SONY

MSX

HIT BIT

GEBRUIKERSGIDS

WICHERT VAN ENGELEN

UITGEVERIJ WOLFKAMP

van Engelen HIT BIT GEBRUIKERSGIDS

Uitgeverij WOLFKAMP
Weteringschans 221
1017 XH Amsterdam

I.S.B.N.: 90 - 70556 - 16 - 2

Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt door middel van druk, fotokopie, microfilm, geluidsband of op welke andere wijze dan ook zonder voorafgaande schriftelijke toestemming van de uitgever.

© 1985, Uitgeverij WOLFKAMP

DE

HIT BIT

GEBRUIKERSGIDS

WICHERT VAN ENGELEN

Uitgeverij WOLFKAMP, Amsterdam

INHOUD

1. INLEIDING

Een programma schrijven

Reeds bij uw eerste kontakten met de wereld der microcomputers heeft u velen horen praten over het schrijven van programma's. De MSX-computers zouden daar bij uitstek geschikt voor zijn en beschikken over een uitgebreide BASIC-taal.
Maar wat wil dat zeggen: "een programma schrijven"? Waarom zijn er boekenkasten over vol geschreven? Waarom zouden we het zelf moeten doen? Er zijn toch allerlei cassettes in de handel met de meest uiteenlopende programma's en spelletjes. Die besparen ons veel intikwerk en nadenken is nauwelijks nodig.
Hier heeft u een van de beste redenen om zelf te gaan (leren) programmeren. Nadenken. Als u meer wilt weten over wat computers zijn en wat ze al dan niet kunnen, kunt u niet volstaan met het in de cassetterecorder stoppen van een cassette met ~~en~~ spel of een boekhoudprogramma.
Maar zelfs als u vindt dat het eigenlijk zonde van de tijd is om te leren programmeren, dan zult u al snel merken dat u er wel iets van af moet weten. De computerwetenschap is nog niet zover gevorderd dat alle programma's echt helemaal op maat zijn. Als u een kant-en-klaar programma optimaal wilt gebruiken, zult u ofwel uw eisen moeten aanpassen, ofwel het programma moeten aanpassen. Daarnaast is voor de werking van veel programma's onontbeerlijk dat de gebruiker iets begrijpt van de werking van de computer. Zelf programma's schrijven voor uw MSX-computer is de beste methode om vertrouwd te raken met de computer en om te begrijpen hoe uw MSX werkt.
In dit boek leert u aan de hand van veel oefening hoe de MSX-computers werken en wat u ermee kunt doen. U leert hoe u een probleem kunt vertalen in computertaal en hoe u problemen die te complex of te vaag zijn voor een computer kunt verdelen in verschillende kleine en minder vage problemen.
Bij het schrijven van dit boek is gebruik gemaakt van een SONY HIT-BIT, type HB-75P. Door de grote uitwisselbaarheid van de MSX-computers is dit boek echter geschikt voor elk MSX-systeem.
De programma's in dit boek zijn meestal zeer eenvoudig. U zult ze snel kunnen begrijpen. De programma's in de laatste hoofdstukken veronderstellen de voorafgaande stof als bekend en zijn dus ingewikkelder. Hoewel de programma's allemaal doen wat ze moeten doen, zijn de meeste programma's niet 'af'. In elk programma is altijd plaats voor verbeteringen, of aanpassingen aan de persoonlijke smaak van de gebruiker. Door de begeleidende tekst bij elk programma zal duidelijk gemaakt worden hoe elk programma

opgebouwd is, en van welke tips en truuks gebruik gemaakt is. Daardoor zal de lezer zelf eventuele aanpassingen snel kunnen aanbrengen.
Enkele van de in dit boek voorkomende programma's zijn aangepaste en verbeterde versies van programma's uit het boek: "Op avontuur met de BBC", uitgeverij Wolfkamp.

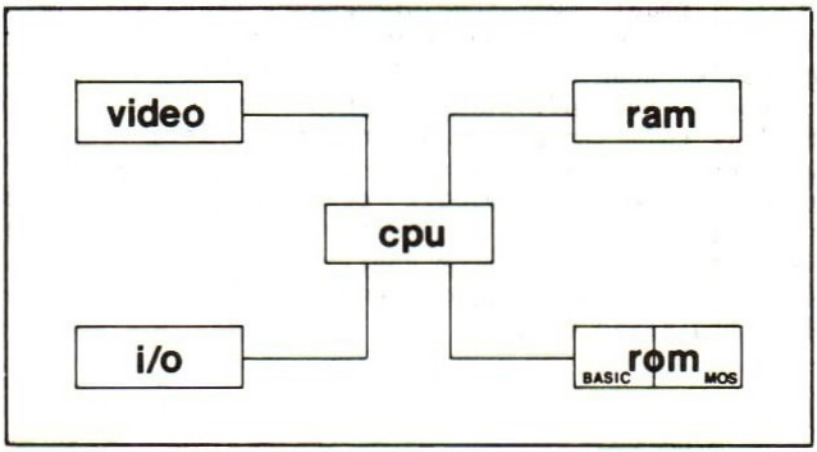
Communicatie mens-computer

De taal waarin we de computer toespreken heet: BASIC. Dit staat voor Beginners All-purpose Symbolic Instruction Code (voor alle doeleinden geschikte symbolische instruktiecode voor beginners). Deze taal werd in de 60-er jaren ontwikkeld in Amerika. Sindsdien zijn er verschillende dialecten ontwikkeld. Een van de nieuwste dialecten is MSX-basic. Dit is een dialect van de algemene BASIC-taal die door de Amerikaanse firma Microsoft speciaal voor de MSX-machines is ontwikkeld.
Naast BASIC kent de computerwereld nog vele andere computertalen die elk vooral geschikt zijn voor gebruik door gevorderden die een speciale toepassing hebben voor de computer: PASCAL (wiskundige doeleinden), FORTRAN (natuurkundige doeleinden), COBOL (administratieve doeleinden), LISP (kunstmatige intelligentie), FORTH (procesbesturing). Daarnaast worden computertalen ontwikkeld die speciaal door kinderen gebruikt kunnen worden. De belangrijkste daarvan is LOGO.
Met BASIC kunt u alle kanten op. Soms gaat het programmeren in BASIC iets omslachtiger, of iets minder elegant dan bij een van de andere talen, maar het voordeel van de eenvoud die BASIC kenmerkt is zo groot, dat zo goed als alle microcomputers standaard met BASIC geleverd worden.
Ook de MSX-computers worden geleverd met een BASIC interpreter (= vertaler). Zoals hierboven al vermeld, wijkt de BASIC van de MSX-computers op verschillende punten af van de standaard BASIC-taal. Normaal gesproken zal dit voor de gebruiker geen nadelen hebben. MSX-BASIC heeft alles dat standaard-BASIC heeft, plus nog wat extra. Alleen als u op de MSX programma's gaat ontwikkelen voor andere (niet-MSX) computers, moet u opletten dat u geen opdrachten gebruikt die niet in standaard-BASIC voorkomen.

2. DE MSX-COMPUTER

De binnenkant

Net als elke microcomputer bestaat elke MSX-computer uit de volgende onderdelen:



sterk vereenvoudigd schema van een MSX-computer

De CPU

De CPU (Central Processing Unit = centrale verwerkingseenheid) is het onderdeel dat centraal in de computer staat. Het bepaalt waar welke gegevens naar toe gaan. Op deze manier 'verwerkt' het de gegevens. Elk gegeven dat de CPU binnenkomt wordt bekeken, waarna de CPU bepaalt wat er verder mee gebeurt. De CPU zou vergeleken kunnen worden met het hoofd van een groot organisatiebureau tijdens een door hen georganiseerd evenement.
De CPU van elke MSX-computer is een Z-80A van de Amerikaanse firma Zilog, en bestaat uit een enkele (doch ingewikkelde) chip.

Dit is wellicht de beste plaats om een veel voorkomende spraakverwarring te verhelpen. Veel mensen maken geen onderscheid tussen een microcomputer en een microprocessor. Een microprocessor is het belangrijkste onderdeel van een microcomputer. Een microcomputer is een machine als de MSX die u wellicht bij u in de buurt heeft staan. Dat wil zeggen een samenraapsel van een toetsenbord, een kastje, wat chips, enkele andere electronische onderdelen, wat draden en diversen. Deze

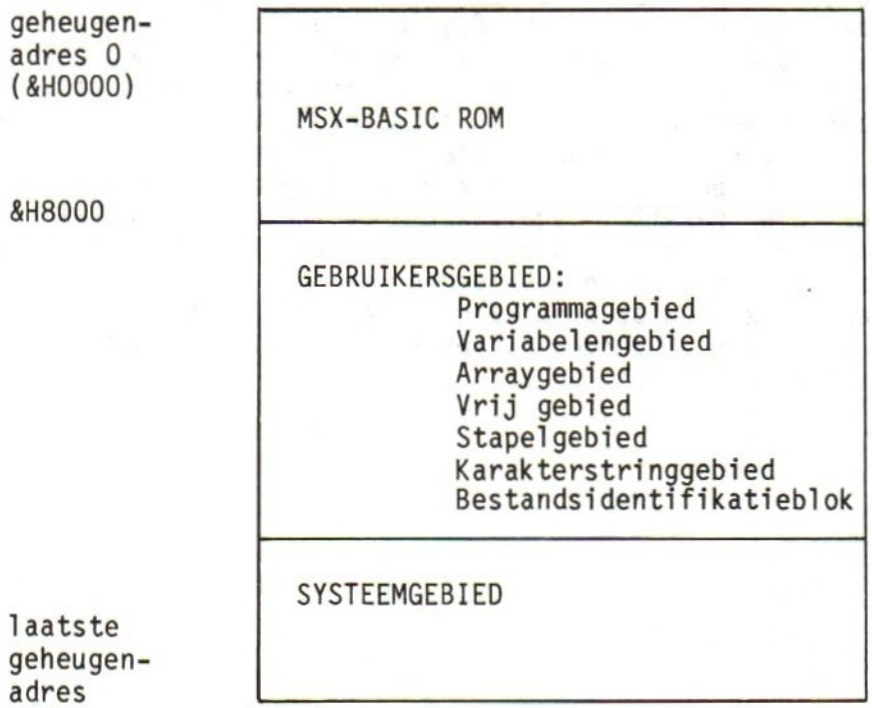
microcomputer is in staat een heleboel opdrachten die de gebruiker geeft uit te voeren en de resultaten ervan weer te geven via een draad op beeldscherm, een printer, een cassetterecorder of andere randapparatuur.
Een microprocessor is het centrale en typerende onderdeel van een microcomputer. Het vormt de CPU. Er zijn op het ogenblik vele merken en soorten microcomputers op de markt, maar slechts enkele soorten microprocessors. Zodoende hebben heel verschillende computers dezelfde microprocessors.
De Japanse computerfabrikanten zijn nog verder gegaan en hebben niet alleen allemaal dezelfde processor gebruikt, maar ook dezelfde geheugenchips, en gezorgd voor een gelijke interne werking van alle door hun gemaakte MSX computers. Het voordeel daarvan springt onmiddellijk in het oog: in de zeer onoverzichtelijk computerwereld waar niets op iets anders past, bestaat nu opeens een heel scala van computers en randapparatuur die zonder problemen aan elkaar gekoppeld kunnen worden en onderling elkaars programma's begrijpen. De ontwikkeling van commerciele software zal daardoor sneller gaan en boeken als deze hoeven slechts een keer geschreven te worden om voor alle MSX-computers bruikbaar te zijn.

Het geheugen: ROM en RAM

Het geheugen van een microcomputer bestaat uit een deel RAM en een deel ROM. RAM staat voor Random Access Memory (=willekeurig toegankelijk geheugen: het vrij beschikbare geheugen). ROM staat voor Read Only Memory (=enkel lees geheugen).
De RAM is dat gedeelte van het geheugen waarover de gebruiker vrij kan beschikken om gegevens (waaronder programma's) op te slaan. Niet elke MSX-computer heeft evenveel RAM-geheugen. SONY heeft computers met 16 K bytes (uitbreidbaar tot 32 of 64 K bytes d.m.v. geheugenblok) of 64 K bytes vrije geheugenruimte.
Een K byte is de eenheid waarin geheugenruimte aangegeven wordt. Een K byte is een kilo-byte en staat voor 1024 geheugenplaatsen van een byte groot. Een byte bestaat uit 8 bits (BInairy digiTs). Elke bit is een plaatsje in de computer dat aan of uit staat, een plus of een min, een een of een nul aangeeft, kortom een soort hele kleine schakelaar. Met deze enorme hoeveelheden 'schakelaars' kan de computer allerlei opdrachten voor u vervullen.
De ROM is dat gedeelte (de naam zegt het al) dat door de gebruiker alleen gebruikt kan worden om gegevens uit te lezen. Nieuwe gegevens kunnen hier niet geplaatst worden. De ROM bevat in elke MSX-computer de BASIC interpreter.
Daarnaast hebben verschillende MSX-computers een programma

'ingebakken'. Zo'n programma (bijvoorbeeld de databank van de SONY) kan door de gebruiker niet veranderd worden (het zit immers in ROM). De nieuwe gegevens die door het programma ontstaan, worden tijdelijk in RAM opgeslagen, en eventueel later op een cassette of diskette weggeschreven.
De BASIC interpreter is de tolk van de computer. De interpreter vormt de grammatika en het woordenboek, waarmee de computer de door de gebruiker in BASIC gegeven opdrachten kan begrijpen. Een computer werkt zelf binair. Denk aan de schakelaar (bits) die alleen een stand aan of uit, 1 of 0 kennen. De taal BASIC wordt alleen begrepen doordat de interpreter het vertaalwerk doet. Het is wel mogelijk de computer direkt in 1-en en 0-en aan te spreken, maar dit is voor de programmeur zeer lastig. Dit boek behandelt alleen het programmeren in de taal BASIC.
De BASIC interpreter heeft 32 Kbytes aan geheugen nodig. Dit lijkt erg veel, maar doordat de interpreter zo uitgebreid is, kan de gebruiker veel verschillende opdrachten geven die de computer allemaal begrijpt.
Het systeemgebied is voor de BASIC programmeur van minder belang. Dit deel van het geheugen verschaft de CPU de gegevens die nodig zijn om de interne huishouding van de computer te regelen.

Schematisch ziet de geheugenopbouw van een MSX-computer er zo uit:



De lengte van het RAM geheugen is afhankelijk van het type MSX-computer. Daarbij komt dat door de mogelijkheden van de Z-80A processor binnenin de MSX, enige haken en ogen kleven aan de hoeveelheid geheugenruimte die de gebruiker vrij kan gebruiken volgens de fabrikant. Zo kan het voorkomen dat de computer na het aanzetten een andere hoeveelheid 'vrije' bytes meldt, dan er volgens de specificatie van de computer vrij moeten zijn. Het voert te ver om op deze plaats hier verder op in te gaan. Daarbij komt dat de MSX nogal zuinig omgaat met zijn geheugenruimte en dat u dus flink lange programma's kunt schrijven voordat u geheugenruimte tekort komt.

I/O en video

Over de twee andere delen van het schema valt iets minder te vertellen.
De VIDEO zorgt voor de weergave op het beeldscherm. De MSX kan weergeven op een monitor (kleur of zwart/wit) of op een gewone televisie (kleur of zwart/wit). Het geluid dat de MSX kan voortbrengen wordt weergegeven door de televisie. Als u een monitor gebruikt, moet u een type nemen dat tevens geluid kan voortbrengen. Een monitor geeft een scherper beeld maar is duurder. Een televisie staat meestal al in huis, maar gezamenlijk gebruik zorgt (soms) voor ongemak door de keuze tussen het geweldige tv-programma XXX en een fascinerend computerprogramma.
De I/O (Input/Output = invoer/uitvoer) zorgt voor de communicatie met randapparatuur zoals een printer, cassetterecorder etc. De MSX is uitgerust met een parallel interface. Dat wil zeggen dat zich in de computer een vertaalkastje bevindt, dat de gegevens van de computer in een signaal omzet dat begrepen kan worden door een printer die parallel gegevens kan ontvangen. (parallel wil zeggen dat verschillende signalen tegelijk via verschillende draden naast elkaar worden verzonden en ontvangen). De MSX gebruikt een Centronics-standaard. Een printer kan dus aangesloten worden zonder verdere kosten dan een verbindingskabel.

De buitenkant

Hoewel dit alles meestal in de bij een MSX meegeleverde handleiding staat, vermelden wij hier voor de volledigheid toch in het kort de aansluiting van de cassetterecorder en de werking van het toetsenbord.

De cassetterecorder

Om de programma's die u schrijft of de gegevens voor programma's te kunnen bewaren, ook nadat de computer uitgezet is, kunt u deze opslaan door middel van materiaal dat niet afhankelijk is van een doorlopende elektrische spanning. Bij grote computers werd daar vroeger ponsband voor gebruikt. Tegenwoordig worden de gegevens bijna altijd op magnetisch materiaal opgeslagen.
Magnetisch materiaal bestaat in vele vormen. De grote computers (main-frames) gebruiken zeer grote, brede banden. Goedkoper zijn de zogenaamde 'hard-disks'. Dit zijn grote schijven van magnetisch materiaal, meestal per vijf verpakt in een grote plastic wisselcassette, die door een speciale machine gelezen en beschreven kunnen worden. Weer iets goedkoper is een systeem waarbij een enkele hard-disk ingebouwd is in het lees- en schrijfapparaat, en dus niet gewisseld kan worden. Deze worden (naar de fabrikant) meestal Winchesters genoemd. Deze laatste soort wordt steeds meer gebruikt in combinatie met microcomputers. Weer goedkoper zijn soortgelijke schijven maar dan van zacht materiaal (floppy disks = diskettes = slappe schijven) en kleiner. Ook voor dergelijke schijven is een speciaal lees- en schrijfapparaat nodig. Door de voor deze apparaten benodigde mechanische constructies kosten deze ruim duizend gulden per stuk. Het ziet er niet naar uit dat sterke prijsdalingen spoedig te verwachten zijn. Voor zakelijk en veelvuldig gebruik van de computer zijn zij echter onontbeerlijk daar de snelheid van schrijven/lezen vele malen sneller is dan goedkopere methoden van magnetische opslag: banden (bandrecorder) en cassettes (cassetterecorder). De meest recente ontwikkeling op dit gebied is dat speciaal voor de MSX-computers een nieuwe soort diskette op de markt is gebracht. Deze is veel kleiner dan een gewone diskette en is gevat in een hoesje van hard plastic. De verwachting is dat deze schijfjes bij een grote markt goedkoper zullen worden dan de tot nu toe populaire diskettes. Daarnaast proberen verschillende fabrikanten om een tussenvorm tussen cassette en diskette te vinden. Deze vormen bestaan meestal uit een cassettecartridge met zeer snel rondspoelende band. Deze vorm is zowel wat prijs als wat snelheid betreft een gemiddelde tussen cassettesystemen en diskettesystemen.
Voor de hobbyist die niet als eerste eis de snelheid heeft, zijn

cassetterecorders aan te raden. De diskettesystemen zijn duur en van de andere systemen is de vraag of zij algemeen erkend zullen worden, waardoor een zekere uitwisselbaarheid ontstaat. Voor een cassetterecorder voldoet elke recorder van een willekeurig merk. In het algemeen voldoen de goedkope recorders zelfs beter dan de dure.
Het aansluiten van de cassetterecorder zodat deze op de juiste manier gegevens opneemt van of afgeeft aan de computer is heel gemakkelijk bij de MSX. Tenslotte heeft men voor standaardisatie gezorgd! Aan de achterkant van de computer vindt u een aansluiting met het opschrift TAPE (band). Hierin past de stekker van het meegeleverde cassettesnoertje. Dit herkent u doordat aan de ene zijde een ronde achtpolige DIN-stekker zit, en aan de andere kant drie kleine stekkers (een witte, een rode en een nog kleinere zwarte). Aan de kant van de cassetterecorder steekt u de witte stekker in de kop- of oortelefoonaansluiting (meestal aangegeven met EAR). De rode stekker gaat in de microfooningang (MIC), en de zwarte gaat in de aansluiting voor afstandsbediening (REMOTE), of in de andere (kleinere) microfooningang (MIC). Heeft u een cassetterecorder met alleen een ronde DIN-aansluiting, dan moet u uw radio/tv-winkelier om een verloopsnoertje vragen. Neem het snoer van MSX naar cassetterecorder mee, en vertel waar alles voor dient. Neem eventueel dit boek mee.
Heeft uw cassetterecorder geen afstandsbediening, dan is het zwarte stekkertje zinloos (let op: bij sommige cassetterecorders is de afstandbediening ingebouwd in de kleine microfooningang!). Bij een eventueel verloopsnoertje naar een DIN-stekker moet hiermee rekening worden gehouden.
Zo, uw computer is nu helemaal compleet. We gaan ervan uit dat u een cassetterecorder heeft. Heeft u een diskettesysteem, dan kunt u dit boek net zo goed gebruiken. Voor het lezen en schrijven heeft u echter iets andere opdrachten nodig. Waar nodig zal hier apart op worden ingegaan.
Hoe u de cassetterecorder moet gebruiken is nu nog niet van belang. Zodra u uw eerste programma geschreven heeft komen we erop terug.

Het beeldscherm

Normaal gesproken wordt als beeldscherm een televisie gebruikt. In dat geval is het aansluiten van het beeldscherm uiterst eenvoudig. Een kabeltje is meegeleverd met de computer. U hoeft alleen de stekker achterin de MSX te steken en de andere stekker in de antenneingang van uw televisie. De SONY HIT BIT levert een antenneschakelaar mee. U steekt de stekker van de computerkabel in de schakelaar en daarnaast steekt u de normale antennestekker in de schakelaar. De stekker van de schakelaar zelf steekt u in

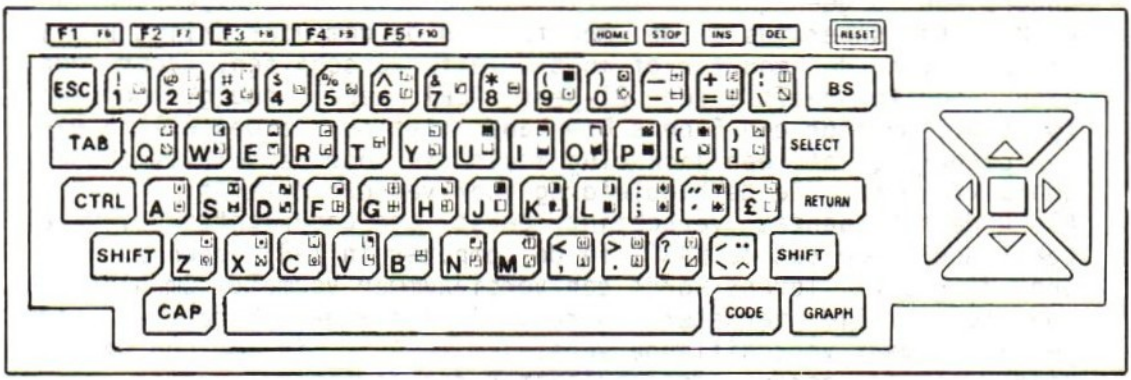
de antenneingang van de televisie. Omschakelen is nu een kwestie van de schakelaar verschuiven.
Met de zenderkeuzeknoppen en de fijnafstelling zoekt u de juiste afstelling van uw televisie voor de computer. Het is het handigste om voor deze afstelling een aparte zendervoorkeuzeknop te gebruiken. Zet het geluid van de televisie niet uit. De MSX heeft geen eigen luidspreker, maar gebruikt die van de televisie. Wenst u een scherper beeld dan bij een televisie mogelijk is (bijvoorbeeld voor tekstverwerking) dan moet u een monitor gebruiken. Let u er wel op dat u een monitor met geluid neemt, anders kunt u geen gebruik maken van de geluidsmogelijkheden van de MSX. Het snoer tussen de monitor en de computer krijgt u bij aankoop van de monitor of moet u zelf aanschaffen. Voor een kleurenmonitor heeft u een 21-pens PERI kabel nodig. Voor een zwart/wit monitor een 5-pens DIN kabel (let op de opstelling van de pennen!), of een jack-plug zoals die voor de televisiekabel gebruikt wordt. Zie de handleiding voor verdere details.
Als u de MSX aanzet verschijnt eerst een copyrighttekst. Deze tekst deelt mee dat de BASIC in de computer in 1983 geschreven is door Microsoft. Tevens wordt een versienummer vermeld. Omdat elke computer altijd verbeterd wordt, is het mogelijk dat op een gegeven moment verschillende versies MSX-BASIC in omloop zijn. Bijna altijd zorgen de fabrikanten dat de latere versies alles kunnen wat de oude versies al konden plus nog wat extra. Het versienummer kan dus van belang worden als u moet weten of een bepaald programma ook op uw computer zonder moeilijkheden gebruikt kan worden.
Bij MSX-computers die een ingebouwd programma hebben verschijnt na korte tijd de aankondiging van dat programma. Bij de SONY HIT BIT is dat de persoonlijke data-bank. Omdat we u in dit boek niet willen leren hoe u een programma kunt gebruiken, maar hoe u zelf een programma kunt maken, verlaten we dit programma snel door het donkere stipje voor de verschillende mogelijkheden met de pijltoetsen helemaal rechts op het toetsenbord vier keer naar beneden te verplaatsen. De stip staat nu voor BASIC. Druk nu op de RETURN toets.
De MSX meldt zich nu met:

```
MSX BASIC version 1.0
Copyright 1983 by Microsoft
28815 Bytes free
Ok
```

De getallen van deze melding kunnen verschillen: u kunt een andere versie hebben en de hoeveelheid geheugenruimte is niet bij elke machine gelijk.
Met Ok geeft de computer aan dat hij voor u klaar staat. We kunnen dus aan de slag.

Het toetsenbord

De communicatie met de computer gaat meestal via het toetsenbord. Hoewel de toetsenborden van de MSX-computers onderling wel enigzins verschillen, zijn er veel overeenkomsten.
De toetsen zijn gerangschikt volgens het QWERTY-patroon. Een patroon waarvan men een tijd geleden, toen de schrijfmachines ontstonden, dacht dat het het snelste tikte. Tegenwoordig zijn nog steeds bijna alle schrijfmachines met dit toetsenpatroon uitgerust.



toetsenbord van de SONY HIT BIT

Hoewel de toetsenborden verschillen, treft u op elk toetsenbord de volgende onderdelen aan:

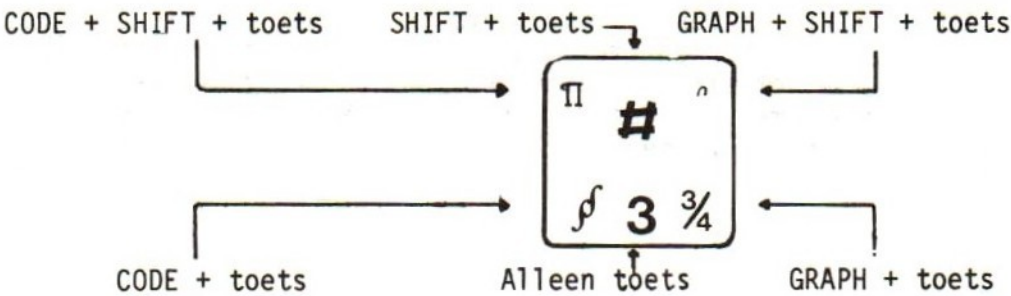
Lettertoetsen

Deze toetsen geven alle normaal gebruikte letters, cijfers en leestekens weer. Daarnaast kunnen met deze toetsen verschillende grafische tekens en symbolen weergegeven worden. Zie hieronder.

Controletoetsen

Aan de linker- en rechterkant van het toetsenbord vindt u een stel speciale toetsen die mede bepalen hoe ingetikte tekst op het scherm komt te staan en wat de normale lettertoetsen bij het indrukken weergeven.
De ESC (ESCape = ontsnap) toets is afkomstig van andere computers en dient daar voor het stopzetten van het programma. In MSX-BASIC wordt deze toets daarvoor niet gebruikt. U kunt uw programma's echter zo maken dat de toets wel werkt.
De TAB (TABulatie) toets zorgt ervoor dat de weergave van de

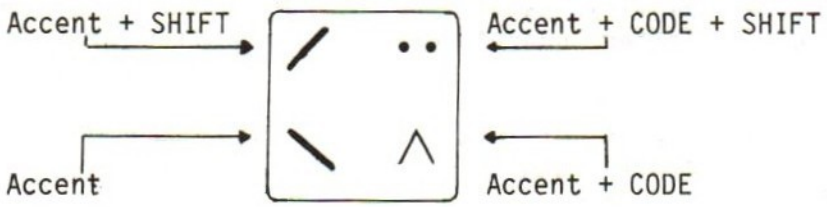
lettertekens verder gaat naar een punt aangegeven door een TAB-stop. U kent dit wellicht van gewone schrijfmachines. De tabulatiestops zijn ingesteld op telkens acht lettertekens tussenruimte. De letters waar 'overheen gesprongen' wordt, worden gewist.
De SHIFT (= verschuif) toets zorgt ervoor dat de lettertoetsen hoofdletters weergeven of het bovenste van twee aangegeven tekens. U moet daarvoor tegelijk de SHIFT en de gewenste toets indrukken.
De CAP (CAPitals = hoofdletters) zorgt voor hetzelfde effect. Als u deze toets indrukt gaat de CAP-indikator (een klein lampje) branden ten teken dat alles dat vanaf nu ingetikt wordt als hoofdletter weergegeven zal worden. Van de toetsen met twee symbolen wordt echter het onderste symbool gegeven! Om het bovenste symbool te krijgen moet altijd de SHIFT gebruikt worden. Om de letters weer als kleine letters weer te geven moet de CAP uitgezet worden. Dit gebeurt door de CAP-toets nogmaals in te drukken.
De CODE-toets zorgt voor weergave van een bijzonder grafisch symbool op elke toets. Welke toets welk symbool weergeeft is te vinden op de toetsen zelf (de linkersymbolen), of op een speciale meegeleverde kaart. CODE met de toets geeft het symbool linksonder, CODE met SHIFT en de gewenste toets tegelijk ingedrukt geeft het symbool linksboven.
De GRAPH (GRAPHics = grafiek) toets zorgt voor weergave van de grafische symbolen aan de rechterkant van de toetsen of de kaart. Ook hier geldt dat alleen de GRAPH met de gewenste toets het grafische symbool rechtsonder geeft, en GRAPH plus SHIFT plus gewenste toets, het grafische symbool rechtsboven geeft.
Een enkele toets kan dus de volgende tekens weergeven:



Alle tekens die in MSX-BASIC mogelijk zijn, plus de manier waarop die via het toetsenbord ingevoerd moeten worden, vindt u in bijlage 1.

De ACCENTEN-toets geeft u de mogelijkheid om accenten op lettertekens te plaatsen. Hiervoor drukt u eerst de accenttoets

op de gewenste manier in:



U ziet nu niets op het scherm verschijnen. Als u vervolgens de toets indrukt van de letter waarop u het accent geplaatst wilde hebben, verschijnt de letter samen met het accent op het scherm.
De SELECT (= keuze) toets is alleen bedoeld voor gebruik in programma's. De toets wordt in MSX-BASIC niet gebruikt.
De RETURN (terug) toets is de belangrijkste toets op het toetsenbord. Om te zorgen dat de computer niet alles wat u intikt onmiddellijk gaat uitvoeren, voordat u klaar bent met de opdracht, wacht de computer met het opnemen van de opdracht totdat u de RETURN-toets indrukt (de enkele uitzonderingen hierop worden later behandeld). Het indrukken van de RETURN-toets is dus het signaal voor de computer dat u klaar bent met de opdracht en dat het ingetikte ingevoerd kan worden.
De STOP-toets spreekt haast voor zichzelf. Door deze toets in te drukken stopt het programma. De toets kan ook gebruikt worden om het 'listen' (=lijsten, het tonen van alle programmaregels in volgorde) te stoppen. Door de toets nogmaals in te drukken wordt de listing of het programma hervat. De toets kan ook tegelijk met de CTRL-toets (zie hieronder) ingedrukt worden. In dat geval kan het programma alleen hervat worden door de CONT-opdracht. De listing is definief onderbroken en moet opnieuw opgestart worden met LIST.
De CTRL (ConTRoL = besturings) toets is tot het laatste bewaard. Met deze toets kunnen de andere toetsen speciale funkties verrichten. In het algemeen zijn deze bijzondere funkties bepaald door een programma. Standaard kunnen in MSX-BASIC de volgende funkties opgeroepen worden:

CTRL en B-toets	cursor springt naar begin van woord (als de cursor al bij het begin van een woord staat, gaat de cursor naar het begin van het voorafgaande woord). Zie ook de opmaaktoetsen.
CTRL en C-toets	schakelt het automatisch nummeren van de regels uit. De invoerwachttijd wordt gelijk aan de standaardtijd.
CTRL en E-toets	alle letters tussen de cursor en het einde van de regel worden gewist.
CTRL en F-toets	cursor naar begin van volgende woord.
CTRL en G-toets	bliep klinkt
CTRL en H-toets	gelijk aan BS-toets.
CTRL en I-toets	gelijk aan TAB-toets.
CTRL en J-toets	cursor een regel naar beneden.
CTRL en K-toets	gelijk aan HOME-toets (zie onder)
CTRL en L-toets	gelijk aan SHIFT + HOME.
CTRL en M-toets	gelijk aan RETURN.
CTRL en N-toets	cursor naar positie rechts van het laatste teken op de regel.
CTRL en R-toets	gelijkaan INS-toets (zie onder).
CTRL en U-toets	regel waar cursor op staat wordt gewist.
CTRL en X-toets	gelijk aan SELECT.
CTRL en \-toets	cursor plaats naar rechts.
CTRL en]-toets	cursor plaats naar links.
CTRL en SHIFT en 6	cursor plaats naar boven.
CTRL en SHIFT en-	cursorplaatsnaar beneden.
CTRL en SHIFT en pond	cursor plaats naar boven.

Opmaaktoetsen

Om te zorgen dat een tekst of welke weergave dan ook, op het beeldscherm er netjes uitziet, is het soms nodig deze (achteraf) op te maken. Hiervoor dient een stel toetsen aan de linkerkant van het toetsenbord. Deze toetsen kunnen allemaal door een programma een bepaalde betekenis krijgen. Standaard doen zij in MSX-BASIC (dus direkt na het aanzetten van de computer) het volgende:
De vier PIJL-toetsen helemaal links, apart van het toetsenbord verplaatsen de cursor. De cursor is een klein wit blokje. Dit blokje geeft aan waar het eerstvolgende in te tikken karakter geplaatst zal worden. Hierdoor kunt u altijd zien waar u met tikken gebleven bent. Door de vier pijl-toetsen kan de cursor naar een willekeurige plaats op het scherm verplaatst worden. De tekst op het scherm wordt hierdoor niet beinvloed.
De BS (Back-Space = plaats terug) toets verplaatst de cursor een

positie naar links en wist daarbij de letter die de cursor tegenkomt. Staat de cursor al helemaal links op het scherm, dan wordt de letter op die positie gewist, en schuift de gehele regel een positie naar links.
De DEL (DELete = wis) toets doet iets vergelijkbaars. De cursor blijft op de positie waar hij is, maar het karakter op die positie wordt gewist. De hele tekst rechts van de cursor wordt een plaats naar links verschoven.
De INS (INSert = invoeg) toets verandert de modus (werkniveau) van de computer. Normaal gesproken schrijft de computer bij het tikken over de oude tekst heen. Door de INS-toets in te drukken, wordt de cursor kleiner, en gaat de computer in de invoeg-modus. Dit wil zeggen dat de tekst de nu ingetikt wordt, de rest van de tekst doet opschuiven, zonder dat er overheen geschreven wordt. De invoeg-modus wordt weer verlaten door nogmaals INS in te drukken, of door de cursor m.b.v. een pijltoets te verplaatsen. De cursor verandert dan weer naar normale grootte.
De HOME (= thuis) toets verplaatst de cursor naar de thuispositie. Dat is de linker bovenhoek. Alle tekst op het scherm blijft onveranderd. Wordt de HOME-toets gelijk met SHIFT ingedrukt, dan wordt het gehele scherm gewist.

Resten nog 6 toetsen. De RESET (=opnieuw instellen) is een toets die u eigenlijk nooit moet gebruiken. Door deze toets in te drukken vergeet de computer alles waar hij mee bezig was en keert hij terug naar een toestand die gelijk is aan de staat van de computer onmiddellijk na het aanzetten. Voor het stoppen van een programma dient u altijd STOP te gebruiken. Slechts in uiterste noodgevallen mag de RESET gebruikt worden.

De funktietoetsen

De toetsen linksboven zijn funktietoetsen. Deze toetsen hebben allemaal een eigen funktie:

funktietoets		SHIFT + funktietoets	
F1	color	F6	color 15,4,4 RETURN
F2	auto	F7	cload"
F3	goto	F8	cont RETURN
F4	list	F9	list. RETURN
F5	run	F10	cls : run RETURN

Waarvoor deze funkties allemaal nodig zijn zal u in de loop van dit boek vanzelf duidelijk worden. Doordat gehele funkties door het indrukken van een enkele toets ingevoerd kunnen worden, kan

er veel sneller geprogrammeerd worden.
De gebruiker kan opdrachten die veel gebruikt worden, zelf in een funktietoets plaatsen door middel van de opdracht KEY.
U tikt daarvoor:

```
KEY nr, "xxxxxxxxxxxxxx"
```

nr staat hierbij voor het nummer van de te definieren toets (van 1 tot en met 10) en "xxxxxxxxxxxxxx" staat voor een rij van maximaal 15 willekeurige lettertekens. Als u dus heel vaak de opdracht GOSUB wilt gebruiken bij het programmeren, kunt u deze als volgt in funktietoets 3 onderbrengen:

```
KEY 3, "GOSUB"        (RETURN)
```

Na het intikken van de tekst moet u niet vergeten op RETURN te drukken. Dan pas weet de computer dat u klaar bent met het intikken van voor hem bedoelde tekst.
De inhoud van funktietoets 3 staat nu tussen de andere omschrijvingen onderin het beeld. Om de omschrijvingen te verwijderen, zodat u deze onderste regel voor andere doeleinden kunt gebruiken tikt u:

```
KEY OFF                (RETURN)          (= toets uit)
```

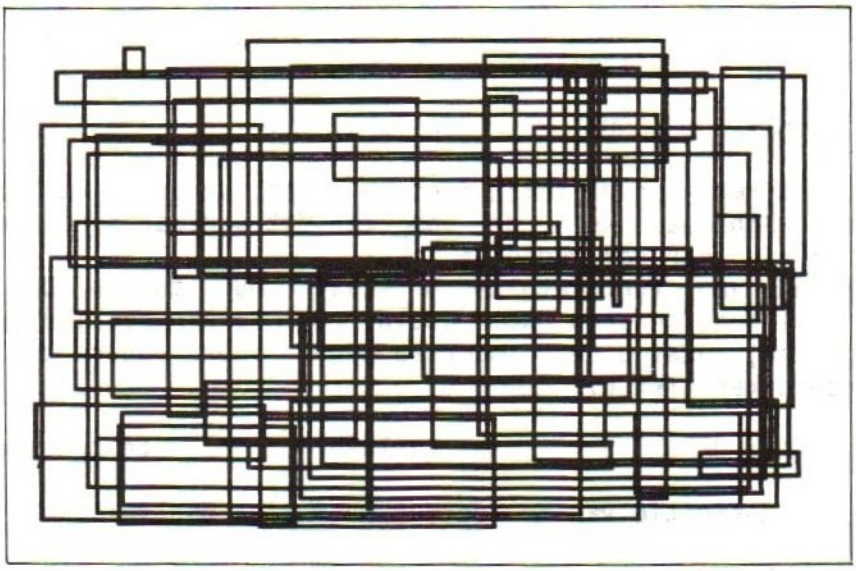
Wilt u de definitie van de toetsen weer zien dan volstaat

```
KEY ON                                   (= toets aan)
```

Om de inhoud van alle toetsen bij elkaar te zien tikt u:

```
KEY LIST
```

U krijgt dan een lijstje van alle funktietoetsdefinities te zien. De nummers van de toetsen moet u er zelf bij denken. Als u bovenstaande teksten ingetikt heeft, ziet u op de derde regel het ingetikte GOSUB staan.
De definities van de toetsen gaan verloren bij het uitzetten van de computer. Als u de computer later weer aanzet, krijgt u automatisch de standaard ingeprogrammeerde definities weer terug. Maakt u veel gebruik van eigen definities, dan is het raadzaam deze in een apart programma te plaatsen.



3. DE SCHERMEN

Het beeldscherm dat op de MSX-computer aangesloten is, kan op vier verschillende manieren opgebouwd worden. Twee manieren om grafiek weer te geven en twee manieren om tekst weer te geven.
U kiest voor een bepaalde schermweergave met de opdracht:

SCREEN nr

nr staat hier voor een getal van 0 tot en met 3.

Scherm 0

Dit is het scherm dat standaard verschijnt als u de MSX aanzet (en doorschakelt naar BASIC). Op dit scherm kunnen 40 karakters naast elkaar gezet worden. Voor elke letter of symbool zijn 6 stippen van links naar rechts en 8 stippen van boven naar beneden beschikbaar. Alle normale letters en cijfers kunnen hiermee moeiteloos worden weergegeven. Een deel van de grafische tekens heeft echter een rooster van 8 x 8 stippen nodig. Deze tekens worden op dit scherm niet helemaal weergegeven, aan de rechterkant wordt een stukje afgesneden.
Er gaan 24 regels op een scherm.

Scherm 1

Dit scherm gebruikt u om een beter te lezen letter te krijgen. Voor elke letter wordt nu een rooster van 8 x 8 stippen gereserveerd. Omdat de letters maar 6 stippen in de breedte nodig hebben, staan ze iets verder van elkaar en zijn daardoor beter te onderscheiden.
De verschillende symbolen en karakters zijn nu allemaal in het geheel te bewonderen.
Door de extra breedte die elke karakterpositie nu heeft gaan er op dit scherm slechts 32 karakters op een regel.
Er gaan 24 regels op een scherm.

Scherm 2

Dit scherm is bedoeld voor het weergeven van fijn getekende grafiek. Voor een gedetailleerde uitleg van deze wijze van tekenen zij u verwezen naar de hoofdstukken over grafiek en kleur.
Het beeld is opgebouwd als een rooster van 256 (links-rechts)

bij 192 (boven-beneden) stippen. Elke stip kan apart gekleurd worden. De kleuring van de stipppen gaat echter per blokje. Bij gebruik van dit scherm kan men dan ook niet verzekerd zijn van een juiste kleurweergave. Dit wordt treffend geillustreerd door het volgende programma dat uw MSX op hol laat slaan. Tik het programma precies zoals hier staat in en druk na elke regel (en niet eerder) op RETURN. Zijn alle regels getikt, druk dan op F5 (voor RUN).

```
10 COLOR 15,1,1
20 SCREEN 2
30 X=INT(RND(1)*256)
40 Y=INT(RND(1)*192)
50 Q=INT(RND(1)*256)
60 R=INT(RND(1)*192)
70 C=INT(RND(1)*15)
80 LINE(X,Y)-(Q,R),C
90 IF INT(C/2)=C/2 THEN BEEP
100 GOTO 30
```

Dit programma tekent op willekeurige plaatsen een lijn met een willekeurige kleur. De eerste tijd gaat alles goed, maar al snel blijkt dat de computer wel de lijnen mooi tekent, maar de kleuren door elkaar gaat gooien. Hoe het programma werkt komt later aan de orde. U kunt het programma stoppen door STOP in te drukken. Wilt u het programma definitief stoppen, dan drukt u op CTRL en STOP.

Scherm 3

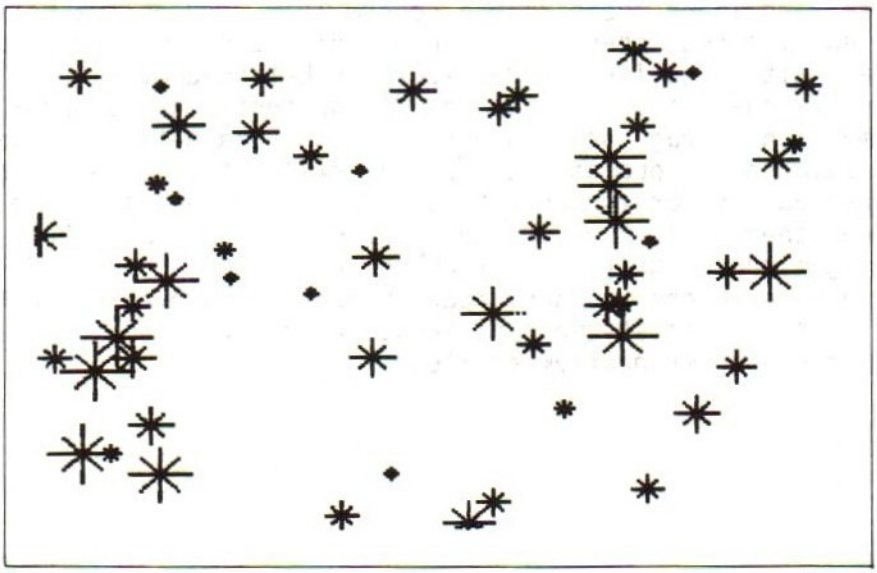
Ook dit scherm heeft een verdeling van 256 x 192 stippen. Dit keer echter kunnen alleen hokjes van 4 x 4 stippen gekleurd worden. De grafische weergaves worden dus minder fijn. Daar staat tegenover, dat elk vakje dat zo ingekleurd word, een eigen kleur kan krijgen.
U kunt dit goed te zien krijgen door in regel 20 van het vorige programma de 2 (van SCREEN) te vervangen door 3. Als het vorige programma nog loopt tikt u tegelijk CTRL en STOP. De computer zegt nu:

```
Break in ...
Ok
```

Hiermee geeft de computer aan dat hij gestopt is in de op de stippeltjes geplaatste regel. Met Ok geeft de computer aan dat het woord weer aan u is, hij wacht op een opdracht. Druk F4 in en druk op RETURN, of tik LIST en druk op RETURN. De computer laat

nu het hele programma zien. Verplaats de cursor (het witte blokje dat aangeeft waar het volgende teken geplaatst gaat worden) met de pijltoetsen naar de 2 achter SCREEN in regel 20 (plaats de cursor bovenop de 2). Tik nu een 3 en druk op RETURN. Verplaats de cursor weer naar onderaan de listing (dat is de weergave van alle programmaregels). Het doet er niet toe waar dat is, als het maar een schone regel is. Anders begrijpt de computer u niet. Tik nu weer F5 of tik RUN en druk op RETURN.
De strepen die nu getrokken worden zijn veel dikker, maar de kleuren lopen niet meer door elkaar.

Voor alle schermen geldt dat er boven en onder het scherm een aparte rand loopt. Deze is bij scherm 1, 2 en 3 onafhankelijk van de kleur van het scherm te kleuren. Bij scherm 0 is de kleur van de rand altijd gelijk aan de kleur van het scherm.
De grafische schermen (2 en 3) springen na gebruik altijd automatisch terug naar scherm 0. Om aan het einde van een programma niet opeens het beeld kwijt te raken, moet de programmeur ervoor zorgen dat het scherm blijvend aanwezig is.
Bij de schermen 1, 2 en 3 kan gewerkt worden met SPRITES. Dit zijn gekleurde grafische vlakken die 'voor' het beeld heen en weer bewogen kunnen worden. Meer hierover en over het maken van grafiek op een tekstscherm of tekst op een grafisch scherm in de hoofdstukken over grafiek en kleur.



4. DIREKTE OPDRACHTEN

De eenvoudigste wijze om uw MSX te gebruiken is hem een opdracht te geven (in BASIC natuurlijk, want anders begrijpt hij u waarschijnlijk niet). Enkele direkte opdrachten die niet tot BASIC horen heeft u hiervoor al kunnen lezen. Zo kunt u door de CTRL en de G-toets tegelijk in te drukken de computer opdracht geven om een bliep te laten horen. Deze opdracht wordt direkt uitgevoerd. De KEY-opdrachten waren ook direkte opdrachten. Onmiddellijk na het indrukken van RETURN, waardoor u de computer opgeeft dat u klaar bent met de opdracht, wordt de opdracht uitgevoerd en de funktietoets geherdefinieerd.
Om nu eens een BASIC-opdracht uit te proberen moet u de computer eerst weer in de begintoestand brengen. Druk op RESET, of onderbreek de stroomtoevoer (aan/uit-knop, stekker) gedurende 10 seconden. Ga vervolgens naar MSX-BASIC (zie hoofdstuk 2).
Tik nu:

```
PRINT "Ik laat een zinnetje zien."
```

en druk op de RETURN-toets.
U ziet, de computer voert onmiddellijk uit wat u hem opdraagt, namelijk het afdrukken (PRINT) van een korte tekst.
Heeft u door gebruik te maken van de SHIFT- en eventueel de CAPS-toets op precies dezelfde plaatsen als hierboven hoofdletters gebruikt? Goed voor de vingervlugheid! De MSX houdt er echter rekening mee dat uw tikvaardigheid nog niet zo groot is. BASIC-opdrachten mogen zowel in hoofdletters als in kleine letters ingetikt worden. Probeert u maar:

```
print "Ik laat een zinnetje zien."
```

Door middel van de RETURN-toets geeft u de machine te kennen dat u klaar bent met het intikken van een regel met opdrachten en dat de computer deze moet uitvoeren (bij direkte opdrachten) of in zijn geheugen moet opslaan (bij indirekte opdrachten).
De aanhalingstekens aan weerszijden van de tekst laten de computer weten dat het daartussen geplaatste niet bedoeld is als BASIC-opdracht, maar als tekst die geheel moet worden afgedrukt. Tikt u ter controle maar:

```
PRINT Ik laat een zinnetje zien.
```

en druk op de RETURN-toets.
Nu begrijpt de computer toch niet helemaal wat de bedoeling was. Hij voert weliswaar een opdracht uit, maar het resultaat is iets

heel anders dan de gebruiker in gedachte had. De MSX drukt namelijk een 0 af in plaats van het zinnetje. Dit komt doordat de computer door het ontbreken van de aanhalingstekens de zin niet meer als een rij karakters (een string) beschouwt, maar als variabele (zie hoofdstuk 6). De waarde van de variabele is 0, dus dat wordt afgedrukt.
De MSX voert alleen goed opgegeven opdrachten uit. Vergelijkt u de volgende twee opdrachten maar:

```
BEEP
BEEEP
```

of:

```
COLOR 15,6,6
CLOR 15,6,6
```

Bij de onderste van elk van deze opdrachten krijgt u een bliep te horen en verschijnt de tekst:

```
Syntax error                    (grammatika-vergissing)
```

Dit betekent dat de MSX de opdracht niet begrijpt en daarom ook niet uitvoert. Het bliepsignaal is om u hierop extra attent te maken. De BEEP-opdracht zorgt voor een bliep en de COLOR-opdracht bepaalt de kleur van letters, scherm en randgebied. In bovenstaand voorbeeld werd de kleur van het scherm en van de rand veranderd van donkerblauw in donkerrood.

Fouten verbeteren

Bij het verbeteren van fouten zijn de pijltoetsen rechts van het toetsenbord en de andere opmaaktoetsen bijzonder handig. De cursor (het witte blokje dat aangeeft waar het eerstvolgende karakter geplaatst zal worden) staat nadat u bovenstaande voorbeelden heeft ingetikt onder het woordje Ok, dat aangeeft dat de computer klaar staat voor nieuwe invoer. Als u op de grote pijltoets naar boven drukt, komt de cursor op de 0 van Ok te staan. Deze letter is nu in inverse (omgekeerde kleuren) weergegeven. Druk nog 7 keer op deze toets en de cursor staat op de B van BEEEP. Verplaats de cursor met de pijl-rechts-toets naar een van de E's. Verwijder deze met de DEL (DELETE = verwijder) toets. Druk nu op RETURN. De opdracht wordt volbracht, een bliep klinkt, en onder de opdracht verschijnt de cryptische tekst:

```
Okntax error
```

U zult al hebben begrepen dat de oude tekst 'Syntax error' blijft staan, en de computer de nieuwe tekst 'Ok' er overheen schrijft. Een deel van de oude tekst blijft zichtbaar. Probeer op dezelfde wijze de mislukte COLOR-opdracht te verbeteren. Gebruik in plaats van DEL om een karakter te verwijderen, nu de INS (INSERT = voeg in) toets om een karakter tussen te voegen (na gebruik wordt INS weer uitgeschakeld door INS nogmaals in te drukken, door RETURN in te drukken of door een pijltoets in te drukken). Probeer daarna met behulp van de toetsen een andere kleur in te vullen (vervang elke 6 door een 4 voor donkerblauw, in een 8 voor middelrood en in een 10 voor donkergeel). Bij het overtikken van karakters hoeft u zowel de DEL als de INS niet te gebruiken.
Probeer de volgende drie opdrachten ook uit:

```
PRINT "Ik laat een zinnetje zien.
PRINT Ik laat een zinnetje zien."
? "Ik laat een zinnetje zien."
```

Gebruik hierbij de opmaaktoetsen om het intikken te beperken.
Bij de eerste fout reageert de computer niet. De zin wordt gewoon afgedrukt. Handig als u eens iets vergeet, maar in een programmaregel met meer dan een opdracht kan dit vervelende effecten teweeg brengen. In het tweede voorbeeld ziet de computer weer een variabele in de tekst. Wen u dus aan om een string karakters altijd tussen aanhalingstekens aan beide zijden te zetten. Dat is de enige manier waarop de computer foutloos zal reageren.
Het derde voorbeeld laat zien dat in plaats van opdracht PRINT ook het korte ? ingetikt kan worden. Ook na deze vorm van de PRINT-opdracht moeten aanhalingstekens gebruikt worden!

Vergeet u bij het intikken van opdrachten en programmaregels met opdrachten de volgende punten niet:
Alle toetsen zijn auto-repeat (=zelf herhalend). Dat wil zeggen dat als u een toets een tijdje ingedrukt houdt, deze zichzelf gaat herhalen. Dit geldt ook voor de pijltoetsen.
Als u het geluid van uw televisie of monitor aan heeft staan, hoort u dat elke toetsaanslag een tikje laat horen. U kunt hiermee op het gehoor controleren of u de toets goed ingedrukt heeft. Elke foutmelding door de computer wordt begeleid door een bliepje.
Elke (regel met een) opdracht moet afgesloten worden door op de RETURN-toets te drukken. Hierdoor wordt de opdracht de computer ingevoerd, en kan de computer aan de uitvoering beginnen, of de opdracht, of de regel met opdrachten opslaan in het geheugen.
Vanaf hier wordt kortheidshalve het indrukken van de RETURN-toets niet aangegeven. U moet na elke opdracht, of na elke programmaregel deze toets indrukken!

Opdrachten

We proberen nog een opdracht:

```
PRINT 13+6
```

Zonder dat u uw MSX ook maar iets heeft verteld kan hij al optellen. En niet alleen dat, hij kan alles wat uw rekenmachine ook kan. (Bewaart u die rekenmachine toch nog maar even. Die is immers iets compacter en handiger te vervoeren dan een MSX met televisie.)
Probeert u maar:

```
PRINT 12-6
PRINT 12/6
PRINT 12*6
PRINT 12^6
```

U ziet: perfectie! Let u wel op de iets afwijkende notatie in BASIC. Het vermenigvuldigingsteken is een asterisk (*). Het deelteken is een schuine streep (/). Machtsverheffen noteert u niet door de exponent schuin boven het te verheffen getal te plaatsen, maar door een pijltje of dakje omhoog (^) gevolgd door de exponent.
De getallen worden allemaal een spatie naar rechts geplaatst. Dit is om te zorgen dat daar het teken van het getal geplaatst kan worden. Van positieve getallen (groter dan nul) wordt het plusteken niet weergegeven. Getallen onder de nul worden voorafgegaan door een minteken (-) om aan te geven dat het negatieve getallen zijn. Probeert u zelf 12 van 6 af te trekken.
Natuurlijk kan de MSX ook wel lastiger berekeningen aan:

```
PRINT 13*6+2*5^2/4+13^2*13
```

Krijgt uw MSX er ook tweeduizendtweehonderdzevenentachtig en een half uit?
De MSX houdt bij berekeningen keurig de volgorde aan die normaal is in de rekenkunde: 'Meneer Van Dalen Wacht Op Antwoord'. Eerst Machtsverheffen (^), dan Vermenigvuldigen en Delen (* en /), dan Worteltrekken (SQR) en dan Optellen en Aftrekken (+ en -). Bij gelijkwaardige bewerkingen zoals optellen en aftrekken werkt de MSX van links naar rechts.
Als u een bepaalde bewerking voor een andere wilt laten uitvoeren kunt u gebruik maken van haakjes:

```
PRINT 3*5^2
PRINT (3*5)^2
```

In het bovenste geval wordt 5 gekwadrateerd (25) en daarna met 3 vermenigvuldigd (75). In het tweede geval wordt 5 met 3 vermenigvuldigd (15) en het resultaat gekwadrateerd (225).

