SONY

MSX

Inleiding in MSX-BASIC



HIT BIT

© 1984 by Sony Corporation

MSX is een handelsmerk van Microsoft Corp.

Inleiding in
MSX-BASIC

INHOUDSOPGAVE

Het begin 4
Over dit boek 4
Bobby de slimme hond 5
Je Sony computer 6

Zet je computer aan 't werk 8
Bevelen en invoer 8
Een paar kunstjes die Bobby niet kent 11
Het antwoord op het scherm 15

Cijfers, letters en variabelen 18
Cijfers of letters: geen verschil 19
Een letter wordt een variabele 20

Maak je eerste programma 28
Opzetten van een programma 30
Schrijven van een programma: een-twee-drie klaar .. 32
Korrigeren van een programma: fouten opsporen 34
Verwerken van het programma 36

Grafische voorstelling: stralende sterren 39
Een grafisch programma 39
Willekeurige getallen 44
Geef je programma wat kleur 48

Bewaren van je programma 55
Aansluiten van de cassetterecorder 56
Laat de computer praten 57
Staat het werkelijk op de band? 58
Laden van een programma 59

Beslissingen en spelletjes 61
Raak of mis—de computer beslist 63
Vereenvoudigen van het programma 64

Bewegende voorstellingen 67
Laat zien, wis uit, laat zien, wis uit 70

Alles bij elkaar genomen 74
Toevoegen van kleur en geluid 74
Raadsel + geluid + video 76

De gokautomaat 80
Wat is een lijst-variabele? 81
Maak je eigen gokautomaat 82
Rij-variabelen 88
Onderbreek een lus met INKEY 91
Wat is de stand? 94
De laatste hand leggen 99
Maak het programma beter leesbaar 100
Zet titels in je programma 103
Je bent een heel eind op weg! 105

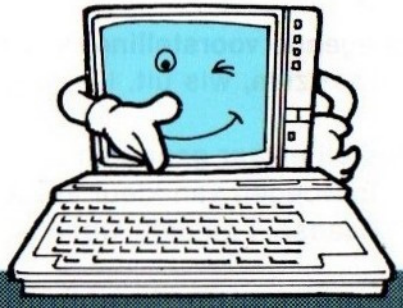
Oefeningen voor BASIC bevelen 107
PRINT 107
INPUT 110
FOR—NEXT 112
IF—THEN en IF—THEN—ELSE 114
DIM (lijst-variabelen) 116

Index 121

Deze handleiding is uitsluitend bedoeld voor gebruik met het Sony MSX systeem.

HET BEGIN

- Over dit boek
- Bobby de slimme hond
- Je Sony computer



OVER DIT BOEK

Dit boek laat je kennismaken met BASIC.

Wat is BASIC? BASIC is een eenvoudige taal om met computers te kunnen "praten". De naam van de taal is BASIC omdat het je een basis geeft, de eenvoudigste manier om je met de computer verstaanbaar te maken.

Maar waarom zouden we met een computer willen praten? Omdat je Sony Computer je met zoveel dingen kan helpen—problemen oplossen, spelletjes doen, wiskunde-sommen maken, gegevens onthouden, en nog veel meer.
Dit boek laat je zien hoe je in niet meer dan een uur of twee al veel plezier met je Sony Computer kan hebben.

Eerst zullen we zien hoe de computer naar je luistert en tegen je praat (in de BASIC taal). Dan gaan we deze speciale taal gebruiken om de computer een paar interessante kunstjes te laten doen—in het begin nog simpele trucs, maar al gauw behoorlijk ingewikkelde dingen. Voor het boek uit is kan je de Sony computer laten werken als een gokautomaat, zo een als je in een flipperhal ziet.

Nadat je je met dit boek vertrouwd hebt gemaakt met BASIC kan je de programma's die in computer-tijdschriften verschijnen zonder moeite lezen. Zo kan je samen met vrienden die ook computers hebben een hoop plezier maken. En als je later nog meer BASIC wil leren, dan zijn er langere en meer gevorderde boeken te krijgen.

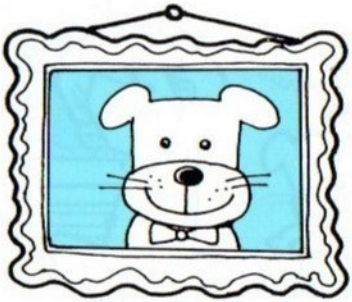
Er zijn een heleboel speciale woorden en bevelen die we gebruiken als we het over computers en computer-programma's hebben. Ze zijn allemaal te vinden in de **Index** achterin dit boek. Als je de betekenis van een woord of bevel wil kontroleren, kan je in de index opzoeken waar je het kan vinden.

En dan is er nog een laatste belangrijk feit dat je moet weten: computers hebben nooit een slecht humeur. Ze zijn simpel en **leuk**, en het maakt ze niets uit hoeveel vergissingen je maakt terwijl je leert ze te gebruiken. Er kan niets "mis gaan" met je Sony computer als je per ongeluk een "verkeerde" toets aanslaat—of welke toets dan ook. Maak je daarover dus geen zorgen. Zoals met alle nieuwe dingen is het heel gewoon dat je vergissingen moet verbeteren en af en toe een paar stappen terug moet om het op de juiste manier over te doen. Je computer heeft echt alle tijd, en alle geduld terwijl jij leert.

In feite is het enige wat echt belangrijk is om te onthouden: dat je met je Sony computer er een vriend bij hebt, die altijd voor je klaar staat om je te laten uitproberen wat je maar wil.

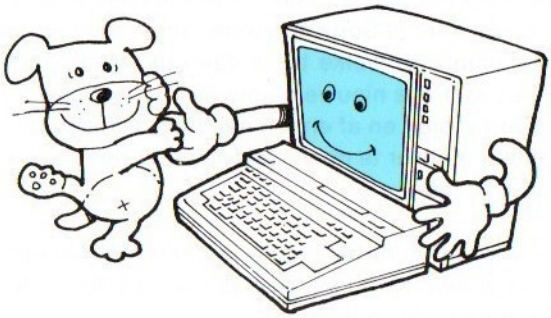
BOBBY DE SLIMME HOND

Terwijl we BASIC leren worden we vergezeld door een vriend.
Dit is ons huisdier, de hond Bobby. Zoals de meeste honden is hij aardig en trouw. Bobby is zomaar een doodgewone hond, maar hij is wel behoorlijk slim. Hij kan een heleboel kunstjes doen als je hem dat vraagt.



Bobby

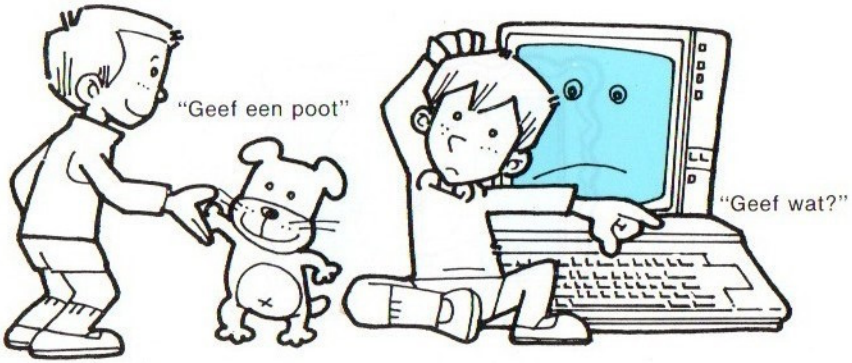
Waarom staat Bobby in een boek over BASIC en computers? Omdat honden knap zijn in het opvolgen van **bevelen**, om kunstjes te doen. En omdat zowel Bobby als je Sony computer bijzonder aardig zijn. Je zal al gauw zien dat je computer ook **bevelen** opvolgt en kunstjes doet, maar niet precies op dezelfde manier. Aan de verschillende manier waarop Bobby en je Sony Computer hun kunstjes doen kan je duidelijk zien hoe de computer luistert, praat en denkt in BASIC.



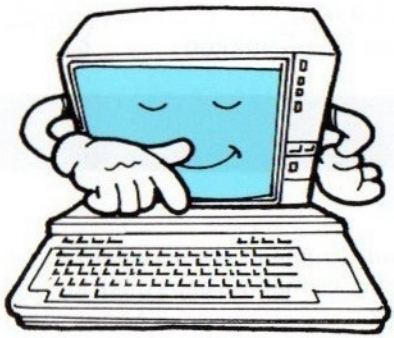
Een aardig stel

JE SONY COMPUTER

Bobby kan een heel stel kunstjes doen. "Zitten", "Doodliggen", "Zoek", "Breng" ... dat is allemaal nogal makkelijk voor Bobby. En je Sony Computer? Probeer maar: "Geef een poot".



Bobby begreep het, en gaf je zijn voorpoot. Maar de computer heeft geen poten. Hij heeft niet eens oren!
Hoe kan je de computer dan vertellen om dit of dat te doen? Het "oor" van de computer—de manier waarop hij je bevelen kan horen—is het **toetsenbord**.



Praat met je vingers

Voordat we beginnen BASIC te leren, en voordat je de volgende bladzijde omslaat, moeten we je vragen om eerst de **Bedieningsaanwijzingen** van je Sony Computer door te lezen om te weten hoe je het toetsenbord gebruikt. Het is niet nodig om alle toetsen uit je hoofd te leren. Wel heel belangrijk is het, om te weten wat de **cursor** is en wat die doet. Probeer maar eens om een paar **letters, cijfers en (grafische) plaatjes** op het scherm te doen verschijnen.

ZET JE COMPUTER AAN 'T WERK

Hoe je ...

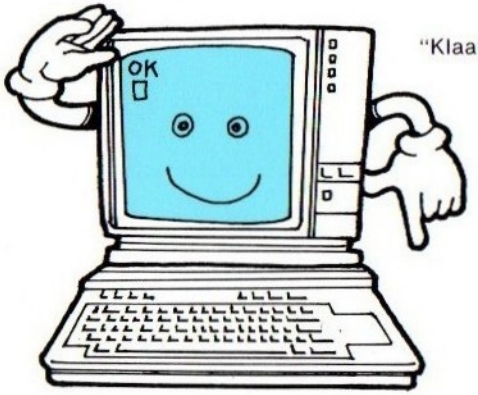
- de computer laat stoppen
- bevelen geeft
- typt in kleur
- de shift toets gebruikt
- de computer sommen laat maken



Nu je de Bedieningsaanwijzingen hebt doorgelezen en weet wat je met het toetsenbord kan doen, kunnen we eens gaan kijken wat je Sony Computer allemaal kan.

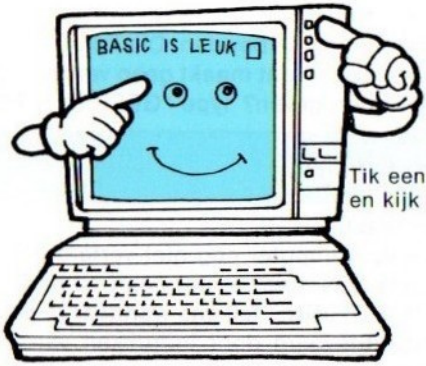
BEVELEN EN INVOER

We hebben eerder gezegd dat het toetsenbord het "oor" is, waarmee de computer onze bevelen kan horen. En hoe weten we dan dat de computer naar ons luistert? Hij laat ons met een opgewekt "Ok" weten dat hij klaar is.



"Klaar wanneer je wilt!"

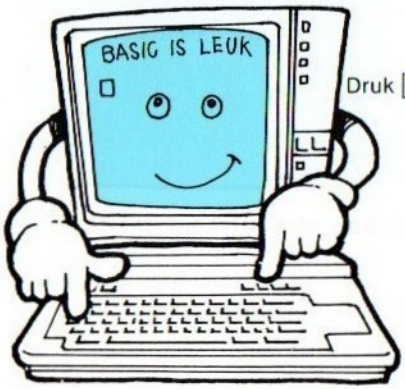
Als je op het toetsenbord een aantal dingen hebt ingetikt, laat het scherm alle letters en cijfers zien die je getypt hebt. De cursor staat ergens middenop het scherm—en het bericht "Ok" is achtergebleven.



Tik een paar woorden in, en kijk hoe de cursor beweegt

Als dat inderdaad zo is, dat het oké van de computer al een tijdje geleden is, dan moeten we de aandacht van de computer trekken. Druk met een vinger de CTRL toets in en met een andere tegelijk de STOP toets. Wanneer CTRL en STOP tegelijk **worden ingedrukt**, stopt de computer met luisteren naar de oude bevelen, en schenkt hij alleen aandacht aan de nieuwe die je geeft.

Bevelen zijn iets anders dan **invoer**. Invoer bestaat uit alle letters en cijfers waarmee je de computer iets vertelt—bijvoorbeeld alle letters die je gebruikt om "BASIC is leuk" op het scherm te schrijven. De invoer zegt de computer niet dat hij iets voor ons moet oplossen. Daar zijn **bevelen** voor nodig, die we met bepaalde toetsen of met bepaalde getypte woorden kunnen geven, om de computer te vertellen wat hij met de invoer moet doen. Denk bijvoorbeeld aan een zakrekenmachientje, waarop je de som 2+2=4 intikt. Daar zijn de "2", de "+" en de andere "2" de invoer, terwijl de "=" het **bevel** aan de calculator is, om 2 en 2 op te tellen en je het antwoord, ofwel de **uitvoer**, te geven: 4. In feite is dat precies hoe de computer werkt.
Nog een belangrijk punt: de letters "Ok" verschijnen **altijd** boven de cursor als je een **bevel** op de juiste manier hebt gegeven; het betekent dat alles wat de computer betreft oké is.

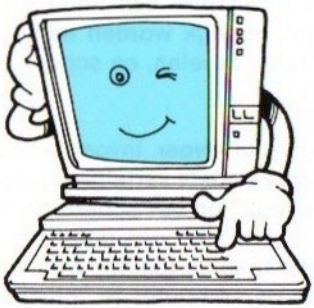


Druk CTRL en STOP in.

Nu is de computer klaar voor je bevelen. Die kan je zowel in hoofdletters als in kleine letters intikken—dat maakt geen verschil. Zullen we de computer eens een poot laten geven? Type "GEEF EEN POOT".

```
GEEF EEN POOT
```

Wat zien we? Geen reaktie.
Dat komt omdat we de computer nog niet verteld hebben om het bevel uit te voeren. De toets waarmee we dat doen zit aan de rechterkant, en draagt het opschrift "RETURN". **Telkens wanneer je de computer een bevel geeft**, wordt dit uitgevoerd door indrukken van de RETURN toets. Die is wat groter dan de andere toetsen, omdat je hem zo vaak gebruikt.

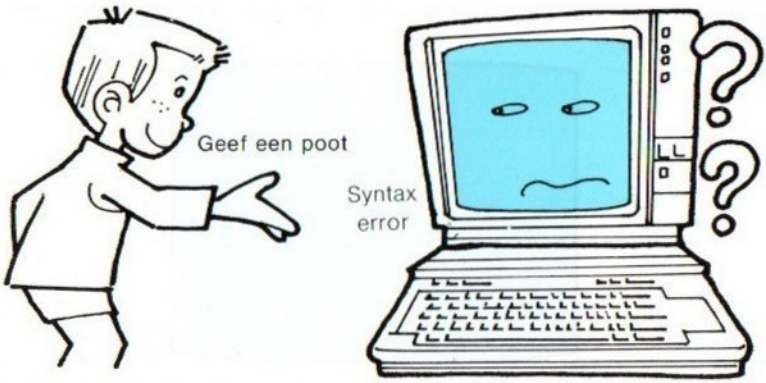


De RETURN toets betekent een bevel.

Druk dus RETURN in, en dan gebeurt er pas iets. De computer geeft echter geen poot, maar hij zegt "piep!" en geeft een antwoord dat er zo uit ziet:



Dat is een **foutmelding**. Het betekent dat de computer je bevel gehoord heeft—maar niet begrepen. En dat is niet zo vreemd, aangezien computers over 't algemeen niet opzitten en pootjes geven. Dit bericht, "syntax error", betekent dat er een taalfout gemaakt is—in de BASIC taal, natuurlijk.



Onze brave hond Bobby kan woorden als "Zitten" en "Poot" en nog een heel stel andere verstaan en begrijpen, maar dat is omdat hij ze van iemand geleerd heeft. Onze computer heeft een ander soort opvoeding genoten, en de bevelen die de computer begrijpt zijn dus heel anders. Die bevelen moeten in de BASIC taal gegeven worden. En dat praten met de computer in BASIC gaat best gemakkelijk, zoals je gauw genoeg zult leren. Laten we nu maar een paar eenvoudige bevelen proberen.

EEN PAAR KUNSTJES DIE BOBBY NIET KENT

Om te beginnen iets gemakkelijks—alhoewel Bobby het daar niet mee eens zou zijn.

```
WIDTH 10
```

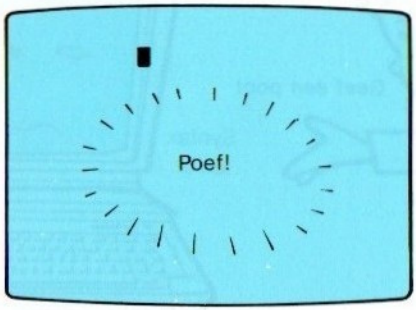
Om dat in de computer in te tikken is niet moeilijk, maar vergeet daarbij niet om de spatiebalk een keer aan te slaan om een **spatie**, wat ruimte, tussen het woord WIDTH en het getal 10 te laten.

W I D T H [ ] 1 0

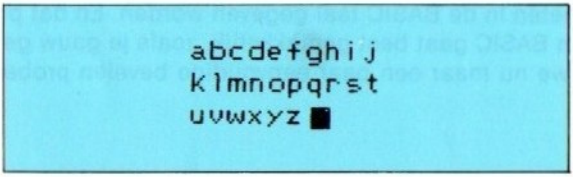
Maak je geen zorgen als je een vergissing maakt. Volg dan gewoon de **Bedieningsaanwijzingen** om de cursor te verplaatsen en vergissingen te korrigeren. De Sony Computer vergeet elke vergissing zodra die verbeterd is. Wanneer het scherm er zo uitziet,

```
WIDTH 10■
```

dan ben je klaar om het bevel te geven. Druk de RETURN toets in—en wat gebeurt er?



De letters zijn verdwenen, en de cursor staat ergens halverwege het scherm. Wat betekent het WIDTH 10 bevel dan? Als we nu iets typen zullen we het al gauw uitvinden. Het is dus weer tijd voor wat **invoer**: we tikken het hele alfabet in, zonder spaties en zonder RETURN.

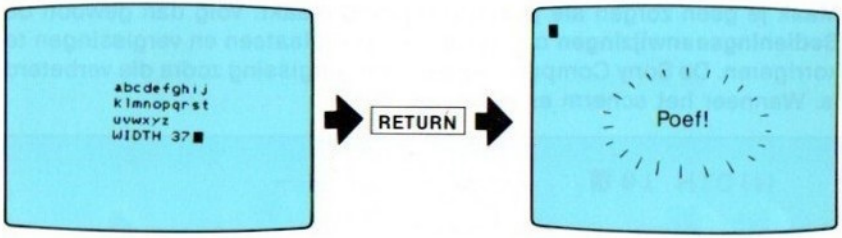


Hoeveel letters staan er op elke regel? Tien. Als er geen andere bevelen volgen, neemt de computer aan dat we elke regel maar tien lettertekens lang, of breed, willen maken. WIDTH 10 is een van de bevelen die de computer kent, en wanneer hij dat bevel hoort maakt hij elke regel op het scherm precies tien lettertekens lang.

Laten we er nog 's een proberen—maar druk eerst de CTRL en de STOP toets tegelijk in, om de laatste invoer te "wissen" en de computer klaar te maken voor het **volgende** bevel:

```
WIDTH 37
```

En druk de RETURN toets in.



Wat nu? Het scherm is opnieuw leeg, met de cursor weer terug in de linkerbovenhoek. Je hebt de computer zojuist opgedragen om alle regels 37 lettertekens lang te maken, en totdat je dat bevel wijzigt zal alles wat je typt verschijnen in regels van 37 lettertekens lang. (Overigens is 37 bij het inschakelen automatisch de lengte van de regels op het scherm, en dat blijft zo totdat je het met een bevel verandert.)

Laten we nog een ander bevel geven dat Bobby niet begrijpt. Druk eerst CTRL en STOP in, en typ dan:

```
COLOR 8
```

(Vergeet niet om daarna RETURN in te drukken.) Plotseling zijn alle letters op het scherm rood! Typ nog maar een paar woorden of letters. Ze zijn allemaal dezelfde kleur.

Wat dacht je van een andere kleur? Zet eerst de cursor weer terug aan de linkerkant van het scherm door de CTRL en STOP toetsen in te drukken. Typ dan COLOR 15 en druk de RETURN toets in—en je tikt weer verder in wit.

Het zal je duidelijk zijn dat 15 voor de computer wit betekent, terwijl 8 rood is—maar zou het niet makkelijker zijn om COLOR RED of COLOR WHITE te typen? Voor mensen, en zelfs voor brave Bobby, zijn woorden gemakkelijker dan getallen. Computers zijn echter knapper met getallen dan met alle andere zaken, en dus gebruiken we getallen voor dingen die de computer moet kunnen begrijpen. BASIC is een taal die vaak in plaats van een woord een getal gebruikt.

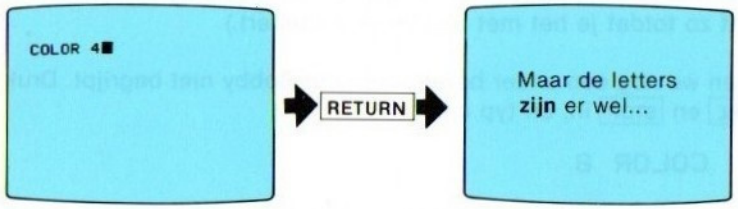
Wanneer je eenmaal gewend bent je computer bevelen te geven zal je vaak getallen gebruiken voor de namen, plaatsen, kleuren of zinnen die de computer moet onthouden. Eigenlijk maakt het niet eens zo veel verschil met het gebruik van nummers voor auto's bijvoorbeeld, of met huisnummers—en voor de computer is het de meest natuurlijke manier van praten. Voordat je het weet zal het je gemakkelijk afgaan om in je BASIC programma's zowel getallen als letters te gebruiken.

We weten nu dat 8 rood betekent en 15 wit. Hoeveel kleuren zijn er in totaal?

code	kleur	code	kleur	code	kleur	code	kleur
0	transparant	4	donkerblauw	8	rood	12	donkergroen
1	zwart	5	lichtblauw	9	lichtrood	13	rose
2	middelgroen	6	donkerrood	10	donkergeel	14	grijs
3	lichtgroen	7	hemelsblauw	11	lichtgeel	15	wit

```
COLOR 4
```

Er zijn in totaal zestien kleuren, en ze zijn best interessant om mee te spelen. Probeer voor de grap dit maar eens:





Donkerblauwe letters op een donkerblauwe achtergrond. Hmm, het ziet er naar uit dat alle sporen zijn uitgewist.

Gebruik de kleurencode om de verschillende kleuren op het scherm uit te proberen. Dan kan je de kleur van je computer net zo instellen als je wilt—er is vast wel een kleur die bij je stemming past!

Zullen we nu eens iets listigs proberen?

```
COLOR 1000
```

(Vergeet de RETURN niet.) Er is geen nummer 1000 in de lijst van kleuren, en de computer weet dat. Een snelle pieptoon laat je weten dat de computer een antwoord heeft. Hij zegt "Illegal function call", waarmee de computer wil zeggen "Mij houd je niet voor de gek".

In feite is het niet mogelijk om de Sony Computer met een bevel voor de gek te houden. Stel je wilt intikken

```
COLOR 6
```

maar je vergist je en typt

```
CILOR 6
```

Als je dan zelf de vergissing niet opmerkt en de [RETURN] toets indrukt, zal de computer je weer een pieptoon laten horen. Deze keer luidt de foutmelding "Syntax error", wat betekent "taalfout".

Natuurlijk maakt iedereen spel- en tikfouten—zelfs de professionals van Sony! Daarom hebben we de [BS] terugsteltoets, de [DEL] wistoets, de [INS] invoegtoets en de toetsen om de cursor te verplaatsen (zie de Bedieningsaanwijzingen). Het verbeteren van vergissingen zal je zo onder de knie hebben.

Hoeveel BASIC woorden zijn er?
Je hebt nu je eerste twee BASIC bevelen geleerd, WIDTH voor breedte en COLOR voor kleur. En je hebt gezien hoe feilloos je Sony Computer reageert op juist gegeven bevelen. Zolang je de spelling goed hebt doet de computer wat hem gezegd wordt, en hij laat een pieptoon horen en geeft uitleg als je een vergissing maakt of een bevel geeft dat niet op de lijst staat. Hoeveel woorden zijn er in totaal in BASIC? Er zijn er ongeveer 100, en daarmee kan je de computer muziek laten maken, je kan hem een tekening laten maken, een wiskundig probleem op laten lossen en wat al niet meer. Alle termen worden verklaard in het MSX-BASIC Programmeer-Handboek. Je hoeft er echter maar een paar te kennen om te kunnen beginnen met programmeren, dus maak je geen zorgen, al lijkt 100 in het begin een heleboel.

Je Sony computer zal je in het voorbijgaan ook een paar dingen leren: hij kent zo'n 35 verschillende **foutmeldingen**, zoals de "Syntax error" en "Illegal function call" die we zojuist gezien hebben. Als je de computer een bevel geeft dat hij niet begrijpt, zal hij je meestal vertellen wat er aan de hand is.

HET ANTWOORD OP HET SCHERM

Bobby heeft zijn favoriete kunstjes, en één daarvan is een poot geven. Je Sony Computer heeft andere interesses. Wiskunde is een van de dingen waar hij verzot op is (en daar is hij dan ook bijzonder goed in).

Laten we eens intikken

```
PRINT 3+5
```

Heb je het plus-teken (+) gevonden? Het zit boven het "=" teken en verschijnt op het scherm wanneer je de [SHIFT] en de is-gelijktoets **tegelijk** indrukt.

Druk vervolgens [RETURN] in om de computer dit bevel te geven.

```
PRINT 3+5
 8
Ok
■
```

Dat was het! Nu heb je de computer gebruikt om een probleem op te lossen en het antwoord op het scherm te geven. Proberen we er nog een:

```
PRINT 100-10
```

Zoals je ziet heb je voor het min-teken de [SHIFT] toets niet nodig.

Druk weer op [RETURN] om het bevel te geven.

```
PRINT 3+5
 8
Ok
PRINT 100-10
 90
Ok
■
```

Natuurlijk geeft de computer het antwoord 90. Dit zijn eenvoudige problemen, maar ook als je triljoenen en biljoenen wilt optellen of aftrekken kan je computer dat in een oogwenk voor je doen.

Hoe zit het met vermenigvuldigen? Geen enkel probleem. We moeten alleen wel onthouden dat de computer daarvoor een **speciaal teken** gebruikt. In plaats van **7×9** zegt hij **7 * 9**.

Typ nu PRINT 7 * 9, en druk op [RETURN].

```
PRINT 7*9
 63
Ok
■
```

Voor delen gebruiken we de [/] toets. In plaats van PRINT 120:40 typen we

```
PRINT 120/40
```

En natuurlijk weet de computer het juiste antwoord wanneer je de RETURN toets indrukt.

```
PRINT 120/40
 3
Ok
■
```

Ben je klaar om wat grotere getallen (7654321×18) te proberen? Ingewikkelde problemen $(2+9) \times (6-8)$?

