft een viercijferig getal, waarvan hij denkt dat dat de geheime code is. Het programma gaat dit ingevoerde getal vergelijken met de geheime code. Komt een cijfer voor en de plaats van dat cijfer komt overeen met de plaats in de geheime code, dan wordt er een zwart blokje afgedrukt. Komt alleen het cijfer voor, maar is de plaats onjuist, dan geeft het programma dit aan door een ster af te drukken. Het moet mogelijk zijn om binnen vijf à zes zetten de code te breken.

Werking van het programma

Tussen regel 10 en 55 wordt de geheime code aangemaakt. Het is een stukje programma dat bestaat uit twee in elkaar geneste lussen. In de buitenste lus wordt een willekeurig cijfer getrokken tussen 1 en 9. Dit cijfer wordt toegewezen aan het I-de element van matrix A. Daarna wordt de tweede lus ingegaan, die controleert of het getrokken cijfer al in gebruik is. Is het cijfer al eens getrokken dan wordt er een nieuw cijfer getrokken. Wordt ten slotte de binnenste lus verlaten, dan betekent dit dat het getrokken cijfer nog niet voorkomt. Op dat moment wordt het cijfer toegevoegd aan de geheime code. Wordt de buitenste lus verlaten dan is de geheime code aangemaakt. Het programma vraagt om invoer. Vanwege ruimtegebrek wordt niet gecontroleerd of het ingevoerde getal ook inderdaad viercijferig is. Het getal wordt direct vergeleken met de geheime code; komt het getal overeen dan is het programma afgelopen. Het programma meldt dit met een kort 'JA'.
In het andere geval zal het programma een scanningsmechanisme in werking brengen. Bij deze methode wordt iedere keer een cijfer van het ingevoerde getal afgezonderd. Dit afzonderen geschiedt van achteren af. Wanneer blijkt dat het afzonderlijke getal voorkomt, wordt in de PRINT-opdracht uitgemaakt of toevallig ook de plaats goed was. Is het cijfer gescanned dan wordt het ingevoerde getal ontdaan van dit cijfer, dit gebeurt in regel 110. Dit gaat net zolang door totdat alle cijfers aan de beurt zijn geweest. Is het programma in deze tak terecht gekomen, dan betekent dit ook dat de speler een foutief getal gegeven had, hij kan het dus opnieuw proberen.

Programma

```
5 RAND
6 LET F=0
7 LET Q=10
10 DIM A(4)
15 FOR I=1 TO 4
20 LET A(I)=INT(RND*9+1)
30 IF I=1 THEN GOTO 50
35 FOR J=1 TO I-1
```

```
40 IF A(I)=A(J) THEN GOTO 20
45 NEXT J
50 LET F=F*Q+A(I)
55 NEXT I
65 PRINT "GETAL"
70 INPUT C
75 PRINT C
80 IF C=F THEN GOTO 125
82 LET A=C
85 FOR I=1 TO 4
90 LET A=A-INT(A/Q)*Q
95 FOR J=4 TO Q/Q STEP -Q/Q
100 IF A<>A(J) THEN GOTO 105
102 PRINT " ■□" AND 5-I=J;"*□" AND 5-I<>J;
105 NEXT J
110 LET A=INT(C/Q**I)
115 NEXT I
120 GOTO 65
125 PRINT "JA"
```

3.9 HOGER - LAGER

Het spelletje 'Hoger-lager' is een leuk spelletje, met name voor kinderen, voor wie de ZX81 net zo gewoon is als een zakrekenmachine.
Het programma trekt een willekeurig getal tussen 1 en 100. De speler doet hetzelfde, voor alle zekerheid kan hij dit getal op een papiertje schrijven. Daarna vraagt de computer om een getal, waarvan de speler denkt dat de computer dat getal in gedachten heeft. Is het getal gelijk aan het getal dat de computer in gedachten had dan is de reactie van het programma een kort 'JA'. Was het getal fout dan geeft de ZX81 een aanwijzing of zijn getal hoger of lager is dan de opgegeven waarde.
Daarna geeft de computer een getal, waarvan hij denkt dat de speler dat in gedachten heeft. Wanneer het getal overeenkomt met het getal van de speler moet men een 1 invoeren. De computer zal reageren door zijn getal bekend te maken. Is het getal van de computer onjuist dan geeft de speler met een 2 aan dat zijn getal hoger ligt en met een 3 wordt aangegeven dat zijn getal lager ligt. Degene, die als eerste het getal heeft gevonden, is winnaar.

Werking van het programma

De opbouw van het programma is zeer eenvoudig, hoewel een gedeelte van een binair zoekproces in het programma is ingebouwd.
De variabele G bevat de waarde waarvan de computer denkt dat dat het getal is dat de speler in gedachten heeft. Tegelijkertijd wordt een onder- en bovengrens vastgesteld, waartussen door het programma gekozen wordt. Het kiezen van de variabele G geschiedt dus niet willekeurig, maar wordt gestuurd door de antwoorden van de speler. Hoe, zal hierna blijken.
Overigens wordt er in het programma van uitgegaan dat de speler eerlijk is en zijn gekozen getal niet stiekum verandert of foutieve antwoorden geeft.

Geeft de speler de computer op dat zijn getal hoger ligt dan het getal dat de computer aangeeft dan wordt de ondergrens, de variabele B, opgetrokken naar de waarde van het getal dat de computer de speler gaf. Geeft de speler lager, dan wordt de bovengrens, de variabele V, omlaag getrokken naar de waarde van het getal dat de computer de speler gaf. Daarna bepaalt het programma in regel 85 opnieuw een getal waarvan de computer denkt dat de speler dat getal in gedachten heeft. Dit getal ontstaat door ondergrens en bovengrens bij elkaar te tellen, door twee te delen en ten slotte de cijfers achter de komma weg te laten. Meestal heeft de computer binnen zo'n zeven à acht stappen het getal gevonden op voorwaarde dat de speler het getal van de computer niet heeft gevonden.

Programma

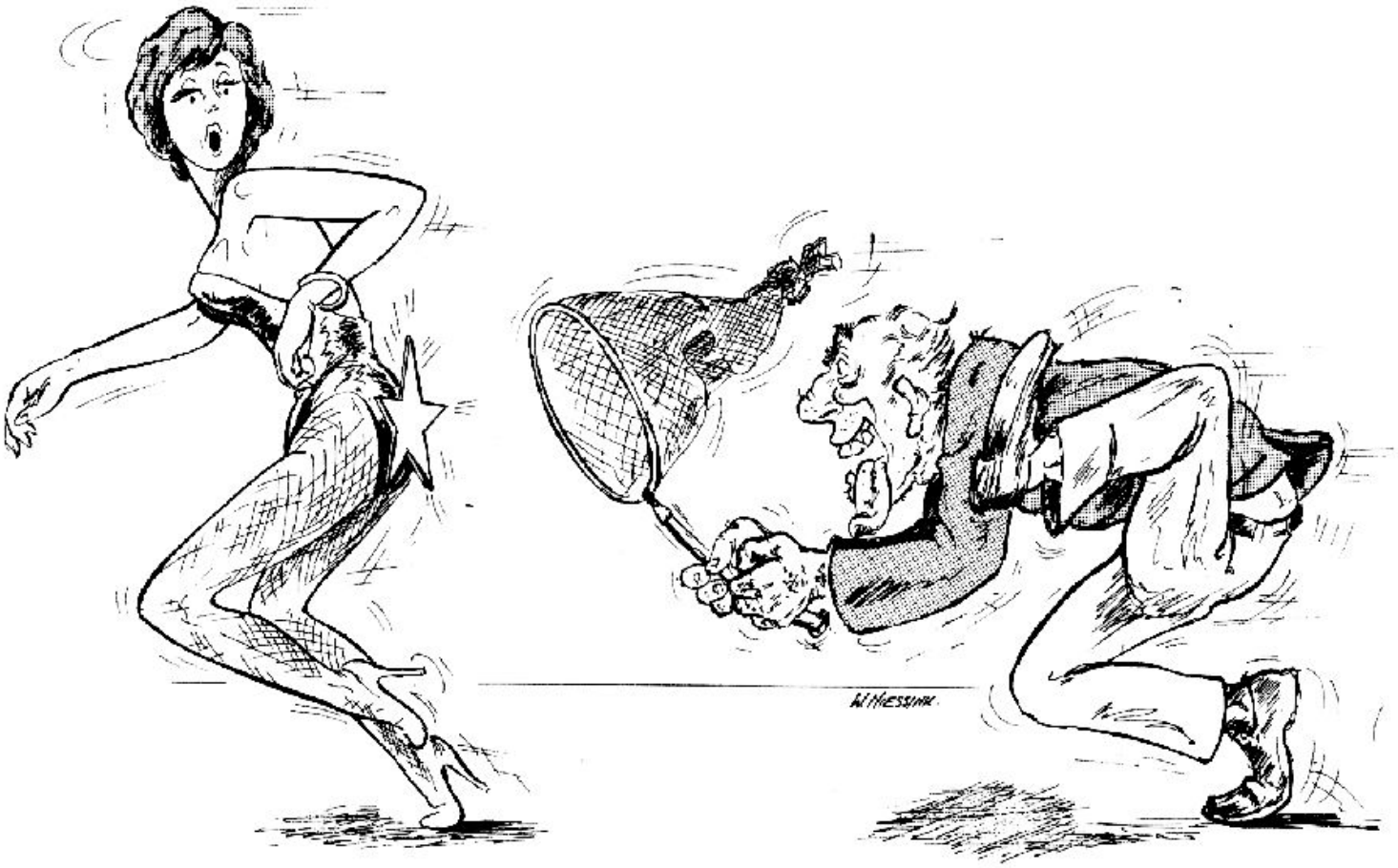
```
5 RAND
10 LET G=51
15 LET B=1
20 LET V=100
25 LET C=INT(RND*100+1)
26 LET L=1
29 CLS
30 PRINT "GETAL ";
35 INPUT D
40 PRINT D
45 IF D=C THEN GOTO 100
50 PRINT "HOGER" AND D<C;"LAGER" AND D>C
55 PRINT "JOUW GETAL ";G
60 PRINT "G=1 H=2 L=3"
65 INPUT K
70 IF K=1 THEN GOTO 95
75 IF K=2 THEN LET B=G
80 IF K=3 THEN LET V=G
85 LET G=INT((B+V)/2)
86 LET L=L+1
90 GOTO (30 AND L/5-INT(L/5)<>0)
   +(29 AND L/5-INT(L/5)=0)
95 PRINT "GETAL ZX81 ";C
100 PRINT "JA"
```

3.10 STERREN VANGEN

In dit spel komen de sterren uit de hemel vallen. Het is de bedoeling dat de speler probeert zoveel mogelijk sterren in zijn mandje te vangen.
Het mandje kan worden bestuurd door de Z-toets en de M-toets. Wordt de M-toets ingedrukt dan gaat het mandje naar rechts, drukt men de Z-toets in dan gaat het mandje naar links. In totaal komen er twintig sterren naar beneden. Boven in het beeldscherm wordt de score bijgehouden en het aantal sterren dat naar beneden is gevallen.

Werking van het programma

Op regel 3 en 5 worden de variabelen van een startwaarde voorzien. Nu eens niet met een numerieke waarde, maar met een functie, die de string omzet in een numerieke waarde.
Ogenschijnlijk neemt deze manier meer ruimte in, maar niets is minder waar. Daarna begint het feitelijke programma. Het beeldscherm wordt opgebouwd en de kolom wordt bepaald waarin de ster naar beneden gaat komen. De sterren komen met een vastgestelde snelheid naar beneden.

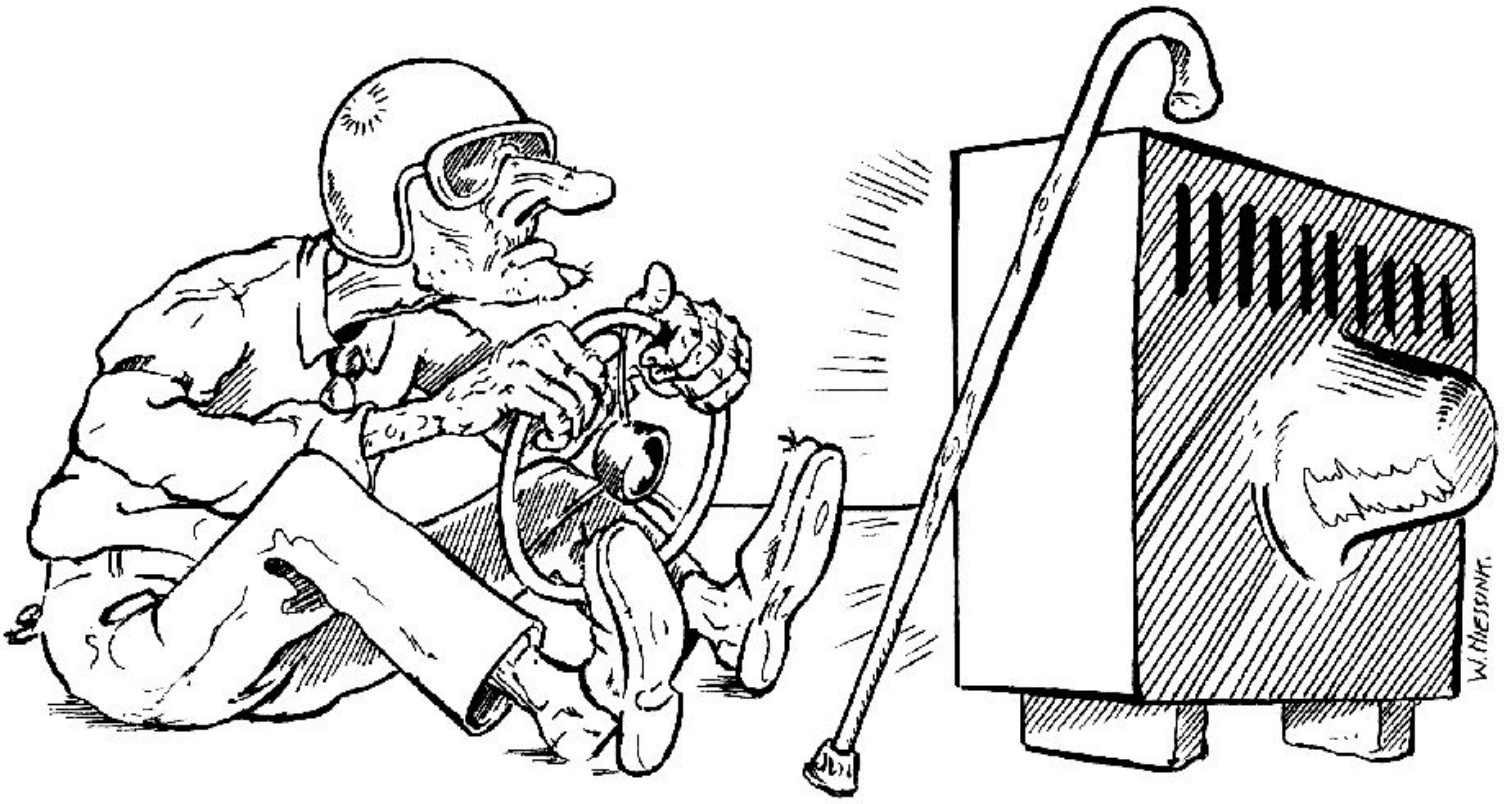


Dit wordt uitgevoerd door de tweede lus, waarvan de variabele I de teller is. Tevens krijgt de speler de kans om tijdens het vallen van de ster zijn mandje op de goede plaats neer te zetten. Wanneer de lus verlaten wordt, kan worden nagegaan door het programma of de ster in het mandje is gevallen.
Het mandje bestaat uit drie karakters, maar de ster moet precies op de plaats komen van het middelste karakter. Dit is de plaats waar het mandje wordt afgedrukt plus één (Y+1). Zit de ster in het mandje dan wordt de score met één verhoogd.
Dit programma is uitermate geschikt om eens mee te experimenteren. Men kan bijvoorbeeld de valsnelheid van de ster beïnvloeden door een willekeurige STEP in te bouwen. Het aantal sterren is beperkt tot 20. Het is vrij gemakkelijk om ook dit variabel te maken. Men hoeft alleen maar de lus op regel 7 te wijzigen.