Er bestaan naast de PRINT-opdracht nog veel meer direkte opdrachten. Als het scherm na al het voorgaande wat rommelig is, tikt u gewoon:

```
CLS
```

Onmiddellijk wordt het gehele scherm gewist en verschijnen linksboven in het beeld Ok en de cursor. Hetzelfde effect, maar dan zonder Ok-melding wordt bereikt door tegelijk SHIFT en HOME in te drukken. Alleen HOME zorgt ervoor dat de cursor naar de positie linksboven springt zonder het beeldscherm te wissen.

5. EEN PROGRAMMA SCHRIJVEN

Een programma

Tikt u het volgende programma in. Na elke regel drukt u natuurlijk op RETURN. Sommige microcomputers accepteren geen regels met overduidelijk fouten. De MSX doet dit wel. Regels die in een programma gezet worden (dus geen direkte opdrachten) worden altijd geaccepteerd. Pas als u het programma laat RUNnen (laten aflopen, dat wil zeggen dat de computer alle opdrachten in het programma in volgorde uitvoert) maakt de MSX melding van gemaakte fouten.
Na de invoer van elke regel verschijnt links in beeld de cursor, u kunt de volgende regel intikken.

```
10 REM EEN VERMENIGVULDIGING
20 LET A=99
30 LET B=12
40 LET P=A*B
50 PRINT A;"*";B;"=";P
```

De cursor staat klaar voor de volgende regel. Die is er niet. U tikt:

RUN (+ RETURN)

(In plaats van RUN in te tikken als losse letters kunt u ook F5 (run) indrukken.)
Het blijkt dat de MSX ook verschillende opdrachten na elkaar kan uitvoeren. Dat had u natuurlijk al verwacht, want waar heb je anders een computer voor.
Let u bij het geven van opdrachten op de volgende punten:

- In een BASIC-programma begint elke regel met een positief geheel getal (het regelnummer).
- De computer voert de verschillende regels in volgorde van de regelnummers uit.
- Op een regel kunnen verschillende opdrachten staan (zie bijvoorbeeld regel 50: de MSX moet volgens die regel 5 verschillende zaken weergeven op het scherm. Dit is eigenlijk een opdracht met verschillende onderdelen. Later zult u echte zogenaamde multi-statement-regels (regel met meer dan een statement = opdracht, funktie of bewering) tegenkomen).
- Het is gebruikelijk bij het schrijven van een programma de regelnummers per 10 te laten verspringen. Op deze manier kunt u later nog eenvoudig regels tussenvoegen.
- U hoeft de regels niet in volgorde in te tikken. De MSX voert

ze echter wel in de juiste volgorde uit, en zet ze in volgorde in het programma.

(De opdracht LET geeft een algemene waarde -een zogenaamde variabele- een bepaalde waarde. LET A=99 wil zeggen: vanaf nu is A gelijk aan 99. LET P=A*B wil zeggen: vanaf nu is P gelijk aan A vermenigvuldigd met B. We komen in het hoofdstuk Variabelen hier nog uitgebreid op terug.)

Hulp bij het maken van programma's

Er zijn verschillende hulpjes in de MSX ingebouwd om het programmeren gemakkelijker en sneller te maken.
Om het gehele ingetikte programma in een keer te kunnen zien tikt u

```
LIST
```

(of F4) gevolgd door RETURN. Is het programma langer dan een beeldscherm vol, dan verdwijnen de bovenste regels. Probeert u dit uit door een programma te maken van 28 regels, met op elke regel alleen een ?. Ziet u dat bij het LISTen van het programma de computer de ?-en omzet in de opdracht PRINT? Doordat de regels bovenaan het scherm weer zo snel verdwijnen is het erg lastig een lang programma te lezen. Om rustig steeds een deel van de programmalisting te lezen gebruikt u de STOP-toets. Als u deze indrukt, stopt het doorrollen (scrollen) van de listing. Door nogmaals op STOP te drukken gaat de listing weer verder. Zo kunt u elk deel van de listing tevoorschijn halen, bekijken en eventueel verbeteren.
Wilt u slechts een enkele regel uit het programma zien, dan tikt u:

```
LIST regelnummer
```

Voor regelnummer tikt u het nummer van de regel die u wilt zien. Om een paar regels te zien kunt u een van de volgende opdrachten tikken:

```
LIST beginregel - eindregel
LIST - eindregel
LIST beginregel -
```

Deze opdrachten zorgen er achtereenvolgens voor dat het programma vanaf beginregel tot en met eindregel gelist wordt, dat vanaf het begin van het programma tot en met de eindregel gelist wordt, en dat vanaf de beginregel tot het einde van het programma gelist wordt.

Heeft u bij de eerste keer intikken van een programma keurig de regels met 10 laten verspringen, maar komt u door verbeteringen of toevoegingen regelnummers tussen twee regels tekort, dan kunt u het programma hernummeren (ook te gebruiken als u de programmalisting te slordig vindt, met al die cijfers tussen de tientallen in). U gebruikt daarvoor een van de volgende opdrachten (van RENUMBER = hernummer):

```
RENUM
RENUM eerste nieuwe regelnummer
RENUM eerste nieuwe regelnr.,eerste oude regelnummer
RENUM , , stapgrootte
RENUM 1-e nw rglnr , 1-e oud rglnr , stapgrootte
```

Achter de opdracht RENUM kunnen drie waarden ingevuld worden (geen van alle verplicht). De eerste waarde geeft de hoogte van het eerste nieuwe regelnummer aan. De tweede waarde geeft de eerste te hernummeren oude regel aan. De derde waarde geeft de stapgrootte tussen de nieuwe regelnummers aan.
Alleen RENUM hernummert alle regels van het programma, beginnend bij 10, met een verschil tussen de regels van 10. Dat wil zeggen dat de nieuwe regelnummers 10, 20, 30, 40, etc. worden.
Door de eerste waarde in te vullen, beginnen de nieuwe regelnummers niet bij 10, maar bij een opgegeven getal. Zo geeft RENUM 200 de regelnummers 200, 210, 220, 230, etc.
Door de tweede waarde in te vullen wordt de eerste te hernummeren regel op te geven. De regels daarvoor worden niet hernummerd. Alle regels erna wel. Als eerste nieuwe regelnummer wordt de eerste opgegeven waarde gebruikt.
Door de derde waarde in te vullen wordt de stapgrootte bepaald. Zo geeft RENUM , , 100 de regelnummers 100, 200, 300, 400, etc. (De twee komma's achter RENUM geven aan dat de eerste twee waarden niet ingevuld zijn.)
De tweede ingevulde waarde mag niet groter zijn dan de eerste ingevulde waarde. In dat geval zou de computer over de oude regelnummers heen gaan nummeren.

Om het telkens intikken van een regelnummer te voorkomen kan de computer gevraagd worden om telkens een regelnummer te genereren. Daarvoor gebruikt u de opdracht:

```
AUTO eerste regelnummer , stapgrootte
```

Met het eerste regelnummer geeft u aan welk regelnummer de computer eerst moet genereren. Vult u niets in, dan kiest de computer 10. Met de stapgrootte geeft u aan hoeveel verschil er tussen de regelnummers moet zitten. Geeft u niets op, dan kiest

de computer 10.
U stopt het automatisch nummeren met CTRL en STOP of door CTRL en C.
Als de computer een regelnummer genereert van een regel waar al wat staat, dan wordt dit aangegeven door een asterisk (*) rechts van het regelnummer. Wilt u de regel niet veranderen dan drukt u gewoon op RETURN, waarna de volgende regel verschijnt. Wilt u de regel wel veranderen, dan verwijdert u de asterisk met de pijl-toets, en tikt u de nieuwe regel in. Om de regel te zien gebruikt u LIST.

Een programma maken

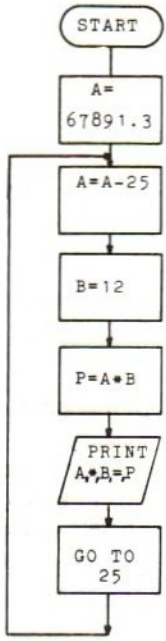
Nu we de mogelijkheid hebben om verschillende opdrachten na elkaar uit te laten voeren, waarbij we weten dat de computer zich houdt aan de regelnummers, hebben we de beschikking over een systeem met verschillende voordelen. Tik:

```
20 LET A=67891.3
```

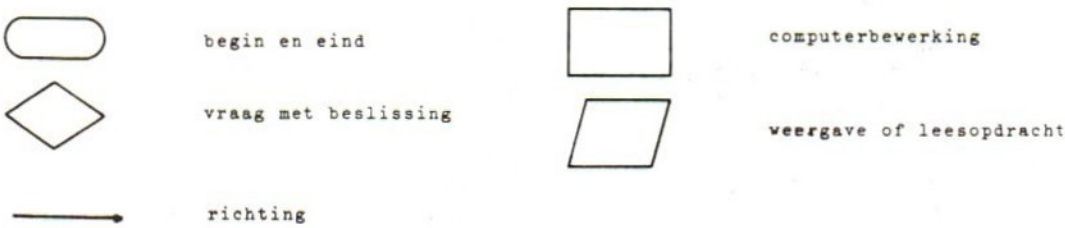
Na het indrukken van RETURN is de oude regel 20 vervangen door de nieuwe. Niets wijst daar echter op. Om het te controleren moet u LIST tikken.
Door het nummeren van de regels zijn deze aanwijsbaar en herkenbaar geworden. We kunnen nu telkens een regel veranderen. We kunnen ook verwijzen naar een bepaalde regel. Voorbeelden daarvan zijn de hiervoor genoemde hulpen als LIST, RENUM en AUTO. Maar ook binnen een programma kan verwezen worden. Tik:

```
25 LET A=A-25
60 GOTO 25
```

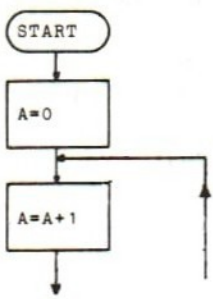
Laat het programma lopen (RUN, RETURN). De MSX geeft een lange lijst sommen met uitkomsten weer. U geeft in regel 60 immers de opdracht om weer terug te gaan naar regel 25 (GOTO = ga naar). Daar aangekomen vervolgt de MSX zijn weg door het programma van laag regelnummer naar hoog regelnummer, om uit te komen bij regel 60, die hem weer terugstuurt naar regel 25. Om het programma te stoppen gebruikt u CTRL en STOP. Om het programma even stil te zetten gebruikt u alleen STOP.
In normaal Engels zijn 'go' en 'to' twee aparte woorden. MSX-BASIC maakt geen onderscheid tussen 'GOTO' en 'GO TO'.
Door de regelnummers zijn zogenaamde lussen mogelijk geworden. Als we bovenstaand programma netjes opschrijven in een stroomdiagram (een schema dat de loop of de stroom van een programma weergeeft) krijgen we:

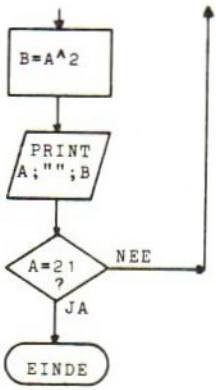


Dit is een eenvoudig stroomdiagram, maar zodra u een iets ingewikkelder probleem moet oplossen is het handig om voordat u het programma schrijft, eerst een stroomdiagram te tekenen. De belangrijkste tekens die in een stroomdiagram gebruikt worden zijn:



Als voorbeeld nemen we het volgende probleem: maak een lijst van de kwadraten van alle getallen van 1 tot 21. Het stroomdiagram kan er dan zo uitzien:





Als dit omgezet wordt in een programma krijgt u onderstaand resultaat.
Het oude programma krijgt u uit het geheugen van de MSX door de direkte opdracht NEW (=nieuw). Door deze opdracht worden oude programma's en alle gegevens uit het geheugen van de MSX gehaald. Hetzelfde effect is ook te bereiken door de RESET-toets in te drukken. Dit duurt echter langer en heeft het nadeel dat bij deze methode ook eventuele in het geheugen aanwezige machinetaalprogramma's gewist worden. Deze blijven bij NEW intakt.

```
10 REM kwadraten van 1 tot en met 20
20 LET A=0
30 LET A=A+1
40 LET B=A^2
50 PRINT A;" in het kwadraat is ";B
60 IF A<21 THEN GOTO 30
70 PRINT "EINDE"
```

In regel 60 laten we de MSX testen of A nog steeds kleiner is dan 20 (A<20). Zo nee, dan is A gelijk aan 20, en is het kwadraat daarvan net uitgerekend. Zo ja, dan moet het programma nog een keer doorlopen worden en wordt de machine terug gestuurd.
Het gebruik van stroomdiagrammen is geen overbodige luxe voor mensen die veel te precies zijn. Zolang u korte eenvoudige programma's schrijft lijkt het dat het stroomdiagram zo vanzelfsprekend is, dat het opschrijven ervan overbodig is. Doet u het toch. De oefening die u opdoet met het maken van stroomdiagrammen voor eenvoudige programma's heeft u nodig voor het maken van stroomdiagrammen van ingewikkelder programma's. Probeert u bij elk door u gebruikt programma in dit boek zelf een stroomdiagram te construeren.
Maakt u vooral veel gebruik van de opdracht REM (REMARK = opmerking). Hiermee kunt u in de listing van uw programma opmerkingen maken ter verduidelijking van het programma. Alle

tekst na een REM-opdracht wordt door de computer niet uitgevoerd. Probeer dus niet na een REM-opdracht een andere opdracht in dezelfde programmaregel te zetten .

```
10 REM wis het scherm : CLS
```
Dit kan niet. De computer negeert de CLS-opdracht.

```
10 CLS : REM wis het scherm
```
Dit kan wel. De computer wist het scherm.

Let op de methode waarop meer dan een opdracht in een programmaregel gezet wordt: de opdrachten worden onderling gescheiden door een dubbele punt (:).
U kunt het woord REM ook vervangen door een apostrofe ('). Als u het programma nu later LIST, wordt de apostrofe niet omgezet in het woord REM (vergelijk de ? voor PRINT). Als u de REM-opdracht gebruikt als laatste opdracht binnen een programmaregel, kunt u de dubbele punt (:) om de opdracht te scheiden weglaten als u de apostrofe gebruikt:

```
10 CLS REM wis het scherm
```
Mag niet (syntax error).

```
10 CLS ' wis het scherm
```
Kan wel.

Hierboven zag u een klein programma met een enkele vertakking. Hieronder vindt u een programma met iets meer vertakkingen. Het programma zet graden Fahrenheit om in graden Celcius, of omgekeerd. De gebruikte formules zijn:

$$°F = 9/5 * °C + 32$$
$$°C = 5/9 * (F - 32)$$

Het programma wordt:

```
10 REM graden omzetten
20 CLS
30 PRINT : PRINT "Dit programma zet graden F om in
   graden  C of andersom."
40 PRINT : PRINT
50 PRINT "Wilt u van F naar C (0),"
60 PRINT "of van C naar F (1)?"
70 PRINT : PRINT "Tik 0 of 1, RETURN."
64 REM keuze
90 INPUT X
100 IF X=0 THEN GOTO 180
110 REM als x=1 gaat prog. hier verder
120 INPUT "Hoeveel graden? ";C
130 LET F=(9/5)*C+32
140 PRINT : PRINT C;"gr Celc.= ";F;"gr Fahr."
150 PRINT
160 REM de twee delen komen bij elkaar
```

```
170 GOTO 230
180 REM nu van F naar C
190 INPUT "Hoeveel graden? ";F
200 LET C= (5/9)*(F-32)
210 PRINT: PRINT F;"gr Fahr.=";C;"gr Celc."
220 REM nu volgt automatisch 230
230 PRINT : PRINT "Wilt u nog een keer?"
240 INPUT "Tik 1 (ja) of 0 (nee), RETURN";Y
250 REM na keuze 1 (ja) terug naar begin
260 IF Y=1 THEN CLS : GOTO 40
270 REM bij keuze 0 (nee),einde
280 PRINT : PRINT
290 PRINT "Tot de volgende keer !"
```

Tik dit programma in en zet het daarna op een cassettebandje. Volg daarvoor de instrukties uit de handleiding. Tik:

```
CSAVE "xxxxxx"                    (bewaren)
```

Zet tussen de aanhalingstekens een naam die u aan het programma wilt geven. De naam mag maximaal 6 tekens bevatten en moet met een normale letter beginnen.
Druk op de cassetterecorder de REC-(opname) of SAVE-(bewaar) en de PLAY- knop in. De cassetteband gaat nu lopen.
Druk nu pas op RETURN! Zodra deze toets ingedrukt wordt, gaat de computer signalen naar de band sturen.
Zodra de computer het gehele programma overgestuurd heeft, verschijnt de vertrouwde boodschap Ok.
Nu kan de STOP-knop op de cassetterecorder ingedrukt worden.
Als u een cassetterecorder heeft met afstandsbediening en deze aangesloten is door middel van de kleine zwarte stekker, zal de computer het aan en uitzetten van de band regelen. U moet zelf wel opletten of de REC- en PLAY-knop ingedrukt zijn. Het enige dat de computer kan sturen is het draaien van de motor. De stand van de knoppen (afspelen of opnemen) kan door de computer niet geregeld worden.
Om het programma later voor gebruik weer in de MSX te laden spoelt u de cassette versneld terug naar de plaats waar het programma staat. De volgende twee methoden voldoen:

1. u gebruikt voor elk programma een nieuw bandje. U heeft daardoor weinig spoelwerk, maar wel wat kosten voor bandjes.
2. u noteert op het papier bij het bandje bij welke nummers van de bandteller een bepaald programma begint.

Om het programma te laden (dat wil zeggen van de band in het geheugen van de computer te zetten) tikt u:

```
CLOAD "xxxxxx"                    (laden)
```

Nu zet u de cassetterecorder op afspelen (let op dat u niet de opnameknop indrukt!). De computer luistert nu naar het signaal

dat van de cassetterecorder komt, en noteert welke programma's hij tegenkomt. Als hij het juiste programma tegenkomt (opgegeven met de naam op de plaats van xxxxxxx), zal hij dat in het geheugen plaatsen.
Weet u niet welk programma op het klaarstaande stukje band staat, of bent u niet zeker van de juiste spelling van het programma, dan tikt u:

CLOAD

De computer zal nu het eerste programma dat hij op de cassette tegenkomt laden.
U kunt CLOAD ook gebruiken om onmiddellijk na het SAVEn van een programma te controleren of het programma goed op de band is gekomen. U spoelt de cassette terug naar het begin van de opname en tikt:

CLOAD? "xxxxxx" (controleren)

De computer vergelijkt nu het programma xxxxxx op de band met het programma xxxxxx in het geheugen. Als de programma's gelijk zijn, is het programma goed op de cassette gekomen, en meldt de computer Ok. Als de programma's niet overeenkomen meldt de computer:

Device I/O error (fout in in- of uitvoer)

U moet nu de band weer terugspoelen naar het begin van de opname en het programma opnieuw SAVEn (opnemen). Zie hiervoor.
Als de naam van het te controleren programma wordt weggelaten, vergelijkt de computer het eerste programma dat hij op de band tegenkomt met het programma in zijn geheugen.

Het laden en lezen van programma's naar en van diskette gaat op vergelijkbare wijze. Zorg eerst voor een geformatteerde diskette (zie handleiding bij de diskette). De opdracht SAVE "xxxxxx" zet het programma op de diskette. De naam van het programma mag niet weggelaten worden! De opdracht LOAD "xxxxxx" leest het programma van diskette in het geheugen. De naam van het programma kan niet weggelaten worden!
Bij een cassettesysteem is het niet erg om voor verschillende programma's dezelfde naam te hebben. Het is weliswaar niet aan te raden, want het kan gemakkelijk tot verwarring leiden, maar de computer zal de programma's gewoon lezen en wegzetten. Bij een diskettesysteem moeten alle programma's op een diskette verschillende namen hebben. Probeert u een nieuw programma met dezelfde naam als een oud programma op diskette te zetten, dan wordt het oude programma gewist. Om te zien welke programma's

zich op de diskette bevinden gebruikt u de opdracht FILES (bestanden). Alle bestanden en programma's worden nu weergegeven.

In het gradenprogramma zijn een paar nieuwe instrukties gebruikt. De INPUT en IF...THEN.. kwamen niet eerder in dit boek voor.
De funktie van INPUT is de gebruiker zelf een getal, getallen, letter of letters in te laten tikken.
Door middel van IF...THEN... springt de computer afhankelijk van een bepaalde voorwaarde naar een ander deel van het programma. Op deze manier kunnen we de twee keuzemogelijkheden van het programma (F of C en 'nog een keer') opgeven aan de computer. Omdat het hier echter alleen om het vertakken (met daarbij een leuk programma) ging, verwijzen we u voor de uitleg van deze instrukties naar de komende hoofdstukken.
Nog een enkele tip. Door een enkele PRINT-opdracht te gebruiken krijgt u een beter leesbaar beeldscherm. Daarbij zorgt zo'n opdracht voor extra ruimte in de programmalisting, waardoor deze beter leesbaar is. Als u alleen meer ruimte in de programmalisting wilt, kunt u gebruik maken van de apostrofe (= REM) zonder enig commentaar. U kunt de verschillende onderdelen binnen een programma zo scheiden.

PS: Een aardig getal om in te tikken in het programma is -40.

6. VARIABELEN

Dit hoofdstuk is vooral bedoeld voor diegenen die weinig of geen wiskundige kennis bezitten. Voor hen geeft dit hoofdstuk een summiere uitleg van het gebruik van variabelen, onmisbaar bij het programmeren. Weet u hier voldoende van af, leest u dit hoofdstuk dan vluchtig door, er staan enkele wetenswaardigheden in m.b.t. het verschil in gebruik van variabelen in de wiskunde en bij het programmeren.
Over rekenen heeft u al eerder in dit boek kunnen lezen. Enkele voorbeelden:

```
2 + 2 = 4
6 - 3 = 3
2 X 2 = 4
6 : 2 = 3
```

Met deze manier van rekenen valt ook door middel van de computer veel uit te rekenen. Maar het wordt wat lastig als bepaalde getallen in een programma telkens veranderen. Elke keer zou daarvoor het programma geLIST en de getallen veranderd moeten worden.
De som 5 X 4 = 20 is goed uitgerekend, maar het is niet interessant om dit door de computer uit te laten rekenen. De som wordt al een stuk interessanter als u weet dat de getallen eigenlijk staan voor:

```
Lengte X Breedte = Oppervlakte
```

Met de hier gegeven 'formule' is het mogelijk veel meer getallenvoorbeelden te gebruiken. De eerste som zegt alleen maar dat 5 maal 4 gelijk is aan 20. De tweede som vertelt veel meer. Voor elke rechthoek geldt dat de lengte maal de breedte gelijk is aan de oppervlakte. Een rechthoek met een lengte van 5 en een breedte van 4 heeft een oppervlakte van 20. Wat hebben we hier gedaan? De woorden Lengte, Breedte en Oppervlakte zijn tijdelijk vervangen door 4, 5 en 20.
Er kunnen voor dezelfde woorden ook andere getallen ingevuld worden:

```
6 (Lengte) X 2 (Breedte) = 12 (Oppervlakte)
```

Als we nu de som in de vorm Lengte X Breedte = Oppervlakte aan de computer opgeven en vervolgens telkens aan de computer vertellen hoe groot de Lengte en hoe groot de Breedte is, kan de computer telkens opnieuw de Oppervlakte van een rechthoek uitrekenen.
Op deze wijze zijn de vaste getallen vervangen door 'getallen' die kunnen varieren, veranderen (zogenaamde VARIABELEN). De formule kan nu algemeen gebruikt worden. Er hoeft nu immers maar

een keer aan de computer opgegeven te worden wat er met de variabelen moet gebeuren, waarna er telkens alleen de nieuwe waarden voor de variabelen opgegeven hoeven te worden.

Lengte X Breedte = Oppervlakte

De computer krijgt opgegeven dat Lengte=5 en Breedte=4.
De computer vult de getallen in. In zijn geheugen staat nu:

5 X 4 = Oppervlakte

Als de computer om het resultaat van de vergelijking gevraagd wordt, zal hij antwoorden 20 (Oppervlakte = 20).
Het grote voordeel van het rekenen met variabelen in plaats van vaste getallen is dat variabelen op een eenvoudige wijze ook andere waarden kunnen krijgen. Zo kan de computer ook opgegeven worden dat Lengte=7 en Breedte=5. De computer vult de vergelijking weer in en in het geheugen staat dan:

7 X 5 = Oppervlakte

Net zoals u kunt rekenen met getallen kunt u ook rekenen met variabelen. Dit heet dan algebra.

Nog een voorbeeld van het gebruik van variabelen.
Laten we aannemen dat u elke week een kiosk binnen stapt om daar een aantal kranten en tijdschriften te kopen. Elke krant kost 2 gulden en elk tijdschrift kost 5 gulden. De prijs die u aan de kassa moet betalen is natuurlijk afhankelijk van het aantal kranten en het aantal tijdschriften dat u koopt. Voor 3 kranten en 4 tijdschriften betaalt u:

3 X 2 gulden + 4 X 5 gulden = 26 gulden

Voor 4 kranten en 7 tijdschriften betaalt u:

4 X 2 gulden + 7 X 5 gulden = 43 gulden

Als u met de computer wilt uitrekenen hoeveel u moet betalen hoeft u niet elke keer de hele rekensom in te tikken. U vertelt de computer met welke formule hij die prijs moet uitrekenen en verder tikt u alleen elke week in welk aantal kranten en welk aantal tijdschriften u wilt kopen. De computer rekent dan zelf de prijs uit. U geeft de computer de volgende formule op:

prijs = aantal kranten X 2 + aantal tijdschriften X 5

of wat korter:

PR = K X 2 + T X 5

Als u vervolgens de prijs wilt weten van 3 kranten en 4 tijdschriften, geeft u aan de computer op: K=3 en T=4. De computer vult vervolgens zelf de waarden in: PR = 3 X 2 + 4 X 5 en geeft als antwoord: PR = 26.
K en T zijn variabelen: we kunnen hiervoor steeds een ander getal invullen.
Let bij deze voorbeelden op het vermenigvuldigingsteken. Dit wordt bij computers aangegeven met een asterisk (*).
Enkele punten vragen bij het gebruik van variabelen wel speciale aandacht:
Bij het programmeren mag niet zo maar elke naam aan een variabele gegeven worden. Er mogen geen spaties in de naam voorkomen. De naam kan maximaal 255 karakters lang zijn, maar alleen de eerste twee letters worden door de computer gebruikt. De tweede letter van een variabelenaam mag elk letterteken zijn, maar de eerste letter moet een letter uit het alphabet zijn. De variabelenaam mag niet gelijk zijn aan een BASIC-sleutelwoord. BASIC-sleutelwoorden zijn alle woorden die in BASIC een funktie vervullen. Zo is de variabelenaam TOETER niet toegestaan omdat deze begint met het BASIC-woord TO.
In de algebra mag u het vermenigvuldigingsteken bij het gebruik van variabelen weglaten. Bij het programmeren mag dit niet. De computer beschouwt 'ab' als de variabele met de naam 'ab', en 'a*b' als het produkt van de variabele 'a' met de variabele 'b'.
U kent een waarde aan een variabele toe met behulp van de BASIC-opdracht

LET

Bijvoorbeeld LET b = 5. Hierdoor wordt aan de variabele 'b' de waarde '5' toegekend.
Daarnaast kan aan een variabele ook een waarde toegekend worden tijdens het RUNnen van het programma door verschillende later in dit boek te behandelen opdrachten als:

INPUT

Bijvoorbeeld INPUT b. Hierdoor wordt aan de variabele 'b' de waarde toegekend die de gebruiker na het verschijnen van een vraagteken op het beeldscherm intikt en invoert door middel van RETURN.
Een variabele kan ook een variabele waarde toegekend krijgen d.m.v. LET. Bijvoorbeeld LET a = b + 3. Hierdoor krijgt de variabele 'a' de waarde 'b + 3'. Let op! Dit is alleen mogelijk als de computer al weet wat de waarde van 'b' is. U moet de computer dit dus van te voren mededelen. Ofwel door middel van een direkte opdracht, of doordat de waardetoekenning aan 'b'

plaatsvindt in het computerprogramma voor de waardetoekenning aan 'a'. Dit geldt in het algemeen voor alle microcomputers. Bijna elke microcomputer reageert op het gebruiken van een variabele voordat daaraan een waarde is toegekend met een foutmelding (meestal: no such variable = zo'n variabele is er niet, of variable unknown = variabele onbekend). Schrijft u dus computerprogramma's die later eventueel door anderen gebruikt moeten worden, dan dient u hierop te letten.
De MSX heeft van deze regel echter geen last. Als u een variabele gebruikt die nog geen waarde toegekend heeft gekregen, gaat de MSX er vanuit dat de variabele de waarde 0 heeft. Maakt u de machine leeg door RESET en tikt u:

```
LET A = B + 3
PRINT A
```

De MSX geeft als antwoord 3. Enerzijds is dit heel handig, want u krijgt nooit problemen met het toekennen van waarden aan variabelen, maar anderzijds kan dit heel gevaarlijk zijn. Maak er een gewoonte van alle variabelen eerst een waarde toe te kennen voordat u ze verder gebruikt.
Bij het toekennen van waarden aan variabelen moet het gelijkteken (=) niet beschouwd worden als deel uitmakend van een bewering:

```
LET A = A + 1
```

Dit dient gelezen te worden als: de waarde van de variabele 'a' is vanaf nu gelijk aan de oude waarde van 'a' plus 1. Of anders gezegd: tel bij de waarde van de variabele 'a' een op en laat het resultaat de nieuwe waarde van 'a' zijn. Het moet niet gelezen worden als 'a' is gelijk aan 'a+1'. Dit is immers een onware bewering.
Het woordje LET is in MSX-BASIC niet verplicht om de waarde van een variabele op te geven. U kunt dus ook tikken:

```
A = 10
```

De waarden van variabelen kunnen zowel getallen (numerieke variabelen) als reeksen van letters (stringvariabelen) zijn. Over de stringvariabelen kunt u meer lezen in het hoofdstuk over strings.
MSX kent drie soorten numerieke variabelen: variabelen waarin alleen gehele getallen opgeslagen kunnen worden (integers), en variabelen waarin reeele getallen kunnen worden opgeslagen, onderverdeeld in opslag met enkele of dubbele precisie.
In een integer variabele kan alleen een geheel getal opgeslagen worden. De minimale waarde die opgeslagen kan worden is -32768. De maximale waarde is 32767. Een integervariabele wordt

onderscheiden van andere numerieke variabelen door het procentteken (%) achter de naam van de variabele. Bijvoorbeeld:

```
A% = 8
```

In een numerieke variabele met enkele precisie kan een getal met maximaal 6 cijfers (voor of achter de komma) opgeslagen worden. Worden de getallen te groot, dan wordt het teveel aan cijfers afgerond en noteert de MSX het getal in de vorm van een exponent met grondtal 10. Bijvoorbeeld:

```
A! = 9.69974E+17
```

Het exponentiele gedeelte kan lopen van -64 tot +62. Enkele precisie variabelen worden aangegeven met een uitroepteken (!) achter de naam van de variabele.
Numerieke variabelen met dubbele precisie worden aangegeven met een Amerikaans aantalsymbool of matje (#) achter de naam van de variabele. Deze variabelen kunnen getallen bevatten met maximaal 14 cijfers. Net als bij getallen met enkele precisie wordt het teveel aan cijfers eventueel afgerond, en het getal genoteerd met een E:

```
A# = 9.6997445234213E+17
```

Als u de MSX niets anders opgeeft, wordt van elke numerieke variabele aangenomen dat dit een dubbele-precisie variabele is. Probeert u maar:

```
RESET                          (naar MSX-BASIC)
LET A=1
LET B=A/3
? B
```

Hierboven las u dat bepaalde variabelen aangegeven worden met bepaalde tekens: integer (%), enkele precisie (!), dubbele precisie (#) en de stringvariabele zult u nog tegenkomen met een dollarteken ($). Als u niets opgeeft achter de naam van een variabele neemt de computer aan dat u een numerieke variabele met dubbele precisie bedoelt.
Het is echter ook mogelijk om variabelen te benoemen. Na dit benoemen hoeft het teken niet meer gebruikt te worden achter de naam van de variabele omdat de computer onthoudt in welke groep van variabelen desbetreffende variabele thuis hoort. De instrukties die hiervoor gebruikt worden zijn:

```
DEFINT letter (-letter)        Integer = geheel getal
DEFSNG letter (-letter)        Single  = enkele precisie
DEFDBL letter (-letter)        Double  = dubbele precisie
```

DEFSTR letter (-letter) String = sliert lettertekens

Bij letter vult u de eerste letter in van de namen van de variabelen de u wilt benoemen. Eventueel kunt u door middel van een streepje en een tweede letter een heel groep variabelen beginnend met een van de opgegeven letters benoemen. Bijvoorbeeld:

```
DEFINT A
```

Deze opdracht zorgt ervoor dat elke variabele beginnend met een A een integer variabele is.

```
DEFINT A - L
```

Deze opdracht zorgt ervoor dat elke variabele beginnend met een van de letters A tot en met L een integer variabele is.

```
DEFSNG R - W
```

Deze opdracht zorgt ervoor dat elke variabele beginnen met een van de letter R tot en met W een enkele-precisie variabele is.
Als een variabele door een DEF-opdracht als hierboven benoemd is, kan hij toch een andere funktie krijgen door gebruik van een van de tekens achter de variabelenaam. Bijvoorbeeld:

```
DEFINT A - L
LET E = 1.456
? E                                   Resultaat: 1
LET E! = 1.456
? E!                                  Resultaat: 1.456
```

De variabeletekens gaan voor de benoeming door DEF. De benoeming van DEF gaat voor de algemene aanname dat alle variabelen dubbele-precisie variabelen zijn.
Een teveel aan cijfers wordt bij het toekennen van een waarde aan een variabele van de verkeerde soort afgekapt. Te weinig cijfers worden niet aangevuld. Vergelijk:

```
LET G% = 1.9555
? G%                                  Resultaat: 1
LET Q# = 1.3
? Q#                                  Resultaat: 1.3
```

De veiligste en mooiste methode van werken met variabelen is om aan het begin van een programma alle variabelen te benoemen. Van elke te gebruiken variabele ligt daarmee de soort vast. Zodoende kunnen minder vergissingen gemaakt worden.

7. INVOER EN BEELDWEERGAVE

Input

Om iets te kunnen doen met de computer moeten we iets kunnen invoeren (INPUT). Om te zien wat de computer doet of wat hij gedaan heeft, is het nodig dat de computer mededelingen kwijt kan (OUTPUT). De MSX-computer doet dit meestal via een televisie of een monitor.
Een deel van de mogelijkheden van het invoeren van gegevens heeft u reeds kunnen lezen in de hoofdstukken 'Inleiding' en 'Een programma schrijven'. Wat echter nog niet aan de orde is geweest is het invoeren van gegevens terwijl het programma aan het lopen is (zoals dat heet: terwijl het programma runt - van het engelse werkwoord run = lopen). De mogelijkheid om gegevens in te voeren tijdens de run van een programma zou verschillende voordelen hebben. Het is niet nodig dat bij elke gegevensverandering het programma stilgezet wordt om het gegeven in te voeren waarna het programma opnieuw gestart moet worden. Bij het spelen van spelletjes is het zelfs onontbeerlijk, want er moet een wisselwerking tussen u en de computer zijn. U moet tijdens de programmarun kunnen reageren op opdrachten of signalen van de computer door bepaalde gegevens op te geven. MSX-BASIC kent hiervoor de opdracht INPUT (=invoer). Tik:

```
NEW
10 INPUT A
20 PRINT A;A^2
```

RUN dit programma (tik RUN of druk op F5). Er verschijnt een vraagteken links in beeld met daarachter de cursor. De MSX vraagt u een waarde in te tikken. Tik 12. (uiteraard gevolgd door RETURN)
Op het beeldscherm verschijnen prompt 12 en 144, gescheiden door twee spaties. Eén spatie wordt veroorzaakt doordat de MSX getallen niet tegen elkaar aanzet, en de andere spatie is de ruimte de gereserveerd is voor een minteken bij een negatief getal.
De INPUT-opdracht werkt vergelijkbaar met de LET-opdracht. Door de INPUT-opdracht wacht de computer totdat u een waarde intikt. Deze waarde wordt toegekend aan de variabele genoemd achter de INPUT-opdracht. Het is ook mogelijk verschillende variabelen een waarde toe te kennen door middel van INPUT. Tik:

```
10 INPUT A,B,C
20 PRINT A;"*";B;"*";C;"=";A*B*C
```

De asterisken tussen de aanhalingstekens worden niet 'uitgerekend' door de computer, maar dienen om netjes tussen de getallen op het scherm gezet te worden. Ze staan tussen aanhalingstekens zodat de computer er niet mee gaat rekenen.
Als u dit programma runt verschijnt er weer een vraagteken met daarachter de cursor. U kunt de drie getallen nu invullen. De getallen die u opgeeft kunt u op twee manieren scheiden: door komma's of door RETURN's. In beide gevallen moet de laatste opdracht een RETURN zijn. U kunt dus tikken:

```
        1,2,3      (RETURN)
of
        1          (RETURN)
        2          (RETURN)
        3          (RETURN)
of
        1,2        (RETURN)
        3          (RETURN)
of
        1          (RETURN)
        2,3        (RETURN)
```

U ziet dat het er niet veel toe doet op welke manier u de verschillende getallen opgeeft. De enige eis die gesteld wordt is dat de getallen gescheiden zijn. Ofwel door een komma, ofwel door een RETURN.
U moet altijd het juiste aantal gegevens invoeren. Dat wil zeggen net zoveel gegevens als de computer voor het programma nodig heeft. Als u te weinig gegevens invult, zal de computer alleen een vraagteken laten zien en wachten totdat u meer gegevens invult. Vult u teveel gegevens in, dan worden de extra gegevens verwaarloosd. Vul in bovenstaand programma eens 1,2,3,4 in. De computer meldt dat hij gegevens weglaat met

```
        ?Extra ignored                    (teveel genegeerd)
```

Spaties die u (per ongeluk) voor, achter of binnen het getal invoert, worden door de computer genegeerd. Probeert u als voorbeeld de cijfers 2, 20, 3 10.

Berichten

Er is een nadeel verbonden aan het op deze wijze gebruiken van de INPUT-opdracht. Bij een klein programma als het bovenstaande is nog wel duidelijk wat er ingevuld moet worden door de gebruiker. Maar bij een langer programma is het niet denkbeeldig dat de

gebruiker vergeet wat de computer precies voor soort invoer verwacht. In het hoofdstuk variabelen heeft u immers al kunnen lezen dat er verschillende soorten variabelen bestaan, die elk verschillende soorten gegevens kunnen bevatten. Het opgegeven van een waarde die normaal gesproken niet door een variabele bevat kan worden leidt tot onverwachte resultaten.
Daarom is er de mogelijkheid om berichten te gebruiken bij de INPUT-opdracht. Tik:

```
10 REM hoeveel cm. rijdt een fiets bij een wielomwenteling?
20 CLS
30 DEFINT A
40 INPUT "WIELDIAMETER (heel getal) ?";A
50 PRINT
60 PRINT "Bij een wieldiameter van";A;"cm"
70 PRINT "gaat uw fiets bij 1 keer draaien"
80 PRINT "van het wiel";A*3.14;"cm vooruit."
90 PRINT : GOTO 40
```

Nu vraagt de computer niet zomaar om een waarde, maar op het beeldscherm staat welke waarde gevraagd wordt, namelijk de wieldiameter, opgegeven als geheel getal.
Een bericht achter de INPUT-opdracht eist dat u tussen het bericht en de eerste variabelenaam altijd een puntkomma (;) plaatst. Achter het bericht kunnen meer dan één variabele geplaatst worden. Bijvoorbeeld:

```
INPUT "Noem vier getallen" ; A,B,C,D
```

De variabelen die een waarde toegekend krijgen met een INPUT-opdracht hoeven niet van dezelfde soort te zijn. Bijvoorbeeld:

```
INPUT "Noem drie getallen (integer, reeel, integer)" ; A%,B,C%
```

Naast het toekennen van numerieke gegevens aan numerieke variabelen kunnen ook stringgegevens aan stringvariabelen opgegeven worden. Ook hierbij moet gelet worden op het verschil tussen de verschillende soorten variabelen:

```
INPUT "Naam, leeftijd" ; A$,B%
```

Voor het invoeren van een hele reeks lettertekens kent MSX de INPUT$-opdracht (spreek uit: inputstring) en de LINE INPUT-opdracht. Deze zullen behandeld worden in het hoofdstuk over strings.

Weergave op het beeldscherm

De keerzijde van het invoeren van gegevens is de weergave door de computer van gegevens op het beeldscherm. Enkele stelregels hiervoor bent u al tegen gekomen. Zo wordt voor elk getal altijd een spatie gereserveerd voor een eventueel minteken. Daarnaast zorgt de MSX ervoor dat twee getallen nooit tegen elkaar aan op het beeldscherm geplaatst worden. De getallen zouden dan niet meer van elkaar te onderscheiden zijn.
Het weergeven van gegevens en berichten door de computer op het beeldscherm gebeurt meestal met de PRINT (=afdruk) opdracht. Het is u natuurlijk al opgevallen dat door deze opdracht te gebruiken de te printen tekst niet altijd op dezelfde afstand van de linker 'kantlijn' begint. Denk aan de getallen met hun spatie voor een minteken. De plaats van de af te drukken tekst kan met verschillende middelen bepaald worden. Allereerst kunnen verschillende leestekens tussen de verschillende te printen gegevens geplaatst worden. Probeert u het volgende programma (tik eerst NEW):

```
10 A=4
20 B=A^2
30 PRINT A;B;B^2
40 PRINT "MSX";"MICRO";"COMPUTER"
```

Als u dit programma runt ziet u dat de teksten tegen elkaar aangezet worden, en de getallen slechts een geringe tussenruimte hebben.

```
 4  16  256
 MSXMICROCOMPUTER
```

Als de MSX er niet op lette dat getallen nooit tegen elkaar aan mogen staan, hadden deze ook een rij gevormd. Het blijkt dat een puntkomma (;) tussen print-gegevens ervoor zorgt dat de gegevens zonder tussenruimte op het scherm geplaatst worden. Hetzelfde effect wordt bereikt door de puntkomma te vervangen door een spatie ( ). Dit kan echter alleen tussen af te drukken strings!. Tussen variabelen moet altijd een puntkomma gebruikt worden. Vervang in bovenstaand programma eerst de puntkomma's in regel 40 door spaties en run het programma. Vervang daarna ook de puntkomma's in regel 30 door spaties en run opnieuw. Let op het verschil.
Een ander teken dat tussen de printgegevens gezet kan worden is een komma (,). Plaats in bovenstaand programma tussen de printgegevens komma's en run het programma. Het resultaat zal er zo uitzien:

```
  4              16
 256
MSX             MICRO
COMPUTER
```

Het lijkt op een rommeltje. De MSX plaatst gegevens die gescheiden zijn door een komma telkens in de volgende kolom. Er zijn twee kolommen op een scherm. Als er dus meer dan twee gegevens zijn, wordt het derde gegeven weer in de eerste kolom geplaatst, maar nu een regel lager. Daardoor staan 256 en COMPUTER in de eerste kolom. Het verspringen van de letterteksten ten opzichten van de getallen moet weer verklaard worden door de plaats voor het minteken.

Tabulatie

Er zijn verschillende manieren waarop de weergave op het scherm beinvloed kan worden. Belangrijk bij al deze methodes is het verschil tussen de verschillende schermen (screens). De MSX heeft twee tekstschermen: SCREEN 0 en SCREEN 1.
SCREEN 0 is 40 karakterposities breed. De posities worden genummerd van 0 tot en met 39. Als u echter uw computer aanzet en probeert om 40 karakters op het scherm te zetten, zult u merken dat er slechts 37 karakters op het scherm passen. De eerste twee karakterposities en de laatste karakterpositie blijven ongebruikt. De MSX heeft de vreemde gewoonte om standaard een ruime kantlijn aan beide zijden van het scherm aan te houden. Wilt u alle 40 karakterposities gebruiken (ook met LOCATE, zie onder) dan moet u eerst expliciet opgeven dat het scherm 40 karakters breed is door de opdracht

```
WIDTH 40
```

SCREEN 1 is 32 karakterposities breed. De posities worden genummerd van 0 tot en met 31. Ook bij dit scherm geldt dat de standaardbreedte niet gelijk is aan de werkelijk mogelijke breedte. Gebruik de opdracht

```
WIDTH 32
```

om het gehele scherm te benutten.
De lengte van elk tekstscherm is 24 regels. De onderste regel wordt echter bezet door de weergave van de funktietoetsen. Deze kunt u uitschakelen met

```
KEY OFF
```

Nu kunt u de onderste regel wel gebruiken. Wilt u de weergave van de funkties weer zien dan gebruikt u:

```
KEY ON
```

U kunt de WIDTH-opdracht ook gebruiken om een andere breedte dan het gehele scherm op te geven. De algemene opdracht luidt:

```
WIDTH xxx                                    (breedte)
```

Op de plaats van de x-en zet u de door u gewenste breedte van het scherm opgegeven in karakterposities. Zo zorgt WIDTH 20 voor een scherm van 20 karakters breed. Als u dit gedaan heeft, verandert de nummering van de karakterposities wel! De WIDTH-opdracht maakt het scherm namelijk niet aan een kant minder breed, maar haalt aan beide kanten van het scherm evenveel posities weg. Daardoor wordt het nieuwe, minder brede scherm keurig in het midden geplaatst. De eerste positie van het nieuwe, minder brede scherm is nu karakterpositie 0! De laatste karakterpositie is gelijk aan de opgegeven breedte minus 1 (vanwege het feit dat de nummering bij 0 begint). U kunt elke gewenste breedte opgeven, met een minimum van 1 en een maximum dat gelijk is aan dat van het gebruikte SCREEN. De hieronder te behandelen opdrachten houden allebei rekening met de breedte die u met WIDTH opgegeven heeft.
Om de computer te dwingen een tekst of een getal niet links op het scherm bij de kantlijn, maar meer naar rechts weer te geven, gebruikt u de opdracht

```
TAB( )                                       (TABulatie)
```

Deze opdracht wordt altijd samen met de PRINT-opdracht gebruikt. Tussen de haakjes plaatst u de karakterpositie waar de af te drukken tekst moet beginnen. Bijvoorbeeld:

```
PRINT TAB(5) "positie vijf"
```

Door deze opdracht wordt de tekst "positie vijf" op de zesde positie vanaf de linkerkant geplaatst (want de nummering begint bij 0). Hetzelfde kan met getallen:

```
PRINT TAB(9) 5*3
```

Nu wordt het getal op positie 10 geplaatst. Voor het getal wordt immers een spatie gereserveerd voor een eventueel minteken.
De 'opmaak' van de PRINT TAB-opdracht is niet veeleisend. Tussen het sluithaakje en de af te drukken tekst mag een spatie staan, maar dat is niet verplicht. Tussen PRINT en TAB mag een spatie staan, maar dat is niet verplicht. Tussen de sluithaak en de

aanhalingstekens van de af te drukken string mag een spatie, een puntkomma of niets staan. Plaatst u daar een komma, dan werkt de TAB-opdracht niet naar behoren, en wordt de tekst in de volgende schermkolom afgedrukt.
De enige eis die wel gesteld wordt is dat tussen TAB en het eerste haakje geen spatie staat. Doet u dit wel, dan worden de haakjes met het getal er tussen als variabelenaam opgevat.
U mag verschillende TAB-opdrachten achter elkaar gebruiken:

```
PRINT TAB(5) "vijf" TAB(15) "vijftien"
```

Terugspringen doet de TAB-opdracht niet. Als u in bovenstaand voorbeeld 15 verandert in 2, springt de cursor niet terug, maar wordt de tweede tekst achter de eerste geplaatst.
Geeft u een getal op groter dan de breedte van het scherm, dan telt de computer gewoon door op de volgende regel.

LOCATE

De opdracht LOCATE (plaats), plaatst de cursor op een bepaalde positie. Daarna kan vanaf die positie iets afgedrukt worden. Dit lijkt nogal op de TAB-opdracht, maar er zijn enkele verschillen:
De LOCATE-opdracht moet voor de PRINT-opdracht gegeven worden.
De LOCATE-opdracht kan een positie niet alleen links-rechts, maar ook boven-onder bepalen.
Zoals hierboven al gemeld, is het scherm, na een WIDTH 40 opdracht verdeeld in 40 posities van links naar rechts, en 24 regels van boven naar onder. Door de LOCATE-opdracht kan de cursor naar een willekeurige positie en regel op het scherm verplaatst worden. Dit gaat op de volgende wijze:

```
10 CLS
20 LOCATE 5,4
30 PRINT "positie vijf, regel 4"
```

Zoals u ziet gebruikt u bij de LOCATE-opdracht geen haakjes, maar een spatie tussen LOCATE en de cijfers (deze spatie is niet verplicht). Het eerste cijfer geeft de letterpositie vanaf links aan, het tweede cijfer de regel vanaf boven. Elk van deze cijfers kan weggelaten worden, waarna de computer aanneemt dat de huidige positie van de cursor bepalend is:

```
10 CLS
20 LOCATE 5,4
30 PRINT "positie vijf, regel 4"
40 LOCATE ,20
50 PRINT "regel 20";
```

```
60 LOCATE 30
70 PRINT "positie 30"
```

Hiervoor is wel nodig dat, indien het om het behouden van een regelnummer gaat, de computer na het afdrukken van de tekst niet automatisch naar de volgende regel springt. Daarom is in regel 50 de puntkomma gebruikt.
Als u de karakterpositie weglaat, moet u de computer opgeven dat u met het overblijvende getal het regelnummer bedoeld. Daarom blijft de komma in regel 40 voor het regelnummer staan.
LOCATE kan niet gebruikt worden als direkte opdracht, omdat de computer na een direkte opdracht altijd terug springt naar de linker kant om een nieuwe opdracht te krijgen.
LOCATE kan ook gebruikt worden om de cursor uit of weer aan te zetten:

```
LOCATE ,,0                                  cursor uit

LOCATE ,,1                                  cursor aan
```

In plaats van de komma's kunnen karakterposities of regels opgegeven worden in dezelfde opdracht.

Het volgende programma maakt gebruik van de LOCATE-opdracht om u goed te laten wennen aan de opmaak van een beeldscherm. Tik het programma in, en oefen ermee.

```
10 SCREEN 0 : WIDTH 40
20 PRINT "01234567890123456789012345678901234567890";
30 KEY OFF
40 FOR A=1 TO 23
50 LOCATE 0,A
60 PRINT A;
70 NEXT     A
80 LOCATE 20,21
90 INPUT "KOLOM ";X
100 LOCATE 20,22
110 INPUT "REGEL ";Y
120 LOCATE X,Y,0
130 PRINT "*(";X;",";Y;")"
140 LOCATE 27,21
150 PRINT "   "
160 LOCATE 20,22
170 PRINT "          "
180 GOTO 80
```

Probeert u zelf te ontdekken waarvoor de puntkomma in regel 60 dient en waarvoor de regels 140 tot en met 170 dienen. Haal telkens iets weg en let op het verschil.

Opmaken van gegevens door PRINT USING

Naast de hierboven genoemde mogelijkheden kan van elke af te drukken tekst individueel ook de opmaak beinvloed worden. Dit doet u door de opdracht:

```
PRINT USING "symbool";
```

Op de plaats van 'symbool' kunnen verschillende symbolen geplaatst worden die elk voor een andere weergave van de af te drukken tekst, of het af te drukken getal zorgen.

```
PRINT USING "!"
```

Door dit symbool wordt alleen het eerste letterteken van de af te drukken tekst weergegeven.

```
PRINT USING "!" ; "MSX"                    resultaat: M
```

Dit symbool is alleen te gebruiken voor strings (niet voor numerieke variabelen).

```
PRINT USING "\spaties\"
```

Hierdoor wordt het aantal karakters aangegeven door het aantal spaties plus 2 weergegeven. Als de af te drukken gegevens korter zijn dan het aantal spaties plus 2 wordt de rest aangevuld met spaties.

```
PRINT USING "\   \";"MSX-COMPUTER"      resultaat: MSX-
```

Het is mogelijk achter een PRINT USING verschillende af te drukken gegevens te plaatsen:

```
PRINT USING "\   \";"MSX","COMPUTER"    resultaat: MSX COMP
```

Elk gegeven wordt nu afgedrukt als het symbool aangeeft. De spatie tussen MSX en COMP wordt veroorzaakt doordat de spaties tussen de schuine strepen plus twee samen vier is, en MSX slecht uit drie tekens bestaat. De rest wordt aangevuld met spaties.

```
PRINT USING "..&.."
```

De opgegeven tekst wordt op de plaats aangegeven door & helemaal

tussengevoegd.

```
PRINT USING "..&..";"MSX"            resultaat: ..MSX..
PRINT USING "& MSX"; "Hier"," Daar"  resultaat: Hier MSX Daar MSX
```

```
PRINT USING "#.##"
```

De matjes (#) geven elk een cijfer van het af te drukken getal weer. De punt (.) geeft de plaats van de decimale komma aan. Dit kan vooral handig zijn bij het weergeven van kolommen met geldbedragen waarbij de guldens netjes onder elkaar moeten komen te staan. Als het af te drukken getal meer cijfers heeft dan opgegeven is met matjes, wordt een % voor het getal afgedrukt. Als er minder cijfers zijn, wordt het getal gecentreerd. Dat wil zeggen dat aan beide zijden van het getal evenveel spaties geplaatst worden. Dit geldt echter alleen voor getallen voor de komma, of getallen zonder komma. Als het getal minder cijfers achter de komma heeft dan de matjes aangeven, wordt de rest opgevuld met nullen. Zijn er teveel cijfers dan wordt het reststand afgerond:

```
PRINT USING "###.##";123.45          resultaat: 123.45
PRINT USING "###.##";123.457         resultaat: 123.46
PRINT USING "###.##";1234.57         resultaat: %1234.57
PRINT USING "###.##";123.4           resultaat: 123.40
PRINT USING "###.##";1.57            resultaat:   1.57
PRINT USING "######" ;12             resultaat:     12
```

Bij het weergeven van de getallen wordt het plusteken (+) niet meegeteld. Het minteken (-) wordt als cijfer meegeteld.

```
PRINT USING "+###"
```

De getallen worden weergegeven met het minteken (negatief, kleiner dan nul) of het plusteken (positief, groter dan nul) voor het getal. Het aantal matjes is bij gebruik van dit symbool vrij. Tevens mag gebruik gemaakt worden van de decimale punt, of een van de andere symbolen.

```
PRINT USING "###+"
```

De getallen worden weergegeven met het teken achter het getal.

```
PRINT USING "+###";123,-123          resultaat: +123-123
PRINT USING "###+";123,-123          resultaat: 123+123-
```

```
PRINT USING "###-"
```

Negatieve getallen worden weergegeven met het minteken achter het getal.

```
PRINT USING "**###"
```

De ruimte voor een numeriek gegeven wordt gevuld met asterisken (*). Het aantal te gebruiken matjes is vrij.

```
PRINT USING "**###";123,12,1      resultaat: **123***12****1
```

```
PRINT USING "£##"
```

Het getal wordt weergegeven met een pondteken (£) ervoor. Dit symbool wordt als cijfer meegeteld bij opmaak volgens een van de andere symbolen.

```
PRINT USING "£##.##";12.34            resultaat: £12.34
```

Door voor het pondteken asterisken te gebruiken, wordt het pondteken onmiddellijk voor het af te drukken getal gezet en worden de plaatsen daarvoor gevuld met asterisken.

```
PRINT USING ",##"
```

Als de komma ergens voor de decimale punt geplaatst wordt, wordt tussen elke drie cijfers een komma geplaatst. Een Amerikaanse methode om duizendtallen weer te geven.

```
PRINT USING "#,##.##";1234.567        resultaat: %1,234.57
```

Gebruik de komma niet als eerste symbool.

```
PRINT USING "##^^^^"
```

Deze symbolen bepalen de opmaak van het exponent-gedeelte bij wetenschappelijk notatie. Er wordt ruimte gemaakt voor E+..

```
PRINT USING "##.##^^^^";123456.78     resultaat: 1.23E+05
```

De meeste van de hierboven genoemde symbolen kunnen gecombineerd

worden. U heeft dit al kunnen zien bij het symbool # dat bijna overal gebruikt wordt om het aantal cijferplaatsen aan te geven.