Ga je gang en probeer maar wat je wilt. Soms kan het gebeuren dat de computer een antwoord geeft dat je niet begrijpt, maar maak je daar geen zorgen over. Ga stap voor stap, denk na over de antwoorden, en binnen een paar minuten doe je de meest formidabele dingen.



Het rekenwonder

CIJFERS, LETTERS EN VARIABELEN

Hoe je ...

- woorden en zinnen op het scherm brengt
- een variabele maakt
  (een symbool met wisselende waarde)
- variabelen in wiskunde gebruikt
- variabelen verschillende namen geeft



Tot op heden hebben we het PRINT bevel gebruikt om de computer wiskundesommen op te laten lossen, en de computer werkt daarbij net zo als een zakrekenmachientje. Nu is het tijd om de computer enkel andere berekeningen te laten maken. Maar eerst moeten we iets meer over PRINT te weten komen, en tegelijk nog wat oefenen in typen, en dus typen we het volgende:

```
PRINT "JAN EN ELLY"
```

De **aanhalingstekens (")** staan boven het ' teken, en om ze in te tikken moet je tegelijk de SHIFT toets indrukken. Wanneer je dan alles getypt hebt, druk je weer op RETURN.

```
PRINT "JAN EN ELLY"
JAN EN ELLY
Ok
■
```

Zoals duidelijk blijkt, "leest" de computer de aanhalingstekens, en vat het PRINT bevel als volgt op: wat zich tussen aanhalingstekens bevindt moet op het scherm komen te staan, maar de aanhalingstekens zelf niet. Hier zijn nog wat voorbeelden:

```
PRINT "ABC TO XYZ"
ABC TO XYZ

PRINT "How are you?"
How are you?
```

Natuurlijk moet je telkens weer de RETURN toets indrukken. Dat zal je wel zo langzamerhand bekend zijn, en dus zullen we je daar van nu af aan niet meer aan herinneren: telkens wanneer je de computer een bevel wil geven, druk je op RETURN.



CIJFERS OF LETTERS: GEEN VERSCHIL

Laten we eens iets anders proberen.

```
PRINT "3+5"
```

Dit lijkt op wat we een paar bladzijden terug deden. Toen zeiden we tegen de computer **PRINT 3+5**, en hij was zo vriendelijk om 8 af te drukken—omdat we geen aanhalingstekens gebruikten.

Met aanhalingstekens krijgen we het volgende:

```
PRINT "3+5"
3+5
Ok
■
```

Het **PRINT** bevel in dit geval—samen met **aanhalingstekens** gebruikt—betekent voor de computer een bevel om te lezen wat er tussen de aanhalingstekens staat, en dat precies zo op het scherm te zetten. Vanwege de aanhalingstekens leest de computer **"3+5"** als drie symbolen om op het scherm te zetten, niet als een rekensom om op te lossen.

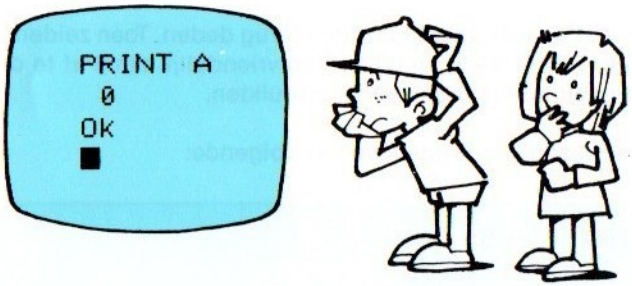


Die worden beide op dezelfde manier gedaan

Dat gebeurt er als een nummer tussen aanhalingstekens staat.

EEN LETTER WORDT EEN VARIABELE

Nu is een nog een interessant punt aan het PRINT bevel. Wat gebeurt er wanneer een letter zonder aanhalingstekens na het PRINT bevel komt?



Wat is hier aan de hand? Geen piep! Geen "syntax error"! In plaats daarvan krijgen we alleen een 0, en dan de letters Ok die betekenen dat de computer ons bevel heeft uitgevoerd, en op het volgende bevel wacht. Maar wat **was** ons bevel dan?

Als er geen foutmelding gegeven is, dan wil dat zeggen dat PRINT A voor de computer een begrijpelijk en juist bevel is. Wat houdt dat bevel in? Om dat te ontdekken moeten we het volgende intikken:

```
LET A=3
```

Na het indrukken van de RETURN toets volgt er geen foutmelding, maar er gebeurt ook niet veel, we zien alleen een vriendelijk Ok. De computer heeft ons bevel opgevolgt en er moet dus van binnen iets gebeurd zijn.



Wat gebeurt er binnenin de computer?

Laten we nu deze twee bevelen eens samenvoegen:

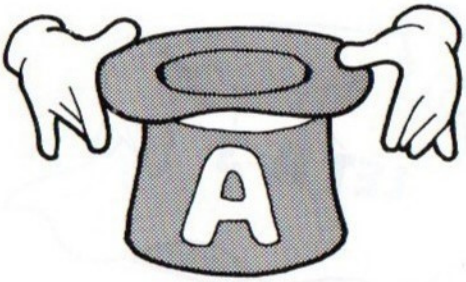
```
LET A=3
Ok
PRINT A
 3
Ok
■
```

Je ziet waarschijnlijk al wat er gebeurt, maar laten we het nog maar eens proberen:

```
LET A=379
Ok
PRINT A
 379
Ok
■
```

A was zojuist nog **3**, maar is nu **379**. als we willen, kunnen we de waarde weer veranderen, gewoon door **LET A=** en een willekeurig nummer in te tikken. A is veranderlijk, en dat is wat variabel betekent. We noemen A een **variabele**.

Een dergelijke **A** is dan ook niet werkelijk een letter van het alfabet meer. In feite is het, na PRINT maar zonder aanhalingstekens, een soort doos of houder waarin we elk getal kunnen stoppen doormiddel van LET.



Een goochelaarshoed is goed met een variabele te vergelijken, want we kunnen alle mogelijke waarden in de hoed stoppen, om ze er weer uit te voorschijn te toveren zodra we ze nodig hebben.



Als we typen LET A=3 dan stoppen we een 3 in de hoed, en we kunnen die weer laten verschijnen door PRINT A te typen. We kunnen met een enkel simpel bevel (LET A=379) de waarde in de hoed veranderen, en zodra we er om vragen, komt dan die nieuwe waarde te voorschijn.

LET is dus een bevel dat onze variabele een bepaalde waarde geeft. In de BASIC taal worden variabelen voor allerlei dingen gebruikt, en daarom zul je in dit boek heel wat variabelen aantreffen. Voor het gemak kunnen we dit zelfde bevel ook wat korter en sneller geven, door het woord LET weg te laten:

```
A=3
```

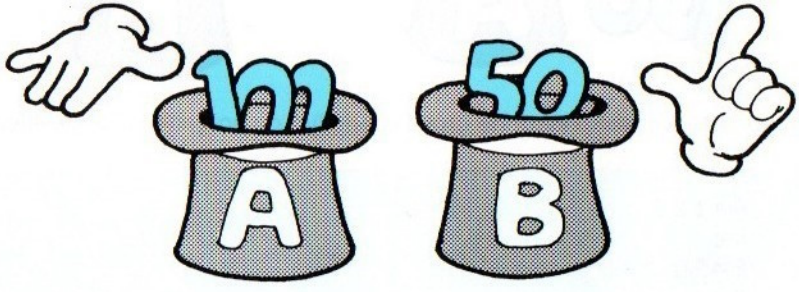
betekent hetzelfde als LET A=3.

Laten we nu eens kijken of je alles onthouden hebt, met een kort overzicht:

```
A=100          Geef A de waarde 100
Ok
PRINT A        Zet A op het scherm
 100           Ziedaar
Ok
PRINT "A"      Geef de letter "A" weer (niet de variabele A)
A              Ziedaar
Ok
B=50           Geef B een waarde
Ok
PRINT B        Zet B op het scherm
 50            Ziedaar
Ok
■
```

Eerder gaven we het bevel PRINT A en de computer antwoordde met 0. Dat betekent dat elke variabele de waarde nul heeft, totdat we hem een andere waarde geven.

Een variabele kan op het scherm door elke willekeurige letter worden vertegenwoordigd. A en B hebben we al gebruikt, en we hebben ze verschillende waarden gegeven.



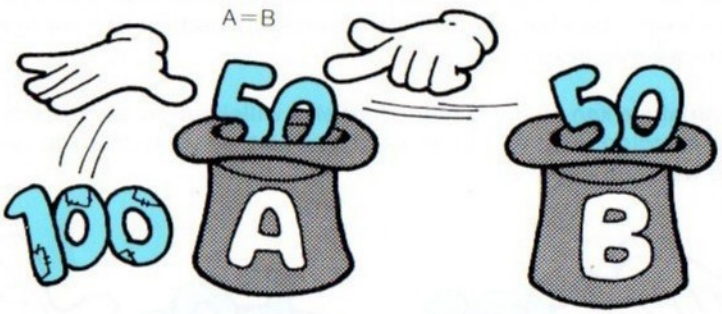
Eens kijken wat we met die variabelen kunnen doen.

We hebben de computer al verteld dat A=100 en B=50. Geef nu de volgende drie bevelen: A=B, PRINT A en PRINT B. Je kan waarschijnlijk wel raden wat er gebeurt, maar doe het maar eens, om te oefenen.

```
A=B
Ok
PRINT A
 50
Ok
PRINT B
 50
Ok
■
```

Kijk eens naar de eerste regel. Toen we A=B intikten, droegen we de computer op om A de waarde van B te geven. Daarvóór hadden we de waarde van A al van 3 in 379 veranderd, en toen verdween de oude waarde in het niets.

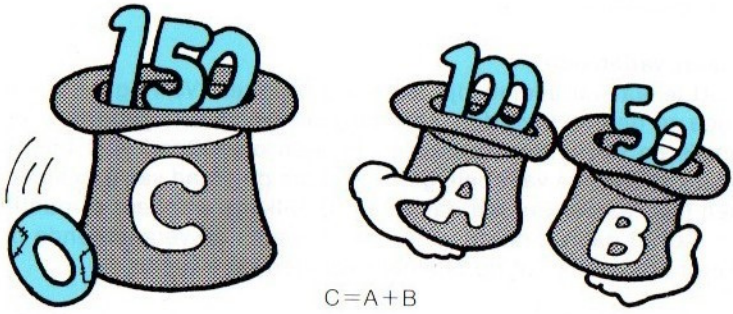
Hier gebeurt hetzelfde, en we veranderen A van 100 in B, ofwel 50.



Nu kunnen we wat wiskunde gaan doen. Typ de volgende vier bevelen:

```
A=100
Ok
B=50
Ok
C=A+B
Ok
PRINT C
 150
Ok
■
```

Hier hebben we drie dozen, A, B en C, en de computer doet een rekensommetje om te vinden hoeveel hij in doos C moet stoppen.

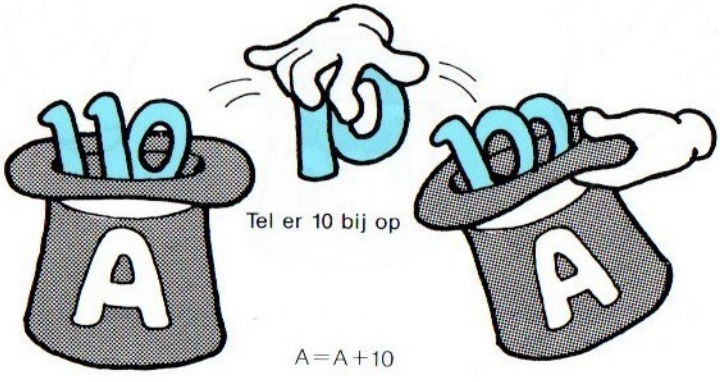


C=A+B

Hier is nog een aardige truc:

```
A=A+10
Ok
PRINT A
 110
Ok
■
```

Is dat een beetje verwarrend? A had een "toegekende waarde" van 100. Maar bij het lezen van dit nieuwe bevel begreep de computer onze aanwijzing A=A+10 als "Geef A een waarde die 10 hoger is dan hij was". En ziedaar, A is nu 110.



A=A+10

Misschien wil je hetzelfde nog eens proberen. En nog eens. Telkens wanneer je het bevel herhaalt, zal het je duidelijk zijn wat het antwoord gaat worden. Je computer **onthoudt altijd** de laatste waarde die aan een variabele is toegekend.

Meer over variabelen

Nu weet je wat variabelen zijn, en hoe ze werken. Vervolgens zullen we zien hoe ze gebruikt worden, en dan gaan we ze zelf gebruiken in programma's en spelletjes. Variabelen hebben ontzettend veel toepassingen. We kunnen een variabele gebruiken om de stand van een spel bij te houden (een aantal punten dat varieert), telkens wanneer er punten bij komen—of af gaan. Andere variabelen bepalen de veranderende plaats van een ruimteschip, of de snelheid van een racewagen.

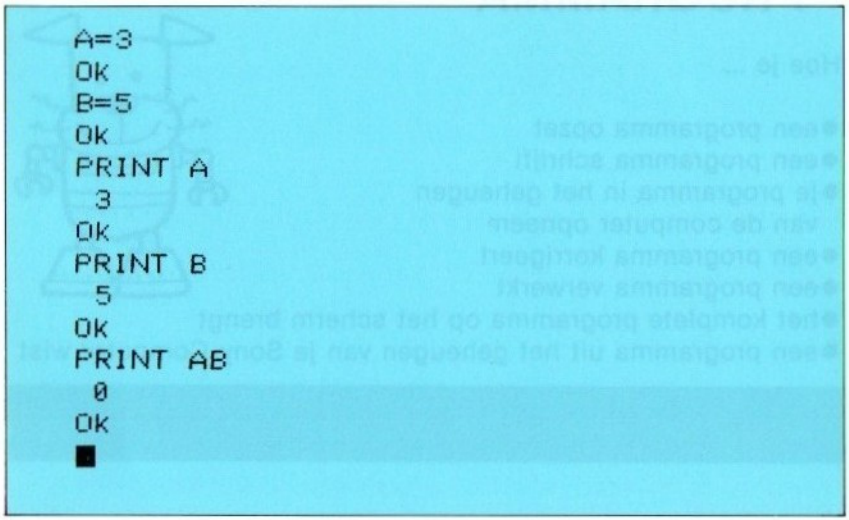
We hebben nu al drie letters—A, B en C—als variabelen gebruikt. Behalve deze drie kunnen we bijna **elke** letter of letterkombinatie gebruiken.

```
UFO=5
Ok
PRINT UFO
 5
Ok
■
```

Zoals je ziet kan een groep van drie letters zonder tussenruimte, in dit geval UFO, een enkel getal vertegenwoordigen.



Bovendien is het belangrijk om te onthouden dat elke letter, of groep letters, zoals A, B of AB, voor een andere waarde staat.

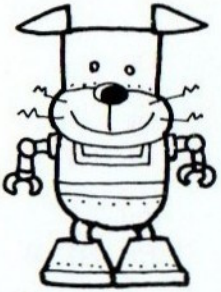


Hier hebben we aan A de waarde 3 toegewezen, terwijl B gelijk is aan 5, maar er is geen waarde aan variabele AB in de computer gebracht, dus de computer geeft AB met de waarde 0 weer (**niet** 35!). Een variabele heeft altijd de waarde 0 tot we hem een andere waarde geven.

De naam die je aan een variabele geeft kan twee, drie, tien—of zefs honderd **lettertekens** lang zijn, en die tekens kunnen zowel **cijfers** zijn als **letters**. Maar **de computer leest altijd alleen maar de eerste twee tekens, en het eerste teken moet altijd een letter zijn**. Dit betekent dat POA, POB, PO1 en PO2 door de computer als een en dezelfde waarde gezien worden. Wees dus wel een beetje voorzichtig met het kiezen van namen voor je variabelen.

Alle woorden die deel uitmaken van de BASIC taal **kunnen niet** als de naam van een variabele gebruikt worden. Zodra je computer een woord van de BASIC taal tegenkomt, vat hij dit op als een **bevel**, en niet als een variabele. Dit betekent natuurlijk dat LET, GOTO en PRINT niet als naam van een variabele gebruikt kunnen worden, maar ook verboden is het gebruik van GOTOA, BLET en alle andere bevelen die we binnenkort zullen leren.

MAAK JE EERSTE PROGRAMMA



Hoe je ...

- een programma opzet
- een programma schrijft
- je programma in het geheugen van de computer opneemt
- een programma korrigeert
- een programma verwerkt
- het komplete programma op het scherm brengt
- een programma uit het geheugen van je Sony Computer wist

Tot op heden hebben we de eerst beginselen van het gebruik van de computer geoefend. Zo hebben we letters en nummers op het scherm gezet, met behulp van een paar symbolen die de computer herkent als woorden van zijn eigen "taal", en we hebben de computer eenvoudige aanwijzingen (bevelen) gegeven. Met andere woorden, we hebben de computer steeds verteld om dingen op het scherm te **doen**, precies op de manier waarop we ze gedaan willen heben. Daarmee hebben we eigenlijk al geleerd hoe mensen met computers communiceren.

Wat de computer gedaan heeft is echter niet veel meer dan wat een goede calculator of zakrekenmachine ook kan. Die kan ook optellen en aftrekken en andere berekeningen uitvoeren, en kan ons ook alle bewerkingen en resultaten op het elektronisch uitleesvenster tonen.

Wat de **computer** voor jou tot een echt opwindend toestel kan maken (en wat het voor geleerden en talloze andere mensen tot een zeer waardevol apparaat maakt) is dat de computer veel meer kan en veel verder gaat. Je Sony Computer kan:

- een reeks **door jou speciaal gegeven aanwijzingen** in zijn **geheugen** opnemen;
- die aanwijzingen opvolgen en zo een groot aantal verschillende opdrachten uitvoeren, spelletjes en trucs doen wanneer je dat wil;
- het resultaat van elke funktie begrijpen, en dat resultaat voor de volgende funktie gebruiken;
- de resultaten van alle eerdere stappen kombineren tijdens het doorlopen van de latere stappen.

Met andere woorden, je computer kan een kompleet **programma** van funkties automatisch uitvoeren, en veel sneller dan welke calculator of rekenmachine dat kan.

Programmeren is het geven van aanwijzingen aan de computer, die hem vertellen welke taken hij moet uitvoeren en in welke volgorde. En daarmee kan je de computer gebruiken om je eigen videospelletjes te kreëren, je eigen videotekeningen en je eigen elektronische muziek te maken. Dat nu is waar we in dit hoofdstuk meer over zullen leren: we gaan ons eerste eenvoudige programma maken.

Een computerprogramma maken is niet moeilijk. Je weet al hoe je de computer een aantal bevelen moet geven, en een programma is niets anders dan een reeks bevelen. Wat belangrijk is in een programma is de **rangschikking:** het **aantal** bevelen en de **volgorde** waarin je ze geeft.

We zullen de uitleg van programma's verduidelijken met behulp van nòg een hond die kunstjes doet. Bobby laten we deze keer met rust, maar we zullen een paar kunstjes proberen met—**Superhond!**

Superhond is een ander soort leuk huisdier: het is een robot! (En waarom niet? Vandaag de dag is toch bijna alles geautomatiseerd?)



Ik ben Superhond, en ik kan programma's uitvoeren!

Bobby kent een heel aantal kunstjes, zoals je weet, maar eigenlijk zijn dat allemaal heel eenvoudige reeksen van handelingen. Als je een bal gooit, gaat hij er achteraan, pakt hem in zijn bek, brengt hem naar je terug, en laat hem los. Hij begrijpt je bevel "geef een poot", tilt zijn voorpoot op, en laat je deze schudden. We houden van Bobby omdat hij best slim is, en om een heleboel andere redenen—maar hij kan in een keer slechts een reeks van niet meer dan drie of vier handelingen onthouden. Om hem aan de gang te houden, moet je hem een volgend kunstje laten doen—een nieuw "programma".

OPZETTEN VAN EEN PROGRAMMA

Superhond is anders. Hij heeft een microprocessor die dingen onthoudt en beslissingen neemt, en dus kan hij in een enkel kunstje veel meer verschillende stappen doorlopen dan Bobby dat kan, net zo als een zakrekenmachientje in de zelfde tijd veel meer wiskundesommen kan oplossen als jij zelf. We kunnen Superhond bijvoorbeeld vertellen:

1. Vind de bal en breng hem terug!
2. Indien de bal wit is, draai dan drie maal rond en blaf!
3. Indien de bal zwart is, blijf dan 10 sekonden lang liggen!

De eerste opdracht is gemakkelijk voor Bobby. Maar de andere twee zijn een stuk ingewikkelder: de hond moet de bal als zwart of wit kunnen herkennen, en precies weten hoeveel malen rond te draaien of hoelang te blijven liggen. En om de kunstjes zonder fouten uit te voeren moet hij op elk moment precies weten hoeveel draaien of hoeveel sekonden er verstreken zijn, en hoeveel keer overblijft voor hij moet blaffen of de bal terugbrengen. Hetgeen inhoudt dat Superhond terwijl hij zijn kunstjes doet voortdurend beslissingen neemt en berekeningen uitvoert.

Er is nog een belangrijk verschil tussen Bobby en Superhond. De hond van de familie is een levend dier, met hersenen die op onze menselijke hersenen lijken, en dus kan hij dingen uit zichzelf doen, zonder dat wij hem dat zeggen. Als Bobby een kat ziet, blaft hij; als hij vuur ziet, rent hij weg. Maar Superhond is een machine. Hij kan heel ingewikkelde kunstjes doen, maar hij kan er niet over "nadenken" of zich herinneren wat hij moest doen. Superhond doet **alleen** die dingen waarvoor hij **geprogrammeerd** is, en hij zal nooit iets doen zonder dat een mens hem vertelt zijn onderdelen—motoren, sensors, komponenten en de microprocessor—in een bepaalde volgorde te gebruiken.

Dus als je het programma schrijft voor de truc van Superhond, dan moet je eerst nadenken over alles wat je de machine wilt laten doen—**elk** klein stapje, en waar dit in de totale volgorde past. Ongeveer als volgt:

1. Ontvang en identificeer het bevel "Vind de bal en breng hem terug!"
2. Gebruik beeldsensors om de bewegende bal te volgen en de plaats te bepalen.
3. Zet mechanische poten in beweging om je naar de bal toe te verplaatsen.
4. Gebruik beeldsensors om de bal als "zwart" of "wit" te herkennen.
5. Gebruik lichaamsdelen om de bal op te pakken en terug te brengen.
6. Beslis of je moet "ronddraaien" of "blijven liggen".
7. Gebruik de nodige lichaamsdelen om de juiste handeling uit te voeren.

8. Bepaal het aantal draaien of het aantal sekonden dat tijdens het uitvoeren verstreken is.
9. Blaf/Blaf niet.
10. Staak alle beweging als het kunstje voltooid is.
11. Houd je gereed voor het volgend programma bevel.

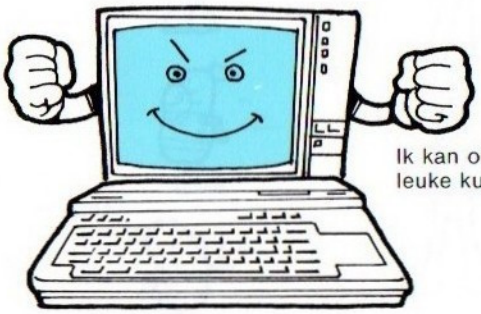
Al deze stappen moet je nalopen om er zeker van te zijn dat je niets vergeten hebt en geen vergissingen gemaakt hebt, en dan moet je ze in de microprocessor van Superhond invoeren. Als je aan alles gedacht hebt—en alles in de juiste volgorde geplaatst hebt—dan zal Superhond elke keer dat hij het kunstje doet, het ook perfekt uitvoeren. Je kan dan het "programma verwerken" zo vaak je maar wilt, één keer, tien keer of tienduizend keer, en hij zal altijd precies hetzelfde doen.



Er zijn dus twee regels voor het schrijven van een programma: je moet **overal** aan denken, alles wat de computer moet doen om het gewenste resultaat te bereiken; en je moet alle stappen in de juiste **volgorde** zetten. De enige "vaardigheid" die je hiervoor nodig hebt is het normale logisch denken van alledag, want computers, zoals alle machines, werken net zoals je brein werkt—logisch.

Laten we nu dan maar het eerste programma voor je Sony Computer schrijven, dan kunnen we zien hoe simpel het eigenlijk is.

SCHRIJVEN VAN EEN PROGRAMMA: EEN-TWEE-DRIE KLAAR



Ik kan ook een heel stel leuke kunstjes doen!

"Volgorde" in een computerprogramma betekent alleen maar dat alle bevelen worden opgeschreven in een volgorde van een-twee-drie. Weet je nog, het kunstje van Superhond: het had drie afzonderlijke gedeelten, en voor elk deel schreven we een nummer.

1. Vind de bal en breng hem terug!
2. Indien de bal wit is, draai dan drie maal rond en blaf!
3. Indien de bal zwart is, blijf dan 10 sekonden lang liggen!

Deze stappen zijn in logische volgorde geschreven: we willen dat hij eerst de bal terugbrengt voor hij de andere dingen doet, dus bevel 1 komt op de eerste plaats. (Bevel 2 en 3 kunnen echter onderling van plaats wisselen zonder dat het kunstje verandert.)

We zullen onze bevelen op de zelfde manier in de computer invoeren: we schrijven ze op het scherm in de zelfde volgorde als waarin ze uitgevoerd moeten worden, en we geven elk bevel een nummer. Onze eerste "taak" voor de computer—ons programma—is het op het scherm brengen van de woorden "JAN EN ELLY".

```
10 PRINT "JAN"
20 PRINT "EN"
30 PRINT "ELLY"
```

Je kent het PRINT bevel al, en hier gebruiken we het drie maal—een maal voor elk van de drie woorden die op het scherm moeten komen. Natuurlijk is de volgorde van de bevelen gelijk aan de woordvolgorde. Het enige verschil met de manier waarop we het PRINT bevel eerder gebruikt hebben is dat ze deze keer elk met een nummer beginnen: 10, 20 en 30. Dit is heel belangrijk: deze regelnummers vertellen de computer **welk bevel het volgende is** tijdens het afwerken van het programma. Met andere woorden, ze geven de werkvolgorde voor de computer aan.

Typ nu het eerste bevel: 10 PRINT "JAN". Vergeet niet aan het eind een maal op RETURN te drukken.

1 0 [ ] P R I N T [ ] " J A N " RETURN

Wat komt er op het scherm te staan?

```
10 PRINT "JAN"
■
```

Prima. Je eerste bevel is ingevoerd, de cursor is naar de volgende regel versprongen, en je bent klaar om de volgende twee bevelen in te voeren, op precies dezelfde manier:

```
10 PRINT "JAN"
20 PRINT "EN"
30 PRINT "ELLY"
■
```

Daar is je programma—en het is al bijna klaar voor gebruik.

KORRIGEREN VAN EEN PROGRAMMA: FOUTEN OPSPOREN

Voor we het programma kunnen verwerken, moeten we een belangrijke laatste stap niet vergeten: we moeten **kontroleren** of alle onderdelen van het programma op de juiste wijze zijn ingevoerd. Bedenk dat je Sony Computer maar een machine is, en dat hij alleen bevelen kan lezen die in de BASIC taal zijn geschreven. Als je een vergissing gemaakt hebt—al is het maar een letter of een spatie op de verkeerde plaats—dan ziet het bevel er voor de computer niet als een BASIC woord uit (of misschien als het verkeerde woord). Dan stopt het gehele programma, of doet de computer iets verkeerd. Bijvoorbeeld:

```
10 PRIMT "JAN"
```

Vergissing

Dergelijke vergissingen zijn **fouten** in het programma. Het PRINT bevel moet goed gespeld worden—anders begrijpt de computer niet eens dat het een bevel is.