Programma

```
1 RAND
3 LET Y=VAL "10"
5 LET B=VAL "0"
6 CLS
7 FOR Z=1 TO 20
8 PRINT AT 0,0;"RONDE ";Z;AT 0,10;"SCORE ";B
10 LET X=INT(RND*VAL "20")
15 FOR I=3 TO 12
18 PRINT AT 12,Y;"□□□"
19 LET Y=Y-(INKEY$="Z" AND Y>=0)+
   (INKEY$="M" AND Y<20)
20 PRINT AT 12,Y;" ▙▬▟ "
25 PRINT AT I-1,X;"□";AT I,X;"*"
30 NEXT I
35 IF X<>Y+1 THEN GOTO 45
40 LET B=B+1
45 PRINT AT 12,X;"□"
50 NEXT Z
```

3.11 SPOOKRIJDER



In dit spel op leven en dood is de speler op een drukke autoweg terecht gekomen en rijdt tegen de rijrichting in. Hij kan niet weg, omdat aan de linkerkant een berg en aan de rechterkant een vangrail is. De weg is overvol met auto's. De speler zal moeten proberen zo lang mogelijk

op de weg te blijven zonder ook maar een auto, berg of vangrail aan te raken. Doet hij dit wel dan is de auto waarin hij rijdt total loss. Het spel is na de crash afgelopen, het enige wat de computer nog meldt is het aantal seconden dat de speler zich als spookrijder heeft kunnen handhaven.
De auto van de speler wordt voorgesteld door een 'A'. Om de auto te kunnen besturen kan hij gebruik maken van de Z-toets en de M-toets. Met de Z-toets gaat de auto naar links en met de M-toets gaat de auto naar rechts. Het is overigens te hopen dat de opgedane ervaring in dit spel nooit aan de praktijk getoetst zal hoeven worden.

Werking van het programma

Het zal langzaam maar zeker opvallen, dat de variaties in spellen oneindig zijn, terwijl toch iedere keer dezelfde standaardelementen terugkomen. Dit laatste geldt ook voor dit programma.
De klok wordt weer op de bekende startwaarde gezet (65535) en een aantal variabelen krijgen een beginwaarde. De variabele Y krijgt de waarde 3. Dit is de plaats waar de auto van de speler op de weg staat. Deze variabele Y wordt beïnvloed door het INKEY$-statement.
De variabele S wordt voor een aantal zaken gebruikt. Deze variabele zorgt er onder andere voor dat tussen elke auto, die op de autoweg wordt gezet zes blanco regels zitten. Dit wordt gerealiseerd met het SCROLL-statement. Tevens wordt de variabele S gebruikt als stuurvariabele in de RND-functie bij het bepalen van de variabele X. De variabele X geeft de kolom aan waar de auto's op de weg worden gezet. Op regel 45 wordt nagegaan of het af te drukken symbool niet toevallig een zwart blokje is en tegelijkertijd wordt gekeken of de auto van de speler niet tegen de berg aan zit. Blijkt aan één van de voorwaarden te zijn voldaan, dan verschijnt er ** CRASH **. De auto van de speler zat óf tegen een andere auto óf tegen de vangrail óf hij was tegen de berg aangeknald. Ten slotte verschijnt de tijd op het beeldscherm, die aangeeft hoe lang de speler zijn dolle rit heeft kunnen volhouden.

Programma

```
1 POKE 16437,255
2 POKE 16436,255
15 RAND
16 LET Y=VAL "3"
20 LET S=VAL "6"
21 LET X=RND*S+S/S
30 PRINT AT 20,9;"■";AT 18,X;"■■";AT 19,X;"■"
   AT 20,X;"■■"
41 SCROLL
43 LET Y=Y-(INKEY$="Z" AND Y>=0)
   +(INKEY$="M" AND Y<10)
44 PRINT AT 20,9;"■";AT 0,Y;
45 IF PEEK (PEEK 16398+PEEK 16399*256)>=CODE "■"
   OR Y=0 THEN GOTO 80
```

```
46 PRINT "[A]"
50 LET S=S-1
60 IF S<0 THEN GOTO 20
70 GOTO 41
80 PRINT AT 0,Y;"** CRASH **"
85 PRINT INT((65535-PEEK 16437*256+PEEK 16436)/50)
```

3.12 PING

Ping komt men op alle spelcomputers tegen. Het heet dan meestal squash. Dat dit spel 'Ping' heet is een eerbetoon aan de uitvinder van het eerste TV-spel, die dit 'Pong' noemde.
Dit ogenschijnlijk eenvoudige spel is aan dit boek toegevoegd om te laten zien dat het programma achter het spel nu niet bepaald eenvoudig te noemen valt. Zeker niet als men bedenkt dat het ook nog in een ZX81 1 K-versie moet passen.
De bedoeling is een bal, die wordt voorgesteld door een ster, zo lang mogelijk in het spel te houden. De bal kan tegen drie muren en een racket stuiteren. Wordt de bal gemist door het racket dan verschijnt er weer een nieuwe bal. In totaal levert het spel vijf ballen.
Ook nu weer kan men met de M- en de N-toets het racket bewegen. Het programma houdt bij hoe lang de bal in het spel is. Deze score wordt aan het eind van het spel afgedrukt.

Werking van het programma

Een van de zaken, die bij de wat grotere programma's opvallen is dat zij voornamelijk gebruik maken van het linker bovengedeelte van het scherm. Dit wordt gedaan om zo min mogelijk geheugenruimte in beslag te nemen.
Het spel is vrij simpel, maar de uitwerking vraagt toch nog wel het een en ander. Overigens, in een 16 K-uitvoering is dit spel veel aantrekkelijker op te zetten, omdat men dan gerust het hele scherm kan gebruiken. Tevens kan men dit spel gebruiken als basis voor een spel waarbij een muur moet worden afgebroken.
Tot en met regel 10 worden een aantal variabelen op een startwaarde gezet. Variabele B geeft het aantal ballen aan waarmee nog kan worden gespeeld. Variabele S geeft aan hoe lang de bal in het spel is. De variabele Y geeft de plaats aan waar het racket staat gepositioneerd. De variabelen P en T geven de locatie aan waar de bal zich bevindt. P geeft de regel aan en T geeft de kolom aan. De variabelen R en L zijn de stuurvariabelen voor de regel en de kolom, zij hebben beide de waarde 1 of -1. Wanneer de bal een verticale muur raakt, wordt het teken van L omgedraaid. Wordt daarentegen het racket of de top van het beeldscherm geraakt, dan wordt het teken van R omgekeerd, tegelijkertijd verandert ook willekeurig het teken van L.
In regel 16 wordt de variabele S met één verhoogd. Dit wordt gedaan met de absolute waarde van R. Dit is gedaan om geen extra waarde te hoeven gebruiken, hetgeen weer ruimtebesparend werkt.
In regel 25 wordt gekeken of de bal zich op regel 9 bevindt. Is dit laatste het geval dan wordt tegelijkertijd gekeken of de bal ook op het racket

gekomen is. Zo niet dan wordt het aantal ballen met één verminderd.

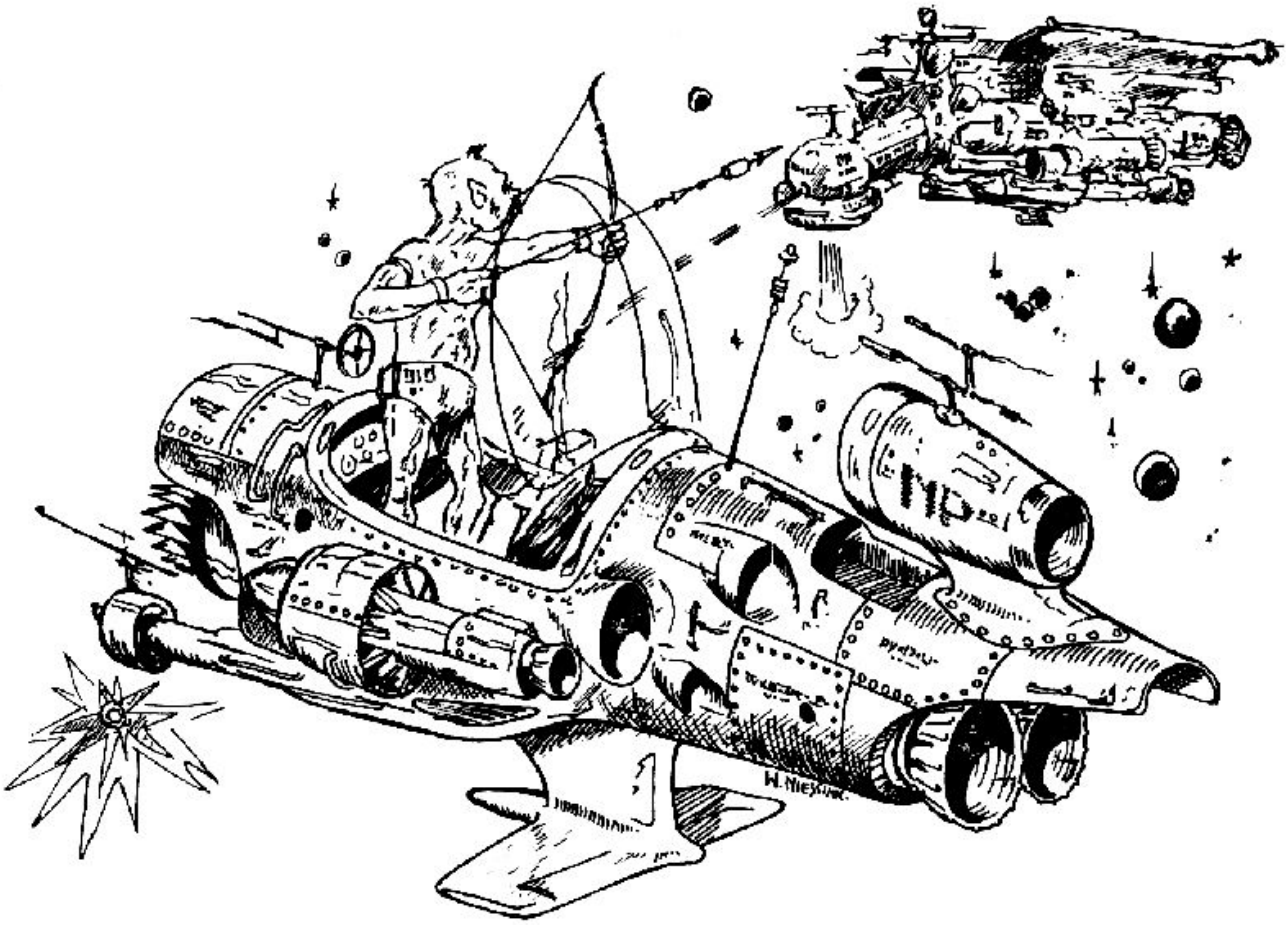
Programma

```
1 LET B=1
2 LET S=B
3 PAUSE 3E4
4 CLS
6 LET Y=6
7 LET P=1
8 LET T=INT(RND*Y+P+P)
9 LET R=P
10 LET L=P
11 FOR I=P TO 10
12 PRINT "□";TAB 13;"□"
13 NEXT I
14 PRINT AT 10,Y;"□□"
15 LET Y=Y+(INKEY$="M" AND Y<13)
   -(INKEY$="N" AND Y>1)
16 LET S=S+ABS(R)
20 PRINT AT 10,Y;" ▄▄ "
25 IF P=9 AND T<>Y AND T<>Y+1 THEN GOTO 85
30 PRINT AT P,T;"□"
32 IF T<=1 OR T>=12 THEN LET L=-L
35 LET P=P+R
40 LET T=T+L
50 IF P>0 AND P<9 THEN GOTO 70
55 LET R=-R
60 LET L=SGN(RND*4.5-2)
70 PRINT AT P,T;"*"
75 GOTO 14
85 LET B=B+1
86 IF B<5 THEN GOTO 3
90 PRINT S
```

3.13 RUIMTESLAG

Aan het einde van de melkweg woedt een slag tussen de ruimtemarine van de Srixis en de Galactische marine. De speler is vuurleider van een van de schepen van de Galactische marine en hij moet proberen zoveel mogelijk ruimteschepen van de Srixis uit te schakelen. Daarbij moet hij wel uitkijken dat hij niet een van de eigen schepen raakt.
De speler wordt beloond met punten voor het uitschakelen van vijandelijke schepen en hij krijgt strafpunten, wanneer hij een van de Galactische schepen treft.
De vijandelijke schepen zijn te herkennen aan het zwarte vlakje met daarin een ster of een H. De schepen van de Galactische marine worden voorgesteld door <> en door 0.
Op het beeldscherm verschijnt een eenvoudig vizier. Bevindt een ruimteschip zich in het vizier dan kan door het indrukken van de I-toets op

het ruimteschip worden geschoten.Met de A-, Z-, N- en M-toets kan het vizier gericht worden. De A- en Z-toetsen verplaatsen het vizier in verticale richting en de N- en M-toetsen verplaatsen het vizier in horizontale richting. Heeft men een ruimteschip te pakken dan wordt dat gemeld met een kortstondig 'RAAK'.
De puntenverdeling is als volgt: Schiet men een [*] neer dan ontvangt men tien punten, schiet men een [H] neer dan krijgt men twintig punten. Schiet men daarentegen een <> neer dan levert dat dertig strafpunten op en 0 zelfs veertig strafpunten. Dus het is uitkijken geblazen. Temeer daar de ruimteschepen niet stil hangen, maar ook bewegen.



Werking van het programma

Zoals de meeste programma's in dit boek, is ook dit spel volgens hetzelfde concept opgebouwd om in de 1K-versie van de ZX81 te passen.
Het spel wordt gespeeld over vijf ronden, dit wordt gestuurd door de FOR...NEXT-lus met P als stuurvariabele. De variabelen X en Y sturen de horizontale en verticale beweging van het vizier. De variabele A regelt welk 'vijandelijk ruimteschip' in beeld komt, deze variabele kan de waarde 1 t/m 4 aannemen. De variabelen Q en T bepalen waar het ruimteschip op het beeldscherm komt te staan. De bewegingen van de ruimteschepen zijn volledig willekeurig, maar in de beginsituatie is het 'vijandelijke ruimteschip' meestal binnen schootsafstand.
Het afdrukken van het ruimteschip geschiedt in regel 35. Deze methode van afdrukken staat in hoofdstuk 1 uitgelegd, zodat dit hier achterwege

blijft. Door het kiezen van deze oplossing bespaart men aanzienlijk veel ruimte in het programma.
Nadat op regel 60 de horizontale en verticale plaats van het vizier zijn aangepast, wordt op regel 60 nagegaan of er geschoten is. Afhankelijk of er geschoten is, wordt de tweede lus verlaten. Daarna wordt bepaald of er raak geschoten is. Zo ja, dan wordt de score bijgewerkt, afhankelijk van het soort ruimteschip dat is getroffen.