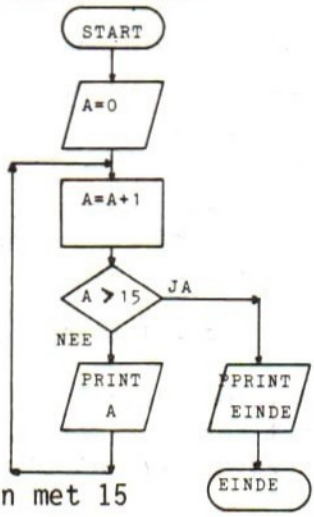
Hengelen

Tot slot nog een leuk programma waarin veel gebruik gemaakt is van de LOCATE-opdracht. Kijk of u het programma begrijpt door alleen de listing te lezen. Maak voor uzelf een stroomdiagram. Er zitten wel veel opdrachten in die tot nu toe nog niet in dit boek behandeld zijn. Mocht het programma u in eerste instantie nog niet duidelijk zijn, lees dan dit boek gewoon verder door. Na verloop van tijd zult u alle hierin behandelde opdrachten begrijpen.

```
10 REM hengelen
20 SCREEN 0 : WIDTH 40 : KEY OFF
30 DEFINT A-Z
152 LET S=0
50 LET PU=0
60 PRINT "moeilijkheidsgraad (30-5)"
70 PRINT "30 = gemakkelijk"
80 PRINT "5  = moeilijk";
90 INPUT M
100 INPUT "SNELHEID (1-5) ";V
110 CLS
120 FOR ST=1 TO 20
130 LOCATE 25,23 : PRINT "score:";PU;
140 LOCATE M+2,0 : PRINT"<<^>>";
150 LET RA=INT(RND(1)*20)+1
160 FOR X=0 TO 37
170 LOCATE X,RA : PRINT" *";
180 LOCATE M+4,S : PRINT":";
190 LOCATE M+4,S+1 : PRINT" ";
224 C=STICK(0)
210 IF S=0 THEN GOTO 240
220 IF C=1 THEN S=S-1
230 IF S=23 THEN GOTO 250
240 IF C=5 THEN S=S+1
250 LOCATE M+4,S : PRINT"#";
260 IF M+4=X AND S=RA THEN :
    BEEP : BEEP : BEEP : PU=PU+1 :GOTO 290
270 FOR Z=1 TO 50-V*5 : NEXT Z
280 NEXT X
290 NEXT ST
300 PRINT "einde"
```

8. LUSSEN

In bijna elk programma is het wel eens handig om de computer een bepaalde handeling een aantal keren te laten verrichten, telkens met een iets ander gegeven, of telkens met dezelfde gegevens. Stel dat het volgende probleem door de computer opgelost moet worden. De getallen van 1 tot en met 15 moeten op het beeldscherm weergegeven worden zonder dat alle vijftien ingetikt hoeven te worden. We kunnen daarvoor een stroomdiagram maken:



PRINT de getallen van 1 tot en met 15

Op deze wijze is er een loop (=lus) in het programma gemaakt. De computer draait in het programma een rondje en komt daar pas uit als A groter is dan 15. De eerste keer dat de computer de lus doorloopt is de waarde van A gelijk aan 0+1=1. De computer drukt dit af, gaat naar de volgende opdracht waar gekeken moet worden of A groter is dan 15. 1 is kleiner dan 15 dus het antwoord is nee. De computer wordt terug gestuurd, telt weer 1 op bij A. A wordt daardoor 2. Dit wordt weer afgedrukt, vergeleken met 15, enz. Dit rondje, deze lus, draait de computer net zo lang tot het antwoord op de vraag 'is A groter dan 15?', positief beantwoord kan worden. Als A=16 is A groter dan 15 en gaat de computer niet terug in het programma, maar gaat hij verder met de volgende regel die het einde van het programma aangeeft. Hiermee is dus een voorbeeld gegeven van een programma waarbij de computer bij het doorlopen van het programma wel op de volgorde van de regelnummers let, maar waarbij de programmeur ervoor kan zorgen dat de computer bepaalde stukken programma verschillende keren doorloopt.
Het programma dat bij bovenstaand stroomdiagram hoort is:

```
10 LET A=0
20 LET A=A+1
30 IF A>15 THEN GOTO 60
40 PRINT A
50 GOTO 20
60 PRINT "EINDE"
```

Maar van een computer als de MSX kunt u hiervoor een slimmigheidje verwachten. Dat is er dan ook. Om de hierboven genoemde gang van zaken te bespoedigen gebruikt u

FOR...NEXT... (vanaf... volgende...)

Tik NEW en het volgende programma in:

```
10 FOR A = 1 TO 15
20 PRINT A
30 NEXT A
```

Met minder en kortere programmaregels bereikt u hetzelfde resultaat. De opdracht werkt als volgt:
In de eerste regel staat

FOR A = 1 TO 15

Vertaald in mensentaal wil dat zeggen: computer, de waarde van A gaat van 1 tot en met 15 lopen en zal telkens met 1 verhoogd worden. Begin maar met de eerste waarde.
Regel 20:

PRINT A

Deze regel geeft de computer de bewerking die A moet ondergaan. In dit geval alleen PRINT, maar alle andere BASIC bewerkingen zijn mogelijk. Ook BASIC-opdrachten waarbij A helemaal geen rol speelt. Er is geen maximum aantal opdrachten, maar ze moeten allemaal tussen de regel met FOR en de regel met NEXT staan.
Regel 30:

NEXT A

Deze regel vertelt de computer de volgende waarde voor A te nemen, terug te gaan naar de regel waarop FOR staat, daar te kijken of de waarde niet groter is geworden dan het aangegeven bereik (TO 15). Als dat niet zo is, gaat de computer naar de volgende regel en voert daar de bewerking uit met eventueel de nieuwe waarde voor A. Vervolgens komt de computer weer bij regel 30 waar A weer met 1 verhoogd wordt, etc. etc.
Zodra A de waarde 16 bereikt, voert de computer de instruktie uit regel 20 niet meer uit en springt hij naar de regel volgend op de

NEXT-statement (= formulering van een opdracht). In dit programma is er geen volgende regel aanwezig en een programma stopt. Met de FOR...NEXT-opdracht kunt u de computer een bepaalde opdracht verschillende keren laten uitvoeren, waarbij u zelf het aantal keren aangeeft en de computer de waarde van een opgegeven variabele verandert.
Maar er is meer te doen met de FOR...NEXT-opdracht. De stappen die de computer neemt bij het veranderen van de waarde kunnen we beinvloeden. Tikt u het volgende demonstratieprogramma maar in:

```
10 REM demonstratie FOR...NEXT
20 CLS
30 PRINT "FOR X=16 TO 25 geeft: "
40 PRINT
50 FOR X=16 TO 25
60 PRINT X
70 NEXT X
80 PRINT
90 PRINT
100 REM pauze
110 FOR X=1 TO 5000 : NEXT X
120 REM ook grotere stappen mogelijk
130 PRINT "FOR X=1 TO 21 STEP 2 geeft: "
140 PRINT
150 FOR X=1 TO 21 STEP 2
160 PRINT X
170 NEXT X
180 PRINT : PRINT
190 REM pauze
200 FOR X=1 TO 5000 : NEXT X
210 REM omlaag
220 PRINT "FOR X=634 TO 517 STEP -9 geeft: "
230 PRINT
240 FOR X=634 TO 517 STEP -9
250 PRINT X
260 NEXT X
270 PRINT : PRINT
280 REM pauze
290 FOR X=1 TO 5000 : NEXT X
300 REM ook niet gehele getallen
310 PRINT "FOR X=117.2 TO 3 STEP -15.7 geeft: "
320 PRINT
330 FOR X=117.2 TO 3 STEP -15.7
340 PRINT X
350 NEXT X
360 PRINT : PRINT
370 REM pauze
380 FOR X=1 TO 5000 : NEXT X
390 REM variabelen gebruiken
```

```
400 PRINT "A=7, B=87, C=8"
410 PRINT "en FOR X=A TO B STEP C geeft: "
420 PRINT
430 LET A=7 : LET B=87 : LET C=8
440 FOR X=A TO B STEP C
450 PRINT X
460 NEXT X
```

RUN dit programma en u krijgt op het beeldscherm een keur van voorbeelden van mogelijke variaties met behulp van de FOR..NEXT-lus.
Daarnaast ziet u in de listing nog enkele andere belangrijke zaken. Er is al eerder op gewezen, maar let u op het gebruik van de REM-statement. Alles wat daarachter staat wordt door de computer overgeslagen. U kunt uw listing ermee verduidelijken.
Regel 100 geeft als kommentaar alleen 'pauze'. In regel 110 staat een manier om de FOR...NEXT-lus als vertrager te laten werken. De enige opdracht die u opgeeft is om een heleboel keer de waarde van de variabele X te verhogen. Daar is de computer even mee bezig, en daardoor vertraagt het programma zoveel dat u de resultaten rustig kunt doorlezen. Deze manier van vertragen is echter niet geschikt voor precieze tijdsbepalingen. Om een nauwkeuriger tijdsmeting uit te leggen wijken we even af van de FOR...NEXT-lussen.

Tijd meten

De MSX maakt heel veel gebruik van zogenaamde interrupts of onderbrekingen. Dit wil zeggen dat datgene waarmee de Z-80A centrale verwerkingseenheid mee bezig is, even onderbroken wordt voor een andere bezigheid. Bijvoorbeeld als de Z80A uw BASIC programma aan het uitvoeren is, terwijl u een toets van het toetsenbord indrukt. De processor houdt dan heel even op met het BASIC-programma om te kijken welke toets u indrukte en dat gegeven in een aparte buffer (opslagruimte) te stoppen voor later gebruik.
Naast deze incidenteel voorkomende onderbrekingen houdt de Z80A elke 50-ste van een seconde even op om een signaal van een aparte 'klok' te ontvangen en de waarde van een aparte variabele TIME (tijd) met een te verhogen. Zo zijn er verschillende taken die de Z80A even tussendoor moet doen. Dit betekent dat de exacte snelheid waarmee uw BASIC programma afgewerkt moet worden niet altijd hetzelfde is. De FOR...NEXT lus is daardoor niet zo heel precies.
De nauwkeurige tijdsbepaling kan echter wel verkregen worden door de variabele TIME te gebruiken. Elke 50-ste van een seconde (dat is de frequentie van het lichtnet) wordt de waarde van de variabele TIME met één verhoogd. De verhoging van de waarde begint op het moment dat u uw MSX aanzet en blijft constant

doorlopen. Alleen in die gevallen dat onderbrekingen niet mogelijk zijn (o.a. bij het lezen van of schrijven naar een cassette) wordt de klok tijdelijk stil gezet. Tikt u:

```
PRINT TIME
```

U krijgt nu de tijd dat MSX-BASIC aangeschakeld is in vijftigsten van seconden. Het uitzetten van de MSX zorgt ervoor dat TIME weer op nul komt te staan. TIME kan maximaal 65535 worden, daarna begint de variabele weer bij 0.
Om in een programma een bepaalde en precieze hoeveelheid tijd uit te trekken kunt u het volgende stukje programma gebruiken:

```
10 TIME = 0
20 EIND = TIME + 500
30 REM we wachten
40 PRINT "We wachten.....";
50 PRINT ".";
60 IF TIME < EIND THEN GOTO 50
70 PRINT : PRINT "10 seconden."
```

In regel 20 geven we op dat EIND gelijk is aan TIME plus 500 (vijftigsten van seconden). Zolang EIND groter is dan TIME zet de MSX punten op het scherm. Omdat dit mis kan gaan als TIME bij het begin van dit programmadeel bijna 65535 is, stellen we in regel 10 de waarde van TIME in op 0. Na 10 seconden (exact) is TIME 500 keer met een verhoogd en dus net zo groot als EIND.
Om de vijftigsten van seconden om te zetten in gewone seconden en minuten gebruikt u de volgende berekeningen:

```
M = FIX(TIME/3000)
S = FIX(TIME/50 - M*60)
```

Er gaan namelijk 50 vijftigsten van een seconde in een seconde, maar na 60 seconden moet de klok weer bij 1 beginnen. Daarom wordt telkens 60 maal het aantal minuten van het totale aantal seconden afgetrokken.

```
FIX       (vast)
```

geeft gehele getallen. Cijfers achter de komma worden verwaarloosd. De statement lijkt op

```
INT       (integer)
```

Maar dit statement geeft het grootste getal dat nog kleiner is dan het opgegeven getal. Vergelijk onderstaande voorbeelden goed:

```
? FIX(5)    geeft  5        ? INT(5)    geeft  5
? FIX(4.9)  geeft  4        ? INT(4.9)  geeft  4
```

```
? FIX(-5)   geeft -5          ? INT(-5)   geeft -5
? FIX(-4.9) geeft -4          ? INT(-4.9) geeft -4
```

Wilt u ook uren uitdrukken dan wordt dit wat lastiger. Er gaan immers 50*60*60 vijftigsten van seconden in een uur. Dat is 180000. De TIME variabele loopt echter maar tot 65535 en begint dan opnieuw met tellen. Daarom moeten de uren uitgedrukt worden in minuten. Een digitale klok kunt u als volgt maken:

```
10 KEY OFF
40SCREEN 1
30 TIME =0
40 U=0
50 M=FIX( TIME/3000 )
60 S=FIX( TIME/50-M*60 )
70 IF M>0 AND INT( M/60 )=M/60 THEN U=U+1
80 LOCATE 6,8 : PRINT "***************"
90 LOCATE 8,10 : PRINT "uur ";U
100 LOCATE 8,11 : PRINT "min ";M
110 LOCATE 8,12 : PRINT "sec ";S
120 LOCATE 6,14 : PRINT "***************"
130 GOTO 50
```

Terug naar de lussen

De FOR...NEXT lus kan ook voor andere doeleinden gebruikt worden dan voor het laten veranderen van een variabele. U kunt de opdracht ook gebruiken om de computer een bepaalde onveranderlijke opdracht een bepaald aantal keren te laten herhalen. De lus dient dan enkel als teller. Een voorbeeld hiervan heeft u als pauzelus al gezien. Als u graag uw naam op de televisie ziet kunt u tikken:

```
10 FOR A=1 TO 22
20 PRINT ".....(uw naam)....."
30 NEXT A
```

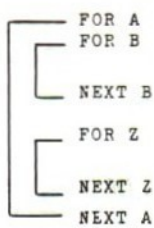
Op deze wijze telt de computer wel telkens 1 op bij A, maar gebruikt hij A niet anders dan om de tel bij te houden.
Het aantal opdrachten dat herhaald moet worden bepaalt uzelf. Elke opdracht die tussen de FOR regel en de NEXT regel staat wordt net zo vaak uitgevoerd als u de computer de lus laat doorlopen.

Lussen binnen lussen

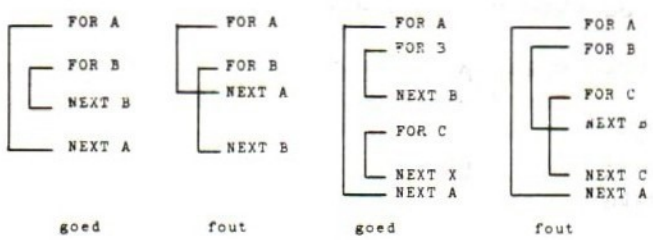
U kunt ook lussen binnen lussen maken. Dit worden genestelde lussen genoemd. De lussen liggen als nesten binnen elkaar. De binnenste lus wordt elke keer als de buitenste lus wordt doorlopen net zo vaak doorlopen als u aangeeft. Een voorbeeld maakt dit duidelijk.
Tik NEW en het volgende programma:

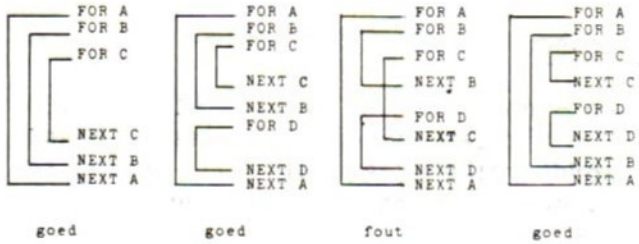
```
10 FOR A=1 TO 3
20 PRINT "BUITENSTE LUS NR. ";A
30 PRINT
40 FOR B=1 TO 4
50 PRINT "binnenste lus nr. ";B
60 NEXT B
70 PRINT
80 FOR Z=1 TO 2000 : NEXT Z
90 NEXT A
```

Dit programma heeft drie lussen, waarvan eentje de speciale pauzelus vormt.
De lus FOR B=1 TO 4..NEXT B ligt helemaal binnen de lus FOR A=1 TO 3..NEXT A. Naast de B-lus ligt de lus FOR Z=1 TO 2000..NEXT Z. Ook deze lus ligt binnen de A-lus. In schema zien de lussen er als volgt uit:



Let u bij het leggen van genestelde lussen goed op dat de binnenste lus ook echt binnen de buitenset lus ligt. De lussen mogen elkaar niet snijden!





voorbeelden van genestelde lussen

Als slot van dit hoofdstuk nog twee verschillende korte rekenprogramma's.

Het eerste programma rekent voor u het gemiddelde van een stel getallen uit.

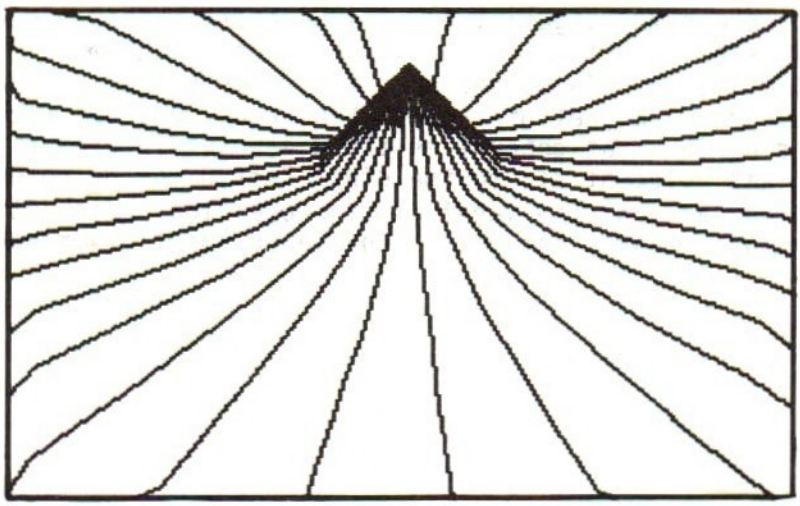
```
10 SCREEN 0
20 PRINT : PRINT "Ik reken voor u het gemiddelde uit "
30 PRINT "van een lijst door u opgegeven"
40 PRINT "getallen."
50 PRINT : INPUT "HOEVEEL GETALLEN IN DE LIJST";N
60 PRINT : PRINT "Druk na elk getal op RETURN."
70 PRINT
80 T=0
90 FOR L=1 TO N
100 PRINT "getal nr.";L;" = ?";
110 INPUT X
120 T=T+X
130 NEXT L
140 G=T/N
150 PRINT : PRINT "Het TOTAAL =";T
160 PRINT : PRINT "Het GEMIDDELDE =";G
```

Het volgende programma berekent de faculteit van een getal. De faculteit van bijvoorbeeld het getal N wordt aangegeven met N! (spreek uit: en faculteit). Dit staat voor N*(N-1) * (N-2) * (N-3) *....3*2*1. Normaal gesproken wordt dit alleen gebruikt bij positieve gehele getallen. Het programma wordt als volgt:

```
10 SCREEN 0
20 PRINT "Berekening van FACULTEIT"
30 PRINT
50 INPUT "EEN GETAL A.U.B.";G
60 LET A=G
70 LET B=G
80 FOR A=G-1 TO 1 STEP -1
90 LET B=A*B
100 NEXT A
```

```
120 PRINT : PRINT
130 PRINT G ;" faculteit is";B
140 PRINT
150 PRINT "Tik RETURN voor nog een keer."
160 INPUT "Tik 1-RETURN voor stop.";N
170 PRINT : IF N<>1 THEN GOTO 30
180 PRINT "EINDE"
```

Als u dit programma ingetikt en geRUNd heeft kunt u van allerlei getallen snel de faculteit vinden. Het programma is niet 'beveiligd' tegen onjuist gebruik. Als de gebruiker een negatief getal invoert werkt het niet goed. Een breuk kan ook niet veilig ingevoerd worden. Wat gebeurt er als u een getal groter dan 48 intikt? Wat betekent die foutmelding? Ziet u dat de notering van de getallen anders is vanaf 17!. (Overflow betekent dat de geheugenplaats waarin het getal opgeslagen wordt, overloopt.)



9. LOGICA EN IF...THEN...

Vergelijken

Uw MSX is een mooie computer, maar zelfstandig denken is er niet bij. Beslissingen neemt de MSX dan ook niet. Maar de computer kan wel twee getallen vergelijken. Het onderlinge verband tussen de twee getallen kan de MSX brengen onder een van de volgende noemers:

=	is gelijk
<> of ><	is niet gelijk
<	kleiner dan
>	groter dan
<= of =<	gelijk of kleiner dan
>= of =>	gelijk of groter dan

Let u op de afwijkingen met de notaties voor deze vergelijkingen zoals u die wellicht kent uit de wiskunde:

<> of >< wordt normaal aangegeven met ≠
>= of => wordt normaal aangegeven met ≥
<= of =< wordt normaal aangegeven met ≤

Als de computer twee getallen heeft vergeleken, kunnen we aan de hand van de uitkomst de computer een bewerking wel of niet laten uitvoeren, een waarde aan een variabele laten toekennen of naar een regel in het programma laten springen. De hiervoor gebruikte opdracht is:

IF...THEN... (als...dan...)

De eerste stippeltjes van IF...THEN... staan voor het vergelijken van twee waarden. Als de waarden zich verhouden zoals het er tussen staande teken (uit de lijst hierboven) aangeeft, dat wil zeggen dat de bewering waar is, dan (THEN) voert de computer de opdracht(en) op het tweede stel stippen uit.
We kunnen ook zeggen: IF de bewering is waar THEN voert de opdracht uit.
In de praktijk blijkt dit een zeer nuttige faciliteit te zijn, want op deze manier kan de computer bepaalde 'beslissingen' nemen. Hier een paar voorbeelden:

```
IF X<10 THEN GOTO 300
```

Als de waarde van X kleiner is dan 10 of anders: als de bewering

X<10 waar is, gaat de computer naar regel 300 (THEN GOTO 300). Zo niet (X<10 is niet-waar), dan vervolgt de computer het programma bij de volgende programmaregel, zonder de opdracht achter THEN uit te voeren. In een miniprogramma ziet het er zo uit:

```
10 INPUT X
20 IF X<10 THEN GOTO 300
30 PRINT "X is groter dan of gelijk aan 10"
40 END
300 PRINT "X is kleiner dan 10"
```

Let erop dat een variabele alleen dan als waarde in een vergelijking gebruikt mag worden als de variabele eerder in het programma een waarde toegekend heeft gekregen. Anders beschouwt de MSX de variabele als 0, hetgeen tot ongewenste effecten kan leiden.

IF A>B THEN GOTO 300

Als de waarde van A groter is dan de waarde van B (A>B is waar) dan gaat de computer naar regel 300. Zo niet (A>B is onwaar) dan vervolgt hij het programma met de volgende regel.

IF Y=0 THEN LET Z=6

Als de waarde van Y gelijk is aan 0 (Y=0 is waar) dan kent de computer aan de variabele Z de waarde 6 toe en vervolgt daarna het programma met de volgende regel. Is de bewering onwaar dan vervolgt de computer het programma met de volgende regel.

IF K<>L THEN PRINT "K is niet gelijk aan L"

De tekst tussen aanhalingstekens wordt alleen afgedrukt als K niet gelijk is aan L (K<>L is waar).

IF P>=Q THEN CLS

Het scherm wordt gewist als de bewering P>=Q waar is. Anders gaat de computer direkt door met de volgende programmaregel.

IF T<=S THEN BEEP

Het programma verzorgt een bliep als T kleiner dan of gelijk aan S is. Zie als voorbeeld dit programma:

```
10 LET S=11
20 FOR T=20 TO 6 STEP -1
30 PRINT "T=";T,"S=";S
40 IF T<=S THEN BEEP
50 FOR X=1 TO 500 : NEXT X
60 NEXT T
70 END
```

ELSE

Bij de opdracht IF ..1.. THEN ..2.., kan bij ..2.. elke BASIC opdracht (of een combinatie opdrachten) geschreven worden. Als de bewering ..1.. waar is, voert de computer de opdracht(en) uit. Als de bewering niet waar is negeert de computer de opdrachten bij ..2.. en gaat hij door met de volgende programmaregel. Naast deze algemene BASIC opdracht kent MSX BASIC nog een uitgebreidere vorm:

IF...THEN...ELSE... (als...dan...anders...)

Deze opdracht zorgt ervoor dat de computer niet onmiddellijk naar de volgende regel springt als de bewering achter IF onwaar is, maar eerst de opdracht achter ELSE uitvoert. Bijvoorbeeld:

```
IF A>Z THEN GOTO 200 ELSE PRINT " "
```

Als de bewering A>Z waar is gaat de computer naar 200, anders (dat wil zeggen A>Z is niet waar) drukt de computer een spatie af.

```
IF Q=10 THEN A=B ELSE A=C
```

Als de bewering Q=10 waar is, krijgt A de waarde van B, anders krijgt A de waarde van C. Hierna vervolgt de computer het programma met de volgende programmaregel. Zie bijvoorbeeld het volgende programma:

```
10 LET B=5
20 LET C=7
30 INPUT "Q:";Q
40 IF Q=10 THEN A=B ELSE A=C
50 PRINT A
60 GOTO 30
```

De volgende punten bij het gebruik van de IF...THEN... vertakkingen zijn nog te vermelden:
Als achter THEN de opdracht GOTO regelnummer komt, mag GOTO weggelaten worden, of mag THEN weggelaten worden.

```
IF X=Z THEN GOTO 200
IF X=Z THEN 200        } Deze regels betekenen hetzelfde
IF X=Z GOTO 200
```

Als achter een ELSE de opdracht GOTO regelnummer komt, mag GOTO weggelaten worden.
Na THEN of ELSE mag meer dan een opdracht gezet worden. De opdrachten moeten gescheiden zijn door dubbele punten (:). De opdrachten achter THEN worden of allemaal uitgevoerd (bewering waar) of geen van alle uitgevoerd (bewering niet waar). Hetzelfde maar dan omgekeerd geldt voor ELSE.

Logische operators

Bij het programmeren in BASIC zult u AND, OR en NOT vooral als logische operators (bewerkers) tegenkomen. Zij werken dan als volgt:

```
ccc AND ddd is alleen waar als ccc waar is EN ddd ook waar is.
ccc OR ddd  is alleen waar als ccc waar is OF ddd waar is OF
            allebei waar zijn.
NOT ccc     is alleen waar als ccc NIET waar is.
```

Een paar voorbeelden:

```
IF X>0 AND X<1 THEN ...
```

Als de waarde van X groter is dan 0 en de waarde van X is ook kleiner dan 1, dan is de totale bewering waar en zal de computer de opdracht achter THEN uitvoeren.

```
IF X>0 OR Y<1 THEN ...
```

Als de waarde van X groter is dan 0 of de waarde van Y kleiner is dan 1, of allebei is waar, dan is de hele bewering waar. De computer voert de opdracht achter THEN uit.

```
IF 2+3=5 AND 4+7=11 is waar
IF 2+3=5 OR 4+3=11 is ook waar
```

```
IF NOT X=Y*A THEN ...
```

Als de waarde van X niet gelijk is aan Y*A is de bewering NOT X=Y*A waar. De opdracht achter THEN wordt uitgevoerd.

Bij elk van deze IF...THEN... opdrachten is het mogelijk ELSE te gebruiken.

De logische operators kunnen ook in combinatie gebruikt worden:

```
IF NOT (X=1 AND Y=9) THEN ...
IF (X=1 AND Y=9) OR (X=-1 AND Y=-9) THEN ...
IF (X=1 OR X=9) AND (A=-1 OR A=-9) THEN ...
```

Bij de eerste regel is de gehele bewering waar als X niet gelijk is aan 1 of Y niet gelijk is aan 9.
Bij de tweede regel is de gehele bewering waar als (X=1 en Y=9) of als (X=-1 en Y=-9).
Bij de derde regel is de gehele bewering waar bij de volgende combinaties:

X=1 en A=-1
X=1 en A=-9
X=9 en A=-1
X=9 en A=-9

U ziet dat de logische operators net als de rekenkundige operators soms haakjes nodig hebben. Worden in een vergelijking zowel logische als rekenkundige operators gebruikt, dan worden eerst de rekenkundige operators uitgewerkt. U kunt deze volgorde veranderen door gebruik te maken van haakjes.

Gebruik

Het gebruik van de logische operators dient met de nodige voorzichtigheid te geschieden. Het gebruik van deze logica leidt snel tot fouten die zeer moeilijk te herkennen zijn, en dus moeilijk uit het programma gehaald kunnen worden.

```
IF X>0 OR X<1 THEN...
```

Dit is een zinloze bewering. Als de waarde van X groter is dan 0 of de waarde van X kleiner is dan 1, of allebei, is de bewering waar. Deze bewering is dus altijd waar. Er bestaat geen waarde voor X die kleiner is dan 0 en groter is dan 1.

```
IF P>0 AND P<0 THEN PRINT "het perpetuum mobile"
```

Er bestaat geen enkele waarde voor P die zowel groter, als kleiner dan 0 is. Deze bewering is altijd onwaar en de opdracht achter THEN zal, ongeacht de rest van het programma, nooit uitgevoerd worden.
Uiteraard kent de MSX voor dit soort fouten geen foutmelding. Bij een onware bewering wordt de opdracht na THEN gewoon genegeerd en gaat de computer verder met de opdracht achter ELSE of met de volgende programmaregel. De computer kan niet kontroleren of u onzinnige beweringen laat uittesten.
Een zeer eenvoudig programma waarin IF...THEN... gebruikt wordt is het onderstaande getal-raad programma.

```
10 REM raad het getal
20 SCREEN 1
30 LET X%=INT(RND(1)*100)
40 PRINT "U moet een getal tussen"
50 PRINT "1 en 100 raden."
60 PRINT "Zodra u het door mij "
70 PRINT "gekozen getal raadt, heeft"
80 PRINT "U GEWONNEN."
90 PRINT
100 INPUT "Welk getal gokt u";G
```

```
110 IF G=X% THEN GOTO 200
120 IF G<X% THEN PRINT G;" is te laag"
130 IF G>X% THEN PRINT G;" is te hoog"
140 GOTO 100
200 FOR S=1 TO 15 : BEEP : NEXT S
210 COLOR 15,6,5
220 PRINT : PRINT "Prima.";G;" had ik gekozen."
```

Nieuw in dit programma is de opdracht

RND() (RaNDom = willekeurig)

Deze opdracht zorgt ervoor dat de computer een willekeurig getal voor ons uitzoekt. De opdracht kan op verschillende manieren gebruikt worden. Als u tussen de haakjes een getal groter dan 0 invult, worden zogenaamd pseudo-willekeurige getallen kleiner dan 1 en groter dan of gelijk aan 0 gegenereerd. De getallen lijken helemaal willekeurig, maar zijn dat niet. Als u het bovenstaande programma verschillende keren RUNt zult u zien dat de computer telkens dezelfde getallen kiest. Het doet er hierbij niet toe welk getal groter dan 0 u tussen de haakjes plaatst. Elke keer wordt dezelfde reeks doorlopen. De reeks begint telkens opnieuw bij een nieuw programma.
Als u tussen de haakjes een 0 invult, geeft de RND-opdracht het laatste gegenereerde willekeurige getal nog een keer. Dit kan handig zijn bij het controleren van een programma, of wanneer in een programma vereist is dat tweemaal exact hetzelfde, maar willekeurige getal gebruikt wordt.
Als u tussen haakjes een negatief getal neemt, springt de computer aan de hand van dat negatieve getal naar een positie binnen de eerder genoemde reeks van pseudo-willekeurige getallen. Door vervolgens een positieve waarde tussen de haakjes te plaatsen, kan de reeks verder afgelopen worden vanaf dat punt. Een negatief getal zorgt namelijk elke keer voor een sprong naar hetzelfde punt binnen de reeks. Als u driemaal ?RND(-1) tikt krijgt u driemaal het getal .04389820420821.
Ook op deze wijze krijgt u elke keer als u het programma doorloopt dezelfde getallen. Er is echter een variabele die helemaal niet willekeurig is, maar wel elke keer dat het programma doorlopen wordt een andere waarde heeft. Dit is de variabele TIME, die immers vanaf het moment dat MSX BASIC aangezet is elke 50-ste van een seconde met 1 verhoogd is. Door deze variabele als negatieve waarde tussen de haakjes te plaatsen, springt de computer elke keer als het programma geRUNd

wordt naar een ander punt binnen de reeks, van waaruit de reeks doorlopen kan worden. U voegt daarvoor in bovenstaand programma regel 25 toe:

```
25 R=RND(-TIME)
```

Uiteraard kunt u in uw eigen prgramma's deze truuk ook gebruiken.

Dit programma is wel wat lang voor wat het doet. Alleen maar een getal laten raden en dan 22 programmaregels! Er kan wel wat vanaf. Om te beginnen is het wat overdreven om eerst te testen of G gelijk is aan X, vervolgens of G kleiner is dan X en daarna ook nog of G groter is dan X. Als de eerste twee beweringen niet waar zijn is de derde automatisch wel waar. Daarom kan regel 130 vervallen en moet regel 120 worden:

```
120 IF G<X% THEN PRINT G;" is te laag" ELSE PRINT G;" is te
    hoog"
```

Om het nog korter te maken kunnen zelfs de regels 120 en 110 samengevoegd worden. Om het overzicht te bewaren, maken we de 'te hoog' en 'te laag' meldingen iets korter. Regel 120 vervalt. Regel 100 wordt:

```
 110 IF G=X% THEN 200 ELSE IF G<X% THEN PRINT "te laag"
     ELSE PRINT "te hoog"
```

U ziet dat hier verschillende IF..THEN.. opdrachten na elkaar gebruikt zijn. Nog meer bekortingen op het programma kunt u alleen verkrijgen door het inkorten van instrukties en mededelingen op het scherm. Deze vragen altijd flink wat programmaregels.

Het bovenstaande programma is eenvoudig uit te breiden tot een spel dat op mastermind lijkt.
Probeert u zelf een stroomdiagram en een programma te schrijven dat:

- vier willekeurige getallen kiest
- de gebruiker om vier getallen vraagt
- de geraden getallen vergelijkt met de willekeurig gekozen getallen
- de gebruiker punten geeft voor het aantal juiste getallen op de juiste plaats
- het spel beeindigt als alle getallen geraden zijn
- dit alles van voldoende instrukties en beeldweergaven voorzien

Als u er zelf niet uitkomt (eerst proberen!) kunt u hieronder een mogelijk oplossing vinden.
U kunt ook uw resultaat vergelijken. Als u aan dezelfde eisen in

een ander, goed werkend programma voldaan heeft, wil dat zeker niet zeggen dat uw programma minder goed is. Er zijn vaak verschillende programma's mogelijk voor een bepaald probleem.

```
10 CLS
20 SCREEN 1 : COLOR 15,4,4
30 A=RND(-TIME)
40 W%=INT(RND(1)*10-1)
50 X%=INT(RND(1)*10-1)
60 Y%=INT(RND(1)*10-1)
70 Z%=INT(RND(1)*10-1)
80 PRINT "RAAD DE 4 GETALLEN."
90 PRINT
100 P=0
110 INPUT "Uw getallen (4 X)";A,B,C,D
120 IF A=W% AND B=X% AND C=Y% AND D=Z% THEN GOTO 500
130 IF A=W% THEN P=P+1
140 IF B=X% THEN P=P+1
150 IF C=Y% THEN P=P+1
160 IF D=Z% THEN P=P+1
170 PRINT P;" goed"
180 GOTO 100
190 END
500 COLOR 15,6,7
510 PRINT : PRINT : PRINT"!!!!!! JUIST !!!!!!"
520 END
```

10. STRINGS: de computer als tekstverwerker

Wat is een string?

In dit hoofdstuk gaan we de mogelijkheden van de computer vergroten. Tot nu toe zijn alleen nog maar cijfers en getallen verwerkt en is tekst gebruikt om iets weer te geven op het scherm. Nu bekijken we hoe tekst bewerkt en verwerkt kan worden door de MSX.
Om tekst te kunnen verwerken gebruikt de MSX zogenaamde STRINGS (=slierten, vrij vertaald: rijtjes). Een string is een aaneengeschakelde rij van karakters. Elk karakter kan gebruikt worden en de string kan net zo lang worden als u wilt, tot een maximum van 255 karaktertekens.
U moet er alleen altijd aan denken dat u de computer meedeelt dat u hem wilt laten werken met strings. De computer mag een karakter (bijvoorbeeld a of X of %) niet als variabele beschouwen en ook niet (bijvoorbeeld 1 of 6 of 34627.8) als getal. Daarom moet u de hele string altijd tussen aanhalingstekens (") zetten en de string een naam geven gevolgd door een dollarteken ($).

```
LET A$ = "string"
```

Dit wil zeggen: beschouw de karakters s-t-r-i-n-g als een string met de naam A$.
U herkent dit vast al. Waar heeft u dit eerder gebruikt? Inderdaad bij de PRINT-opdracht. Daar krijgt de string geen naam en hoeft de computer de string alleen maar af te drukken. Maar ook bij de PRINT-opdracht moet de computer de karakters niet als variabelen of getallen beschouwen en ze niet bewerken. Daarom staan de af te drukken karakters tussen aanhalingstekens.
We gaan een stapje verder. Het is mogelijk om de MSX een serie ingetikte karakters als string te laten verwerken. Daarvoor gebruikt u dezelfde INPUT opdracht als bij de invoer van getallen. Alleen vermeldt u nu dat u een string invoert en zorgt u ervoor dat de string opgeslagen wordt in een variabele die een string kan bevatten ($).

```
INPUT "tik iets in";A$
```

Als u nadat u de mededeling op het scherm gezien heeft iets intikt en daarna op RETURN drukt, beschouwt de computer de INPUT als een string. U heeft immers door het dollarteken achter de variabele A te kennen gegeven dat wat er ingetikt wordt als string gelezen moet worden.
We gebruiken de tot nu toe opgedane kennis voor een tamelijk lang

maar eenvoudig programma. Leest u het eerst een keer door. Als u het niet helemaal begrijpt kunt u het beter eerst intikken en laten runnen om te zien wat het programma doet. Als u het helemaal begrijpt kunt u de regels naar eigen behoefte veranderen. U heeft met dit programma het allereerste begin van een tekstverwerker op de MSX.

```
10 SCREEN 0
20 WIDTH 40 : KEY OFF
30 INPUT "Naam van jarige: ";N$
40 INPUT "Adres: ";A$
50 INPUT "Plaats: ";P$
60 INPUT "Verjaardagsdatum: ";V$
70 INPUT "Telefoonnr.: ";T$
80 CLS
90 PRINT:PRINT
100 PRINT N$
110 PRINT A$
120 PRINT P$
130 PRINT:PRINT
140 PRINT "Beste ";N$;","
150 PRINT
160 PRINT "Op deze heugelijke dag, ";V$
170 PRINT "feliciteer ik jou, ";N$
180 PRINT "hartelijk met je verjaardag."
190 PRINT "Ik hoop dat je vandaag, ";V$
200 PRINT "een leuke dag zult hebben."
210 PRINT "Hoewel ik het erg druk heb, probeer"
220 PRINT "ik op de ";A$;" in"
230 PRINT P$;" langs te komen."
240 PRINT "Lukt dat niet, dan draai ik ";T$
250 PRINT "om je per telefoon nog even te feli-"
260 PRINT "citeren. Tot dan, ";N$
270 PRINT "Hartelijke groeten,"
280 PRINT:PRINT
290 PRINT "      XXXXXXXXXXXXXXXX"
300 GOTO 300
```

Vul bij de XX-en uw eigen naam in en u heeft een persoonlijke verjaardagsbrief aan een van uw vrienden. Als u een printer heeft kunt u hem zelfs op papier laten zetten en zodoende verzenden. Maar dit is niet alleen een persoonlijke brief aan die vrien(in), maar door het programma nog een keer te runnen kunt u een andere naam met bijhorend adres en verjaardagsdatum intikken en zodoende heeft u voor al uw vrienden een persoonlijke verjaardagsbrief. Uiteraard blijft het nu beperkt tot een grapje op uw beeldscherm maar stel u maar eens voor dat u de beschikking heeft over zo'n programma met een keurige tekst, een computer met een luxe printer waarvan het schrift niet te onderscheiden is van dat van

een gewone schrijfmachine en de mogelijkheid om de namen en de adressen door de computer in te laten vullen. Ziedaar de manier waarop al die nette, vriendelijke en hoogst 'persoonlijke' brieven van boekenverkopers, cursusaanbieders en reclamebureaus tot stand komen.
Voor echte tekstverwerking komt wel wat meer kijken. Een tekstverwerkend programma moet alle ingetikte tekst kunnen bewerken en verwerken en opslaan. Tot de mogelijkheden van zo'n programma horen:

- correctie van tekst op een willekeurige plaats
- sorteren op alphabet
- opzoeken van bepaalde woorden of karaktercombinaties
- lay-out van een bladzijde
- tellen van het aantal ingevoerde tekens of woorden.

Op het moment dat dit boek geschreven wordt bestaan er nog maar een paar tekstverwerkende programma's voor de MSX. De verwachting is dat dit aantal snel zal stijgen en dat de programma's ook ontwikkeld zullen worden op diskettes, waardoor de snelheid aanzienlijk toeneemt.

Stringvariabelen

Voordat we verder gaan met de strings halen we eerst een stukje over de variabelen op. In het hoofdstuk 'Variabelen' leerde u rekenen met letters in plaats van cijfers. Zo'n letter gaf aan dat er op die plaats verschillende waarden ingevuld konden worden.
Daarnaast leerde u dat de inhoud van een variabele niet alleen een getal hoeft te zijn, maar ook een reeks van lettertekens kan zijn. Zo'n reeks lettertekens wordt een string genoemd. Let op! Sommige vertalingen van Engelse boeken spreken over een rij, waar in dit boek gesproken wordt over een string (het Engelse woord). Zo is een stringvariabele in sommige vertalingen een rijvariabele.
Om de computer duidelijk te maken wanneer een bepaalde letter, een bepaalde variabele bedoeld is als naam voor een getal en wanneer een variabele bedoeld is als naam voor een string, plaatsen we ter onderscheiding een dollarteken achter de naam van een string.

Rekenen met strings en getallen

De computer beschouwt dat wat u hem aangeeft als string niet als variabele of als te verwerken getal. De maximale lengte van een string is 254 lettertekens (een programmaregel mag ook niet meer dan 254 tekens bevatten). De minimumlengte van een string is nul

karakters. Dit heet een nulstring of een lege string. U geeft dat als volgt aan:

```
LET A$ = ""
```

De aanhalingstekens worden niet meegeteld als horend bij de string. Ze geven slechts het begin en einde aan. De MSX geeft elke stringvariabele die nog niet van een inhoud is voorzien de inhoud leeg. Als u tikt:

```
RESET     (de aparte RESET-toets indrukken, naar BASIC)
PRINT Q$
```

Drukt de computer niets op het scherm af en meldt daarna 'Ok'. De inhoud van Q$ is leeg, ook al heeft u Q$ nog helemaal niet gedeklareerd (van een waarde voorzien).
Omdat elk letterteken in een string voor kan komen is het goed er even bij stil te staan dat ook een getal een string kan zijn. Neem bijvoorbeeld het letterteken 7. U kunt dit karakter op twee manieren bekijken:

```
als een getal:   7
als een string: "7"
```

U kunt van de ene manier van kijken naar de andere overschakelen. We nemen een string met de naam A en de inhoud van die string is het letterteken 7.

```
10 LET A$="7"
```

De twee manieren waarop we de 7 kunnen beschouwen kunnen niet zomaar samen gebruikt worden. Het zal niet lukken de MSX de opdracht te geven:

```
20 PRINT A$+8
```

U heeft de computer eerst verteld dat de 7 een string was (met de naam A) en de computer heeft geleerd om strings niet te beschouwen als variabelen of getallen. Dus blijft de computer er vanaf en geeft hij bij het runnen de foutmelding:

```
Type mismatch in 20 (type klopt niet in regel 20)
```

U heeft het type string en getal door elkaar gebruikt. Dat past niet.
Het is echter wel mogelijk om strings in getallen om te zetten en andersom. Allereerst wordt elk karakter in de computer opgeslagen als een getal. Deze getallen stammen van de ASCII-set. Hierbij

heeft elk karakter een bepaalde waarde. U vindt de waarden van de karakters in de bijlagen van dit boek. U kunt er ook een kort programma voor maken. De MSX kent een opdracht om de ASCII waarde van een karakter als resultaat te geven. Dit is

ASC(..)

Tussen de haakjes plaats u een karakter of de naam van een string (een stringvariabele). Een programma dat de gebruiker telkens een karakter laat intikken om vervolgens de ASCII waarde te geven is:

```
10 CLS
20 INPUT "Karakter ";A$
30 PRINT "ASCII waarde van ";A$;" = ";ASC(A$)
40 GOTO 20
```

U ziet dat in regel 40 bij de ASC opdracht geen aanhalingstekens staan. Dat is omdat de computer niet de ASCII waarde van de string 'A$' moet geven, maar de ASCII waarde van de string met de naam A$. Let u er ook op dat de ASC opdracht alleen de ASCII waarde van het eerste letterteken van een string geeft. Als u verschillende karakters achter elkaar intikt bij de INPUT krijgt u toch alleen de ASCII waarde van het eerste karakter.
De andere kant op kan natuurlijk ook. U geeft de ASCII waarde op en de computer vertelt welk teken daarbij hoort.

CHR$(..) (CHaRacter=karakter)

Met het volgende karakter kunt u snel een groot deel van de karakters van de MSX zien.

```
5 SCREEN 1 : KEY OFF
10 FOR X=33 TO 255
20 PRINT CHR$(X);
30 NEXT X
```

Run dit programma zowel met screen 0 als screen 1 (verander regel 5). Ziet u verschil? Er zijn nog enkele vermeldenswaardige aspecten aan dit kleine programma. Het alphabet kent 26 letters. De MSX heeft dus 52 waarden nodig voor alle letters (hoofdletters en kleine letters). Daarnaast is nog een stel waarden nodig voor allerlei tekens en voor de cijfers. De eerste 32 waarden heeft de computer nodig voor allerlei opdrachten aan het beeldscherm en randapparatuur. Om de in de bijlage genoemde tekens toch te zien, moeten deze waarden met een dubbele code opgeroepen worden: eerste een CHR$(1), en dan een CHR$ met de code, verhoogd met 64.

Het programma voor alle karakters wordt:

```
10 SCREEN 1 : KEY OFF
20 FOR X=1 TO 31
30 PRINT CHR$(1);CHR$(X+64);
40 NEXT X
50 FOR X=32 TO 255
60 PRINT CHR$(X);
70 NEXT X
```

Voor ons probleem, het optellen van A$ (waarin zich '7' bevindt) en 8 geeft dit alles echter geen oplossing. Er is nog een stel opdrachten die het mogelijk maken om strings waarin zich alleen getallen bevinden om te zetten in de waarde van die getallen en andersom om getallen om te zetten in strings met als inhoud die getallen. Om de waarde van een string te krijgen gebruikt u:

```
VAL(..)                          (VALue = waarde)
```

Als we dit toepassen op de string A$ ("7") tikken we:

```
LET X=VAL(A$)
```

De inhoud van A$ is '7', dus geeft VAL(A$) de waarde 7. De variabele X heeft als waarde 7 gekregen.
De VAL opdracht werkt alleen als tussen de haakjes een string (of de naam van een string) staat waarin getallen voorkomen. Het eerste karakter moet een plusteken (+) een minteken (-) of een cijfer zijn. Is dit niet zo, dan wordt de door VAL gegeven waarde nul. Enkele voorbeelden:

```
LET A$="88796"
LET X=VAL(A$)
PRINT X                          resultaat: -88796

LET X=VAL("5456")
PRINT X                          resultaat: 5456

PRINT VAL("getal")               resultaat: 0

PRINT VAL("3 getallen)           resultaat: 3

PRINT VAL("3*3=9")               resultaat: 3
```

Andersom kan natuurlijk ook. Daarvoor gebruikt u:

STR$(..)

Elke waarde die hier tussen haakjes geplaatst is, wordt tot string gemaakt. Natuurlijk kan ook een variabelenaam tussen de haakjes geplaatst worden. De waarde van de variabele wordt dan tot string gemaakt. Bijvoorbeeld:

```
LET X=654
LET A$=STR$(X)
PRINT A$                                    resultaat: 654
```

Merkt u op dat aan de plaats van de uitkomst soms te zien is of een getal een string of een getal is. Een getal wordt immers altijd voorafgegaan door een spatie of door een minteken. De waarde of de variabelenaam tussen de haakjes moet numeriek zijn. De optelling die we wilden maken van A$ (inhoud '7') met 8 lossen we als volgt op:

```
LET A$="7"
PRINT VAL(A$)+8                             resultaat: 15
```

Naast de hiervoor genoemde mogelijkheden om strings om te zetten in getallen en vice versa, zijn er nog enkele stringfunkties voor het omzetten van strings in hexadecimale, octale of binaire getallen. Om hier iets meer over te kunnen vertellen, is een kort intermezzo over getalstelsels nodig.

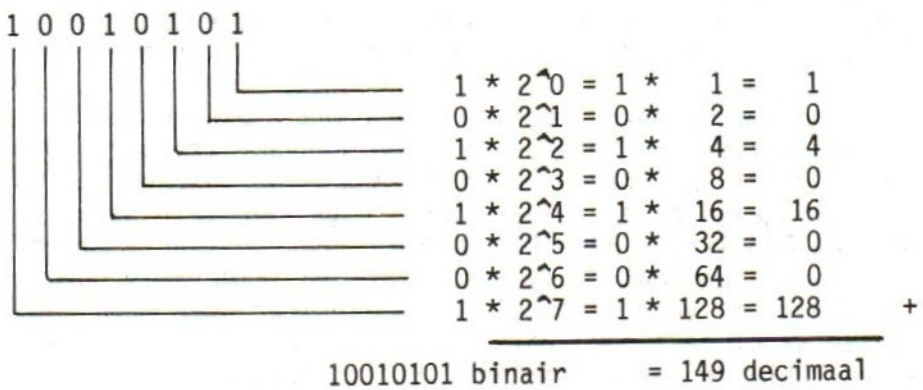
Getalstelsels

De mens rekent meestal in een tientallig stelsel. Dat wil zeggen dat hij van 0 tot en met 9 telt en eventueel weer opnieuw bij 0 begint, daarbij onthoudend hoe vaak hij al tot negen geteld heeft.
Toch is rekenen met andere soorten getallen niet onmogelijk. De meeste lezers zullen het Engelse geldsysteem van vroeger nog wel kennen. Heel moeilijk voor iemand die gewend is aan een decimaal (=tientallig) geldstelsel, maar voor Engelsen zelf geen enkel probleem. Het Engelse en Amerikaanse systeem voor het meten en weergeven van lengtematen (yards, feet en inches) is ook niet tientallig, maar is voor de dagelijkse gebruiker geen probleem.
Een computer werkt zelf ook niet decimaal. Een computer bestaat uit heel veel kleine schakelaartjes. Elke schakelaar kan of aan of uit staan. Daarom kan een computer slechts tot een tellen: 0 en 1. Daarna begint de computer opnieuw bij 0 waarbij onthouden wordt hoe vaak er al tot 1 geteld is. Deze manier van tellen wordt het twee-tallige of binaire getalsysteem genoemd.

Het lijkt onzinnig om ons te verdiepen in de interne telwijze van de computer als het resultaat toch altijd in een voor de mens eenvoudig te begrijpen tientallige vorm op het scherm komt te staan. In de hoofdstukken over grafiek en kleur zult u al snel merken dat het erg handig is als u weet hoe de computer intern allerlei gegevens opslaat. Als u later in machinetaal gaat programmeren dan is deze kennis zelfs onontbeerlijk.
Een MSX-computer is een 8-bits computer. Dat betekent dat de geheugenbits gegroepeerd zijn in groepen van 8-bits. Zo'n groep heet een byte. Binnen zo'n groep van 8 bits kan geteld worden tot het getal 11111111 (binair). Dat is gelijk aan 255 decimaal. Omdat elk binair cijfer staat voor een telling van 0 tot en met 1, kunnen binaire getallen omgezet worden in decimale door gebruik te maken van machten van 2.

Binair	Decimaal	Binair	Decimaal
00000000	0	00001011	11
00000001	1	00001100	12
00000010	2	00001101	13
00000011	3	00001110	14
00000100	4	00001111	15
00000101	5	00010000	16
00000110	6	00010001	17
00000111	7	00010010	18
00001000	8	00010011	19
00001001	9	00010100	20
00001010	10	00010101	21

Het omzetten van een binair getal naar een decimaal getal gaat als volgt:



Let op! Het tellen van de machten van twee begint rechts. De kleinste macht staat het meest naar rechts. Dat geldt immers ook voor decimale cijfers: het kleinste cijfer staat het meest naar

rechts (de eentallen, links daarvan de tientallen, links daarvan de honderdtallen, links daarvan de duizendtallen, etc.)
U ziet dat het omzetten met machten van twee gaat. Andersom dient u zo groot mogelijke machten van twee van het getal af te trekken. Om 95 decimaal om te zetten in een binair getal gaat u als volgt te werk:

```
95      aftrekken 2^7 ?          lukt niet       bit 8 = 0
95      aftrekken 2^6 ?          lukt (rest 31)  bit 7 = 1
31      aftrekken 2^5 ?          lukt niet       bit 6 = 0
31      aftrekken 2^4 ?          lukt (rest 15)  bit 5 = 1
15      aftrekken 2^3 ?          lukt (rest 7)   bit 4 = 1
 7      aftrekken 2^2 ?          lukt (rest 3)   bit 3 = 1
 3      aftrekken 2^1 ?          lukt (rest 1)   bit 2 = 1
 1      aftrekken 2^0 ?          lukt (rest 0)   bit 1 = 1
```

95 decimaal = 01011111 binair.

Let op dat bit 8 de meest linkse bit is, en bit 1 de meest rechtse bit. Zie hiervoor.
Om de computer duidelijk te maken dat een bepaald getal een binair getal is en geen decimaal getal, plaatst u &B voor het getal.
Het omzetten van een binair getal (van acht bits) in een decimaal getal kan ook door de computer gebeuren:

```
PRINT &B00110100              resultaat: 52
```

Het omzetten van een decimaal getal in een binair getal gaat met behulp van een stringfunktie:

```
BIN$(..)
```

Tussen haakjes plaatst u een decimaal getal en de computer geeft een string met als inhoud het binaire equivalent weer:

```
PRINT BIN$(237)               resultaat: 11101101
```

Enige kennis over binaire getallen is ook voor de BASIC programmeur belangrijk voor het werken met grafiek. De andere twee getalstelsels die de MSX kent, zijn vooral van belang voor de machinetaalprogrammeur en voor het verkrijgen van enkele speciale effecten. Volledigheidshalve worden het achttallig stelsel en het zestientallig stelsel hier kort behandeld.