Als je op dit moment een vergissing in het programma ontdekt, dan kan je daar blij om zijn: hoe eerder je een fout op het spoor komt, des te beter, en korrigeren is heel eenvoudig.

Je weet al hoe je de cursor moet verplaatsen met de vier toetsen met pijlen. De cursor staat nu aan het eind van het programma, dus eerst brengen we hem met de juiste toets omhoog naar de "10". Dan verplaatsen we hem zes stapjes naar rechts met de volgende toets, zodat hij op de onjuiste letter M komt te staan.

```
10 PRIMT "JAN"
```

Zet de cursor hier

Om de korrektie uit te voeren hoef je alleen maar de juiste letter te typen.

```
10 PRINT "JAN"
```

Druk op [N].

Druk dan een maal op [RETURN] ... en de fout is weg!

```
10 PRINT "JAN"
20 PRINT "EN"
```

De cursor komt na het indrukken van [RETURN] hier te staan.

Als er verder nog fouten in het programma zitten, kun je ze verbeteren door de cursor op dezelfde manier te verplaatsen. (Als dat nodig mocht zijn, kan je in de **Bedieningsaanwijzingen** opzoeken hoe je een vergeten letter of spatie invoegt, of een overbodige letter of spatie weghaalt.) Als alle fouten gekorrigeerd zijn, breng je de cursor met de pijl-toetsen weer naar de onderste regel—zodat het scherm er dan weer zo uitziet:

```
10 PRINT "JAN"
20 PRINT "EN"
30 PRINT "ELLY"
■
```

Misschien heb je in dit eenvoudige programma wel helemaal geen fouten gemaakt. Toch is het altijd goed om alles grondig te **kontroleren**. Je programma's zullen al gauw uit meer bevelen bestaan, en die zullen stuk voor stuk ook langer zijn. Elke fout, of die nu klein of groot is, kan een programma onderbreken, en iedereen—ook de experts van Sony—maakt wel eens een spelfout.

Van een ding kan je op aan: als er een vergissing gemaakt is, zal je Sony Computer het je vroeger of later vertellen. Dan verschijnt er een **foutmelding** op het scherm, en stopt het programma eerder dan de bedoeling was, of je computer doet iets anders dan je wilde. Dan zul je de hele lijst van bevelen moeten doorlopen om de fout te vinden en te verbeteren. Het is dus beter om al in het begin, wanneer je de bevelen schrijft, de fouten in je programma op te sporen.

Regels voor het programmeren

- Bedenk zorgvuldig **alle stappen die in je programma noodzakelijk zijn** om je computer te laten doen wat je wil.
- Zorg dat je alle bevelen achtereenvolgens schrijft—in de **logische volgorde** die nodig is voor een juist resultaat.
- Gebruik altijd het **regelnummer** aan het **begin** van elk bevel.
- **Elk bevel moet na invoeren gekontroleerd worden** op fouten—en elke gevonden fout moet gekorrigeerd worden.

VERWERKEN VAN HET PROGRAMMA

Nu je alle bevelen hebt ingevoerd en de korrekties hebt gemaakt ben je klaar om het programma te **verwerken, of "runnen"**. Het bevel dat we daarvoor gebruiken:

```
RUN
```

En nadat je dit getypt **en** RETURN gedrukt hebt, "rent" de computer als het ware door je programma, en komt het scherm er als volgt uit te zien:

```
10 PRINT "JAN"
20 PRINT "EN"
30 PRINT "ELLY"
RUN
JAN
EN
ELLY
Ok
■
```

Je Sony Computer had maar een paar sekonden nodig om de bevelen te "lezen" en op het scherm te zetten:

```
JAN
EN
ELLY
```

Typ nogmaals het RUN bevel, vlak achter de cursor, en druk weer op RETURN. Het resultaat is hetzelfde, nietwaar? Het resultaat van **dit** programma zal altijd hetzelfde zijn, totdat je de bevel wijzigt of uitwist. Probeer het nog een paar maal en je zult het zelf zien. Elke keer zal er JAN EN ELLY op het scherm verschijnen. Dat betekent dat je Sony Computer het programma **onthoudt**.

Je kan het geheugen van de computer kontroleren wanneer je maar wil door **LIST** te typen (en dan RETURN in te drukken). We zullen het LIST bevel elke keer dat we een programma schrijven gebruiken.



Een van de bijzonder nuttige eigenschappen van je Sony Computer is dat hij je bevelen onthoudt. Alles wat je daarvoor moet doen is de computer laten weten dat hij het moet onthouden, door **een nummer voor het bevel** te zetten.

```
10 PRINT "JAN"
```

Nadat een regelnummer is ingetikt is het een programma.

Dit bevel heeft een nummer, dus de computer weet dat het deel van een programma uitmaakt, en hij zal het bevel in zijn geheugen opslaan. 10 is het **regelnummer**, en een regelnummer kan lopen van 0 tot 65529.

Hoe lang zal je computer de bevelen van je programma onthouden? Voor altijd!—of tot je hem zegt het te vergeten. Het bevel om iets te vergeten is

```
NEW
```

Om er zeker van te zijn dat de computer iets onthoudt, voeren we een paar genummerde bevelen in, en dan LIST. Heeft de computer de bevelen netjes op een rijtje gezet? Goed. Dan typen we nu NEW.

```
LIST
10 PRINT "JAN"
20 PRINT "EN"
30 PRINT "ELLY"
Ok
NEW
Ok
■
```

Je bevelen staan nog steeds op het scherm, maar de computer is ze al vergeten. Probeer nog maar eens LIST. Het programma is uit het geheugen gewist.

Er zijn twee andere manieren om de computer bevelen te laten vergeten. Een manier is het indrukken van de RESET toets, en de andere is uitschakelen van de computer. **Doe deze beide dingen dus niet** tenzij je zeker weet dat je de computer alles wilt laten vergeten.



Later zullen we leren hoe we programma's kunnen **opslaan**, zodat we ze weken of zelfs jaren later weer kunnen gebruiken, met behulp van een cassetterecorder. Tot die tijd zal je computer alles vergeten zodra je hem uitschakelt.

GRAFISCHE VOORSTELLING: STRALENDE STERREN

Hoe je ...



- meer BASIC woorden gebruikt
- het scherm uitwist
- punten en strepen zet
- de computer een handeling laat herhalen
- een variabele gebruikt
- een willekeurig getal gebruikt
- met COLOR meer kleurrijke beelden maakt

EEN GRAFISCH PROGRAMMA

Zullen we een paar nieuwe BASIC woorden gebruiken, en kijken wat er gebeurt? Typ de volgende bevelen:

```
10 SCREEN 2
20 PSET (100,100)
30 PSET (150,100)
40 PSET (100,150)
50 PSET (150,150)
60 GOTO 20
```

Dit is een **programma**—een aantal stappen die de computer gebruikt om een bepaalde taak te verrichten. Nu het programma zich in het geheugen van de computer bevindt kun je het RUN bevel geven om te zien wat dit programma doet.

Alle letters zijn verdwenen, en er zijn vier witte stippen op het scherm verschenen. De cursor is ook verdwenen, en als je een willekeurige toets indrukt gebeurt er op het scherm helemaal niets, zie je wel? De computer is bezig het programma te verwerken. Waarom duurt dat zo lang?

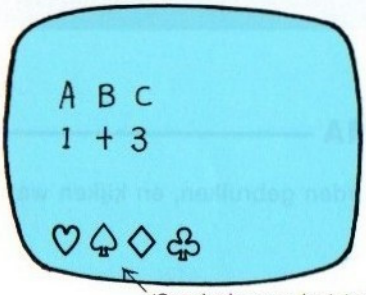
Laten we eens zien wat dit programma betekent. Druk eerst CTRL en STOP in, om het programma af te zetten. De stippen verdwijnen dan, en de cursor is weer terug. Typ nu LIST, en dan kunnen we het programma regel voor regel gaan bekijken. Beginnen we met regel 10:

SCREEN 2

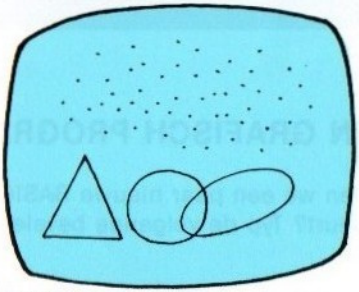
betekent "scherm 2" en is het bevel dat het mogelijk maakt om **grafische voorstellingen** (plaatjes) op het scherm te brengen. Grafische voorstellingen bestaan uit stippen, lijnen, cirkels en andere figuren die op het scherm verschijnen. (Alle dingen die op het scherm worden weergegeven maar **niet** tot de grafische voorstellingen behoren—letters, cijfers en symbolen—worden **tekens** of lettertekens genoemd.)

SCREEN bevelen vertellen de computer in welke vorm informatie op het scherm moet worden weergegeven. Er zijn twee soorten SCREEN bevelen.

SCREEN 0 of SCREEN 1 is voor **tekens**



SCREEN 2 is voor **grafische voorstellingen**



(Symbolen op het toetsenbord zijn tekens)

SCREEN 2 laat de computer dus grafische voorstellingen maken. Toen je het RUN bevel gaf, verdwenen de tekens, de letters en cijfers, van het scherm. Dit was omdat SCREEN 2 de computer de opdracht geeft om grafische voorstellingen af te beelden. Bij het indrukken van CTRL en STOP werd het "grafisch" bevel opgeheven, en kwamen de tekens terug. Dit was de werking van de eerste regel van het programma; als je dit begrepen hebt, kunnen we verder gaan.

```
20 PSET (100,100)
30 PSET (150,100)
40 PSET (100,150)
50 PSET (150,150)
```

De verklaring: PSET is het bevel om **stippen** op het scherm te zetten, en de getallen tussen de haakjes geven voor de computer de plaats van de stippen aan.

Herhaal, herhaal, herhaal, herhaal

De laatste regel van het programma bevat een heel belangrijk bevel:

```
60 GOTO 20
```

GOTO betekent "ga naar", en dit bevel zegt de computer hier dus om **terug te gaan** naar regel 20, om vandaar opnieuw te beginnen. Wanneer de computer regel 20 bereikt,

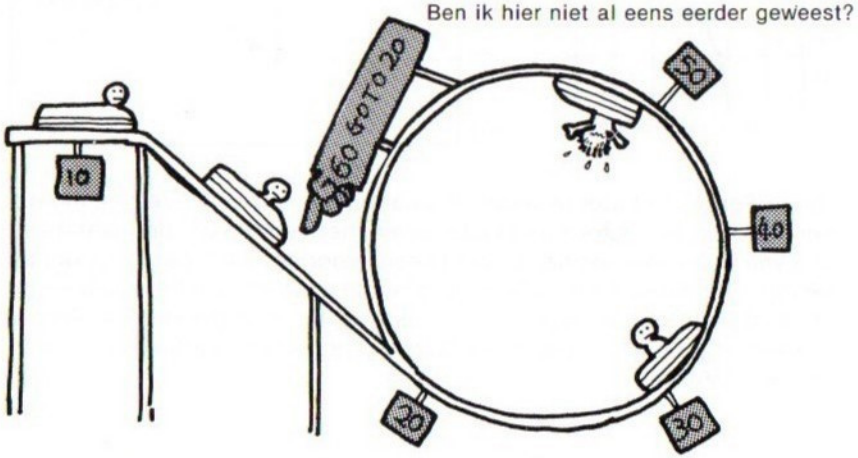
```
20 PSET (100,100)
```

herhaalt de computer natuurlijk het programma. Bij regel 60 aangekomen, gaan we weer terug naar regel 20, herhalen het programma, gaan terug naar 20, herhalen het programma...

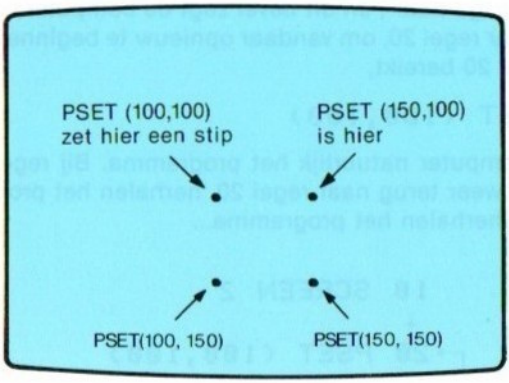
```
10 SCREEN 2
 ↓
20 PSET (100,100)
 ↓
30 PSET (150,100)
 ↓
40 PSET (100,150)
 ↓
50 PSET (150,150)
 ↓
60 GOTO 20
```

Dit gedeelte wordt eindeloos herhaald

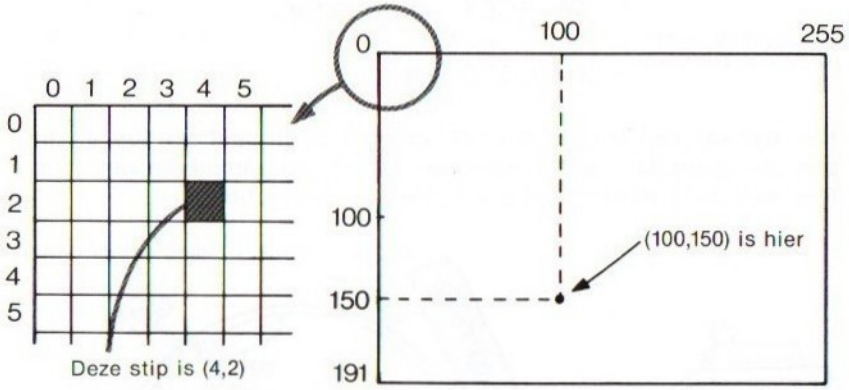
Het deel van een programma dat herhaald wordt heet een **lus**, en dit is een veel gebruikt, nuttig hulpmiddel bij het programmeren van computers. (Gelukkig worden niet alle lussen eindeloos herhaald.)



Dan gaan we nu zien hoe die PSET bevelen de stippen op het scherm brengen.



Zie je hoe de nummers werken? Voor allebei de stippen aan de linkerkant is het eerste getal 100, en voor beide stippen aan de rechterkant is het 150. Hier is een deel van het scherm vergroot weergegeven.



Om je Sony Computer te vertellen waar hij een stip moet zetten, moet je hem twee getallen voor de plaats geven, het eerste voor de plaatsbepaling van links naar rechts, en het tweede voor de plaatsbepaling van boven naar beneden. Er zijn 256 verschillende posities van links naar rechts (0 t/m 255) en 192 van boven naar beneden (0 t/m 191). We kunnen hiermee dus een stip op elke gewenste plaats op het scherm zetten, van (0,0) tot en met (255,191).

Grafische voorstellingen en variabelen
Je zult je nog herinneren dat elk getal door een variabele vervangen kan worden. Welnu, dat geldt ook voor PSET getallen. Als we het links-rechts getal omzetten in variabele X, en we maken het boven-beneden getal variabele Y, dan kunnen we ons bevel schrijven als

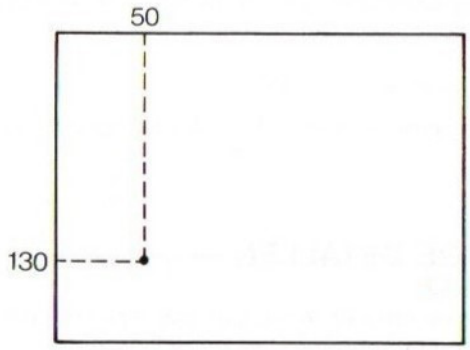
```
PSET (X,Y)
```

en dan kunnen we elk gewenst getal invullen voor X en Y, om de stippen overal te plaatsen waar we maar willen.

Kijk, zo gebruik je de variabelen voor een enkele stip. Geef eerst het NEW bevel om je Sony Computer het laatste bevel te doen vergeten, en voer dan het nieuwe programma in:

```
10 SCREEN 2
20 X=50:Y=130
30 PSET (X,Y)
40 GOTO 30
```

Leg het programma in het geheugen vast door op RETURN te drukken, en geef daarna het RUN bevel. Je computer begint dan met het bevel van regel 10, dat hem opdraagt het scherm vrij te maken voor **grafische voorstellingen**. In regel 20 maakt de computer kennis met twee variabelen, en in regel 30 gebruikt hij deze twee om een stip op het scherm te zetten.



Welnu, X is 50 en Y is 130, dus er komt een witte stip op het scherm in positie (50,130). Vervolgens leest de computer regel 40—die hem terugstuurt naar regel 30. Dat betekent dat de enige manier om dit programma te onderbreken is om CTRL en STOP in te drukken.

Waarom gebruikten we variabelen in plaats van gewoon PSET (50,130) te schrijven? En waarom lieten we de stip eindeloos herhalen met regel 40? Deze stappen zijn niet nodig als we slechts heel even een enkele stip op het scherm willen zetten. Maar dit is wel de juiste manier om met een kort programma een groot aantal stippen te doen verschijnen. En dat wordt onze volgende stap, na de onderstaande opmerkingen over leestekens en spelling.

Opgelet: Wees voorzichtig met leestekens

Het programma dat we nu gebruiken heeft drie verschillende soorten **leestekens**—een , (komma) en een : (dubbele punt) en ( ) ofwel haakjes. Deze leestekens zijn gelijk aan **lettertekens**, en ze moeten daarom net als de letters en cijfers in de bevelen **op de juiste plaats gezet worden**. Onthoud goed dat je Sony Computer een bevel waarin een spelfout is gemaakt niet begrijpt, en dat leestekens deel uitmaken van de spelling.

Elk leesteken heeft zijn eigen speciale betekenis in BASIC. De komma geeft aan waar een getal eindigt is en het volgende begint (het is dus nooit een decimale komma). De dubbele punt geeft het einde van een bevel aan als op dezelfde regel nog een bevel volgt. Haakjes zijn nodig wanneer twee of meer nummers, of twee of meer variabelen samen gebruikt worden—en haakjes **komen altijd per paar**, eerst het linker "(" en later het rechter ")". Verderop in dit boek leer je nog een aantal andere leestekens zoals ze in BASIC bevelen gebruikt worden.

Zoals we al eerder zeiden maakt iedereen wel eens een spelfout. Als dit gebeurt zal je Sony Computer gewoonlijk het bevel niet begrijpen en dus een foutmelding doorgeven. Bijvoorbeeld,

```
Syntax error in 30
```

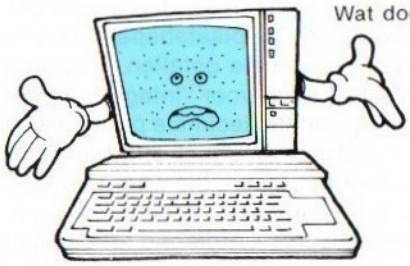
betekent dat er in regel 30 waarschijnlijk een spelfout geslopen is.

WILLEKEURIGE GETALLEN

Geef het NEW bevel om het geheugen van de computer te wissen, en voer het volgende programma in het geheugen in:

```
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y)
50 GOTO 20
```

Als je dit programma met RUN verwerkt, zal het scherm zich langzamerhand vullen met meer en meer stippen, tot je het onderbreekt... inderdaad, met CTRL en STOP.



Wat doe je met mijn gezicht?!

Nu moeten we de twee nieuwe regels in het programma verklaren.

```
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
```

Wat we direkt zien is dat deze twee regels bepaalde waarden toekennen aan de variabelen X en Y. Maar wat zijn die waarden? Kijk eerst eens naar RND(1). RND is de afkorting van "random", wat **willekeurig** betekent. Het is een van de **funkties** in BASIC. Probeer dit bevel maar eens, om een beter begrip te krijgen van de **funktie voor willekeurige getallen**.

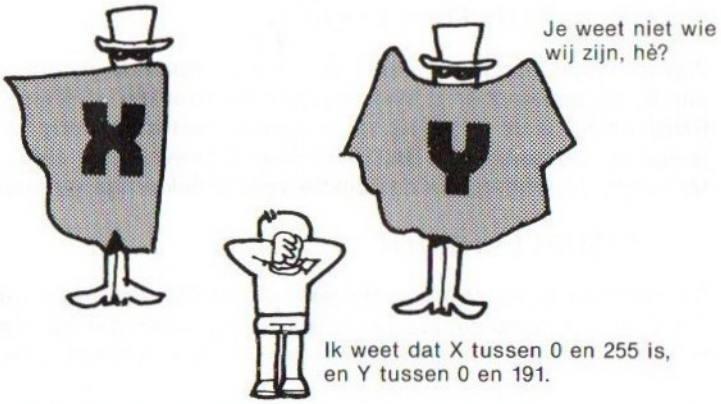
```
X=RND(1):PRINT X
```

Tik dit bevel in en druk op RETURN: je ziet dan een gebroken getal van 14 cijfers, een getal dat kleiner is dan 1 maar groter dan nul, en dat begint met een **punt voor de decimalen** (in plaats van een komma). Bijvoorbeeld:

```
X=RND(1):PRINT X
 .59521943994623
Ok
■
```

Probeer het bevel nog eens. Veranderde het getal? Doe het nog maar eens. Je computer heeft een lange lijst willekeurige getallen, en hij kiest ze een voor een, om ze voor X in te vullen.

Om terug te gaan naar regel 20 in het programma, we hebben daar (RND(1) * 256), wat betekent dat het willekeurig getal wordt vermenigvuldigd met 256. Maar als we een van die gebroken getallen vermenigvuldigen met 256 zal het antwoord bijna altijd ook een breuk bevatten—terwijl de posities van onze stippen moeten worden uitgedrukt in **gehele** getallen, zoals 1 of 30 of 192, omdat de plaatsbepaling op het scherm alleen plaatsvindt aan de hand van gehele getallen. Daarom begint het bevel met **INT**. Deze afkorting INT staat voor "integer", wat **geheel getal** betekent. Gebruik van INT zorgt dat de decimale punt en alle cijfers daar achter afgekapt worden, zodat het resultaat altijd een geheel getal is. Nu kunnen we de waarde van X begrijpen. Die waarde verandert elke keer dat het programma de lus herhaalt. Telkens wanneer je computer X=INT(RND(1) * 256) leest, wordt X afgerond tot een geheel getal tussen 0 en 255—maar je weet nooit welk nummer het wordt. En telkens wanneer de computer regel 40 leest, vult hij ook voor Y een geheel getal in, tussen 0 en 191, op precies dezelfde manier.



De volgende regel kennen we al,

```
40 PSET (X,Y)
```

ook al weten we niet wat de waarden van X en Y precies zijn. De computer vertelt ons de waarden niet, want er is geen PRINT bevel gegeven. In plaats daarven blijft hij meer en meer stippen ergens op het scherm zetten, op alle plaatsen die hem worden aangegeven met de gehele getallen voor links-rechts en de boven-beneden.

We hebben echter alleen nog maar verklaard hoe een enkele stip in feite op het scherm komt. Waar komen al die andere dan vandaan? Daarvoor hebben we de lus. De volgende regel,

```
50 GOTO 20
```

laat het programma terugkeren naar

```
20 X=INT(RND(1)*256)
```

Elke keer worden dan nieuwe waarden aan X en Y toegekend, zodat

```
40 PSET (X,Y)
```

een stip op een nieuwe plaats zet, met als gevolg dat we stip na stip na stip zien verschijnen, over het gehele scherm.

Programma's: twee soorten bevelen

Een programma is, zoals we al zeiden, **een reeks bevelen in een vaste, genummerde volgorde**, vastgelegd in het geheugen van de computer. Een programma kan honderden of duizenden regels (bevelen) hebben, maar dat is in de meeste gevallen niet nodig; we hebben zojuist al duizenden stippen gemaakt met een programma van slechts vijf regels.

Een van die vijf regels—het bevel GOTO 20—liet de computer op zich geen aparte handeling verrichten, maar vertelde de computer **waar die naar toe moest gaan**. Dit was de regel die ons korte programma zo veel stippen liet zetten—met behulp van een eindeloos herhaalde lus.

We hebben nu twee verschillende soorten bevelen gezien:

1. **Bevelen die de computer vertellen wat die moet doen, zoals:**

   PRINT PSET WIDTH COLOR enz.

en

2. **Bevelen die de volgorde van het programma veranderen, zoals:**

   GOTO IF—THEN FOR—NEXT enz.

Er zijn maar een paar bevelen van de tweede soort, en we zullen daar verderop in dit boek meer over leren. Het zijn waarschijnlijk de meest interessante hulpmiddelen bij het programmeren: net als GOTO, het bevel dat we gebruikten om een lus te maken, worden ze alle gebruikt om een **klein** programma **uiterst krachtig** te maken.

GEEF JE PROGRAMMA WAT KLEUR

We gaan het programma met de stippen wat interessanter maken—met kleur! Laten we om te beginnen het programma, dat zich in het geheugen van de computer bevindt, op het scherm brengen met het LIST bevel. (Als je ondertussen de computer uitgeschakeld had, zul je het programma opnieuw moeten intikken.)

```
LIST
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y)
50 GOTO 20
```

Verander nu regel 40 in: 40 PSET (X,Y),2

```
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y)■
50 GOTO 20
```

Plaats de cursor in deze positie, en typ ,**2** . (Vergeet de komma niet—die is belangrijk.) Druk eenmaal op RETURN om het zojuist getypte in te voeren, en nogmaals om de cursor onder het programma te zetten. Geef nu het RUN bevel.

De stippen zijn nu veranderd van wit in middelgroen, omdat 2 het codenummer voor die kleur is. Aan de hand van de kleurentabel op blz. 13 kunnen we uit een heel aantal kleuren kiezen. Of, wat ook interessant is, we kunnen de kleur een variabele maken, en de computer van stip tot stip de kleur laten veranderen.

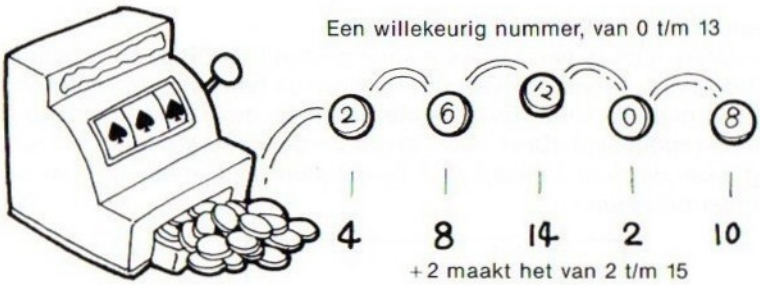
Ga terug naar regel 40 en wijzig de kleurcode in de variabele C.

```
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y),C
50 GOTO 20
```

Nu moeten we C nog een waarde geven, ofwel een bevel voor welke kleur we willen. Maar het is wel grappig om de waarde ook hier door de computer te laten bepalen, geheel willekeurig. Hier volgt het bevel waarmee je dat bereikt—maar voer dit op het moment nog even niet in.

```
35 C=INT(RND(1)*14)+2
```

Wat betekent het? Het is een bevel met een willekeurig getal, net als die voor X en Y, met een stapje meer aan het eind. Een willekeurig getal tussen 1 en 13 wordt gekozen, en dan wordt er 2 bij opgeteld (+2). Dat betekent dat het getal tussen 2 en 15 ligt. Maar er zijn 16 kleuren, van 0 t/m 15. Waarom laat dit bevel de kleuren 0 en 1 vallen? Omdat 0 transparant is, en doorzichtige stippen kun je nu eenmaal niet zien, en 1 is zwart, wat ook bepaald onzichtbaar is tegen een zwarte achtergrond.