Programma

```
4 RAND
5 LET S=0
6 FOR P=1 TO 5
10 LET X=10
15 LET Y=10
16 LET A=INT(RND*4+1)
17 LET Q=X+RND*2-1
18 LET T=Y+RND*2-1
30 FOR J=1 TO 20
31 CLS
33 LET T=T+RND
34 LET Q=Q+RND*2-1
35 PRINT AT Q,T;"[*]" AND A=1;"[H]" AND A=2;"<>" AND
   A=3;"O" AND A=4
40 PRINT AT Y,X-3;"I-+-I"
45 LET X=X-(INKEY$="N" AND X>3)
   +(INKEY$="M" AND X<29)
50 LET Y=Y-(INKEY$="A" AND Y>0)
   +(INKEY$="Z" AND Y<20)
55 LET I=CODE INKEY$
60 IF I<>46 THEN NEXT J
70 IF I<>46 OR ABS(T-Y)>1 OR ABS(Q-X)>2 THEN GOTO 85
71 PRINT AT Y-3,X;"RAAK"
75 IF A>3 THEN LET S=S-A*10*2
80 LET S=S+A*10
85 NEXT P
90 PRINT AT 0,0;"SCORE ";S
```

3.14 JACHT

Er is een wild zwijn opgedreven en de speler moet zien dit zwijn te pakken te krijgen. Het zwijn rent van links naar rechts over het beeldscherm. De speler begint rechts. Haalt het zwijn de overkant dan heeft de speler verloren. Weet de speler het zwijn de pas af te snijden dan heeft de speler gewonnen.
Het spel stopt zodra het zwijn de overkant heeft gehaald of als de speler het zwijn te pakken heeft. Het is begrijpelijk dat het zwijn in paniek is. Er is dus niet te voorspellen welke richting het op zal rennen.
Het zwijn wordt voorgesteld door een flikkerend zwart blokje, de speler wordt voorgesteld door een sterretje. Het sterretje kan in de verticale

richting worden bestuurd met de A- en de Z-toets. In de horizontale richting zijn dit de N- en de M-toets.

Werking van het programma

De variabelen R en T bepalen de plaats op het beeldscherm waar het sterretje zich bevindt. Met behulp van het INKEY$-statement kunnen R en T worden gevarieerd. Voor R liggen de grenzen echter tussen 0 en 21 en voor T liggen deze tussen 0 en 31.
De variabelen X en Y bepalen de plaats waar het zwijn zich bevindt. Blijkt nu dat de variabele X overeenstemt met R en Y met T dan stopt het spel en is de speler winnaar.

De bewegingen van het zwijn zijn puur willekeurig, dit wordt bereikt door te manipuleren met de RND-functie. Heeft het zwijn de overkant bereikt en dat is wanneer de variabele Y groter is dan 31, dan heeft de speler verloren. De Y-waarde bepaalt feitelijk wie er heeft gewonnen of verloren. Is de Y-waarde groter dan 31, dan heeft de speler verloren. Is de Y-waarde kleiner dan 31 dan heeft de speler gewonnen.

Programma

```
5 RAND
6 LET R=10
7 LET T=31
10 LET X=10
15 LET Y=0
16 PRINT AT R,T;"□"
17 LET R=R-(INKEY$="A" AND R>0)+
   (INKEY$="Z" AND R<21)
18 LET T=T-(INKEY$="N" AND T>0)+
   (INKEY$="M" AND T<31)
```

```
19 IF INT(X+.5)=R AND INT(Y+.5)=T THEN GOTO 45
20 PRINT AT X,Y;"■";AT R,T;"*"
27 PRINT AT X,Y;"□"
30 LET X=X+RND*2-1
35 LET Y=Y+RND
40 IF INT (Y+.5)<32 THEN GOTO 16
45 PRINT AT X,Y-1;"⊡";AT 20,0;"WINST" AND Y<31;
   "VERLIES" AND Y>31
```

3.15 STAD

Het programma 'Stad' levert een mooi plaatje. Het is als het ware een zichzelf vormend beeldscherm. Als het goed is, is het plaatje iedere keer anders.
De werking van het programma is recht toe recht aan. Er wordt een karakter gekozen en in variabele P geplaatst. Daarna wordt een nieuwe X- en Y-richting bepaald en ten slotte wordt het karakter op de betreffende regel en kolom afgedrukt.

Programma

```
10 RAND
20 LET X=20
```

```
30 LET Y=20
40 LET P=INT(RND*7+129)
50 LET X=X+SGN(RND*2-1)
60 IF X<0 THEN LET X=0
65 IF X>30 THEN LET X=30
70 LET Y=Y+SGN(RND*2-1)
75 IF Y>20 THEN LET Y=20
80 IF Y<0 THEN LET Y=0
90 PRINT AT X,Y;CHR$ P
100 GOTO 40
```

4. Educatieve programma's

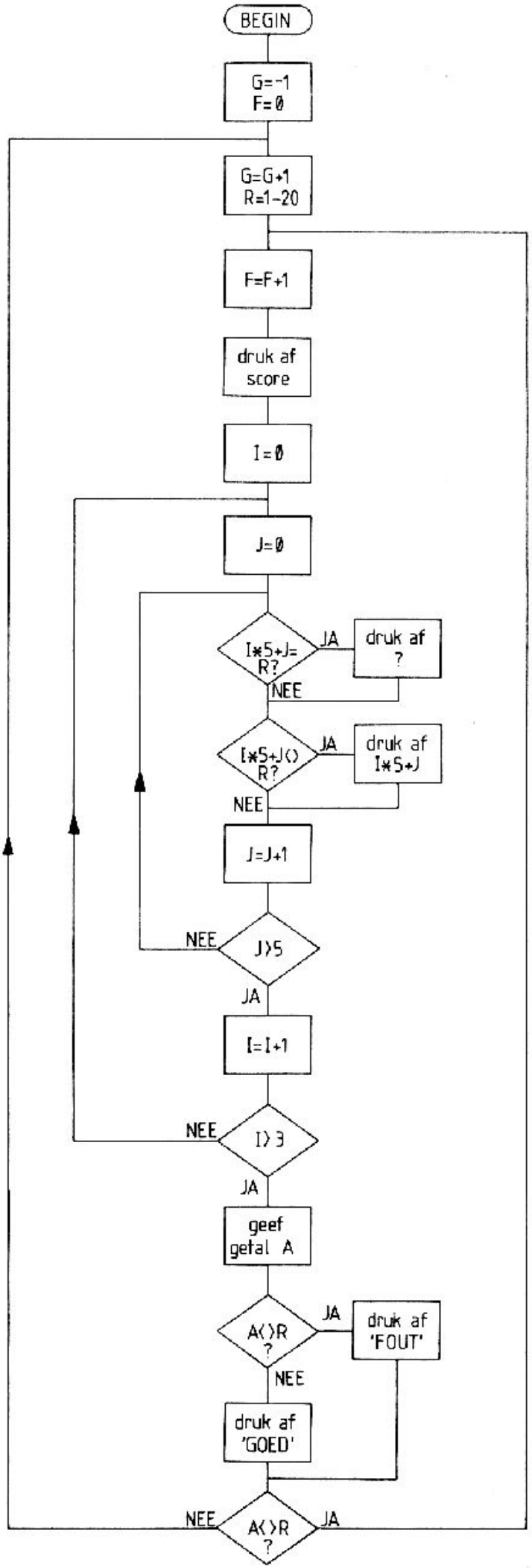
De educatieve programma's zijn gemaakt voor kinderen tussen 6 en 11 jaar. In ieder geval moeten zij na enige uitleg begrijpen wat de bedoeling is van het programma. De volgorde van de programma's is willekeurig. Ze zijn zeker niet naar moeilijkheidsgraad gerangschikt. Men moet wel bedenken dat in een 1K-versie van de ZX81 geen geavanceerde educatieve programma's zijn te ontwikkelen. De programma's zijn bedoeld om aan te geven dat dit soort programmatuur ook op de ZX81 mogelijk is.
De ene helft van de programma's bestaat uit taalkundige problemen, de andere helft uit rekenkundige problemen. De rekenkundige programma's zijn gericht op hoofdrekenen. De taalkundige programma's richten zich met name op het herkennen van letters.
In geen van de educatieve programma's wordt een klok bijgehouden. De score daarentegen wordt wel bijgehouden! In de meeste programma's verschijnt op het beeldscherm, hoeveel vragen er gesteld zijn en hoeveel antwoorden goed waren.

4.1 WELK GETAL ONTBREEKT ?

Op het beeldscherm verschijnen vier rijen met in totaal twintig getallen. Deze getallen lopen van 1 tot en met 20. Eén van de twintig getallen ontbreekt. Hiervoor in de plaats wordt een zwart blokje met een vraagteken afgedrukt. Nadat het scherm is opgebouwd, zal het programma vragen 'WELK GETAL IS WEG?'. Na invoer van een getal door de gebruiker wordt nagegaan of het ingevoerde getal overeenkomt met het weggelaten getal. Het resultaat wordt de gebruiker verteld. Hierna staat het spel stil, wanneer men dan een willekeurige toets aanraakt wordt een nieuwe reeks op het beeldscherm gezet.
Het programma houdt bij hoeveel vragen er zijn gesteld en geeft tevens aan hoeveel vragen goed waren beantwoord. Was een vraag fout beantwoord, dan wordt dezelfde reeks nogmaals op het beeldscherm gezet.

Werking van het programma

Wanneer men het stroomschema (pagina 50) bestudeert met de listing er naast, zal waarschijnlijk het grootste deel van het programma duidelijk zijn. Misschien zal menigeen zich afvragen wat het nut is van de twee ge-



Stroomschema van 'Welk getal ontbreekt?'.

neste lussen in regel 40 en 42. Echter voordat hierop wordt ingegaan, zal eerst iets over variabele R uit de doeken worden gedaan. De waarde van variabele R is het getal dat in de af te drukken twintig getallen wordt weggelaten. De twee geneste lussen tussen regel 40 en 60 zorgen er voor dat twintig keer een getal kan worden afgedrukt! Slechts in één van die twintig keer wordt een vraagteken afgedrukt! Dit gebeurt als de inhoud van variabele R gelijk is aan het getal dat op dat moment zou worden afgedrukt. Regel 75 en 80 zijn typisch kunstgrepen, die in hoofdstuk 2 zijn uitgelegd en die alleen op de ZX81 mogelijk zijn.

Programma

```
5 RAND
10 LET G=-1
15 LET F=0
20 LET G=G+1
25 LET R=INT(RND*20+1)
30 LET F=F+1
34 CLS
35 PRINT AT 0,2;"VRAGEN ";F;AT 0,19;"GOED ";G
40 FOR I=0 TO 3
42 FOR J=1 TO 5
45 IF I*5+J=R THEN PRINT AT 4+I,J*3;"?";
50 IF I*5+J<>R THEN PRINT AT 4+I,J*3;I*5+J;
55 NEXT J
56 PRINT
60 NEXT I
65 PRINT AT 13,3;"WELK GETAL IS WEG ";
70 INPUT A
75 PRINT A;" NIET GOED" AND A<>R;" GOED" AND A=R
76 IF INKEY$="" THEN GOTO 76
80 GOTO (20 AND A=R)+(30 AND A<>R)
```

4.2 WELKE LETTER IS ANDERS?

Dit programma is speciaal voor de kleintjes gemaakt van vier à vijf jaar. Op het beeldscherm worden negen letters, drie aan drie afgedrukt. Acht letters zijn hetzelfde en één letter is anders. Het is de bedoeling die ene afwijkende letter te herkennen en aan het programma op te geven. Het antwoord wordt door het programma bekeken. Het resultaat is GOED of FOUT. Was het antwoord toevallig fout dan worden dezelfde letters op het beeldscherm gezet. De afwijkende letter wordt steeds op een andere plaats afgedrukt, zodat er geen systeem in zit. Gokken is dus uitgesloten!
Boven in het beeldscherm wordt bijgehouden hoeveel vragen er zijn gesteld en hoeveel antwoorden goed waren. Na de reactie van het programma GOED of FOUT stopt het programma enige tijd om daarna met een nieuwe serie letters te komen.

Werking van het programma

Hoewel het programma voor de kleintjes is bedoeld, zal het toch door een ouder moeten worden uitgelegd. Bij de ontwikkeling van het programma is er wel van uitgegaan dat het door kleine kinderen gespeeld zou worden.

Zo is bijvoorbeeld niet van INPUT-statements gebruik gemaakt, maar van INKEY$.
Het programma maakt veel gebruik van de randomgenerator. De variabele A bevat de decimale waarde van de letter die acht keer wordt afgedrukt. De variabele B bevat de decimale waarde van de te herkennen letter. Zoals in de beschrijving is opgemerkt, is de plaats waar de te herkennen letter wordt afgedrukt volkomen willekeurig. De variabele T geeft aan op welke plaats in de 3*3 matrix, de te herkennen letter wordt afgedrukt.
Wanneer men de andere programma's in dit hoofdstuk bekijkt, is men best in staat dit programma zodanig te wijzigen dat het ook met cijfers kan werken. Hiertoe moeten onder andere de uitdrukkingen op regel 25 en 30 worden aangepast.

Programma

```
5 RAND
10 LET G=-1
15 LET Q=0
16 LET G=G+1
20 LET Q=Q+1
25 LET A=INT(RND*26+38)
30 LET B=INT(RND*26+38)
35 IF A=B THEN GOTO 30
40 CLS
41 PRINT AT 0,8;"VRAGEN ";Q;AT 0,19;"GOED ";G
45 LET T=INT(RND*9+1)
50 FOR I=0 TO 2
55 FOR J=1 TO 3
60 PRINT AT I+10,J*2+10;CHR$ A AND T<>I*3+J;CHR$ B
   AND T=I*3+J
70 NEXT J
75 PRINT
80 NEXT I
85 PRINT AT 17,0;"WELKE LETTER IS ANDERS ?";
90 LET Z=CODE INKEY$
91 IF Z=0 THEN GOTO 90
95 PRINT CHR$ Z;" FOUT " AND Z<>B;" GOED " AND Z=B
96 PAUSE 25
97 POKE 16437,255
100 GOTO (16 AND Z=B)+(20 AND Z<>B)
```

4.3 WELKE LETTER ONTBREEKT?

Om dit programma te kunnen spelen, dient men wel het alfabet een beetje te kennen. Het programma zet het hele alfabet, behalve de Z, op het beeldscherm.
Eén letter uit het alfabet wordt weggelaten, hiervoor in de plaats wordt een zwart blokje met vraagteken afgedrukt. Het is de bedoeling dat de speler deze ontbrekende letter intoetst. Het programma bepaalt dan of de ingevoerde letter inderdaad gelijk was aan de weggelaten letter. Was het

antwoord fout dan wordt dezelfde vraag nogmaals gesteld. Boven in het beeldscherm wordt de score en het aantal goede antwoorden bijgehouden. Na elke beantwoording stopt het programma een aantal seconden.