Het achttallig of octale stelsel is een getalsysteem waarvan het grondtal 8 is. Elk cijfer stelt een macht van 8 voor. Vergelijk dit met het binaire stelsel hiervoor waarbij elk cijfer een macht

van twee voorstelde, of het tientallige stelsel waarbij elk getal een macht van 10 voorstelt (1, 10, 100, 1000 etc.). In het octale stelsel wordt geteld van 0 tot en met 7, daarna wordt weer bij 0 begonnen waarbij onthouden wordt hoe vaak er al tot 7 geteld is.

decimaal	octaal	decimaal	octaal
0	0	16	20
1	1	17	21
2	2	18	22
3	3	19	23
4	4	20	24
5	5	21	25
6	6	22	26
7	7	23	27
8	10	24	30
9	11	25	31
10	12	26	32
11	13	27	33
12	14	28	34
13	15	29	35
14	16	30	36
15	17	31	37
		32	40

Het omzetten van getallen uit het ene getalstelsel in het andere gaat net als bij de binaire getallen. Het meest rechtse cijfer staat voor 8^0=1, het cijfer links daarvan staat voor 8^1=8, het cijfer daar weer links van staat voor 8^2=64, etc.
Het getal 2306 octaal wordt als volgt omgezet in een decimaal cijfer:

```
2 3 0 6
        6 * 8^0 =   6 *   1 =    6
      0 * 8^1 =   0 *   8 =    0
    3 * 8^2 =   3 *  64 =  192
  2 * 8^3 =   2 * 512 = 1024    +

2306 octaal =                  1222 decimaal
```

Het omzetten van een decimaal getal in een octaal getal:

```
967 (decimaal) aftrekken 8^4=4096 ?  lukt niet         0
967            aftrekken 8^3= 512 ?  lukt (rest 455)   1
455            aftrekken 8^2=  64 ?  lukt 7X (rest 7)  7
  7            aftrekken 8^1=   8 ?  lukt niet         0
  7            aftrekken 8^0=   1 ?  lukt 7X (rest 0)  7
                                     ------------------
                                     967 decimaal =  1707 octaal
```

Ook voor octale getallen moet de computer een speciaal teken hebben om ze niet te verwarren met decimale getallen. Dit teken is: &O. De computer kan het omrekenwerk voor u doen. Van octaal naar decimaal gaat als volgt:

```
PRINT &O76531                    resultaat: 32089
```

Van decimaal naar een string die een octaal getal bevat:

```
PRINT OCT$(5546)                 resultaat: 126521
```

Als laatste getalsysteem komt het zestientallig of hexadecimale stelsel aan bod. Dit stelsel heeft een grondtal 16. Elke cijfer staat voor een macht van 16. Er is een extra probleem verbonden aan het hexadecimale stelsel. Er bestaan slecht 10 normale cijfers, terwijl het hexadecimale stelsel er zestien nodig heeft. Dit is opgelost door de ontbrekende cijfers aan te geven met de letters A tot en met F.

decimaal	hexadecimaal	decimaal	hexadecimaal
0	0	16	10
1	1	17	11
2	2	18	12
3	3	19	13
4	4	20	14
5	5	21	15
6	6	22	16
7	7	23	17
8	8	24	18
9	9	25	19
10	A	26	1A
11	B	27	1B
12	C	28	1C
13	D	29	1D
14	E	30	1E
15	F	31	1F
		32	20

Voor het met de hand omzetten van decimale getallen naar hexadecimale getallen en vice versa wordt de lezer verwezen naar de uitleg hierover bij de binaire en de octale getallen. Met hexadecimale getallen gaat het precies hetzelfde. Nu moet er echter gewerkt worden met machten van 16 (1, 16, 256, 4096, etc.). De cijfers 10 tot en met 15 worden in het hexadecimale getalstelsel aangegeven met A tot en met F.
De computer gebruikt het symbool &H om hexadecimale getallen van decimale te kunnen onderscheiden.
Ook deze omzettingen kan de computer voor zijn rekening nemen:

```
PRINT &H7FB4                    resultaat: 32692
```

Of van decimaal naar een string met een hexadecimaal getal als inhoud:

```
PRINT HEX$(6797)                resultaat: 1A8D
```

Rekenen met strings

Naast het werken met getallen en strings en het eventueel omzetten van het een naar het ander kan de MSX ook alleen string verwerken. Daarvoor kent MSX een stel aparte stringopdrachten. Naast de stringopdrachten die u in dit hoofdstuk al tegen gekomen bent zijn dat:

```
STRING$(..,..)      (string string)
SPACE$(..)          (space = spatie)
LEFT$(..,..)        (left = links)
MID$(..,..,..)      (middle = midden)
RIGHT$(..,..)       (right = rechts)
LEN(..)             (lenght = lengte)
INSTR(..,..,..)     (in string = binnen string)
```

De eerste twee opdrachten zijn vooral bedoeld voor het maken van mooie strings. Met STRING$ kunt u een bepaald letterteken verschillende malen herhalen:

```
PRINT STRING$(10,36)            resultaat: $$$$$$$$$$
```

Door deze opdracht wordt het karakter horend bij ASCII code 36 (het tweede getal achter STRING$) 10 maal herhaald. Het resulaat kan natuurlijk weer in een string opgeslagen worden.
Het is ook mogelijk het eerste karakter van een opgegeven string een aantal malen te laten herhalen:

```
LET A$="MSX"
LET B$=STRING$(14,A$)
PRINT B$              resultaat: MMMMMMMMMMMMMM
```

De tweede opdracht zorgt voor een aantal spaties. Het aantal gewenste spaties wordt tussen de haakjes geplaatst:

```
PRINT "*";SPACE$(8);"**"    resultaat: *        **
```

De opdracht LEFT$(..,..) geeft de karakters die links in de opgegeven string staan. Bijvoorbeeld:

```
A$="boekje"
PRINT LEFT$(A$,3)              resultaat: boe
```

Tussen de haakjes vult u eerst de naam van de string die onderzocht moet worden in. Daarna het aantal karakters dat genomen moet worden. In het voorbeeld nam de computer de drie meest linkse karakters.
Er bestaat net zo'n opdracht voor de rechterletters:

```
A$="boekje"
PRINT RIGHT$(A$,2)             resultaat: je
```

Om de mogelijkheden helemaal compleet te maken kent MSX BASIC ook MID$. Deze neemt karakters ergens uit de string.

```
A$="boekje"
PRINT MID$(A$,3,2)             resultaat: ek
```

Bij MID$ vult u eerst de naam van de string in die onderzocht moet worden, dan de plaats van het karakter waar begonnen moet worden met kopieren en dan het aantal karakters waarna gestopt moet worden met kopieren. De plaatsen worden geteld van links naar rechts. Spatie's, leestekens e.d. zijn ook plaatsen die meetellen. Als u geen tweede getal invult achter MID$ wordt de rest van de string na het opgegeven eerste karakter gekopieerd. Vult u als laatste getal een getal in dat groter is dan het aantal nog beschikbare letters, dan wordt de rest van de string tot het einde gekopieerd. Vult u als laatste getal een 0 in, dan wordt een lege of nulstring gegeven. Vult u als eerste getal (de karakterpositie waar begonnen moet worden met kopieren) een getal in dat groter is dan de lengte van de string dan wordt ook een lege string gegeven. Voorbeelden:

```
A$="boek,pen"
PRINT MID$(A$,5)              resultaat: ,pen
PRINT MID$(A$,5,99)           resultaat: ,pen
PRINT MID$(A$,5,0)            resultaat:          (lege string)
PRINT MID$(A$,10,3)           resultaat:          (lege string)
```

Met dezelfde MID$ opdracht is het mogelijk om delen van strings te veranderen in andere strings. De algemene vorm daarvoor is:

```
MID$(..,..,..)=q$
```

De stringvariabele en de getallen tussen de haakjes van MID$ bepalen welke letters van welke string vervangen worden. De letters die ervoor in de plaats komen zijn afkomstig van de string die achter het gelijkteken geplaatst is:

```
A$="boek,pen"
B$="om dezelfde reden"
PRINT A$                          resultaat: boek,pen
PRINT MID$(A$,3,4)                resultaat: ek,p
MID$(A$,3,4)=B$
PRINT MID$(A$,3,4)                resultaat: om d
PRINT A$                          resultaat: boom den
```

Het is niet nodig de gehele string op te geven achter het gelijkteken. Ook daar kan gebruik gemaakt worden van MID$:

```
A$="boek,pen"
B$="om dezelfde reden"
MID$(A$,3,4)=MID$(B$,13,4)
PRINT A$                        resultaat: boredeen
```

Op precies dezelfde wijze kan in plaats van MID$ ook RIGHT$ achter het gelijkteken gebruikt worden:

```
A$="boek,pen"
B$="om dezelfde reden"
MID$(A$,5,4)=RIGHT$(B$,4)
PRINT A$                        resultaat: boekeden
```

De lengte van de oorspronkelijke string (in de voorbeelden telkens A$) kan door het gelijkteken niet veranderd worden.

Naast het gelijk maken van (delen van) strings is het ook mogelijk de bewerking + op strings los te laten. Daardoor worden strings gewoon opgeteld. Niet de waarde van de strings, maar de inhoud wordt opgeteld. Dat wil zeggen dat de karakters van de ene string worden geplaatst achter de karakters van de andere string:

```
A$="3"
B$="7"
PRINT A$+B$          resultaat: 37  (en dus niet 10 !)

A$="abcdefg"
B$="uvwxyz"
PRINT A$+B$          resultaat: abcdefguvwxyz
PRINT B$+A$          resultaat: uvwxyzabcdefg
```

De volgorde waarin de karakters van de twee strings geplaatst worden is afhankelijk van de volgorde waarin de strings opgeteld worden.
Het volgende programmagrapje maakt gebruik van de LEFT$, RIGHT$, LEN en het optellen van strings om een naam op toppopwijze op uw scherm te zetten. Voer na het runnen uw naam in en zie uzelf over het scherm golven. Probeer om met de in dit hoofdstuk behandelde ingredienten andere effecten te bereiken. Juist een computer die een bepaalde handeling vele malen snel kan herhalen zorgt voor bijna onbeperkte mogelijkheden met dit soort grapjes.

```
10 SCREEN 1 : KEY OFF
20 INPUT "Uw naam s.v.p. ";X$
30 LET NA$="******** "+X$+" ********"
40 COLOR 15,INT(RND(1)*15),INT(RND(1)*15)
50 FOR X=1 TO LEN(NA$)
60 PRINT LEFT$(NA$,X)
70 NEXT X
80 FOR X=1 TO LEN(NA$)
90 PRINT TAB(X) RIGHT$(NA$,LEN(NA$)-X)
100 NEXT X
110 FOR X=1 TO LEN(NA$)
120 PRINT TAB(LEN(NA$)-X) RIGHT$(NA$,X)
130 NEXT X
140 FOR X=1 TO LEN(NA$)
150 PRINT LEFT$(NA$,LEN(NA$)-X)
160 NEXT X
170 GOTO 40
```

Om de lengte van een string te kunnen bepalen gebruikt u de opdracht LEN.

```
A$="boek,pen"
```

```
PRINT LEN(A$)                    resultaat: 8
```

Bij deze opdracht hoeft u alleen de naam van de te tellen string op te geven.

Als laatste de opdracht INSTR. Deze wordt gebruikt om te kijken of een bepaalde string voorkomt in een andere string.

```
A$="boek,pen"
B$="k"
PRINT INSTR(A$,B$)               resultaat: 4
```

Achter INSTR vult u eerst de naam van de te doorzoeken string in en daarna het karakter of de naam van de string waarvan gecontroleerd moet worden of hij in de andere string voorkomt. Soms is het gemakkelijk als er niet aan het begin van de string begonnen wordt met zoeken, maar ergens verderop. Daarvoor kunt u een derde gegeven invullen achter INSTR:

```
A$="boek,pen,drukletter"
B$="k"
PRINT INSTR(5,A$,B$)             resultaat: 13
```

Normaal gesproken geeft de INSTR opdracht de plaats weer waar de gezochte string voor het eerst voorkomt in de te onderzoeken string. Er wordt daarna niet verder gezocht. Maar de mogelijkheid om het zoeken elders dan bij het begin van de string te laten beginnen, samen met het feit dat INSTR de waarde 0 geeft als de string helemaal niet voorkomt in het onderzochte deel, zorgen ervoor dat we de hele string kunnen onderzoeken op het aantal keer dat een bepaalde string erin voorkomt.

```
10 SCREEN 1 : KEY OFF
20 INPUT "Voer een string (max. 254) in ";A$
30 PRINT
40 INPUT "Te zoeken letter(s) ";B$
50 X=0
60 T=0
70 PRINT
80 X=INSTR(X+1,A$,B$)
90 T=T+1
100 IF X<> 0 THEN GOTO 80
110 PRINT "De letter(s) '";B$;"' komt";T-1;"keer voor."
```

Dit programma geeft u in regel 20 de mogelijkheid een string in te voeren van maximaal 255 tekens. Daarna kunt u door regel 40 een string opgeven waarnaar gezocht moet worden. De laatste string mag langer dan één karakter zijn. Elke keer dat de

programmalus doorlopen wordt, wordt het zoeken naar de string een plaats verder dan de vorige vondst begonnen. De eerste keer wordt bij het begin begonnen. Elke keer dat de opgegeven string gevonden wordt, wordt er 1 opgeteld bij T (teller). Dit gaat net zo lang door totdat de string niet meer gevonden wordt. Dan is X gelijk aan 0 en stapt de computer uit de lus. Het resultaat wordt afgedrukt door regel 110. Er moet 1 van T afgetrokken worden omdat ook de keer dat X gelijk is aan 0 (en er dus geen string meer gevonden werd) 1 opgeteld werd bij T.

Vergelijken van strings

Naast alle hierboven genoemde mogelijkheden, kunnen strings ook vergeleken worden. U gebruikt daarvoor dezelfde vergelijkingstekens als bij de rekenkunde:

=	gelijk aan
<>,><	niet gelijk aan
<	kleiner dan
>	groter dan
<=,=<	kleiner dan of gelijk aan
>=,=>	groter dan of gelijk aan

Deze vergelijkingen krijgen bij het gebruik van strings wel een eigen betekenis. Twee strings zijn alleen gelijk (=) als ze even lang zijn en precies dezelfde karakters bevatten. In alle andere gevallen zijn twee strings niet precies gelijk (<>). Strings worden karakter voor karakter vergeleken. Een karakter is groter dan het karakter waarmee vergeleken wordt, als de ASCII waarde van de eerste hoger is. Een string is dus groter dan een andere string als het eerste karakter groter is, dat wil zeggen later in het alphabet of de ASCII codelijst staat (B>A, Z>A, F>B) en de string minstens zo lang is als de string waarmee vergeleken wordt. Zijn de eerste karakters gelijk, dan wordt gekeken naar het tweede karakter. Zijn die ook gelijk, dan wordt gekeken naar het derde karakter, etc. Precies zoals in een woordenboek de woorden gerangschikt staan. Zijn alle karakters gelijk dan wordt gekeken of een bepaalde string langer is. De langste is de grootste (ABCD>ABC, XYZ>XY).
Een extra moeilijkheid is het verschil tussen hoofd- en kleine letters. Hoewel te verwachten is dat kleine letters kleiner zijn dan hoofdletters is dit niet zo. A<a, Ab<ab. Hoofdletters zijn kleiner dan kleine letters. U kunt dit zien aan de ASCII waarden. De ASCII waarden van de kleine letters zijn groter dan die van de hoofdletters.
De overige vergelijkingstekens spreken verder voor zichzelf.

Een leuk programma dat strings manipuleert en vergelijkt is de zogenaamde
palindroomtester. Een palindroom is een woord dat van voor naar achter en van achter naar voor gelezen hetzelfde is. Bijvoorbeeld het woord 'pap'. Maar er bestaan ook mooiere zoals: renner, avondnova of de beroemde parterreserretrap.
Dit programma laat u een woord invoeren, draait het om en controleert of het resultaat hetzelfde is.

```
10 SCREEN 0 : KEY OFF
20 INPUT "Een woord aub.    ";A$
30 Z$=""
40 FOR X= LEN(A$) TO 1 STEP -1
50 Z$=Z$+MID$(A$,X,1)
60 NEXT X
70 PRINT "Omgekeerd krijgt u: ";Z$
80 PRINT
90 IF Z$=A$ THEN PRINT "Dit is een PALINDROOM"
100 PRINT
110 GOTO 20
```

Nog leuker zijn de palindromen die uit verschillende woorden bestaan. Indien die getest moeten worden op hun omkeerbaarheid zitten we met het probleem van de spaties. In het volgende programma worden twee extra strings gemaakt. In de eerste komen de omgekeerde woorden zonder spaties. In de tweede wordt het omgekeerde resultaat hiervan geplaatst. Zijn deze twee gelijk, dan is er sprake van een palindroom.

```
10 SCREEN 0 : KEY OFF
20 INPUT "Een woord aub.    ";A$
30 Z$=""
40 G$=""
50 H$=""
60 FOR X=LEN(A$) TO 1 STEP -1
70 Z$=Z$+MID$(A$,X,1)
90 IF MID$(A$,X,1)<>" " THEN G$=G$+MID$(A$,X,1)
100 NEXT X
110 FOR P=LEN(G$) TO 1 STEP -1
120 H$=H$+MID$(G$,P,1)
130 NEXT P
140 PRINT "Omgekeerd krijgt u: ";Z$
150 PRINT
160 IF G$=H$ THEN PRINT "Dit is een PALINDROOM"
170 PRINT : GOTO 20
```

Een paar mooie voorbeelden om in te tikken: a man a plan a canal panama, mooie zeeman nam anna mee zei oom, en natuurlijk: mooi dit idioom. Probleem bij het programma blijft dat de computer niet kan weten waar woorden in het nederlands moeten worden afgebroken (zelfs engels is wat dat betreft te moeilijk voor hem). Als u de spaties op vreemde plaatsen lelijk vindt staan kunt u in regel 140 in plaats van Z$ ook G$ laten afdrukken. U krijgt dan het omgekeerde woord zonder spaties.
U kunt het programma zodanig verbeteren dat ook leestekens worden verwaarloosd. Een extra complicatie treedt daarbij op: Als u bij een stringinput opdracht een komma gebruikt, denkt de computer dat u daarmee aangeeft dat de eerste string klaar is en de tweede string begint. Probeert u in bovenstaand programma maar in te tikken: pap,pap.
De computer geeft de melding:

?Extra ignored (het teveel genegeerd)

Alleen het eerste woord wordt omgedraaid. Door de komma dacht de computer dat u twee verschillende woorden opgaf, terwijl de INPUT opdracht maar om een enkel woord vroeg (A$).
U kunt dit opvangen door gebruik te maken van:

LINE INPUT

Door deze opdracht wordt een hele regel met lettertekens ingevoerd. De computer let daarbij niet op het gebruik van komma's. Het indrukken van RETURN bepaalt het einde van de op te geven string. Verander in bovenstaand programma in regel 20 de INPUT opdracht in een LINE INPUT opdracht en probeer het pap,pap voorbeeld nog een keer. Verander regel 90 met behulp van de AND operator zo dat ook leestekens uitgefilterd worden. Als u het programma zo verbeterd heeft kunt u nog veel meer palindromen verzinnen zoals: Koot, mannen, nam ei. Bie, mannen, nam 't ook.

Geheugenruimte bij strings

Alle strings worden in een apart geheugengebied (string area) opgeslagen. Dit gebied is bij het aanzetten van de computer beperkt van omvang. Zolang u als programmeur daar niets tegen doet blijft dat zo. Probeert u in het palindromenprogramma maar eens een heel lang woord in te voeren (tik twee regels vol met lettertekens). U krijgt dan de foutmelding

Out of string space (te weinig stringruimte)

In het stringgebied kunnen maximaal 200 lettertekens geplaatst worden. Omdat het laatste palindroomprogramma vier strings maakt, kan elke string slechts ongeveer 50 lettertekens bevatten.
Om de voor strings beschikbare geheugenruimte te vergroten gebruikt u:

```
CLEAR ..                    (wis)
```

Achter CLEAR zet u het aantal lettertekens dat het stringgebied moet kunnen bevatten. Wees hier niet te royaal mee, want alle geheugenruimte die u voor strings gebruikt kunt u niet meer ergens anders voor gebruiken.
Vergeet ook niet dat elke string maximaal 254 lettertekens lang kan worden. Komen er in het programma vier strings voor dan is het niet nodig meer ruimte te reserveren dan 1016 geheugenplaatsen (CLEAR 1016).
Verwar CLEAR niet met de CLS opdracht (CLEAR SCREEN). Deze laatste opdracht wist alleen het scherm (door het schermgeheugen te wissen).

11. LIJSTEN MET GEGEVENS

Inleiding

De tot nu in dit boek behandelde programma's werkten meestal met slechts enkele gegevens. Soms moest de computer zelf gegevens maken of een bepaald gegeven varieren.
Naast hun rekenfunktie staan computers echter vooral bekend om hun mogelijkheden tot het verwerken van grote hoeveelheden gegevens in tamelijk korte tijd. Dit hoofdstuk laat zien op welke manier deze gegevens aan de computer opgegeven kunnen worden en hoe de computer deze gegevens kan opslaan.
Tot nu toe werd hiervoor de INPUT of de LINE INPUT opdracht gebruikt. Deze opdracht zorgt ervoor dat de computer tijdens het programma om een gegeven vraagt. Dit gegeven wordt toegekend aan een variabele en daarmee gaat de computer verder aan het werk. Een andere mogelijkheid was via de LET opdracht. Deze zorgt ervoor dat een waarde aan een variabele toegekend wordt. Beide methoden hebben het nadeel dat het slechts om beperkte hoeveelheden gegevens gaat. Daarbij komt nog het bezwaar dat de gegevens zoals die tot nu toe via de INPUT opdracht werden ingevoerd slechts een programmarun lang blijven bestaan. De eenvoudigste manier om een stel gegevens aan de computer op te geven is door middel van een DATA (gegevens) lijst.

DATA-lijsten

Met de BASIC opdracht

```
DATA
```

kunt u een lijst gegevens aan de computer opgeven. De gegevens worden onderling gescheiden door komma's. Zowel getallen als strings zijn mogelijk. In de DATA opdracht hoeven de strings niet geopend en gesloten te worden door aanhalingstekens. Als er echter komma's, dubbele punten of spaties in de string voorkomen moet deze wel door aanhalingstekens omgeven worden. Strings met en strings zonder aanhalingstekens kunnen door elkaar in een DATA opdracht gebruikt worden.
Het is wel nodig dat de bij een string horende leesopdracht (zie hieronder) aan de computer duidelijk maakt dat het om een string gaat. Een lijst kan meer dan een programmaregel beslaan. In dat geval moet aan het begin van elke regel de DATA opdracht herhaald worden. Ondanks een verdeling over verschillende regels ziet de computer de lijst toch als een geheel.

De computer kan de gegevens van de lijst lezen door middel van de opdracht:

```
READ
```

De gegevens worden gelezen in de volgorde waarin ze in de datalijst staan van links naar rechts, van laag naar hoog regelnummer. Achter de READ wordt de naam gegeven van de variabele waaraan het gegeven toegekend dient te worden. Bijvoorbeeld:

```
10 READ A
20 PRINT A
30 DATA 318, 645, 75
```

of

```
10 READ A$
20 PRINT A
30 DATA voorbeeldsliert, tekst, tekst
```

Uit dit voorbeeld blijkt dat de DATA en READ opdracht niet bij elkaar hoeven te staan. Het is de gewoonte de datalijst aan het einde van het programma te plaatsen. Dit verhoogt de leesbaarheid van het programma.
Op bovenstaande manier gebruikt levert deze werkwijze niet zo veel op. Maar het is mogelijk aan verschillende variabelen waarden toe te kennen door middel van de datalijst:

```
10 READA,B,C
20 PRINT A,A*B,C
30 DATA 10,6,80,90,120
```

De computer houdt zelf bij waar hij in de datalijst gebleven was. Daardoor kunt u met een datalijst volstaan, terwijl op verschillende plaatsen in het programma naar deze lijst verwezen wordt:

```
10 READ A,B,C
20 PRINT A,A*B,C
:
:
70 READ A,E
80 PRINT A*E,D,E*D
:
:
200 DATA 1,2,3,4,5,6,7,8,9
```

In dit geval zijn alle gegevens in een programmaregel gezet. Programma's waarin verschillende keren verwezen wordt naar een

lange datalijst zijn beter leesbaar als u elke bij elkaar horende groep gegevens in een aparte dataregel zet. Dit maakt terugzoeken eenvoudiger.
Dat de computer zo aardig is om de tel bij te houden in de datalijst maakt het programmeren voor ons gemakkelijker. Stel dat we een eenvoudige lijst met telefoonnummers willen maken. Als we een naam intikken, moet de computer het bijhorende telefoonnummer voor ons opzoeken. Bij onderstaand programma zijn alle gegevens strings.

```
10 SCREEN 0 : KEY OFF
20 INPUT "NAAM ";NA$
30 PRINT
40 READ DA$,TE$
50 IF NA$<>DA$ THEN GOTO 40
60 PRINT TE$
100 DATA JAN,434423, PIET,0453-8876,GIJS,765453,MARIE,
    078-987876,KLAARTJE,243251
200 DATA KEES,020-565354,TRUUS,766545,SJEF,0931-766754/7654,
    GEERT,03454-9786
```

In dit programma leest de computer elke keer een naam uit de lijst (DA$) en een bijhorend telefoonnummer (TE$). Hij doet dit net zo lang to de opgegeven string gelijk is aan een in de lijst voorkomende naam (IF NA$<>DA$ THEN GOTO 40). Dan springt hij uit de lus en geeft het telefoonnummer weer. Het voordeel van zo'n programma is dat u een bijna eindeloze lijst namen met bijhorende telefoonnummers kunt intikken en dan de computer een telefoonnummer bij een bepaalde naam kunt laten opzoeken. Let u bij bovenstaand programma erop dat er verschil is tussen een hoofdletter en een kleine letter. Voor de MSX is Jan een ander dan JAN of dan jan.
Het is ook mogelijk een lijst met gegevens verschillende keren te gebruiken. Daarvoor kent de MSX de opdracht:

```
RESTORE              (herstel)
```

Deze opdracht beinvloedt de plaats waar in het geheugen van de MSX een pijltje bij de gegevens uit de lijst staat. Dit pijltje, een zogenaamde datawijzer (gegevenwijzer), zorgt ervoor dat de computer weet waar hij in de lijst gebleven was. RESTORE zonder enige toevoeging zorgt ervoor dat de datawijzer terugspringt naar het eerste gegeven achter de DATA opdracht in een programma.

```
10 READ A,B,C
20 PRINT A,B,C
:
:
40 RESTORE
50 READ D,E,F
60 PRINT D,E,F
:
:
100 DATA 1,2,3,4,5,6
```

De waarden van A, B en C zullen respectievelijk 1, 2 en 3 zijn. De waarden van D, E en F zijn ook 1, 2 en 3. De datawijzer is door de opdracht in regel 40 teruggesprongen naar het eerste gegeven.
Achter RESTORE kan ook een regelnummer opgegeven worden. In dat geval springt de datawijzer naar het eerste gegeven achter de DATA opdracht op de opgegeven regel.
RESTORE hoeft niet alleen gebruikt te worden om de computer terug te laten gaan naar gegevens die hij al eerder gebruikt heeft. Het is door middel van het opgeven van een regelnummer ook mogelijk om een stel gegevens (al dan niet tijdelijk) over te slaan.
RESTORE geeft eveneens de mogelijkheid om te kiezen tussen verschillende datalijsten. Elke datalijst zorgt voor woorden om uw talenkennis op te frissen. De gebruiker kan kiezen welke lijst hij wil gebruiken.

```
10 SCREEN 1 : KEY OFF
20 PRINT "     ***************      "
30 PRINT "        TALENTEST         "
40 PRINT "         *******          "
50 PRINT : PRINT "U kunt kiezen uit: " : PRINT
60 PRINT "1- NED. >> DUITS"
70 PRINT "2- DUITS >> NED."
80 PRINT "3- NED. >> FRANS"
90 PRINT "4- FRANS >> NED."
100 PRINT "5- NED. >> ENGELS"
110 PRINT "6- ENGELS >> NED."
120 PRINT "7- NED. >> ZWEEDS"
130 PRINT "8- ZWEEDS >> NED."
140 PRINT
150 INPUT "Uw keuze";K
160 IF K=1 OR K=2 THEN RESTORE 320
170 IF K=3 OR K=4 THEN RESTORE 330
180 IF K=5 OR K=6 THEN RESTORE 340
190 IF K=7 OR K=8 THEN RESTORE 350
200 CLS : T=0
```

```
210 FOR X=1 TO 4
220 READ N$,V$
230 IF INT(K/2)=K/2 THEN 270 ELSE PRINT:PRINT N$
240 INPUT "Uw vertaling: ";A$
250 IF A$=V$ THEN T=T+1 : PRINT "PRIMA!" ELSE PRINT "fout"
260 NEXT X : GOTO 310
270 PRINT:PRINT V$
280 INPUT "Uw vertaling: ";A$
290 IF A$=N$ THEN T=T+1 : PRINT "PRIMA!" ELSE PRINT "fout"
300 NEXT X
310 PRINT T ; " goede antwoorden!"
320 DATA hebben,haben,nieuw,neu,schaduw,schatten,arend,adler
330 DATA brand,feu,huis,maison,gebit,denture,tafel,table
340 DATA hoog,high,boek,book,huis,house,brief,letter
350 DATA vader,far,gevaar,fara,auto,bil,diep,djup
```

Dit is een computerboek en geen talenboek , dus is de gegevenslijst zeer kort. U kunt hem zelf zo lang maken als u wilt. Als u net als in het voorbeeldprogramma meer dan één taal in de gegevenslijst wilt hebben staan, kunt u nog verder bekorten door telkens maar een keer een nederlands woord te plaatsen met daarachter de vertalingen in alle gewenste talen. Probeert u zelf te verzinnen hoe u dan het juiste woord uit de lijst haalt en hoe u daarna weer zorgt voor het lezen van een nederlands woord. Het is wat gepuzzel, maar u moet eruit kunnen komen.

Arrays

Met de READ en DATA opdracht is het mogelijk lange lijsten van gegevens of waarden aan de computer op te geven en deze toe te kennen aan een stel variabelen of telkens aan dezelfde variabele. Soms moet u een stel gegevens tegelijk verwerken en niet zoals in de voorbeeldprogramma's alleen even doorlopen om de juiste er uit te zoeken. Voor zo'n lange lijst met gegevens kunnen we gewoon een heleboel variabelen kiezen waaraan we de gegevens toekennen. Maar als het gaat om veel gegevens, wordt het nogal lastig om al die variabelen uit elkaar te houden en te onthouden welke variabele wat voorstelt.
Daarom kent de MSX de mogelijkheid om lijsten met genummerde variabelen te maken Zo'n lijst heet een ARRAY (=tabel). Variabelen die in een array geplaatst zijn, zijn gemakkelijk te hanteren en men kan er eenvoudig naar verwijzen. De enige voorwaarde voor het gebruik van een array is dat u de computer van te voren vertelt hoeveel geheugenruimte er gereserveerd moet worden voor de lijst. Dit doet u door middel van de opdracht:

DIM (DIMension = afmeting)

Een-dimensionale arrays.

Stel u wilt een lijst maken van 40 gegevens. U krijgt dan te maken met de volgende problemen:

- Hoe kan deze lijst onderscheiden worden van een andere soortgelijke lijst?
- Hoe kunnen gegevens van de lijst onderling onderscheiden worden?
- Hoe moet naar een bepaald gegeven in de lijst verwezen worden?
- Hoeveel geheugenruimte moet er gereserveerd worden?
- Hoe worden de gegevens in de lijst geplaatst?

De eerste vraag is het eenvoudigste te beantwoorden. Elke lijst krijgt een eigen naam. Als eerste naam kiezen we bijvoorbeeld L (van lijst).
De eenvoudigste manier om een bepaald gegeven in de lijst aan te geven is de gegevens gewoon te nummeren. Er kan dan verwezen worden naar het eerste, het tweede, het derde, het twaalfde of het X-ste gegeven van de lijst L.
De notatie hiervoor is:

L(1) - eerste gegeven in de lijst L
L(2) - tweede gegeven in de lijst L
L(X) - het X-de gegeven in de lijst L (als er niet eerder in de programmaloop een waarde aan X is toegekend, is X gelijk aan 0)

Op deze wijze is elk gegeven in elke lijst aan te geven.
De DIM opdracht wordt gebruikt om voldoende geheugenruimte te reserveren. U hoeft niet zelf uit te rekenen hoeveel dit moet zijn. U geeft achter DIM alleen tussen haakjes de lengte van de lijst weer. Bijvoorbeeld:

```
DIM L(40)
```

voor een lijst met 40 gegevens. (Omdat de computer bij 0 begint te tellen, heeft u met DIM L(40) eigenlijk ruimte voor 41 gegevens gereserveerd. Zolang u niet echt gebrek aan geheugenruimte heeft kunt u dit beter zo laten: alweer voor de leesbaarheid.)
Om de lijst met gegevens te vullen kan de LET-opdracht gebruikt worden:

```
10 DIM L(40)
20 LET L(1)=2
30 LET L(2)=4
40 LET L(3)=5
:
:
410 LET L(40)=650
```

Maar dit kost veel intiktijd. Het is juist zo eenvoudig te verwijzen naar een bepaald gegeven in de lijst. U kunt zich uit het vorige stuk vast nog wel de datalijst herinneren. Veel beter is dan ook:

```
10 DIM L(40)
20 FOR X=1 TO 40
30 READ L(X)
40 NEXT X
50 DATA 2,4,5,..,..,..,.....................,650
```

Dit bespaart enkele honderden programmaregels.

Twee-dimensionale arrays

Soms moet er niet een lange lijst met getallen verwerkt worden, maar een hele bladzijde. De getallen hebben dan (soms) niet alleen in één richting iets met elkaar te maken, maar ook in de richting haaks daarop. De getallen staan in met elkaar te maken hebbende kolommen en rijen. Voor de normale lijst van gegevens achter elkaar wordt een zogenaamde een-dimensionale array gebruikt. De eerste dimensie wordt voorgesteld als een lijn. Een dergelijke een-dimensionale array wordt ook wel een vector genoemd.
Voor een bladzijde met getallen wordt een twee-dimensionale array gebruikt. De tweede dimensie wordt voorgesteld als een plat vlak. Een andere naam voor zo'n array is een tabel. In een twee-dimensionale array staan de gegevens in rijen en kolommen gerangschikt.
Een twee-dimensionale array met 25 gegevens ziet er bijvoorbeeld zo uit:

2	4	6	8	10
12	14	16	18	20
22	24	1	3	5
7	9	11	13	15
17	19	21	23	25

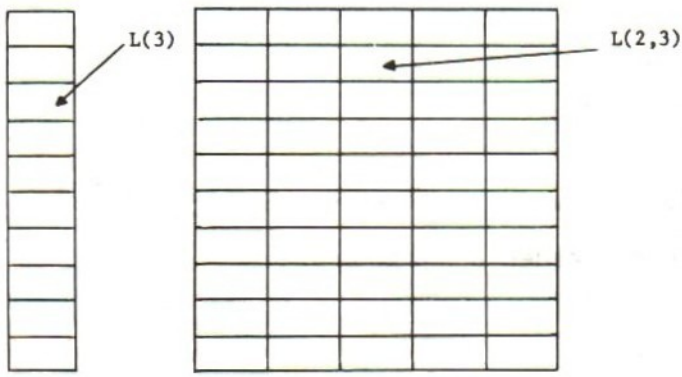
Het verwijzen naar een bepaald gegeven in dit array is iets

lastiger. U noemt nu niet meer het zoveelste gegeven in de lijst, maar u verwijst naar de zoveelste regel, zoveelste kolom:

L(3,2) is het tweede gegeven van de derde regel (24)
L(2,4) is het vierde gegeven van de tweede regel (18)
L(X,Y) is het Y-de gegeven van de X-de regel, of regel X, kolom Y

Het reserveren van voldoende ruimte in het geheugen voor een twee-dimensionale array gaat met twee getallen:

```
DIM L(5,5)
```



een- en tweedimensionale array

Het is niet nodig dat het array vierkant is. Als u bijvoorbeeld 100 gegevens op wilt bergen in een array van 4 kolommen van elk 25 gegevens of 25 rijen van elk 4 gegevens maakt u het volgende programma:

```
10 DIM K(25,4)
20 FOR Y=1 TO 4
30 FOR X=1 TO 25
40 READ K(X,Y)
50 NEXT X : NEXT Y
60 DATA ga1, ga2, ga3, ga4, .., .., ......., ga25
70 DATA gb1, gb2, gb3, gb4, .., .., ......., gb25
80 DATA gc1, ............................., gc25
90 DATA ................................., gd25
```

Let u hierbij op de volgende punten:

- U kunt zelf kiezen of u de gegevens per kolom of per regel laat lezen. In het voorbeeld hierboven wordt per kolom gelezen.

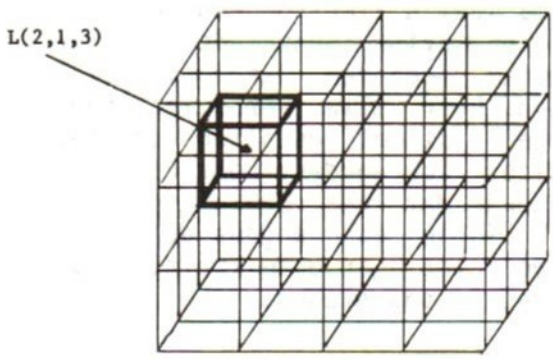
- Houdt u bij het intikken van de datalijst goed in de gaten in welke volgorde er ingelezen wordt.

- Het verhoogt de leesbaarheid en vermindert de kans op fouten als u de gegevens in de datalijst per dataregel logisch bij elkaar zet. In het voorbeeld werden alle gegevens van een kolom in een dataregel bij elkaar gezet. Natuurlijk kunnen de gegevens ook per rij ingevoerd worden.

- Een onderverdeling van de datalijst in verschillende regels is ook handig als bepaalde delen uit de lijst verschillende keren gebruikt moeten worden. De RESTORE opdracht kan alleen naar een bepaalde regel verwijzen.

Drie-dimensionale arrays

Het is ook mogelijk om drie-dimensionale arrays te maken (de derde dimensie wordt voorgesteld als ruimtelijk, bijvoorbeeld een kubus). U moet zich dit voorstellen als een heel boek met gegevens. Het eerste getal van de DIM opdracht of van de verwijzing naar een bepaald gegeven geeft de bladzijde aan. Het tweede getal geeft een regel op die bladzijde aan en het derde getal geeft een kolom aan. Voor drie-dimensionale arrays reserveert u geheugenruimte op dezelfde wijze als voor andere arrays: DIM (W,X,Y).



drie-dimensionale array

De MSX kan nog veel meer dimensies diep gaan bij het maken van arrays (tot 255 dimensies diep). Ruimtelijk is dat zeer moeilijk voor te stellen.
Probleem bij deze, maar in mindere mate ook bij alle al genoemde arrays is dat ze ontzettend veel geheugenruimte vragen. Gebruik ze selectief en maak een array niet groter dan nodig is.

String-arrays

Tot nu toe las u in dit hoofdstuk telkens 'gegevens', terwijl de voorbeelden alleen getallen lieten zien. De computer accepteert ook strings als gegevens.
Het is mogelijk een array te maken met strings als inhoud. Daarvoor moet de arraynaam afgesloten worden met een dollarteken($). String-arrays kunnen net als numerieke arrays in verschillende dimensies gemaakt worden door middel van de DIM opdracht. Stringarrays vragen echter nog meer geheugenruimte.

Werken met arrays

De variabelen in een array worden niet anders behandeld dan andere variabelen. Dat wil zeggen dat u er alle logische of rekenkundige bewerkingen en vergelijkingen op los kunt laten. Bij stringarrays zijn alle stringmanipulaties mogelijk.
Een uitgebreid voorbeeld van het werken met arrays vindt u in het hoofdstuk 'Sorteren'. Daarin worden lijsten met getallen, maar ook lijsten met strings op volgorde gezet.

De energierekening

Het volgende programma laat u zien hoe een gegevenslijst omgezet kan worden in een mooie grafiek. Gedurende een jaar is elke week de stand van de gasmeter opgenomen. Daardoor viel het wekelijkse verbruik uit te rekenen. Door alle verbruikcijfers onder elkaar te zetten zou een lange lijst met getallen ontstaan waarvan de lezer niet veel wijzer zou worden. Door de verbruikcijfers te verwerken in een grafiekje dat het verbruik per week afzet tegen de tijd (de weken), is in een oogopslag te zien hoe het energieverbruik zich door het jaar ontwikkelt. De verbruikcijfers zijn in een stel dataregels opgeslagen.

```
10 SCREEN 2 : KEY OFF
15 DRAW "bm10,0"
20 FOR Y%=0 TO 180 STEP 10
30 DRAW "m10,=y%;"
40 DRAW "nm8,=y%;"
50 NEXT Y%
60 DRAW "bm10,180"
```

```
70 FOR X%=10 TO 218 STEP 4
80 DRAW "M=x%;,180"
90 DRAW "nm=x%;,183"
100 NEXT X%
110 READ G%
120 H%=180-G%
130 DRAW "bm10,=h%;"
140 FOR A=1 TO 48
150 READ G%
160 H%=180-G%
170 R%=10+A*4
180 DRAW "m=r%;,=h%;"
190 NEXT A
200 GOTO 200
500 DATA 107
510 DATA 77,93,88,76,102,136,149,119,88,100,77,120,120,120,
    64,53,58,63,50,44
520 DATA 55,34,18,14,14,14,13,7,7,7,7,7,7,9,9,8,14,15,16,20,
    15,28,23,25,37,32,70,113
```

De regels 15-100 zorgen ervoor dat langs de verticale lijn links op het scherm een verdeling komt te staan voor de verbruikte hoeveelheid gas in een week (kubieke meters). Langs de horizontale lijn beneden op het scherm komen streepjes voor de weken te staan. Voor elke week een streepje.
De verschillende opdrachten in dit programma zijn nog niet behandeld. U kunt hiervoor de hoofdstukken over grafiek en kleur lezen. Beter is het om later als u dat deel gelezen heeft, dit programma nog eens te bekijken.
De regels 110-130 lezen het eerste gegeven uit de datalijst en plaatsen de grafische cursor (het potlood) op de lijn helemaal links, op een hoogte die bepaald wordt door het eerste gegeven (H%=180-G%). De regels 140-190 doen precies hetzelfde voor alle andere gegevens. De plaats waar de lijn van de grafiek naartoe getrokken moet worden is in de hoogte afhankelijk van een verbruikscijfers uit de datalijst, en in de breedte afhankelijk van de grootte van A (R%=10+A*4). Regel 200 zorgt ervoor dat de grafiek niet onmiddellijk na het tekenen weer verdijnt. Probeert u deze regel weg te halen en let op het effect.
De regels 500 tot en met 520 bevatten de verbruikscijfers. Voor elke week een. In dit programma lag van te voren vast dat het aantal gegevens precies voldoende zou zijn voor een jaar. Maar het kan dat u het programma regelmatig wilt veranderen, zodat u elke week een nieuwe bijgewerkte grafiek kunt zien. Daarvoor moet u A in regel 140 niet van 1 tot 51 laten lopen, maar van 1 tot Z. De waarde van Z is dan afhankelijk van het aantal data in de datalijst. Dat kunt u aangeven door als eerste gegeven in de datalijst het aantal gegevens in de datalijst op te geven.

Bijvoorbeeld door een nieuwe regel 505 te maken:

```
505 DATA 51 : REM aantal gegevens in datalijst
```

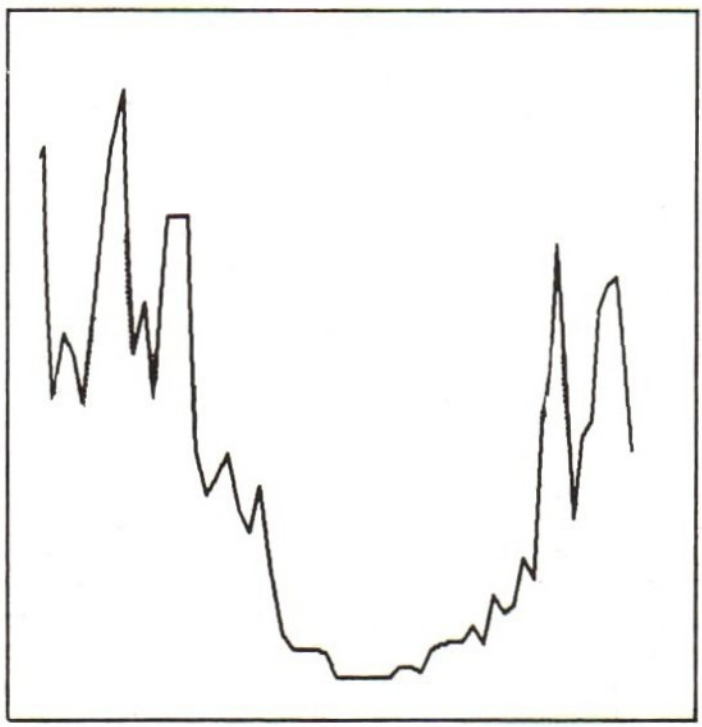
Aan Z moet nu de juiste waarde toegekend worden door een lees (READ) opdracht. Maak een regel 135:

```
135 READ Z : REM aantal gegevens in datalijst
```

Tot slot hoeft alleen regel 140 veranderd te worden:

```
140 FOR A=1 TO Z
```

U kunt in dit programma eenvoudig andere gegevens invullen. U past daarvoor de datalijst aan. Bekijk op deze wijze uw eigen energieverbruik door het jaar heen of zet hele andere gegevens in de grafiek. Als u het oorspronkelijke programma eerst SAVEt op een cassete of diskette kunt u naar hartelust experimenteren zonder angst voor het vernielen van wat tikwerk.



12. SORTEREN

Bij het sorteren van lijsten met gegevens is het onontbeerlijk om de gegevens tijdelijk op te kunnen slaan in een array. De gegevens worden daarna op volgorde in de array geplaatst. De computer maakt bij het sorteren gebruik van de nummering van variabelen in een array.
Een sorteerprogramma kan voor vele toepassingen handig zijn. U kunt er lange lijsten met getallen mee op volgorde zetten of u kunt er bestanden met adressen, cassettebandjes, diskettes, platen, computerprogramma's, etc. mee op alphabetische volgorde zetten.

Wat gebeurt er bij het sorteren van een lijst getallen?
Allereerst wordt de lijst doorzocht op zoek naar het kleinste getal. Dit getal zullen we even K noemen. In de array van getallen staat dit getal op een plaats die we even X noemen. Het kleinste getal vinden we op L(X). Het laagste getal wordt gewisseld met het eerste getal van de lijst: L(X) wordt gewisseld met L(1). Nu worden de getallen vanaf L(2) tot en met het einde L(E) doorzocht. De kleinste waarde die nu gevonden wordt, wordt in L(2) geplaatst. Daarna wordt gezocht vanaf L(3) tot en met het einde. Dit gaat net zo lang door totdat de lijst alleen nog maar wordt doorzocht vanaf het een na laatste getal L(E-1) tot en met het laatste getal L(E). Deze twee worden vergeleken en eventueel gewisseld. De lijst is dan klaar.
Voor het wisselen van twee getallen wordt bij de meeste computers een hulpvariabele gebruikt. Om te zorgen dat u deze constructie kunt herkennen in programma's die niet voor de MSX zijn geschreven volgt hier een korte uitleg:

```
LET Q = L(B)
LET L(B) = L(A)
LET L(A) = Q
```

Door deze drie regels wordt de waarde van de variabele L(B) tijdelijk opgeslagen in Q. Vervolgens wordt de waarde van de variabele L(A) gekopieerd in L(B). Daardoor is L(A) vrij (de oude waarde zit er nog wel in, maar deze is niet meer nodig). Aan de variabele L(A) wordt nu de waarde van Q toegekend. Daarin bevindt zich de oude waarde van L(B) en daardoor is de waarde van L(B) terecht gekomen in L(A) en de waarde van L(A) is terecht gekomen in L(B).
De MSX-computers doen het op precies dezelfde wijze, maar ze doen het intern. De gebruiker heeft er niets mee te maken. Het enige dat de gebruiker hoeft te doen is aan de computer opdracht te

geven twee variabelen te verwisselen:

SWAP (verwissel)

Deze opdracht verwisselt de inhoud van twee variabelen. Dit kan zowel bij arrayvariabelen als bij alle andere soorten variabelen gebruikt worden. Het is niet mogelijk de inhoud van twee ongelijksoortige variabelen te wisselen.
Om L(A) met L(B) te wisselen gebruikt u dus: SWAP L(A),L(B). De volgorde van de twee variabelen is niet belangrijk.
De hiervoor beschreven methode van het rangschikken van van gegevens lijkt lastiger dan op het eerste gezicht nodig lijkt voor zoiets 'eenvoudigs' als sorteren. De computer begint met getal L(1) en vergelijkt dat met alle andere getallen. Telkens als een getal kleiner is, worden de twee getallen omgewisseld en zoekt de computer de lijst verder af, beginnend met het volgende getal. Een programma zou er zo uit kunnen zien:

```
10 REM sorteerprogamma
20 SCREEN 0 : KEY OFF
30 READ N
40 DIM L(N)
50 PRINT : PRINT "De datalijst: "
60 PRINT
70 FOR A=1 TO N
80 READ L(A)
90 PRINT L(A);
100 NEXT A
110 FOR X=1 TO N-1
120 FOR Y=X+1 TO N
130 IF L(Y) >= L(X) THEN GOTO 150
140 SWAP L(X),L(Y)
150 NEXT Y
160 NEXT X
170 PRINT : PRINT
180 PRINT : PRINT "De gesorteerde lijst: "
190 PRINT
200 FOR A=1 TO N
210 PRINT L(A);
220 NEXT A
230 DATA 9
240 DATA 12,3,109,16,7,88,8,-1,43,65
```

In regel 30 wordt het eerste gegeven gelezen. Dit gegeven hoort niet tot de te sorteren lijst maar dient alleen om het aantal gegevens in de datalijst op te geven. De regels 70,80 en 100 lezen de gegevens in en plaatsen deze in de array L. De gegevens worden ter controle door regel 90 op het scherm afgedrukt.
Het eigenlijke sorteren gebeurt in de regels 110-160. Telkens

worden twee getallen vergeleken. Is het getal dat op een hogere plaats in de array staat groter dan het ermee vergeleken getal dan worden de twee getallen niet verwisseld. In dat geval springt de computer naar regel 150 waar de eerste van twee NEXT-opdrachten ervoor zorgt dat het volgende getallenpaar genomen wordt.
Het resultaat wordt afgedrukt door de regels 170-220. In regel 230 staat het gegeven dat de lengte van de datalijst aangeeft. In regel 240 staat de datalijst.
Wilt u stap voor stap zien hoe de computer de getallen telkens verwisselt dan kunt u de volgende regels toevoegen:

```
122 FOR A=1 TO N
124 LOCATE 20,A*2+5
126 PRINT L(A);
128 FOR Z=1 TO 50 : NEXT Z
129 NEXT A
```

Zelfs met een zeer kleine vertraging (regel 128) om het wisselen zichtbaar te maken, kost het veel tijd voordat zo'n korte lijst met gegevens geordend is.
Het bovenstaande programma is een leuk sorteerprogramma, en het stukje waarin het sorteren daadwerkelijk gebeurt, kan goed als subroutine in andere programma's gebruikt worden. Het programma is echter niet echt efficient. De computer moet immers elke keer opnieuw de hele lijst met getallen doorwerken om te kijken of de twee getallen omgewisseld moeten worden of niet. Er bestaat een veel snellere wijze van sorteren. Dit wordt de 'bubble-sort' genoemd. Bij deze manier van sorteren worden telkens de getallen die direkt naast elkaar staan (of bij een lijst met onder elkaar staande getallen, de onder elkaar staande getallen) vergeleken. De getallen worden gewisseld als de volgorde van grootte van de twee verkeerd is. De computer vergelijkt dan het hoogste van de twee getallen met het getal dat daar weer naast staat, en wisselt deze eventueel om. Etc.
Bij de eerste keer dat de computer door de lijst loopt wordt L(1) vergeleken met L(2). Als L(2) kleiner is dan L(1) worden ze gewisseld. Daarna wordt L(2) waarin op dat moment het grootste van de twee getallen L(1) en L(2) zit, vergeleken met L(3). Ook nu wordt er gewisseld als L(3) kleiner is. Dit proces gaat zo door totdat L(E-1) vergeleken is met L(E) en deze eventueel gewisseld zijn.
Bij de tweede keer dat de computer door de lijst loopt worden op dezelfde manier de getallen van L(1) tot en met L(E-1) vergeleken. In L(E) zit nu immers het grootste getal van de vergelijkingen in de eerste ronde. Het proces wordt herhaald totdat uiteindelijk alleen L(1) en L(2) vergeleken worden. Op deze wijze worden de grootsten naar beneden gebracht en komen de

kleine getallen vanzelf naar boven bubbelen (vandaar de naam). Om te zorgen dat de computer niet alle keren de lijst blijft doorlopen terwijl de lijst eigenlijk al op volgorde staat, kan een extra controle ingebouwd worden. Elke keer als de computer met een nieuwe ronde begint, wordt er een geheugenplaatsje op 'klaar' gezet. Als de computer bij het doorlopen van de lijst een keer wisselt wordt deze geheugenplaats op 'niet-klaar' gezet. Voordat de computer met een nieuwe ronde door de lijst begint, kijkt hij bij deze geheugenplaats of deze op 'klaar' of 'niet-klaar' staat. Staat de geheugenplaats op 'klaar' dan is er in de vorige ronde niets gewisseld en staan alle getallen op volgorde. Staat de geheugenplaats op 'niet-klaar' dan is er in de vorige ronde nog minstens een wisseling gemaakt en moet er nog een ronde komen. Het programma ziet er zo uit:

```
10 REM bubblesort
20 SCREEN 0 : KEY OFF
30 READ N
40 DIM L(N)
50 PRINT : PRINT "De datalijst: "
60 PRINT
70 FOR A=1 TO N
80 READ L(A)
90 PRINT L(A);
100 NEXT A
110 REM het sorteren
120 LET E=N-1
130 LET K$="niet-klaar"
140 REM begin lus
150 LET K$="klaar"
160 FOR G=1 TO E
170 IF L(G)<=L(G+1) THEN GOTO 210
180 REM wisselen
190 SWAP L(G),L(G+1)
200 LET K$="niet-klaar"
210 NEXT G
220 IF K$<>"klaar" THEN GOTO 150
230 PRINT : PRINT
240 PRINT : PRINT "De gesorteerde lijst: "
250 PRINT
260 FOR A=1 TO N
270 PRINT L(A);
280 NEXT A
290 DATA 20
300 DATA 13,21,99,76,56,74,39,-1,111,-44
310 DATA 38,92,37,52,17,-1,-999,67,778,2
```

Het is natuurlijk korter om in plaats van de stringvariabele K$ met de waarde 'niet-klaar' of 'klaar' een integer variabele met

de waarden 0 of 1 te nemen. In bovenstaand programma is alleen gebruik gemaakt van de woorden om u het gebruik van een zogenaamde 'vlag' duidelijk te maken. Een vlag is een teken dat aangeeft of een bepaalde situatie aanwezig is. Alleen als de vlag op 'klaar' staat moet opgehouden worden met het doorlopen van de lijst.
Het sorteren met behulp van de bubblesort is niet altijd de snelste methode. In het hobby-gebruik is echter de meest voorkomende situatie dat enkele gegevens toegevoegd worden aan een al bestaande en gesorteerde lijst. In deze gevallen is de bubblesort de snelste methode die er is.
Behalve het sorteren van getallen kunt u ook woorden op alphabetische volgorde laten zetten. Na het voorgaande moet onderstaand programma geen onoverkomenlijke problemen meer opleveren. Lees eventueel het hoofdstuk over strings nog eens door. Om het programma voor veel doeleinden geschikt te maken, kan het niet alleen sorteren naar de naam van een bepaald programma, maar kan het ook sorteren naar cassette (alle programma's van een bepaalde cassette bij elkaar) of naar jaar (chronologische volgorde van de programma's).
In de regels 230, 330 en 430 komt u opdrachten tegen die nog niet in dit boek behandeld zijn. Dit zijn subroutines. Dit zijn stukjes programma die verschillende keren herhaald worden. Van verschillende plaatsen in het programma kan naar deze subroutines verwezen worden door middel van de opdracht GOSUB gevolgd door het regelnummer waar de subroutine begint. Aan het einde van de subroutine staat de opdracht RETURN waardoor de computer terugspringt naar de regel volgend op de regel van waar naar de subroutine gesprongen werd. Meer hierover in het hoofdstuk 'uitstapjes'.

```
10 REM allessorteerder
20 SCREEN 0 : KEY OFF
30 READ N
40 DIM N$(N):DIM C(N):DIM P(N):DIM J(N)
50 FOR A=1 TO N
60 READ N$(A):READ C(A):READ P(A):READ J(A)
70 NEXT A
80 REM keuze maken
90 PRINT "U kunt kiezen uit sorteren naar:"
100 PRINT "- PROGRAMMA-NAAM (N)"
110 PRINT "- CASSETTE-NUMMER (C)"
120 PRINT "- JAAR (J)" : PRINT
130 INPUT "Uw keuze ";A$
140 IF A$="N" OR A$="n" THEN GOTO 280
150 IF A$="C" OR A$="c" THEN GOTO 380
160 IF A$="j" OR A$="J" THEN GOTO 180
170 GOTO 130
180 E=N-1 : K=0
190 REM jaar
200 K=1
210 FOR G=1 TO E
```

```
220 IF J(G)<=J(G+1) THEN GOTO 240
230 GOSUB 710
240 NEXT G
250 E=E-1
260 IF K<>1 THEN GOTO 200
270 GOTO 480
280 E=N-1 : K=0
290 REM programmanaam
300 K=1
310 FOR G=1 TO E
320 IF N$(G)<=N$(G+1) THEN GOTO 340
330 GOSUB 710
340 NEXT G
350 E=E-1
360 IF K<>1 THEN GOTO 300
370 GOTO 480
380 E=N-1 : K=0
390 REM cassettenummer
400 K=1
410 FOR G=1 TO E
420 IF C(G)<=C(G+1) THEN GOTO 440
430 GOSUB 710
440 NEXT G
450 E=E-1
460 IF K<>1 THEN GOTO 400
470 REM afdrukken
480 CLS
490 PRINT "de gesorteerde lijst is:"
500 LOCATE 0,2 : PRINT "PROGRAMMANAAM      CAS.  NR.  JAAR"
510 FOR A=1 TO N
520 LOCATE 0,A+4 : PRINT N$(A)
530 LOCATE 20,A+4 : PRINT C(A)
540 LOCATE 25,A+4 : PRINT P(A)
550 LOCATE 30,A+4 : PRINT J(A)
560 NEXT A
570 DATA 11
580 DATA "sorteer",1,000,1984
590 DATA "flipper",10,050,1985
600 DATA "vlakken",7,125,1984
610 DATA "kubus",6,150,1984
620 DATA "tekstverw.",12,75,1985
630 DATA "raketje",3,000,1983
640 DATA "helicopter",3,050,1983
650 DATA "palindroom",1,100,1984
660 DATA "ijsschots",9,50,1985
670 DATA "gemiddelden",3,150,1983
680 DATA "testje",0,0,0
710 SWAP N$(G),N$(G+1)
720 SWAP C(G),C(G+1)
730 SWAP P(G),P(G+1)
740 SWAP J(G),J(G+1)
750 K=0
760 RETURN
```

13. BESTANDEN VERWERKEN

Het nut van bestanden

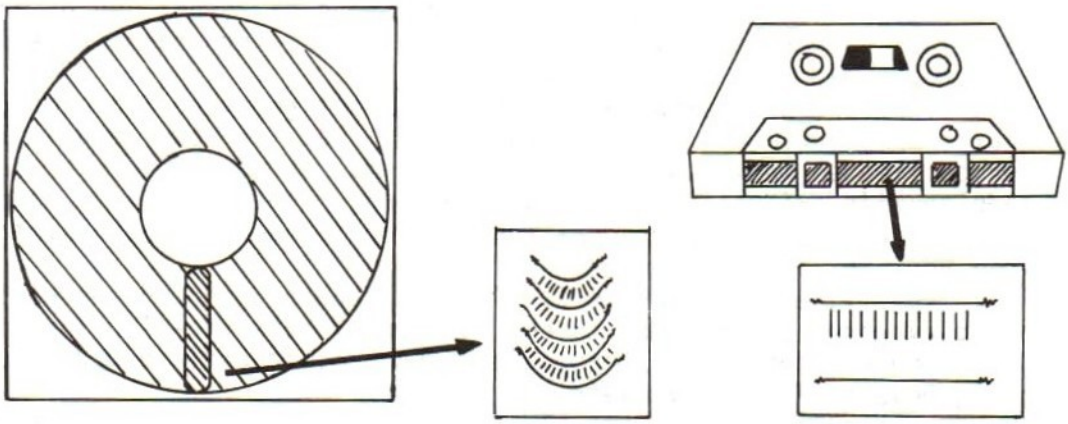
Zoals in de vorige hoofdstukken al bleek is het handig om een lijst van gegevens niet elke keer in een programma op te hoeven nemen. De meest efficiente methode is om aparte programma's en aparte lijsten met gegevens te hebben. Stel u heeft een hele serie met gegevensverwerkende programma's en een aantal lijsten met gegevens. Deze gegevenslijsten staan geheel los van de programma's. Op deze manier kunt u telkens een lijst met gegevens laten verwerken door een bepaald programma. Een paar standaard programma's zouden dan kunnen zijn; een programma om de gegevens in een grafiek weer te geven, een sorteerprogramma, een printprogramma (al dan niet met tabellen), een programma om nieuwe gegevens in te voeren en oude gegevens te verbeteren of te verwijderen, een statistisch programma, een spelprogramma of een leerprogramma.
Niet elk soort programma kan dezelfde soort gegevens gebruiken, maar er zijn wel heel veel overlappingen. Het is ook mogelijk verschillende funkties in een programma op te nemen, maar de nadelen zijn duidelijk: een lang programma, minder overzicht door een ingewikkelder programmastruktuur en minder geheugenruimte vrij voor de lijst met gegevens.
Als mogelijke gegevensverzamelingen kunnen genoemd worden; energieverbruik, programmatheek, platen of cassette-discotheek, bibliotheek, onderzoekcijfers, woordenboeken, educatieve opgaven, spelgegevens of tekengegevens.
De MSX beschikt over ruim voldoende faciliteiten voor het maken van aparte gegevensbestanden. Deze gegevensbestanden worden in het Engels 'files' genoemd. Dit woord zult u in de computerliteratuur veel tegenkomen. Hoewel een deel van de opdrachten die de MSX gebruikt voor het werken met een bestand onafhankelijk zijn van het opslagmedium (cassette of diskette) zijn er wel enkele verschillen bij het openen en sluiten van een bestand.