Waar zullen we deze regel in het programma zetten? Het moet komen voor de stippen worden gezet in regel 40—dus onze nieuwe regel krijgt het nummer 35. (Dat is de reden waarom onze regels genummerd zijn als 10, 20, 30 ... om ruimte te laten voor nieuwe bevelen die we later willen toevoegen.) Laten we het nu dan maar in het programma zetten.

```
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y),C
50 GOTO 20
35 C=INT(RND(1)*14)+2
■
```

30, 40, 50, 35. Denk je dat de computer het zal begrijpen? Natuurlijk begrijpt die het, want hij kan tellen, en zet de regels automatisch in de juiste volgorde. Dit kan je gemakkelijk kontroleren—je hoeft daarvoor alleen maar opnieuw het LIST bevel te geven.

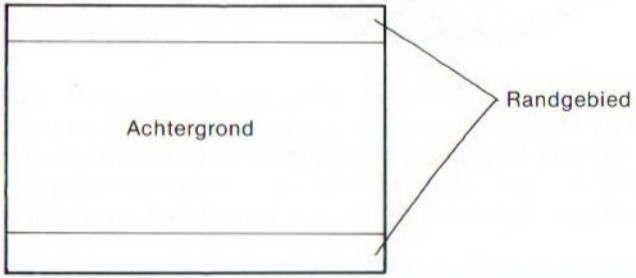
```
LIST
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
35 C=INT(RND(1)*14)+2
40 PSET (X,Y),C
50 GOTO 20
```

Nu zijn we zover gekomen dat we onze stippen in **sterren** kunnen veranderen. Sterren kun je alleen 's nachts zien, dus we zullen het scherm zwart maken. Tik deze nieuwe regel in:

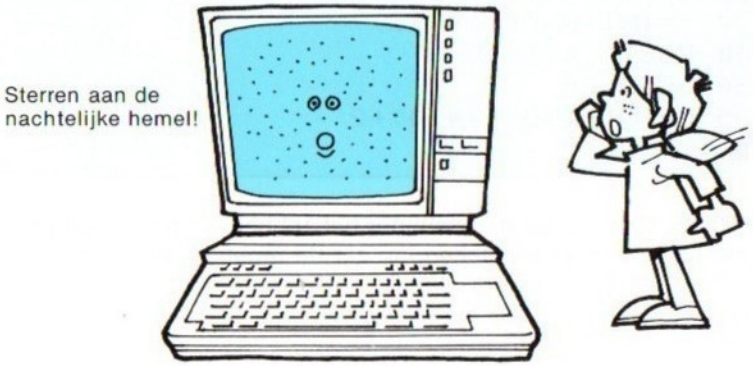
```
5 COLOR 15,1,1
```

Nieuwe bevelen voor kleuren en lijnen

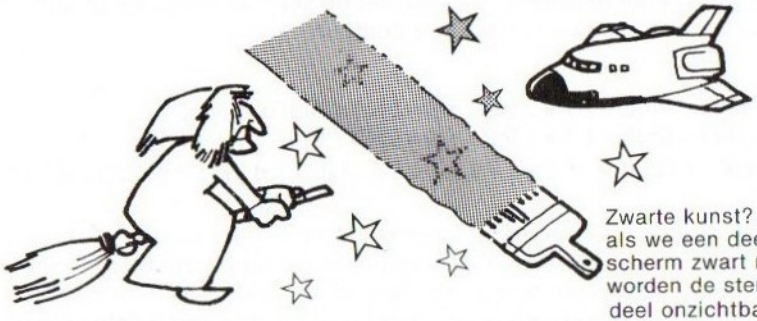
Die laatste regel is een nieuwe manier om het COLOR bevel te gebruiken. Hij bevat drie codes: 15 geeft de kleur van de **tekens** of de voorgrond, de eerste 1 geeft de kleur van de **achtergrond**, en de laatste 1 geeft de kleur van het **randgebied**. (Deze keer maken we de achtergrond en het randgebied dezelfde kleur. De keuze van de getallen is vrij, maar laten we nu deze maar proberen.)



Hiermee zijn we klaar, en: RUN.



Maar het is nog niet helemaal gedaan. We kunnen ze nog meer op werkelijke sterren laten lijken. Eerst moeten we het aantal sterren dat we maken beperkt houden—anders krijgen we nog een heel melkwegstelsel vol. Gebruik hiervoor CTRL en STOP, zoals gewoonlijk.



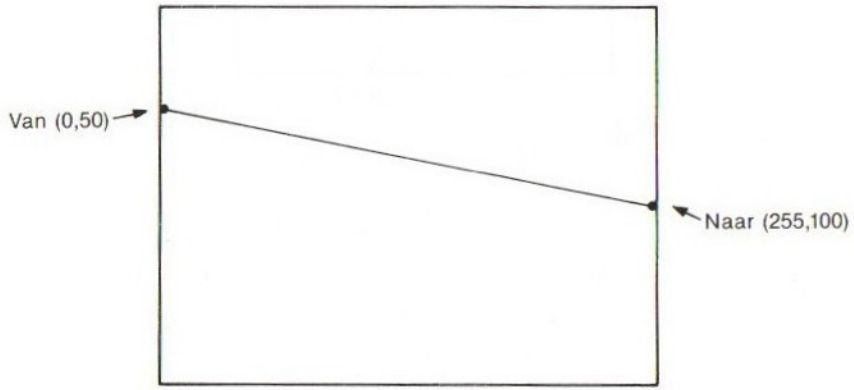
Zwarte kunst? Nee, maar als we een deel van het scherm zwart maken, worden de sterren in dat deel onzichtbaar.

Zwarte lijnen op het scherm trekken is natuurlijk een manier om een deel van het scherm zwart te maken. Het PSET bevel is alleen voor stippen; voor lijnen gebruiken we **LINE**. Hier volgt een bevel dat de computer vertelt waar hij een lijn moet trekken, en welke kleur daarvoor te gebruiken.

```
LINE(0,50)-(255,100),2
```

Voor het [-] verbindingsstreepje in het midden kun je overigens het minteken gebruiken.

Het verbindingsstreepje staat voor "van/naar", en dit bevel zorgt dat er een lijn wordt getrokken **van** positie (0,50) **naar** positie (255,100). Je weet natuurlijk dat het laatste gedeelte van dit bevel staat voor de kleur middelgroen. Als je het bovenstaande LINE bevel in je programma gebruikt zal dit op de volgende wijze een groene lijn trekken:

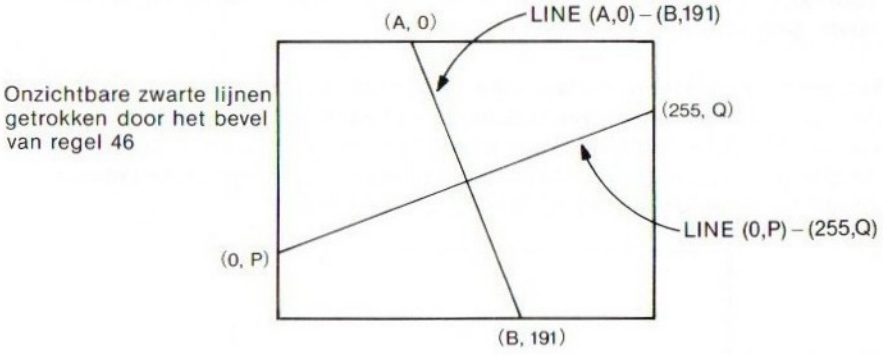


Nu je het LINE bevel kent, kan je het gebruiken om het sterrenprogramma te verbeteren. We willen een zwarte lijn trekken om een aantal sterren te doen verdwijnen, nietwaar? Dat betekent dat we de kleurcode in 1 moeten wijzigen. Bovendien gaan we weer variabelen gebruiken, om willekeurig een aantal lijnen te trekken, samen met de serie willekeurige stippen. Voer daarom drie nieuwe regels in je computer in:

```
42 A=INT(RND(1)*256):B=INT(RND(1)*256)
44 P=INT(RND(1)*192):Q=INT(RND(1)*192)
46 LINE (A,0)-(B,191),1:LINE (0,P)-(255,Q),1
```

Het bevel in deze drie regels is langer dan een regel van het scherm, maar dat vormt geen probleem. De computer zal automatisch het bevel op twee regels schrijven, terwijl het nog steeds **één regel** van het programma uitmaakt, aangezien er voor het bevel slechts **één regelnummer** gegeven is.

Regel nummer 46 bevat twee LINE bevelen, met vier nieuwe variabelen. Regels 42 en 44 zijn de bevelen met willekeurige getallen die we al kennen, en zij kennen waarden toe aan variabelen A, B, P en Q. Een paar willekeurige zwarte lijnen zou er als volgt kunnen uitzien:



Druk na het invoeren van de nieuwe regels 42, 44 en 46 telkens op RETURN om het bevel in te voeren, en typ tenslotte het bevel LIST om het hele programma te zien.

```
 5 COLOR 15,1,1
 10 SCREEN 2
 20 X=INT(RND(1)*256)
 30 Y=INT(RND(1)*192)
 35 C=INT(RND(1)*14)+2
 40 PSET (X,Y),C
 42 A=INT(RND(1)*256):B=INT(RND(1)*256
 )
 44 P=INT(RND(1)*192):Q=INT(RND(1)*192
 )
 46 LINE (A,0)-(B,191),1:LINE (0,P)-(2
 55,Q),1
 50 GOTO 20
```

Merk overigens op dat de bevelen in de nieuwe regels allemaal voor de GOTO lus van regel 50 komen. Dit betekent dat ze eindeloos herhaald worden, net als dat met de sterren het geval is. Telkens als de computer de bevelen van regel 20 tot en met regel 46 uitvoert maakt hij nieuwe stippen op andere plaatsen en vervolgens nieuwe lijnen op andere plaatsen. En als zich in de baan van een zwarte lijn een stip bevindt, dan zal deze onzichtbaar worden. Nu kun je het RUN bevel geven ... en we hebben sterren die echt **fonkelen!**

Als je deze lijnen werkelijk wilt zien, zodat je de werking van het geheel beter kunt begrijpen, dan kan dat door de kleurcode in het bevel van regel 46 te wijzigen van 1 in 2. Dan zul je groene lijnen op het scherm zien verschijnen.

OPMERKING: Ook na het stoppen van dit programma blijft het scherm zwart. Je kunt de oorspronkelijke donkerblauwe achtergrond en rand weer oproepen met dit bevel:

COLOR 15,4

15 betekent witte lettertekens, 4 is een donkerblauw scherm.

Wat je al wel gezien hebt maar nog niet was uitgelegd, is dat telkens wanneer je begint met BASIC het scherm automatisch in de SCREEN 0 stand komt te staan. Hierbij is de randkleur altijd gelijk aan de achtergrondkleur. Als je liever anders gekleurde randen om het scherm hebt, kun je dit doen door eerst het SCREEN 1 bevel te geven, en dan bijvoorbeeld het kleurbevel COLOR 15, 4, 7. Om het scherm daarna weer de oorspronkelijke kleur aan te doen nemen is het voldoende om het SCREEN 0 bevel te geven.

BEWAREN VAN JE PROGRAMMA



Hoe je ...

- de computer op een cassetterecorder aansluit
- een programma naar de cassetterecorder overbrengt
- kontroleert of een programma opgeslagen is
- een programma van een cassette in de computer invoert

Je zult wel eens een foto van een grote computer gezien hebben—zo een waarmee men de baan van een raket bestuurt, of de werking van een hele fabriek regelt, of ingewikkeld medisch onderzoek doet. Dan heb je waarschijnlijk ook de onderdelen opgemerkt die er uitzien als grote bandrecorders. En dat is eigenlijk precies wat het zijn. Bij de grote computers worden ze "tape units" genoemd, en ze worden gebruikt voor het opnemen van de informatie die de computer verwerkt, zodat die opgeslagen kan worden.

Een deel van de opgeslagen informatie bestaat uit programma's—de aanwijzingen voor de machine, dingen als PSET, SCREEN, PRINT en alle andere bevelen die je al kent. Wat ook opgeslagen wordt zijn de **gegevens** of data—feiten zoals de puntenverdeling van een spelletje, antwoorden op problemen, en wetenschappelijke formules.

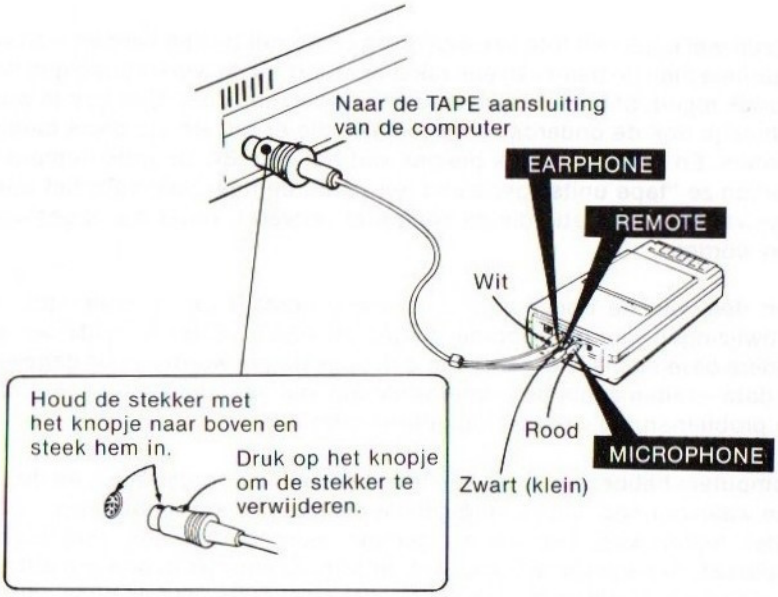
Computers hebben een zeer snelle ontwikkeling doorgemaakt, en de dingen waarvoor nog niet zo lang geleden een grote, zgn. "mainframe" computer nodig was, kunnen nu gedaan worden met een veel kleiner apparaat, zoals je Sony Computer. Je Sony Computer is zo klein dat een kind hem kan optillen, maar hij is sneller en slimmer dan de oude computers die vele miljoenen guldens kostten en een hele kamer in beslag namen.

Het zal je dus niet verrassen dat je Sony Computer ook in staat is tot het vastleggen van programma's en gegevens op de band. Aangezien hij een stuk kleiner is dan een "mainframe" computer kun je ook een kleinere magneetband gebruiken. Met je Sony Computer kun je een doodgewone cassetterecorder gebruiken, zo een als je ook gebruikt voor dikteren of het opnemen van muziek.

AANSLUITEN VAN DE CASSETTERECORDER

Het is heel eenvoudig om een cassetterecorder op de computer aan te sluiten met het speciale snoer dat hiervoor bij je Sony Computer wordt geleverd. Eén uiteinde van het snoer heeft een DIN stekker, een metalen ring met daarbinnen een aantal pennen. Deze stekker past in de aansluitbus gemerkt "TAPE" aan de achterkant van de computer.

Het andere uiteinde van het snoer heeft drie kleine stekkers, een rode, een witte en een zwarte. De rode kun je aansluiten op de mikrofooningang van je cassetterecorder (MIC of MICROPHONE) en de witte steek je in de hoofdtelefoonuitgang (PHONE, EARPHONE of HEADPHONES). Als je cassetterecorder een afstandsbedieningsaansluiting (REMOTE) heeft kun je hierin het zwarte stekkertje steken. (Als er geen REMOTE aansluiting is kun je het zwarte uiteinde los laten liggen.)



Nu zijn de computer en de cassetterecorder klaar om met elkaar te praten.

LAAT DE COMPUTER PRATEN

De computer zal de cassetterecorder vertellen wat er in zijn geheugen staat, zodra we hem dit bevel geven:

CSAVE "bestandsnaam"

De C in CSAVE staat voor cassette, en "SAVE" betekent hier bewaren of opslaan; we kunnen het programma dat zich in het geheugen van de computer bevindt opslaan door het op een cassette op te nemen. De "bestandsnaam" (en vergeet de aanhalingstekens niet wanneer je de naam typt) kan elke naam zijn die je voor je programma verzint, zolang je een bestand aan die naam kan herkennen—en zolang de cassetterecorder dit kan, en de computer.

Een bestandsnaam kan bestaan uit maximaal zes tekens. Dit kunnen zowel letters, cijfers als grafische tekens zijn, maar het **eerste** teken van een bestandsnaam moet een **letter** van het alfabet zijn.

STER is een goede bestandsnaam voor het programma dat we in het laatste hoofdstuk gemaakt hebben. Hij bestaat uit vier tekens, begint met een letter, en het is een handige naam omdat hij ons aan de inhoud van het programma doet denken.

Laten we het opslaan:

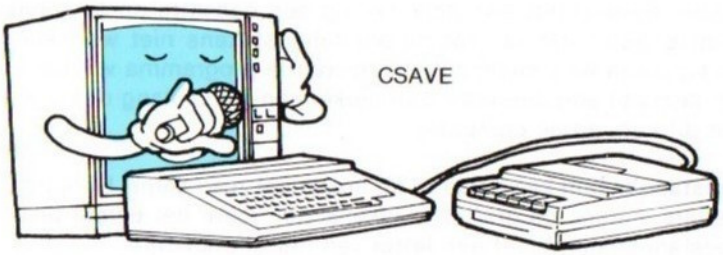
```
CSAVE "STER"
```

Even wachten, druk nog niet op RETURN. Zodra je dat doet, begint de computer de informatie die je wilt opslaan over te sturen naar de cassetterecorder. Eerst moeten we kontroleren of de cassetterecorder klaar is voor het opnemen van de informatie.

Als je recorder is uitgerust met een afstandsbedieningsaansluiting (REMOTE) en je hebt het snoer hierop aangesloten, dan kun je de recorder op opnemen zetten. Je merkt dan echter dat de band nog niet gaat lopen. Dat komt omdat de computer nu de cassetterecorder "bedient". Druk dan op RETURN en de band begint te lopen, het programma wordt opgenomen, en tenslotte slaat de cassetterecorder automatisch af. Nadat de opname beëindigd is toont het scherm

```
CSAVE "STER"
Ok
■
```

Als je cassetterecorder geen afstandbedieningsaansluiting (REMOTE) heeft, moet je **de opnametoets (REC of ●)** indrukken, en de band begint dan te lopen. **Wacht** dan een paar sekonden tot de bandsnelheid geheel stabiel is, en druk op RETURN. Wanneer je dan Ok op het scherm ziet verschijnen, kun je op de recorder **de stoptoets (STOP of ■)** indrukken, en dan ben je bijna klaar.



Nu heb je het programma op twee plaatsen. Het is nog steeds in het geheugen van de computer te vinden, en als alles goed gegaan is staat het ook op de band.

STAAT HET WERKELIJK OP DE BAND?

Het is belangrijk om **altijd te kontroleren** of de cassetterecorder je programma werkelijk op de band heeft gezet—en om dat eerst te doen, voor je iets anders doet. Je Sony Computer kan het kontroleren voor je doen, door de inhoud van zijn geheugen te vergelijken met de gemaakte opname. Maak eerst de aansluiting van het snoer op de afstandsbedieningsaansluiting (REMOTE) los. **Spoel dan de band terug** tot kort voor het punt waar de CSAVE opname gestart is. **Stel het volume van de cassetterecorder in op ongeveer de helft van het maximale volume. Sluit dan de "REMOTE" stekker weer aan** en typ het volgende bevel:

```
CLOAD? "STER"
```

Als er geen afstandsbedieningsaansluiting (REMOTE) is moet je **eerst RETURN indrukken** voor je op de cassetterecorder het afspelen start. Met een "REMOTE" stekker kun je de recorder op afspelen zetten en dan zal bij het indrukken van de RETURN toets de band automatisch beginnen te lopen. Zodra de computer dan het begin van het programma hoort, verschijnt op het scherm:

```
Found:STER
```

Terwijl de computer de cassette voor zichzelf afspeelt vergelijkt hij het opgenomen programma punt voor punt met het programma in zijn eigen geheugen, en geeft daarna zijn Ok als alles in orde is bevonden. (Vergeet niet daarna bij een cassetterecorder zonder afstandsbedieningsaansluiting (REMOTE) zelf de band te stoppen.)

Dit opslaan is een belangrijke handeling—je wilt immers niet al het werk dat je bij het schrijven van het programma gedaan hebt verliezen—en soms kunnen zich hierbij problemen voordoen. Als het bericht "Found" niet op het scherm verschijnt, dan kan dat bijvoorbeeld zijn omdat je de band niet ver genoeg hebt teruggespoeld.

Als het Ok van de computer **niet** verschijnt, dan kun je het volume van de cassetterecorder hoger instellen en het nogmaals proberen. Als dit niet helpt, dan is er sprake van elektrische storing of het programma was niet juist opgenomen. Kontroleer in dat geval alle verbindingen, en doe het geheel over vanaf CSAVE "STER".

Als het kontroleren goed verloopt en je ziet het Ok, dan weet je dat het programma is opgeslagen. Je kan dan de naam van het programma op de cassette noteren en die opbergen tot je het programma later weer wilt gebruiken.

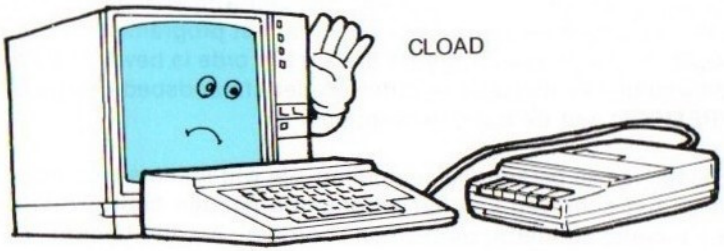
Als je **zeker weet** dat het programma veilig op de band is opgeslagen kan je NEW intikken of op de RESET toets drukken. Hierdoor wordt het **geheugen van de computer** gewist—maar niet het **extern geheugen** op de band.

LADEN VAN EEN PROGRAMMA

Gebruiken van een eerder op de band opgeslagen programma wordt "laden van een programma" genoemd, en het is nu een goed moment om dat te oefenen. De handelwijze is bijna hetzelfde als bij het kontroleren dat je zojuist gedaan hebt. Zorg dat de cassetterecorder klaar staat voor het **afspelen** naar de computer, en dat het juiste gedeelte van de band voor staat—vlak voor het begin van het opgenomen programma. Geef dan dit bevel:

```
CLOAD "STER"
```

Je zult zien dat er nu geen vraagteken (?) in voorkomt. Als je het programma overigens een andere bestandsnaam hebt gegeven, dan moet je die tussen de aanhalingstekens typen in plaats van STER.



Nadat de computer de band beluistert heeft, verschijnen op het scherm de woorden:

```
CLOAD "STER"
Found:STER
Ok
■
```

Vergeet niet de band stop te zetten—en dan ben je klaar om je programma met RUN te gaan verwerken.

Soms echter zal de computer aangeven:

```
Device I/O error
```

I/O betekent "input/output" ofwel **invoer/uitvoer**, en in dit geval slaat het op de geluidsverbinding met de cassetterecorder, waar iets mis mee is. Je kunt dan het volume wat hoger instellen, terugspoelen en het laden opnieuw proberen.

BESLISSINGEN EN SPELLETJES

Hoe je ...

- de computer beslissingen laat nemen
- een voorwaardelijke formule programmeert
- op je computer een spelletje speelt
- je programma vereenvoudigt
- je programma verbetert



Herinner je je Bobby nog, onze aardige hond? Hij is heel goed in het doen van eenvoudige kunstjes, zoals "Zit", "Breng" en "Geef een poot". Tot op heden heeft je Sony Computer net als Bobby alleen nog maar eenvoudige kunstjes gedaan—niet meer dan het opvolgen van onze bevelen als "PRINT", "GOTO" en "PSET", enz.

Wat later maakten we kennis met Superhond, en die deed een paar moeilijker kunstjes, gebaseerd op "als de bal wit is..." of "als de bal zwart is...". Superhond kon **beslissingen nemen**. Je weet nu al dat je Sony Computer slim genoeg is om dit soort dingen te doen, en dus gaan we nu een paar "super-programma's" schrijven, programma's die beslissingen nemen.

Wis om te beginnen je laatste programma met het NEW bevel, en typ dan het volgende programma:

```
10 A=INT(RND(1)*5)+1
20 INPUT B
30 IF A=B THEN GOTO 50
40 GOTO 10
50 PRINT "Raak!"
60 GOTO 10
```

Als je zeker weet dat je geen vergissingen hebt gemaakt bij het intikken, kun je het bevel RUN geven en op RETURN drukken.

```
RUN
? ■
```

De computer heeft ons al eerder een vraagteken getoond, maar deze keer blijft de cursor op dezelfde regel staan en verspringt niet naar de volgende. Dat betekent dat de computer wacht tot jij iets **invoert**—vanwege het INPUT bevel dat je in regel 20 gegeven hebt. Iets invoeren—in dit programma een getal van 1 tot 5—doe je door het indrukken van een toets, zoals je weet. Dus laten we de computer maar eens het getal 3 geven. (Wat denk je dat er gebeurt als we de computer een letter of een grafisch teken geven?)

Druk op [3] en [RETURN]. Wat krijg je nu op het scherm te zien?

```
? 3
Raak!
? ■
```

```
? 3
? ■
```

In beide gevallen zien we aan het eind weer een vraagteken, zodat je weet dat de computer om nog een getal vraagt. En nog een. En nog een. En nog een. Dat is het spelletje: is het nummer dat je geeft goed geraden—"Raak!"—of is het een misser? Hoe vaak krijg je "Raak!" te zien? (Volgens de wetten van de kansberekening zal het ongeveer een keer raak zijn in vijf pogingen, ofwel twee keer op de tien, ofwel 20 keer raak op de 100). Je zal er wel eerder moe van worden dan de computer (die wordt **nooit** moe), dus druk maar op [CTRL] en [STOP].

Wat gebeurt er hier? Kijken we naar het programma. De eerste regel is een oude bekende, het **bevel voor een willekeurig getal**. Telkens als de computer de lus doorloopt (regel 60), kiest hij een nummer van 1 tot 5, en geeft die waarde aan variabele A.

Nu maken we in regel 20 kennis met een nieuwe vriend:

```
20 INPUT B
```

B is een tweede variabele, en die wacht tot je hem via het toetsenbord een waarde toekent. Zodra je een getal ingetikt hebt, hebben beide variabelen een waarde en gaat de computer naar de volgende regel.



A is een getal van 1 tot 5



B geven we bij de eerste keer raden een waarde van 3

RAAK OF MIS—DE COMPUTER BESLIST

Denk bij regel 30 nog maar eens terug aan de kunstjes van Superhond—twee daarvan begonnen met het woord indien, waarvoor we in BASIC het Engelse woord IF gebruiken.

```
30 IF A=B THEN GOTO 50
```

IF gaat in BASIC **altijd** samen met THEN. Dit betekent dat je zinnen maakt als "INDIEN (IF) je honger hebt, DAN (THEN) moet je iets eten" of "INDIEN je een Sony Computer hebt, DAN kun je leren programmeren".