Werking van het programma

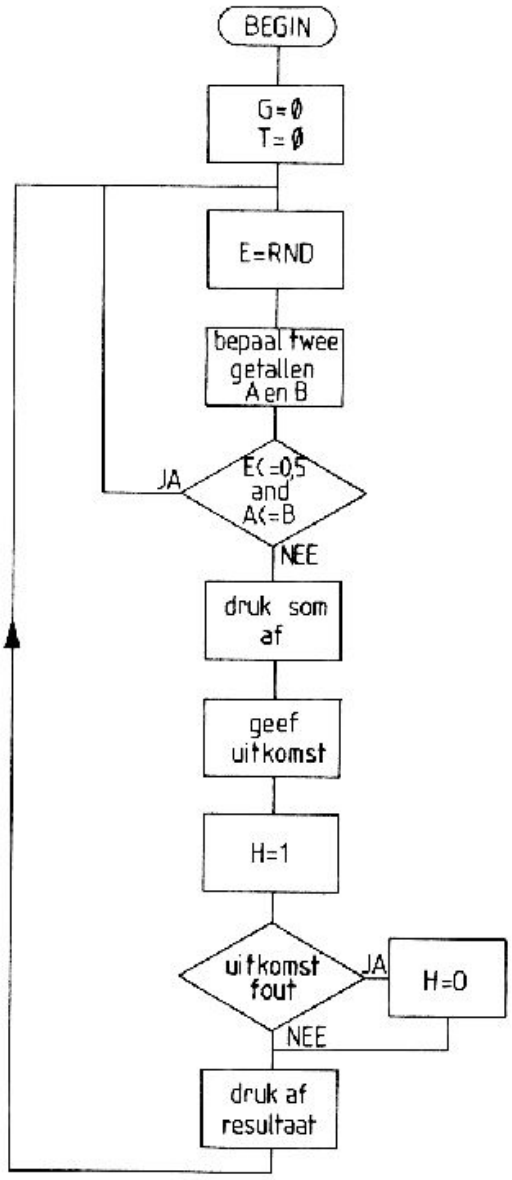
Wanneer men dit programma legt naast het programma 'Welk cijfer ontbreekt?', zal het menigeen opvallen, dat beide programma's sterk op elkaar lijken. Dat is ook niet zo verwonderlijk. In dit programma wordt met de numerieke waarde van de letters gewerkt. Volgens de reeds bekende appendix A (Engelse handleiding) beginnen de letters bij 38 en lopen door tot 63. De variabele R krijgt een willekeurig getal tussen 1 en 25. Wanneer nu 37 bij de variabele R wordt opgeteld, wordt als het ware een van de decimale waarden van de letters van het alfabet gekozen.
Bij het programma 'Welk getal ontbreekt?' werd gebruik gemaakt van het INPUT-statement om het getal in het programma op te nemen. Het gebruik van INKEY$ was geen haalbare kaart, omdat twee karakters moesten worden ingetoetst. Bij dit programma wordt maar één karakter ingevoerd en dus stond niets meer in de weg om van het INKEY$-statement gebruik te maken.
Het programma is ook een goede demonstratie van de werking van de functies CHR$ en CODE.

Programma

```
5 RAND
10 LET G=-1
15 LET F=0
20 LET G=G+1
25 LET R=INT(RND*25)+38
30 LET F=F+1
34 CLS
35 PRINT AT 0,2;"VRAGEN ";F;AT 0,19;"GOED ";G
40 FOR I=0 TO 4
42 FOR J=1 TO 5
43 LET L=I*5+J+37
45 IF L=R THEN PRINT AT I+4,J*3;"[?]";
50 IF L<>R THEN PRINT AT I+4,J*3;CHR$ L;
55 NEXT J
56 PRINT
60 NEXT I
65 PRINT AT 13,3;"WELKE LETTER IS WEG ";
70 LET A=CODE INKEY$
71 IF A=0 THEN GOTO 70
76 PRINT CHR$ A;" GOED";AND A=R;" FOUT"; AND A<>R
77 FOR I=1 TO 25
78 NEXT I
79 IF A>62 THEN STOP
80 GOTO (20 AND A=R)+(30 AND A<>R)
```

4.4 OPTELLEN EN AFTREKKEN

Het programma optellen en aftrekken geeft eenvoudige sommetjes op. Het kunnen optelsommetjes zijn of aftreksommetjes. Het programma vraagt om een antwoord. Is het antwoord goed, dan meldt het programma dit met GOED. Was het fout dan wordt FOUT afgedrukt en wordt de vraag opnieuw gesteld. Na de melding van de computer GOED of FOUT stopt het programma een paar seconden. Hierna verschijnt pas de nieuwe opgave.
Boven in het beeldscherm wordt bijgehouden hoeveel sommen zijn opgegeven en hoeveel van de sommen goed zijn beantwoord.
Men kan het programma onderbreken door een negatief getal op te geven als uitkomst van een som. Het programma bepaalt puur willekeurig of het een optelsom of een aftreksom geeft.



Stroomschema van 'Optellen en aftrekken'.

Werking van het programma

De variabele T houdt het aantal opgegeven sommen bij, terwijl de variabele G het aantal goed gegeven antwoorden bijhoudt. De variabele E bepaalt in

principe of de som, die door het programma gemaakt zal gaan worden een optel- of aftreksom zal zijn. Wanneer E kleiner is dan 0,5 wordt de som een aftrekking. In het andere geval wordt de som een optelling. Dit betekent dat er een iets hogere kans is op optelsommen. Blijkt dat E kleiner is dan 0,5 dan moet tevens gelden dat A groter moet zijn dan B. De variabelen A en B zijn de twee getallen die bij elkaar worden opgeteld of van elkaar afgetrokken. Indien de twee getallen van elkaar worden afgetrokken, dient A groter te zijn dan B, omdat anders de mogelijkheid gaat ontstaan dat de uitkomst negatief wordt.
In het programma wordt van een zogenaamde switch gebruik gemaakt. Dit is een bekende manier van werken in de automatisering. In regel 60 wordt de variabele H keihard op één gezet. Dit betekent dat zonder meer wordt aangenomen dat het gegeven antwoord goed is! Daarna wordt het gegeven antwoord gecontroleerd. Blijkt dan dat het antwoord fout is dan wordt de variabele H op nul gezet. Met behulp van de H-waarde wordt de uitvoer gestuurd. Is H gelijk aan 1 dan wordt GOED afgedrukt, is H gelijk aan 0 dan wordt FOUT afgedrukt. Tevens wordt de waarde van variabele H bij de waarde van variabele G opgeteld. Was het antwoord fout, dan heeft H de waarde 0, dus aan de waarde G verandert dan feitelijk niets. De gevolgde methode heeft het voordeel dat het programma overzichtelijker wordt en toch niet meer ruimte inneemt. Het laatste is juist bij een 1 K-versie van de ZX81 van belang.

Programma

```
5 RAND
6 LET T=0
7 LET G=0
10 LET E=RND
15 LET A=INT(RND*10+1)
20 LET B=INT(RND*10+1)
25 IF E<=.5 AND A<=B THEN GOTO 10
26 CLS
27 PRINT AT 0,0;"SOMMEN ";T;AT 0,20;"GOED ";G
30 IF E<=.5 THEN PRINT AT 4,3;A;"□-□";B;"□=□";
40 IF E>.5 THEN PRINT AT 4,3;A;"□+□";B;"□=□";
50 INPUT C
55 IF C<0 THEN STOP
60 LET H=1
70 IF E>.5 AND C<>A+B THEN LET H=0
71 IF E<=.5 AND C<>A-B THEN LET H=0
75 PRINT C;AT 6,3;"FOUT" AND H=0;"GOED" AND H=1
80 LET G=G+H
85 LET T=T+1
90 FOR I=1 TO 25
95 NEXT I
97 GOTO 10
```

4.5 VERMENIGVULDIGEN EN DELEN

De programma's 'Optellen en aftrekken' en 'Vermenigvuldigen en delen' zijn eigenlijk programma's, die bij elkaar horen. Het zou het leukst zijn ge-

weest, wanneer deze twee programma's in één programma zouden kunnen worden gezet. Wanneer men een 16K-versie heeft, is dit op eenvoudige wijze te realiseren.
Dit programma geeft vermenigvuldig- en deelsommen op. Bij vermenigvuldigingen blijft de uitkomst altijd onder de honderd. De delingen hebben een uitkomst onder de tien. De sommen die worden opgegeven zijn puur willekeurig. Het zijn delingen en vermenigvuldigingen door elkaar. Geeft men een fout antwoord dan wordt de vraag opnieuw gesteld. Het programma geeft aan of het antwoord goed of fout was. Hierna stopt het programma een aantal seconden, waarna de volgende vraag wordt gesteld. Geeft men als uitkomst een negatief getal op, bijvoorbeeld -1, dan stopt het programma.
Boven in het beeldscherm wordt aangegeven hoeveel sommen zijn opgegeven en hoeveel antwoorden goed waren.

Werking van het programma

De werking van dit programma wijkt niet af van het programma 'Optellen en aftrekken'. Zodat op de werking niet zo diep zal worden ingegaan. In regel 21 wordt een vermenigvuldiging uitgevoerd. De variabele C is nodig om hiermee de deelsommen te kunnen opgeven en controleren. Tevens bepaalt in dit programma de variabele E of de som een deling of vermenigvuldiging wordt.

Programma

```
 5 RAND
 6 LET T=0
 7 LET G=0
10 LET E=RND
15 LET A=INT(RND*10+1)
20 LET B=INT(RND*10+1)
21 LET C=A*B
26 CLS
27 PRINT AT 0,0;"SOMMEN ";T;AT 0,20;"GOED ";G
30 IF E<=.5 THEN PRINT AT 4,3;C;" : ";A;" = ";
40 IF E>.5 THEN PRINT AT 4,3;A;" * ";B;" = ";
50 INPUT D
55 IF D<0 THEN STOP
60 LET H=1
70 IF E>.5 AND D<>A*B THEN LET H=0
71 IF E<=.5 AND D<>C/A THEN LET H=0
75 PRINT D;AT 6,3;"FOUT" AND H=0;"GOED" AND H=1
80 LET G=G+H
85 LET T=T+1
90 FOR I=1 TO 25
95 NEXT I
97 GOTO 10
```

4.6 ALFABETISEREN

Bij dit programma kan men leren alfabetiseren. Op het beeldscherm worden een aantal willekeurige letters neergezet. Het is mogelijk dat één letter vaker voorkomt. De speler moet proberen deze letters in alfabetische volgorde te plaatsen door ze twee aan twee van plaats te verwisselen.
Boven de letters zijn cijfers aangegeven. Deze cijfers geven de posities van de letters weer. Wil men twee letters met elkaar van plaats laten verwisselen dan geeft men de posities van de letters op. Denkt men dat wisselen verder geen zin heeft, omdat de letters in volgorde staan, dan geeft men op beide vragen een nul als antwoord. Hierna zal het programma controleren of de letters inderdaad in alfabetische volgorde staan. Is het goed gedaan dan wordt er GOED afgedrukt, was het fout dan verschijnt er FOUT.

Werking van het programma

Er wordt een matrix A gedimensioneerd met zeven elementen. Elk element wordt gevuld met een willekeurige decimale waarde van een letter (38 t/m 63). Omdat er geen controle wordt uitgeoefend op welke letters al getrokken zijn, is het mogelijk dat een letter meerdere malen voorkomt. Wil men dit voorkomen, dan kan men de routine overnemen uit het programma 'Mastermind' die nagaat of een bepaald cijfer al getrokken is. Waarschijnlijk zal men geheugenproblemen krijgen, indien men deze routine inbouwt.
Na het opbouwen van de matrix met de decimale waarden van de letters, wordt de rij afgedrukt. Erboven verschijnt een rij met getallen, de zogenaamde letternummers. Deze nummers geven de posities van de letters aan. De speler wordt gevraagd, welke letters hij met elkaar wil verwisselen. Hij moet hiervoor twee letternummers opgeven. Vanwege ruimtegebrek wordt er geen controle op de invoer uitgevoerd. Wanneer men getallen opgeeft, die buiten de indices van de matrix vallen, stopt het programma met een foutmelding.
Voor het verwisselen van de inhoud van de elementen wordt gebruik gemaakt van een hulpvariabele A. Dit fysiek wisselen geschiedt tussen regel 85 t/m 91. Heeft de speler ten slotte aangegeven dat volgens hem de rij in alfabetische volgorde staat, dan hoeft dit alleen maar te worden gecontroleerd. Er moet dan gelden dat het element n uit de matrix nooit groter kan zijn dan het element n+1. Blijkt deze conditie niet op te gaan dan staat de rij niet in alfabetische volgorde.
Ook in dit programma wordt op een handige manier gebruik gemaakt van een verkapte stuurvariabele. Wanneer er wordt aangegeven dat men klaar is, krijgt de variabele X automatisch de waarde 0. Wanneer X=0, betekent dit dat de rij in alfabetische volgorde staat. Heeft X de waarde 1 gekregen na de controle, dan betekent dit dat er een fout in de rij zit.

Programma

```
5 RAND
6 LET H=7
```

```
10 DIM A(H)
15 FOR I=1 TO H
20 LET A(I)=INT(RND*26)+38
30 NEXT I
32 CLS
35 PRINT AT 5,0;"LETTERNR.";AT 7,0;"LETTER";
40 FOR I=1 TO H
45 PRINT AT 5,6+I*3; I ;AT 7,6+I*3;CHR$ A(I);
50 NEXT I
55 PRINT AT 9,4;"WISSEL VAN ";
60 INPUT X
70 PRINT X;AT 10,10;"NAAR ";
75 INPUT Y
80 PRINT Y
81 IF X=0 THEN GOTO 93
85 LET A=A(X)
90 LET A(X)=A(Y)
91 LET A(Y)=A
92 GOTO 32
93 FOR I=1 TO 6
94 IF A(I)>A(I+1) THEN LET X=1
96 NEXT I
99 PRINT AT 14,4;"GOED" AND X=0;"FOUT" AND X=1
```

4.7 ZET OP VOLGORDE VAN LAAG NAAR HOOG

Bij dit laatste educatieve programma wordt van de speler gevraagd een willekeurige rij van zeven getallen op volgorde te zetten van laag naar hoog. Het kan zijn dat een getal meerdere malen voorkomt.
De speler moet proberen door de getallen onderling met elkaar te verwisselen de rij op volgorde te krijgen. Om dit te bereiken moeten steeds twee getalnummers worden opgegeven die boven de getallen staan. Denkt de speler dat de rij op volgorde staat, dan geeft hij op beide vragen het antwoord nul. Hierna zal het programma bepalen of de rij inderdaad in volgorde staat. Is dit inderdaad het geval dan verschijnt er GOED en in het andere geval FOUT.

Bij dit programma wordt geen werking beschreven, omdat het programma erg veel lijkt op het voorgaande programma.

Programma

```
5 RAND
6 LET H=7
10 DIM A(H)
15 FOR I=1 TO H
20 LET A(I)=INT(RND*20+1)
30 NEXT I
32 CLS
35 PRINT AT 5,0;"GETALNR.";AT 7,0;"GETAL";
```

```
40 FOR I=1 TO H
45 PRINT AT 5,6+I*3;I;AT 7,6+I*3;A(I);
50 NEXT I
55 PRINT AT 9,4;"WISSEL VAN ";
60 INPUT X
70 PRINT X;AT 10,10;"NAAR ";
75 INPUT Y
80 PRINT Y
81 IF X=0 THEN GOTO 93
85 LET A=A(X)
90 LET A(X)=A(Y)
91 LET A(Y)=A
92 GOTO 32
93 FOR I=1 TO 6
94 IF A(I)>A(I+1) THEN LET X=1
96 NEXT I
99 PRINT AT 14,4;"GOED" AND X=0;"FOUT" AND X=1
```

5. Algemene programma's

De laatste programma's uit dit boek zijn programma's die betrekking hebben op tal van onderwerpen. Het zijn onder andere financiële, statistische, wiskundige en algemene programma's.
Vooral de laatste programma's in dit hoofdstuk zijn interessant voor diegenen die de overstap willen maken naar het programmeren in machinecode. Er zit een programma bij om het binnenste van de ZX81 bloot te leggen. Juist dit laatste hoofdstuk maakt duidelijk hoe veelzijdig een computer is. De computer is niet alleen te gebruiken om spelletjes te spelen, maar ook om uiterst complexe berekeningen uit te voeren zoals de bepaling van de integraal van de normale verdeling of de bepaling van de annuïteit. Hiermee is wel bewezen dat de ZX81 niet een stuk speelgoed is, maar een echte computer.