Het verschil tussen cassette- en diskbestanden

Voor een goed begrip van het werken met bestanden is het handig iets af te weten van de manier waarop bestanden worden opgeslagen op cassette of diskette.
Elke verzameling van bij elkaar horende gegevens wordt een bestand genoemd. De bestanden worden als magnetische signalen vastgelegd op magnetische dragers. In de hobbysfeer zullen dit

voornamelijk cassettes of diskettes (kleine, soepele schijven al dan niet in een harde kunststof hoes gedaan) zijn. Het grote verschil tussen een diskette en een cassette is dat de laatste een lange lijst van gegevens bevat, die gelezen moet worden door bij het begin van de lijst te beginnen en bij het einde van de lijst te eindigen. Dit is een gevolg van de wijze waarop signalen op een cassette worden gezet. Het bandje spoelt immers langzaam langs de koppen van de recorder en tijdens dit spoelen worden signalen op de cassette gezet. Bij het aflezen moet het bandje naar het begin worden terug gespoeld en gaat het aflezen ook weer van begin tot einde.

Een disk bestaat - eenvoudig voorgesteld - uit een heleboel cassettebandjes binnen elkaar, waarbij de bandjes in een cirkel zijn gelegd. Bij het 'einde' van zo'n cirkelvormig bandje (een track=spoor) gekomen is onmiddellijk weer 'het begin'



schematische weergave van een cassettebandje en een diskette

De kop van een diskdrive (de machine waarmee disk gelezen en beschreven kunnen worden) kan zeer snel van het ene spoor naar het andere spoor springen. Door de cirkelvorm en de mogelijkheid om van het ene spoor naar het andere spoor te springen kan bij gebruik van disks direkt een bepaald gegeven uit een bepaald bestand gelezen worden. Het bestand hoeft niet van begin tot eind doorgelezen te worden. De diskette bevat naast de gegevens altijd een inhoudsopgave van de bestanden, waardoor de computer weet waar de kop naartoe moet springen.
Doordat een disk ook nog vele malen sneller draait dan een cassette spoelt, is de snelheid waarmee gewerkt kan worden bij disks vele malen hoger dan bij cassettes.

Het grote nadeel van disks en diskdrives is echter dat zij een stuk duurder zijn. Zolang snelheid niet een eerste vereiste is bij het gebruik van bestanden is een cassetterecorder een goede en goedkope oplossing.
In dit boek vindt u uitgelegd hoe u met cassettebestanden moet werken. Waar nodig zullen de afwijkingen voor een diskbestand kort uitgelegd worden.

Het maken van een bestand

Een bestand is te vergelijken met de lijsten van gegevens zoals die tot hiertoe telkens in DATA-opdrachten werden gezet. Er zijn echter twee grote verschillen:

- Gegevens in DATA-opdrachten kunnen alleen gebruikt worden in het programma waarin de gegevens staan. Gegevens uit een bestand kunnen door verschillende programma's gebruikt worden.
- Gegevens in DATA-opdrachten moeten door de gebruiker ingetikt worden, terwijl de gegevens in bestanden door de computer gemaakt kunnen zijn.

Er zijn veel methoden waarop het gebruik van bestanden aanschouwelijk gemaakt kan worden. Ongeacht de voorstellingswijze dienen de volgende punten in gedachte te worden gehouden:

- elk bestand heeft een eigen naam.
- een bestand bestaat uit een aantal geordende gegevens, waarbij elk gegeven een vaste plaats heeft binnen dat bestand.
- bestanden moeten voor gebruik worden geopend en na gebruik (dit is heel belangrijk!) weer netjes gesloten!
- gegevens worden een voor een in een bestand gezet of uit het bestand gelezen (eigenlijk gekopieerd, want het origineel blijft in het bestand staan).

Het onderstaande programma geeft de gebruiker de mogelijkheid om telkens een getal in te tikken. De computer schrijft het weg naar een cassette.

```
10 CLS
20 PRINT "Doe cassette in recorder"
30 PRINT "spoel naar juiste positie"
40 PRINT "zet op OPNEMEN"
50 PRINT "druk RETURN als klaar"
60 INPUT X
70 PRINT : PRINT "Moment a.u.b."
80 OPEN"cas:xxx" FOR OUTPUT AS #1
90 PRINT : PRINT
100 INPUT "Geef GETAL:";GE
110 PRINT #1,GE
120 PRINT
```

```
130 INPUT"Klaar? ( j/n)";A$
140 IF A$<>"j" THEN PRINT : GOTO 100
150 PRINT : PRINT "Moment a.u.b."
160 CLOSE#1
170 END
```

De eerste zeven programmaregels zijn bedoeld om de gebruiker van het programma te laten weten wat er allemaal moet gebeuren. Er moet een leeg stuk cassette klaar staan in de cassetterecorder en de cassetterecoder moet op opnemen gezet worden. Let hierbij op! Als u niet goed oplet kunt u per ongeluk een bestand wegvegen door er een nieuw bestand overheen op te nemen. De MSX kan niet eerst kijken of er al wat op de band staat.
In regel 80 wordt een bestand met de naam xxxx geopend. Daarvoor gebruikt u de opdracht

OPEN "CAS:....." FOR OUTPUT AS #..

Met deze opdracht geeft u aan dat u een bestand wilt openen om er gegevens in op te slaan. Achter het woord OPEN komt tussen aanhalingstekens eerst te staan wat voor soort opslagmedium gebruikt wordt om het bestand op weg te schrijven. U kunt daarvoor kiezen uit:

CAS:	cassetterecorder
CRT:	tekstscherm
GRP:	grafisch scherm
LPT:	printer

Naast deze voor elke MSX-computer geschikte beschrijvingen van het opslagmedium is het mogelijk andere beschrijvingen te gebruiken als u ROM uitbreidingen heeft. Lees daarvoor de handleiding bij de betrokken uitbreiding.
U merkt dat de diskdrive er niet tussen staat. Voor het wegschrijven van een bestand naar diskette gebruikt u:

OPEN "A:..." FOR OUTPUT AS #..

Op het eerste stel stippeltjes zet u zowel bij een cassette als een diskettesysteem de naam van het bestand. U kunt daar zelf een naam van maximaal zes lettertekens voor kiezen. Het eerste letterteken moet een letter uit het alphabet zijn.
De mogelijkheden CRT: en LPT: zijn niet echt zinvol. Deze gebruikt u om tekst naar een tekstscherm of naar een printer te sturen. Voor beide toepassingen bestaan gemakkelijker opdrachten (PRINT en LPRINT). Omdat het niet mogelijk is om zomaar tekst op het grafische scherm te zetten is GRP: wel belangrijk. Hoe u daarmee tekst op het grafische scherm zet leest u in de hoofdstukken over grafiek.

Na het aangeven van het opslagmedium en de naam van het te openen bestand (dit alles als een string tussen aanhalingstekens) geeft u aan of het bestand geopend wordt om uit te lezen of om gegevens in weg te schrijven. In het bovenstaande programmavoorbeeld werd een bestand geopend om gegevens in op te slaan. Het was immers een geheel nieuw bestand dat we maakten. De keuze is tussen:

FOR OUTPUT	voor uitvoer
FOR INPUT	voor invoer
FOR APPEND	voor toevoegen

Van de laatste mogelijkheid zult u niet snel gebruik hoeven te maken. De mogelijkheid om gegevens uit een bestand te lezen en om gegevens in een bestand weg te schrijven zijn voldoende. Ook voor het aanvullen van een al bestaand bestand. De laatste mogelijkheid werkt ook niet bij een cassettesysteem.
Na het aangeven van het gebruik van de link tussen de computer en en opslagmedium moet aangegeven worden onder welk nummer we in het programma naar het bestand willen verwijzen:

```
AS #..
```

Het teken # is het amerikaanse symbool voor nummer. Achter # vult u een getal tussen 0 en 15 in. Aan het gebruik van de nummers 0 en de nummers 2 tot en met 15 zitten enkele haken en ogen.
De MSX gaat ervan uit dat u slechts een bestand tegelijk wilt gebruiken. Zeker als u een cassettesysteem heeft is dit ook de normale gang van zaken. Als u meer dan een bestand tegelijk open wilt hebben, zult u immers meer dan een cassette voor het wegschrijven in het bestand tegelijk klaar moeten hebben staan. Dat betekent een veelvuldig wisselen van cassettes. Bij een disksysteem is het door de grotere flexibiliteit veel eenvoudiger meer dan een bestand tegelijk te openen. De computer zoekt samen met de diskdrive wel voor u uit waar welk bestand staat en waar de gegevens neergezet moeten worden. Deze flexibiliteit kent de cassette door de methode van doorvoeren langs de recorderkoppen niet.
Zolang u slechts een bestand tegelijk gebruikt, hoeft u de computer niet op te geven dat u er meer tegelijk wilt gebruiken maar kunt u ook alleen nummer 1 als bestandsnummer gebruiken: AS #1. Wilt u meer bestanden tegelijkertijd gebruiken dan moet u dit eerst opgeven door de opdracht

```
MAXFILES =..
```

Op de stippen kunt u een getal tussen de 0 en 15 invullen. De MSX kiest zelf altijd 1. U gebruikt deze opdracht dus eigenlijk alleen als u een disksysteem heeft.

Door MAXFILES 0 te kiezen beperkt u de mogelijkheden van het maken en lezen van een bestand aanzienlijk. Nadat MAXFILES op 0 gezet is kunt u alleen nog maar CSAVE, CLOAD, CLOAD?, SAVE en LOAD gebruiken, dat wil zeggen dat u programma's kunt opslaan en lezen maar geen gegevensbestanden meer kunt maken. Het vergroten van MAXFILES kost extra geheugenruimte.

Na het openen van het bestand kunnen we gegevens in het bestand gaan zetten. Regel 100 van bovenstaand programma vraagt de gebruiker een getal in te voeren.
Regel 110 zet dat getal in het bestand. Daarvoor wordt dezelfde PRINT-opdracht gebruikt die u al vele malen in dit boek tegenkwam. In dit geval moet het getal echter niet op het beeldscherm gezet worden (PRINT) of op papier gezet worden (LPRINT), maar in een bestand gezet worden:

```
PRINT #1,..
```

Achter PRINT geeft u met # en een nummer aan in welk bestand het gegeven dat achter de komma staat, geplaatst moet worden. Merkt u op dat het nummer achter PRINT # gelijk moet zijn aan het nummer dat u aan het bestand heeft gegeven in de OPEN opdracht. Bij cassettesystemen zal dit altijd nummer 1 zijn. Bij een disksysteem kunt u verschillende bestanden tegelijk open hebben, en kunt u door het nummer achter PRINT # bepalen naar welk bestand een bepaald gegeven gaat.
U kunt in een PRINT # opdracht verschillende gegevens weg laten schrijven. Elk gegeven moet gescheiden zijn van het vorige door een komma. U kunt zowel de inhoud van numerieke variabelen als stringvariabelen wegschrijven. Daarnaast is het mogelijk direkt bepaalde gegevens weg te schrijven.

```
PRINT #1, GE                          numerieke variabele
PRINT #1, A$                          stringvariabele
PRINT #1, GE, A$                      combinatie
PRINT #1, "string", 1234              string en getal
PRINT #1, "string", 1234, A$, GE      combinatie
```

U ziet dat op alle mogelijke manieren gegevens weg geschreven kunnen worden. Het enige waar u op hoeft te letten is dat bij het lezen van de gegevens uit het bestand, de juiste gegevens aan de juiste variabelen toegekend worden. Het is onmogelijk een string toe te kennen aan een numerieke variabele! Zie hiervoor ook de volgende paragraaf over invoer.
Door de lus in regel 140 wordt de invoer van de gegevens telkens herhaald totdat de gebruiker op de vraag in regel 130 'j' antwoordt.
In regel 160 volgt een bijzonder belangrijke opdracht. Nadat de

gebruiker klaar is met een bestand dient dit gesloten te worden. Alleen op die manier vormt de in het bestand gevatte informatie een afgesloten geheel. Dit is nog veel belangrijker bij het werken met diskettes!

```
CLOSE #1
```

Let er op dat het nummer van het te sluiten bestand overeenkomt met het nummer van het bestand dat eerder geopend is. Bij cassettesystemen zal dit altijd 1 zijn.
Heeft uw cassetterecorder afstandbediening en is die op de juiste wijze aangesloten dan zal de computer regelen dat de cassettemotor op de juiste momenten gaat draaien. Vergeet u niet eerst de opdracht

```
MOTOR ON
```

te geven, zodat de MSX weet dat er op een cassettemotor gelet moet worden. Om het effect weer uit te zetten (bijvoorbeeld voor versneld doorspoelen) gebruikt u

```
MOTOR OFF
```

Het lezen van een bestand

Om de gegevens die u door het intikken van allerlei getallen in het bestand gezet hebt weer tevoorschijn te halen gebruikt u een kort programma dat veel op het vorige programma lijkt:

```
10 CLS
20 PRINT "Doe cassette in recorder"
30 PRINT "spoel naar juiste positie"
40 PRINT "zet op AFSPELEN"
50 PRINT "druk RETURN als klaar"
60 INPUT X
70 PRINT : PRINT "Moment a.u.b."
80 OPEN"cas:xxx" FOR INPUT AS #1
90 PRINT : PRINT
100 INPUT #1,GE
110 PRINT GE
120 PRINT
130 GOTO 100
160 CLOSE#1
170 END
```

Als u dit programma runt moet u als daarom gevraagd wordt, de cassette waarop de gegevens staan die u met het vorige programma ingetikt heeft, in de cassetterecorder plaatsten en naar de juiste positie spoelen. Zet de cassetterecorder op afspelen en druk op RETURN.
In regel 80 wordt het bestand weer geopend:

```
OPEN "cas:xxx" FOR INPUT AS #1
```

Achter de aanduiding van het aparaat waar de gegevens vandaan komen (cas:) staat de naam van het bestand dat gelezen moet worden. Bij cassettesystemen kunt u de naam weglaten bij het lezen of schrijven. Laat u de naam weg bij het lezen dan wordt het eerste bestand dat de computer op de band tegenkomt gelezen. Laat u een naam weg bij het schrijven dan wordt het bestand naamloos op cassette gezet. Bij disksystemen moet altijd een naam staan.
Heeft u in dit programma dezelfde naam gebruikt als bij het wegschrijven in het vorige programma, of heeft u in dit programma geen naam opgegeven, dan zal de computer eerst een tijdje alleen 'Moment a.u.b.' laten zien, en daarna de getallen die u bij het eerste programma intikte laten zien.
Om de gegevens vanaf een bestand in te lezen gebruikt u de opdracht:

```
INPUT #..
```

Deze opdracht is vergelijkbaar met de gewone INPUT opdracht. Achter INPUT # zet u het nummer dat u aan het bestand toegekend heeft door de OPEN opdracht. Bij cassettesystemen een 1. Vervolgens plaatst u de te lezen gegevens achter de opdracht gescheiden door komma's. Let er bij het opgegeven van de te lezen gegevens op dat een string alleen toegekend kan worden aan een stringvariabele.
Als u nog even kijkt naar de voorbeelden die we gaven bij de PRINT # opdracht, passen daarbij de volgende INPUT # opdrachten:

```
INPUT #1, GE                    Numeriek gegeven
INPUT #1, A$                    String gegeven
INPUT #1, GE, A$                Combinatie
INPUT #1, X$, Y                 String en numeriek gegeven
INPUT #1, X$, Y, A$, GE         Combinatie
```

Let bij het wegschrijven en weer teruglezen van gegevens op de volgorde. De computer verandert de volgorde nooit. U dient de gegevens terug te lezen in de volgorde waarin ze weggeschreven zijn, anders kunt u moeilijkheden krijgen met de toekenning van de variabelen.

Achter de INPUT # opdracht hoeft in tegenstelling tot de gewone INPUT opdracht geen RETURN te volgen.
Regel 110 van het programma drukt het net uit het bestand gehaalde gegeven af. Spaties die aan het begin van een numeriek gegeven staan worden verwaarloosd. De aanhalingstekens van de strings worden ook niet afgedrukt.
Regel 130 zorgt voor een lus. Daardoor blijft de computer telkens gegevens uit het bestand lezen. De opdracht CLOSE #1 wordt daardoor nooit uitgevoerd. Als de computer geen gegevens meer in het bestand kan vinden terwijl er toch gegevens gelezen moeten worden geeft hij als foutmelding:

```
Input past end in ....      (invoer voorbij het einde in ...)
```

Het programma is dus niet goed. Er bestaat een speciale opdracht om te kijken of het einde van het bestand al bereikt is. Door deze opdracht in de lus op te nemen kunnen we de computer opdracht geven om het kanaal naar het bestand weer te sluiten aan het einde van het bestand en te stoppen met het programma.

```
EOF              End Of File          (einde van het bestand)
```

Op de stippels plaatst u het nummer dat aan het bestand is toegekend door de OPEN opdracht.
EOF is -1 als het einde van het bestand is bereikt en 0 zolang dat niet zo is. De eenvoudigste manier om bovenstaande foutmelding te voorkomen is door regel 130 in het programma te vervangen:

```
130 IF EOF(1)=0 THEN GOTO 100
```

De waarden -1 en 0 bij EOF zijn niet toevallig. De computer noteert iets als waar door het de waarde -1 te geven en noteert iets als niet-waar door het de waarde 0 te geven. Het is-gelijk teken en de nul zijn in regel 130 daardoor niet nodig. Er moet getest worden of EOF niet waar is. Dat is gelijk aan NIET waar:

```
130 IF NOT EOF(1)=1 THEN GOTO 100
```

Omdat EOF in de computer opgeslagen is als -1 of 0, en het testen van de waarheid van en IF..THEN.. bewering ook met 0 en -1 gaat, is het onnodig om =1 achter EOF te zetten. Omdat GOTO in een IF..THEN bewering ook niet nodig is, kan de regel bekort worden tot:

```
130 IF NOT EOF(1) THEN 100
```

Wilt u alleen na het bereiken van EOF het kanaal sluiten dan kunt

u gebruiken:

```
IF EOF(1) THEN CLOSE#1
```

Om dit programma te kunnen gebruiken was het nodig dat u de cassette terug draaide naar het punt waar het bestand begint. Om vergissingen te voorkomen is het raadzaam daarvoor altijd een 'ronde' tellerstand op de cassetterecorder te nemen. Bijvoorbeeld:

```
000 = bibliotheek boeken
100 = bibliotheek platen
200 = bibliotheek programma's
300 = kaart
etc.
```

Heeft u de cassetterecorder op de juiste tellerstand gezet dan geeft u de opdracht RUN. Voor de zekerheid geeft het programma u alsnog de gelegenheid de juiste tellerstand te vinden. Daarna moet de cassetterecorder op afspelen gezet worden, en op RETURN gedrukt worden. Een cassetterecorder met afstandsbediening zal nu door de computer gestuurd draaien en stilstaan en daarbij de gegevens lezen en weergeven op het scherm. Een gewone cassetterecorder zal gewoon blijven draaien en telkens een stukje bestand tegenkomen en dat lezen en afdrukken (tot EOF).
Hier blijkt het nadeel van casseterecorders zonder afstandsbediening. Bij het opnemen van de gegevens in een bestand blijft de cassetterecorder draaien terwijl de computer gegevens verzameld tot een 'blok' gegevens vol is. Telkens als een blok vol is, zet de computer dit weg op de cassette. Tussen de verschillende blokken kunnen bij het langzaam met de hand invoeren van gegevens (zoals in het eerste voorbeeldprogramma) grote stukken leegte tussen de blokken ontstaan. Bij het afdraaien van de cassette om het bestand te lezen zorgt dit voor veel oponthoud. Een cassetterecorder met afstandsbediening heeft dit bezwaar niet omdat de computer de recorder na elk blok stilzet totdat het volgende blok compleet is. Het nadeel is te verhelpen door een andere cassetterecorder of een disksysteem te nemen. Maar u kunt ook eerst alle gegevens in een datalijst zetten en deze snel achter elkaar naar het bestand wegschrijven. Daardoor wordt het onderscheid tussen een bestand en een datalijst in een programma wel kleiner, maar het voordeel van een bestand dat een lijst met gegevens slechts een keer ingevoerd hoeft te worden om voor veel doeleinden en programma's geschikt te zijn, blijft.
Onderstaand programma geeft een voorbeeld hoe u gegevens eerst in een datalijst zet en deze lijst daarna door de computer in een dicht beschreven bestand laat zetten. De gebruikte gegevens zijn

tekencoordinaten voor aardrijkskundig programma. Het eerste gegeven in de aparte DATA opdracht geeft alleen het aantal te gebruiken gegevens aan.

```
10 CLS
20 PRINT "Doe cassette in recorder"
30 PRINT "spoel naar juiste positie"
40 PRINT "zet op OPNEMEN"
50 PRINT "druk RETURN als klaar"
60 INPUT X
70 PRINT "Moment a.u.b."
80 OPEN "cas:kaart" FOR OUTPUT AS #1
90 READ A
100 FOR C=1 TO A
110 READ X,Y
120 PRINT #1,X,Y
130 NEXT C
140 CLOSE #1
200 DATA 260
210 DATA 75,37,70,63,64,83,57,97,53,103,49,104,50,106,52,107,54,110,59,114
220 DATA 59,114,62,115,65,116,68,115,72,117,72,119,67,119,63,123,56,123,56,127,59,136
230 DATA 56,137,50,135,44,126,40,125,32,127,26,130,26,134,31,135,32,137,35,137
240 DATA 38,141,42,141,45,139,46,138,49,138,59,141,62,144,66,144,67,143,66,137
250 DATA 70,134,70,139,74,138,76,135,81,135,82,136,79,139,80,141,87,139,89,134
260 DATA 92,133,94,135,93,143,100,148,108,148,112,147,114,152,123,152,124,155,120,163
270 DATA 120,171,117,177,117,179,120,181,120,183,123,183,126,182,131,183,132,181,132,177
280 DATA 136,175,136,171,132,167,130,167,128,161,132,161,132,157,135,153,134,151,132,151
290 DATA 132,147,139,139,139,131,134,122,128,115,125,114,125,110,135,109,135,107,140,106
300 DATA 147,111,158,103,160,104(163,97,161,94,156,95,156,92,160,91,171,83,172,77
310 DATA 170,71,168,67,164,68,160,67,155,62,157,59,156,56,158,53,167,52,168,39
320 DATA 175,31,176,19,177,17,176,15,168,13,167,11,163,10,161,3,158,1,146,2
330 DATA 138,8,136,5,128,6,114,11,107,10,101,23,101,27,104,31,100,41,100,43
340 DATA 110,44,116,42,117,44,112,47,111,51,114,54,116,55,126,53,128,54,129,56
350 DATA 126,59,125,67,122,71,116,76,115,81,110,83,106,83,96,80,92,77,86,76
360 DATA 91,73,93,69,93,65,94,63,94,64,93,65,90,63,89,57,91,56,94,57
370 DATA 96,54,99,53,100,51,94,49,91,51,90,49,91,41,88,37,85,38,84,39
380 DATA 83,41,81,40,78,37,78,35,76,35,75,37,0,0,64,119,62,121,57,119
390 DATA 54,120,50,117,50,114,48,113,44,114,43,112,44,111,49,111,51,113,64,119
400 DATA 0,0,51,122,50,124,48,123,45,121,41,120,38,121,36,118,42,115,50,119
410 DATA 51,122,0,0,54,125,55,129,54,132,49,129,48,127,54,125,0,0,58,144
420 DATA 44,155,40,156,3q,151,34,149,26,153,22,151,20,146,23,141,30,141,40,145
430 DATA 46,143,47,141,50,140,58,144,0,0,96,73,95,77,102,78,104,81,110,81
440 DATA 111,76,121,69,124,65,123,59,120,59,115,57,113,58,96,73,0,0,76,27
450 DATA 75,33,77,34,80,31,82,30,83,25,82,23,81,22,76,27,0,0,88,11
460 DATA 83,16,82,21,85,19,86,17,90,13,88,11,0,0,111,2,96,5,92,8
470 DATA 92,11,95,12,97,9,100,8,102,6,110,4,111,2
```

In regel 80 wordt een bestand geopend en een kanaal gekozen (#1). Regel 90 leest hoe lang de lijst is (DATA in regel 200). Regel 100 zorgt voor een herhalingslus. Regel 110 leest telkens een X-coordinaat en een Y-coordinaat. Regel 120 zet deze coordinaten weg in het bestand. Het is niet noodzakelijk dat u telkens beide gegevens door een enkele PRINT # opdracht laat wegzetten. Als u dat overzichtelijker vindt kunt u ook gebruik maken van:

```
PRINT #1, X
PRINT #1, Y
```

Op deze wijze worden de gegevens in precies dezelfde volgorde weggeschreven.

De kaart van Nederland

Om uit deze berg gegevens iets aardigs te halen moet u het programma dat gegevens leest ietsje uitbreiden. Het is niet de bedoeling dat de gegevens alleen als getallen op uw beeldscherm verschijnen. Omdat het een lijst met tekencoordinaten is, moet het leesgedeelte voorzien worden van een tekengedeelte. De werking van de verschillende tekenopdrachten komt later in dit boek aan de orde. U kunt dit programma nu wel al gebruiken voor een fraaie kaart van Nederland:

```
10 CLS
20 PRINT "Doe cassette in recorder"
30 PRINT "spoel naar juiste positie"
40 PRINT "zet op AFSPELEN"
50 PRINT "druk RETURN als klaar"
60 INPUT Z
70 SCREEN 2
80 COLOR 9,15,4
90 CLS
100 LINE-(75,37),15
110 OPEN "cas:kaart" FOR INPUT AS#1
120 INPUT #1,X,Y
130 IF X>0 AND Y>0 THEN LINE-(X,Y),9 ELSE INPUT#1,X,Y:LINE-(X,Y),15
140 IF EOF(1) THEN GOTO 160
150 GOTO 110
160 CLOSE#1
170 GOTO 170
```

De regels 10-60 kent u uit de vorige programma's.
Regel 60 zorgt ervoor dat de computer wacht totdat de gebruiker door het indrukken van RETURN aangeeft dat de cassetterecorder loopt.
Regel 70 stelt het grafische scherm in waardoor tekeningen mogelijk zijn.
Regel 80 stelt de kleuren in. Rode lijnen, een witte achtergrond en een blauwe rand. Door het wissen van het scherm in regel 90 worden deze kleuropdrachten effectief.
Regel 100 trekt een witte (en dus niet zichtbare) lijn naar het beginpunt van de tekening van Nederland.
Regel 110 opent een kanaal om een bestand 'kaart' op de cassetterecorder te kunnen gebruiken voor invoer met als nummer 1.
Regel 120 leest telkens opnieuw twee gegevens. Dat zijn de X- en de Y-coordinaat van het volgende punt waar een lijn naartoe getrokken moet worden.
Regel 130 bekijkt of naar een punt een lijn getrokken moet worden in rood, of dat er een eiland begint, zodat de lijn in wit (=onzichtbaar) getrokken moet worden. Het begin van een eiland wordt telkens weergegeven doordat zowel de X- als de Y-coordinaat 0 is. Twee nullen achter elkaar betekent dus: sla deze coordinaten over, lees nieuwe coordinaten maar trek de lijn nu in wit en niet in rood.
Regel 140 laat de computer naar regel 160 springen als het einde van het bestand bereikt is.
Regel 150 zorgt voor een herhalingslus.
Regel 160 sluit het kanaal.
Regel 170 zorgt ervoor dat het programma niet eindigt. Als het programma klaar is keert de MSX altijd terug naar het tekstscherm 0. Als dat gebeurt verdwijnt de tekening. Zolang het programma blijft lopen is de tekening zichtbaar. Om het programma te stoppen gebruikt u CTRL en STOP.

Het tekenen van de kaart ging niet echt snel. De beperkende factor is de snelheid waarmee bestanden op een cassette verwerkt kunnen worden. U kunt een snellere weergave bereiken door niet telkens een stukje kaart te laten tekenen, maar eerst alle gegevens weer op te slaan in een array. Zodra het hele bestand gelezen is, geeft de computer de kaart weer. Er is hierbij geen sprake van echte versnelling, maar slechts van het rekken van het geduld van de gebruiker. In onderstaand programma is gebruik gemaakt van een tekstje en wat geluid om de gebruiker te laten wachten. Als u de hoofdstukken over grafiek en geluid gelezen heeft kunt u een kleine tekening en wat muziek gebruiken tijdens het wachten.

Het volgende wat grotere programma kent nog een verbetering. Naast de omtrekken van Nederland zijn ook enkele grote plaatsen ingetekend. Gebruik het volgende korte programma om alle gegevens van de steden in een apart bestand te zetten. Plaats het bestand direkt achter het bestand met coordinaatgegevens van de kaart.

```
10 CLS
20 PRINT "cassette op OPNEMEN"
30 PRINT "RETURN als klaar"
40 INPUT Z
50 OPEN "cas:stad" FOR OUTPUT AS #1
60 READ A
70 FOR C=1 TO A
80 READ X,Y,ST$
90 PRINT #1,X,Y,ST$
100 PRINT X;",";Y,ST$
110 NEXT C
120 CLOSE #1
200 DATA 19
210 DATA 119,174,"Maast",84,76,"Amste",72,74,"Haarl",62,90,"Den H"
220 DATA 68,105,"Rotte ",78,124,"Breda",101,116,"Den B",98,90,"Utrec"
230 DATA 128,99,"Arnhe",158,79,"Henge",117,22,"Leeuw",147,15,"Groni"
240 DATA 106,64,"Lelys",150,34,"Assen",68,85,"Leide",77,36,"Den H"
250 DATA 122,111,"Nijme",109,135,"Eindh",130,172,"Heerl"
```

Het grootste deel van dit programma kent u al. Nieuw is dat in regel 80 geen twee getallen gelezen worden, maar twee getallen en een string. Deze worden alledrie in het bestand gezet in regel 90. Daarbij worden ter controle de getallen en de string op het scherm afgedrukt.
Het volgende programma maakt geen gebruik van de strings. Deze zijn toegevoegd om u de mogelijkheid te bieden een combinatie van tekening en vraag en antwoordspel te maken. Probeert u of het mogelijk is om de gebruiker telkens de kaart te laten zien met een stad ingetekend. De gebruiker moet dan de naam van de stad invoeren. De eerste vijf karakters van het antwoord moeten vergeleken worden met de string die hoort bij de coordinaten van de getekende stad.
Het onderstaande programma biedt u het tekengedeelte.

```
10 CLS
20 DIM X(300)
30 DIM Y(300)
40 DIM PX(20)
50 DIM PY(20)
60 DIM P$(20)
```

```
70 A=0
80 PRINT "Cassette op juiste positie"
90 PRINT "Zet op AFSPELEN"
100 PRINT "RETURN als klaar"
110 INPUT Z
120 PRINT:PRINT:PRINT"    Even de gegevens laden! "
130 PRINT:PRINT
140 OPEN "cas:kaart" FOR INPUT AS #1
150 A=A+1
160 INPUT #1,X(A),Y(A)
170 IF A/3=INT(A/3) THEN BEEP
180 IF EOF(1) THEN GOTO 200
190 GOTO 150
200 CLOSE#1
210 CLS
220 PRINT:PRINT:PRINT"     Even de steden laden!"
230 PRINT:PRINT"   ******************************"
240 OPEN "cas:stad" FOR INPUT AS #1
250 FOR A=1 TO 19
260 INPUT #1,PX(A),PY(A),P$(A)
270 BEEP
280 NEXT A
290 CLOSE #1
300 CLS
310 SCREEN 2
320 COLOR 9,15,4
330 CLS
340 LINE-(75,37),15
350 FOR A=1 TO 260
360 IF X(A)>0 AND Y(A)>0 THEN LINE-(X(A),Y(A)),9 ELSE
    PSET (X(A+1),Y(A+1)),15 : A=A+1
370 NEXT A
380 FOR A=1 TO 19
390 PSET (PX(A),PY(A))
400 LINE (PX(A),PY(A))-(PX(A)+3,PY(A)-3),9,B
410 BEEP
420 LINE (PX(A),PY(A))-(PX(A)+3,PY(A)-3),9,BF
430 NEXT A
440 GOTO 440
```

Vergelijkt u in dit programma de regels 150, 180, 190, 200 en 250, 280 en 290. De eerste serie regels zorgt voor het afsluiten van het bestand nadat het einde van het bestand geconstateerd is door de computer. De tweede serie regels sluit het bestandskanaal na een van te voren bepaald aantal gegevens af. Deze laatste methode is alleen geschikt als u precies weet hoe groot het bestand is.
Het programma gaat er vanuit dat het bestand 'stad' direkt achter

het bestand 'kaart' op de cassette staat. Wilt u de mogelijkheid hebben tussen het lezen van 'kaart' en van 'stad' van cassette te wisselen of de cassette door te spoelen, dan moet u een pauze als in de regels 80-110 inbouwen tussen regels 200 en 210.

14. UITSTAPJES

Inleiding

Tot op heden waren de in dit boek gebruikte programma's niet zo erg lang en waren er niet zo veel vertakkingen. Maar eens komt het moment dat voor een bepaald probleem een kort programma niet meer voldoet. De computer moet zoveel opdrachten krijgen dat daar lange programma's voor nodig zijn. Gaat zo'n lang programma gepaard met veel vertakkingen dan wordt het programma steeds slechter te lezen. Afgezien daarvan is ook het maken van zo'n lang programma lastig omdat de programmeur snel in de war kan raken en niet meer weet waar hij/zij was.
Om zo'n probleem aan te pakken kan men gebruik maken van 'differentiatie' en 'specialisatie'. Ofwel: het werk wordt verdeeld (differentiatie) en de werkers leggen zich toe op een bepaalde bewerking (specialisatie).
Dit kan ook bij het programmeren gedaan worden. Door bepaalde delen van het programma gespecialiseerde taken te laten volbrengen wordt het hoofdprogramma korter en beter leesbaar. In de ideale situatie is het hoofdprogramma niet meer dan een opzichter die zelf als enige taak heeft de verdeling en de coordinatie van het werk dat door andere delen van het programma gedaan wordt.

Subroutines

BASIC kent voor het laten uitvoeren van een bepaalde taak door een bepaald programmaonderdeel de subroutine. Het woord legt al iets uit. 'Sub' duidt op een ander niveau van deelwerkzaamheden en 'routine' duidt op het veelvuldig door dit programmaonderdeel uitvoeren van dezelfde taak.
Als voorbeeld het laten uitrekenen van de grootste gemene deler van twee getallen door de computer. De taken die de computer achter elkaar moet uitvoeren zijn:

- weergeven van de instrukties
- invoer van de getallen
- getallen op volgorde van grootte zetten
- rekenen
- afdrukken van het resultaat

Als deze deeltaken lijken wel erg klein voor het maken van aparte vertakkingen in een programma. Maar alleen door als voorbeeld een klein overzichtelijk takenpakket te nemen kan de subroutine goed

duidelijk gemaakt worden.
Als alle aparte taken door een strikte taakscheiding door aparte subroutines gedaan worden, wordt het programma als volgt:

```
10 REM grootste gemene deler
20 GOSUB 200
30 INPUT "Uw eerste GETAL";X
40 INPUT "Uw tweede GETAL";Y
50 GOSUB 300
60 GOSUB 400
70 GOSUB 500
80 END
200 REM instrukties
210 CLS : KEY OFF
220 PRINT "Dit progr. gebruikt de"
230 PRINT "Euclidische methode om de"
240 PRINT "grootste gemene deler van twee"
250 PRINT "getallen te vinden."
260 RETURN
300 REM grootste getal in A
310 IF X<Y THEN SWAP X,Y
320 RETURN
400 REM het rekenwerk
410 A=X:B=Y
420 RE=A MOD B
430 A=B
440 B=RE
450 IF RE<>0 THEN GOTO 420
460 RETURN
470 RETURN
500 REM afdrukken van resultaat
510 CLS:PRINT:PRINT:PRINT
520 PRINT "     **************************"
530 PRINT:PRINT "     DE GROOTSTE GEMENE DELER"
540 PRINT:PRINT "     van";X;"en";Y;"is";A
550 PRINT:PRINT "     **************************"
560 RETURN
```

Het te strikt toepassen van subroutines blijkt ook tot ongewenste resultaten te leiden. Hoewel het hoofdprogramma zeer kort is (regel 10-80) is dit geheel onleesbaar. De lezer van het programma moet telkens de subroutines bekijken om te zien wat het programma doet. Het is daarom aan te raden de REM opdrachten die de werking van een subroutine verduidelijken onmiddellijk achter de verwijzing naar de subroutine (GOSUB) te plaatsen.
Let u op de speciale deling in regel 420:

```
MOD
```

Dit is een deling van gehele getallen met een rest. Zo is 17/4 gelijk aan 4.25. Maar 17 MOD 4 is gelijk aan 4 (en een rest van 1).
Subroutines worden aangeroepen door de opdracht

```
GOSUB                          (ga naar subroutine)
```

Deze opdracht wordt gevolgd door het regelnumer waar de subroutine begint. Het is het handigste om daar hoge regelnummers voor te kiezen, zodat er altijd voldoende ruimte overblijft voor een ononderbroken hoofdprogramma. Zorg er voor dat de regel waar naar toe gesprongen wordt wel bestaat! Anders geeft de computer de foutmelding:

```
Undefined line number          (niet gedefinieerde regel)
```

Let op! U mag bij GOSUB alleen regelnummers als gehele getallen invoeren. U kunt hier niet gebruik maken van een berekening of een variabele.
Om van de subroutine weer terug te keren naar het hoofdprogramma dient de opdracht:

```
RETURN                         (keer terug)
```

Deze opdracht zorgt ervoor dat de computer terugspringt naar de regel volgend op de regel waar de GOSUB stond die bij deze RETURN hoort. In regel 50 staat GOSUB 300. De RETURN in regel 320 zorgt ervoor dat de computer terug springt naar de regel volgend op regel 50 en dat is in dit programma regel 60.
De opdracht

```
END                            (einde)
```

in regel 80 zorgt ervoor dat het programma stopt. Zou het programma aan het einde van het hoofdprogramma niet gestopt worden, dan zou de computer beginnen aan de regels vanaf 200. Maar deze regels waren bedoeld als subroutine en niet als hoofdprogramma. Komt de computer nu op deze manier bij regel 260 dan vindt hij daar een RETURN zonder dat hij eerst een GOSUB die daar bijhoort is tegen gekomen. Dit leidt tot de foutmelding:

```
RETURN without GOSUB           (RETURN zonder GOSUB)
```

Zorg er dus altijd voor dat de subroutines achter het hoofdprogramma staan, en dat de computer niet per ongeluk na het doorlopen van het hoofdprogramma (met vertakkingen) aan de subroutines kan beginnen. Plaats een END opdracht.

Het is mogelijk om een bepaalde subroutine verschillende keren aan te roepen vanaf verschillende programmaregels. Dit is vooral handig als een bepaalde taak verschillende keren gedaan moet worden. De computer houdt zelf bij vanaf welke regel hij naar de subroutine sprong en waar hij dus na de subroutine weer moet terugkeren.
De belangrijkste kenmerken van subroutines zijn:

- het zijn aparte programmaonderdelen
- ze zijn verschillende keren vanaf verschillende plaatsen aanroepbaar
- ze worden aangeroepen door GOSUB
- ze worden verlaten door RETURN
- variabelen uit het hoofdprogramma kunnen in de subroutine gebruikt worden

Kiezen tussen subroutines

Naast de hierboven genoemde manier van de GOSUB opdracht gebruiken, kent de MSX ook de opdracht

ON... GOSUB (Bij....Ga naar subroutine...)

Deze opdracht laat de computer springen naar een subroutine die aangegeven wordt door een getal achter ON.
Stel u heeft een hele serie waarden. U wilt dat de computer aan de hand van deze waarden naar een bepaalde subroutine springt. Bijvoorbeeld:

```
5 CLS : KEY OFF
10 INPUT "GEHEEL GETAL (>0, <8)";A
20 PRINT
30 ON A GOSUB 100,150,200,300,400,450,525
40 PRINT
50 GOTO 10
99 REM eerste subroutine
100 PRINT "Deze subroutine begint op 100"
110 PRINT "U koos dus getal 1"
120 RETURN
149 REM tweede subroutine
150 PRINT "Dit is regel 150"
160 PRINT "u koos getal 2."
170 RETURN
```

```
199 REM derde subroutine
200 PRINT "U heeft 3 gekozen"
210 RETURN
299 REM vierde subroutine
300 PRINT "U drukte een 4 in."
310 RETURN
399 REM vijfde subroutine
400 PRINT "5 was het getal."
410 RETURN
449 REM zesde subroutine
450 PRINT "een 6 was uw keuze."
460 RETURN
524 REM zevende subroutine
525 PRINT "7- geluksgetal."
530 RETURN
```

In regel 10 wordt om een geheel getal van 1 tot en met 7 gevraagd. Regel 30 zorgt ervoor dat er afhankelijk van de ingetikte waarde gesprongen wordt naar een subroutine die begint bij het als eerste, tweede, derde, vierde, vijfde, zesde of zevende opgegeven regelnummer achter GOSUB. U kunt daar zelf regelnummers voor kiezen. U moet er alleen op letten dat de regel waarnaar verwezen wordt bestaat en dat de bedoelde subroutine daar begint. Het is niet nodig dat de regelnummers op volgorde van grootte staan. Het is dus mogelijk een opdracht te geven als:

```
ON A GOSUB 100, 1000, 500, 3450, 200
```

Als u als waarde een 0 of een getal groter dan het aantal regelnummers achter de GOSUB opdracht invult, gaat de computer gewoon door met de volgende regel en wordt de gehele ON..GOSUB opdracht verwaarloosd.
Vult u een negatieve waarde of een getal groter dan 255 in dan krijgt u een foutmelding:

```
Illegal function call      (Illegale aanroep van funktie)
```

De waarde achter ON mag een variabele, een getal of een berekening zijn. Als het resultaat van de berekening of de waarde van de variabele geen geheel getal is, worden de cijfers achter de komma verwaarloosd. Voor de waarden kleiner dan 0 en groter dan 255 geldt hetzelfde als hiervoor.
Het terugkeren vanuit elke subroutine geschiedt door een RETURN opdracht. Door deze opdracht gaat de computer verder met de regel volgend op de regel met de ON..GOSUB opdracht.
Let in het voorbeeldprogramma op het gebruik van de REM opdrachten. Deze zijn allemaal geplaatst onmiddellijk voor het begin van de subroutine. Sommige programmeurs vinden dat alle REM's uit een programma gehaald moeten worden omdat ze onnodig het programma verlengen en geheugenruimte vergen. Door de REM's

telkens vlak voor de subroutine te plaatsen, wordt het programma niet beinvloed door het verwijderen van de REM's. Normaal gesproken is het echter beter om de REM opdrachten te laten staan daar zij de opbouw van het programma toelichten.

Kiezen tussen vertakkingen

De hierboven beschreven opdracht ON komt in de meeste BASIC dialecten voor. De MSX heeft het gebruik van ON echter verder ontwikkeld. Ook bij de opdracht GOTO kan de ON-opdracht gebruikt worden:

```
10 INPUT "Tik 1, 2 of 3";A
20 BEEP : ON A GOTO 10,20,40
30 PRINT
40 PRINT "u koos gelukkig getal 3."
```

Deze opdracht verschilt niet veel van de ON..GOSUB opdracht. Nu springt de computer echter niet naar een bepaalde subroutine, maar vertakt naar gelang de opgegeven waarde. In bovenstaand programma springt de computer terug naar regel 10 als u een 1 intikt. Als u een 3 intikt springt de computer naar regel 40, drukt een tekstje af en komt bij het einde van het programma. Als u echter 2 intikt, springt de computer naar regel 20 om daar in een eeuwige lus te geraken. Elke keer opnieuw is A dan 2, zodat de computer telkens weer naar 20 springt. U zult dit kunnen constateren door een geluid dat gevormd wordt door een heel stel snel achter elkaar klinkende blieps.

De ON.. opdracht kan ook gebruikt worden om de computer te laten controleren of een bepaalde gebeurtenis al heeft plaatsgevonden, en als dat zo is, naar een bepaalde subroutine te springen.
Om te beginnen de opdracht

ON KEY GOSUB

Met deze opdracht kunt u een subroutine koppelen aan het indrukken van een of meer van de funktietoetsen. Wordt een van de funktietoesten tijdens het doorlopen van het programma ingedrukt dan springt de computer naar de bijbehorende subroutine, voert deze uit en gaat weer verder met het hoofdprogramma. Tijdens het uitvoeren van de subroutine die bij een bepaalde funktietoets hoort, let de computer tijdelijk niet op het indrukken van de funktietoets. Dit om ongewenste herhalingen te voorkomen. Alleen als u in de subroutine de toets opnieuw indrukt, zal de computer onmiddellijk na het verlaten van de subroutine de

subroutine opnieuw gaan doorlopen. U kunt dit voorkomen door in de subroutine een opdracht KEY OFF op te nemen, zodat de computer niet meer op de toets let. Wilt u dat de computer na het doorlopen van de subroutine vervolgens weer op de toets gaat letten dan moet u daartoe weer opdracht geven met KEY ON.
Een voorbeeld van het gebruik van deze opdracht:

```
10 CLS
20 ON KEY GOSUB 100,200,300
30 KEY (1) ON : KEY (2) ON
40 KEY (3) ON
50 FOR A = 1 TO 9999
60 PRINT A
70 NEXT A
80 END
99 REM eerste subroutine
100 BEEP
110 PRINT "funktietoets 1"
120 PRINT
130 RETURN
199 REM
200 BEEP : BEEP
210 PRINT : PRINT "funktietoets 2"
220 PRINT
230 RETURN
299 REM
300 BEEP : BEEP : BEEP
310 PRINT : PRINT "funktietoets 3"
320 PRINT
330 RETURN
```

In regel 20 wordt de computer opgegeven dat hij bij het indrukken van een van de funktietoetsen 1, 2 of 3 naar een subroutine op regel 100, 200 of 300 moet springen.
In regel 30 en 40 worden de funktietoesten 'aangezet'. Vanaf deze regels let de computer voortdurend op de drie funktietoetsen.
De regel 50-70 vormen een normaal programma. De getallen 1 tot en met 9999 worden afgedrukt. Maar tijdens het doorlopen van dit programma let de computer de hele tijd op de funktietoetsen. Drukt u er eentje in dan wordt de aangegeven subroutine afgewerkt, waarna de computer weer verder gaat met het gewone programma. Merkt u op dat als u de funktietoetsen vele malen snel achter elkaar indrukt, of als u de subroutine te lang zou maken, de afwerking van de subroutines aanzienlijk vertraagd.

Op vergelijkbare wijze werkt

```
ON STRIG GOSUB
```

STRIG staat in deze opdracht voor SPACEBAR (spatiebalk) en TRIGGER (vuurknop van joystick). Door deze opdracht let de computer op het indrukken van de spatiebalk of van de vuurknop van een aangesloten joystick. Er vijf knoppen waarop gelet wordt:

0 = spatiebalk
1 = vuurknop 1 van joystick 1
2 = vuurknop 1 van joystick 2
3 = vuurknop 2 van joystick 1
4 = vuurknop 2 van joystick 2

Net als bij de ON KEY GOSUB opdracht zijn ongewenste herhalingen voorkomen doordat de computer tijdens het doorlopen van de subroutine alleen let op het opnieuw indrukken van de toets of de vuurknop.
Het volgende programma is gelijk aan het vorige, maar nu is de spatiebalk geactiveerd.

```
10 CLS
20 ON STRIG GOSUB 100
30 STRIG(0) ON
50 FOR A = 1 TO 9999
60 PRINT A
70 NEXT A
80 END
99 REM subroutine
100 BEEP
110 PRINT "spatiebalk"
120 PRINT
130 RETURN
```

In regel 20 wordt de computer erop gewezen dat hij op de spatiebalk moet letten. In regel 30 wordt de spatiebalk 'aangezet'. Let erop dat er geen spatie mag staan tussen de G en het openingshaakje.
Als u alleen de tweede vuurknop en de eerste joystick wilt activeren hoeft u niet voor de voorgaande (niet gebruikte) vuurknoppen allemaal regelnummers in te vullen. U plaats daarvoor alleen komma's. Dus:

```
ON STRIG GOSUB 100,,300
```

Op deze wijze gaat de computer naar regel 100 als de spatiebalk is ingedrukt en naar regel 300 als vuurknop 2 van joystick 1 is ingedrukt. Let er hierbij wel op dat u deze vuurknop activeert:

STRIG(3) ON

Bij het maken van tekenen in de vorm van sprites kunt u als opdracht gebruiken:

ON SPRITE GOSUB

Deze opdracht zal in het hoofdstuk over sprites behandeld worden.

Bij het opsporen van fouten en het repareren daarvan kunt u als opdracht gebruiken:

ON ERROR GOSUB

Deze opdracht zal behandeld worden in de bijlage over foutmeldingen.

Een opdracht waarmee u de computer na een bepaald tijdsinterval een handeling kunt laten verrichten is:

ON INTERVAL GOSUB

Achter het woord INTERVAL plaatst u een is-gelijk-teken en een tijdsinterval. Het interval geeft u aan in 50-sten van een seconde. Nadat de computer de opdracht

INTERVAL ON

tegengekomen is, let hij op de tijd. Zijn er voldoende 50-sten van een seconde verlopen, dan springt hij naar een subroutine.
Het volgende programma combineert een ON KEY en een ON INTERVAL opdracht.

```
10 CLS
20 KEY OFF
30 ON KEY GOSUB 100
40 KEY (1) ON
50 FOR A=1 TO 9999
60 PRINT A
70 NEXT A
80 END
99 REM
100 PRINT
110 BEEP
120 PRINT "funktietoets ingedrukt"
130 ON INTERVAL=100 GOSUB 500
```

```
140 INTERVAL ON
150 RETURN
500 PRINT
510 BEEP:BEEP:BEEP:BEEP
520 PRINT "twee seconden verlopen"
530 PRINT
540 INTERVAL OFF
550 RETURN
```

In de subroutine die aangeroepen wordt zodra funktietoets 1 ingedrukt wordt, wordt de computer opdracht gegeven de tijd bij te gaan houden (regel 130 en 140). Onmiddellijk daarna gaat de computer terug naar het hoofdprogramma, maar het tellen van de tijd gaat door. Na 100 50-sten van een seconde (twee seconden) treedt de tweede subroutine in werking. Deze drukt een tekst af, laat wat bliepjes horen en zet de tijdtelling weer uit.