IF [voorwaarde] THEN [handeling]

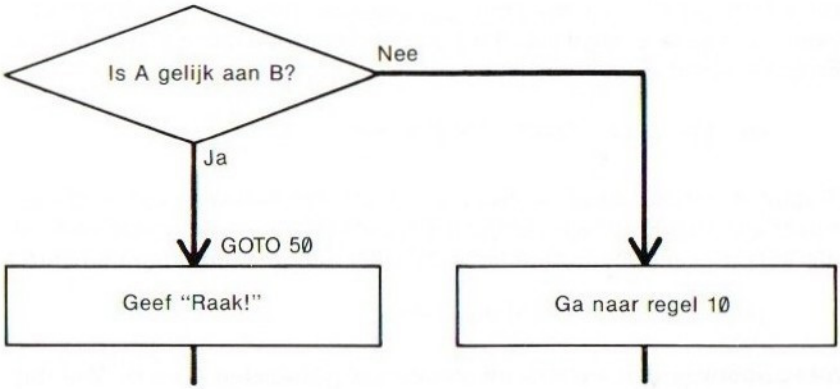
Welke **voorwaarden** kunnen we stellen aan variabelen A en B? Wel, bijvoorbeeld dat ze aan elkaar gelijk zijn (A=B), of dat A groter is (A>B), of dat B groter is (A<B).

De "handeling" kan uitgedrukt worden door elk bevel. In dit geval is het GOTO 50.

Deze regel als geheel wordt een **voorwaardelijke formule** genoemd. Er bestaan in BASIC zes van deze formules, en een aantal daarvan kan op twee verschillende manieren geschreven worden.

Symbool	Betekenis	Voorbeeld	
=	Gelijk aan	IF A=B	Indien A gelijk is aan B
>	Groter dan	IF A>B	Indien A groter is dan B
<	Kleiner dan	IF A<B	Indien A kleiner is dan B
>=	Groter dan of gelijk aan	IF A>=B	Indien A groter dan of gelijk is aan B
=>		IF A=>B	
<=	Kleiner dan of gelijk aan	IF A<=B	Indien A kleiner dan of gelijk is aan B
=<		IF A=<B	
<>	Niet gelijk, ongelijk aan	IF A<>B	Indien A niet gelijk is aan B
><		IF A><B	

Regel 30 is dus een voorwaardelijke formule. Als het gedeelte met IF waar is (A=B), dan vervolgt de computer met het uitvoeren van het bevel (GOTO 50). Maar als A en B niet aan elkaar gelijk zijn, dan vervolgt de computer met de volgende regel van het programma, regel 40, zonder eerst het bevel na THEN te lezen. Regel 40 is zoals je ziet een bevel om terug te gaan naar het begin, regel 10, waar de computer opnieuw een willekeurig getal kiest—wat jij dan weer moet zien te raden.



Nu zal het je duidelijk zijn dat "Raak!" alleen kan worden weergegeven indien A en B—B is het getal dat je bij het raden intikt—aan elkaar gelijk zijn. "Raak!" betekent dat het getal dat je voor variabele B invoerde gelijk is aan de waarde die de computer aan variabele A gaf.

Nadat hij "Raak!" op het scherm heeft gezet gaat de computer door naar regel 60, die hem terugstuurt naar 10, en het spelletje wordt herhaald. In beide gevallen gaat het spelletje door, of A en B nu gelijk zijn of niet.

VEREENVOUDIGEN VAN HET PROGRAMMA

Het is van belang dat computerprogramma's altijd zo eenvoudig mogelijk geschreven zijn, opdat ze makkelijker te begrijpen zijn en er minder kans is op vergissingen bij het typen. Laten we nog eens naar ons raadselprogramma kijken.

```
10 A=INT(RND(1)*5)+1
20 INPUT B
30 IF A=B THEN GOTO 50
40 GOTO 10
50 PRINT "Raak!"
60 GOTO 10
```

We weten nu: als A=B, dan geeft de computer "Raak!" weer, Dat klinkt alsof er een enkele handeling plaatsvindt, maar in feite staan hiervoor twee bevelen in het programma. Het eerste komt na THEN in regel 30 en luidt: GOTO 50. Het tweede vind je in regel 50 en dit luidt: PRINT "Raak!" Een enkele handeling moet in een enkel bevel uitgedrukt kunnen worden—en dus veranderen we regel 30 in

```
IF A=B THEN PRINT "Raak!"
```

En nu hebben we de bevelen in regel 50 en 60 niet meer nodig, zie je wel?

```
10 A=INT(RND(1)*5)+1
20 INPUT B
30 IF A=B THEN PRINT "Raak!"
40 GOTO 10
```

Veel eenvoudiger! Als A en B gelijk zijn, dan gaat de computer na het weergeven van "Raak!" verder met regel 40 en, GOTO 10, terug naar 10. Als A en B niet gelijk zijn, gebeurt het zelfde, behalve het weergeven van "Raak!". Ons spelletje werkt nu met slechts vier bevelen.

Breng nu de wijzigingen op het scherm aan. Verander eerst het bevel in regel 30 van GOTO 50 in PRINT "Raak!". Wis dan regel 50 en 60 uit, als volgt: Zet de cursor op een open regel onder regel 60 en onder Ok, en typ 50. Druk op RETURN, en regel 50 is uit het geheugen van de computer gewist. Doe hetzelfde, typ 60 en RETURN om regel 60 te wissen.

Zijn regel 50 en 60 van het scherm verdwenen? Nee, ze zijn er nog! Maar ze zijn wel uit het geheugen gewist. Als je nu het bevel LIST geeft, dan zal de computer op het scherm brengen wat er nu in het geheugen aanwezig is:

```
LIST
10 A=INT(RND(1)*5)+1
20 INPUT B
30 IF A=B THEN PRINT "Raak!"
40 GOTO 10
```

Dat ziet er beter uit, vind je niet?

Enkele verbeteringen

We hebben het raadselprogramma nu zo eenvoudig gemaakt als maar mogelijk is, en nu kunnen we proberen een aantal verfijningen aan te brengen, handige en grappige dingen toe te voegen. Natuurlijk houden we alle nieuwe snufjes ook wel weer zo simpel mogelijk.

Als A gelijk is aan B, dan laat de computer je dat weten met "Raak!". Maar als ze niet gelijk zijn, dan krijg je dat niet te zien voordat het programma alweer herhaald wordt. Het programma is voor je vrienden wat gemakkelijker te begrijpen als we regel 30 als volgt wijzigen:

```
30 IF A=B THEN PRINT "Raak!" ELSE PRINT "Mis"
```

ELSE betekent "zo niet,...". Het is na een voorwaardelijke formule een handig bevel, omdat het die stap van het programma verduidelijkt.

IF [voorwaardelijke formule] THEN [bevel 1] ELSE [bevel 2]

Aangezien de regel met dit nieuwe bevel langer wordt dan 37 lettertekens, zal hij "overlopen" of doorlopen op de volgende regel van het scherm. Regel 40 zal hierdoor tijdelijk verdwijnen. Daar hoef je je geen zorgen over te maken—hij is nog steeds in het geheugen van de computer aanwezig; om hem terug te brengen hoef je alleen maar opnieuw het LIST bevel te geven!
Er is nog een verbetering die het programma wat duidelijker maakt wanneer je het aan je vrienden toont. We kunnen regel 20 wijzigen in:

```
20 INPUT "Noem een getal(1-5) ";B
```

Je zal de betekenis van dit nieuwe bevel gauw genoeg zien als je het op het scherm typt, op [RETURN] drukt en dan het RUN bevel geeft.

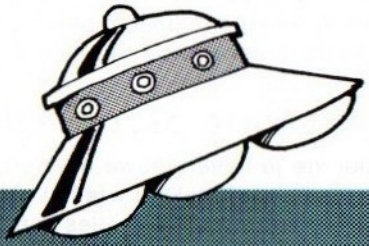
```
RUN
Noem een getal(1-5) ? ■
```

Dit maakt het een heel stuk duidelijker, nietwaar? Veel beter dan het kale "?" bericht dat we een paar minuten geleden te zien kregen. Nu kan je het spelletje aan je vrienden tonen en zal iedereen je programma onmiddellijk begrijpen.

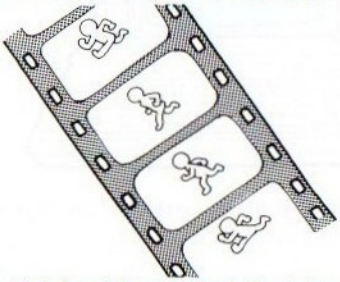
BEWEGENDE VOORSTELLINGEN

Hoe je ...

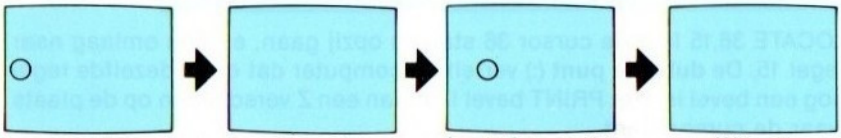
- lettertekens over het scherm kan laten bewegen
- een UFO film maakt
- variabelen gebruikt om een programma in te korten

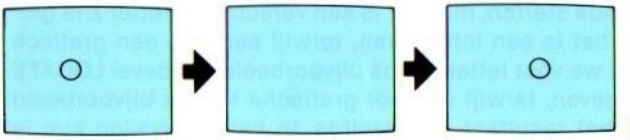


Zoals je wel zult weten bewegen de figuren op een film niet echt. Een film is niets meer dan een reeks stilstaande beelden, snel achtereen weergegeven. Doordat elk beeld slechts een klein beetje verschilt van het vorige en het volgende wordt de illusie van beweging gekreëerd.



Televisie is hetzelfde. Het beeld op ons televisiescherm verandert een groot aantal malen per sekonde, en het resultaat is dat we beweging denken te zien. Nu is het scherm van je Sony Computer bijna het zelfde als een gewoon TV-scherm, dus natuurlijk kunnen we dit ook gebruiken om bewegende beelden te maken. Dat gaat zo:



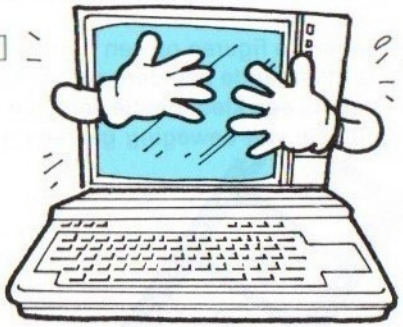


Om het 0 teken over het scherm te doen bewegen laten we het eerst dichtbij de rand van het scherm zien, wissen het dan uit, laten het dan even verderop weer zien, wissen het weer uit... enzovoort. Om één stap van deze serie te doen moeten we de computer opdragen om te wissen, dan de juiste positie te bepalen, en vervolgens om iets af te beelden.

```
CLS
LOCATE 36,15:PRINT "Z"
```

Hier zie je twee nieuwe bevelen, en dan PRINT. CLS staat voor "Clear Screen", wat betekent **wis het scherm**. Als we dit bevel geheel afzonderlijk invoeren verdwijnt alles van het scherm.

CLS + RETURN



"Wat is er met mijn gezicht gebeurd?"

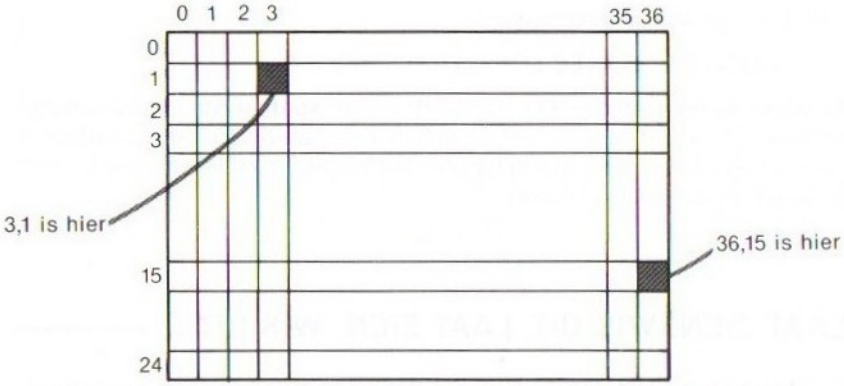
Om te zien wat het LOCATE bevel betekent moet je bovenstaande drie bevelen maar eens samen invoeren.

```
LOCATE 36,15:PRINT "Z"

Ok                                  Z
■
```

LOCATE 36,15 laat de cursor 36 stapjes opzij gaan, en dan omlaag naar regel 15. De **dubbele punt** (:) vertelt de computer dat er op dezelfde regel nog een bevel is. Het PRINT bevel laat dan een Z verschijnen op de plaats waar de cursor staat.

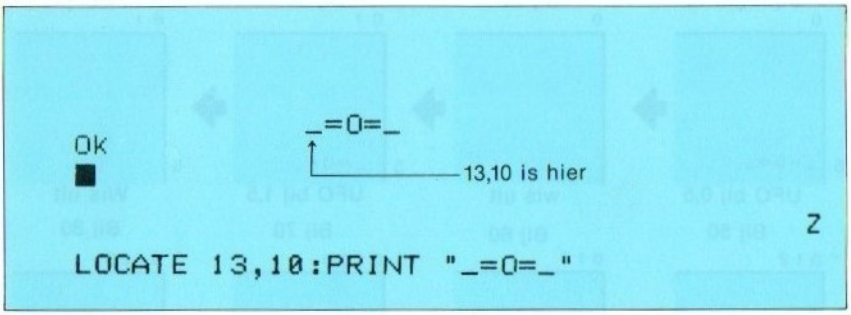
Het lijkt sterk op het PSET bevel dat we eerder gebruikt hebben voor het maken van de stralende sterren, maar er is een verschil: de letter Z is groter dan een stip, en het is een letterteken, terwijl een stip een grafisch teken is. Zo zie je dat we voor lettertekens bijvoorbeeld het bevel LOCATE 36,15: PRINT "iets" geven, terwijl we voor grafische tekens bijvoorbeeld PSET (36,15) typen; het resultaat is hetzelfde. In beide gevallen kun je 36,15 beschouwen als een **adres**.



Het PRINT bevel, dat op het LOCATE bevel en de dubbele punt (:) volgt, is niet beperkt tot een enkele letter als "Z". Je kan met het LOCATE bevel een plaats aangeven voor **elk gewenst aantal** letters of woorden. De funktie van LOCATE is slechts om de cursor naar een bepaalde plaats te doen bewegen, voor het PRINT bevel wordt uitgevoerd. Je kan LOCATE gebruiken om een paragraaf midden op een bladzijde te laten beginnen, om een spelletje te starten aan de onderkant van het scherm, of om wat dan ook te plaatsen waar je het wilt hebben.

Laten we de fantasie eens inschakelen. Ben je klaar om een UFO, een vliegende schotel, te zien? (UFO staat voor "Unidentified Flying Objects"—ruimteschepen van een andere planeet—en zij gebruiken hoogstwaarschijnlijk ook computers.) We kunnen een UFO maken met niet meer dan vijf tekens:

```
LOCATE 13,10:PRINT "_=O=_"
```



Vind je dat het wel wat op een UFO lijkt? Je kan ook andere vormen maken, hoor, er zijn genoeg verschillende letters en andere tekens op het toetsenbord te vinden. Deze UFO bestaat uit onderstrepingen (_), isgelijk tekens (=), en de letter O.

Laten we nu iets anders doen:

```
LOCATE 13,10:PRINT "     "
```

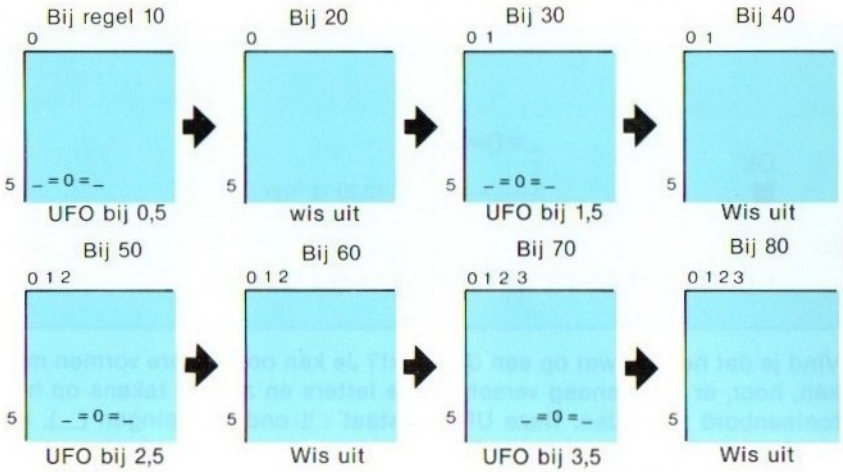
Druk op RETURN en de UFO verdwijnt! De vijf spaties zijn in de plaats gekomen van de vijf tekens. Een spatie is een handig foefje om letters of hele woorden van het scherm te laten verdwijnen zonder het hele scherm te wissen met het CLS bevel.

LAAT ZIEN, WIS UIT, LAAT ZIEN, WIS UIT ...

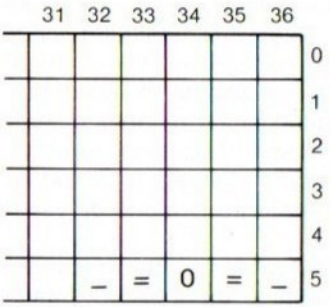
De twee bevelen LOCATE en CLS zijn dus samen de manier om een bewegende voorstelling te maken. Hieronder volgt een programma dat onze UFO kan laten bewegen. (Nog even niet gaan typen. Kijk eerst of je goed begrijpt hoe het werkt.)

```
5 CLS
10 LOCATE 0,5:PRINT "_=0=_"
20 LOCATE 0,5:PRINT "     "
30 LOCATE 1,5:PRINT "_=0=_"
40 LOCATE 1,5:PRINT "     "
50 LOCATE 2,5:PRINT "_=0=_"
60 LOCATE 2,5:PRINT "     "
70 LOCATE 3,5:PRINT "_=0=_"
80 LOCATE 3,5:PRINT "     "
```

Zo zouden deze bevelen er op het scherm uit komen te zien:



De eerste regel, CLS, wist het gehele scherm, en daarna zet elk paar regels—10 en 20, 30 en 40, 50 en 60, 70 en 80—de UFO op een andere plaats, een ander adres. Als we dit voortzetten tot we de rechterkant van het scherm bereiken, zo:



Dan zullen de laatste bevelen zo luiden:

```
650 LOCATE 32,5:PRINT "_=0=_"
660 LOCATE 32,5:PRINT "     "
```

Het gehele programma is dan 67 regels lang. Hallo zeg...! Dan kunnen we beter gewoon TV gaan kijken!

Maar loop nog niet weg. Er is een veel slimmere manier. Je hebt al gemerkt dat na regel 5 alle bevelen sterk op elkaar lijken, en dat is de sleutel. Het programma wordt grotendeels gevormd door paren regels die bijna precies hetzelfde zijn.

```
10 LOCATE 0,5:PRINT "_=0=_"
20 LOCATE 0,5:PRINT "     "
```

Dit deel verandert nooit

Dit is altijd 5

Alleen deze positie verandert: 0,0,1,1,2,2,.....32,32

Alleen het eerste nummer van het adres varieert, en dat loopt systematisch op. Dat betekent (en je hebt het misschien al geraden) dat het eerste nummer een **variabele** wordt. Nu zie je hoe handig variabelen zijn: we kunnen het hele programma inkorten tot maar zeven regels.

```
100 CLS
110 I=0
120 LOCATE I,5:PRINT "_=0=_"
130 LOCATE I,5:PRINT "     "
140 I=I+1
150 IF I<33 THEN GOTO 120
160 END
```

Wis het geheugen van de computer met het NEW bevel, en voer dan dit programma in. (We beginnen deze keer bij regel nummer 100, zodat we later wat meer regels kunnen toevoegen.) Kijk nou je UFO eens: hij vliegt!

Laten we het programma regel voor regel doornemen, zodat we kunnen zien **waarom** hij vliegt. Regel 100 maakt het scherm vrij. (We zouden niet willen dat onze UFO ergens tegenaan vliegt.) Regel 110 geeft de variabele eerst een naam, en dan een waarde. "I" wordt hier de naam van de variabele die de positie van links naar rechts aangeeft, en aangezien we aan de linkerkant van het scherm beginnen geven we hem de **beginwaarde** nul. Regel 120 en 130 vormen het regelpaar dat we eerder gezien hebben als "laat zien/wis uit", maar nu is hierin de variabele "I" opgenomen.

Regel 140 verandert de waarde van de variabele: hij luidt I=I+1. Vergeet niet dat dit geen wiskunde is, dit is BASIC. Daarom betekent "=" niet "is gelijk aan", maar "krijgt de waarde ...". Zodoende wordt "I" met 1 verhoogd voordat we naar regel 150 gaan.

Regel 150 zorgt dat de computer terug gaat naar regel 120, zolang "I" kleiner is (<) dan 33. Nu verhogen we "I" telkens met één, en dus wordt de UFO telkens een stapje verder naar rechts weergegeven. Zodra "I" niet meer kleiner is dan 33 leest de computer het bevel "GOTO 120" niet meer, maar gaat in plaats daarvan verder naar regel 160. In feite zou zelfs zonder het bevel "END" van regel 160 het programma stoppen, omdat er dan na regel 150 geen bevel meer zou volgen.

Maar dit programma bevat een **lus** waarbinnen de computer keer op keer rond gaat van regel 120 naar 150, en weer terug naar 120. Deze lus is afhankelijk van de voorwaarde dat "I" kleiner is dan 33, en daarom noemen we dit een **voorwaardelijke lus**.

Een nog betere manier

Laten we dat programma nog eens goed bekijken.

```
100 CLS
110 I=0          <-- De beginwaarde is nul
120 LOCATE I,5:PRINT "_=O=_"
130 LOCATE I,5:PRINT "     "
140 I=I+1
150 IF I<33 THEN GOTO 120   <-- De waarde loopt bij elke rondgang door de lus met één op; de lus stopt bij 33
160 END
```

De belangrijkste onderdelen van dit programma zijn de regels 110, 140 en 150. Die vertellen de computer waar hij moet beginnen (bij 0), en dat de lus 33 maal herhaald moet worden. Dergelijke aanwijzingen voor het herhalen van een lus komen in vele programma's voor, en BASIC beschikt hiervoor over een speciale kombinatie van twee bevelen, **FOR** en **NEXT**.

```
100 CLS
110 FOR I=0 TO 32
120 LOCATE I,5:PRINT "_=0=_"
130 LOCATE I,5:PRINT "      "
140 NEXT I
150 END
```

Herhaal van 0 t/m 32

Verhoog de waarde van I met 1; ga terug naar regel 120

Deze twee bevelen, FOR in regel 110 en NEXT in regel 140, kunnen we de Lus Experts noemen. Hiermee hebben we maar twee regels nodig waar we eerder drie gebruikten.

Wat belangrijk is om goed te onthouden is deze vorm:

FOR [variabele] = [beginwaarde] **TO** [eindwaarde]

aan het begin van de lus, en

NEXT [variabele]

aan het eind.

In ons programma zijn dat

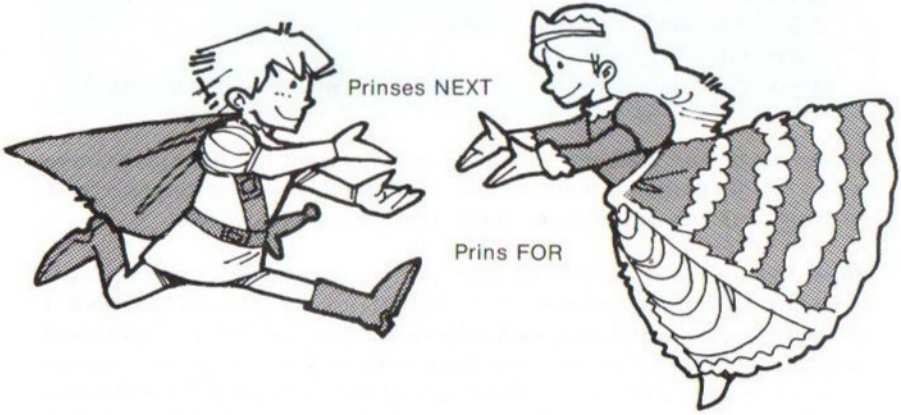
```
110 FOR I=0 TO 32
```

en

```
NEXT I
```

FOR vertelt de computer hoeveel maal de lus rondgegaan moet worden. NEXT is een heel slim bevel, dat de stappen telt en de computer naar de volgende regel van het programma stuurt zodra de variabele de eindwaarde bereikt.

FOR en NEXT gaan **altijd** samen als een paar.



Als Prins FOR zijn Prinses NEXT nergens kan vinden, is de computer hierover zo bedroefd dat hij een foutmelding weergeeft en weigert verder nog iets te doen voor je de fout korrigeert.

ALLES BIJ ELKAAR GENOMEN

Hoe je ...

- kleur en geluid aan je programma's toevoegt
- twee programma's samenvoegt
- een stroomschema maakt
- je programma verbetert



TOEVOEGEN VAN KLEUR EN GELUID

Nu zijn we klaar om een paar eerder geleerde funkties toe te voegen—zodat we een kleurrijke, hoorbare bewegende UFO krijgen. Voer dit programma in je computer in:

```
100 COLOR 15,1:CLS
103 Y=INT(RND(1)*22)
106 C=INT(RND(1)*14)+2:COLOR C
110 FOR I=0 TO 32
120 LOCATE I,Y:PRINT "_=0=_"
130 LOCATE I,Y:PRINT "     "
133 A=I MOD 12
136 IF A=0 THEN BEEP
140 NEXT I
150 GOTO 100
```

Herhaald met FOR—NEXT

Je kan dit blijven bekijken zolang als je wilt, want het hele programma is een lus die herhaald blijft worden, maar laten we eerst maar op CTRL en STOP drukken om te gaan kijken hoe de bevelen werken.

Regel 100 doet twee dingen: hij stelt de kleur in (zwarte achtergrond, witte tekens) en hij wist het scherm. 103 geeft ons een nieuwe variabele, Y, en kent deze een **willekeurige waarde** van 0 tot 21 toe. 106 doet iets soortgelijks: hij geeft een willekeurige waarde tusen 2 en 15 aan een nieuwe variabele C, en maakt die variabele dan tot het nummer dat de kleur aangeeft. (Kleur nummer 1 kunnen we niet gebruiken, want zwarte tekens zouden onzichtbaar zijn!)

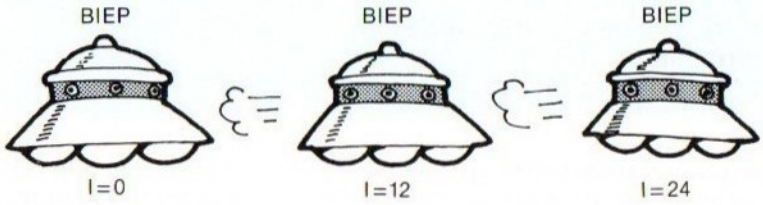
Beginnend met regel 110 hebben we dan de FOR-NEXT lus, en deze keer maakt de variabele Y deel uit van het "adres" van beide LOCATE bevelen. De horizontale positie (van links naar rechts) wordt bepaald door I, die op regelmatige wijze varieert van 0 tot 32, maar de vertikale positie (van boven naar beneden) is in regel 103 **willekeurig** (RND) gemaakt: Y kan elke waarde aannemen van 0 t/m 21. Daarom verschijnen die verschillend gekleurde UFO's steeds op een andere hoogte.

Kijk nu naar regel 133 en 136:

```
A=I MOD 12
IF A=0 THEN BEEP
```

Hier vind je twee nieuwe woorden. BEEP is het BASIC woord voor een pieptoon—het geluid dat je UFO maakt. Maar hij piept alleen als A gelijk is aan nul, en A krijgt de waarde van I MOD 12, en dus willen we weten wat MOD betekent. MOD staat voor "modulus" en betekent de **rest**, die overblijft als je een getal door een ander deelt. I MOD 12 is het restgetal dat je overhoudt wanneer je I door 12 deelt. Als I gelijk is aan 15 dan is I MOD 12 gelijk aan 3. Maar als I 24 is, dan is I MOD 12 gelijk aan 0.