5.1 HOEVEEL HYPOTHEEK KAN IK KRIJGEN ?

Hypotheken worden altijd berekend aan de hand van het te lenen bedrag. Zeker mensen die in een dure huurwoning zitten vragen zich wel eens af of het niet aantrekkelijker zou zijn om een huis te kopen zonder nog een huis op het oog te hebben.
Dit programma berekent aan de hand van de woonlasten, die men bereid is per maand te betalen, wat daarbij het hypotheekbedrag zou zijn. Natuurlijk zijn nog wat extra gegevens nodig. Om de hoogte van de hypotheek te berekenen moet men de rente weten waartegen men kan lenen. De looptijd moet bekend zijn en men moet weten of men van plan zou zijn maandelijks, driemaandelijks, halfjaarlijks of jaarlijks te betalen. De gebruikelijke looptijd bij hypothecaire leningen is dertig jaar, hoewel vijfentwintig jaar ook voorkomt. De rente van de hypotheeklening kan men wel uit de krant halen, waarbij dan tevens vermeld moet staan of deze rente geldt bij halfjaarlijkse of maandelijkse aflossing. De berekening gaat uit van een normale annuïteitenlening.

Werking van het programma

Het programma vraagt als eerste het bedrag per maand op. Dit is het bedrag dat men maximaal wenst te betalen zonder rekening te houden met belastingvoordeel. De vragen 'rente-percentage' en 'looptijd' spreken voor

zich. De vraag 'hoeveel betalingen per jaar' heeft betrekking op het aflossingsschema. Wordt de lening maandelijks afgelost dan levert dat twaalf betalingen per jaar op. Wordt er om de drie maanden afgelost dan zijn er vier betalingen per jaar. Bij halfjaarlijkse betalingen zijn dat twee betalingen per jaar.

Programma

```
10 PRINT "BEDRAG PER MND ";
20 INPUT B
30 PRINT B
40 PRINT "RENTE-PERC ";
50 INPUT R
60 PRINT R
70 PRINT "LOOPTIJD ";
80 INPUT T
90 PRINT T
100 PRINT "HOEVEEL BETALINGEN PER J. ";
110 INPUT L
120 PRINT L
130 LET Y=1-1/((1+(R/L)/100)**(T*L))
140 PRINT "KAPITAAL ";INT(Y*(B*12/L)/(R/L)+.5)*100
```

5.2 SPAREN

Zo goed als alle bank- en giro-instellingen hebben spaarrekeningen waarop men kan sparen via een automatische overschrijving van een vast bedrag. Dit zijn op zich zeer aantrekkelijke spaarvormen. Het is voor de spaarder echter vaak moeilijk om te bepalen hoe groot ongeveer het kapitaal wordt na een aantal jaren. En juist dit laatste gegeven kan interessant zijn.
Dit programma stelt de gebruiker in staat zulke berekeningen uit te voeren. Om deze berekening uit te voeren zijn een aantal gegevens noodzakelijk.

- Wat is het bedrag dat men denkt per periode weg te zetten?
- Gedurende hoeveel jaar denkt men dat te doen?
- Hoeveel perioden per jaar?
- Tegen welk rentepercentage?

Het aantal perioden per jaar vraagt een nadere uitleg. Meestal wordt gekozen voor een bedrag per maand. Dit betekent dat er gedurende twaalf perioden per jaar een bedrag wordt gestort naar de spaarbank. Het is ook heel goed mogelijk dat men halfjaarlijks een bedrag stort. In dat geval is er sprake van twee perioden. Men moet er op bedacht zijn dat de perioden wel even lang zijn, anders gaat dit programma niet op.

Programma

```
10 PRINT "WELK PERIODE-BEDRAG ";
15 INPUT B
16 PRINT B
```

```
20 PRINT "HOEVEEL JAAR ";
25 INPUT J
26 PRINT J
30 PRINT "HOEVEEL PERIODEN ";
35 INPUT P
36 PRINT P
40 PRINT "RENTEPERC ";
45 INPUT R
46 PRINT R
50 LET K=(R/P/100+1)
55 LET S=((1-K**(J*P+1))/(1-K)-1)*B
56 PRINT
60 PRINT "TOTALE INLEG ";B*J*P
75 PRINT "KAPITAAL ";INT(S*100+.5)/100
```

5.3 KOPEN OP AFBETALING

Dat kopen op afbetaling een dure vorm van geldlenen is, weet ondertussen iedereen wel. Maar hoe duur precies dat weet niet iedereen. Dat is overigens niet zo verwonderlijk, omdat de berekening niet eenvoudig is.
Meestal lopen huurkoopovereenkomsten langer dan een jaar en dat maakt het allemaal wat ingewikkelder. Om de berekening uit te kunnen voeren heeft het programma een aantal gegevens nodig:

- Wat is het bedrag van de overeenkomst?
- Hoeveel maandelijkse termijnen zijn er?
- Wat is het bedrag dat men maandelijks moet betalen?

Het aantal maandelijkse termijnen is meestal 12, 18 of 24 maanden. Het bedrag van de overeenkomst is het bedrag dat men in handen heeft gekregen, dus zonder de eigen aanbetaling.
Aan de hand van deze gegevens kan worden bepaald wat het bedrag is dat men aan rente en kosten betaalt. Dit bedrag is het verschil tussen het geleende bedrag en het bedrag dat men in totaal terugbetaalt. Aan de hand van de looptijd van de lening wordt de gemiddelde looptijd berekend en met deze gegevens kan ten slotte het percentage worden berekend.
Het is best wel eens aardig een staatje uit de radiogids of de krant te pakken, waarin een financieringsinstelling geld aanbiedt en dan te berekenen hoeveel rente men feitelijk betaalt.

Programma

```
10 PRINT "BEDRAG ";
15 INPUT G
16 PRINT G
20 PRINT "MAANDELIJKSE TERMIJN ";
25 INPUT T
26 PRINT T
30 PRINT "MAANDELIJKS BEDRAG ";
35 INPUT B
40 PRINT B
```

```
45 LET S=T*B
80 LET T=(T+1)/2
85 LET R=S-G
86 LET K=(R/T*12)/(G/100)
87 PRINT
90 PRINT "TOTAALBEDRAG ";S
95 PRINT "RENTE+KOSTEN ";R
100 PRINT "PERCENTAGE ";INT(K*100+.5)/100
```

5.4 BEREKENING VAN DE ANNUITEIT

De annuïteit is een vast bedrag dat op vaste tijdstippen aan de bank betaald moet worden. In dit bedrag zit een gedeelte rentevergoeding en een gedeelte aflossing. Dit soort leningen wordt vaak aangegaan bij de aankoop van huizen. Hier spreekt men dan meestal van hypothecaire leningen.
Om zo'n annuïteit te berekenen zijn een aantal gegevens vereist. Ten eerste moet men de hoogte van de lening weten. Ten tweede moet het rentepercentage bekend zijn. Ten derde moet de looptijd gegeven zijn en als vierde punt moet gegeven zijn hoeveel betalingen per jaar plaatsvinden. Met deze vier gegevens, hoogte van de lening, looptijd, rente en aantal betalingen per jaar kan elke annuïteit worden berekend. Dit programma is zeker handig, omdat veel tabellen maar een beperkt aantal annuïteiten opgeven. Meestal lopen deze met 0,5% op. Dit programma stelt de gebruiker in staat elke berekening uit te voeren.
Wat betreft het aantal betalingen per jaar moet men in de gaten houden dat twee betalingen per jaar betekent dat er sprake is van halfjaarlijkse betalingen. Is er sprake van maandelijkse betaling dan zijn er twaalf betalingen per jaar.

Voorbeeld:

Stel, men wil een premie B-woning kopen, die *f*.180.000,- kost. Men heeft de beschikking over *f*.50.000,- eigen geld. De lening, die men wil afsluiten is dan *f*.130.000,- groot tegen een rentepercentage van 9,4% met een looptijd van 30 jaar en maandelijkse aflossing.
De invoer ziet er dan als volgt uit:

```
KAPITAAL 130000
RENTEPERC 9.4
LOOPTIJD 30
HOEVEEL BETALINGEN PER J. 12

ANNUITEIT 1083.64
BEDRAG PER MAAND 1083.64
```

De werking van het programma is recht toe recht aan. Er worden totaal geen controles uitgevoerd. Wanneer de gebruiker gegevens fout intoetst, komt het programma gegarandeerd met verkeerde gegevens. Het is daarom aan te raden, indien men over voldoende ruimte beschikt, controles in te bouwen.

Programma

```
10 PRINT "KAPITAAL ";
20 INPUT K
25 PRINT K
30 PRINT RENTEPERC ";
35 INPUT R
36 PRINT R
40 PRINT "LOOPTIJD ";
45 INPUT T
50 PRINT T
55 PRINT "HOEVEEL BETALINGEN PER J. ";
60 INPUT L
65 PRINT L
70 LET A=(R/L)/100
75 LET X=(1+A)**(T*L)
80 LET B=INT((K*A)/(1-1/X)*100+.5)/100
85 PRINT "ANNUITEIT ";B
90 PRINT "BEDRAG PER MAAND ";B*L/12
```

5.5 DAG VAN DE WEEK

Wanneer men een bepaalde datum heeft, is het met een formule mogelijk de dag van de week te bepalen. Vaak ziet men in agenda's een eeuwigdurende kalender staan waarmee de dag van de week ook te bepalen valt. Dit soort eeuwigdurende kalenders zijn vaak zeer gecompliceerde cijferreeksen. Dit programma lost dit voor de gebruiker op. Het zal opvallen dat veel mensen er van opkijken dat de computer in staat is de dag te bepalen. Wanneer men dit programma voor vrienden en kennissen demonstreert moet men maar eens opletten, meestal worden geboortedata genomen.
In het programma worden geen controles op de dag, maand of jaar uitgevoerd, zodat ze wel goed ingevoerd moeten worden omdat anders het programma met foute uitspraken komt. Eerst wordt het dagnummer gevraagd, daarna de maand en dan het jaar. Het jaar moet worden ingevoerd als een viercijferig getal.

Werking van het programma

In dit programma wordt een van de kunstgrepen uitgehaald, waarop in hoofdstuk 2 nogal diep is ingegaan. Op regel 70 zijn in een PRINT-statement de eerste twee letters van de dagen van de week opgenomen. Nu is het niet de bedoeling dat de hele string wordt afgedrukt, maar wel de initialen van de dag. Het slicingmechanisme zorgt ervoor dat het goede gedeelte van de string alleen wordt afgedrukt. Stel dat de variabele A de waarde 1 bevat. In dat geval moeten alleen de eerste twee letters worden afgedrukt. Is de waarde van A bijvoorbeeld 6 dan moeten de elfde en twaalfde letter worden afgedrukt.
Tegelijkertijd valt op dat men geen twee regels nodig heeft, zoals gesuggereerd in hoofdstuk 2. Het kan in één PRINT-statement ook worden gereali-

seerd. De truc is bruikbaar op de ZX81; of dezelfde truc bij andere computers te gebruiken valt, staat te bezien.

Programma

```
15 PRINT "DATUM DD,MM,JJJJ ";
20 INPUT D
22 PRINT D;
25 INPUT M
26 PRINT M;
30 INPUT J
31 PRINT J;" =";
35 IF M>2 THEN GOTO 50
40 LET J=J-1
45 LET M=M+12
50 LET E=INT(J/100)
55 LET Y=J-E*100
60 LET A=INT(2.6*(M-2)-.2)+D+Y+INT(Y/4)+INT(E/4)-2*E
65 LET A=A-INT(A/7)*7+1
70 PRINT "ZOMADIWODOVRZA" (A*2-1 TO A*2)
```

5.6 TUSSENLIGGENDE DAGEN

Voor een aantal zaken kan het handig zijn het aantal tussenliggende dagen tussen twee data te weten. Meestal is dat een heel gereken. Men moet er rekening mee houden dat juli en augustus beide 31 dagen hebben. Verder dient men rekening te houden met schrikkeljaren, enz. Dit programma houdt met alle bovengenoemde zaken rekening. Het bepalen geschiedt aan de hand van het werkelijke aantal dagen in een maand en niet zoals voor renteberekening gebruikelijk is op basis van dertig dagen.

Werking van het programma

In het programma wordt om twee datums gevraagd, hierbij is het verstandig om de oudste datum het eerst te geven en de jongste datum daarna. Het programma vraagt eerst het dagnummer, daarna het maandnummer en als laatste het jaar in vier cijfers. Deze datum wordt aan een subroutine gegeven, die de datum omrekent in een aantal dagen. Daarna wordt de tweede datum opgegeven op dezelfde manier als de eerste. Deze wordt eveneens aan dezelfde subroutine geleverd, die deze omrekent naar een aantal dagen. Door nu beide aantallen dagen van elkaar af te trekken heeft men het gewenste aantal tussenliggende dagen.
Het zal wel duidelijk zijn bij de berekening van het aantal dagen aan de hand van de eerste datum dat dit aantal dagen in een variabele moet worden onthouden, anders wordt het bij de tweede berekening overschreven. De eerste keer wordt dit aantal dagen onthouden in de variabele F. De tweede keer is dat niet nodig.

Programma

```
5 PRINT " BEGINDATUM DD,MM,JJJJ ";
10 INPUT D
15 INPUT M
20 INPUT J
25 PRINT D;"-";M;"-";J-1900
26 GOSUB 60
27 LET F=L
30 PRINT "EINDDATUM DD,MM,JJJJ ";
35 INPUT D
40 INPUT M
45 INPUT J
50 PRINT D;"-";M;"-";J-1900
55 GOSUB 60
56 PRINT "TUSSENLIGGEND ";ABS(F-L);" DAGEN"
57 STOP
60 LET L=J*365+D
65 LET K=M
70 IF M<=2 THEN GOTO 85
75 LET L=L-INT((K*.4)+2.3)
80 LET J=J+1
85 LET L=L+INT((M*31)+(J-1)/4)
90 IF J=1900 THEN LET L=L+1
95 RETURN
```