De laatste uit deze serie opdrachten is vooral bedoeld als een beveiliging tegen inbraak in een programma. Elk programma is te stoppen door de STOP toets eventueel samen met de CTRL toets te gebruiken. Door de computer opdracht te geven naar een subroutine te springen als de STOP toets wordt ingedrukt in plaats van te stoppen, is het voor de gebruiker onmogelijk het programma te stoppen. Alleen door de RESET knop in te drukken is het programma gestopt, maar dan ook uit het geheugen gewist. Het volgende programma moet u op een cassette zetten voordat u het runt, anders moet u het weer opnieuw intikken. Het programma kan niet gestopt worden zonder het te vernietigen.

```
10 CLS
20 ON STOP GOSUB 100
30 STOP ON
40 PRINT "En ik kom NIET!"
50 GOTO 40
99 REM
100 PRINT
110 BEEP
120 PRINT "Nee hoor, ook niet door STOP"
130 PRINT : BEEP
140 RETURN
```

Funkties

Naast de subroutines bestaan de funkties. Het wezenlijke verschil met een subroutine is dat een funktie altijd een uitkomst tot resultaat heeft: een getal of een string. Een subroutine geeft niet altijd een antwoord als resultaat, maar doet soms alleen

iets. Een subroutine kan gewoon een aantal opdrachten achter elkaar uitvoeren. MSX kent verschillende funkties. Zo is de bewerking

SQR (SQuare Root = vierkantswortel)

een wiskundige funktie die de wortel van een getal of van de waarde van een variabele of van een berekening berekent.
U kunt echter ook uw eigen funkties definieren. Nadat u in een programma een funktie gedefinieerd heeft, en de computer de regels waarin dat gebeurt doorlopen heeft, kan een eigen gemaakte funktie net zo gebruikt worden als elke andere al in MSX BASIC aanwezige funktie. Dat wil zeggen dat overal in het programma deze funktie gebruikt kan worden om een bepaald resultaat te verkrijgen. Bijvoorbeeld:

```
10 DEF FNKWADRAAT(A)=A*A
```

Deze funktie geeft het kwadraat van een getal. Ongeacht welk getal aan de funktie opgegeven wordt, de funktie geeft het kwadraat van het opgegeven getal terug. Probeert u dat uit met onderstaand programma:

```
10 CLS : KEY OFF
20 DEF FNKWADRAAT(A)=A*A
30 INPUT "een getal:";X
40 PRINT X;"in het kwadraat is"; FNKWADRAAT(X)
50 GOTO 30
```

In regel 20 wordt de funktie gedefinieerd. U doet dit door eerst de opdracht

```
DEF FN
```

te tikken met direkt daarachter de naam van de funktie. Achter de naam van de funktie komen tussen haakjes de in de funktiedefinitie gebruikte variabelen. Vervolgens een is-gelijk teken en tot slot de berekening die u de funktie telkens wilt laten doen.
Het aanroepen van een funktie gaat op dezelfde wijze als het aanroepen van een variabele of van een berekening. U geeft FN en de naam van de funktie op. Tussen de haakjes plaatst u de waarden die de computer moet gebruiken bij het rekenen. Als waarde kunt u zowel getallen als variabelen gebruiken. Nadat FNKWADRAAT gedefinieerd is kan deze op de volgende manieren aangeroepen worden:

```
PRINT FNKWADRAAT(13)
of
LET A=10:LET B=5
PRINT FNKWADRAAT(A)*FNKWADRAAT(B)
```

De definitie van een funktie mag niet meer dan een enkele programmaregel in beslag nemen.
Bij het aanroepen van een funktie moet u erop letten dat u de funktieaanroep evenveel waarden meegeeft als er in de berekening gebruikt moeten worden. In bovenstaand voorbeeld werd er telkens een enkele waarde gebruikt en dus ook een enkele waarde opgegeven bij het aanroepen van de funktie. Het is ook mogelijk meer dan een waarde te gebruiken:

```
DEF FNVERSCHIL(X,Y)=ABS(X-Y)
```

Deze funktiedefinitie geeft telkens het absolute verschil tussen twee waarden (dat wil zeggen dat als het verschil negatief is, het minteken genegeerd wordt).
Deze funktie is aan te roepen door:

```
PRINT FNVERSCHIL(10,7)
PRINT FNVERSCHIL(A,B)
Z=FNVERSCHIL(A,B)*FNVERSCHIL(P,6)
R=FNVERSCHIL(A,FNVERSCHIL(P,Q))
```

U ziet dat allerlei combinaties mogelijk zijn en dat het ook mogelijk is binnen een funktieaanroep een funktie aan te roepen.

De variabelen de binnen een definitie gebruikt worden zijn zogenaamde lokale variabelen. Dat wil zeggen dat de waarde van de variabele binnen de procedure onafhankelijk is van de waarde van een eventueel gelijkluidende variabele in het hoofdprogramnma is.

15. EENVOUDIGE GRAFIEK

De verschillende schermen

De MSX kent verschillende schermen waarmee gewerkt kan worden. Deze worden allemaal aangestuurd door één grafische chip. Deze chip kan zelf 16K aan RAM-geheugen besturen, zodat de centrale verwerkingseenheid geen geheugenruimte verliest aan de opbouw van het beeldscherm. Voor de programmeur betekent dit dat hij/zij meer vrije geheugenruimte ter beschikking heeft.
De verschillende schermen zijn:

modus	grafiek	tekst	kleur	oplossend vermogen (resolutie)	sprites
0	nee	ja	2	40 bij 24 karaktertekens	nee
1	nee/ja	ja	2	32 bij 24 karaktertekens	ja
2	ja	nee/ja	16	256 bij 192 punten	ja
3	ja	nee/ja	16	64 bij 48 blokjes	ja

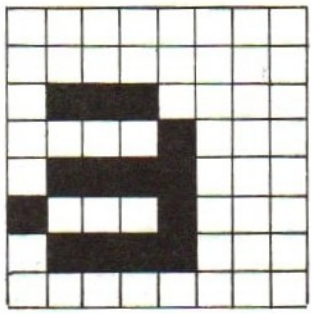
Omdat dit schema enige toelichting nodig heeft volgt hier een korte uiteenzetting van de mogelijkheden van elk scherm.

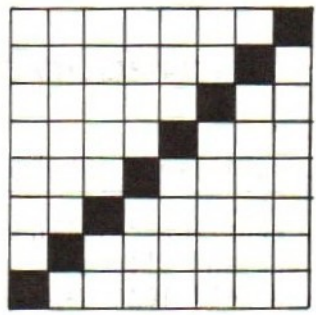
Scherm 0

Scherm 0 is het scherm waarin de computer staat als u hem net aangezet heeft, of onmiddellijk na een RESET. Dit is de zogenaamde 'default' modus, ofwel de standaard-modus. Dit beeldscherm biedt u veertig karakters op een regel. De standaardinstelling is echter 37 karakters op een regel. Dit is gedaan omdat niet elke televisie het gehele door de MSX gemaakte beeld kan weergeven. De randen vallen soms weg. Wilt u wel alle veertig karakters gebruiken dan dient u de opdracht

WIDTH (breedte)

te gebruiken.
De listings van uw programma zijn in deze schermmodus het beste te lezen. Vooral voor tekst is dit scherm dan ook bij uitstek geschikt.
Elk karakter wordt weergegeven in een rooster van 8 bij 8 hokjes. Hier ziet u de 'a' en een grafische schuine streep sterk vergroot:





De 'a' en de '/' sterk vergroot.

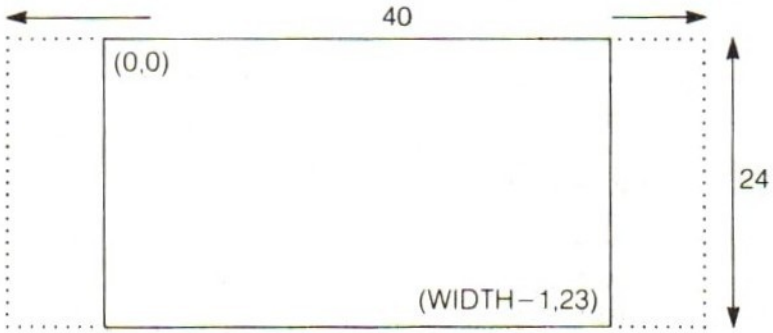
Let u goed op het gebruik van het rooster. De grafische schuine streep loopt van helemaal links in het rooster naar helemaal rechts in het rooster en van uiterst beneden naar uiterst boven. De 'a' daarentegen laat aan alle kanten veel wit vrij. Er zijn hier twee redenen voor:

- Zowel onder als boven de 'a' moet nog enige speling zijn om letters met 'poten' (b, d, f, g, h, j, k, l, p, q, t en y) in hetzelfde rooster weer te kunnen geven.
- Schermmodus 0 gebruikt voor elk karakter slechts 6 van de 8 roosterhokjes in de breedte. De twee rechterhokjes vallen af. Omdat ook in scherm 0 de letters niet tegen elkaar aan mogen komen te staan is de derde kolom van rechts ook leeg. Op deze wijze kunnen roosters van telkens 6 hokjes tegen elkaar aan geplaatst worden zonder dat de letters tegen elkaar aan komen te staan.

De grafische tekens die zowel in schermmodus 0 als 1 gebruikt kunnen worden hebben meestal een rooster van 8 bij 8 hokjes helemaal nodig. Deze karakters worden in schermmodus 0 niet volledig weergegeven.
In schermmodus 0 kunt u geen gebruik maken van sprites (zie het hoofdstuk over sprites). U kunt slechts twee van de zestien beschikbare kleuren gebruiken, maar u kunt zelf kiezen welke twee. De standaardinstelling is witte letters op een blauwe achtergrond (omdat dit meestal op een kleurentelevisie het scherpste beeld geeft). U kunt de kleur van de rand niet zelf bepalen. Deze is altijd gelijk aan de kleur van de achtergrond.
Zolang u niet de opdracht KEY OFF gebruikt zullen de funktiedefinities op de 23-e regel van het scherm getoond worden. Alle grafische opdrachten zijn in deze modus verboden. Probeert u zo'n opdracht toch dan volgt de foutmelding:

Illegal function call (illegale aanroep van funktie)

scherm 0, 40 bij 24 karakterposities, tekst SCREEN 0



Scherm 1

Dit scherm kan 32 bij 24 karaktertekens weergeven. Zoals u bij scherm 0 al las hebben deze karakterstekens wel de beschikking over het gehele 8 bij 8 rooster. De standaardinstelling van het scherm is 29 karakterposities breed. Wilt u dit veranderen dan moet u de WIDTH opdracht gebruiken.
U kunt maar twee van de 16 kleuren gebruiken, maar u kunt de rand een afwijkende, derde kleur geven. U gebruikt hiervoor de opdracht

COLOR

Achter COLOR kunt u drie getallen plaatsen. Het eerste getal staat voor de voorgrond, het tweede getal staat voor de achtergrond en het derde getal staat voor de randgebieden. Als u slechts een of twee van de geboden mogelijkheden wilt gebruiken, plaatst u voor de andere getallen een komma.

```
COLOR 1,7,15    zwarte voorgrond, blauwe achtergrond, witte rand
COLOR 11,4,2    gele voorgrond, blauwe achtergrond, groene rand
COLOR ,,6       donkerrode rand
COLOR 15        witte voorgrond
COLOR 5,,13     lichtblauwe voorgrond, magenta rand
COLOR ,1        zwarte achtergrond
```

De kleuren waaruit u kunt kiezen zijn:

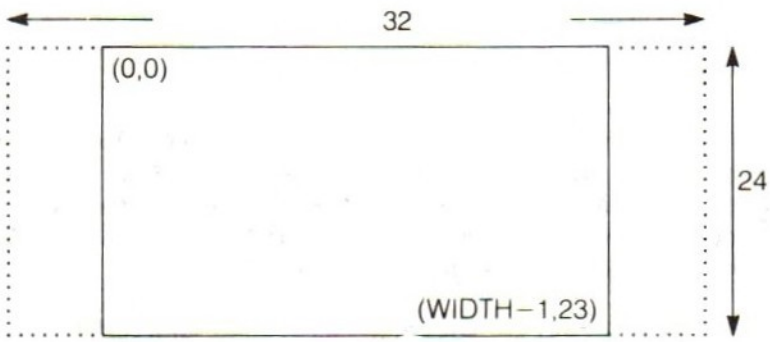
code	kleur	code	kleur
0	transparant	8	middelrood
1	zwart	9	lichtrood
2	middelgroen	10	donkergeel
3	lichtgroen	11	lichtgeel
4	donkerblauw	12	donkergroen
5	lichtblauw	13	magenta
6	donkerrood	14	grijs
7	middelblauw	15	wit

(transparant betekent: kleur van de rand)

U kunt in scherm 1 meer dan twee kleuren gebruiken, maar dan moet u gebruik maken van spritegrafiek.
Grafische opdrachten zijn verboden en leiden tot een foutmelding.

scherm 1, 32 bij 24 karakterpos., tekst en grafische symbolen

SCREEN 1

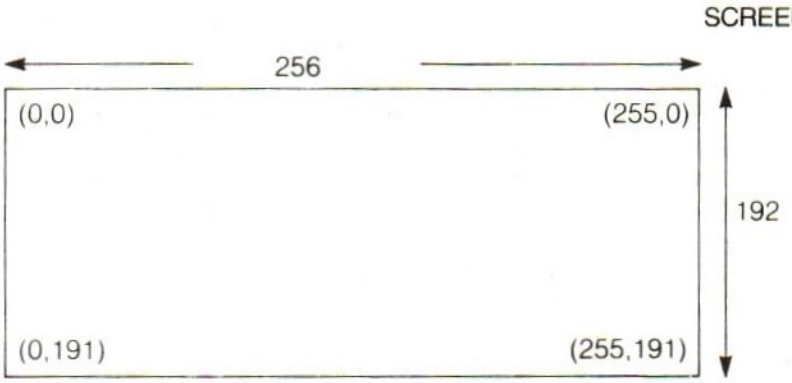


Scherm 2

Schermmodus 2 is bedoeld voor het maken van grafieken met hoog oplossend vermogen (ook wel genoemd: grafieken met een hoge resolutie).
In deze schermmodus is het beeld niet verdeeld in karakterposities waarbij op elke positie een bepaald teken geplaatst kan worden, maar is het scherm verdeeld in 256 bij 192 stippen. Van elke stip (pixel of dot genoemd) kunt u bepalen of hij aan of uit staat. De kleur van elke stip is niet afzonderlijk

te bepalen. De verdeling van de kleuren is bij de MSX minder fijn dan de verdeling van de stippen. De kleuren zijn verdeeld in blokken van 8 stippen breed en 1 stip hoog. Waar kent u die 8 stippen in de breedte van? Inderdaad, van de karakterposities. Die zijn immers ook 8 stippen breed.
U moet bij het maken van tekeningen in schermmodus 2 dus goed op de kleuren letten. Per blokje van 8 bij 1 kunt u slechts een voorgrond en een achtergrondkleur bepalen. Kiest u een tweede voorgrondkleur of een tweede achtergrondkleur, dan verandert het gehele blokje in de nieuwe kleur, en niet alleen de laatst gezette stip. Tot welke resultaten het kan leiden als u hier niet goed op let heeft u in het begin van het boek in een demonstratieprogramma kunnen zien.
De manier waarop de grafieken werken in deze schermmodus zal in hoofdstuk 17 behandeld worden.
Scherm 2 is een grafische scherm. Dat wil zeggen dat de nadruk gelegd is op het maken van grafieken. Het weergeven van tekst is niet goed mogelijk. De toetsdefinities kunnen niet weergegeven worden en u heeft een speciale truuk nodig om te PRINTen op een grafisch scherm (zie hoofdstuk 17).
De COLOR opdracht werkt in tegenstelling tot de tekstschermen niet onmiddellijk bij de grafische schermen. Om de achtergrond van kleur te kunnen veranderen moet u eerst een COLOR opdracht geven en vervolgens een CLS opdracht. Individuele tekeningen kunnen wel van kleur veranderd worden, maar daar gebruikt u de grafische opdrachten voor.
De INPUT opdracht kan niet gebruikt worden. U dwingt de computer met die opdracht terug te gaan naar het standaardscherm (scherm 0). De tekening komt daardoor te vervallen.
Scherm 2 heeft net als scherm 3 de eigenaardigheid dat de computer het beeld niet vasthoudt. Aan het einde van het programma springt de computer automatisch terug naar het standaardscherm 0. Om dit te voorkomen moet aan het einde van het programma een oneindige lus ingebouwd worden die ervoor zorgt dat het programma blijft lopen, zodat de computer niet terugspringt naar scherm 0.
In scherm 2 kunnen sprites gebruikt worden.

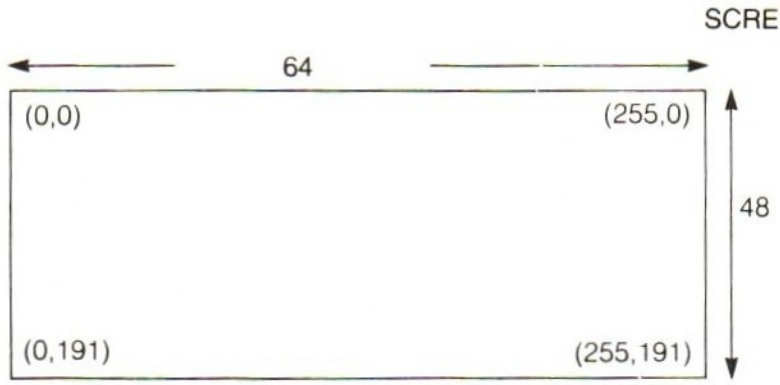
scherm 2, 256 bij 192 stippen, hoog oplossend vermogen grafiek



Scherm 3

Dit grafische scherm is minder fijn verdeeld dan scherm 2. U heeft nu de beschikking over 64 bij 48 blokjes. Daar staat tegenover dat u van elk blokje het aan en uit staan kunt bepalen en de kleur. U kunt elk blokje een eigen kleur geven. De kleuren zullen nooit, zoals in scherm 2, door elkaar gaan lopen.
De plaatsaanduiding van scherm 3 is precies gelijk aan die van scherm 2. Dat is prettig want daardoor kunt u programma's die gemaakt zijn voor het ene scherm gemakkellijk gebruiken voor het andere scherm. Op die manier kunt u snel controleren waarmee een tekening mooier uitkomt: fijne lijnen, of heldere kleuren. Soms kan het zelfs voor leuke 'gecomputeriseerde' tekeningen zorgen. Laat u de kaart van Nederland uit het hoofdstuk 'bestanden verwerken' maar eens tekenen in modus 3.
Schermmodus 3 werkt overigens net als scherm 2.

scherm 3, 64 X 48 blokken, veel kleuren grafiek



Grafiek op scherm 1

Hoewel de schermen 2 en 3 bedoeld zijn voor het maken van grafische weergaven is het soms toch handig om ook op een gewoon tekstscherm wat tekeningen te kunnen maken. Het kan voorkomen dat u tekst met een afbeelding moet verluchtigen.
Speciaal voor de tekstschermen zijn de MSX-computers uitgerust met een schat aan karaktertekens waaronder een heel stel grafische symbolen. Door deze symbolen op de juiste manier achter en onder elkaar te zetten kun u tekeningen maken.
U kunt alle grafische tekens (niet meegerekend de bijzondere lettertekens) op uw scherm zetten met behulp van het volgende programma.

```
10 SCREEN 1 : KEY OFF
15 PRINT
20 FOR A=16 TO 31
30 PRINT CHR$(1);CHR$(A+64);CHR$(32);
40 NEXT A
50 PRINT:PRINT
60 FOR A=192 TO 215
70 PRINT CHR$(A);CHR$(32);
80 NEXT A
90 PRINT:PRINT
100 FOR A=219 TO 223
110 PRINT CHR$(A);CHR$(32);
120 NEXT A
```

Aan het maken van een boek over een computer is bij de hoofdstukken over grafiek altijd het nadeel verbonden dat de precisie en scherpheid waarmee een computer grafische tekens op het scherm kan zetten, niet of nauwelijks op papier te evenaren is. Om onduidelijkheden bij het intikken te voorkomen zult u in deze paragraaf alle grafische tekens in de programmalistings tegenkomen als een PRINT CHR$(..) opdracht. Door deze opdracht drukt de computer het symbool af dat hoort bij de tussen haakjes geplaatste ASCII-code. Ale ASCII-codes vindt u in de bijlagen. De grafische symbolen met ASCII-code 1 tot en met 31 vragen speciale aandacht. De normale funkties van deze codes is in de officiele ASCII-set voorbehouden aan onder andere het besturen van een printer. Om deze codes toch te kunnen gebruiken als grafische tekens, dient u eerst een CHR$(1) te geven en

daarna een CHR$ met als waarde de gekozen ASCII-code plus 64! Bij het intikken is het niet absoluut noodzakelijk dat grafische symbolen intikt met behulp van de CHR$ opdracht. Als u een grafisch teken opneemt in een string, zal de computer er niets mee doen en bij het afdrukken de string precies zo weergeven als u opgegeven heeft.
De grafische tekens kunt u intikken met behulp van de GRAPH, SHIFT en CODE toetsen:

- Symbool rechts-boven: GRAPH + SHIFT + toets
- Symbool rechts-onder: GRAPH + toets
- Symbool links-boven: CODE + SHIFT + toets
- Symbool links-onder: CODE + toets

Sommige MSX-computers hebben alle symbolen op de toetsen aangegeven. Anderen geven de symbolen op een apart lijstje. De SONY HIT BIT geeft voor het type zonder aanduiding op de toetsen een speciale kaart mee die rechtop in een gleuf boven de toetsen geplaatst kan worden, zodat de symbolen toch snel afgelezen kunnen worden.

Het volgende programma geeft een voorbeeld van een eenvoudige tekening op scherm 1. De tekening kan ook geplaatst worden op scherm 0 (probeert u maar). In dat geval wordt de tekening smaller, en vallen er aan de rechterkant van elk grafisch teken twee kolommen met stippen weg.

```
10 SCREEN 1
20 COLOR 4,15,7
30 CLS : KEY OFF
40 B1$="  "+CHR$(206)+"  "
50 B2$="  "+CHR$(1)+CHR$(74)+"  "
60 B3$=" "+CHR$(1)+CHR$(88)+CHR$(219)+CHR$(1)+CHR$(89)+" "
70 B4$="  "+CHR$(1)+CHR$(74)+"  "
80 B5$="  "+CHR$(219)+"  "
90 B6$=" "+CHR$(1)+CHR$(93)+CHR$(219)+CHR$(1)+CHR$(94)+" "
100 B7$=" "+STRING$(3,CHR$(195))+" "
110 LOCATE 0,13
120 PRINT B1$
130 PRINT B2$
140 PRINT B2$
150 PRINT B3$
160 PRINT B4$
170 PRINT B5$
180 PRINT B6$
190 PRINT B7$
```

De regels 10-30 dienen om het scherm leeg te maken en de juiste kleuren te kiezen.
De regels 40-100 bepalen hoe de tekening eruit komt te zien. Dit zijn niets anders dan optellingen van karakters en strings. Elke rij van de tekening wordt in een aparte stringvariabele geplaatst. Het zijn deze opdrachten die u indien u dat wenst bij het intikken kunt vervangen door direkt ingetikte strings waarin de gevraagde symbolen voorkomen. In de ASCII-lijst in de bijlagen kunt u aflezen welke symbolen geplaatst moeten worden.
Regel 110 plaatst de cursor een eindje naar beneden.
De regels 120-190 zorgen ervoor dat de raket op het scherm afgedrukt wordt. Let u op regel 140. Hierin wordt de string van de tweede rij nogmaals afgedrukt. Daardoor wordt de raket wat langer. Wilt u een nog langere raket dan kunt u toevoegen:

```
165 PRINT B3$
168 PRINT B4$
```

Op dezelfde wijze kunt u de raket net zo lang of kort maken als u zelf wilt.
De opdracht

LOCATE (plaats)

geeft de computer opdracht om de tekstcursor naar een bepaalde positie op het scherm te verplaatsen. Door na die opdracht iets af te drukken, wordt de tekst op een bepaalde plaats op het scherm gezet. Het eerste getal achter LOCATE bepaalt hoever de cursor naar rechts verplaatst wordt. Het tweede getal bepaalt hoever de cursor naar beneden verplaatst wordt.

Lift-off

Om de raket te laten vertrekken is beweging nodig. In dit geval kunnen we beweging op het scherm maken door de raket telkens opnieuw op het beeldscherm te printen. Elke keer wordt de raket een beetje verschoven ten opzichte van de vorige keer. In de lage resolutie van het tekstscherm wil dat zeggen dat de raket telkens een karakterpositie of een regel hoger afgedrukt wordt. Probeert u het volgende programma.

```
10 SCREEN 1
20 COLOR 4,15,7
30 CLS : KEY OFF
40 B1$="  "+CHR$(206)+"  "
50 B2$="  "+CHR$(1)+CHR$(74)+"  "
60 B3$=" "+CHR$(1)+CHR$(88)+CHR$(219)+CHR$(1)+CHR$(89)+" "
```

```
70 B4$="  "+CHR$(1)+CHR$(74)+"  "
80 B5$="  "+CHR$(219)+"  "
90 B6$=" "+CHR$(1)+CHR$(93)+CHR$(219)+CHR$(1)+CHR$(94)+" "
100 B7$=" "+STRING$(3,CHR$(195))+" "
110 LOCATE 0,15
120 PRINT B1$
130 PRINT B2$
140 PRINT B2$
150 PRINT B3$
160 PRINT B4$
170 PRINT B5$
180 PRINT B6$
190 PRINT B7$
200 FOR A=14 TO 1 STEP -1
205 FOR Z=1 TO 50 : NEXT Z
210 LOCATE 0,A
220 PRINT B1$ : PRINT B2$
230 PRINT B2$ : PRINT B3$
240 PRINT B4$ : PRINT B5$
250 PRINT B6$ : PRINT "     "
260 NEXT A
270 GOTO 270
```

Tot regel 190 is het programma bijna identiek aan het vorige. Alleen de plaats waar de raket voor het eerst getekend wordt is ietsje lager. Op deze wijze kan de raket langer vliegen.
In regel 200 wordt een lus geopend die de raket verschillende malen gaat tekenen. De lus in regel 205 is alleen als vertraging bedoeld.
In regel 210 wordt de tekstcursor naar een bepaalde positie op het scherm verplaatst. Doordat deze positie afhankelijk is van A, is deze positie elke keer dat de lus doorlopen wordt een plaatsje hoger.
De regels 220-250 zorgen ervoor dat de raket telkens opnieuw geprint wordt. Let u op het tweede deel van regel 250. Hier wordt een aantal blanko spaties afgedrukt. Daardoor wordt de raket van onderen gewist. Zou u dit deel van de regel weglaten dan zou u een zeer lange raket krijgen. Probeert u dit om het resultaat te zien. Verwijder het tweede deel van regel 250 tijdelijk. De raket lijkt zo meer op een kerstboom.
In regel 270 wordt het programma in een oneindige lus geplaatst. Hoewel u in de paragrafen hiervoor las dat dit alleen nodig is bij de grafische schermen is er toch ook hier van deze truuk gebruik gemaakt. Weglaten van deze regel zou ervoor zorgen dat de cursor en de tekst Ok vlak onder de raket verschijnen. Dit is ontsierend.
Om de raket wat echter te doen lijken is in het volgende programma de vuurmond opgenomen en is de raket van geluid voorzien. De raket blijft op de grond staan totdat u de 's' van

'start' indrukt.

```
10 SCREEN 1
20 COLOR 4,15,7
30 CLS : KEY OFF
40 B1$="  "+CHR$(206)+"  "
50 B2$="  "+CHR$(1)+CHR$(74)+"  "
60 B3$=" "+CHR$(1)+CHR$(88)+CHR$(219)+CHR$(1)+CHR$(89)+" "
70 B4$="  "+CHR$(1)+CHR$(74)+"  "
80 B5$="  "+CHR$(219)+"  "
90 B6$=" "+CHR$(1)+CHR$(93)+CHR$(219)+CHR$(1)+CHR$(94)+" "
100 B7$=" "+STRING$(3,CHR$(195))+" "
110 LOCATE 0,15
120 PRINT B1$
130 PRINT B2$
140 PRINT B2$
150 PRINT B3$
160 PRINT B4$
170 PRINT B5$
180 PRINT B6$
190 PRINT B7$
200 T$=INKEY$ : IF T$<>"s" THEN 200
210 LOCATE 2,21 : PRINT CHR$(1);CHR$(79)
220 FOR A=14 TO 2 STEP -1
230 GOSUB 290
240 NEXT A
250 LOCATE 0,22 : PRINT "    "
260 GOSUB 290
270 GOTO 260
280 REM subroutine
290 FOR Z=1 TO 50 : NEXT Z
300 LOCATE 2,A+7 : PRINT CHR$(1);CHR$(79)
310 LOCATE 0,A
320 PRINT B1$ : PRINT B2$
330 PRINT B2$ : PRINT B3$
340 PRINT B4$ : PRINT B5$
350 PRINT B6$ : PRINT "     "
360 RETURN
```

Let u op de opdrachten in regel 200. Door de opdracht

```
INKEY$
```

kijkt de computer naar het toetsenbord. Telkens als de computer deze opdracht tegenkomt kijkt hij welke toets ingedrukt is. Door

als tweede deel van de regel een lus te maken naar het begin van de regel, blijft de computer bij deze regel wachten tot een bepaald teken ingedrukt is. In dit geval moet de computer telkens opnieuw regel 200 uitvoeren totdat de 's' ingedrukt wordt.

Lucifers

Het volgende programma laat zien dat ook met alleen lage resolutie grafiek best leuke programm's te maken zijn. Het spel zelf is heel eenvoudig. Door echter veel 'aankleding' te gebruiken wordt het voor de speler toch een aantrekkelijk geheel.

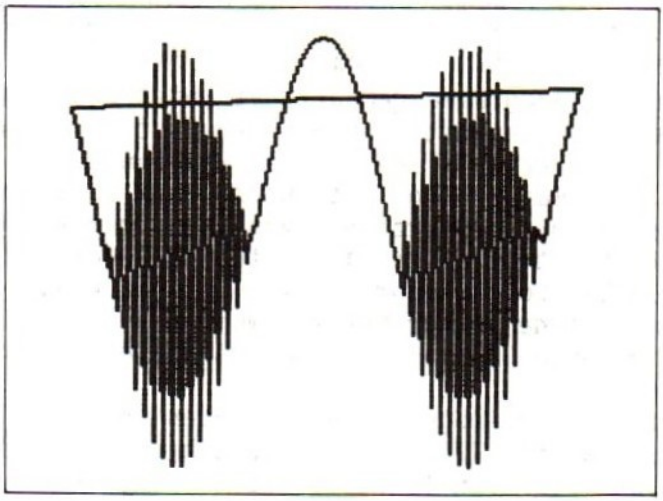
```
10 REM luciferssspel
20 SCREEN 1 : KEY OFF
30 COLOR 1,5,5
40 PRINT : PRINT : PRINT
50 PRINT STRING$(25,210)
60 PRINT : PRINT
70 PRINT "   HET 23 LUCIFERS SPEL "
80 PRINT : PRINT "Elke beurt mag u 1,2 of 3"
90 PRINT "lucifers weghalen. Ik doe"
100 PRINT "hetzelfde.
110 PRINT "De speler die de laatste"
120 PRINT "lucifer moet weghalen "
130 PRINT "heeft verloren."
140 PRINT : PRINT
150 PRINT STRING$(25,210)
160 PRINT : PRINT
170 PRINT "Willekeurig gekozen mag ..."
180 LU=23
190 FOR A=1 TO 10 : BEEP
200 FOR B=1 TO 300 : NEXT B : NEXT A
210 IF RND(1)<.6 THEN GOTO 300 : REM speler begint
220 SCREEN 0 : COLOR 6,11
230 PRINT : PRINT
240 PRINT "IK beginnen."
250 PRINT "Ik haal 2 lucifers weg."
260 LU=21 : GOSUB 1000 : REM weergave luc.
270 GOTO 500 : REM spel-lus
280 REM speler
300 SCREEN 0 : COLOR 6,11
310 PRINT : PRINT
320 PRINT "U beginnen."
330 LU=23 : GOSUB 1000 : REM weergave luc.
340 REM begin spel-lus
500 PRINT : PRINT : PRINT "Hoeveel lucifers neemt u weg?"
510 INPUT "(1,2 of 3)";SP
```

```
520 IF SP=1 OR SP=2 OR SP=3 THEN LU=LU-SP
    ELSE PRINT "U SPEELT VALS!!":GOTO 500
530 CLS : GOSUB 1000 : REM weergave luc.
540 PRINT : PRINT : PRINT "Ik denk ...."
550 FOR Z=1 TO 20 : BEEP :FOR Q=1 TO 20 : NEXT Q : NEXT Z
560 IF LU=4 THEN MS=3 : GOSUB 2000 : GOTO 500
570 IF LU=3 THEN MS=1 : GOSUB 2000 : GOTO 500
580 IF LU=2 THEN MS=1 : GOSUB 2000 : GOTO 500
590 IF LU=1 THEN GOTO 700 : REM gewonnen
600 IF LU<= 0 THEN GOTO 900 : REM verloren
610 IF LU>4 THEN MS=4-SP : GOSUB 2000 : GOTO 500
620 PRINT "fout in programmaloop"
630 END
699 REM speler gewonnen
700 SCREEN 1 : COLOR 11,6,7
710 LOCATE 2,2
720 FOR LR=1 TO 26
730 PRINT CHR$(1);CHR$(79);
740 NEXT LR
750 FOR BB=3 TO 23
760 LOCATE 2,BB
770 FOR LR=1 TO 26
780 PRINT CHR$(1);CHR$(86);
790 NEXT LR
800 COLOR INT(RND(1)*15),INT(RND(1)*15),INT(RND(1)*15)
810 BEEP : NEXT BB
820 COLOR 12,15,3
830 LOCATE 6,12 : PRINT "U HEEFT GEWONNEN"
840 END
899 REM speler verloren
900 SCREEN 1 : COLOR 15,1,1
910 LOCATE 6,12 : PRINT "U HEEFT VERLOREN"
920 END
999 REM weergave lucifers
1000 PRINT
1010 PRINT "    Nog";LU;"lucifer(s) aanwezig."
1020 LOCATE 6,6
1030 FOR LR=1 TO LU
1040 PRINT CHR$(1);CHR$(79);
1050 NEXT LR
1060 FOR BB=7 TO 14
1070 LOCATE 6,BB
1080 FOR LR=1 TO LU
1090 PRINT CHR$(1);CHR$(86);
1100 NEXT LR : NEXT BB
1110 RETURN
2000 CLS
2010 PRINT "Ik neem";MS;"lucifers weg."
```

```
2020 LU=LU-MS
2030 GOSUB 1000 : REM weergave luc.
2040 GOTO 500 : REM spel-lus
2050 RETURN
```

De regels 10-160 zorgen voor een introduktiebladzijde. De speler krijgt te zien om welk spel het gaat en hoe de spelregels zijn. Let op het gebruik van de STRING$ opdracht in de regels 50 en 150. Deze zorgen voor herhaling van een bepaald symbool, zodat sierranden gemaakt kunnen worden.
De regels 170-200 simuleren dat de computer aan het denken is. In werkelijkheid denkt de computer natuurlijk vele malen sneller. Door hier opzettelijk een ruime vertraging in te bouwen, wordt de speler de kans geboden de spelregels te lezen. Een andere methode hiervoor heeft u in vorige programma's gezien: de speler moet op een toets drukken als hij klaar is met lezen. Regel 190 zorgt voor tien bliepjes. Regel 200 zorgt voor een pauze tussen de bliepjes.
Regel 210 zorgt ervoor dat niet telkens dezelfde begint. Door een willekeurig getal te nemen met de RND opdracht krijgt de computer een waarde tussen 0 en 0.99999999999999 (als de waarde achter RND positief is). Door deze waarde te vergelijken met .5 zal gemiddeld een op de twee keer de computer de vergelijking als waar beschouwen en naar regel 300 springen. Springt de computer naar regel 300 dan begint de speler. Gaat de computer verder met regel 220 dan begint de computer.
De regels 220-250 geven aan dat de computer begint en dat de computer begint met 2 lucifers weg te halen.
Regel 260 geeft het nieuwe aantal lucifers aan (23-2=21) en springt naar de subroutine op regel 2000. Deze subroutine zorgt elke keer opnieuw voor de weergave van het juiste aantal lucifers.
Regel 270 laat de computer het programmadeel waarin behandeld wordt dat de speler begint, overslaan. Op regel 500 begint de lus die de rest van het spel regelt.
De regels 300-330 zijn gelijk aan 220-250, maar in dit geval begint de speler.
Op regel 500 begint de lus. Hier zal de computer net zolang terug komen tot het spel beeindigd is.
De regels 500-520 vragen de speler om aan te geven hoeveel lucifers hij/zij weg wil wegnemen. Let op regel 520 die valsspelen voorkomt. Regel 530 zorgt weer voor de weergave van de lucifers (subroutine op 1000).
De regels 540-550 zorgen voor een vertraging en opnieuw een simulatie van een 'denkende' computer.
De regels 560-610 bepalen hoeveel lucifers de computer telkens wegneemt.

Om de regels niet te lang te maken is het deel dat telkens hetzelfde is in een subroutine geplaatst (beginnend op regel 2000). Als regel 610 bereikt wordt, is het spel nog net in het laatste stadium en gebruikt de computer een eenvoudige aftreksom om te berekenen hoeveel lucifers er weggenomen moeten worden.
Regel 620 is een extra regel die alleen gebruikt wordt in het test-stadium van een programma. Als het programma goed loopt, kunt u deze regel weghalen. De regel geeft aan dat de computer die regel nooit mag bereiken. Doet hij dat toch, dan wordt er op het scherm afgedrukt dat er ergens een fout in de programmaloop zit. Heeft u een programma gemaakt dat niet goed wil lopen en waarvan u vermoedt dat dit ligt aan een fout in de programmaloop, plaats dan op verschillende punten waarvan u zeker bent dat de computer er niet mag komen dit soort regels. U kunt elke PRINT opdracht ook nog voorzien van een regelaanduiding. Dit vergemakkelijkt het opzoeken van fouten helemaal.
De regels 700-840 zorgen voor een knipperend, kleurig beeldscherm en de melding dat de speler heeft gewonnen.
De regels 900-920 geven een sobere melding dat de speler verloren heeft.
Wilt u de speler de mogelijkheid bieden het spel nog eens te spelen zonder dat hij RUN hoeft in te tikken, dan kunt u tussen regel 920 en 999 een INPUT opdracht plaatsen die de gebruiker vraagt of hij nog een spel wil. Naar aanleiding van het antwoord wordt het programma beeindigd of wordt er terug gesprongen naar het begin. Wilt u een programma waarvan de gebruiker niet zomaar de listing kan zien (daarin staat immers de 'taktiek' van de computer) dan gebruikt u de ON STOP opdracht uit het vorige hoofdstuk. Wil de speler het spel beeindigen dan moet het programma niet echt gestopt worden, maar in een oneindige lus geplaatst worden. Anders kan de gebruiker de listing toch nog heel gemakkelijk zien.
De regels 1000-1110 vormen de subroutine die de weergave van de lucifers verzorgd. Eerst wordt het aantal lucifers in letters en cijfers gemeld (regel 1010). Vervolgens wordt het juiste aantal luciferkoppen getekend (1030-1050). Tot slot worden een aantal malen net zoveel streepjes getrokken als er lucifers zijn. Doordat de streepje onder elkaar komen te staan, worden lucifers getekend (regel 1060-1100).
De regels 2000-2050 zorgen voor het rekenwerk van de computer en de weergave van zijn beslissing.



16. GRAFIEK EN KLEUR

Inleiding

In het vorige hoofdstuk las u al dat scherm 3 vooral geschikt is voor het weergeven van kleuren. Daarnaast heeft scherm 3 het 'voordeel' dat het alleen blokjes kent en geen stippen. Een lijn op scherm 3 getrokken ziet er dan ook uit als een serie tamelijk grove blokken. Enerzijds is dit een nadeel omdat op deze wijze op scherm drie geen fijne lijnen getrokken kunnen worden, maar anderzijds kunt u er leuke effekten mee bereiken.
Voor het tekenen van een mannetje hoeft u alleen een stel lijnen in de juiste kleur te trekken. De computer vult de blokken in waardoor het mannetje vaste vorm krijgt. Probeert u het volgende programma maar eens.

```
10 SCREEN 3 : COLOR 5,15,7
20 CLS
30 LINE (48,4)-(44,18),9
40 LINE (44,4)-(44,18),9
50 LINE (44,18)-(36,46),12
60 LINE (36,46)-(28,58),4
70 LINE (28,58)-(10,70),4
80 LINE (10,70)-(10,76),1
90 LINE (36,46)-(52,44),4
100 LINE (52,44)-(60,62),4
110 LINE (60,62)-(66,62),1
120 LINE (44,18)-(26,26),11
130 LINE (26,26)-(26,34),11
140 LINE (44,18)-(50,34),11
150 LINE (50,34)-(60,34),11
160 LINE (60,34)-(64,30),11
170 GOTO 170
```

In de regels 30-160 wordt telkens de opdracht

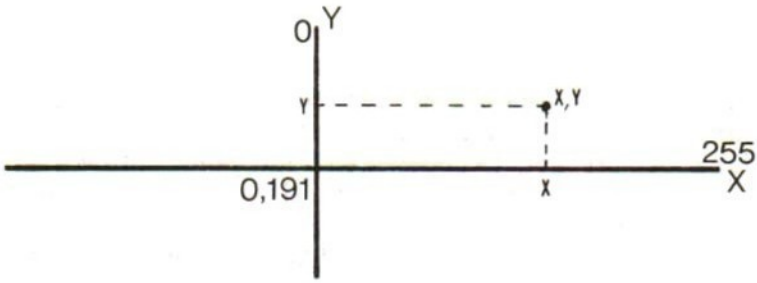
LINE

gebruikt. Deze opdracht zorgt ervoor dat de computer een lijn van het ene punt naar het andere punt trekt.

Plaatsbepaling

De punten worden opgegeven in een rechthoekig coordinatenstelsel. Van links naar rechts loopt de X-coordinaat van 0 tot en met 255.

Van boven naar onder loopt de Y-coordinaat van 0 tot en met 191.



In dit assenstelsel ziet u aangegeven hoe u een bepaald punt door middel van twee getallen kunt aangeven. Het eerste getal is altijd de X-coordinaat (van links naar rechts) en het tweede getal is altijd de Y-coordinaat (van boven naar beneden). De lezers die wiskunde gehad hebben worden extra gewaarschuwd: de schaal van de Y-AS loopt VAN BOVEN NAAR BENEDEN!
De LINE opdracht heeft twee getallenparen nodig. Een stel getallen voor het beginpunt van de te trekken lijn en een stel getallen voor het eindpunt van de te trekken lijn. Beide getallenparen worden tussen haakjes geplaatst, en tussen de twee paren komt een liggende streep.
Het is toegestaan om het eerste getallenpaar weg te laten (wel het streepje intikken!). In dat geval zal de lijn getrokken worden naar het tweede punt vanaf de huidige plaats van de grafische cursor.
De grafische cursor is vergelijkbaar met de tekstcursor. Het verschil is dat de grafische cursor niet zichtbaar is. Maar u kunt de plaats van de cursor wel beinvloeden. De cursor bevindt zich altijd daar waar de laatste grafische instruktie hem naartoe heeft gebracht. De grafische cursor kunt u overal plaatsen door alleen een enkele punt te zetten:

```
PSET ( , )                    (Point set = stel punt in)
```

Achter PSET plaatst u tussen haakjes twee getallen. Deze getalen geven de coordinaten van het af te drukken getal aan. Als derde getal kunt u achter de haakjes een kleurgetal invullen. Zonder dat derde getal wordt de punt getekend in de huidige voorgrondkleur. Met de kleurcode ingevuld wordt de punt getekend in de door de code opgegeven kleur.
Een vergelijkbare opdracht is:

```
PRESET ( , )        (point reset = stel punt opnieuw in)
```

Deze opdracht maakt de kleur van een punt gelijk aan de achtergrondkleur. Het effect daarvan is dat de punt gewist wordt. Door als derde getal een kleurcode in te vullen wordt het punt niet in de achtergrondkleur getekend maar in de opgegeven kleur. In dat geval is de opdracht hetzelfde als PSET met kleurcode.

Er is geen verschil tussen het coordinatenstelsel van scherm 2 en scherm 3. Het volgende programma kunt u zowel in scherm 2 als scherm 3 gebruiken. Probeert u beiden uit.

```
10 SCREEN 2 : COLOR ,15,14
20 FOR A=1 TO 300
30 X=INT(RND(1)*255)
40 Y=INT(RND(1)*192)
50 C=INT(RND(1)*15)
60 PSET (X,Y),C
70 NEXT A
80 GOTO 80
```

Dit programma zet driehonderd keer een punt. De punt wordt telkens op een willekeurige plaats gezet. De regels 30 en 40 zorgen voor een willekeurige X-coordinaat en een willekeurige Y-coordinaat. Regel 50 zorgt voor een willekeurige kleur.
Als u het programma runt zoals het hier staat krijgt u een heleboel hele kleine gekleurde puntjes te zien. Dat zijn de lichtpunten van scherm 2 met hoog oplossend vermogen.
Als u het programma runt in scherm 3 krijgt u allemaal gekleurde blokjes te zien. Elk blokje bestaat uit 4 bij 4 = 16 stippen van scherm 2.
Probeert u het effect als u in scherm 2 veel meer dan 300 stippen laat zetten. Verander in regel 20 het maximum aantal van de lus in 19000. Het programma duurt nu wat langer, maar het resultaat is leerzaam: de kleurweergave in schermodus 2 is niet perfect.

Het volgende programma laat zien dat ook in schermmodus 3 hele fraaie en soms ook nuttige grafieken gemaakt kunnen worden. Het volgende programma tekent een grafiek van twee sinusfunkties en een cosinusfunktie door elkaar heen. Doordat elke grafiek zijn eigen kleur heeft, zijn ze onderling goed te onderscheiden.

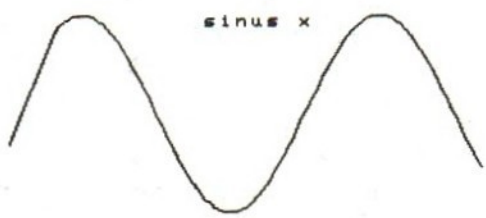
```
10 SCREEN 3 : COLOR ,15,15
20 PSET (0,100),6
30 FOR X=0 TO 12.5 STEP .1
40 LINE -(X*20,SIN(X)*70+100),6
50 NEXT X
60 PSET (0,100),5
70 FOR X=0 TO 12.5 STEP .1
80 LINE -(X*20,SIN(X*2.5)*50+100),5
```

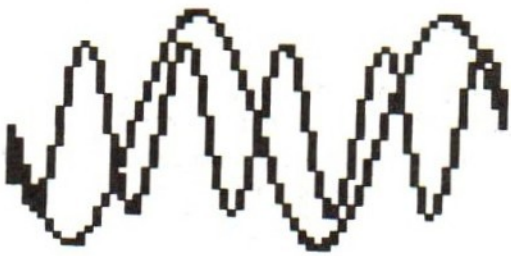
```
90 NEXT X
100 PSET (0,100),10
110 FOR X=0 TO 12.5 STEP .1
120 LINE -(X*20,COS(X)*70+100),10
130 NEXT X
140 GOTO 140
```

De grafieken worden getekend door de regels 30-50, 70-90 en 110-130. De PSET opdrachten in het programma dienen om de grafische cursor telkens aan het begin van de grafiek te zetten. De computer tekent door de opdracht:

LINE (,) - (,),

Tussen het eerste stel haakjes wordt het beginpunt van de lijn opgegeven. Tussen het tweede stel haakjes de coordinaten van het eindpunt. Achter de haakjes komt desgewenst de kleur van de lijn. Wordt er geen kleur opgegeven dan tekent de computer de lijn in de huidige voorgrondkleur. Door het eerste coordinatenpaar met haakjes weg te laten wordt de lijn getrokken vanaf de huidige positie van de grafische cursor naar het opgegeven punt. Van deze techniek is in dit programma veel gebruik gemaakt. Daardoor hoeft er minder uitgerekend te worden, waardoor het programma sneller wordt.





Wilt u het programma versnellen dan kunt u in de regel 30, 70 en 110 de stapgrootte van de lus vergroten tot .2 of .3. Hoe groter de stap, hoe groter de onnauwkeurigheid van de grafiek. Maar zeker bij .2 als stapgrootte is dat nog niet te zien.
Let u op de wijze waarop de normale sinusberekening aangepast is om te zorgen voor een mooie weergave van de grafiek. Bij een

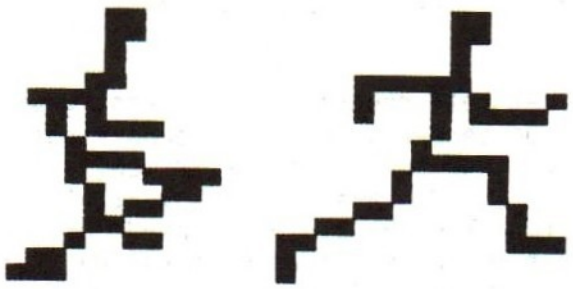
onbehandelde grafiek zou de Y-coordinaat altijd de waarde van SINus(X) hebben. De waarde van deze sinusfunktie ligt echter altijd tussen -1 en 1. Dat betekent een verschil van twee. Als we dit direkt op het scherm willen zetten krijgen we een rechte lijn. De blokjes van scherm 3 tellen immers 4 coordinaatpunten van links naar rechts en 4 coordinaatpunten van boven naar beneden. De sinusfunktie gaat dus minder dan een blokje op en neer. Daarom vergroten we de afstand tussen de minimum en maximum Y-waarde door een vergrotingsfactor. In bovenstaand programma werd dat: SIN(X)*70, SIN(X*2.5)*50 en COS(X)*70. De SIN(X) en COS(X) funktie werden evenveel vergroot en de SIN(X*2.5) werd iets minder vergroot. Op dezelfde wijze werd de grafiek in de X-richting vergroot. Voor alle drie de grafieken was dat X*20. Het totaalresultaat is dus dat de grafiek meer in de Y-richting dan in de X-richting vergroot is. De grafieken zijn dus een beetje vervormd.
Als laatste aanpassing is de hele grafiek een eind naar beneden verschoven. Dit is gedaan omdat zowel de sinus als de cosinusgrafiek ook negatieve uitkomsten hebben. Deze zouden dan bovenaan het scherm wegvallen. Let op! De Y-as loopt van boven naar beneden en niet van beneden naar boven. De grafieken staan onderste - boven. Het verschuiven gebeurt door telkens 100 op te tellen bij de Y-coordinaat. Zie de regels 40, 80 en 120.

Het volgende programma maakt ook gebruik van het verschuiven en vergroten van tekeningen. In dit programma wordt het mannetje van het eerste programma in dit hoofdstuk opnieuw getekend. Het mannetje wordt echter getekend in hele kleine coordinaten. Door deze nu allemaal met een bepaalde factor te vermenigvuldigen kan het mannetje groter of kleiner gemaakt worden. Door een verschuivingsfactor in te bouwen kan het mannetje over het scherm verplaatst worden. Tik het programma eerst in:

```
10 SCREEN 3 : COLOR 5,15,7
20 CLS
30 DIM PA(13) : DIM XA(13) : DIM YA(13) : DIM KA(13)
40 DIM PB(13) : DIM XB(13) : DIM YB(13) : DIM KB(13)
50 DIM PC(13) : DIM XC(13) : DIM YC(13) : DIM KC(13)
60 DIM PD(13) : DIM XD(13) : DIM YD(13) : DIM KD(13)
70 FOR Z = 1 TO 13
80 READ PA(Z),XA(Z),YA(Z),KA(Z)
90 NEXT Z
100 FOR Z = 1 TO 13
110 READ PB(Z),XB(Z),YB(Z),KB(Z)
120 NEXT Z
130 FOR Z = 1 TO 13
140 READ PC(Z),XC(Z),YC(Z),KC(Z)
150 NEXT Z
160 FOR Z = 1 TO 13
```

```
170 READ PD(Z),XD(Z),YD(Z),KD(Z)
175 NEXT Z
180 A=4 : B=4 : Y=90
185 FOR X=20 TO 230 STEP 30
190 GOSUB 1190
240 FOR L=1 TO 13
260 IF PA(L)=0 THEN PSET(XA(L)*A+X,YA(L)*B+Y),KA(L)
270 LINE -(XA(L)*A+X,YA(L)*B+Y),KA(L)
280 NEXT L
285 FOR W=1 TO 100 : NEXT W : CLS
290 GOSUB 1190
340 FOR L=1 TO 13
360 IF PB(L)=0 THEN PSET(XB(L)*A+X,YB(L)*B+Y),KB(L)
370 LINE -(XB(L)*A+X,YB(L)*B+Y),KB(L)
380 NEXT L
385 FOR W=1 TO 100 : NEXT W : CLS
390 GOSUB 1190
440 FOR L=1 TO 13
460 IF PC(L)=0 THEN PSET(XC(L)*A+X,YC(L)*B+Y),KC(L)
470 LINE -(XC(L)*A+X,YC(L)*B+Y),KC(L)
480 NEXT L
485 FOR W=1 TO 100 : NEXT W : CLS
490 GOSUB 1190
500 REM
540 FOR L=1 TO 13
560 IF PD(L)=0 THEN PSET(XD(L)*A+X,YD(L)*B+Y),KD(L)
570 LINE -(XD(L)*A+X,YD(L)*B+Y),KD(L)
580 NEXT L
585 FOR W=1 TO 100 : NEXT W : CLS
590 NEXT X
890 END
900 REM
910 DATA 0,-3,-3,11,1,-3,-5,11,1,2,-5,11,1,3,-3,11,1,6,-3,11,1,7,-4,11
920 DATA 0,-7,7,1,1,-7,5,1,1,-2,3,4,1,0,0,4,1,4,0,4,1,5,5,4,1,7,5,1
930 DATA 0,2,-4,11,1,-1,-5,11,1,2,-5,11,1,3,-2,11,1,6,-2,11,1,6,-3,11
940 DATA 0,-7,1,1,1,-5,1,1,1,-2,4,4,1,0,0,4,1,4,2,4,1,4,7,4,1,6,7,1
950 DATA 0,1,-1,11,1,-2,-2,11,1,2,-5,11,1,1,-2,11,1,3,-2,11,1,4,-2,11
960 DATA 0,-5,3,1,1,-4,0,1,1,0,3,4,1,0,0,4,1,4,3,4,1,-1,6,4,1,1,7,1
970 DATA 0,0,-1,11,1,-2,-4,11,1,2,-5,11,1,1,-3,11,1,3,-2,11,1,4,-2,11
980 DATA 0,-1,7,1,1,-3,7,1,1,1,4,4,1,0,0,4,1,7,1,4,1,2,4,4,1,4,5,1
1180 REM
1190 PSET (0+X,0+Y),12
1200 LINE -(2*A+X,-6*B+Y),12
1210 LINE -(3*A+X,-9*B+Y),9
1220 LINE (2*A+X,-6*B+Y)-(2*A+X,-9*B+Y),9
1230 RETURN
```

De regels 10 en 20 dienen voor de gebruikelijk schermopmaak.
De regels 30-60 dienen voor het dimensioneren van een stel arrays waarin de gegevens uit de datalijst geplaatst worden. De computer kan gegevens het snelste uit een array halen, en daar bij tekeningen snelheid zeer belangrijk is, worden de gegevens eerst in een array geplaatst.
Er worden vier verschillende standen van een mannetje getekend. Voor elke tekening zijn dertien tekenpunten nodig. Voor elk punt moet gegeven zijn of er lijn naartoe getrokken moet worden (eerste gegeven = 1) of dat de grafische cursor naar dat punt verplaatst moet worden zonder een lijn te trekken (eerste gegeven = 0). Vervolgens moeten van elk punt zowel de X-coordinaat als de Y-coordinaat bekend zijn (tweede en derde gegeven) en tot slot moet de kleur van de tekenopdracht bekend zijn (vierde gegeven). Er worden dus 4 maal 4 arrays gemaakt van elk 13 gegevens lang. Als u dat liever wilt, kunt u natuurlijk ook minder arrays maken, die dan meer dimensionaal zijn. Het eindresultaat blijft hetzelfde.
De regels 70-175 lezen alle gegevens in de arrays.
Regel 180 is heel belangrijk. Deze regel geeft een waarde aan de vergrotingsfactor van elke tekening en de verschuivingsfactor. U zult straks zien dat u vooral deze regel telkens kunt veranderen om allerlei speciale effecten te bereiken. Regel 185 zorgt voor een lus die het grootste deel van het programma omvat. Deze lus doet niets anders dan telkens als alle vier de tekeningen gemaakt zijn, de verschuivingsfactor van links naar rechts te vergroten. Daardoor 'loopt' het mannetje van links naar rechts over het scherm.