Telkens wanneer I oploopt van 0 tot 32, dus voor elke UFO, wordt I MOD 12 drie keer gelijk aan nul: bij 0, 12 en 24. Daarom geeft elke UFO die het scherm passeert drie keer een pieptoon.



RAADSEL+GELUID+VIDEO

Een van de meest interessante dingen bij het programmeren van computers is het samenvoegen van twee programma's tot een geheel. Het kombineren van twee of meer programma's kan heel handig en bovendien leuk zijn, zoals je hier zult zien.

Daar gaan ze weer:

Raadselprogramma

```
10 A=INT(RND(1)*5)+1
20 INPUT "Noem een getal";B
30 IF A=B THEN PRINT "Raak!" ELSE PRI
NT "Mis"
40 GOTO 10
```



UFO programma

```
100 COLOR 15,1:CLS
103 Y=INT(RND(1)*22)
106 C=INT(RND(1)*14)+2:COLOR C
110 FOR I=0 TO 32
120 LOCATE I,Y:PRINT "_=O=_"
130 LOCATE I,Y:PRINT "     "
133 A=I MOD 12
136 IF A=0 THEN BEEP
140 NEXT I
150 GOTO 100
```

Als we gewoon allebei de programma's een voor een in de computer stoppen zie je dat ze niet goed samen gaan. Het programma loopt tot regel 40, keert dan terug naar regel 10, en de UFO's komen nooit aan de beurt. De eerste wijziging die we moeten aanbrengen is dan ook om de UFO's aan te laten sluiten op het eerste programma, en de beste plaats voor aansluiting is bij een juist geraden getal. Dat is regel 30:

```
30 IF A=B THEN GOTO 100 ELSE PRINT "Mis"
```

We hebben nog een klein probleempje: hoe onderbreken we de UFO's nadat iemand het juiste nummer raadt? We willen niet eindeloos naar de UFO's blijven kijken, maar stopzetten van de computer is ook niet de bedoeling—we willen terug naar het raadseltje.

Daarvoor moeten we regel 150 wijzigen in:

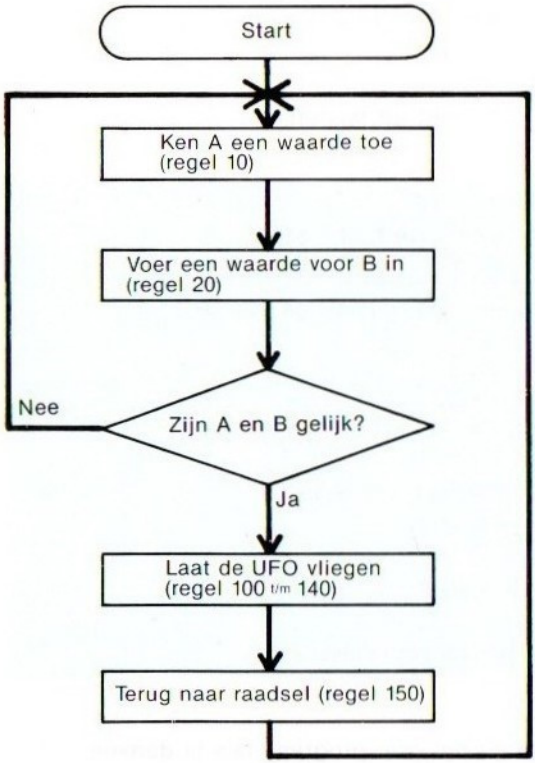
```
150 GOTO 10
```

Nu komt het programma er als volgt uit te zien:

```
10 A=INT(RND(1)*5)+1
20 INPUT "Noem een getal";B
30 IF A=B THEN GOTO 100 ELSE PRINT "M
is"                  ↑
                     └──Indien Raak! ga naar UFO
40 GOTO 10
100 COLOR 15,1:CLS
103 Y=INT(RND(1)*22)
106 C=INT(RND(1)*14)+2:COLOR C
110 FOR I=0 TO 32
120 LOCATE I,Y:PRINT "_=0=_"
130 LOCATE I,Y:PRINT "      "
133 A=I MOD 12
136 IF A=0 THEN BEEP
140 NEXT I
150 GOTO 10← Terug naar het raadsel
```

Het stroomschema: een manier om over programma's te denken
Onze programma's worden al langer en ingewikkelder, met lussen en dubbele funkties. Wanneer computer-professionals ingewikkelde programma's maken, dan werken ze het geheel zorgvuldig uit voor ze bevelen gaan schrijven. De beste manier om dat te doen is met een **stroomschema**.

In dit boek gebruiken we geen stroomschema's, en ze zijn niet echt nodig om je Sony Computer te begrijpen. Je kunt dus gerust naar het volgende hoofdstuk gaan, als je wilt. Maar om je eens te laten zien wat een stroomschema is, hebben we er hier een voor ons raadsel/UFO programma.





Het programma start bij het bovenste blok, en elk van de andere blokken geeft een bewerking waarvoor een bevel nodig is (of verscheidene bevelen). De lijn geeft de volgorde van de verwerking aan, ofwel de logische lijn van het programma. Uit het ruitvormige blok komen twee verschillende lijnen; dat betekent dat hier een voorwaarde (IF ...) door de computer gekontroleerd wordt, waarna de computer bepaald welke richting hij dient te volgen. Overigens kun je hier ook goed zien dat zowel de "Nee" lijn als de onderste regel een lus maken waardoor het programma herhaald wordt.

Verbeteren van het raadsel-spelletje

Het verschil tussen een redelijk programma (een dat werkt) en een **uitstekend** programma wordt vaak bepaald door de verbeteringen die worden aangebracht nadat het programma is geschreven. Hier volgen drie dingen die we zouden kunnen verbeteren in ons raadselprogramma.

- Na elke UFO verschijnt het volgende getalraadsel in de zelfde kleur als de UFO. Zou het raadsel niet beter leesbaar zijn als het alleen in wit op het scherm zou komen?

● Het raadsel gaat willekeurig het scherm op en neer, altijd een regel onder de laatst passerende UFO. Kan het niet stilstaan?
● Met het voorbij vliegen van de eerste UFO verandert het scherm van donkerblauw in zwart, en daarna blijft het zwart. Zou het niet beter zijn om het hele programma lang dezelfde achtergrondkleur aan te houden?

Met een paar simpele toevoegingen en ingrepen kunnen we deze problemen vlot oplossen.

```
5 COLOR 15,1                    <-- Voeg toe
7 CLS:LOCATE 0,0                <-- Voeg toe
10 A=INT(RND(1)*5)+1
20 INPUT "Noem een getal";B
30 IF A=B THEN GOTO 100 ELSE PRINT "M
is"
40 GOTO 10
100 CLS                         <-- Wijzig
103 Y=INT(RND(1)*22)
106 C=INT(RND(1)*14)+2:COLOR C
110 FOR I=0 TO 32
120 LOCATE I,Y:PRINT "_=0=_"
130 LOCATE I,Y:PRINT "      "
133 A=I MOD 12
136 IF A=0 THEN BEEP
140 NEXT I
150 GOTO 5                      <-- Wijzig
```

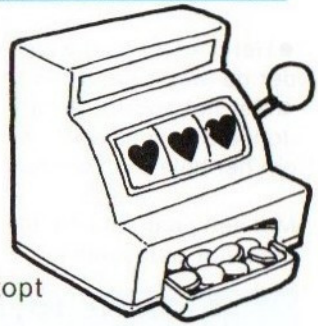
Kijk voordat je verder leest eerst eens of je deze wijzigingen uit jezelf begrijpt ...

Heb je het begrepen? De nieuwe regels 5 en 7 lossen alle problemen netjes op. Regel 5 geeft het bevel voor een zwarte achtergrond en witte letters aan het begin. Regel 7 zorgt dat het raadsel altijd bovenaan het scherm komt te staan.

Het belangrijkste feit om te onthouden bij het wijzigen van een programma is echter dit: **een wijziging op een plaats betekent vaak extra wijzigingen elders**. In dit gevel kwam het COLOR bevel in regel 5 uit regel 100, dus moeten we regel 100 wijzigen in alleen CLS. En aangezien het programma nu begint met regel 5 in plaats van regel 10 moeten we regel 150 ook veranderen om de twee nieuwe regels telkens te laten verwerken.

Nu hebben we een goed, soepel verlopend programma. Ben je er helemaal tevreden mee? Of wil je het nog wat interessanter maken?... Waarom zou de UFO altijd in een rechte lijn moeten vliegen?... Je Sony Computer kan een heleboel kunstjes doen—alles wat je hem opdraagt.

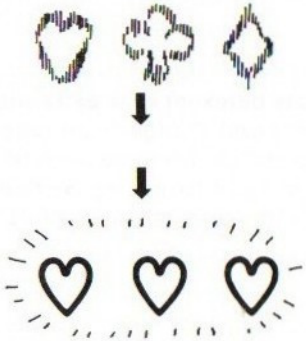
DE GOKAUTOMAAT



Hoe je ...

- een gokautomaat programmeert die...
  - —de aanduiding verandert tot je hem stopt
  - —de verschillende tekens vergelijkt
  - —de puntentelling bijhoudt aan de hand van je inzet
- een lijst-variabele gebruikt
- het ON-GOTO bevel gebruikt
- een rij-variabele gebruikt
- een programma met een druk op een toets stopt
- het formaat van de aanduiding instelt
- in een IF-THEN bevel verscheidene voorwaarden zet
- een lang programma in gedeelten weergeeft

Bij het juist raden van een getal in ons laatste programma werd je beloond met een voorbij vliegende UFO. Deze keer maken we van je Sony Computer een "gokautomaat" zoals in Las Vegas. Telkens wanneer je speelt, kan je winnen of verliezen! De computer kan niet met geld omgaan zoals de echte gokautomaten in Las Vegas dat doen, maar de puntentelling bijhouden gaat hem heel goed af.





Hier volgen de regels: eerst zal de computer je vragen hoeveel punten je wilt inzetten. Dan verandert hij razendsnel de symbolen in een "venster", tot je hem stopzet. Vervolgens laat hij je de uitslag zien: als alle drie symbolen gelijk zijn, dan win je drie maal het aantal dat je inzet. Als twee erven gelijk zijn, dan win je net zo veel als je ingezet hebt. Als echter alle drie symbolen verschillend zijn, dan verlies je twee maal je inzet. Je start met een **pot** van 100 punten, en als je dat aantal tot 300 weet uit te breiden, dan win je het spel. Maar als je puntental daalt tot je op 0 staat, dan heb je verloren en het spel is over.

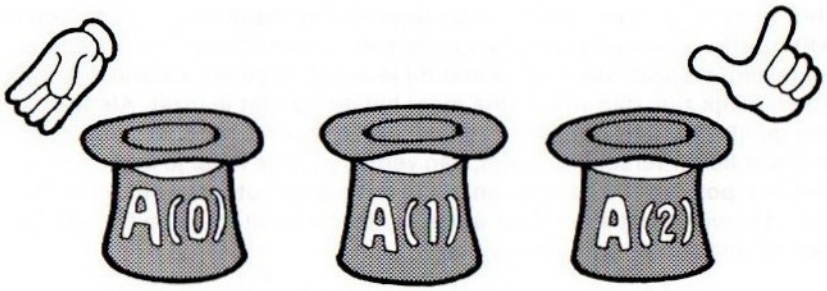
Klinkt dat goed? Laten we het gaan programmeren!

Het voornaamste onderdeel van dit spel wordt gevormd door de drie symbolen in het venster. Op elk van de drie posities verschijnen en verdwijnen vier verschillende symbolen (♥, ♠, ♦, ♣) in voortdurende opeenvolging—zo snel dat je niet werkelijk kunt zien welk symbool op een bepaald moment wordt aangegeven, je kunt alleen maar raden. Natuurlijk gebruikt ons programma voor de verschillende symbolen in het venster een aantal variabelen.

WAT IS EEN LIJST-VARIABELE?

We hebben nu al verscheidene malen bij het programmeren variabelen gebruikt, dus we begrijpen hoe een letter of een naam een wisselende waarde kan hebben. We hebben namen gebruikt als A, B en Q, en we hebben ze waarden gegeven die veranderden afhankelijk van het aantal malen dat een lus werd herhaald, of van de plaats waar we een UFO lieten vliegen, of geheel willekeurig.

Deze keer hebben we een variabele nodig die de gedaante kan aannemen van vier verschillende symbolen. Op de drie posities in het venster verschijnen in wisselende volgorde precies dezelfde symbolen, zodat we voor alle drie dezelfde variabele kunnen gebruiken. Maar elke afzonderlijke positie is onafhankelijk van de andere twee: soms zullen ze overeenkomen, en vaak niet. Om een dergelijk systeem te maken gebruiken we onze variabele op drie verschillende plaatsen—drie maal de zelfde variabele. Ze heten alle drie A, maar elk ervan heeft bovendien een nummer (tussen haakjes), zodat we weten welke variabele voor welke plaats gebruikt wordt.



A(0), A(1) en A(2) zijn **lijst-variabelen**. Is het wat verwarrend? Denk je dat het makkelijker zou zijn met drie verschillende namen? Je zult zo zien dat lijst-variabelen programmeren een stuk eenvoudiger maken, omdat ze bijzonder handelbaar en flexibel zijn.

Onze lijst van variabelen is drie variabelen **breed**, en zo **lang** als het aantal verschillende waarden dat we ze willen geven. Je kan ze zien als variabelen **met meer dimensies**, dat kan je misschien helpen bij het onthouden van het BASIC bevel voor lijst-variabelen: **DIM**. In dit programma schrijven wij dit bevel in de regel

```
10 DIM A(2)
```

om drie lijst-variabelen te maken, A(0), A(1) en A(2). (Andere voorbeelden van bevelen voor lijst-variabelen: **DIM A(10)**, waarmee we 11 variabelen maken van A(0) t/m A(10); en **DIM A(2), B(2)**, waarmee we zes variabelen maken—A(0) t/m A(2) en B(0) t/m B(2).)

Lijst-variabelen bieden nog een extra mogelijkheid: ook het getal tussen haakjes kan vervangen worden door een variabele, zodat je bijvoorbeeld A(I) krijgt, wat het programmeren sterk vereenvoudigt.

MAAK JE EIGEN GOKAUTOMAAT

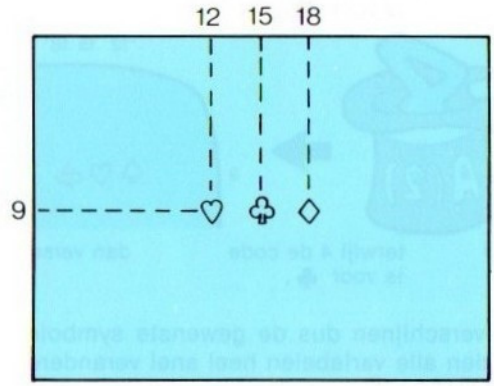
We gebruiken onze lijst-variabelen voor verschillende symbolen—harten (♥), schoppen (♠), ruiten (♦) en klaveren (♣). Maar eerst moeten we deze symbolen een cijfercode geven. We hebben al eerder een cijfercode gebruikt voor de verschillende kleuren, en dat was gemakkelijk omdat je Sony Computer die codes al kende. Deze keer moeten we echter onze eigen cijfercode uitwerken.

♥—1 ♠—2
♦—3 ♣—4

Zo weten we dat op een bepaalde plaats het ruiten-teken verschijnt wanneer variabele A(1) de waarde 3 aanneemt.

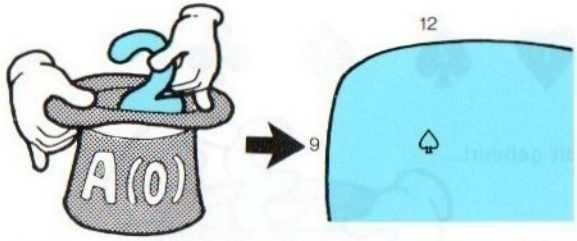


Vervolgens moeten we bepalen waar we het venster van de automaat op het scherm zetten. Middenin is een goede plaats.

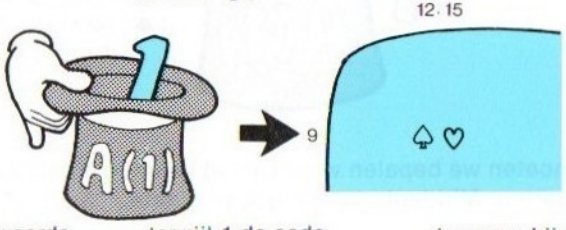


We zetten het eerste symbool, ingevuld door A(0), bij 12,9. Het tweede, van A(1), komt bij 15,9. Het derde, van A(2), bij 18,9.

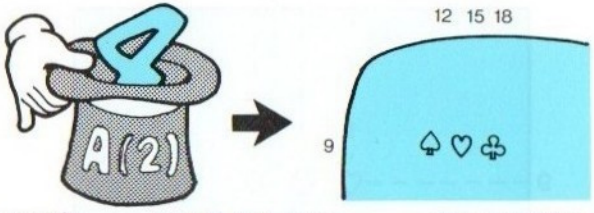
Hiermee is het plan voor het programma van een "gokautomaat" kompleet. Laten we het systeem dat we ontworpen hebben eens de revue laten passeren.



Als A (0) de waarde 2 aanneemt terwijl 2 de code is voor ♠, dan verschijnt ♠ bij 12,9.



Als A (1) de waarde 1 aanneemt, terwijl 1 de code is voor ♥, dan verschijnt ♥ bij 15,9.



Als A (2) de waarde 4 aanneemt, terwijl 4 de code is voor ♣, dan verschijnt ♣ bij 18,9.

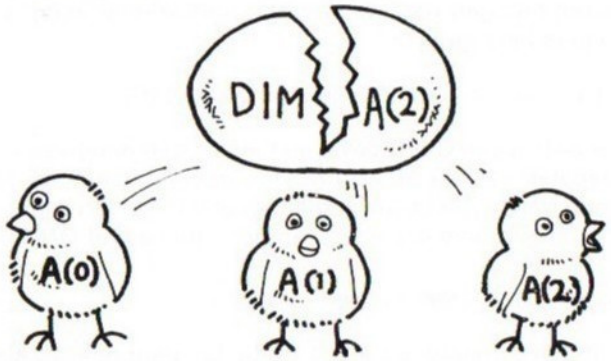
Met dit systeem verschijnen dus de gewenste symbolen op de juiste plaats. En aangezien alle variabelen heel snel veranderen—net zo snel als de stand van de UFO bij ons vorige programma—zal de opeenvolging van de symbolen zo snel verlopen dat je ze niet al te duidelijk kan zien.

We hebben een ontwerp voor het programma gemaakt, we begrijpen de opzet van het geheel, en we zijn klaar om de bevelen te geven.

```
10 DIM A(2)
20 CLS
30 FOR I=0 TO 2
40 A(I)=INT(RND(1)*4)+1
50 LOCATE I*3+12,9
60 ON A(I) GOTO 70,80,90,100
70 PRINT "♥":GOTO 110
80 PRINT "♠":GOTO 110
90 PRINT "♦":GOTO 110
100 PRINT "♣"
110 NEXT I
120 GOTO 30
```

Regel 10 maakt onze drie lijst-variabelen, zoals we al eerder zagen. Wanneer je lijst-variabelen wilt gebruiken, dan is het nodig om dit de computer bij het **begin** van het programma te laten weten.

```
10 DIM A(2)
```



Regel 20, inderdaat, wist het scherm. Regel 30 begint met FOR, en we weten nu dat een FOR bevel in het programma altijd samen gaat met een NEXT bevel ergens verderop. Hier loopt het FOR-NEXT gedeelte van regel 30 t/m regel 110, en alle bevelen in dit gedeelte worden met een lus herhaald.
Na FOR maakt regel 30 een nieuwe variabele genaamd I, die tussen de haakjes van onze lijst-variabelen komt te staan.

Komt regel 40 je bekend voor? Het is hetzelfde soort bevel voor **willekeurige getallen** als we al eerder in het raadselspelletje gebruikt hebben. Wanneer variabele I de **beginwaarde** 0 aanneemt, wordt regel 40

```
A(0)=INT(RND(1)*4)+1
```

Dit betekent dat A(0) elke waarde van 1 tot 4 kan aannemen, zie je wel? A(0) is de variabele voor de linker positie, en de cijfers 1-2-3-4 zijn code voor de symbolen ♥, ♠, ♦, ♣. Met andere woorden, wanneer I de waarde 0 aanneemt draagt regel 40 de computer op om een van de symbolen te kiezen voor de linker positie.

Nu zijn we op het punt dat we de computer moeten vertellen waar we de symbolen op het scherm willen plaatsen.

```
50 LOCATE I*3+12,9
```

Wanneer I de waarde 0 heeft, voor de linker positie, bevindt de cursor zich bij 12,9. Wanneer I de waarde 1 aanneemt, voor de middelste positie, wordt de plaats 15,9. En I wordt 2 voor de rechter positie, zodat de plaats gelijk wordt aan $2\times3+12,9$—ofwel 18,9.

Nu heeft de computer een symbool gekozen, en de cursor is naar de bijbehorende positie versprongen. Maar de computer zal het symbool niet op het scherm brengen voor we hem dat uitdrukkelijk zeggen. Hiervoor is regel 60 heel belangrijk.

```
60 ON A(I) GOTO 70,80,90,100
```

Wat we hier zien is een GOTO bevel met als bestemming vier verschillende regels tegelijk—70, 80, 90 en 100. Hoe weet de computer nu waar hij eerst naar toe moet? Die beslissing hangt af van het ON A(I) deel van dit bevel. Dit is de gebruikelijke vorm van het zogenaamd ON-GOTO bevel:

ON [iets] GOTO [regel 1], [regel 2], [regel 3], ...

Dit betekent, bijvoorbeeld: als iets 1 is, ga dan naar regel 1; als iets 2 is, ga dan naar regel 2; als iets 3 is, ga dan naar regel 3 ... Met andere woorden, de waarde van "iets" bepaalt automatisch naar welke regel de computer gestuurd wordt.



ON—GOTO regelt het verkeer

Laten we bijvoorbeeld zeggen dat A(0) de waarde 3 heeft (de code voor ♦). Regel 60 stuurt de computer door naar regel 90 wanneer de variabele deze waarde heeft.

```
90 PRINT "♦":GOTO 110
```

Wat gebeurt er als variabele A(0) de waarde 2 heeft? Dan stuurt regel 60 de computer door naar

```
80 PRINT "♠":GOTO 110
```

Zo zie je hoe de computer een van de vier symbolen kiest en dit op een van de drie posities afbeeldt.

Na elk van de PRINT bevelen gaat de computer door naar regel 110. (In regel 100 hebben we geen GOTO 110 bevel nodig omdat de computer hier zoals gewoonlijk automatisch naar de volgende regel gaat.)

Regel 110 wijzigt de waarde van I van 0 in 1. Vervolgens gaat de computer terug naar regel 40. En wat gebeurt er dan? Denk even na, en zie of je het begrijpt voordat we het uitleggen.

Oké. Wanneer I de waarde 1 heeft, wordt regel 50

```
LOCATE 1 * 3+12,9
```

wat wil zeggen LOCATE 15,9. Anders gezegd, de plaats waar het symbool zal verschijnen is verschoven van 12,9 naar 15,9—van de linker positie naar de middelste positie. Een van de vier symbolen, willekeurig gekozen, zal op het scherm verschijnen bij 15,9. Daarna wijzigt de computer variabele I van 1 in 2, en keert terug naar regel 40. En deze keer wordt de plaats 18,9—de rechter positie. De computer is dan het programma drie maal rond gegaan (en dat in veel minder dan een sekonde!).



Dat zou je op het scherm kunnen zien, maar in feite weten we nooit welke drie symbolen de computer op een bepaald moment willekeurig kiest.

De regels 30 t/m 120 vormen een lus. De computer gaat hierin keer op keer rond, en kiest telkens met behulp van de lijst-variabelen een symbool voor de linker positie, de middenpositie, de rechter positie, links, midden, rechts ... Aangezien het programma een eindloze lus vormt lijkt het net of de symbolen voorbij een venster lopen op de rand van een draaiende trommel, zoals het bij een echte gokautomaat werkt.

Als je het programma nog niet uitgeprobeerd had, voer het dan nu in je computer in. Kontroleer of je geen vergissingen gemaakt hebt, en geef het RUN bevel om het programma te verwerken. Je ziet dan dat we in onze opzet geslaagd zijn: we hebben met behulp van lijst-variabelen een gokautomaat gemaakt. Als we scherm 0 gebruiken wordt echter de rechterkant van elk symbool niet getoond.

Maar hoe stoppen we die eindloze lus nu zo dat we het puntental kunnen aflezen? En hoe weet de computer de puntentelling? Om werkelijk op onze gokautomaat te kunnen spelen moeten we echt nog wat meer bevelen in het programma zetten. Daarvoor gebruiken we weer een ander soort variabelen, waarin letters in plaats van getalswaarden gebruikt kunnen worden, en die we **rij-variabelen** zullen noemen. Maar dat vergt een aparte uitleg.

RIJ-VARIABELEN

We hebben geleerd en gezien dat een variabele de waarde van elk gewenst **getal** kan aannemen—of dat nu 1, 2, 3, 191 of 252 is. Variabelen kunnen echter **ook** de "waarde" van een **letter** krijgen—of van een aantal letters op een rijtje.

Laten we een paar eenvoudige oefeningen doen zodat we zien hoe een variabele de betekenis van een rij letters kan krijgen. Typ eerst **PRINT A$** en druk op RETURN. (Het $ teken bevindt zich boven de 4 op het toetsenbord, en je hebt er de SHIFT toets voor nodig.)

```
PRINT A$

Ok
■
```

Er gebeurde dus helemaal niets. Er kwam geen pieptoon of foutmelding, dus een vergissing is het niet, en de computer zette niets op het scherm. Voer nu het volgende bevel in:

```
A$="ABC"
PRINT A$
```

Druk op RETURN, en je krijgt dit op het scherm te zien:

```
A$="ABC"
Ok
PRINT A$
ABC
Ok
■
```

Zoals je ziet wordt A$ gebruikt als een variabele met de waarde "ABC". Vergelijk dit met een ander bevel:

```
A=345
Ok
PRINT A
 345
Ok
■
```

Dat ziet er bekend uit, nietwaar? Variabele A krijgt de getalswaarde 345 togewezen, en aan het eind van de naam van variabele A staat nu geen $ (dollar-teken).

Heb je de regel begrepen? **Als de naam van een variabele eindigt in een $ teken, heeft die variabele de waarde van een aantal letters (die we "rij" noemen). Zonder het $ teken aan het eind neemt de variabele de waarde van een getal aan.**



Nog een ding moet je onthouden: de waarde die je toekent moet tussen **aanhalingstekens** staan, als volgt:

```
A$="ABC"
```

Probeer de onderstaande voorbeelden uit, en het verschil tussen zogenaamde numerieke (getals-) variabelen en rijvariabelen zal je geheel duidelijk worden.

```
A=123                    A krijgt de waarde 123
Ok
B=234                    B krijgt de waarde 234
Ok
PRINT A+B                Zet A+B op het scherm
 357                     De som is 357
Ok
A$="123"                 A$ krijgt de waarde "123"
Ok
B$="234"                 B$ krijgt de waarde "234"
Ok
PRINT A$+B$              Zet A$+B$ op het scherm
123234                   Samen wordt dat "123234"
Ok
A$=987                   A$ krijgt de waarde 987
Type mismatch            Piep!! Niet mogelijk
Ok
■
```

Wanneer we invoeren A$="123", dan krijgt variabele A$ de waarde 123—niet het getal honderd-drie-en-twintig, maar de **rij tekens** een-twee-drie. En in de volgende regel krijgt B$ de waarde 234 (de cijfers twee-drie-vier). Waarom? Omdat we aanhalingstekens gebruiken, en het $ teken. Daaraan leest de computer af dat we een rijvariabele gebruiken, en dus leest hij 123 niet als een getal, maar als een rij tekens, net alsof het letters zijn.