5.7 GEMIDDELDE EN STANDAARDDEVIATIE

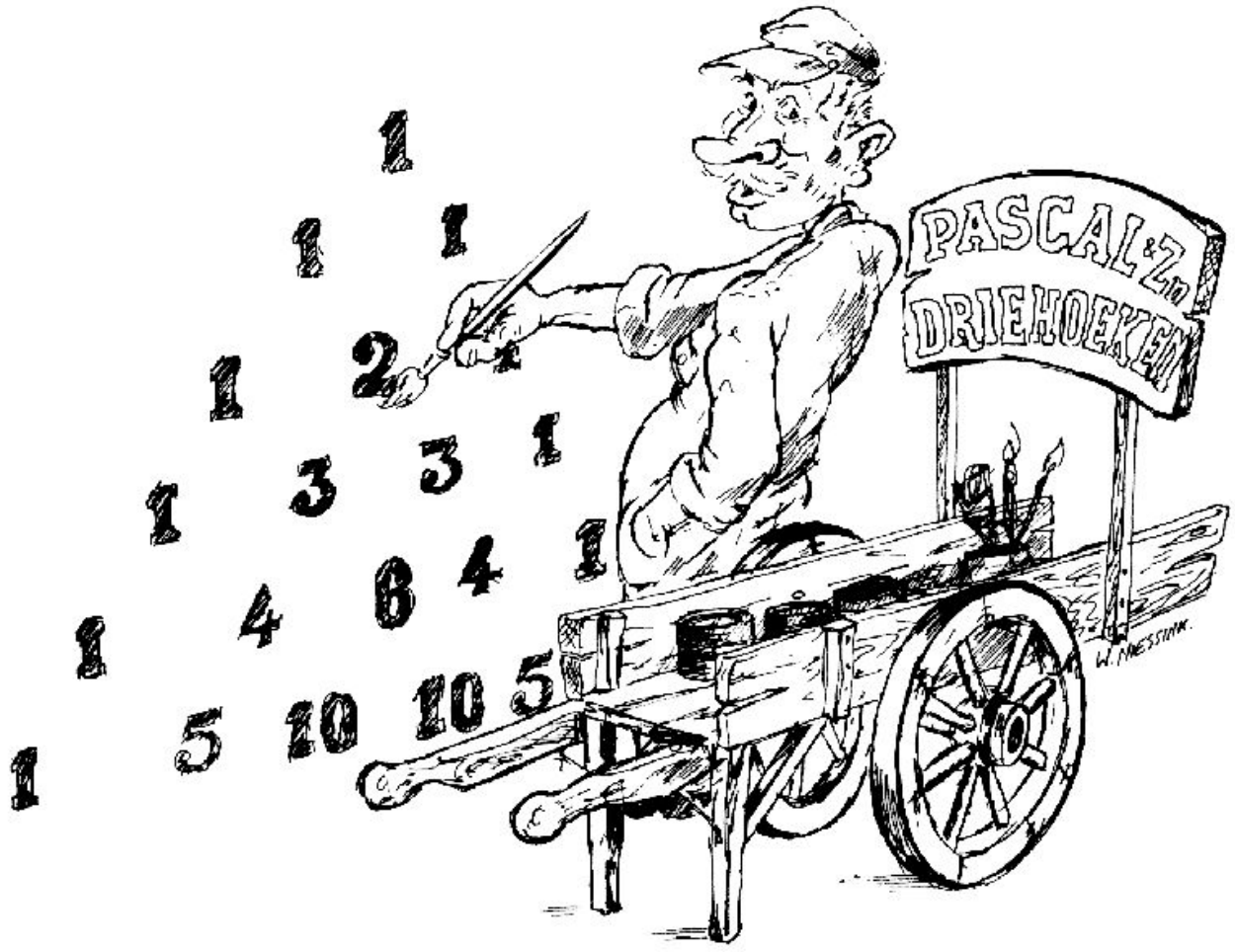
Het gemiddelde en de standaarddeviatie zijn twee grootheden, die een populatie of een steekproef redelijk karakteriseren. Het gegeven programma is voor de bepaling van het gemiddelde en de standaarddeviatie van de populatie. Om het gemiddelde en de standaarddeviatie van een steekproef te berekenen moet men de gebruikte formules iets wijzigen. Hierover later meer.
Het programma vraagt opeenvolgend de gevonden waarden uit de populatie. Tevens wordt bijgehouden hoeveel waarnemingen men inbrengt. Heeft men geen waarnemingen meer dan moet men STOP intoetsen. Verder moet men er bij dit programma op bedacht zijn dat men geen fouten bij de invoer maakt. Gebeurt dit toch dan zal men helemaal opnieuw moeten beginnen.
De waarnemingen worden als string ingevoerd en daarna pas geconverteerd naar een numerieke waarde. Op deze manier is het ook mogelijk om STOP in te toetsen als teken dat men geen invoer meer heeft.
Bij bepaling van de standaarddeviatie uit de steekproef moet er rekening mee worden gehouden dat men niet door T deelt maar door (T-1)!

Programma

```
10 LET S=0
15 LET M=0
```

```
20 LET T=0
25 SCROLL
30 PRINT "GEEF WAARDE ";T+1,
40 INPUT A$
45 PRINT A$
50 IF A$="STOP" THEN GOTO 85
60 LET T=T+1
70 LET S=S+VAL(A$)
75 LET M=M+(VAL(A$))**2
80 GOTO 25
85 CLS
90 PRINT "GEMIDDELDE ";S/T
95 PRINT "STANDAARDDEV. ";SQR(M/T-(S/T)**2)
```

5.8 DE DRIEHOEK VAN PASCAL



Men kan de binomiaalcoëfficiënten zeer overzichtelijk rangschikken in een driehoek. Deze driehoek wordt de driehoek van Pascal genoemd, naar de bekende Franse wiskundige Blaise Pascal, die leefde van 1623 tot 1662.
Behalve de driehoek is ook nog een computertaal naar deze geleerde genoemd.
Met behulp van de driehoek van Pascal is een tweeterm in de vorm $(A+B)^n$ gemakkelijk oplosbaar. Het programma lost dit razendsnel op. Het enige dat men hoeft op te geven is het regelnummer van de driehoek. Bijvoor-

beeld $(A+B)^2 = A^2 + 2AB + B^2$. Het regelnummer dat hierbij hoort is 2. Het programma geeft dan de coëfficiënten weer. In het geval van het bovenstaande voorbeeld zijn dat 1, 2 en 1. Zoals men ziet zijn tweetermopgaven geen probleem meer met dit programma. Of de macht nu 4 of 14 is, dat maakt voor het programma niets uit.
Overigens laat dit programma zien hoe complex lijkende onderwerpen eenvoudig kunnen worden opgelost.

Programma

```
10 PRINT "GEEF RIJ-NR ";
20 INPUT R
25 PRINT R
30 LET N=R
40 GOSUB 150
50 LET Q=Z
60 FOR J=0 TO R
70 LET N=R-J
80 GOSUB 150
90 LET X=Z
100 LET N=J
110 GOSUB 150
120 PRINT Q/(X*Z);"□";
130 NEXT J
135 PRINT
140 GOTO 10
150 LET Z=1
160 FOR J=1 TO N
170 LET Z=Z*J
180 NEXT J
190 RETURN
```

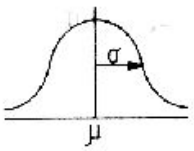
5.9 DE NORMALE VERDELING

De normale verdeling wordt erg veel toegepast bij statistische berekeningen. De kromme is ontwikkeld door de Duitse astronoom Karl Frederick Gauss. Bij de berekening van banen van hemellichamen werden toevallige fouten gemaakt. Gauss was in staat een verdeling te maken van deze fouten en zo is de normale verdeling ontstaan.
De normale verdeling wordt dan ook in tal van wetenschappen gebruikt. Verder dient men in de gaten te houden dat de normale verdeling een wiskundig model is en geen natuurwet, zoals die van de zwaartekracht.
De formule van de normale verdeling heeft de volgende vorm:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-1/2\left(\frac{x-\mu}{\sigma}\right)^2}$$

Waarbij e : Het getal e is 2.718...
μ : Het rekenkundig gemiddelde

σ : De standaarddeviatie
π : Het getal π is 3,141...



Bij een gemiddelde van 0 en een standaarddeviatie van 1 is de oppervlakte onder de curve gelijk aan 1 ofwel 100%. In veel statistische boeken staan achterin hele tabellen voor het berekenen van de oppervlakte of de kans. De berekening geschiedt aan de hand van de Z-waarde, welke de mate van excentriciteit voorstelt. De Z-waarde is gelijk aan het aantal malen dat de standaardafwijking ligt van het gemiddelde. Stel het gemiddelde is 12 en de standaardafwijking is 2. Men wil de kans weten dat X ligt tussen 12, het rekenkundig gemiddelde en 14. De formule om de mate van excentriciteit te berekenen is:

$$Z = \frac{x - \mu x}{\sigma_x}$$

In het bovenste voorbeeld is $Z = \frac{14 - 12}{2}$

Werking van het programma

Het programma vraagt om de Z-waarde. Aan de hand van deze waarde wordt de integraal of de oppervlakte bepaald van 0 tot aan de Z-waarde. De uitgevoerde berekening wordt een iteratief proces genoemd, omdat een zelfde berekening vele malen herhaald wordt. Wanneer men het programma goed bekijkt, blijkt het een numerieke benadering te zijn van de integraal. Door de kleine stapgrootte in de lus op regel 80, die door de variabele F wordt gestuurd, duurt het programma vrij lang. Vergroot men de waarde in de variabele F dan verloopt het programma wel sneller, maar daardoor wordt de integraal onnauwkeuriger. Het zal duidelijk zijn, wanneer men de stapgrootte nog kleiner kiest, dat de integraal beter wordt.

Programma

```
10 PRINT "Z-WAARDE ";
20 INPUT Z
30 PRINT Z
31 FAST
32 LET Z=ABS(Z)
35 LET E=EXP(1)
40 LET S=1/SQR(2*PI)
45 LET F=.01
50 LET A=F
60 LET B=Z-F
```

```
70 LET T=0
80 FOR X=A TO B STEP F
90 LET T=T+S*E**(-X**2/2)
100 NEXT X
110 LET B=S*E**(-Z**2/2)
125 SLOW
130 PRINT F/2*(S+2*T+B)
```

5.10 CIRKELS

Het programma 'Cirkels' produceert een plaatje van 7 cirkels, die in elkaar zijn afgedrukt. De gebruikte formule is gebaseerd op de formule van de cirkel $X^2+Y^2=R^2$.Wanneer men deze formule omwerkt, komt men op de formule voor X=C*cosinus P en voor Y=C*cosinus P. Waarbij de straal van de cirkel gelijk is aan de waarde van de variabele C. De variabelen A en B geven aan waar het middelpunt van de cirkel ligt.
De afleiding van deze formule valt buiten het kader van dit boek, maar men kan deze afleiding in elk wiskundeboek vinden.

Werking van het programma

De ZX81 berekent de goniometrische functies aan de hand van radialen en niet aan de hand van graden. Dit betekent dat er een omzetting moet plaatsvinden van radialen naar graden. Dit is te bereiken door de hoek te vermenigvuldigen met PI/180. De omtrek van een cirkel met een straal van 1 is gelijk aan 2*PI. De omtrek van de cirkel is gelijk aan 360 graden. Uit bovenstaande vergelijking volgt dat 2PI/360 precies de omrekenfactor is van graden naar radialen. Heeft men deze omrekening gedaan dan kan men wél met de goniometrische functies rekenen.

Programma

```
10 LET A=20
20 LET B=20
21 FOR C=15 TO 1 STEP -2
30 FOR X=0 TO 360 STEP 4
40 LET P=X*PI/180
50 PLOT A+C*COS P,B+C*SIN P
60 NEXT X
70 NEXT C
```

5.11 HISTOGRAM

Een grafiek zegt meer dan 1000 getallen. Alleen is het meestal een vervelend werkje een grafiek te tekenen. Daarom is het bijzonder handig als de ZX81 dit overneemt. Het wordt helemaal gemakkelijk wanneer de ZX81 ook zorgt dat de verhouding in stand blijft.
Het programma vraagt om tien waarden, die overeenstemmen met de sta-

ven van het histogram. De waarden, die men invoert moeten nul of positief zijn. Heeft het programma alle tien de waarden, dan wordt hiervan een liggend staafdiagram afgedrukt.

Werking van het programma

Aan de hand van de ingevoerde waarden wordt de hoogste waarde opgespoord (variabele A). De hoogste waarde bepaalt de langste kolom. Nu kan de langste kolom maar 22 printposities innemen. Er moet dus een correctiefactor worden bepaald om de hoogste waarde gelijk te maken aan 22. Dit is dan tegelijkertijd de omrekeningsfactor voor alle getallen. Op deze manier blijven de verhoudingen tussen de getallen gelijk. Het quotiënt van 22 gedeeld door de hoogste waarde die optreedt is de omrekeningsfactor. Het quotiënt is het getal waarmee elke waarde wordt vermenigvuldigd en afhankelijk van die waarde worden er zwarte blokjes op het beeldscherm gezet. Bij de hoogste waarde worden dan automatisch 22 zwarte blokjes afgedrukt.
Voor alle duidelijkheid worden de kolommen genummerd en wordt de ingevoerde waarde ook nog vermeld.

Programma

```
10 DIM A(10)
20 FOR I=1 TO 10
25 PRINT "STAAF ";I
30 INPUT A(I)
35 PRINT A(I)
40 IF I=1 THEN LET A=A(I)
45 IF A(I)>A THEN LET A=A(I)
55 IF I=5 THEN CLS
60 NEXT I
65 CLS
70 LET A=ABS(22/A)
75 FOR I=1 TO 10
80 PRINT I;TAB 3;A(I);TAB 7;
85 FOR J=1 TO A(I)*A
90 PRINT "■";
95 NEXT J
100 PRINT
105 PRINT
110 NEXT I
```

5.12 DE DRIE HOEKEN VAN EEN DRIEHOEK

Hoe vaak gebeurt het niet dat men wel de lengte van de zijden van een driehoek kent, maar niet de hoeken zelf. Het is meestal een ingewikkelde klus dit met de hand uit te rekenen, maar met een computer is het niet zo moeilijk meer. Hier is het programma om dit te berekenen.
In het programma zijn geen controles ingebouwd. Het kan voorkomen

dat een foutmelding ontstaat. Dit gebeurt als de arcsinus 1 wordt berekend. De ZX81 begint dan te sputteren. Een oplossing is om vooraf te testen of inderdaad die uitzonderingssituatie dreigt. In dit geval moet men de betreffende hoek waar dit voor geldt op 90 graden stellen.
In dit programma wordt ook weer gebruik gemaakt van conversie. Dit keer niet van graden naar radialen, maar omgekeerd van radialen naar graden. De conversiefactor dient dan geïnverteerd te worden. Dus 180/PI.
Bij de oplossing van dit probleem is gebruik gemaakt van de hieronderstaande formules. De genoemde variabelen komen overeen met die in het programma.

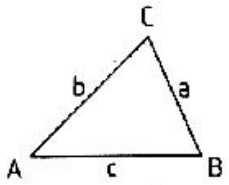
$$\angle A=\sin^{-1}\frac{2S}{bc}$$

$$\angle B=\sin^{-1}\frac{2S}{ca}$$

$$\angle C=180°-\angle A-\angle B$$

$$S=\sqrt{Q(Q-a)(Q-b)(Q-c)}$$

$$Q=\frac{(a+b+c)}{2}$$



Voert men bijvoorbeeld voor alle zijden de waarde 1 in, dan moet als uitkomst voor alle hoeken de waarde 60 verschijnen.

Programma

```
5 LET D=180/PI
10 PRINT "ZIJDE A ";
15 INPUT A
20 PRINT A,,"ZIJDE B ";
25 INPUT B
30 PRINT B,,"ZIJDE C ";
40 INPUT C
42 PRINT C
45 LET Q=(A+B+C)/2
50 LET S=SQR((Q*(Q-A)*(Q-B)*(Q-C)))*2
55 LET E=ASN(S/(B*C))*D
60 LET F=ASN(S/(A*C))*D
65 LET G=180-E-F
70 PRINT "HOEK A ";E
75 PRINT "HOEK B ";F
80 PRINT "HOEK C ";G
```

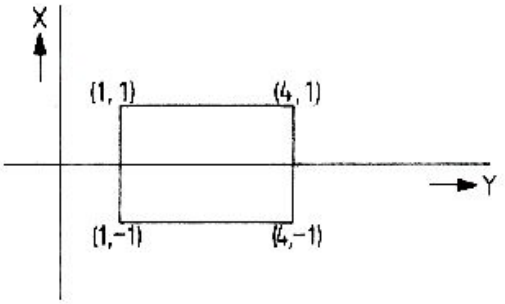
5.13 DE OPPERVLAKTE VAN EEN POLYGOON

De oppervlakte van een polygoon is in de praktijk vaak erg moeilijk te bepalen. Probeer maar eens de oppervlakte van een stad of van een meer te

bepalen; dat zal zeker niet meevallen. Wanneer men de oppervlakte van een polygoon wil bepalen moet men van elk hoekpunt de coördinaten kennen. Daarna voert men deze coördinaten in waarbij men er wel op moet letten dat de volgorde blijft bestaan, anders klopt de uitkomst niet. Fouten maken is er niet bij, want het programma heeft geen correctieroutine. Heeft men alle X-Y-coördinaten ingevoerd dan geeft men bij X -999 en bij Y -999. Heeft men deze waarden ingevoerd dan wordt het laatste stukje van het programma uitgevoerd en verschijnt de oppervlakte.
Bij de berekening van de oppervlakte van een polygoon is gebruik gemaakt van de hierna volgende formule:

$$S = 1/2((X1-X2)(Y1+Y2)+(X2-X3)(Y2+Y3) \ldots+(Xn-X1)(Yn+Y1))$$

Als voorbeeld kan onderstaande figuur dienen. De oppervlakte die hieruit moet komen is 6.