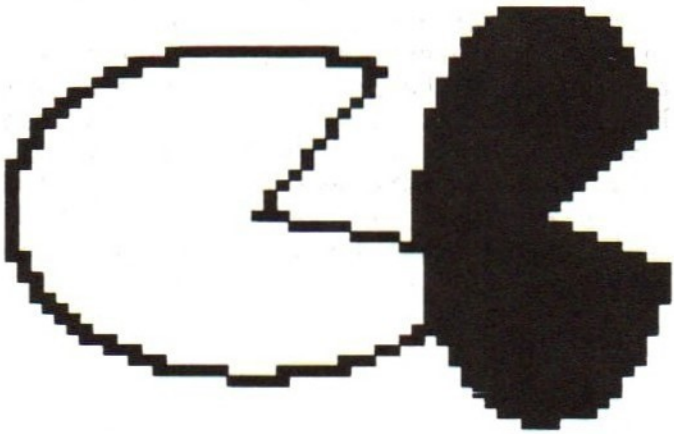


Regel 190 verwijst naar een subroutine die begint op regel 1190. Deze subroutine verzorgt het tekenen van het hoofd en lijf van het mannetje. Deze blijven namelijk altijd op dezelfde plaats, ongeacht de houding van de benen of de armen. Slechts de vergrotingsfactor en de verschuivingsfactor kunnen de plaats van het lichaam veranderen, maar deze factoren zijn door het gehele programma heen hetzelfde.
De regels 240-280 tekenen de eerste stand van het mannetje. Regel 260 kijkt of er een punt gezet moet worden. Als het eerste gegeven een 0 is, moet er naar dat punt (een uiteinde van een arm of been) geen lijn getrokken worden, maar moet de grafische cursor verplaatst worden. Bij deze opdracht kunt u de vergrotings- en verschuivingsfactoren al zien. De X-coordinaat XA(L) wordt altijd vermenigvuldigd met A en verschoven met X. De Y-coordinaat YA(L) wordt altijd vermenigvuldigd met B en verschoven met Y. Hoeft er geen punt gezet te worden dan tekent regel 270 een lijn naar het volgende punt.
Regel 285 zorgt voor een vertraging, zodat het getekende poppetje wat langer te zien is voordat hij door de CLS opdracht in dezelfde regel weer uitgewist wordt om plaats te maken voor de volgende tekening.
De regels 290-385, 390-485 en 540-585 doen precies hetzelfde, maar dan voor tekening B, C en D.
Regel 590 vormt de tweede opdracht van de lus van regel 185.
De regels 910-980 vormen een lange datalijst. De gegevens zijn telkens verdeeld in een regel voor de gegevens van de armen en een regel voor de gegevens van de benen. Dit vier keer, voor elke stand van het mannetje twee dataregels.
De regels 1190-1230 vormen een subroutine voor het tekenen van lijf en hoofd. Ook hier komt u de vergrotings- en verschuivingsfactoren weer tegen. In dit geval vindt u in de tekenopdrachten tevens de coordinaten verwerkt omdat deze constant zijn.

Doordat in het hele programma konsekwent gebruik gemaakt is van variabelen om de vergrotingen en verschuivingen te bepalen kunt u door eenvoudige ingrepen het mannetje heel anders maken.
Door in regel 180 de A en de B te verkleinen tot elk 2, wordt het mannetje twee keer zo klein getekend. Door een combinatie van coordinatensysteem en de stapgrootte van de X-lus in regel 185 heeft dit als bijkomend effect dat het mannetje schichtig om zich heen kijkt. Na elke stap draait hij zijn hoofd. Zo'n klein mannetje in schermmodus 3 zorgt wel voor een gepropt effect. Probeert u daarom ook een grotere waarde voor A en B uit (bijvoorbeeld 6 of 8). U kunt ook twee verschillende waarden nemen. Het mannetje wordt dan extra lang of extra breed.

Om de stappen die genomen worden te veranderen kunt u de stapgrootte in regel 185 veranderen.
Heeft u alle mogelijke combinaties uitgeprobeerd dan kunt u in regel 10 de schermmodus veranderen in SCREEN 2. Probeer de verschillende effecten die hierboven zijn beschreven nogmaals uit.

Het nadeel van alle hierboven genoemde effekten dat geen van alle zorgt voor een echt goed bewegend poppetje. De computer heeft teveel tijd nodig om het poppetje telkens te tekenen. Daardoor ontstaat een geknipper dat een goede bewegingssimulatie in de weg staat. Er zijn twee manieren om dit te veranderen: De eerste manier is het programma te herschrijven in machinetaal. Deze taal is veel sneller dan BASIC. Het nadeel daarvan is dat dit zeer veel extra kennis vraagt. Vanwege de ruimte is een machinetaalcursus niet in dit boek opgenomen. De tweede methode geeft echter ook in BASIC al redelijke resultaten. U moet daarvoor gebruik maken van sprites (patroontekeningen). Zie daarvoor hoofdstuk 17 en 18 .



17. GRAFIEK: HOGE RESOLUTIE

Het tekenen zoals dat in het vorige hoofdstuk gebeurde was wel kleurig, maar de lijnen werden een beetje dik. Daarom kent de MSX schermmodus 2. Deze modus geeft u de mogelijkheid om tekeningen met fijne lijnen te maken. Het beeldscherm is verdeeld in een rooster met 255 lijnen van links naar rechts en 191 lijnen van boven naar beneden. Het volgende programma biedt u de mogelijkheid om te oefenen met het hoge resolutie grafische scherm. U kunt door de vier pijltoetsen in te drukken een lijn over het scherm laten lopen. Op deze wijze zijn hele tekeningen te maken.

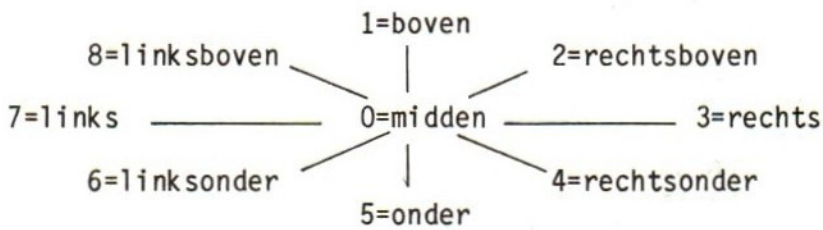
```
10 SCREEN 0 : KEY OFF
20 PRINT "      TEKENBORD "
30 PRINT : PRINT : PRINT
40 PRINT "U kunt tekenen met de pijlen"
50 PRINT "Het indrukken van de spatiebalk"
60 PRINT "zorgt ervoor dat het tekenen ophoudt"
70 PRINT "of juist weer begint."
80 PRINT : INPUT "Wat is uw beginpositie (X,Y)";X,Y
90 SCREEN 2
100 COLOR 4,15,14:CLS
110 Q=1:KL=4
120 ON STRIG GOSUB 250
130 STRIG(0) ON
140 PSET(X,Y),KL
150 REM lus
160 P=STICK(0)
170 IF P=0 THEN GOTO 220
180 IF P=3 THEN X=X+1 : GOTO 220
190 IF P=5 THEN Y=Y+1 : GOTO 220
200 IF P=7 THEN X=X-1 : GOTO 220
210 IF P=1 THEN Y=Y-1 : GOTO 220
220 LINE -(X,Y),KL
230 GOTO 160
240 REM subroutine
250 IF Q=1 THEN KL=4 : Q=0 : RETURN
260 KL=15 : Q=1 : RETURN
```

De regels 10-80 zorgen voor een introduktietekst die de gebruiker uitlegt wat er gaat gebeuren. Daarnaast wordt gevraagd om de beginpositie in te tikken. In regel 90 wordt er van scherm veranderd, zodat er een grafisch scherm is.
In regel 110 wordt Q op 1 ingesteld. Q is de variabele die aangeeft of er al dan niet getekend moet worden. Telkens als de

spatiebalk een keer wordt ingedrukt, verandert Q van 0 naar 1 of van 1 naar 0. De variabele KL staat voor de kleur waarin de lijn getrokken moet worden. Als KL gelijk is aan 4 wordt een blauwe lijn getrokken. Als KL gelijk is aan 15 wordt een witte lijn getrokken en is de lijn dus niet zichtbaar.
In regel 120 wordt de STRIG startklaar gezet. De computer let vanaf nu op het indrukken van de spatiebalk. Als die ingedrukt wordt dan wordt er gesprongen naar een subroutine op regel 250.
Regel 130 zet het effect van regel 120 'aan'.
Regel 140 verplaatst de cursor naar het punt dat opgegeven is door de gebruiker. Daar wordt een blauwe punt gezet.
Van regel 160 tot en met regel 220 is de lus die het tekenen verzorgt.
Regel 160 geeft de waarde van de pijltoetsen door aan de variabele P. U gebruikt hiervoor de opdracht

```
STICK( )
```

Tussen haakjes plaatst u een 0, 1 of 2. Een 0 geeft aan dat de computer moet letten op de pijltoetsen op het toetsenbord. Een 1 geeft aan dat de computer moet letten op de joystick die aangesloten is op ingang 1. Een 2 geeft aan dat de computer moet letten op de joystick die aangesloten is op ingang 2. De waarden die deze opdracht afgeeft zijn afhankelijk van de stand van de joystick of het indrukken van de pijltoetsen:



In de regels 170-210 worden de waarden van X en Y verhoogd of verlaagd aan de hand van het resultaat van de opdracht in regel 160. Drukt de gebruiker op de pijltoets naar boven dan is P gelijk aan 1. Volgens regel 210 wordt de Y-waarde dan met 1 verlaagd. Dit zorgt ervoor dat de grafische cursor zich een positie naar boven op het scherm zal bewegen (regel 220). Drukt de gebruiker op de pijltoets naar beneden dan is P gelijk aan 5. Door regel 190 wordt er dan 1 bij de Y-waarde opgeteld zodat een lijntje naar een positie lager op het beeldscherm getrokken wordt. Hetzelfde geldt voor het indrukken van de pijl-links en pijl-rechts waardoor de X-waarde respectievelijk kleiner en groter wordt doordat P respectievelijk 7 en 3 is.
Regel 220 zorgt voor het trekken van de lijn in kleur KL.
Regel 230 sluit de lus af.

Regel 250 en 260 vormen de subroutine die aangeroepen wordt als de spatiebalk ingedrukt wordt. Allereerst wordt gekeken welke waarde Q heeft. Is dat 1 dan wordt KL de waarde 4 gegeven. Q wordt vervolgens veranderd in 0 en de subroutine wordt verlaten. Is de waarde van Q niet gelijk aan 0 dan wordt de waarde van KL gelijk aan 15 en wordt Q gelijk gemaakt aan 1 waarna uit de subroutine gesprongen wordt. Het effect is dat het indrukken van de spatiebalk zorgt voor het wisselen van wel/niet tekenen.
Bij het niet tekenen kan de gebruiker wel telkens controleren waar de grafische cursor is door de spatiebalk twee keer in te drukken. Op de plaats van de grafische cursor zal even een blauwe stip verschijnen.
Wilt u in twee kleuren tekenen dan kunt u KL in plaats van 15 gelijk maken aan bijvoorbeeld 14. U kunt dan in blauw en grijs tekenen.
U kunt het nog mooier maken door de subroutine te veranderen in:

```
250 IF KL=15 THEN KL=0 ELSE KL=KL+1
260 RETURN
```

U kunt nu in elke kleur tekenen die u wilt. Door te letten op het eindpuntje van de lijn kunt u zien welke kleur nu gaat komen. Als u een stukje niet getekend wilt hebben drukt u net zo vaak op de spatiebalk totdat de witte kleur verschijnt. U verplaatst de cursor naar de door u gewenste positie en drukt vervolgens net zo vaak op de spatiebalk todat de door u gewenste kleur verschijnt. Dan kunt u weer verder tekenen.
U kunt terwijl u de pijltoetsen ingedrukt houdt op de spatiebalk drukken. De MSX let ook tijdens het verwerken van de pijltoetsen op het indrukken van de spatiebalk.

Stippen

In het vorige hoofdstuk leerde u de PSET opdracht kennen. Deze opdracht kan ook in scherm 2 gebruikt worden. U zag in het vorige hoofdstuk de mogelijkheid van het tekenen van een heleboel stippen op willekeurige plaatsen.
Doordat een willekeurige stip op het scherm ingevuld kan worden in een willekeurige kleur kunnen zeer sterke grafische afbeeldingen gemaakt worden. De gebruiker kan bijna alles wat in zijn/haar fantasie voorkomt tekenen. Het volgende programma tekent op willekeurige plaatsen een stip. De stippen beginnen echter in het midden van het scherm en zullen daar omheen blijven cirkelen.

```
10 SCREEN 2 : COLOR 4,15,15
20 PSET (126,95)
25 X=0 : Y=0 : DX=0 : DY=0
30 X=INT(RND(1)*10-4.5)
40 Y=INT(RND(1)*10-4.5)
50 DX=DX+X
60 DY=DY+Y
70 PSET (126+DX,95+DY)
80 GOTO 30
```

Alle opdrachten uit dit programma kent u al. Het moet niet zo moeilijk zijn de werking van dit programma te doorgronden. Een aspect van het programma trekt extra aandacht. Als u het programma een tijdje laat lopen komen er geen stippen meer bij. Het programma blijft echter in een oneindige lus stippen zetten. Als u het programma een tweede keer laat runnen gebeurt weer precies hetzelfde en ontstaat zelfs weer hetzelfde 'willekeurige' figuur. Dit komt omdat de MSX geen echte willekeurige getallen genereert, maar slechts een lijstje met pseudo-willekeurige getallen afwerkt. De tweede keer dat u dit lijstje afwerkt is het precies hetzelfde. U kunt u het willekeurige effect van de lijst vergroten door de variabele TIME.

Lijnen

Met behulp van de PSET opdracht is het ook mogelijk lijnen te trekken. Probeert u het volgende programma maar.

```
10 SCREEN 2
20 FOR X=1 TO 250
30 PSET(X,100),4
40 NEXT X
50 GOTO 50
```

In dit programma wordt X van 1 tot 250 telkens met 1 verhoogd. Op het punt (X,100) wordt vervolgens een stip gezet. Door op deze wijze een heleboel stippen achter elkaar te zetten krijgt u een rechte lijn. Op precies dezelfde wijze kunt u natuurlijk ook een lijn maken die van boven naar beneden loopt. U hoeft dan alleen de Y-coordinaat te laten veranderen door een FOR..NEXT.. lus. Om een scheve lijn te krijgen verandert u allebei de coordinaten. Bijvoorbeeld:

```
10 SCREEN 2
20 FOR X=1 TO 250
30 PSET(X,X/2),4
40 NEXT X
50 GOTO 50
```

In dit programma worden zowel de X- als de Y-coordinaat veranderd. Daarvoor wordt in een FOR..NEXT.. lus de X-coordinaat van 1 naar 250 verhoogd en wordt de Y-coordinaat telkens gelijk gemaakt aan de helft van de X-coordinaat. De scheve lijn die u nu te zien krijgt is dus een afbeelding van de funktie Y=X/2. (Ondersteboven vanwege de draaiing van de Y-as!)

Er is nog een andere opdracht om lijnen te trekken. Ook deze bent u al tegengekomen. Het is de opdracht

```
LINE ( , ) - ( , )
```

Tussen de haakjes worden de coordinaten van begin- en eindpunt opgegeven. Zo zou het programma hierboven herschreven kunnen worden tot:

```
10 SCREEN 2
20 LINE(1,1)-(250,125),4
30 GOTO 30
```

U ziet dat het programma flink wat korter kan. Daarbij komt dat het tekenen veel sneller gaat dan bij het telkens opnieuw zetten van een punt.
De LINE opdracht kent nog twee speciale effecten. De mogelijkheid om een rechthoek te tekenen en de mogelijkheid om een gevulde rechthoek te tekenen.

```
LINE (100,90) - (130,160),4,B
```

Als u deze regel in regel 20 van het vorige programma invult zal de computer voor u een rechthoek tekenen waarvan de linkerbovenhoek wordt aangegeven door het eerste coordinatenpaar, en de rechteronderhoek aangegeven wordt door het tweede coordinatenpaar. De letter B achter de opdracht geeft de computer te kennen dat hij een Box (doos, rechthoek) moet tekenen. Onthoudt u dus: bij de letter B in een LINE opdracht staan de coordiatenparen voor de linkerbovenhoek en de rechteronderhoek.
Op precies dezelfde wijze kunt u ook een gevulde rechthoek laten tekenen. U hoeft dan alleen achter de B een F te zetten. (van Fill = vullen). Probeert u dit ook uit in regel 20 van het vorige programma.
Als u wilt zien hoe snel dit alles gaat kunt u het volgende programma uitproberen.

```
10 SCREEN 2 : COLOR ,1,1 : CLS
20 KL=INT(RND(1)*15)
30 X1=INT(RND(1)*255)
40 X2=INT(RND(1)*255)
50 Y1=INT(RND(1)*191)
60 Y2=INT(RND(1)*191)
70 LINE (X1,Y1)-(X2,Y2),KL,BF
80 GOTO 20
```

Dit programma tekent willekeurige rechthoeken, die allemaal gevuld zijn. Als u open rechthoeken wilt hebben, moet u in regel 70 BF veranderen in B. Zelfs de keuze van BF of B kunt u willekeurig maken. Verander regel 70 in:

```
70 IF Q=1 THEN LINE(X1,Y1)-(X2,Y2),KL,BF ELSE
   LINE(X1,Y1)-(X2,Y2),KL,B
```

en voeg toe:

```
65 Q=INT(RND(1)*2)
```

Krommen

Met alleen rechte lijnen zijn zeer leuke tekeningen te maken, maar wat krommingen erin is wel zo mooi. U zag dat al aan de sinusgrafieken in het vorige hoofdstuk.
Een kromme lijn kan gezien worden als een verzameling punten of als een verzameling van zeer korte rechte lijnstukjes die allemaal onder een hoek ten opzichte van elkaar staan. De eenvoudigste manier om een gekromde lijn te krijgen is een verzameling punten te tekenen.

```
10 SCREEN 2
20 FOR X=0 TO 250
30 LET Y=INT(X*X/200)
40 PSET (X,Y)
50 NEXT X
60 GOTO 60
```

Dit programma tekent de grafiek van de funktie Y=X*X. Eigenlijk tekent het programma de funktie Y=X*X/200. De deling door 200 is bedoeld om de grafiek op een beetje redelijke wijze op het scherm te laten passen. Wordt een dergelijke schaalaanpassing niet gebruikt dan zou de grafiek als een bijna rechte lijn naar

beneden gaan. De grafiek staat ook ondersteboven. Wilt u dit veranderen (dit geldt voor alle andere programma's in dit hoofdstuk) dan moet u voor de Y-coordinaat niet de waarde van Y opgeven, maar 191-Y.
Probleem bij de grafiek is dat er weliswaar een schitterende kromme lijn uit komt, maar dat deze benedenaan (of bovenaan) een beetje uit elkaar dreigt te vallen doordat de punten te ver uit elkaar komen te staan. Dit kan verbeterd worden door geen punten te zetten, maar lijnen te trekken. Verander regel 40 in

```
40 LINE - (X,190-X)
```

en voeg toe

```
15 PSET (0,191)
```

Om het verschil tussen de twee mogelijkheden nog groter te maken kunt u beide programmaversies nog een keer runnen, maar dan met een delingsfactor van 100 in plaats van 200. Verander regel 30 in

```
30 LET Y=INT(X*X/100)
```

Om de grafiek helemaal op het scherm te krijgen (negatieve waarden van X ontbreken nu nog) kunt u weer gebruik maken van schaalvergroting, verkleining of verschuiving. Het resultaat wordt:

```
10 SCREEN 2
15 PSET(0,INT(-125*-125/90))
20 FOR X=-125 TO 125
30 LET Y=INT(X*X/90)
40 LINE -(X+125,190-Y)
50 NEXT X
60 GOTO 60
```

Een aardige toepassing van deze technieken is het volgende programma dat de loop van een stuiterbal simuleert. Een bal wordt voorwaarts weggeworpen.

```
10 SCREEN 0
20 BX=25
30 BY=50
40 VS=0 : N=0
50 PRINT : PRINT : PRINT"Dit prog. laat de loop van een "
60 PRINT "stuiterende bal zien."
70 PRINT
80 INPUT "Horizontale beginsnelheid (0-25)";HS
90 SCREEN 2
```

```
100 PSET (250,190)
110 LINE - (25,190)
120 LINE - (25,0)
130 PSET (BX,BY)
140 REM lus
150 BX=BX+HS/2
160 VS=VS+2
170 BY=BY+VS
180 IF BY>=190 THEN GOSUB 500 : GOTO 210
190 IF BX>= 255 THEN GOSUB 1000 : GOTO 210
200 IF BX<= 25 THEN GOSUB 1500
210 LINE - (BX,BY)
220 FOR A=1 TO (190-BY)/2 : NEXT A
230 IF N<=20 THEN GOTO 150
240 GOTO 240
250 END
499 REM grond
500 BY=190
510 VS=-.9*VS
520 N=N+1
530 BEEP
540 RETURN
999 REM muur rechts
1000 BX=255
1010 HS=-.9*HS
1020 BEEP
1030 RETURN
1499 REM muur links
1500 BX=25
1510 HS=-.9*HS
1520 BEEP
1530 RETURN
```

De eerste regels dienen voor het kiezen van de schermmodus en voor het instellen van de verschillende variabelen. De variabelen BX en BY geven de positie van de bal aan. VS is de vertikale snelheid, HS is de horizontale snelheid en N is het aantal keren dat de bal gestuiterd heeft. De horizontale snelheid kan door de gebruiker ingesteld worden door regel 80.
In de regels 90-130 wordt een ander scherm ingesteld en worden er kaders getrokken. Daarna wordt de grafische cursor klaar gezet op het beginpunt.
Regel 150 bepaalt de X-coordinaat van de bal. Deze is telkens gelijk aan de oude plaats plus de helft van de door de gebruiker ingevoerde horizontale snelheid.
De vertikale snelheid neemt toe (door de zwaartekracht). De vertikale snelheid is telkens gelijk aan de oorspronkelijke snelheid plus 2 (regel 160). De vertikale plaats (de

Y-coordinaat) is afhankelijk van de vorige plaats en van de vertikale snelheid.
In de regels 180, 190 en 200 kan naar aparte subroutines gesprongen worden. Dit is afhankelijk van de plaats van de bal. Bevindt de bal zich vlak bij de grond dan gaat de computer naar regel 500. Voor een botsing met de linker of rechtermuur gaat de computer naar regel 1500 of 1000.
Regel 210 zorgt voor het tekenen van de baan van de bal.
Regel 220 zorgt voor een vertraging. De vertraging is afhankelijk van de hoogte van de bal. Hoe hoger op het scherm de bal is, hoe langzamer de bal gaat. Ook dit is bedoeld om een betere simulatie te geven.
Regel 230 zorgt voor een beperkt aantal stuiters en regel 240 zorgt voor het blijven van het grafische scherm.
De subroutine van regel 500 zorgt ervoor dat de bal precies de grond raakt (BY=190) en keert de richting van de snelheid om (VS=-.9*VS). Daarbij wordt de snelheid een beetje teruggebracht, omdat een stuiterbal bij het raken van een object altijd een deel van zijn snelheid verliest. Wilt u de baan van een slechte stuiterbal zien, dan moet u in regel 510 de factor .9 verkleinen. Een absolute stuiterbal heeft een verliesfactor van 1.
De subroutine van regel 1000 is vergelijkbaar met de subroutine van regel 500. Nu wordt echter het stuiteren tegen de rechtermuur behandeld. De horizontale snelheid wordt omgedraaid en een beetje verkleind.
De subroutine van regel 1500 is gelijk aan die van 1000, maar dan voor de linkermuur.

Cirkels, ellipsen en taartstukken

Naast de PSET, PRESET en LINE opdrachten kent de MSX ook speciale opdrachten voor het tekenen van cirkels of delen daarvan. Dit is bijzonder handig. Een cirkel kan weliswaar net zo getekend worden als de grafiek die u hiervoor zag, maar omdat dit allemaal in BASIC is, gaat het veel langzamer dan de routine die zich in machinetaal al in de MSX bevindt.
Voor het tekenen van een cirkel geeft u het middelpunt van de cirkel en de straal op:

```
CIRCLE ( , ),
```

Tussen haakjes de coordinaten van het middelpunt, achter de haakjes de straal. Eventueel kunt u daarachter nog een kleurcode aangeven.

```
10 SCREEN 2
20 CIRCLE (100,90),50
30 CIRCLE (40,160),30,1
40 GOTO 40
```

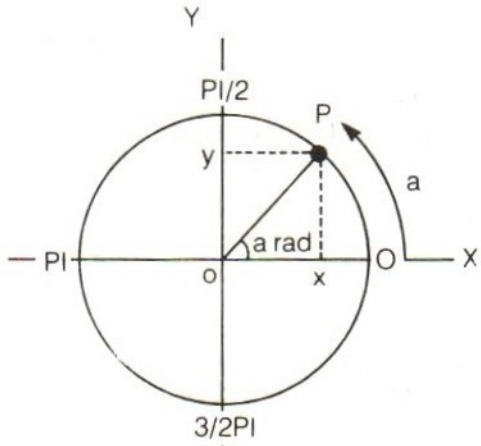
Dit kleine programma tekent eerst een cirkel met een straal 50 en middelpunt (100,90) in de huidige voorgrondkleur. Vervolgens wordt een cirkel met een straal 30 en (40,160) als middelpunt in zwart getekend.
Door dit kleine programma heeft u gelijk al gezien hoe ellipsen getekend kunnen worden. U geeft een cirkel op en u krijgt een ellips. Dit heeft te maken met de verhouding links-rechts en onder-boven van een televisie. Om nette cirkels te krijgen moet u de cirkels in de Y-richting met 1.4 vermenigvuldigen:

```
10 SCREEN 2
20 CIRCLE (100,90),50,,,,1.4
30 CIRCLE (40,160),30,1,,,1.4
40 GOTO 40
```

De verhouding tussen de Y-as en de X-as van een cirkel is het laatste gegeven dat u bij een CIRCLE opdracht op kunt geven. Getallen groter dan 1.4 zorgen voor ellipsen die hoog en smal zijn. Getallen kleiner dan 1.4 zorgt voor ellipsen die breed en laag zijn. Om u een idee te geven van de mogelijkheden hier weer een al bekend programma om op willekeurige plaatsen een tekening te maken. In dit geval geen lijnen of rechthoeken maar ellipsen.

```
10 SCREEN 2 : COLOR ,1,1 :CLS
20 KL=INT(RND(1)*15)
30 X=INT(RND(1)*255)
40 Y=INT(RND(1)*255)
50 EL=RND(1)*3
60 ST=INT(RND(1)*70)
70 CIRCLE (X,Y),ST,KL,,,EL
80 GOTO 20
```

Zoals u in de beide vorige programma's zag, passen er tussen de aanduiding van de kleur en de verhouding tussen Y-as en X-as nog twee getallen. In de vorige twee programma's werden die getallen niet gebruikt en werden er dus komma's geplaatst.
Met deze twee getallen kunt u de computer opgeven om een deel van een cirkel te tekenen. U geeft daarvoor de beginhoek en de eindhoek op. Deze hoeken moet u uitdrukken in radialen en moeten liggen tussen 0 en 2*PI. Een volledige cirkel bestaat uit 2 PI radialen. Zie onderstaande tekening:



De hoeken worden gemeten vanaf de X-as, tegen de wijzers van de klok in.
De waarde PI bevindt zich niet in de MSX. U kunt de waarde uitrekenen door 4 maal de arctangens van 1 te nemen. In onderstaand programma ziet u hiervan een voorbeeld.

```
10 SCREEN 2
20 COLOR 4,1,1 : CLS
30 PI=4*ATN(1)
40 CIRCLE (100,90),60,,0,PI,1.4
50 GOTO 50
```

Dit programma tekent een halve cirkel. Het middelpunt van de cirkel ligt op het punt (100,90), de straal is 60 en het deel tussen 0 en PI wordt getekend. Dat wil zeggen dat de bovenste helft van de cirkel getekend wordt. Probeert u in bovenstaand programma de verschillende mogelijkheden uit. Verander zo vaak u wilt de getallen die nu 0 en PI zijn. U hoeft niet alleen getallen op te geven die uitgedrukt in PI zijn. U kunt ook gewone getallen opgeven. Het gaat erom dat u een aantal radialen opgeeft. Als u echter meer dan ongeveer 6.28 radialen (2*PI) opgeeft krijgt u een foutmelding te zien.
Door de verhouding 1.4 te veranderen kunt u ook delen van ellipsen tekenen. U moet een ellips daarbij zien als een vervormde cirkel.
Door de getallen die de begin- en eindhoek aangeven te voorzien van een minteken (-) wordt niet alleen een deel van de cirkel getekend, maar worden de uiteinden van de cirkelboog verbonden met het middelpunt van de cirkel. U krijgt daardoor het effect van een 'taartpunt'. Verander in bovenstaand programma regel 40 in:

```
CIRCLE (100,90),60,,-.5,-5.5,1.4
```

Deze technieken kunt u samenvoegen tot een aardig programma om mooie tekeningen te maken. De computer wordt als een soort spirograph gebruikt. Door het herhalen van een tekening, telkens met een kleine afwijking krijgt u fraaie resultaten. Probeert u zelf het programma op verschillende punten te veranderen en kijk welke resultaten u krijgt.

```
10 SCREEN 2
20 COLOR 4,1,1 : CLS
30 PI=4*ATN(1)
40 FOR A=1 TO 30
50 CIRCLE (100+A,60+A),10+A*2,,-.5,-6,1.4
60 NEXT A
70 GOTO 70
```

Inkleuren

Naast de mogelijkheden die de LINE opdracht biedt voor het vullen van een rechthoekige vorm, kent de MSX ook een opdracht voor het vullen van een willekeurige vorm. De opdracht daarvoor is

```
PAINT ( , ),
```

Tussen de haakjes komen de coordinaten van een punt dat binnen de te vullen vorm moet liggen. De waarde die na de haakjes komt geeft de kleur aan waarmee het vlak gekleurd moet worden. De kleur van de rand moet in schermmodus 2 gelijk zijn aan de kleur van het vlak.
Na een PAINT opdracht zoekt de computer beginnend bij het opgegeven punt van links naar rechts en van boven tot onder de vorm af. De opgegeven kleur wordt net zo veel getekend totdat de vorm helemaal vol is. Bekijkt u het volgende programma dat eerst een ellips in rood tekent en deze vervolgens inkleurt.

```
10 SCREEN 2 : COLOR ,15,15 : CLS
20 CIRCLE (150,100),80,6,,,.5
30 FOR Z=1 TO 200 : NEXT Z
40 PAINT (150,100),6
50 GOTO 50
```

Probeert u het effect uit als u in regel 40 de PAINT opdracht naar een andere kleur laat zoeken dan de kleur waarin de ellips getekend is. Verander regel 40 in

```
40 PAINT (150,100),9
```

Ook minder regelmatige vormen zijn mogelijk. Verander regel 20 en 40 in

```
20 CIRCLE (150,100),80,9,-1,-6,.5
40 PAINT (140,100),9
```

Merkt u op dat regel 40 veranderd moet worden om te zorgen dat het punt van waaruit gePAINT gaat worden binnen de vorm ligt.

In schermmodus 2 is het mogelijk om de rand van een andere kleur te maken dan het binnenste van de vorm. Probeer het volgende programma uit.

```
10 SCREEN 3 : COLOR ,15,15 : CLS
20 CIRCLE (120,100),110,4,-1,-6,.6
30 FOR Z=1 TO 200 : NEXT Z
40 PAINT (110,100),9,4
50 GOTO 50
```

Dit programma tekent een blauw stuk taart. Vervolgens wordt de taart rood ingekleurd terwijl de rand blauw blijft. U moet daarvoor achter de coordinaten van PAINT twee getallen invullen. Het eerste getal slaat nog steeds op de kleur waarmee de vorm gekleurd moet worden. Het tweede getal geeft de kleur van de rand aan tot waar er gekleurd moet worden. U kunt daardoor selectief inkleuren. Bekijkt u het volgende programma maar eens.

```
10 SCREEN 3 : COLOR ,15,15 : CLS
20 CIRCLE (120,100),110,4,-1,-6,.6
30 FOR Z=1 TO 200 : NEXT Z
40 CIRCLE (195,100),80,1,-.5,-6,2
50 FOR Z=1 TO 200 : NEXT Z
60 PAINT (190,130),9,1
70 GOTO 70
```

Tekst op het grafische scherm

Om tekst op het grafische scherm te krijgen is een speciale truuk nodig. U kunt bij een grafisch scherm immers de INPUT of de PRINT opdracht niet gebruiken.
Om nu toch teksten bij grafieken te kunnen plaatsen, moet u een bestandskanaal openen en de tekst via dit kanaal naar 'het bestand grafisch scherm' sturen. Het volgende programma maakt al veel duidelijk.

```
10 SCREEN 2 : COLOR 1,15,15 : CLS
20 PSET (0,100)
30 FOR X=1 TO 10 STEP .1
40 LINE -(X*24,80-SIN(X)*60),4
50 NEXT X
60 OPEN "grp:" AS #1
70 PSET (100,20),15
80 PRINT #1,"sinus x"
90 CLOSE #1
100 GOTO 100
```

Regel 10-50 kent u. Deze stellen het scherm in, bepalen de kleuren en tekenen een sinusgrafiek.
In regel 60 wordt een kanaal geopend met als aanduiding

```
"GRP:"
```

Hierdoor wordt de tekst naar het grafische scherm gestuurd.
Regel 70 verplaatst de grafische cursor. Let u hierbij op. U bent bezig op het grafische scherm en de teksten worden dan ook geplaatst op de positie van de grafische cursor. De opdracht LOCATE werkt bij de grafische schermen niet.
Regel 80 stuurt de tekst naar het bestand. In dit geval het grafische scherm.
Regel 90 sluit het bestand weer.
Door gebruik te maken van de COLOR opdracht kan de tekst van een kleurtje worden voorzien. Voegt u in bovenstaand programma regel 65, 68 en 85 toe en verandert u regel 70:

```
65 FOR KL=1 TO 14
68 COLOR KL
75 PSET (70+3*KL,10+4*KL),15
85 NEXT KL
```

U ziet dat de letters boven op elkaar getekend worden. Dat komt doordat de computer in het grafische scherm niet eerst iets uitwist voor hij iets nieuws neerzet. Dit kunt u vooral goed zien door het vorige programma in een aangepaste versie op scherm 3 te

runnen.

```
10 SCREEN 3 : COLOR 1,15,15 : CLS
20 PSET (0,100)
30 FOR X=1 TO 10 STEP .1
40 LINE -(X*24,80-SIN(X)*60),4
50 NEXT X
60 OPEN "grp:" AS #1
70 FOR KL=1 TO 14
80 COLOR KL
90 X=10+3*KL:Y=5*KL
100 DRAW "bm=x;,=y;"
110 PRINT #1,"sinus x"
120 NEXT KL
130 CLOSE #1
140 GOTO 140
```

U kunt aan dit programma zien dat het heel goed mogelijk is om door enkele malen een tekst ietsje verschoven af te drukken verschillende fraaie letters te maken zijn. Experimenteert u hier veel mee. De resultaten kunnen verbluffend zijn. Zeker als u de teksten in verschillende richtingen verschuift.

Varia

U kunt met alle tot nu toe behandelde technieken allerlei leuke tekeningen maken. Elk van deze tekeningen gaat ervan uit dat de computer een eenvoudige tekening vele malen herhaalt, net als het programma hiervoor.

Het volgende programma tekent een bril.

```
10 SCREEN 2
20 COLOR 4,15,9 : CLS
30 FOR X=-7 TO 4 STEP .2
40 Z=12*SIN(X)
50 FOR Y=-Z TO Z
60 LINE -(X*15+125,90-Y*7)
70 NEXT Y
80 NEXT X
90 GOTO 90
```

Probeert u de verschillende effecten uit door de stapgrootte in de regel 30, 40 en 50 te veranderen. U kunt het bereik ook telkens veranderen. OP basis van dit ene programma zijn vele verschillende tekeningen te maken.

De DRAW opdracht

Naast de tekenopdrachten die elk een speciale funktie hebben kent de MSX ook een algemene tekenopdracht. Dit is:

```
DRAW "      "                          (teken)
```

Deze algemene tekenopdracht simuleert het tekenen met een potlood en papier. De opdracht zelf vertelt de computer niet veel meer dan dat de gegevens die na de opdracht in een string staan allemaal tekenopdrachten zijn. Van deze 'deelopdrachten' bestaat een hele reeks. Elke deelopdracht bestaat uit een enkele letter gevolgd door een of meer getallen.
De vier eenvoudigste deelopdrachten zijn

```
U                          (Up = omhoog)
D                          (Down = omlaag)
R                          (Right= rechts)
L                          (Left = links)
```

Na deze deelopdrachten dient u de lengte van de te trekken lijn op te geven. De lijn wordt opgegeven in een aantal stippen. Bijvoorbeeld

```
10 SCREEN 2
20 DRAW "D55R87"
30 GOTO 30
```

Op deze wijze wordt een lijn getrokken - vanaf de huidige positie van de grafische cursor - 55 lichtpunten naar beneden en 87 lichtpunten naar rechts. Merkt u op dat de lijn elders op het scherm getrokken wordt als u het programma een tweede keer runt.

De grafische cursor staat dan immers elders.
Om lijnen te trekken die niet evenwijdig aan de X-as of Y-as lopen kunt u een of meer van de volgende deelopdrachten gebruiken om diagonale lijnen te trekken:

E	(naar rechtsboven)
F	(naar rechtsonder)
G	(naar linksboven)
H	(naar linksonder)

U gebruikt deze opdracht net als de U, D, R en L deelopdrachten. Nu geeft het getal achter de deelopdracht echter niet de absolute lengte van de lijn aan, maar de verplaatsing in de X-richting en de Y-richting. De werkelijke lengte van de getrokken lijn is dus de wortel uit tweemaal het kwadraat van het opgegeven getal (stelling van Pythagoras).
Probeert u het volgende programma uit:

```
10 SCREEN 2
20 PSET (150,50)
30 DRAW "G100"
40 GOTO 40
```

U ziet nu een diagonale lijn.
De kracht van de DRAW opdracht ligt in de mogelijkheid om alle deelopdrachten achter elkaar te plaatsen. Vervangt u regel 30 door

```
30 DRAW "G100U60R110D30F40U100G100"
```

U kunt de deelopdrachten ook onderbrengen in een stringvariabele:

```
10 SCREEN 2
20 PSET (150,50)
30 A$= "G100U60R110D30F40U100G100"
40 B$= "U150L50F100"
50 DRAW A$+B$
60 GOTO 60
```

In dit programma worden door de DRAW opdracht twee verschillende strings achter elkaar als deelopdrachten beschouwd. Het is ook mogelijk om een string binnen een andere string te gebruiken. Verandert u regel 40 in

```
40 B$= "U150XA$;L50F100"
```

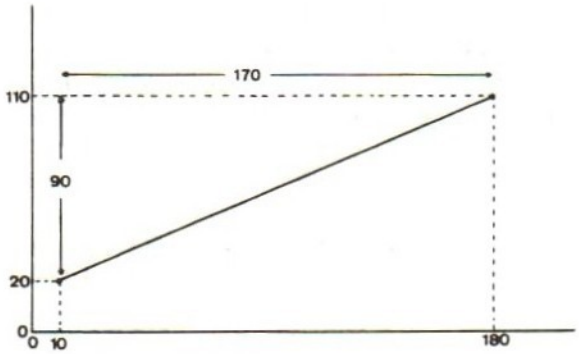
Op deze wijze wordt B$ uitgebreid met A$. Nadat de lijn 150 lichtpunten omhoog getrokken wordt, worden de deelopdrachten van A$ uitgevoerd. Daarna volgt de rest van B$ weer. Om een stringvariabele op deze wijze als deelopdracht te gebruiken moet u X plaatsen voor de variabelenaam en een puntkomma (;) achter de variabelenaam.
De volgende belangrijke deelopdracht is

M (Move = beweeg)

Deze deelopdracht trekt een lijn naar een punt dat aangegeven is door twee getallen achter de M. De beide getallen zijn de X-coordinaat en de Y-coordinaat. Allebei de coordinaten zijn absoluut. Dat wil zeggen dat er gerekend wordt vanaf de oorsprong (het punt 0,0). De beide getallen moeten gescheiden zijn door een komma, maar er hoeft niets te staan tussen de M en het eerste getal.

```
10 SCREEN 2
20 DRAW "M195,53"
30 GOTO 30
```

De eerste keer dat u dit programma runt ziet u een lijn. Hoe de lijn precies loopt is afhankelijk van de huidige positie van de grafische cursor. Deze wordt bepaald door het laatste gerunde programma. Runt u dit programma net nadat u de computer aangezet heeft, dan ziet u een lijn vanaf het punt (0,0). Welke lijn er ook getrokken wordt, hij eindigt altijd bij het punt (195,53). Als u het programma nu voor een tweede keer runt ziet u alleen een stip op deze positie. De grafische cursor staat immers al op de plaats waar een lijn naartoe getrokken moet worden.
Door de coordinaten van het punt te voorzien van een teken (een plus of min), wordt de lijn niet getrokken naar de absolute positie (X,Y), maar naar de relatieve positie. Dat wil zeggen dat de lijn getrokken wordt naar een punt dat X verder ligt en Y hoger of lager.
Door een teken te gebruiken geven de coordinaten een verplaatsing aan. Absolute coordinaten geven een plaats aan, relatieve coordinaten geven een verplaatsing aan. In de volgende tekeningen ziet u het verschil tussen een verplaatsing en een plaats.



absolute en relatieve verplaatsing

In programmavorm ziet het er zo uit. Kunt u op het beeldscherm zien wat de verplaatsing is en wat de plaats?

```
10 SCREEN 2
20 PSET (50,50)
30 DRAW "nl50nu50"
40 DRAW "c1m100,130"
45 DRAW "c15nl100nu130"
50 PSET (150,50)
60 DRAW "nr50nd80"
70 DRAW "c6m+50,+80"
80 GOTO 80
```

Het programma tekent twee lijnen. De ene lijn wordt aangegeven door de coordinaten van het beginpunt en het eindpunt (allebei weergegeven door witte lijnen naar de X-as en Y-as). De andere lijn wordt aangegeven door twee lijnen die de verplaatsing richting X en richting Y aangeven.
Naast de M deelopdracht kunt u in dit programma nog twee andere nieuwe deelopdrachten vinden.

C (Color = kleur)

De C deelopdracht bepaalt de kleur van de erna komende deelopdrachten. De C opdracht gebruikt dezelfde kleurcodes als de COLOR opdracht. In tegenstelling tot de COLOR opdracht, verandert de C deelopdracht de huidige voorgrondkleur niet.
De deelopdracht

N (Not = niet)

Deze deelopdracht laat de computer wel tekenen, maar verplaatst de grafische cursor niet. Deze opdracht moet telkens voor een

andere grafische deelopdracht geplaatst worden. Hij werkt alleen bij die deelopdracht waar hij onmiddellijk voor staat. Probeert u in bovenstaand programma de tweede N van elke DRAW opdracht te verwijderen en let op het verschil met de oorspronkelijke tekening.
N moet eigenlijk een voorvoegsel genoemd worden, want zonder een echte grafische deelopdracht erachter doet N niets.
N kan goed gebruikt worden voor tekeningen waar lijnen vanuit een bepaald punt getrokken moeten worden. Bijvoorbeeld een ster:

```
10 SCREEN 2
20 COLOR ,1,1 : CLS
30 A$="nu5nd5nr5nl5"
40 B$="ne3nf3ng3nh3"
50 X=INT(RND(1)*255)
60 Y=INT(RND(1)*255)
70 KL=INT(RND(1)*15)
80 SC=INT(RND(1)*10)
90 DRAW "bm=x;,=y;s=sc;c=kl;Xa$;Xb$;"
100 GOTO 50
```

In de regels 30 en 40 wordt de vorm van een sterretje gegeven. In regel 30 omhoog, omlaag, rechts en links. In regel 40 de verschillende diagonale kanten.
De regels 50-100 vormen een lus die deze ster telkens weer opnieuw tekenen, met telkens een andere maat, een andere kleur en op een andere plaats. De regels 50 en 60 zorgen voor een willekeurige plaats. Regel 70 zorgt voor een willekeurige kleur en regel 80 zorgt voor een willekeurige schaalfactor.
Regel 90 maakt gebruik van de willekeurige getallen zoals die in de regels daarvoor aan de variabelen zijn toegekend. De methode om een string binnen een DRAW string op te nemen kende u al: X ervoor en ; erachter. Een soortgelijk systeem bestaat voor het opnemen van een variabele in plaats van een getal. Daarvoor plaatst u voor de variabele een = en achter de variabelenaam een ;.
Een nieuwe deelopdracht in dit programma is

S (Scale = schaal)

Deze deelopdracht bepaalt de schaalfactor waarmee de deelopdrachten die
erachter staan vermenigvuldigd worden. Standaard is S=4 ingevuld. Dat wil zeggen dat als u achter S een getal kleiner dan 4 zet de tekeningen verkleind worden en dat als u achter S een getal groter dan 4 zet de tekeningen vergroot worden. Als u achter S een 4 of een 0 zet, krijgt u weer de normale grootte.

In regel 90 komt u ook een nieuw voorvoegsel tegen.

B (Blanco = leeg)

Dit voorvoegsel kan voor elke deelopdracht gezet worden en zorgt ervoor dat de erachter staande deelopdracht niet getekend wordt. De grafische cursor verplaatst zich echter wel. In samenhang met de M deelopdracht is dit voorvoegsel bijna gelijk aan de PSET opdracht.
De laatste deelopdracht die mogelijk is, is

A (Angle = hoek)

Deze opdracht zorgt ervoor dat u het coordinatenstelsel een kwart slag kan draaien. Achter A kunnen de volgende waarden ingevuld worden:

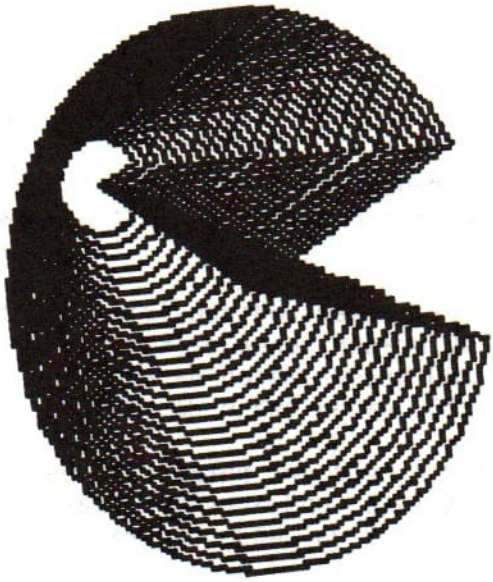
A0	terug naar beginpositie (0,0 is linksboven)
A1	90 graden tegen de klok in
A2	180 graden tegen de klok in
A3	270 graden tegen de klok in

De coordinaten blijven echter hetzelfde. De opdracht werkt dan ook niet goed bij absolute opdrachten, maar wel bij relatieve opdrachten. Probeert u het volgende programma uit.

```
10 SCREEN 2
20 COLOR ,1,1 : CLS
30 A$="bm=x;,=y;c=kl;s=sc;u5ng1nf1"
40 X=INT(RND(1)*255)
50 Y=INT(RND(1)*191)
60 KL=INT(RND(1)*15)
70 SC=INT(RND(1)*20)
80 DRAW "a2Xa$;"
90 GOTO 40
```

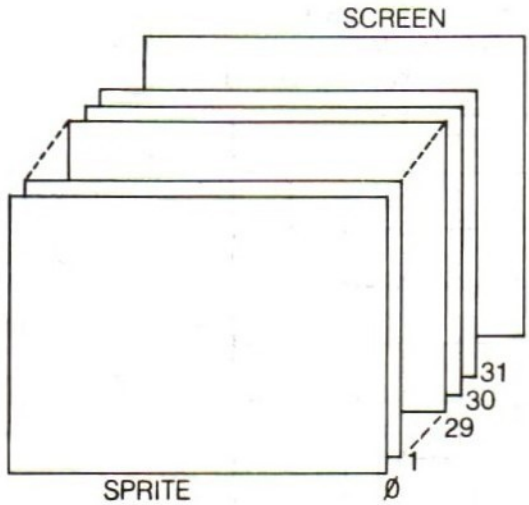
Het resultaat is een ware regen van pijlen. Door regel 80 te veranderen en regel 75 toe te voegen krijgt u een schat aan pijlen die alle kanten op staan.

```
75 RI=INT(RND(1)*4)
80 DRAW "A=RI;XA$;"
```

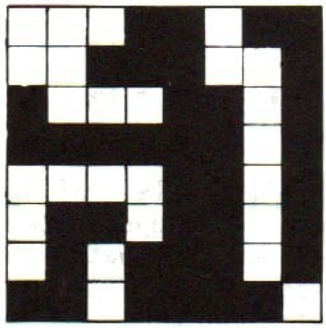


18. GRAFIEK: SPRITES

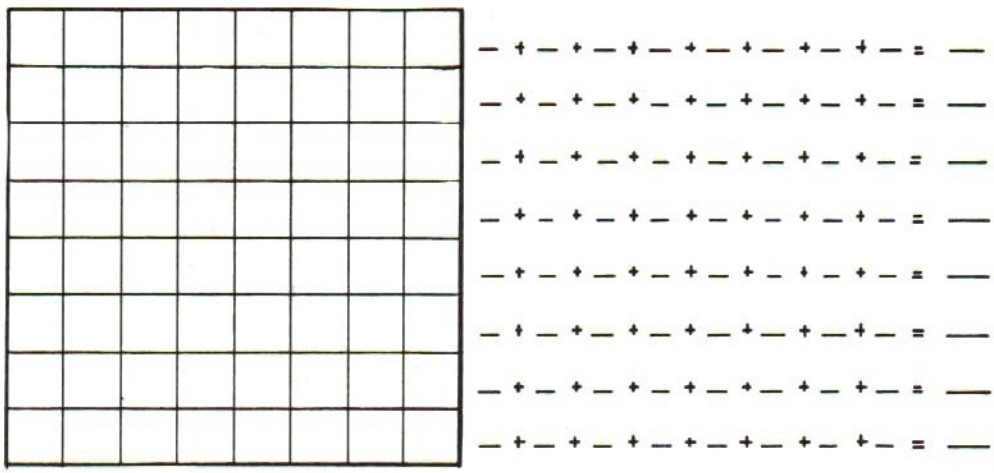
In de vorige hoofdstukken heeft u alle grafische opdrachten leren kennen die de MSX heeft. Naast deze opdrachten kunt u ook sprites gebruiken. Een sprite is een vast patroon lichtstippen in een rooster van 8 bij 8 of van 16 bij 16. Een sprite kan in verschillende maten op het scherm gezet worden en als geheel verplaatst worden.
Bij het werken met sprites moet u zich het beeldscherm voorstellen als bestaande uit een scherm met daarvoor 32 vellen doorzichtig tekenpapier. Op elk vel tekenpapier kan een sprite gemaakt worden. Doordat alle 32 vellen doorzichtig zijn, en de (gekleurde) sprites niet, kunt u een tekening maken met een hele serie sprites. Sprites die gedeeltelijk achter een andere sprite vallen worden niet getekend. Het effect is dat sprites op de bovenste vellen tekenpapier bovenop sprites van de onderliggende vellen getekend worden. De spritevlakken zijn genummerd van 0 (de voorste of bovenste) tot en met 31 (de achterste of onderste). Na alle spritevellen komt nog het normale beeldscherm. Zie onderstaande tekening.



U maakt een sprite door een rooster van 8 bij 8 vakjes te maken en daarin de vakjes in te kleuren die u op het scherm getekend wilt hebben. Als voorbeeld een zittend aapje.



U ziet rond dit aapje allerlei getallen staan. Het is de bedoeling dat elke rij van de gemaakte tekening apart bekeken wordt. Computertechnisch moet nu elk gekleurd vlakje gezien worden als een 1 en elk niet gekleurd vlakje als een 0. Het binaire getal dat dan voor elke rij gekregen wordt is het gegeven dat de computer kan gebruiken. U kunt het uzelf gemakkelijk maken. U hoeft alleen maar naar de 'waarde' van elk vlakje te krijgen. Zo is een vlakje helemaal links 128 waard. Het vlakje onmiddellijk rechts daarvan is 64 waard. Het vlakje daar weer rechts van is 32 waard, enzovoort tot het meest rechtse vlakje dat slechts 1 waard is. Door per rij de waarden van alle gekleurde vlakjes bij elkaar op te tellen krijgt u de totale waarde van elke rij in decimale notering. U kunt daarvoor een rooster als het onderstaande gebruiken:



128 64 32 16 8 4 2 1

Als voorbeeld bekijken we de bovenste rij van het aapje. Het meest linkse vlakje is niet gekleurd: geen 128. Het volgende vlakje is ook niet gekleurd: geen 64. Het daarop volgende vlakje is ook niet gekleurd: geen 32. Het vierde vlakje is echter wel gekleurd: 16. Het volgende hokje is ook weer gekleurd: 8. Het

hokje rechts ervan is leeg: geen 4. Het op een na laatste hokje is gekleurd: 2. En het laatste hokje is ook gekleurd: 1. Tellen we deze waarden bij elkaar op dan krijgen we 16+8+2+1=27. Op precies dezelfde wijze moeten alle rijen van een tekening bekeken en uitgerekend worden.
Nu heeft u op een gegeven moment een tekening gemaakt en daarvan de 8 waarden berekend (voor elke rij een waarde). Hoe leiden die waarden nu tot een tekening op het beeldscherm?
U begint met de computer te laten weten dat de waarden samen een sprite vormen:

```
SPRITE$( )=CHR$( )+CHR$( ).....+CHR$( )
```

Achter SPRITE$ komt het spritenummer dat u aan deze sprite wilt geven. In de rest van het programma wordt naar de sprite verwezen door dit nummer. Achter het is-gelijk-teken plaatst u 8 CHR$() funkties. Tussen de haakjes van deze 8 funkties plaatst u de acht waarden die u gevonden heeft bij het omzetten van de tekening. Kijkt u maar naar het volgende programma

```
10 SCREEN 2
  (253)+CHR$(13)+CHR$(109)+CHR$(93)+CHR$(222)
20 SPRITE$(1)= CHR$(27)+CHR$(57)+CHR$(141)+CHR$
30 PRESET(100,100)
40 PUT SPRITE 1,(100,100)
50 GOTO 50
```

In regel 20 wordt de sprite gedefinieerd. U komt daar alle getallen die hierboven bij de tekening van de sprite gemaakt zijn tegen.
Regel 30 plaatst de grafische cursor ergens midden op het scherm. Deze regel is eigenlijk overbodig. Als u de sprite niet op een speciale plaats zet, wordt de sprite daar geplaatst waar de grafische cursor zich bevindt. (de plaats van de grafische cursor wordt dan linksboven in de sprite).
In regel 40 wordt echter alsnog precies bepaald waar de sprite moet komen. Dit gaat met de opdracht

```
PUT SPRITE  ,( , ), ,                    (zet sprite)
```

Deze opdracht bepaalt op welk doorzichtige tekenvel (spritevlak) de sprite komt te staan, waar op dat vlak de sprite komt te staan, in welke kleur de sprite getekend wordt en om welke sprite het gaat.
Er zijn 32 spritevlakken genummerd van 0 (de voorste) tot en met 31 (de achterste). Sprites die getekend zijn op meer naar voren liggende vlakken (dat wil zeggen, vlakken met een lager nummer) worden over sprites op spritevlakken met een hoger nummer heen getekend. Per spritevlak kunt u slechts een sprite maken.