Wanneer we de computer in dat geval de opdracht geven om ze samen te voegen, met A$+B$, dan kombineert de computer de beide rijtjes tot een enkele rij tekens, ongeacht of dit letters of cijfers zijn. Een soortgelijk voorbeeld zou dus "ABC"+"BCD"="ABCBCD" luiden.

Tenslotte voerden we A$=987 in. Hierop gaf de computer een foutmelding:

> Type mismatch (wat betekent: je typt twee dingen die niet samen gaan)

Zonder de aanhalingstekens is 987 een getal (negenhonderdzeven-entachtig), wat **niet samen gaat** met een rijvariabele met het $ teken.



Nu weten we dus: als de naam van een variabele eindigt in een $ teken, dan kan daaraan een letter of een rij letters worden toegewezen, en die waarde moet tussen "aanhalingstekens" staan. Dan is het nu weer tijd om terug te gaan naar ons programma met de gokautomaat.

ONDERBREEK EEN LUS MET INKEY

De FOR-NEXT lus die we in het radselspelletje gebruikten stopte vanzelf toen de variabele de laatste waarde bereikte. Maar regel 120 in dit programma—het bevel GOTO 30—betekent dat de lus in ons programma voor de gokautomaat nooit zal stoppen.



Natuurlijk kan je de computer altijd laten stoppen door op CTRL en STOP te drukken. Maar dat brengt het hele programma tot stilstand, en als je daarna het RUN bevel geeft om weer te starten, dan zit je nog steeds met een eindloze lus. Waar het ons nu om gaat is de gokautomaat te onderbreken, met de bedoeling om naar de symbolen te kijken, te zien of ze overeenkomen en hoeveel punten we winnen of verliezen. En daarna willen we het programma weer starten, zodat het spel vanaf hetzelfde punt vervolgd wordt.

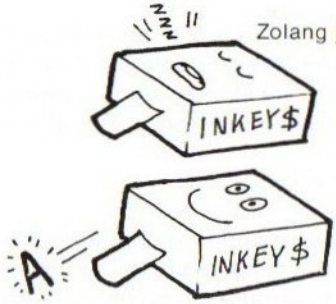
Een goede manier om de gokautomaat te stoppen is door het toevoegen aan het programma van de volgende twee regels:

```
103 K$=INKEY$
106 IF K$="A" THEN END
```

K$ is natuurlijk een rij-variabele. En in regel 106 is het duidelijk: zodra K$ de waarde A heeft, dan stopt (END) de gokautomaat de voorbij draaiende symbolen.

Wat betekent **INKEY$**? Het is een funktie in de BASIC taal. "KEY" betekent hier **toets**, en het hele woord dus "intoets"—wat wil zeggen dat je op dit punt elke gewenste toets kan indrukken. En dan zie je dat regel 103 luidt: **K$ krijgt de waarde van elke lettertoets die je indrukt**. Zolang je geen toets indrukt vervolgt de computer het programma, zonder K$ een waarde te geven. Ook als je B of C of X indrukt, laat de computer de gokautomaat gewoon doordraaien. Maar als je op de A toets (hoofdletter) drukt, treedt regel 106 in werking en die draagt de computer op om te stoppen.

Waar je overigens wel om moet denken is dat de computer alles wat tussen aanhalingstekens staat precies letterlijk neemt, zodat hij een hoofdletter A en een kleine letter a niet als dezelfde letter opvat. Als je dus bij het schrijven van het programma een kleine letter a gebruikt, zul je later de gokautomaat alleen maar kunnen stoppen met die zelfde kleine letter a. Als je het programma schrijft met hoofdletters, dan kun je voor het stoppen van de symbolen alleen een hoofdletter A gebruiken. Let dus bij het spelen met de gokautomaat op of het lampje van de CAP toets brandt.



Zolang je geen toets indrukt, doe ik niets

Zodra je A indrukt, geef ik A

Hiermee heeft onze lus met FOR-NEXT en GOTO 30, die van regel 30 t/m regel 120 loopt, een END bevel er middenin gekregen. Maar de computer zal dat bevel niet horen (en dus de gokautomaat niet stoppen), tenzij we hem dat uitdrukkelijk zeggen, door het indrukken van de A toets. Met andere woorden, de gokautomaat blijft de symbolen veranderen tot we hem stoppen.

```
30 FOR I=0 TO 2
        |
        |
        |
        |
        |
        |
103 K$=INKEY$
106 IF K$="A" THEN END
110 NEXT I
120 GOTO 30
```

De FOR—NEXT lus die op drie plaatsen een symbool afbeeldt

Het programma stopt zodra je op de A toets drukt

WAT IS DE STAND?

Wanneer de gokautomaat de symbolen niet meer verandert, is het tijd om te zien hoeveel punten we gewonnen (of verloren!) hebben. Daarvoor moeten we de regels in het programma zetten. Weet je de regels nog? We beginnen met een **pot** van 100 punten. Dan bepalen we de **inzet**—en we kunnen van 1 tot 100 punten inzetten. Laten we deze gegevens eerst maar in het programma zetten.

```
23 P=100:LOCATE 2,18:PRINT USING "POT
:####";P
26 LOCATE 0,20:INPUT "INZET";B
```

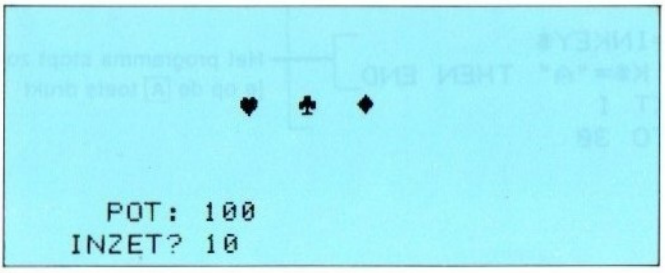
Kijk eens naar onze nieuwe regel 23. Onze pot zal elke keer dat we inzetten anders zijn, dus we gebruiken hiervoor een variabele, P. De inzet P heeft een beginwaarde van 100. Om de stand weer te geven gebruiken we het LOCATE bevel, gevolgd door een nieuw soort PRINT bevel:

```
PRINT USING "POT:# # # # ";P
```

Dit nieuwe PRINT bevel draagt de computer op om de informatie in een bepaald **formaat** weer te geven. Het # teken staat voor "cijfers". Dit bevel laat de computer weten: breng de variabele P op het scherm, en gebruik hiervoor (dit is wat USING betekent) de vier posities op het scherm naast "POT :". Het spel begint met een pot van 100, zodat de eerste aanduiding bij 2,18 als volgt zal luiden:

```
POT: ␣ 100
     ‾‾‾‾
   4 cijfers
```

Regel 26 draagt de computer op om je inzet vlak onder de pot op het scherm te zetten. Je zult je inzet ook vaak wijzigen, dus ook hier gebruiken we een variabele, B. Het INPUT bevel laat de computer wachten tot je een waarde voor B—ofwel je inzet—via het toetsenbord invoert. Als je de eerste keer 10 punten wilt inzetten en dus 10 intikt, verschijnt dit als volgt op het scherm:



Na regel 23 en 26 gaat de computer dan verder naar het FOR-NEXT gedeelte van het programma. We kunnen nu de nieuwe bevelen in de regels 23, 26, 103 en 106 in ons programma zetten.

```
10 DIM A(2)
20 CLS
23 P=100:LOCATE 2,18:PRINT USING "POT
:####";P
26 LOCATE 0,20:INPUT "INZET";B
30 FOR I=0 TO 2
40 A(I)=INT(RND(1)*4)+1
50 LOCATE I*3+12,9
60 ON A(I) GOTO 70,80,90,100
70 PRINT "♥":GOTO 103
80 PRINT "♠":GOTO 103
90 PRINT "♦":GOTO 103
100 PRINT "♣"
103 K$=INKEY$
106 IF K$="A" THEN END
110 NEXT I
120 GOTO 30
```

Invoegen

Wijzigen

Invoegen

Vergeet niet dat we het GOTO 110 bevel in regel 70, 80 en 90 moeten wijzigen in GOTO 103. Want wat zou er gebeuren als we dat vergaten? Juist.

Nu is de gokautomaat klaar om je inzet te ontvangen, maar we hebben de computer nog niet verteld hoe die je pot moet veranderen om de puntentelling volgens de regels bij te houden. Daarvoor zullen we een paar nieuwe bevelen aan het eind van het programma toevoegen, te beginnen met regel 130. Maar eerst kijken we nog eens naar regel 106.

```
106 IF K$="A" THEN END
```

Wanneer we door het indrukken van de [A] toets zorgen dat de gokautomaat de symbolen niet meer verandert, is het niet werkelijk de bedoeling het programma met END te beëindigen. In plaats daarvan willen we van de computer de stand te horen krijgen. En dus moeten we het bevel van regel 106 wijzigen in:

```
106 IF K$="A" THEN GOTO 130
```

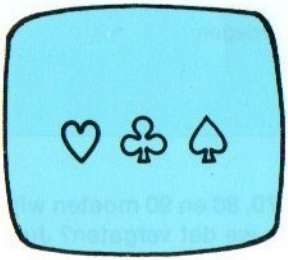
Aan de drie symbolen die op het scherm worden weergegeven zullen we gauw genoeg zien of we gewonnen of verloren hebben. Als ze alle drie gelijk zijn, dan winnen we drie maal onze inzet. (Als de pot 100 is, en we zetten 10 in, en we krijgen vervolgens drie gelijke symbolen op het scherm, dan wordt onze pot verhoogd tot 130.) Als twee van de symbolen gelijk zijn, dan winnen we een aantal punten dat gelijk is aan onze inzet (zodat de nieuwe pot 110 wordt). Als we echter drie verschillende symbolen zien, dan verliezen we twee maal onze inzet (en de nieuwe pot is dan 80). Stel bijvoorbeeld we zien op het scherm

♥ ♣ ♠

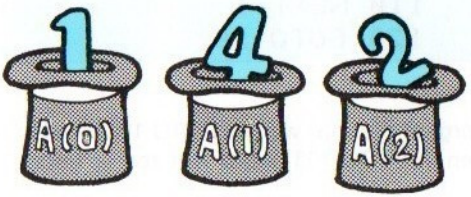
dan weten we dat we een aantal punten verliezen.

De computer kijkt echter niet naar de symbolen op het scherm. Hij gebruikt de variabelen in zijn geheugen. De computer weet direkt of A(0), A(1) en A(2) aan elkaar gelijk zijn door de waarden van die lijst-variabelen te vergelijken.

Dit zien wij



En dit heeft de computer in zijn geheugen



Als het scherm aangeeft:

♥ ♥ ♥

dan weet je ongetwijfeld wat er in het geheugen van de computer staat.

Zoals je ziet is het voor de computer heel eenvoudig om te bepalen of twee, drie of geen van de symbolen hetzelfde zijn. En dat betekent dat de bevelen die we in het programma moeten zetten voor het bijhouden van de puntentelling ook niet zo moeilijk zijn.

```
130 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 150
140 IF A(0)=A(1) OR A(1)=A(2) OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
150 LOCATE 6,18:PRINT USING "####";P
160 GOTO 26
```

Het eerste van deze bevelen voor de puntentelling komt in regel 130. Zodoende treedt het gedeelte van het programma dat voor de puntentelling zorgt alleen in werking nadat toets [A] is ingedrukt en de gokautomaat stopt.

```
130 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 150
```

Je ziet dat dit IF-THEN bevel werkt met **twee** voorwaarden, verbonden door **AND**. als aan **beide** voorwaarden voldaan wordt—dus als A(0)=A(1) en A(0)=A(2)—dan zijn alle drie lijst-variabelen gelijk, en ook de drie symbolen op het scherm. Indien (IF) dit waar is, dan

```
THEN P=P+B * 3:GOTO 150
```

Dat betekent dat je inzet, B, met 3 vermenigvuldigd wordt en het resultaat bij de pot, P, opgeteld; daarna gaat de computer verder naar regel 150. Als je bij een pot van 100 dus 10 ingezet hebt, dat is de nieuwe stand 130.

Als echter niet alle drie lijst-variabelen gelijk zijn, dan zal de computer het THEN bevel niet opvolgen. In plaats daarvan gaat hij dan naar de volgende regel van het programma.

```
140 IF A(0)=A(1) OR A(1)=A(2) OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
```

Hier zien we een IF bevel met maar liefst drie voorwaarden, en deze zijn verbonden door **OR**, het BASIC woord voor of. Dit betekent dat als een van de mogelijke paren variabelen gelijk zijn (ofwel als twee van de symbolen gelijk zijn),

```
THEN P=P+B
```

waarmee je inzet bij de pot wordt opgeteld, tot in ons voorbeeld een nieuwe stand van 110. Als echter aan geen van de drie voorwaarden voldaan wordt (als alle symbolen verschillend zijn), dan gaat de computer door naar

```
ELSE P=P-B * 2
```

En je verliest! Je inzet wordt verdubbeld en het resultaat van je pot afgetrokken—zodat de nieuwe stand maar 80 bedraagt.

Nu weet de computer de stand dus, en is de volgende stap het weergeven ervan.

```
150 LOCATE 6,18:PRINT USING "####";P
```

Dit is eenvoudig te begrijpen, want het is vrijwel gelijk aan het PRINT-formaat bevel van regel 23: eerst LOCATE, dan PRINT USING (met gebruik van) het formaat van de vier cijfers, en dan de naam van de variabele die een nieuwe waarde aanneemt. Telkens wanneer je de gokautomaat stopt, rekent de computer je nieuwe puntental uit en zet dit bij positie 6,18 op het scherm.

En wat gebeurt er dan? Het spel gaat door en je moet opnieuw inzetten. En dus stuurt regel 160 de computer terug naar regel 26, waar deze wacht tot je de inzet via het toetsenbord invoert.

Nu kun je de bevelen voor de puntentelling in het programma opnemen. Als je het goed doet, komt het er als volgt uit te zien:

```
10 DIM A(2)
20 CLS
23 P=100:LOCATE 2,18:PRINT USING "POT
:####";P
26 LOCATE 0,20:INPUT "INZET";B
30 FOR I=0 TO 2
40 A(I)=INT(RND(1)*4)+1
50 LOCATE I*3+12,9
60 ON A(I) GOTO 70,80,90,100
70 PRINT "♥":GOTO 103
80 PRINT "♠":GOTO 103
90 PRINT "♦":GOTO 103
100 PRINT "♣"
103 K$=INKEY$
106 IF K$="A" THEN GOTO 130      <-- Wijzigen
110 NEXT I
120 GOTO 30
130 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 150
140 IF A(0)=A(1) OR A(1)=A(2) OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
150 LOCATE 6,18:PRINT USING "####";P
160 GOTO 26
```

Toevoegen

Wacht eens even! Dit programma is te lang. Het past niet meer op het scherm. Het eerste deel verdwijnt als je het LIST bevel geeft. Om dit probleem op te lossen is het nodig de computer te vertellen **wat** hij met LIST op het scherm moet zetten, op deze manier:

```
LIST 10-100
```

Met dit bevel verschijnen alleen regel 10 t/m 100 op het scherm. Om dan alleen het laatste gedeelte te zien kun je bijvoorbeeld intikken

```
LIST 110-
```

Gebruik nu deze LIST bevelen om het programma grondig te kontroleren op eventuele vergissingen. Als je er zeker van bent dat het programma geheel in orde is, kun je het RUN bevel geven—en heb dan maar veel plezier met je nieuwe gokautomaat!

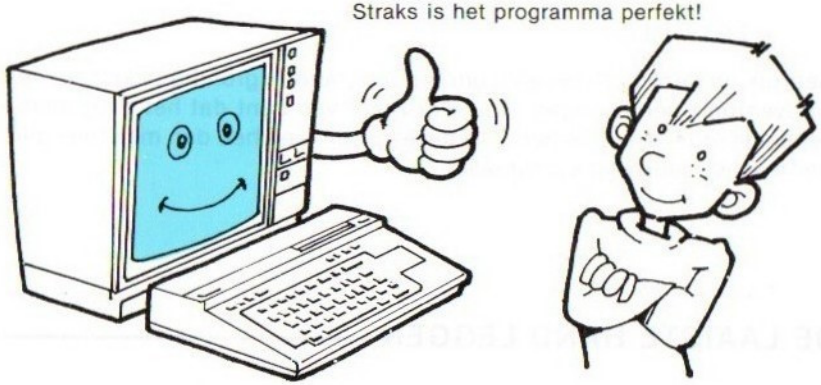
DE LAATSTE HAND LEGGEN

Je Sony Computer is nu een gokautomaat geworden die heel goed loopt. Hij doet een aantal ingewikkelde kunstjes: vraagt je om de inzet, verandert de symbolen op drie plaatsen, en berekent het puntental wanneer je de symbolen stopt. Het programma dat we hiervoor gemaakt hebben heeft maar 20 regels met bevelen—echt niet veel wanneer je bedenkt dat de computer **telkens weer** opgedragen moet worden om iets op het scherm te zetten, te berekenen, stoppen, wachten op een inzet, enzovoort.

Er ontbreekt echter een ding. In het begin zeiden we dat je het spel zou winnen wanneer je 300 punten bereikte, en dat je zou verliezen wanneer je puntental 0 bedroeg. Nu heb je waarschijnlijk al op je gokautomaat gespeeld tot je de 300 passeerde, of tot je onder 0 uitkwam, of misschien wel allebei.

Laten we ons goede programma verbeteren en er een prima programma van maken, door bevelen voor winnen en verliezen in te bouwen. Bovendien zijn er nog een paar verfijningen aan te brengen zodat we een **perfekt** programma krijgen.

- We kunnen het spel wat beter in de hand houden door de mogelijkheid voor een andere, verschillende inzet pas te bieden nadat je eerst op het toetsenbord de spatiebalk indrukt.
- Verderop in het programma is de spatiebalk ook wat handiger dan de [A] toets om de symbolen te stoppen, een letterlijk meer voor de hand liggende keus.
- Telkens wanneer we opnieuw inzetten staat de vorige inzet nog op het scherm. Dit is wat verwarrend.



Hier volgen de wijzigingen die we in het programma moeten aanbrengen.

```
10 DIM A(2)
20 CLS
23 P=100:LOCATE 2,18:PRINT USING "POT
:####";P
24 LOCATE 6,20:PRINT "          "      <-- Invoegen
26 LOCATE 0,20:INPUT "INZET";B
30 FOR I=0 TO 2
40 A(I)=INT(RND(1)*4)+1
50 LOCATE I*3+12,9
60 ON A(I) GOTO 70,80,90,100
70 PRINT "♥":GOTO 103
80 PRINT "♠":GOTO 103
90 PRINT "♦":GOTO 103
100 PRINT "♣"
103 K$=INKEY$
106 IF K$=" " THEN GOTO 130
110 NEXT I      <-- Wijzigen
120 GOTO 30
```

```
130 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 150
140 IF A(0)=A(1) OR A(1)=A(2) OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
150 LOCATE 6,18:PRINT USING "####";P
152 GOTO 170
154 K$=INKEY$
156 IF K$=" " THEN GOTO 24
160 GOTO 154
170 IF P<300 AND P>0 THEN GOTO 154
180 IF P>=300 THEN LOCATE 3,5:PRINT "
GEWONNEN!"
190 IF P<=0 THEN LOCATE 3,5:PRINT "VE
RLOREN!"
200 END
```

Invoegen

Wijzigen

Invoegen

Kan je het allemaal volgen? Kijk eerst een paar minuten om te zien of je de wijzigingen uit jezelf begrijpt, voordat je de uitleg leest.

Regel 170 is de eerste van een stel bevelen voor winnen of verliezen. Als het puntental minder dan 300 is èn (AND) meer dan 0, dan gaat het spel door en vervolgt de computer het programma. Als het puntental 300 of meer bedraagt, is het spel over en komt bovenaan het scherm GEWONNEN! te staan. Als het puntental 0 is, of onder 0 komt—VERLOREN! En als je gewonnen of verloren hebt gaat de computer naar regel 200 en is het spel uit (END).

Regel 24 wist bij het begin van elke ronde van het spel de vorige inzet. De bevelen verplaatsen de cursor naar de plaats waar de inzet is weergegeven, en laten op die plaats zes spaties verschijnen.

Regel 106 is een heel eenvoudige wijziging. Die zorgt dat we de gokautomaat stoppen door het indrukken van de spatiebalk in plaats van met de A toets. Misschien zie je het niet meteen, maar voor je computer is een **spatie tussen aanhalingstekens (" ")** ook een **letterteken**.

De regels 160, 154 en 156 betekenen samen dat het spel weer begint wanneer we de spatiebalk indrukken.

Dat waren de vier verbeteringen die we wilden aanbrengen: winnen/verliezen, en drie kleine verfijningen.

Regel 152 verandert het spel in feite niet, maar hij is noodzakelijk vanwege de andere wijzigingen in het programma. Deze regel komt vlak nadat het puntental wordt aangegeven, en hij stuurt de computer door naar de nieuwe regels voor winnen en verliezen, te beginnen met regel 170. Als je nog niet gewonnen of verloren hebt stuurt die regel 170 op zijn beurt de computer terug naar regel 154, en gaat het spel verder nadat je de spatiebalk indrukt.

MAAK HET PROGRAMMA BETER LEESBAAR

Toen we begonnen het Gokautomaat-programma te schrijven, gebruikten we voor de regelnummers tientallen: 10, 20, 30, 40 ... Later, toen we het programma steeds kompleter maakten, hebben we een heel aantal van de "open" regels gebruikt om nieuwe bevelen in te voegen. Als gevolg hiervan zitten we nu met regelnummers in ongebruikelijke groepen, zoals 90, 100, 103, 106, 110, 120. Om het programma wat beter leesbaar te maken kunnen we de regels **opnieuw nummeren**.

Voer het volgende bevel in:

```
RENUM
```

De regels blijven in dezelfde volgorde staan, maar alle regelnummers zijn nu veelvouden van tien, zoals je kunt zien na het geven van het LIST bevel:

Nu zijn de regelnummers veel gemakkelijker te volgen. Maar wat is er gebeurt met onze GOTO bevelen, waarin immers naar bestaande regelnummers verwezen wordt? Geen probleem! Kijk bijvoorbeeld maar naar regel 100 van het nieuwe programma:

```
100 PRINT "♥":GOTO 140
```

Dit GOTO bevel—GOTO 140—luidde GOTO 103 voor je het RENUM bevel gaf. Nu is regel 103 in het nieuw genummerde programma regel 140 geworden, en GOTO 103 is mee veranderd in GOTO 140. Je Sony Computer is echt heel slim, vooral waar het getallen betreft, en wanneer je het Gokautomaat-programma weer met RUN gaat verwerken zal je zien dat er in het spel niets veranderd is.

ZET TITELS IN JE PROGRAMMA

Stel dat je het programma aan een vriend wilt laten zien. Of stel dat je zelf een programma weer eens wilt lezen, verscheidene maanden nadat je het geschreven hebt. Als er dan 20 of 30 regels BASIC op het scherm verschijnen, kan het best moeilijk zijn om uit alle funkties van de regels wijs te worden, want de lezer is niet vertrouwd met het programma. Om dit probleem op te lossen biedt BASIC je de mogelijkheid om "opmerkingen" middenin je programma te zetten, als volgt:

```
5 REM *** DE GOKAUTOMAAT ***
7 REM
10 DIM A(2)
20 CLS
30 P=100:LOCATE 2,18:PRINT USING "POT
:####";P
40 LOCATE 6,20:PRINT "       "
50 LOCATE 0,20:INPUT "INZET";B
55 '
60 ' *** ZET AUTOMAAT IN WERKING ***
62 '
65 FOR I=0 TO 2          — Was regel 60
70 A(I)=INT(RND(1)*4)+1
80 LOCATE I*3+12,9
90 ON A(I) GOTO 100,110,120,130
100 PRINT "♥":GOTO 140
110 PRINT "♠":GOTO 140
120 PRINT "♦":GOTO 140
130 PRINT "♣"
140 K$=INKEY$
150 IF K$=" " THEN GOTO 180
160 NEXT I
170 GOTO 60
175 '
180 ' *** BEREKEN PUNTENTAL ***
182 '
185 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 200          — Was regel 180
190 IF A(0)=A(1) OR A(1)=A(2) OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
200 LOCATE 6,18:PRINT USING "####";P
```

```
210 GOTO 250
220 K$=INKEY$
230 IF K$=" " THEN GOTO 40
240 GOTO 220
245 '
250 ' *** WINNEN OF VERLIEZEN ***
252 '                                  Was regel 250
255 IF P<300 AND P>0 THEN GOTO 220
260 IF P>=300 THEN LOCATE 3,5:PRINT "
GEWONNEN!"
270 IF P<=0 THEN LOCATE 3,5:PRINT "VE
RLOREN!"
280 END
```

Onze bovenstaande "opmerkingen" zijn de **titel** van het programma en de **subtitels** van de verschillende onderdelen. Ze maken het programma wat langer, maar veel gemakkelijker te doorzien en te doordenken. En ze hebben geen enkele invloed op het spel zelf, want de computer leest deze nieuwe regels niet—ze staan er alleen voor het gemak van de mensen die het programma gebruiken.

Laten we eens een paar van de nieuwe regels met opmerkingen onder de loep nemen:

```
5 REM *** DE GOKAUTOMAAT ***
7 REM
```

Het REM bevel draagt de computer op alle letters en cijfers die na het bevel zelf op dezelfde regel staan te negéren. Je kan na REM intikken wat je maar wilt, het zal in het programma niets uitmaken. In regel 5 zien we de titel van het programma. Regel 7 is echter op het REM bevel na leeg. Op die manier kreëren we ruimte rondom de titel—net als op de titelpagina van een boek. De ruimte en ook de sterren voor en na de titel maken het makkelijker om de titel tussen al die regels BASIC terug te vinden.

Kijk nu eens naar regel 55, 60 en 62. Je ziet daar een kortere manier om het REM bevel te geven: met de ' (apostrof ofwel enkel aanhalingsteken). Natuurlijk zijn deze regels niet werkelijk deel van het programma. De computer leest ze niet als bevelen of invoer, omdat ze met ' beginnen. De subtitels, de sterren en de open regels zijn bedoeld voor mensen, niet voor de computer.

De computer ziet **wel** de regelnummers, want die komen voor de ' (of REM). In regel 150 zien we bijvoorbeeld:

```
150 IF K$=" " THEN GOTO 180
```

Maar regel 180 is slechts een opmerking:

```
180 ' *** BEREKEN PUNTENTAL ***
```

De computer vindt geen bevel om uit te voeren in regel 180 en gaat daarom door naar de volgende regel. Regel 182 bevat ook geen bevel, dus de computer gaat naar 185. Maar de **lezer**—jij of je vriend—gaat van regel 150 naar regel 180, en ziet daar onmiddellijk dat dit gedeelte van het programma dient om het puntental te berekenen.

JE BENT EEN HEEL EIND OP WEG!