Programma

```
10 LET S=0
15 LET I=0
20 LET I=I+1
25 CLS
30 PRINT "GEEF X ";I;
40 INPUT A
45 PRINT " ";A,,"GEEF Y ";I;
50 INPUT B
55 PRINT " ";B
60 IF I<>1 THEN GOTO 80
65 LET Q=A
70 LET Z=B
75 GOTO 90
80 IF A=-999 AND B=-999 THEN GOTO 105
85 LET S=S+(X-A)*(Y+B)
90 LET X=A
95 LET Y=B
100 GOTO 20
105 LET S=(S+(X-Q)*(Y+Z))/2
110 PRINT "DE OPPERVLAKTE ";S
```

5.14 WEERSTANDEN

Een van de meest vervelende zaken voor een elektronicahobbyist is het onthouden van de kleurcodes van weerstanden. Meestal is dat een moeilijk gezoek en gereken, zeker voor iemand die er niet regelmatig mee omgaat en maar incidenteel iets bouwt. Wat zou het dan gemakkelijk zijn dit klusje aan de computer te geven en deze het werk te laten uitvoeren. De twee programma's doen eigenlijk alle twee hetzelfde; zij converteren kleuren naar ohms of omgekeerd. Het zou natuurlijk mooier zijn geweest, wanneer zij in een programma zouden zijn ingebouwd, maar ruimtegebrek dwong ons tot deze oplossing. In het programma OHM geeft men de drie kleuren van de weerstand op, het programma bepaalt daarna het aantal ohms. De berekening is niet zo belangrijk, die kan men in een elektronicaboek vinden. Wat wel aardig is, dat ook hier weer gebruik wordt gemaakt van het slicingmechanisme om te bepalen welke kleur was ingevoerd.
De herkenning van de kleur wordt bepaald door de eerste drie letters van de kleur. Deze moeten dus kunnen worden teruggevonden in het interne bestand. Afhankelijk van het rangnummer in de string (het interne bestand) wordt deze waarde toegekend aan het element van de matrix. Wanneer dit voor alle drie de kleuren is uitgevoerd is het probleem opgelost. Er hoeft alleen nog maar een berekening te worden uitgevoerd en klaar is de ZX81.
De werking van het programma WEERSTAND is precies andersom.

Programma 1 (kleurcode naar ohms)

```
1 DIM A(3)
10 LET A$="ZWABRUROOORJGEEGROBLAVIOGRYWIT"
20 FOR I=1 TO 3
30 PRINT "GEEF KLEUR ";I
40 INPUT B$
45 PRINT B$
50 FOR J=1 TO 10
60 IF B$(1 TO 3)<>A$(J*3-2 TO J*3) THEN GOTO 70
65 LET A(I)=J-1
70 NEXT J
80 NEXT I
90 PRINT " O H M= ";(10*A(1)+A(2))*10**A(3)
```

Programma 2 (ohms naar kleurcode)

```
5 LET A$="ZWABRUROOORJGEEGROBLAVIOGRYWIT"
10 PRINT "GEEF DE WEERSTAND IN OHM"
20 INPUT R
25 PRINT R;" = ";
30 LET A=INT(LN(R)/LN(10))
40 LET B=A-1
50 LET C=INT(R/10**A+.1)
60 LET D=INT((R-C*10**A+.1)/10**B)
70 PRINT A$((1+C)*3-2 TO (1+C)*3);" ";
75 PRINT A$((1+D)*3-2 TO (1+D)*3);" ";
80 PRINT A$ ((1+B)*3-2 TO (1+B)*3)
```

5.15 BETROUWBAARHEIDSINTERVALLEN

De beide programma's die hierna volgen berekenen beide betrouwbaarheidsintervallen. Er is echter één verschil, het programma 5.15a berekent het interval op basis van het rekenkundig gemiddelde en de standaarddeviatie van de steekproef, terwijl 5.16b de betrouwbaarheidsintervallen berekent op basis van de proportie of het percentage.
De betrouwbaarheidsintervallen kunnen worden gebruikt bij het toetsen van hypothesen. Men dient er wel rekening mee te houden dat er van uitgegaan wordt dat er sprake is van tweezijdige toetsing. Verder moet de grootte van de steekproef groter zijn dan 50 trekkingen, anders kan er een te grote fout optreden.
Door de programma's worden onder andere de volgende betrouwbaarheidsintervallen berekend: 99%, 95%, 90% en 80%.

Werking van de programma's

De beide programma's maken gebruik van dezelfde truc om met de goede factor te rekenen. De factoren met de bijbehorende percentages zijn afkomstig van de normale verdeling. Om ruimte te sparen is de factor en het percentage in een getal verpakt en wel op een zodanige manier dat de beide getallen weer gescheiden kunnen worden. Achter de decimale punt van een element van de matrix A is het percentage opgeslagen. Vóór de decimale punt is de factor opgeslagen, maar deze is wel met 100 vermenigvuldigd. Bijvoorbeeld, in de programma's komt A(1)=257.99 voor. Het betrouwbaarheidspercentage is 99 en de factor is 2,57. Om het percentage uit het element van de matrix te krijgen moet de volgende manipulatie worden uitgehaald: A=INT((A(1)-INT(A(1)) *100. Alleen op dit punt gaat er iets mis. Het blijkt dat in bepaalde gevallen de waarde van A niet exact gelijk is aan de waarde die wordt gewenst. Waarom dit gebeurt valt buiten de strekking van dit boek, maar het heeft iets te maken met interne getalrepresentatie. Om dit probleem nu op te lossen wordt een afrondingsbewerking gedaan: A=INT(a+0.5).
De factor zelf is nog gemakkelijker te bepalen. Hiertoe worden de cijfers achter de decimale punt afgekapt en het resultaat door honderd gedeeld. In formulevorm ziet dat er als volgt uit: A=INT(A(1))/100.
Wanneer men meer over betrouwbaarheidsintervallen en hun toepassingen wil weten kan men het beste eens in statistische leerboeken kijken.

Programma 1 (op basis van steekproef)

```
5 DIM A(4)
10 LET A(1)=257.99
15 LET A(2)=195.95
20 LET A(3)=164.90
25 LET A(4)=128.80
30 PRINT "GEMIDDELDE IN STEEKPR. ";
35 INPUT G
40 PRINT " ";G;"STANDAARDDEV IN STEEKPR. "
45 INPUT S
```

```
50 PRINT " ";S
52 PRINT "GROOTTE STEEKPR. ";
55 INPUT N
60 PRINT " ";N
65 LET P=S/SQR(N)
70 CLS
75 PRINT "BI";TAB 5;"ONDERGR.";TAB 15;"BOVENGR."
80 FOR I=1 TO 4
85 LET A=(A(I)-INT(A(I)))*100
90 PRINT INT(A+.5);
95 LET A=INT(A(I))/100
100 PRINT TAB 5;INT((G-P*A)*100)/100;TAB 15;
    INT((G+P*A)*100)/100
105 NEXT I
```

Programma 2 (op basis van proportie)

```
5 DIM A(6)
6 LET Z=100
10 LET A(1)=257.99
15 LET A(2)=195.95
20 LET A(3)=164.90
25 LET A(4)=128.80
26 LET A(5)=103.70
27 LET A(6)=84.60
30 PRINT "PERCENTAGE IN DE STEEKPR. ";
35 INPUT P
40 PRINT " ";P
52 PRINT "GROOTTE STEEKPR. ";
55 INPUT N
60 PRINT " ";N
61 LET P=P/Z
65 LET V=SQR(P*(1-P)/N)
70 CLS
75 PRINT "BI";TAB 5;"ONDERGR.";TAB 15;"BOVENGR.";
80 FOR I=1 TO 6
85 LET A=(A(I)-INT(A(I)))*Z
90 PRINT INT(A+.5);
95 LET A=INT(A(I))/Z
100 PRINT TAB 5;INT((P-V*A)*Z*Z)/Z;TAB 15;
    INT((P+V*A)*Z*Z)/Z
105 NEXT I
```

De laatste programma's uit dit hoofdstuk zijn voor al diegenen voor wie BASIC geen geheimen meer heeft en die het inwendige van de ZX81 willen onderzoeken. Of voor hen, die in machinecode willen gaan programmeren.
Meestal worden in computers andere talstelstels gehanteerd dan waarmee men gewend is te rekenen. In het dagelijks leven wordt gewerkt met het tientallig stelsel. Dit is het stelsel met het grondtal tien ofwel met radix 10. Andere talstelsels zijn ook mogelijk, bijvoorbeeld het binaire

of tweetallige stelsel. De radix is dan 2. Programma 5.16 biedt de gebruiker de mogelijkheid om een waarde in het decimale talstelsel om te zetten naar een eigen gekozen talstelsel. Het programma 5.17 werkt net andersom. Hierbij geeft men de waarde op in een gekozen talstelsel en de waarde wordt dan geconverteerd naar het decimale stelsel.
Programma 5.20 biedt de mogelijkheid om geheugendumps te maken van de ZX81. Per geheugenlocatie wordt de inhoud in het binaire talstelsel en in het hexadecimale talstelsel afgedrukt.
Vooral één gedeelte van het ROM-geheugen biedt leuke toepassingen die met dit programma gevonden zijn.
De laatste programma's zijn gemaakt op basis van een stukje ROM-geheugen.

5.16 VAN RADIX 10 NAAR ANDERE RADIX

In het decimale talstelsel is de radix 10, in het binaire talstelsel is de radix 2 en in het hexadecimale talstelsel is de radix 16. Bij al deze talstelsels worden voor de eenvoud een aantal karakters per groep bij elkaar gezet. Bijvoorbeeld bij het binaire talstelsel zijn dat vier karakters, bij het octale talstelsel zijn dat er meestal drie. Deze extra vraag is toegevoegd om de leesbaarheid van het getal in het andere talstelsel te bevorderen. Na deze vraag verschijnt de vraag RADIX?: hier geeft men op naar welk grondtal men wil converteren. Bij hexadecimaal is dit 16. Wil men bijvoorbeeld naar het 18-tallig stelsel converteren dan geeft men 18 op. Ten slotte moet de decimale waarde worden ingevoerd, die geconverteerd moet worden naar het eerder genoemde grondtal. Na de berekening en de uitvoer stopt het programma. Door het indrukken van een willekeurige toets start het programma opnieuw.
Men dient er op bedacht te zijn dat het minst significante karakter rechts staat en het meest significante links.
Uit het programma komt duidelijk naar voren dat gebruik gemaakt wordt van de decimale waarden van de karakters, die in appendix A van de Engelstalige handleiding zijn genoemd. Het cijfer nul heeft als decimale waarde 28. Achter de numerieke tekens komen in volgorde de alfabetische tekens, zodat hierdoor de berekening vrij eenvoudig wordt. Bij het restant van de deling hoeft alleen maar 28 te worden bijgeteld om het af te drukken karakter te krijgen. Juist de decimale karaktertabel gaf de mogelijkheid dit programma zo eenvoudig te houden.
De theorie achter het programma, de conversie van het ene talstelsel naar het andere talstelsel, staat onder andere beschreven in het boek 'Microcomputers' van A.J.Derksen.

Programma

```
10 CLS
11 PRINT "HOEVEEL KAR. PER GROEP"
12 INPUT K
13 LET X=0
14 PRINT "RADIX? ";
15 INPUT R
```

```
20 PRINT R
25 PRINT "WAARDE IN 10 ";
30 INPUT W
31 PRINT W
32 LET I=25
33 PRINT "RESULTAAT"
34 LET I=I-1
35 LET S=INT(W/R)
40 LET T=W-S*R
43 IF X/K-INT(X/K)=0 THEN LET I=I-1
44 LET X=X+1
45 PRINT AT 10,I;CHR$(T+28)
50 LET W=S
55 IF W>0 THEN GOTO 34
60 PAUSE 4E4
65 RUN
70 GOTO 10
```

5.17 VAN ANDERE RADIX NAAR RADIX 10

De invoer van dit programma is precies tegenovergesteld aan het programma hiervoor. De invoergegevens die nodig zijn om de decimale waarde te berekenen zijn de radix van de in te voeren waarde en de waarde zelf in de betreffende radix. Met dit programma is men in staat binaire en hexadecimale waarden te vertalen in decimale waarden.

Werking van het programma

Het programma vraagt om de radix van de te converteren waarde. Bij een octaal getal is dit bijvoorbeeld 8, bij een hexadecimaal getal is dit 16. Daarna wordt de te converteren waarde gevraagd door het programma. Omdat het mogelijk is dat er in de waarde lettertekens voorkomen, wordt de waarde opgevangen in een string. In het programma wordt geen controle uitgevoerd op de inhoud van de string. Indien men foutieve invoer geeft, kan het programma gaan stotteren. Bijvoorbeeld een getal M in radix 16 is fout, omdat het cijfer F het hoogste cijfer is.
Na invoer van de gevraagde gegevens wordt de omzetting uitgevoerd. Hierna stopt het programma. Door het indrukken van een willekeurige toets start het programma opnieuw. Het is duidelijk dat zonder het slicingmechanisme en de decimale waarden van de karakters dit programma niet mogelijk was.

Programma

```
5 CLS
10 PRINT "RADIX ";
20 INPUT R
25 PRINT R
30 PRINT "WAARDE IN RADIX ";R;
```

```
35 INPUT A$
40 PRINT A$
41 LET S=0
45 FOR I=1 TO LEN A$
50 LET C=CODE A$(I)-28
55 LET S=S+C*R**(LEN A$-I)
60 NEXT I
65 PRINT "IN DECIMAAL ";S
70 PAUSE 4E4
75 RUN
80 GOTO 5
```

5.18 GEHEUGENDUMP

Dit programma is ontstaan uit pure nieuwsgierigheid. De ontwerpers van de ZX81 hebben eigenlijk maar weinig informatie verstrekt over de inhoud van de ROM. In de handleiding is er wel iets over verteld, kijk maar in hoofdstuk 26 t/m 28. Er zitten echter ook hele leuke dingen in de ROM, die zelfs voor iemand die nog maar net met programmeren bezig is, heel interessant kunnen zijn.
Met name de geheugenlocaties 7680 tot en met 8191 zijn heel belangrijk om eens te bekijken. Het blijkt dat dit de adressen zijn waar de karaktergenerator zich bevindt. Men doet er verstandig aan dit met blokken van acht adressen te bekijken. Dit loopt dan namelijk gelijk met de hoogte van de karakters. Verder is het mogelijk om te zien hoe BASIC-programma's zijn opgeslagen.

Werking van het programma

Het programma vraagt om het decimale startadres, daarna om het decimale eindadres. Als resultaat verschijnt op het beeldscherm het adres en de inhoud van het adres in binaire en in hexadecimale vorm. Op deze manier is bestudering vrij gemakkelijk.
Het programma maakt gebruik van een subroutine uit het vorige programma. Aan de subroutine worden de variabele R met de waarde van de radix gegeven en de variabele Z. Deze laatste variabele geeft aan uit hoeveel karakters de geconverteerde waarde zal gaan bestaan. Uit de subroutine komt een string terug met daarin de geconverteerde waarde van de geheugenlocatie.

Programma

```
15 PRINT "VANAF WELK ADRES"
20 INPUT B
25 PRINT "TOT WELK ADRES"
30 INPUT F
35 FOR X=B TO F
45 LET R=16
50 LET Z=2
```

```
55 GOSUB 94
60 LET B$=A$
65 LET R=2
67 LET Z=8
70 GOSUB 94
75 SCROLL
80 PRINT X;TAB 10;A$;TAB 20;B$
85 NEXT X
90 STOP
94 LET W=PEEK X
95 LET A$=""
96 FOR Q=1 TO Z
100 LET S=INT(W/R)
110 LET A$=CHR$(W-S*R+28)+A$
115 LET W=S
126 NEXT Q
130 RETURN
```

5.19 GROTE KARAKTERS

Bij het programma 'Geheugendump' bleek dat er in de ROM van de ZX81 kon worden gekeken. Er kwam onder andere naar voren dat de karaktergenerator zich op geheugenlocatie 7680 tot en met 8191 bevindt. Bij veel microcomputers is het niet mogelijk de karaktergenerator te benaderen. Bij dat soort computers moeten allerhande trucs worden verzonnen om grote karakters op het beeldscherm te krijgen. Het gevolg is dat er een bijzonder groot en complex programma nodig is. Bij de ZX81 is dat niet nodig. Een klein programma laat al zien hoe dit werkt.
Wanneer met het geheugendump-programma naar de karaktergenerator is gekeken, is het direct opgevallen dat de enen de zwarte velden vertegenwoordigen en de nullen de witte velden in de letter.

Werking van het programma

De gebruiker geeft aan het programma een karakter op. Hiervan wordt met de CODE-functie de decimale waarde bepaald. Nu is de relatieve plaats bekend van het karakter in de karaktergenerator. Elk karakter bestaat uit acht lijnen boven elkaar, acht opeenvolgende geheugenlocaties. De plaats waar een karakter begint in de karaktergenerator is dan te berekenen. Het beginadres is dan het beginadres van de karaktergenerator plus acht maal de decimale waarde van het ingevoerde karakter. Daarna is het een kwestie van een lus acht maal laten doorlopen die de opeenvolgende geheugenlocaties PEEKt. Met behulp van een klein beetje rekenwerk worden de enen omgezet in zwarte blokjes en de nullen in spaties.

Programma

```
10 PRINT "KARAKTER ";
20 INPUT A$
```

```
25 PRINT A$
30 LET C=CODE A$
40 LET S=7680
50 FOR R=0 TO 7
60 LET P=PEEK(S+C*8+R)
70 FOR H=8 TO 1 STEP -1
80 LET L=2**H
90 IF P<L THEN GOTO 120
100 PRINT AT R+3,8-H;"■"
110 LET P=P-L
120 NEXT H
130 NEXT R
```

5.20 RECLAMETEKST

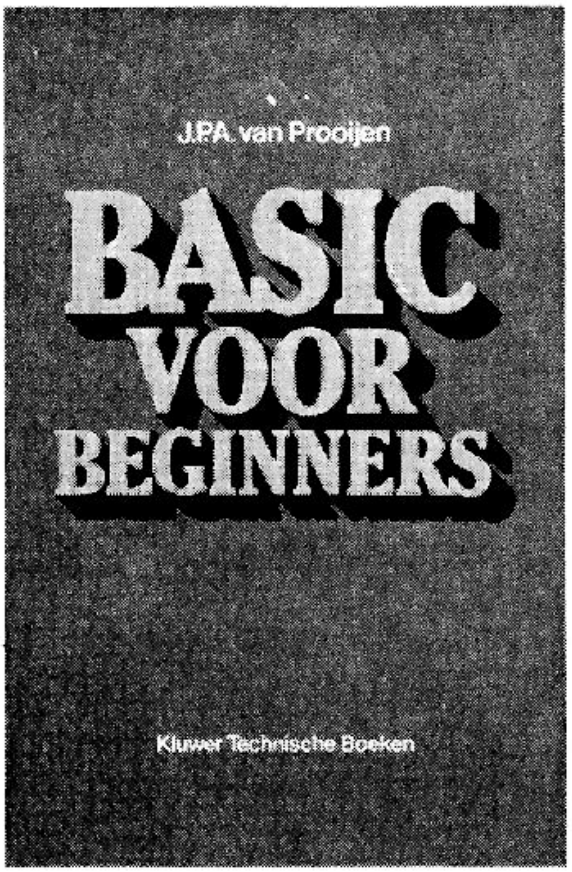
Dit programma doet hetzelfde als het voorgaande programma met dit verschil dat de gebruiker een tekst kan invoeren. Deze tekst wordt dan vergroot met hetzelfde programma als 5.19. Jammer genoeg kan de tekst niet groter zijn dan vier karakters en scrollen is ook niet mogelijk vanwege de beperkte geheugenruimte van de 1K-versie van de ZX81. In beide programma's is gewerkt met zwarte blokjes. In principe moet het ook mogelijk zijn hetzelfde te realiseren met het PLOT-statement, maar dat mag men zelf eens uitzoeken! Met beide programma's als voorbeeld moet dat zeker lukken.

Programma

```
1 LET S=7680
10 DIM A$(4)
20 PRINT "TEKST ";
30 INPUT A$
40 PRINT A$
50 FOR I=1 TO LEN A$
60 LET C=CODE A$(I)
70 FOR R=0 TO 7
80 LET P=PEEK(S+C*8+R)
90 FOR H=8 TO 1 STEP -1
100 LET L=2**H
110 IF P<L THEN GOTO 140
120 PRINT AT R+4,(I-1)*8+8-H;"■"
130 LET P=P-L
140 NEXT H
150 NEXT R
160 NEXT I
```

BASIC voor beginners
J.P.A. van Prooyen

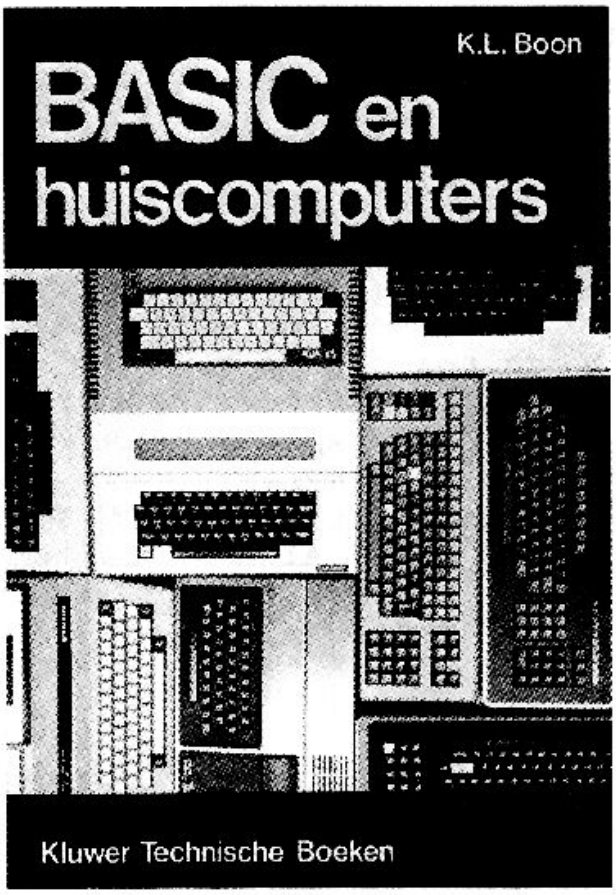
BASIC is de meest populaire taal onder de microcomputergebruikers. Dit boek is bestemd voor diegenen die direct aan de slag willen. Het geeft geen ellenlange inleidingen over de werking van de computer of over moeilijke programmeer technieken, maar biedt direct de behandeling van de BASIC-instructies, afgewisseld met programmavoorbeelden.



94 pagina's, formaat 14,5 x 21,5 cm.
ISBN 90 201 1257 0

BASIC en huiscomputers
K.L.Boon

Dit boek maakt de potentiële koper of gebruiker van een microcomputer vertrouwd met de begrippen en eigenschappen van die computer. Omdat BASIC dé programmeertaal is voor huiscomputers, bevat dit boek tevens een zeer uitvoerige cursus in die taal.
Door de uitgewerkte programmavoorbeelden kan de lezer zijn kennis testen en zal hij gestimuleerd worden om zelf te gaan programmeren.



152 pagina's, formaat 14,5 x 21,5 cm.
ISBN 90 201 1305 4

ZX81
A.Sickler

De ZX81 zal binnenkort verreweg de meest verkochte huiscomputer zijn. Er wordt zelfs voorspeld dat er meer ZX81 computers verkocht zullen worden dan alle computers te zamen. De reden is zoals altijd eenvoudig:
de ZX81 is goedkoop.
Men mag verwachten dat velen de ZX81 zullen gebruiken voor een eerste kennismaking met computers en programmeren.
Dit boek levert voor al deze geïnteresseerden een praktische handleiding.

168 pagina's, formaat 14,5 x 21,5 cm.
ISBN 90 201 1515 4

Zakboekje voor de ZX81
Instructiesets, tabellen en converlijsten
Wessel Akkermans

De ZX81 is in een jaar tijd de meest verkochte personal computer geworden. Om de tienduizenden bezitters te helpen bij het snel opzoeken van de bij het programmeren van hun ZX81 benodigde gegevens heeft de auteur dit zakboekje samengesteld. Het boekje bevat onder meer instructiesets voor BASIC en voor machinetaal, tabellen met karaktersets en conversielijsten met behulp waarvan decimale getallen kunnen worden omgezet in hexadecimaal, octaal of binair en omgekeerd.

96 pagina's, formaat 11 x 17 cm.
ISBN 90 201 1639 8

BASIC-computerspellen
Kluwer Software-reeks
M.Th.A.M.Vijftigschild

Dit boek is bedoeld om de computer zijn nut te laten bewijzen als ontspannend element.
De programma's in dit boek variëren van kort tot redelijk uitgebreid en hebben ondanks hun luchtig karakter toch een leerzame achtergrond.
Door de gestructureerde opbouw is men namelijk snel in staat de werking te doorzien en eventuele aanpassingen naar eigen wens aan te brengen.

138 pagina's, formaat 17,5 x 25,5 cm.
ISBN 90 201 1601 0

Het computerboek
Wat iedereen over computers en hun toepassingen moet weten
R.Bradbeer/P.De Bono/P.Laurie

Iedereen heeft tegenwoordig wel op de één of andere manier met comters te maken.
Computers worden ingevoerd op scholen, kantoren, winkels, fabrieken en banken.
Dit boek vertelt in eenvoudig Nederlands alles over zowel de grote computers als de kleinere 'personal computers'. U leest hoe ze werken, wat er mee wordt gedaan en wat je er zelf mee kunt doen. Er zijn voor de geïnteresseerde 'leek' weinig werkelijk begrijpbare boeken op dit gebied. Toch is er grote behoefte aan: in Engeland werden binnen enkele maanden meer dan 50.000 exemplaren van dit boek verkocht!

240 pagina's, formaat 21,5 x 18,5 cm.
ISBN 90 201 1602 9

Dit boek bevat een verzameling serieuze en minder serieuze toepassingen voor een van de meest populaire microcomputers: de ZX 81.
Deze toepassingsprogramma's zijn bedoeld voor de bezitters van een ZX 81 die snel enige resultaten op het beeldscherm willen toveren.
De programma's zijn speciaal geschreven voor de ZX 81 en maken daarom alleen gebruik van de beschikbare BASIC-versie en houden rekening met de beperkte geheugenruimte.
Een aantal programma's is geschikt of kan geschikt gemaakt worden voor de ZX 80.

ISBN 90 201 1604 5

Dit boek bevat een verzameling serieuze en minder serieuze toepassingen voor een van de meest populaire microcomputers: de ZX 81.
Deze toepassingsprogramma's zijn bedoeld voor de bezitters van een ZX 81 die snel enige resultaten op het beeldscherm willen toveren.
De programma's zijn speciaal geschreven voor de ZX 81 en maken daarom alleen gebruik van de beschikbare BASIC-versie en houden rekening met de beperkte geheugenruimte.
Een aantal programma's is geschikt of kan geschikt gemaakt worden voor de ZX 80.

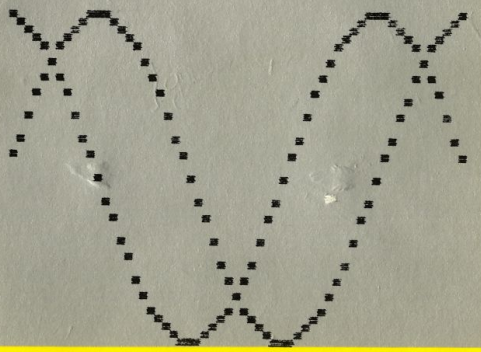
ISBN 90 201 1604 5

MAX VOORBURG TOEPASSINGEN EN SPELLEN VOOR DE ZX 81

Max Voorburg

Toepassingen en spellen voor de ZX81

```
 10 REM ************************
 20 REM *** SINUS/COSINUS  ***
 30 REM ************************
 40 REM
 50 FOR X=1 TO 65
 60 LET Y=X*PI/24
 70 PLOT X,SIN (Y)*21+21
 80 PLOT X,COS (Y)*21+21
 90 NEXT X
100 STOP
110 REM ************************
```

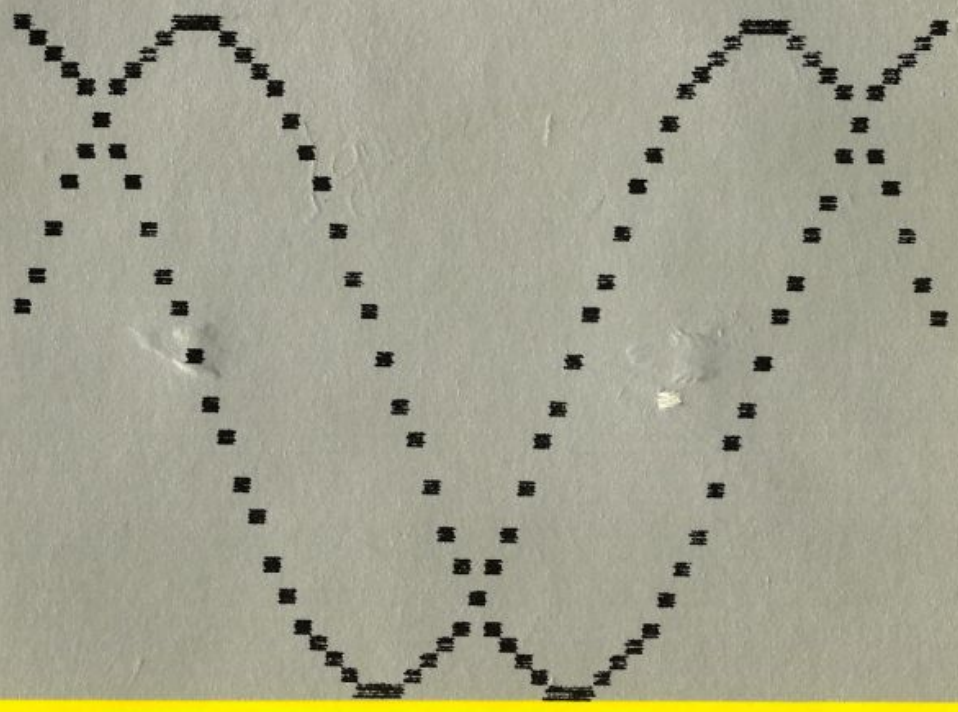


Kluwer Technische Boeken

Max Voorburg

Toepassingen en spellen voor de ZX81

```
 10 REM  ************************
 20 REM  *** SINUS/COSINUS  ***
 30 REM  ************************
 40 REM
 50 FOR X=1 TO 63
 60 LET Y=X*PI/24
 70 PLOT  X,SIN  (Y)*21+21
 80 PLOT  X,COS  (Y)*21+21
 90 NEXT  X
100 STOP
110 REM  ************************
```



Kluwer Technische Boeken