Gefeliciteerd! Na alle oefeningen en spelletjes in dit boek heb je nu een praktische kennis van een nieuwe taal—BASIC. Misschien was het maar een paar uur, of een of twee dagen geleden dat je begon met de eenvoudigste bevelen, zoals PRINT 3+5. En nu begrijp je al ingewikkelde uitdrukkingen als A(I)=INT(RND)(1) * 4)+1 en PRINT USING "# # # #";P, en je kunt die ook **gebruiken**. Je Sony Computer zal je van nu af jarenlang goed van pas komen en je veel plezier en ontwikkeling bieden, nu je geleerd hebt "zijn taal te spreken".

Als je doorgaat met het schrijven en gebruiken van programma's zul je steeds meer handigheid krijgen in het omgaan met getallen, grafische voorstellingen, bevelen, funkties en spelletjes. 't Is niet meer dan een kwestie van tijd voor je een volleerd **expert** bent, en die tijd zal snel genoeg voorbijgaan terwijl je het plezier van je Sony Computer deelt met je familie en vrienden.

In het volgende hoofdstuk van dit boek kun je meer oefening opdoen met de bevelen die je kent, en je kunt er bovendien nog een paar bij leren. Daarna kun je met het **MSX-BASIC Programmeer-Handboek** doorgaan met het uitproberen van nog veel meer nieuwe BASIC bevelen, om uit te vinden waar die voor dienen. En je kunt volop experimenteren met je computer, om nieuwe kleuren, grafische voorstellingen, geluiden en spelletjes te ontwerpen.

Voor het maken van **je eigen programma** is het goed, eerst een paar minuten te besteden aan het uitwerken van een plan. Je zult misschien een stroomschema willen maken. Wanneer je dan weet welke **funkties** en welk soort **bevelen** je nodig hebt, kun je de Index achterin dit boek gebruiken om ze op te zoeken en de uitleg nog eens door te nemen. Werk zorgvuldig, schrijf een aantal bevelen en probeer ze uit. Je Sony Computer kan je op twee manieren behulpzaam zijn: Hij doet altijd **precies** wat je hem opdraagt; en hij vergeet eventuele vergissingen altijd onmiddellijk als je het NEW bevel geeft.

Praat met je vrienden over je programma's. Laat ze zien wat je met je computer kan doen. Hun ideeën zullen waarschijnlijk wat anders zijn dan de jouwe, en samen kun je meer plezier hebben. Als je vrienden zelf ook een computer hebben kun je niet alleen ideeën maar ook programma's uitwisselen, met behulp van een cassetterecorder of gewoon door je programma op papier te zetten.

Een ding om te onthouden: de beste manier om plezier te beleven met je Sony Computer is om nieuwe dingen uit te proberen en te zien wat er gebeurt. Je kunt met de BASIC taal je computer een onbegrensd aantal mogelijke opdrachten geven. Ben je klaar voor de avontuurlijke reis door het rijk van je eigen fantasie? Veel succes!

OEFENINGEN VOOR BASIC BEVELEN

PRINT

```
10 PRINT "JAN"
20 PRINT "EN"
30 PRINT "ELLY"
RUN
JAN
EN
ELLY
```

Herinner je je dit eenvoudige programma nog? Laten we het een beetje veranderen.

```
10 PRINT "JAN";
20 PRINT "EN";
30 PRINT "ELLY"
RUN
JANENELLY
```

Toevoegen

Met de ; (puntkomma) na "JAN" en na "EN" geeft de computer de drie woorden samen weer, op een regel. In feite kun je alle bevelen samen op slechts een regel van het programma zetten:

```
10 PRINT "JAN";"EN";"ELLY"
RUN
JANENELLY
```

Belangrijk

Maar drie woorden zonder tussenruimte zijn niet makkelijk te lezen, dus proberen we het anders.

```
10 PRINT "JAN",
20 PRINT "EN",
30 PRINT "ELLY"
RUN
JAN           EN
ELLY
```

Belangrijk

14 letters breed

Als we een komma gebruiken in plaats van een puntkomma komen de eerste letters van elk woord 14 stapjes uit elkaar te staan. Hetzelfde gebeurt als je het bevel met een enkele regel invoert.

```
10 PRINT "JAN","EN","ELLY"
RUN
JAN            EN
ELLY
```

14 letters breed

Belangrijk

Probeer nu dit bevel eens, met een enkele spatie tussen het eind van elk woord en het tweede aanhalingsteken.

```
10 PRINT "JAN ";"EN ";"ELLY"
RUN
JAN EN ELLY
```

Onthoud goed: de **spatie** is een belangrijk teken.

Laten we nu nog eens kijken op welke manieren cijfers op het scherm gezet kunnen worden.

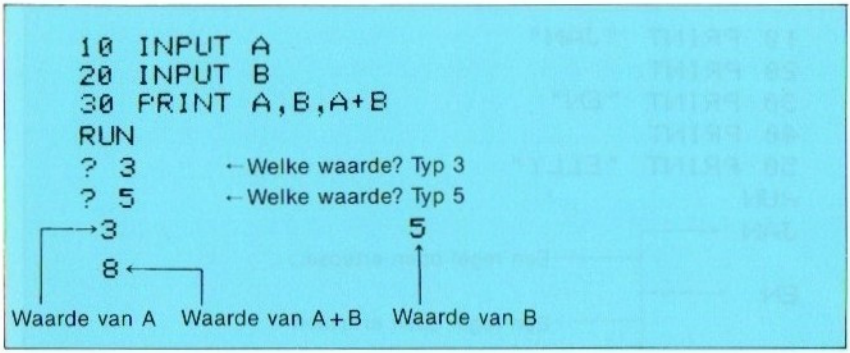
```
10 PRINT "3+5=";
20 PRINT 3+5
RUN
3+5= 8
```

← Geef de cijfertekens weer

← Maak de som en geef het resultaat weer

Regel 10 is een bevel om tekens weer te geven, en regel 20 is een bevel om een berekening uit te voeren. Maar deze twee verschillende funkties worden verbonden door de ; aan het eind van regel 10. Daarmee zeggen we de computer het geheel op een regel te zetten, net zoals we de som normaal op papier zouden schrijven. (In dit antwoord tref je voor de 8 een spatie aan. Die ruimte is bedoeld voor het weergeven van het minteken (−) als het antwoord een **negatief getal**, ofwel onder nul is—bijvoorbeeld 3−5= −2.

Laten we het INPUT bevel en het PRINT bevel eens tegelijk in een eenvoudig programma gebruiken.



Het INPUT bevel draagt de computer op je te **vragen** welke waarde aan de variabele toegekend moet worden, en te wachten tot je via het toetsenbord een waarde **invoert**. Pas nadat je op RETURN gedrukt hebt gaat de computer naar het volgende bevel. Regel 30 zet de drie waarden op het scherm.

Hier volgt een andere, duidelijker manier om dezelfde funkties te vervullen.

```
10 INPUT "A=";A
20 INPUT "B=";B
30 C=A+B
40 PRINT "A+B=";C
RUN
A=? 3
B=? 5
A+B= 8   ←Resultaat van regel 40
```

Regel 10 en 20 dienen om de computer een bepaalde vraag te laten weergeven, in plaats van alleen het ? (vraagteken). Regel 30 maakt een nieuwe variabele C, en geeft deze de waarde van A+B. Regel 40 draagt de computer op het antwoord en het probleem op het scherm te zetten, te beginnen met A+B=. De drie regels die nu na RUN op het scherm verschijnen zijn heel wat gemakkelijker te begrijpen dan de drie kale cijfers aan het eind van het laatste voorbeeld.

Soms is uitvoer op het scherm gemakkelijker te lezen wanneer er tussen woorden of tussen groepen woorden **regels open** worden gehouden.

```
10 PRINT "JAN"
20 PRINT
30 PRINT "EN"
40 PRINT
50 PRINT "ELLY"
RUN
JAN
                    Een regel open ertussen
EN
                    Een regel open ertussen
ELLY
```

Als op een regel alleen het PRINT bevel wordt ingevoerd (zoals in regels 20 en 40) dan wordt in de uitvoer een regel open gehouden.

INPUT

```
10 INPUT "A is ";A
20 INPUT "B is ";B
30 PRINT "A+B=";A+B
40 PRINT "A-B=";A-B
RUN
A is ? 15
B is ? 3
A+B= 18
A-B= 12
```

Hier hebben we nog een programma waarin het INPUT bevel en het PRINT bevel gekombineerd worden. Je ziet in een oogopslag dat de computer hier wordt opgedragen om waarden voor A en B te **vragen**, en twee berekeningen op het scherm **weer te geven**. Laten we nu de twee INPUT bevelen samenvoegen en nog een paar berekeningen toevoegen.

```
10 INPUT "A en B zijn ";A,B
20 PRINT "A=";A,"B=";B
30 PRINT
40 PRINT "A*B=";A*B
50 PRINT "A/B=";A/B
RUN
A en B zijn ? 15,3
A= 15          B= 3

A*B= 45
A/B= 5
```

In regel 10 kun je twee waarden voor de twee verschillende variabelen invoeren, maar de , (komma) tussen de twee waarden (in dit geval 15 en 3) is **zeer** belangrijk. Als de computer 153 zou weergeven in plaats van 15,3 zou dit niet erg begrijpelijk zijn. (Het PRINT bevel in regel 20 bevat ook belangrijke leestekens. Weet je wat de , en de ; betekenen?)

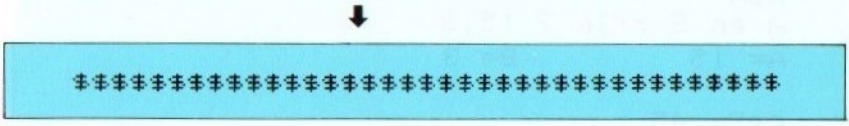
Nu gaan we een **rij-variabele** aan ons INPUT bevel toevoegen.

```
10 INPUT "Naam";N$
20 PRINT N$;" is geweldig!"
RUN
Naam? Jan
Jan is geweldig!
```

Wanneer de naam van een variabele eindigt in een $ teken, dan "neemt die de waarde aan" van een letter of een woord. Hier hebben we een programma dat zowel voor een rijtje letters als voor een getal variabelen gebruikt.

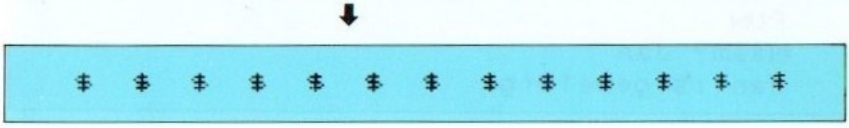
```
10 INPUT "Naam";N$
20 INPUT "Leeftijd";Y
30 PRINT N$;" is ";Y;" jaar oud."
RUN
Naam? Jan
Leeftijd? 10
Jan is 10 jaar oud.
```

```
10 CLS
20 FOR I=0 TO 36
30 LOCATE I,10
40 PRINT "$"
50 NEXT I
```



In dit programma laat de herhaalfunktie (**lus**) van het FOR-NEXT bevel de computer $ tekens op het scherm zetten van positie 0,10 tot positie 28,10. Hier volgt het zelfde programma met een nieuwe aanwijzing er aan toegevoegd.

```
10 CLS
20 FOR I=0 TO 36 STEP 3
30 LOCATE I,10
40 PRINT "$"
50 NEXT I
```



Wat is de betekenis van **STEP 3** in regel 20? Zoals je ziet is de cursor na elk $ teken **drie posities** verplaatst. Met andere woorden, de waarde van I verandert nu in stappen van 3 tegelijk, zodat het $ teken verschijnt bij positie 0,10 en bij 3,10 en bij 6,10 ... —in plaats van bij positie 0,10 en 1,10 en 2,10 ... De $ tekens vullen bijna dezelfde breedte van het scherm als bij het programma hierboven, van 0,10 tot **36**,10.

Door het toevoegen van **de aanwijzing STEP en een getal** kun je dus **een variabele in stappen doen veranderen**. In het laatste voorbeeld gebruikten we de variabele voor de positie, en dus veranderde de plaats in stappen. Het volgende programma gebruikt de STEP aanwijzing voor het veranderen van de manier waarop de computer telt.

```
10 FOR I=50 TO 0 STEP -5
20 PRINT I,
30 NEXT I
RUN
 50             45
 40             35
 30             25
 20             15
 10             5
 0
```

I=50 TO 0 laat de computer tellen zoals in elke FOR-NEXT lus. Er zijn 51 waarden voor I van 50 t/m 0. Maar de **STEP −5** aanwijzing zorgt dat de waarde vermindert in stappen van −5, en dus zijn er slechts 11 waarden voor I.

FOR—FOR—NEXT—NEXT: Een lus binnen een lus

```
10 FOR I=0 TO 2
20 PRINT "I=";I
30 FOR J=0 TO 4
40 PRINT "J=";J;
50 NEXT J
60 PRINT
70 NEXT I
RUN
I= 0
J= 0  J= 1  J= 2  J= 3  J= 4
I= 1
J= 0  J= 1  J= 2  J= 3  J= 4
I= 2
J= 0  J= 1  J= 2  J= 3  J= 4
```

FOR ⟆ NEXT

FOR ⟆ NEXT

Dit programma laat zien hoe we een FOR-NEXT lus binnen een andere, ruimere FOR-NEXT lus kunnen plaatsen. Variabele I wordt geregeld door de eerste en de laatste regel van het programma, en variabele J wordt geregeld door regel 30 en regel 50. Het resultaat is dat voor elke waarde van I de computer vijf kleinere lussen doorloopt om alle waarden voor J te vinden. I verandert van 0 tot 2, en voor elke I verandert J van 0 tot 4.

Hier is nog een manier om een lus te gebruiken.

```
10 INPUT N,S
20 CLS
30 FOR I=0 TO N STEP S
40 LOCATE 5,10:PRINT "I=";
50 PRINT USING "####";I
60 FOR J=0 TO 500
70 NEXT J
80 NEXT I
```

Dit programma gebruikt de bevelen INPUT, FOR-NEXT en STEP om variabele I in een lus te laten veranderen van 0 t/m N, in stappen van S. Natuurlijk zullen we bij verwerken van het programma de variabelen N en S via het toetsenbord waarden toekennen.

Maar wat is de funktie van variabele J en de bijbehorende korte lus in regel 60 en 70? J verschijnt zelfs nooit op het scherm. Deze variabele wordt nergens voor gebruikt, maar toch zal de computer, voor hij doorgaat met de berekening en de waarde van I op het scherm zet, eerst 501 waarden voor J vinden. Dit noemen we een **loze** lus. Het enige waar deze voor dient is het invoegen van een kort oponthoud—net zolang als de computer nodig heeft om 501 FOR-NEXT lussen te doorlopen—verder niets.

De funktie van de J lus is dus het invoegen van een **tijdsinterval** in de grotere I lus. Bij het verwerken van dit programma zul je zien dat de computer telkens na het weergeven van een waarde van I een pauze neemt. Zo zie je hoe snel je Sony Computer 501 eenvoudige bewerkingen kan uitvoeren (nogal snel!) en hoe je een loze lus kan gebruiken om in een programma wat extra tijd in te voegen. Kan je een loze lus schrijven die de computer wat langer bezighoudt?

IF—THEN en IF—THEN—ELSE

Hier is weer eens een raadseltje, waarbij je het getal dat de computer kiest moet zien te raden. Maar dit spelletje verschilt wat van het raadseltje en ook het gokautomaatspel dat we eerder in dit boek zagen. Dat waren spelen gebaseerd op puur geluk, en deze keer komt er wat vaardigheid bij te pas om het juiste getal zo snel mogelijk te vinden. Als je het slim doet, moet je het getal in zes of zeven pogingen kunnen raden. Als je maar willekeurig wat raadt, kan het veel langer duren.

```
10 X=INT(RND(1)*100)+1
20 INPUT "Noem een getal";A
30 IF A=X THEN GOTO 70
40 IF A>X THEN PRINT "KLEINER"
50 IF A<X THEN PRINT "GROTER"
60 GOTO 20
70 PRINT "Raak!"
```

Regels 30, 40 en 50 laten de computer het getal dat jij raadt vergelijken met de waarde van X, om je dan voor de volgende poging een hint te geven.

Zie je de wijziging die we in het volgende programma hebben aangebracht?

```
10 X=INT(RND(1)*100)+1
20 INPUT "Noem een getal";A
30 IF A=X THEN 70
40 IF A>X THEN PRINT "KLEINER"
50 IF A<X THEN PRINT "GROTER"
60 GOTO 20
70 PRINT "Raak!"
```

In feite is dit programma gelijk aan het vorige, ook al is regel 30 gewijzigd. IF—THEN heeft de zelfde betekenis als IF—THEN—GOTO. Het programma zou zelfs nog hetzelfde blijven als we er 30 IF A=X GOTO 70 van maakten. Anders gezegd, IF—GOTO betekent hetzelfde als IF—THEN—GOTO. Je kan een van beide bevelen, THEN of GOTO, zonder problemen weglaten (echter niet allebei!).

Nu gaan we twee regels van dit programma samenvoegen tot een enkele regel:

```
10 X=INT(RND(1)*100)+1
20 INPUT "Noem een getal";A
30 IF A=X THEN 70 ELSE
   IF A>X THEN PRINT "KLEINER"
50 IF A<X THEN PRINT "GROTER"
60 GOTO 20
70 PRINT "Raak!"
```

De twee IF—THEN bevelen van regel 30 en 40 vormen nu samen een IF—THEN—ELSE bevel. Er is geen regel 40 meer, maar de betekenis blijft gelijk. Als dat mogelijk is, dan moeten we ook regel 30 en regel 50 van dit programma tot een regel samen kunnen voegen.

```
10 X=INT(RND(1)*100)+1
20 INPUT "Noem een getal";A
30 IF A=X THEN 70 ELSE
   IF A>X THEN PRINT "KLEINER"
   ELSE PRINT "GROTER"
60 GOTO 20
70 PRINT "Raak!"
```

Je kunt kontroleren of deze vier programma's inderdaad gelijk zijn door ze een voor een op je computer uit te proberen.

DIM (Lijst-variabelen)

Toen we in het Gokautomaat-programma lijst-variabelen gebruikten, hadden we het bevel

DIM A(2)

waarmee we drie variabelen maakten:

A(0) A(1) A(2)

Vervolgens vervingen we de cijfers tussen haakjes door een variabele, (I), en lieten de computer weten dat I=0 TO 2. We hadden nog steeds de zelfde drie lijst-variabelen, en de computer gebruikte ze een voor een in een FOR-NEXT lus.

Lijst-variabelen kunnen ook meer dan een getal tussen de haakjes hebben. Als we bijvoorbeeld intikken:

DIM A(3,4)

dan hebben we daarmee 20 variabelen:

A (Ø, Ø)	A (1, Ø)	A (2, Ø)	A (3, Ø)
A (Ø, 1)	A (1, 1)	A (2, 1)	A (3, 1)
A (Ø, 2)	A (1, 2)	A (2, 2)	A (3, 2)
A (Ø, 3)	A (1, 3)	A (2, 3)	A (3, 3)
A (Ø, 4)	A (1, 4)	A (2, 4)	A (3, 4)

Begrijp je hoe we aan 20 verschillende variabelen komen? Het eerste getal tussen haakjes kan vier verschillende waarden aannemen, en het tweede getal vijf waarden—4 maal 5 is 20. (Hoeveel variabelen zouden we krijgen als we DIM A(9,9) schrijven?)

We lieten je nu die 20 variabelen niet in een eenvoudige lijst, maar in een **tabel** zien. (Dit kan je beschouwen als een twee-DIMensionale lijst, waarbij de eerste waarde van links naar rechts verandert, en de tweede waarde van boven naar beneden). Zoals je ziet zijn lijst-variabelen heel handig voor programma's die tabellen maken. Om deze twee-dimensionale lijst-variabelen in een programma te gebruiken, zouden we de getallen tussen haakjes door variabelen vervangen, bijvoorbeeld A(I,J).
Hier is een voorbeeld van een echte tabel:

	Nederlands	Engels	Wiskunde	Totaal
Eerste rapport	68	88	70	226
Tweede rapport	73	53	91	217
Derde rapport	92	98	82	272
Totaal	233	239	243	715
Gemiddelde	77	79	81	238

Deze tabel toont de cijfers die iemand op school voor drie vakken gehaald heeft. Met inbegrip van de totaalcijfers en de gemiddelden staan er precies 20 verschillende getallen in de tabel. Dat betekent dat we hetzelfde DIM A(3,4) bevel kunnen gebruiken om genoeg variabelen voor deze tabel te maken. Dan moeten we een manier zien te vinden om aan onze lijst-variabelen de gewenste waarden toe te kennen. Hier volgt een manier:

```
100 FOR J=Ø TO 2
110 INPUT A(Ø,J)
120 NEXT J
```

Deze drie regels zijn voor de cijfers voor Nederlands, die variabelen A(0,0), A(0,1) en A(0,2) worden. Telkens wanneer de computer in de FOR-NEXT lus regel 110 bereikt, zal hij ons vragen om de waarden in te voeren (68, 73, 92).

Om vervolgens het totaal van de drie waarden voor Nederlands te berekenen, voeren we in:

```
200 T=0                    ..... T is de variabele voor het
210 FOR J=0 TO 2               Nederlands Totaal
220 T=T+A(0,J)
230 NEXT J
240 A(0,3)=T
```

Op dezelfde manier kunnen we het eerste rapportcijfer voor Engels aan variabele A(1,0) toewijzen en het eerste rapportcijfer voor Wiskunde aan variabele A(2,0). Het volgende programma zal dan op de juiste wijze het totaal voor de drie vakken aan A(3,0) toewijzen.

```
300 T1=0                   ..... T1 is de variabele voor het
310 FOR I=0 TO 2               Totaal van het Eerste Rapport
320 T1=T1+A(I,0)
330 NEXT I
340 A(3,0)=T1
```

Om de andere cijfers, totalen en gemiddelden in het programma op te nemen kunnen we soortgelijke FOR-NEXT lussen gebruiken met lijstvariabelen in de vorm van A(I,J). Het gehele programma komt er dan zo uit te zien:

```
 10 ' *** RAPPORTCIJFER-PROGRAMMA ***
 20 '
 30 DIM A(3,4)
 40 CLS
 50 '
 60 ' *** VOER CIJFERS IN ***
 70 FOR I=0 TO 2
 80 FOR J=0 TO 2
 90 ON I+1 GOTO 100,120,140
100 LOCATE 0,J:PRINT "Nederlands,rapp
ort ";J+1;
110 INPUT A(0,J):GOTO 160
120 LOCATE 0,J+3:PRINT "Engels,rappor
t ";J+1;
```

```
130 INPUT A(1,J):GOTO 160
140 LOCATE 0,J+6:PRINT "Wiskunde,rappor
t ";J+1;
150 INPUT A(2,J)
160 NEXT J
170 NEXT I
180 '
190 '* BEREKEN TOTALEN EN GEMIDDELDEN *
200 FOR J=0 TO 2
210 T=0
220 FOR I=0 TO 2
230 T=T+A(I,J)
240 NEXT I
250 A(3,J)=T
260 NEXT J
270 FOR I=0 TO 3
280 T=0
290 FOR J=0 TO 2
300 T=T+A(I,J)
310 NEXT J
320 A(I,3)=T
330 A(I,4)=INT(A(I,3)/3)
340 NEXT I
350 '
360 ' *** MAAK TABEL ***
370 CLS
380 LOCATE 5,0
390 PRINT "NED  ENG   WIS   TOTALEN"
400 FOR S=1 TO 3
410 LOCATE 2,S+1:PRINT S
420 NEXT S
430 LOCATE 1,5:PRINT "TTL"
440 LOCATE 1,6:PRINT "GMD"
450 FOR I=0 TO 3
460 FOR J=0 TO 4
470 LOCATE I*6+4,J+2
480 PRINT A(I,J)
490 NEXT J
500 NEXT I
```

Om het geheel makkelijker te kunnen overzien verdelen we het programma in drie stukken—Voer Cijfers In, Bereken Totalen en Gemiddelden, en Maak Tabel. Als je bovendien op papier zet waar elke variabele wordt ingevoerd, dan maakt dat het veel gemakkelijker om te onthouden hoe het programma werkt.

Natuurlijk kunnen lijst-variabelen ook voor andere doeleinden gebruikt worden dan "gokautomaten" en tabellen. En, zoals je al wel geraden zult hebben, het aantal variabelen tussen de haakjes kan meer dan twee bedragen: A(P,Q,R) of B(X,Y,Z,XY,XZ,YZ) ... of elke ander kombinatie, tot een maximum van 255 verschillende variabelen! Computer professionals gebruiken vaak lijstvariabelen bij het ontwerpen van video-spelletjes, of voor het bijhouden van de financiën van grote ondernemeingen, of voor andere ingewikkelde opdrachten. Je Sony Computer kan lijst-variabelen gebruiken voor een heel aantal dingen, zoals je zelf wel zult uitvinden als je je blijft oefenen in het programmeren, met vrienden of uit andere boeken. Je hebt op dit gebied al een goede basis, want je hebt nu al in twee verschillende programma's lijstvariabelen gebruikt.

Dus veel geluk er mee, en blijf oefenen!

INDEX

A

Aanhalingstekens (" ") 18–20, 90–91
Adres 68, 71, 75
Apostrof (') 104

B

Beginwaarde 72–73, 85, 94
Bestandsnaam 57, 59
Bevelen 6, 8–10, 15, 27, 47

C

CLS 68, 71
COLOR 13–14, 50
CSAVE 57
CTRL toets 9

D

Decimale-punt 45
DIM 82, 85, 116–117
Dollar-teken ($) 88–89
Dubbele punt (:) 44

E

ELSE (IF—THEN—ELSE) 66, 97, 115–116
Enkel aanhalingsteken 104

F

FOR—NEXT 72–73, 112–114
Formaat 94
Fouten 34
Foutmelding 10, 15, 20, 35, 44

G

GOTO 40–41
Grafische voorstellingen .. 39–40, 43

H

Haakjes ( ) 44

I

IF—THEN 63, 115
IF—THEN—GOTO 95, 115
INKEY $ 91–93
INPUT 62, 110–111
INT (integer = geheel getal) 46
Invoer 9

K

Kleur 11–13, 50
Kleurentabel 13
Komma (,) 44

L

Leestekens 44
LET 20–22
Lettertekens 27, 40, 44
LINE 51
LIST 36, 99
LOCATE 68–69
Loze lus 114
Lus 41, 72–73, 113–114
Lijst-variabelen ... 81–82, 116–117

M

MOD (modulus) 75

N

Negatief getal 108
NEW 37, 44

O

ON-GOTO 86–87
Opmerkingen 103–105

P

PRINT 15–20, 107–110
PRINT USING 94
Programma 28, 39, 47
Programmeren 29, 35
PSET 40

R

Regelnummer 33, 37
REM 104
RENUM 102
RETURN-toets 10

RND (1) 45, 85–86
RUN 36
Rij-variabelen 88–90, 111

S
SCREEN 40
SHIFT-toets 15
Spatie 11, 70, 101, 108
STEP 112–113
STOP-toets 9
Subtitels 104

T
Tabellen 117
Tekens 40–50
Titels 103–105
TO 73
Toetsenbord 6–7

U
USING (PRINT USING) 94

V
Variabelen 21, 27, 71, 94, 111
Verbindingsstreepje (–) 51
Volgorde 29, 32, 47, 49
Voorwaardelijke formule ... 63, 66
Voorwaardelijke lus 72

W
WIDTH 11–12
Willekeurige getallen 45, 85

Sony Corporation Printed in Japan

3-795-898-71 (1)