De plaats van de sprite wordt bepaald door de opgegeven coordinaten tussen de haakjes. Worden deze weggelaten dan wordt de sprite geplaatst op de huidige positie van de grafische cursor.
De kleur van de sprite wordt bepaald door de kleurcode die onmiddellijk na de coordinaten opgegeven wordt. De te gebruiken kleurcodes zijn gelijk aan die van de COLOR opdracht. Wordt er geen kleurcode opgegeven, dan neemt de computer aan dat de huidige voorgrondkleur bedoeld is.
Als laatste gegeven moet u aangeven om welke sprite het gaat, door het spritenummer op te geven als laatste getal. Laat u dit nummer weg dan gaat de computer ervan uit dat u de sprite bedoelde met het nummer gelijk aan het nummer van het spritevlak.

Extra vermelding verdient de mogelijkheid om in plaats van de CHR$() opdracht in regel 20 de letter op te geven die bij de bedoelde code hoort. U kunt regel 20 vervangen door:

```
20 SPRITE$(1)=CHR$(27)+CHR$(57)+CHR$(141)+CHR$(253)
 +CHR$(13)+"m"+"]"+CHR$(222)
```

Welke letters bij welke codes horen kunt u opzoeken in de ASCII-tabel (zie bijlage).

Spritevlakken

Om de werking van meer dan een spritevlak te laten zien, tekent het volgende programma drie verschillende sprites in drie kleuren. Doordat de sprites telkens op een andere plaats gezet worden, zullen ze elkaar op een gegeven moment kruisen. De sprite op het vlak met het laagste nummer zal daarbij over de andere sprites heenglijden.

```
10 SCREEN 2,1
20 COLOR 4,10,10 : CLS
30 LINE (0,0)-(255,191),4
40 LINE (0,191)-(255,0),8
50 SPRITE$(1)= CHR$(27)+CHR$(57)+CHR$(141)+CHR$(253)+CHR$(13)+"m"+"0"+CHR$(222)
60 SPRITE$(2)=CHR$(1)+CHR$(3)+CHR$(7)+CHR$(15)+CHR$(27)+CHR$(63)+CHR$(127)+CHR$(255)
70 SPRITE$(3)=CHR$(16)+CHR$(48)+CHR$(112)+CHR$(255)+CHR$(255)+CHR$(112)+CHR$(48)+CHR$(16)
80 PUT SPRITE 1,(120,100),1,1
90 PUT SPRITE 2,(10,100),9,2
100 PUT SPRITE 3,(240,100),15,3
110 FOR Z=1 TO 300 : NEXT Z
120 FOR X=1 TO 650
130 PUT SPRITE 1,(120-X/10,100),1,1
140 PUT SPRITE 2,(10+X*1.5,100),9,2
```

```
150 PUT SPRITE 3,(240-X,100),15,3
160 FOR Z=1 TO 25 : NEXT Z
170 NEXT X
180 GOTO 180
```

Behalve het effect van over elkaar heen glijden van de verschillende sprites biedt dit programma nog enkel andere wetenswaardige punten.
In de regels 20-40 wordt het scherm van kleur voorzien en worden er twee gekleurde lijnen over het scherm getrokken met behulp van de normale grafische opdrachten. Deze lijnen worden geheel niet beinvloed door de sprite opdrachten en blijven zichtbaar. De spritevlakken zijn volkomen doorzichtig.
In regel 10 komt u een oude opdracht in een nieuw jasje tegen. De opdracht

SCREEN

kende u al. Maar u gebruikte daarbij altijd slechts een getal dat de schermmodus aangeeft. Het tweede getal achter SCREEN geeft de opmaak van de sprites weer. U heeft hierbij de keuze uit vier getallen:

SCREEN , 0	8X8 sprites, niet vergroot
SCREEN , 1	8X8 sprites, wel vergroot
SCREEN , 2	16X16 sprites, niet vergroot
SCREEN , 3	16X16 sprites, wel vergroot

Het vergroten van de sprites wil zeggen dat de sprites zowel in de hoogte als in de breedte met een faktor twee worden vergroot. In bovenstaand programma wordt gewerkt met 8X8 sprites die vergroot zijn.
Naast deze twee veel gebruikte funkties van de opdracht SCREEN zijn er nog drie andere funkties die minder vaak van pas zullen komen, maar die we hier volledigheidshalve wel willen noemen:

SCREEN modus, spritevorm, toetssignaal,baud-snelheid, printer

Het derde getal dat ingevuld kan worden achter SCREEN bepaalt of de toetsen van het toetsenbord een korte tik laten horen bij het indrukken. Als u hier niets invult blijft de situatie zoals hij was voor de SCREEN opdrachten. Onmiddellijk na het aanzetten van de computer geeft het indrukken van een toets een korte tik. Als u een 0 invult, wordt het toetsenbord geruisloos. Als u een getal tussen 1 en 255 invult gaat het toetsenbord tikken.
Het vierde getal dat opgegeven kan worden bepaalt de snelheid waarmee gegevens naar de cassetterecorder worden weggeschreven of van de cassetterecorder worden ingelezen. De snelheid wordt

uitgedrukt in 'baud'. 10 baud is 1 teken per seconde. Vult u hier 1 in dan worden de gegevens verstuurd of ontvangen met 1200 baud (standaardwaarde). Vult u hier 2 in dan worden de gegevens doorgegeven met 2400 baud. Sommige commerciele programmacassettes zijn opgenomen op 2400 baud. U moet dan eerst door middel van de SCREEN opdracht de juiste baudsnelheid instellen voordat u kunt gaan laden.
Het laatste getal geeft aan wat voor soort printer u gebruikt. Gebruikt u een echte MSX-printer dan worden alle symbolen van de MSX op de juiste manier op papier gezet. Printers die niet aan de MSX standaard voldoen kunnen deze symbolen niet allemaal afdrukken. Als u een getal van 1 tot en met 255 invult, drukt de computer in plaats van de vreemde symbolen telkens spaties af. Invullen van 0 zorgt ervoor dat de speciale MSX symbolen wel afgedrukt worden.

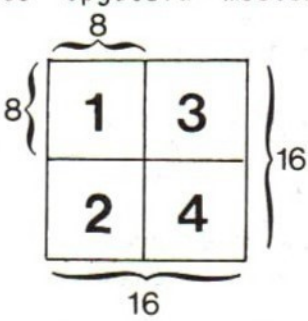
Regel 110 in het programma zorgt voor een vertraging.
Regel 160 zorgt voor een lange lus.
De regels 130-150 zorgen voor het plaatsen van de sprites. Elke keer dat de lus doorlopen wordt, worden de sprites opnieuw geplaatst. Omdat er slechts een sprite per vlak mogelijk is, hoeft de programmeur niet te letten op het uitvegen van de 'oude' sprites bij het bewegen.
Regel 160 zorgt voor een vertraging. Zonder deze vertraging zouden de sprites sneller over het scherm gaan.

Er zijn 32 spritevlakken. U kunt echter maximaal 256 sprites van 8X8 of 64 sprites van 16X16 maken. Deze kunnen dan beurtelings op een spritevlak gezet worden. Zo'n grote hoeveelheid sprites kost echter de nodige geheugenruimte, dus de rest van het programma kan dan niet meer erg lang zijn.

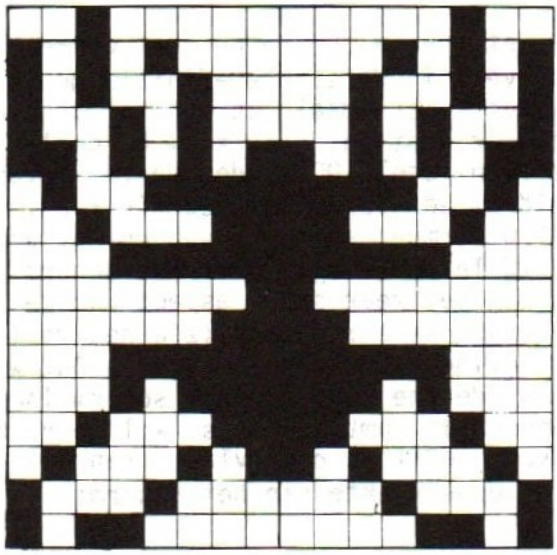
Grote sprites

Naast de sprites van 8X8 is het ook mogelijk om sprites van 16X16 lichtpunten te maken. Dit zijn in feite optelsommen van vier kleine sprites. De volgorde waarin de sprites opgeteld moeten worden is:

1 = linksboven
2 = linksonder
3 = rechtsboven
4 = rechtsonder



Als voorbeeld een vogelspin:

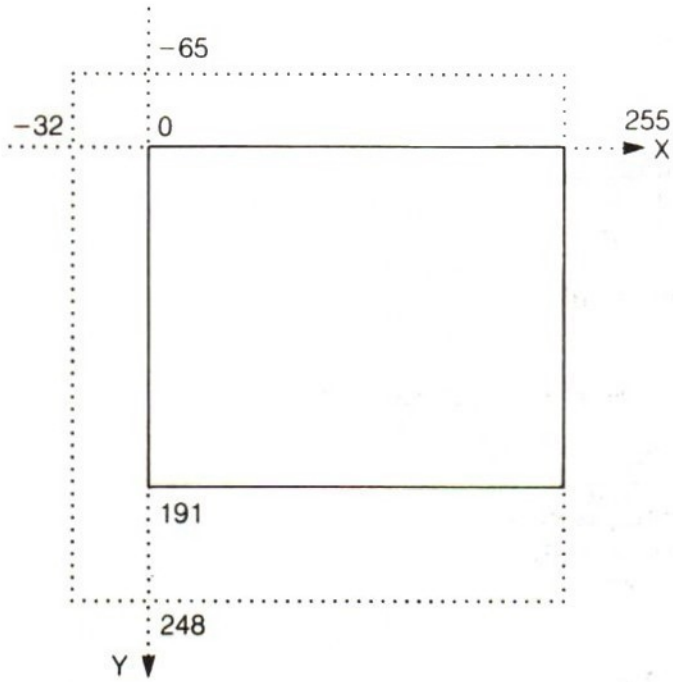


Het programma waarmee u deze vogelspin over het scherm laat lopen is:

```
10 SCREEN 2,3
20 DIM S$(4)
30 COLOR 15,12,12
40 FOR A=1 TO 4
50 FOR B=1 TO 8
60 READ G
70 S$(A)=S$(A)+CHR$(G)
80 NEXT B
90 NEXT A
100 SPRITE$(1)=S$(1)+S$(2)+S$(3)+S$(4)
110 REM
120 PSET (125,80)
130 FOR Z=1 TO 50
140 X=INT(RND(1)*255)
150 Y=INT(RND(1)*191)
160 DRAW "nm=x;,=y;"
170 NEXT Z
180 REM
190 FOR Y=1 TO 700 STEP .5
200 PUT SPRITE 1,(125,190-Y),1,1
210 NEXT Y
220 GOTO 220
230 DATA 32,168,164,148,213,75,35,31
240 DATA 1,3,31,23,35,69,136,176
250 DATA 4,21,37,41,171,242,196,248
260 DATA 128,192,248,232,196,162,17,13
```

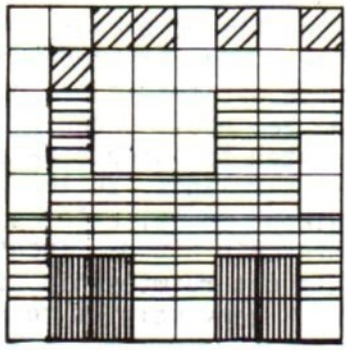
Door dit programma loopt de spin, die gemaakt is uit vier

sprites, over het scherm. Let u op de wijze waarop de grote hoeveelheid gegevens in de spritedefinitie geplaatst wordt. In de regel 40-90 wordt vier maal een achttal gegevens gelezen. Deze gegevens worden per acht in een string geplaatst met de nodige CHR$ opdrachten. In regel 100 worden de zo gevormde strings aan elkaar gekoppeld om een lange spritedefinitie te vormen.
De regels 120-170 vormen een spinneweb op het achtergrondscherm.
De regels 190-210 laten de spin over het scherm lopen.
U ziet hierbij nog een zeer sterk aspect van de sprites bij de MSX. Zowel in horizontale richting (zie het vorige programma) als in vertikale richting is het beeld 'rond'. Dat wil zeggen dat een sprite die aan de ene kant van het scherm loopt, aan de andere kant weer tevoorschijn komt. Er is wel een kleine pauze te bemerken. Dat komt omdat de oppervlakte van de spritevlakken iets groter is dan de oppervlakte van het scherm:



Sprites met meer kleuren

U las hiervoor dat een sprite maar een kleur kan hebben. Dat klopt. Om toch een sprite met meer dan een kleur te krijgen kunt u meer dan een sprite maken en die over elkaar laten vallen. De achtergrond van de sprite is immers altijd transparant. Als voorbeeld een treintje.



Om het treintje te laten rijden runt u het volgende programma.

```
10 SCREEN 2,1
20 DIM S$(3)
30 COLOR 15,1,1
40 FOR A=1 TO 3
50 FOR B=1 TO 8
60 READ G
70 S$(A)=S$(A)+CHR$(G)
80 NEXT B
90 SPRITE$(A)=S$(A)
100 NEXT A
110 SPRITE$(1)=S$(1)+S$(2)+S$(3)
120 REM
130 LINE (0,80)-(255,190),10,BF
140 LINE (0,0)-(255,79),7,BF
150 FOR X=1 TO 900
160 IF INT(X/5)=X/5 THEN PUT SPRITE 1,(255-X,64),15,1
170 PUT SPRITE 2,(255-X,64),12,2
180 PUT SPRITE 3,(255-X,64),9,3
190 NEXT X
200 GOTO 200
210 DATA 53,64,0,0,0,0,0,0
220 DATA 0,0,71,70,126,254,153,0
230 DATA 0,0,0,0,0,0,102,102
```

Dit programma is grotendeels gelijk aan het vorige. Let u op regel 160. Door gebruik te maken van de voorwaarde IF INT(X/5)=INT/5, wordt de rook van het treintje slechts een op de vijf keer verplaatst. Dit doet het treintje nog veel echter lijken.

Botsingen en verbergen van sprites

Het kan gebeuren dat twee sprites op elkaar botsen. Of liever gezegd; het oog ziet dat twee sprites tegen elkaar aan zullen gaan. In werkelijkheid zijn de sprites ieder gebonden aan hun eigen spritevlak en botsen ze nooit tegen elkaar aan. Het kan echter handig zijn om de computer te laten bepalen op welk moment precies twee sprites onder elkaar langs zullen glijden. U kunt die wetenschap goed gebruiken in allerlei spelletjes waarbij bijvoorbeeld een vreemd ruimteschip uit de lucht geschoten moet worden. Als zowel de kogel als het ruimteschip een sprite zijn, is het met de volgende techniek eenvoudig om te kijken wanneer er van een treffer sprake is.
Voor het ontdekken van een botsing gebruikt u

ON SPRITE GOSUB

U kent deze opdracht al uit de serie ON....opdrachten. Deze opdracht moet dan ook in werking gesteld worden met

SPRITE ON

Om te zorgen dat de subroutine niet continue aangeroepen wordt (de sprites bevinden zich gedurende langere tijd boven elkaar) is de eerste opdracht van de subroutine het weer afzetten van de spritebotsingbepaler (SPRITE OFF).
Omdat de ON SPRITE GOSUB opdracht niet kan bepalen waar welke sprites met elkaar in botsing komen, maar alleen dat er twee sprites met elkaar in botsing komen, moet in de rest van het programma opdrachten worden ingebouwd die bepalen waar de botsing plaats heeft gevonden. In het volgende programma zijn de kleuren van het treintje weer tijdelijk verwijderd. De kleuren werden immers gemaakt door drie sprites boven op elkaar te tekenen. De ON SPRITE GOSUB opdracht zou niet telkens opnieuw een botsing in zien.
Het programma toont u een ernstig ongeluk waarbij een vliegtuig en een trein betrokken zijn.

```
10 ON SPRITE GOSUB 280
20 SCREEN 2,1
30 DIM S$(3)
40 COLOR 15,1,1
50 FOR A=1 TO 3
60 FOR B=1 TO 8
70 READ G
80 S$(A)=S$(A)+CHR$(G)
90 NEXT B
100 SPRITE$(A)=S$(A)
110 NEXT A
```

```
120 SPRITE$(1)=S$(1)+S$(2)+S$(3)
130 REM
140 LINE (0,80)-(255,190),10,BF
150 LINE (0,0)-(255,79),7,BF
160 SPRITE ON
170 FOR X=1 TO 900 STEP 2
180 PUT SPRITE 1,(255-X,64),1,1
190 PUT SPRITE 2,(100-X*5,10+X/10),1,2
200 PUT SPRITE 3,(116-X*5,10+X/10),1,3
210 FOR Z=1 TO 40 : NEXT Z
220 NEXT X
230 GOTO 230
240 DATA 53,64,71,70,126,254,157,102
250 DATA 0,1,3,127,255,7,3,1
260 DATA 2,199,143,255,248,0,128,192
270 REM botsingsroutine
280 SPRITE OFF
290 KEY OFF
300 FOR Q=1 TO 14 : BEEP : BEEP : SCREEN 1 : COLOR ,Q,Q+1 :NEXT Q
310 END
```

Het verbergen van sprites gaat door middel van het opgeven van een speciale Y-waarde als coordinaat. Geeft u de waarde 208 als Y-coordinaat dan wordt de sprite op het betreffende vlak verwijderd en worden alle sprites op spritevlakken met een hoger vlaknummer verwijderd.
Als u als waarde voor de Y-coordinaat 209 opgeeft, wordt alleen het betreffend spritevlak gewist. Verandert u in bovenstaand programma de regels 300 en 310.

```
300 PUT SPRITE 1,(255-X,209),1,1
310 GOTO 310
```

Nu zal bij de botsing de trein in rook opgaan en het vliegtuig zich van angst niet meer kunnen verroeren. Verandert u regel 300 in

```
300 PUT SPRITE 1,(255-X,208),1,1
```

dan verdwijnen beide voertuigen onmiddellijk uit beeld.

Vier sprites en beweging

Er is een beperking aan het gebruik van sprites. Op een willekeurige horizontale lijn kunt u nooit meer dan vier sprites plaatsen. Als u toch meer sprites plaatst, wordt de rest gewoon niet getekend.

Het is zeer goed mogelijk met sprites bewegende voorwerpen te maken. Als voorbeeld nemen we het aapje uit het begin van het hoofdstuk en laten dat over het scherm springen.

```
10 SCREEN 2,1
20 DIM S$(3)
30 COLOR 15,1,1
40 FOR A=1 TO 3
50 FOR B=1 TO 8
60 READ G
70 S$(A)=S$(A)+CHR$(G)
80 NEXT B
90 SPRITE$(A)=S$(A)
100 NEXT A
110 FOR Y=174 TO 0 STEP -2
120 IF INT(Y/12)=Y/12 THEN PUT SPRITE 2,(X,208),4,2 :
    PUT SPRITE 1,(100+Y/2,Y),4,1 : FOR Z=1 TO 150 : NEXT Z
130 PUT SPRITE 1,(100+Y/2,209),4,1
140 PUT SPRITE 2,(100+Y/2,Y),4,2
150 PUT SPRITE 3,(100+Y/2,Y+16),4,3
160 NEXT Y
170 GOTO 170
180 DATA 27,57,141,253,13,109,93,222
190 DATA 27,57,13,253,141,13,13,14
200 DATA 16,16,48,32,32,96,0,0
```

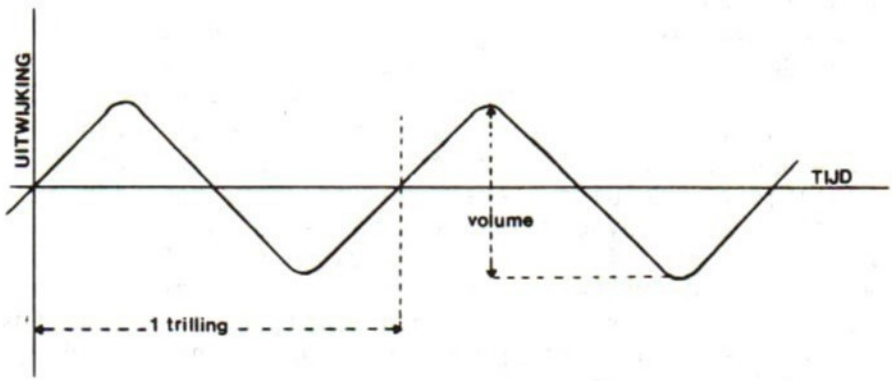
19. GELUID EN MUZIEK

Zoals een schrijfmachine een bel heeft om de aandacht voor het einde van de regel te trekken, zo hebben computers al heel lang een eenvoudige geluidsmogelijkheid voor een 'beep'. Deze bliep is altijd bedoeld geweest om de aandacht te trekken van de gebruiker.
Met de opkomst van de spelletjes bleek dat zo'n eenvoudige piep niet voldoende was. Tegenwoordig beschikken de meeste microcomputers over een regelbare bliep, zodat eenvoudige geluiden (een-stems) gemaakt kunnen worden.
De MSX computers hebben voor het maken van geluid een speciale eenheid de zogenaamde PSG (Programmable Sound Generator = programmeerbare geluidsgenerator) nummer AY-3-8912, ontwikkeld door General Instruments in Amerika. Het is wellicht de meest populaire geluidschip die er bij er voor kleine microcomputers op de markt is. Deze geluidschip bevat drie verschillende geluidsgeneratoren (zeer fraaie piepers). Met deze drie zeer veelzijdige geluidsgeneratoren is het mogelijk om een haast onbegrensde hoeveelheid verschillende geluiden te maken. De mogelijkheden zijn zo groot dat de MSX niet alleen op verschillende manieren kan piepen, maar zeer fraaie muziek kan voortbrengen met drie stemmen tegelijk. In dit hoofdstuk zullen alleen de programmeerprincipes besproken worden, uiteraard vergezeld van de nodige voorbeelden. Omdat muziek zeer smaakgebonden is en het aantal mogelijkheden van de MSX te groot is om in een dik boek te zetten, zult u zelf door middel van experimenteren de leukste geluiden moeten zien te vinden. Waar mogelijk is aangegeven welke experimenten u kunt gaan doen. Speciale experimenteerprogramma's zijn opgenomen.

Geluid

Geluid is het trillen van lucht. Dit trillen kan veroorzaakt worden door het trillen van een snaar (eventueel via een klankkast), het trillen van het vel van een trommel, het trillen van een rietje (in een fagot bijvoorbeeld) of op vele andere manieren. Het trillen van de lucht moet niet opgevat worden als op en neer trillen, maar als een opeenvolging van verdichtingen en verdunningen van de lucht. De lucht wordt achtereenvolgens even in elkaar geduwd en uit elkaar getrokken.
Op papier worden trillingen meestal weergegeven als een golf. Deze vorm is het resultaat van het tegen elkaar uitzetten van tijd en amplitude. De amplitude is het verschil tussen een verdichting en een verdunning. Er zijn twee basisgrootheden bij

geluid: het volume en de frequentie van de trilling. De sterkte van het volume is hoorbaar. Op papier is het volume af te lezen aan de hoogte van de golf. Hoe hoger de golf hoe harder het geluid.
De frequentie is ook hoorbaar. Het zegt iets over de toonhoogte van het geluid. Hoe groter de frequentie, hoe hoger het geluid klinkt. De frequentie geeft aan hoe vaak een verdichting en verdunning in een seconde voorkomen. Men spreekt ook wel over het aantal trillingen per seconde. Dit wordt aangegeven in Hertz (Hz = aantal trillingen per seconde). Het menselijk oor kan geluiden van 30 Hz (een zeer lage brom) tot ongeveer 15000 Hz (een zeer hoge piep) horen. Het niveau van spreken en muziek ligt ongeveer tussen 100 en 4000 Hz.

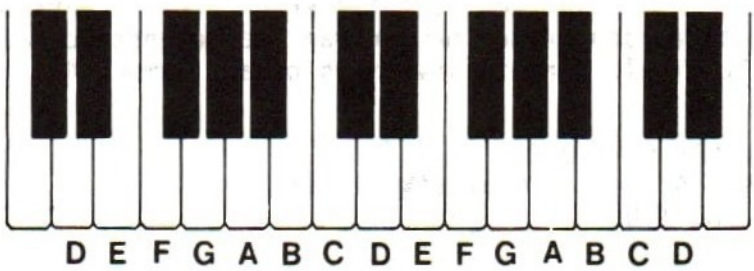


De BASIC besturing

Het maken van geluid door de MSX kan door de gebruiker bepaald worden door twee opdrachten: PLAY en SOUND.
De PLAY opdracht is voornamelijk ontworpen om muziek te maken. Als basis voor de opdracht is het gebruik van een piano genomen. De opdracht werkt op vergelijkbare wijze als de DRAW opdracht. Dat wil zeggen dat de opdracht gevolgd wordt door een string met daarin een aantal deelopdrachten. Doordat de gebruiker de keuze heeft uit een keur van deelopdrachten kan elke gewenste muziek gespeeld worden.
De SOUND opdracht is meer ontworpen voor het maken van geluidsopdrachten. Deze opdracht werkt veel direkter op de geluidsgenerator. Het geluid kan nog individueler bepaald worden. Het is wel mogelijk om met de SOUND opdracht dezelfde effecten te bereiken als met de PLAY opdracht, maar dat vergt veel meer werk. Het is daarom beter om de ruwe onderverdeling aan te houden en de SOUND opdracht alleen te gebruiken voor het maken van speciale geluidseffecten die niet mogelijk zijn met de PLAY opdracht. U zult daarbij zien dat er enkele overlappingen tussen de twee opdrachten zijn.

De PLAY opdracht

De PLAY opdracht is bedoeld om de MSX muziek te laten maken. U moet daarbij bedenken dat een deel van de weergave van de muziek afhankelijk is van uw televisie of monitor. De MSX heeft geen eigen luidspreker, maar gebruikt die van uw televisie of monitor. Vindt u een bepaald geluid te hard of te zacht dan kunt u dat opvangen door de opdrachten te veranderen, maar ook door het volume van de televisie bij te stellen. Dit is mede afhankelijk va het feit of u het televisiegeluid normaal gesproken aan heeft staan voor de klikken van het toetsenbord en de weergave van de blieps.
Het toetsenbord van een piano heeft als basis gediend bij het ontwerpen van de PLAY opdracht. Bijna elk stuk dat u op een piano kunt spelen, kunt u ook door de MSX laten spelen. Daarnaast kan de MSX veel meer. Het is raadzaam om bij de bespreking van de verschilende deelopdrachten telkens het gebruik van een piano voor ogen te houden. Nog mooier is het als u een piano bij de hand heeft. U kunt verschillende deelopdrachten dan vergelijken met een echte piano.



toetsenbord piano

De toetsen van een piano geven verschillende noten aan, gegroepeerd in octaven. Een octaaf bestaat uit 8 hele noten die C, D, E, F, G, A en B genoemd worden. De eerstvolgende noot na dit rijtje is de C van een octaaf hoger. Daarnaast kent de piano nog zwarte toetsen die halve noten aangeven (op papier aangegeven met #). Een octaaf kent dus: C, C#, D, D#, E, F, F#, G, G#, A, A#, B.
De play opdracht heeft het toonbereik op precies dezelfde wijze aangegeven. Om een bepaalde noot te spelen geeft u de MSX gewoon de gewenste noot op!

```
PLAY "A"
```

Als u dit intikt en op RETURN drukt hoort u via uw televisie een toon spelen. Dit is de A van de middelste octaaf van de piano. De zwarte toetsen op de piano (de halve noten) kunt u aangeven door een # teken, een + teken of een - teken. Een # of + duidt op een toon die een halve noot hoger is dan de aangegeven noot. Een - duidt op een halve noot lager. Het is niet mogelijk om halve noten aan te geven die niet op de piano bestaan.

```
PLAY "C#"
PLAY "C+"
PLAY "D-"
```

Deze drie opdrachten laten precies hetzelfde horen.
Om alle noten van een octaaf te horen kunt u dus de opdracht geven:

```
PLAY "CC#DD#EFF#GG#AA#B"
```

De MSX kent net als de meeste piano's acht octaven. Deze geeft u aan door middel van een hoofdletter O (niet verwarren met een nul - 0) met daarachter het nummer van de gewenste octaaf. De octaaf die u hoort als u niets invult is octaaf nummer 4.

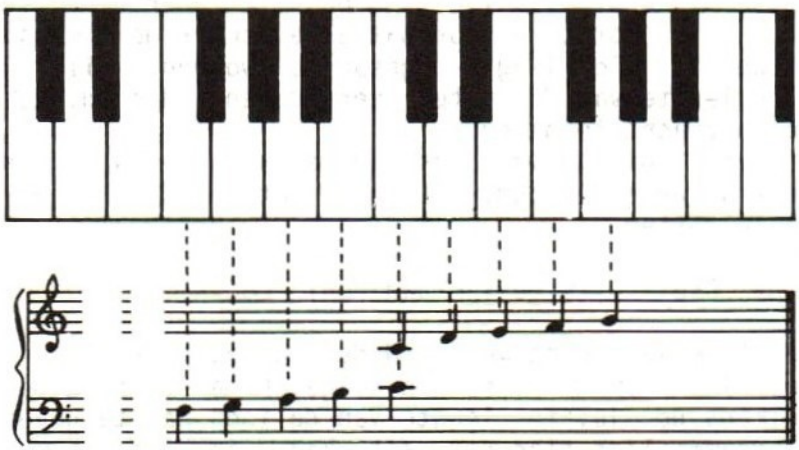
```
10 FOR OC=1TO 8
20 PLAY"o=oc;cc#dd#eff#gg#aa#b"
30 NEXT OC
```

Er is nog een andere methode om de noten aan te geven. Alle noten zijn genummerd. U kunt een noot met zijn nummer aanduiden door er een N voor te zetten. N kan lopen van 0 tot en met 96. Er zijn echter in 8 octaven van elk 12 noten slechts 96 noten. Met de N deelopdracht heeft u de beschikking over 1 noot extra: de rust. Als u achter N de waarde 0 invult hoort u tijdelijk niets. N1 is gelijk aan C# in de eerste octaaf (O1). N96 is de C in de negende octaaf (niet verkrijgbaar met de O deelopdracht). O1C (de C in de eerste octaaf) is niet verkrijgbaar met de N deelopdracht.

```
10 FOR NO=0 TO 96
20 PLAY "N=no;"
30 NEXT NO
```

Dit programma geeft hetzelfde resultaat als het vorige. Doordat de PLAY opdrachten allemaal gescheiden zijn, is de weergave van

de verschillende tonen een beetje gescheiden. U zult merken dat de allerhoogste tonen niet echt fraai meer klinken. De frequentie daarvan is te hoog.



toetsenbord

Nu de verschillende noten vastliggen is het mogelijk om muziek te maken. Een eenvoudige manier is het maken van een lijstje met waarden die de MSX moet afwerken. Door de lijst zorgvuldig samen te stellen kan een wijsje gemaakt worden.

```
10 FOR A=1 TO 16
20 READ NO
30 PLAY "N=no;"
40 NEXT A
50 DATA 36,38,40,38,36,40,38,36,36,35,33,35,36,33,35,36
```

Het is weliswaar leuk om een computer een wijsje te horen spelen, maar het klinkt niet beter dan een beginnende pianoleerling bij les 1.
Heel snel al leert elke muziekmaker dat niet elke noot even lang moet klinken. Noten worden onderverdeeld in:

L1	hele noten
L2	halve noten
L4	kwart noten
L8	achtste noten
L16	zestiende noten
L32	tweeendertigste noten

De lengte van de noten wordt aangegeven door de deelopdracht L. In het lijstje hierboven ziet u dat L1 zorgt voor hele noten, L2 voor halve noten, etc. De hoogste waarde achter L is 64. U kunt ook tussenliggende nootlengtes gebruiken. Zo zorgt L5 voor noten met een lengte van een vijfde, dit is echter minder gebruikelijk. De deelopdracht L zorgt ervoor dat alle achter de deelopdracht staande noten in die lengte gespeeld worden. Wilt u dus halverwege de lengte van de noten veranderen, dan dient u een nieuwe L deelopdracht te geven.
De lengte van individuele noten kan ook aangegeven worden door het getal dat u anders achter L zou zetten, nu achter de te spelen toon te zetten.

PLAY "C#8" is gelijk aan PLAY "L8C#"

Dit gaat niet in samenhang met de N deelopdracht. Wilt u de N opdracht gebruiken om de noten aan te geven dan dient u met de L opdracht telkens de juiste lengte van de toon aan te geven. Het volgende programma laat zien hoe dit kan. In dit programma is weer gebruik gemaakt van de N deelopdracht om de verschillende noten aan te geven. Het voordeel hiervan is dat de verschillende noten als cijfers in een datalijst gezet kunnen worden.

De lengte van de tonen wordt bij elke noot opnieuw met een L deelopdracht opgegeven. Dat lijkt omslachtig, maar ook nu is het voordeel dat de gegevens als getallen opgeslagen kunnen worden.

```
10 PLAY "T250"
20 FOR Z=1 TO 2
30 FOR A=1 TO 43
40 READ NO,LE
50 PLAY "L=le;N=no;"
60 NEXT A
70 LE=LE/2
80 PLAY "l=le;N=no;"
90 RESTORE
100 NEXT Z
110 DATA 36,2,38,4,40,2,38,2,43,2,31,2,31,1,33,2,35,4,36,2,35,2
120 DATA 36,2,40,2,38,1,35,2,36,4,38,2,35,2,38,2,40,2,41,2,38,2,31,2,31,2,36,2,38,2
130 DATA 40,2,43,2,43,2,41,2,40,2,38,2,36,2,31,2,40,2,38,4,36,1,36,
    2,36,2,45,2,45,2,43,2,35,2,36,1
```

Net als bij de DRAW deelopdrachten moeten variabelen die binnen de string van de PLAY opdracht gebruikt worden, voorafgegaan worden door een is-gelijk-teken en gevolgd worden door een puntkomma.
In de eerste regel van bovenstaand programma komt u een nieuwe deelopdracht tegen. De T. Dit staat voor Tempo en deze deelopdracht bepaalt de snelheid waarmee de muziek gespeeld wordt. T kan elke gehele waarde tussen 32 en 255 achter zich

krijgen. 32 is langzaam, 255 is snel. Uiteraard is de snelheid van het muziekstuk mede afhankelijk van de aanwezigeheid van hele of halve noten.
Net als de O en de L deelopdracht hoeft u de T opdracht maar een keer op te geven waarna dat tempo blijft gelden totdat de gebruiker een nieuwe T deelopdracht geeft.
Herkende u het gespeelde wijsje?

Een deelopdracht die voor wat rust zorgt in een muziekstuk is de R. Dit staat voor Rest (rust). De getallen die achter de R komen te staan zijn gelijk aan die achter de L kunnen staan en geven daarmee een rust die gelijk is aan een hele, halve, achtste, etc. noot.
Een extra mogelijkheid bij het bepalen van de lengte van een toon is de punt (.). Als u een punt achter een bepaalde noot zet, wordt deze met de helft verlengd. U kunt een punt ook achter een rust zetten.

Het volume van de gespeelde muziek is op twee manieren te beinvloeden: door de volumeknop van de televisie en door de deelopdracht V.
Achter de V van Volume zet u een getal tussen 0 en 15. De beginstand is 8.

Alle hierboven genoemde deelopdrachten van de PLAY opdracht bepalen welk geluid de computer via de televisie laat horen. Maar van alle geluiden kon gezegd worden dat ze nogal 'computerig' klonken. De laatste twee deelopdrachten van PLAY zijn bedoeld om de computer een bepaald instrument te laten imiteren. Daarnaast kunt u met deze opdrachten het geluid van de MSX nog veel minder op een al bestaand instrument laten lijken, zodat u allemaal synthetische geluiden krijgt.
De eerste van de twee opdrachten is de S deelopdracht. S staat voor Sound (geluid). De SOUND opdracht is gedeeltelijk gelijk aan deze deelopdracht. Door deze deelopdracht geeft u de computer opdracht om bij het laten klinken van het geluid een bepaald golfvormpatroon of omhullende te gebruiken. Dat wil zeggen dat het volume van de toon niet gedurende de gehele tijd dat de toon klinkt gelijk is, maar verloopt volgens een bepaald patroon. De MSX biedt de gebruiker de keuze uit 8 verschillende golfpatronen (Engels = envelope).

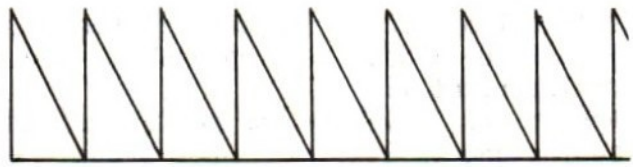
S = 0, 1, 2, 3 of 9

Dit patroon zorgt voor een zeer snel tot volledige hardheid komende toon die daarna geleidelijk afzwakt.

S = 4, 5, 6, 7 of 15

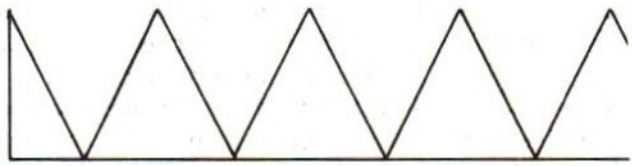
Dit patroon zorgt voor een toon die langzaam tot volledig volume klimt en daarna abrupt stopt.

S = 8



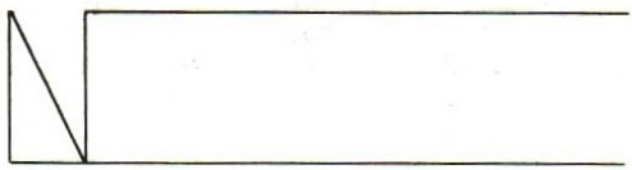
Dit patroon zorgt voor een toon die telkens zeer snel tot volle geluidssterkte toeneemt en daarna geleidelijk in hardheid daalt. Dit wordt herhaald.

S = 10



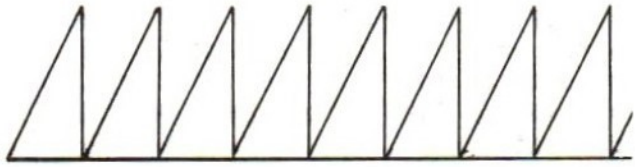
Dit patroon zorgt voor een toon die zeer snel tot volle geluidssterkte groeit en vervolgens steeds opnieuw geleidelijk in sterkte daalt en weer stijgt.

S = 11



Dit patroon zorgt voor een toon die eerst zeer snel tot volle geluidssterkte groeit, vervolgens langzaam daalt, weer zeer snel tot volle sterkte groeit en daarna op volle sterkte blijft.

S = 12

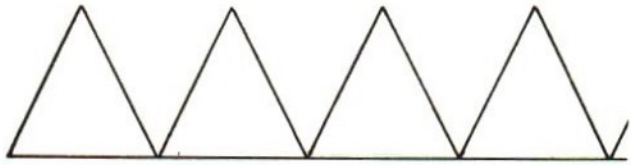


Dit patroon zorgt voor een herhaling van een langzaam stijgend en zeer snel dalend volume.

S = 13

Dit patroon zorgt voor een geleidelijk in volume stijgende toon die daarna op volle sterkte blijft.

S = 14



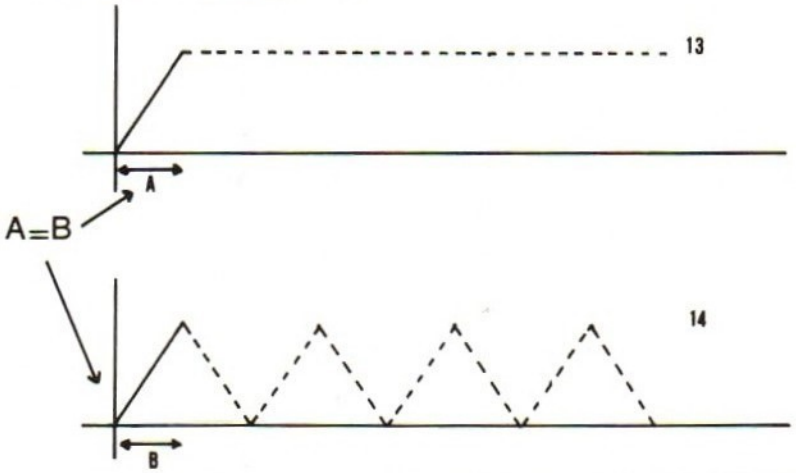
Dit patroon zorgt voor een herhaling van een in volume langzaam stijgende en weer langzaam dalende toon.

Het onderstaande programma laat u het effect van de verschillende golfpatronen op de twaalf noten van een octaaf horen. Merkt u op dat veel golfpatronen hetzelfde zijn. Er zijn 16 mogelijk in te vullen getallen, maar 'slechts' acht verschillende golfpatronen. Als u zelf geen waarde ingevuld heeft kiest de MSX golfvorm 13.

```
10 PLAY "t90m700011"
20 FOR GO=0 TO 10 STEP 2
25 PRINT : PRINT "golfpatroon:";GO
30 PLAY "s=go;cdef"
40 IF PLAY(1)<>0 THEN GOTO 40
50 NEXT GO
```

In het programma vindt u op regel 10 de laatste nieuwe PLAY deelopdracht.
Dit is de M deelopdracht van Magnitude. Deze deelopdracht geeft aan hoe lang het duurt voor het golfpatroon afgelopen is. Het getal geeft de lengte van de cyclus aan waarin de met de S opdracht gekozen vorm herhaald wordt. Hoe groter het getal hoe langer de cyclus is. U kunt achter M een getal zetten tussen 1 en 65535. Een lange cyclus wil zeggen dat het gehele golfpatroon

langer klinkt. Het duurt langer voordat het golfpatroon herhaald wordt als de cyclus lang is. Heeft u dus erg lange waarden gekozen achter de M deelopdracht, dan wordt het verschil tussen de verschillende golfpatronen te klein om te horen. Als we golfpatroon S13 en S14 allebei heel erg gerekt worden (M is erg groot), en de tijd die de toon ter beschikking heeft om te klinken (snel tempo, dat wil zeggen T is hoog en een korte noot, dat wil zeggen L is hoog), zal slechts een deel van het golfpatroon gespeeld worden. Dat kan ertoe leiden dat er geen verschil te horen is:



De waarde van M in het begin (als u niets anders instelt) is 255. Zowel de S als de M deelopdracht hoeven maar een keer opgegeven te worden en gelden dan voor alle volgende PLAY opdrachten (tenzij nieuwe S en M waarden gegeven worden). De uiteindelijke klank is mede afhankelijk van het tempo (T) en de lengte van de noot (L). Probeert u bovenstaand programma op verschillende plaatsen te veranderen en luister naar het resultaat. Het programma zoals dat hierboven staat heeft alle golfpatronen flink uitgerekt, waardoor u het verschil goed kunt horen. Tevens kunt u horen dat ook met de PLAY opdracht de nodige vreemde geluiden te maken zijn.

Een opdracht in het laatste programma die nog niet besproken is is de PLAY opdracht. Deze moet niet verward worden met de PLAY opdracht om muziek voort te brengen. De PLAY() opdracht is bedoeld om te kijken of de computer al klaar is met het spelen van de muziek. Dit kan bijzonder handig zijn. De computer maakt de geluiden namelijk veel sneller aan dan ze uiteindelijk gemaakt worden. Het kost elektronisch slechts een fractie van een seconde om een geluid te maken dat volgens de PLAY opdracht een tijdje moet klinken. Als de computer sneller is dan de uitvoering van het geluid, wordt een deel van de geluidsopdrachten voor de luidspreker vast in een buffer gezet. Door de PLAY() opdracht te

gebruiken kan gekeken worden of die buffer leeg is. Als u het vorige programma runt zonder regel 40 ziet u dat de computer reeds verschillende keren door de lus is terwijl u nog maar een paar tonen gehoord heeft. Dat komt doordat het programma veel verder is dan er aan geluiden voortgebracht is. De rest zit al in de buffer. De opdracht

```
PLAY(0)
```

let op de buffer van geluidsgenerator 0 (de enige die we tot nu toe gebruikt hebben). Als deze leeg is wordt de waarde 0 gegeven. Anders is de waarde van PLAY(0) gelijk aan -1.

Het uiteindelijke geluid dat de MSX voortbrengt is voornamelijk afhankelijk van een van de deelopdrachten. Elke combinatie van deelopdrachten zorgt weer voor andere effecten. De effecten van het kiezen van een andere noot zijn tamelijk voorspelbaar, maar de effecten van vooral de S en de M deelopdracht kunnen verrassend zijn.

Als u het volgende programma runt krijgt u een keur van geluiden te horen terwijl het beeldscherm aangeeft wat er gebeurt. Alle geluiden klinken in toonhoogte C.

```
10 FOR TE=32 TO 250 STEP 50
20 FOR MA=1 TO 6553 STEP TE*10
30 FOR GO=8 TO 15
40 PRINT "golf:";GO;"  magn.:";MA;"  tempo:";TE : PRINT
50 PLAY "S=go;M=ma;T=te;c1"
60 IF PLAY(1)<>0 THEN GOTO 60
70 BEEP
80 NEXT GO
90 NEXT MA
100 NEXT TE
```

Het volgende programma laat het effect van golfpatroon S1 zien. Dit golfpatroon benadert wellicht het beste het geluid van een piano dat immers snel tot volle geluidssterkte stijgt (bij het aanslaan) en daarna geleidelijk wegsterft (het uittrillen van de snaar). Zoals het programma hier staat klinkt de piano wel overdreven stacato. Door in regel 10 de waarde voor M te verhogen tot 10000 klinkt het al veel echter. Experimenteert u zelf verder met dit programma. Bekijk ook welke effecten u krijgt als u de verschillende programmalussen verandert.

```
10 PLAY "T255S0M1000164"
20 A=12: C=6
30 FOR X=A TO A+4
40 PLAY "N=x;"
50 NEXT X
60 FOR X=A+12 TO A+16
70 PLAY "N=x;"
80 NEXT X
90 IF A>60 THEN C=-C
100 IF A<10 THEN C=-C
110 A=A+C : GOTO 30
```

Drie geluidsgeneratoren

Tot nu toe werden alle programma's ten gehore gebracht op een kanaal. De MSX kent echter drie verschillende geluidsgeneratoren in dezelfde chip. De tweede en derde programmeert u door ze achter de PLAY opdracht voor de eerste geluidsgenerator te zetten, gescheiden door komma's.

```
10 A$="o3dfa4df2a2r4"
20 B$="o4dfad2r4"
30 C$="o5dfao4d2r4"
40 PLAY A$,B$,C$
50 PLAY "","",C$
60 PLAY "",B$,""
70 PLAY A$,"",""
80 PLAY A$,B$,""
90 PLAY A$,"",C$
100 PLAY "",B$,C$
110 PLAY "v612xa$;",B$,C$
```

Met dit programma kunt u horen dat de verschillende geluidskanalen elk onafhankelijk van elkaar kunnen klinken. Van elk van de generatoren is het volume, het tempo en de lengte der noten te bepalen. Er is slechts een golfpatroon voor alle drie de generatoren mogelijk.

De geluidseffecten

Voor het maken van geluidseffecten kent de MSX de opdracht

SOUND , (geluid)

Achter deze opdracht komt telkens een registercode, een komma en

de waarde die in het register gezet moet worden.
U moet de PSG voorstellen als een doos met 14 hokjes. Datgene dat u in de hokjes stopt bepaalt hoe een geluid gaat klinken. De computer loopt telkens alle hokjes door.
Opvallend aan de SOUND opdracht is dat u de computer gegevens geeft zodat een bepaald geluid gemaakt kan worden. Een van de gegevens die niet ingevoerd kan worden is de tijdsduur van het geluid. Dat wil zeggen dat de SOUND opdracht zelf niet kan zorgen voor het na verloop van tijd uitschakelen van het geluid. Daarvoor moet het programma zorgen. Voordeel is echter dat het programma nadat de registers ingesteld zijn, helemaal niet meer op het geluid hoeft te letten en verder gaat met de afhandeling van de rest van het programma.
De PSG kunnen in principe slechts twee soorten geluiden voortbrengen: een geluid dat bestaat uit een golf en een ruisgeluid. Doordat de eigenschappen van deze twee geluiden bepaald kunnen worden (bijvoorbeeld het golfpatroon van de ontwikkeling van het volume, zie de PLAY opdracht) kunnen zeer veel verschillende geluiden gemaakt worden.
Om de twee basisgeluiden te horen tikt u eerst de volgende twee opdrachten:

```
SOUND 7,62
SOUND 8,8
```

U hoort nu een doordringende toon. De toon klinkt als een doorlopende toeter. Tik nu:

```
SOUND 7,55
```

Nu hoort u het ruizen van de zee, of van een radio die niet goed afgestemd is. Dit is de zogenaamde 'witte ruis'.
Om beide geluiden te horen hebben we niets anders gedaan dan een geluidskanaal geopend, het volume bepaald en bepaald of er een ruis of een golftoon moest komen. U zet de ruis weer uit door CTRL STOP.

De veertien registers die u kunt vullen met verschillende waarden zijn:

Register	Funktie
0	Fijne frequentiebepaling kanaal A
1	Ruwe frequentiebepaling kanaal A
2	Fijne frequentiebepaling kanaal B
3	Ruwe frequentiebepaling kanaal B
4	Fijne frequentiebepaling kanaal C
5	Ruwe frequentiebepaling kanaal C
6	Frequentie van de ruisgenerator
7	Keuze ruis/toon en kanaal
8	Volume kanaal A
9	Volume kanaal B
10	Volume kanaal C
11	Frequentie van golfpatroon
12	Frequentie van golfpatroon
13	Keuze van golfpatroon

Registers 0, 1, 2, 3, 4 en 5

Deze registers bepalen de frequenties van de verschillende kanalen. Voor elk kanaal zijn er twee registers. Een voor de grove afstelling en een voor de fijne. De verdeling en minimum en maximum instelwaarden zijn als volgt:

register	funktie	bereik
0	fijn A	0-255
1	grof A	1-15
2	fijn B	0-255
3	grof B	1-15
4	fijn C	0-255
5	grof C	1-15

De frequentie wordt als volgt vastgesteld:

```
frequentie = fijnregister + (grofregister * 256)
```

Deze formule geeft u echter niet de frequentie in hertz, maar slechts een decimale waarde. Hoe hoger deze waarde, hoe lager de toon. U kunt het hoogste en laagste geluid van de MSX horen door:

```
SOUND 0,0
SOUND 1,0
```

```
SOUND 7,62
SOUND 8,8
```

Het laagste geluid krijgt u door in bovenstaande rij de eerste twee opdrachten te vervangen door:

```
SOUND 0,255
SOUND 1,15
```

Wilt u de frequentie weten in hertz (soms heeft u dit nodig voor het stemmen van de MSX met een bepaald muziekinstrument), dan kunt u de volgende formule gebruiken:

frequentiewaarde = 178900 / (16 X frequentie in Hz)

Het getal 178900 is de klokfrequentie van de PSG.

Register 6

Dit register bepaalt de frequentie van het ruiskanaal. U heeft de keuze uit 32 waarden (0-31). Hoe lager de waarde hoe hoger de ruis klinkt. Het volgende programma laat ze allemaal horen:

```
10 FOR FR=0 TO 31
20 SOUND 7,55
30 SOUND 8,8
40 SOUND 6,FR
50 FOR Z=1 TO 300 : NEXT Z
60 BEEP
70 NEXT FR
80 STOP
```

Register 7

Dit is wellicht de lastigste van de registers. Dit register zet het geluid van elk van de drie kanalen aan en bepaalt of het kanaal een toon of een ruis voortbrengt. De waarden die opgegeven moeten worden om dit te bewerkstelligen kunnen gevonden worden door naar de 8 bits van dit register te kijken:

Bit	bitwaarde	funktie	bit	bitwaarde	funktie
0	0	toon op A	0	1	geen toon op A
1	0	toon op B	1	1	geen toon op B
2	0	toon op C	2	1	geen toon op C
3	0	ruis op A	3	1	geen ruis op A
4	0	ruis op B	4	1	geen ruis op B
5	0	ruis op C	5	1	geen ruis op C

U krijgt de decimale waarde door de bits die u nodig heeft bij elkaar op te tellen. Dit gaat precies zo als bij het maken van een rij van een spritepatroon. Lees dat hoofdstuk eventueel na. In het kort gaat het als volgt: De meest rechtse bit is 1 waard, de bit links daarvan is 2 waard, de bit daar weer links van is 4 waard, de bit daar weer links van is 8 waard. Enzovoorts tot de meest linkse bit die 128 waard is. De twee linkse bits worden voor dit register echter nooit gebruikt en moeten altijd gelijk gesteld worden aan 0.
Stel u wilt ruis en toon op A, niets op B en ruis op C. De acht bits worden dan: 00010110. Decimaal is dat 0+2+4+0+16+0+0+0=11. Probeer met onderstaand programma uit of dit klopt.

```
10 SOUND 7,22
20 SOUND 2,13
30 SOUND 6,15
40 FOR VO=8 TO 10
50 SOUND VO,8
60 FOR Z=1 TO 600 : NEXT Z
70 SOUND VO,0
80 FOR Z=1 TO 500 : NEXT Z
90 NEXT VO
100 SOUND 9,8
110 SOUND 8,8
```

Als u het omrekenen teveel werk vindt kunt u de bits ook voorstellen als nullen en enen en het getal als binair getal aan de computer opgeven. U kunt in bovenstaand programma regel 10 vervangen door

```
10 SOUND 7,&B00010110
```

Registers 8, 9 en 10

Deze registers bepalen het volume van de geluiden. Dit is meestal niet zo nodig omdat er ook een volumeregelaar op de televisie of

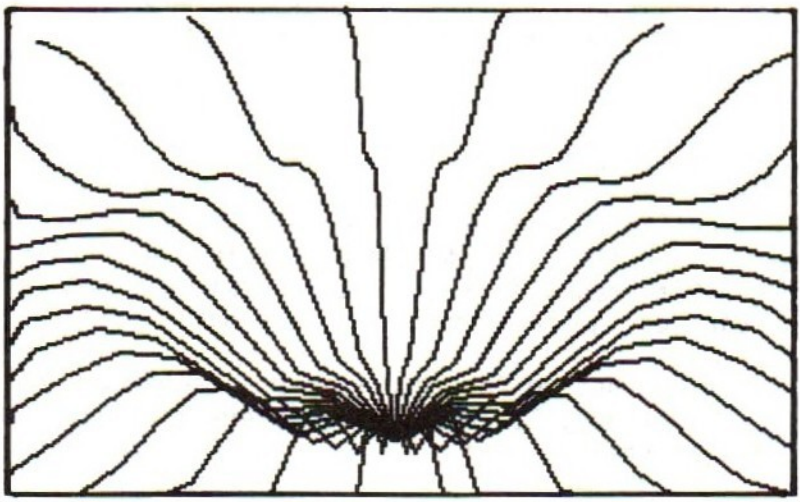
op de monitor zit. Maar voor een goede samenhang met het klikken van het toetsenbord is het toch geen overbodige luxe. De minimumwaarde is 0, u hoort dan niets. De maximumwaarde is 15. Let u daarbij wel op de volumeknop van de televisie, anders wordt u de kamer uit getetterd.
Naast de waarden van 0 tot en met 15 kunt u ook de waarde 16 invullen. Daarmee geeft u de computer te kennen dat u het volume wilt laten bepalen door een golfpatroon. Deze golfpatronen zijn gelijk aan de golfpatronen van de PLAY opdracht. Voor een nadere uitleg kunt u dan ook het deel over de PLAY opdracht nalezen.

Registers 11, 12 en 13

Deze drie registers bepalen het gebruik van golfpatronen bij de drie kanalen. Van elk van de drie kanalen die als volume 16 opgekregen heeft (register 8, 9 of 10) ontwikkelt het volume zich als een van de golfpatronen die behandeld zijn bij de PLAY opdracht.
De frequentie van het golfpatroon (zie bij magnitude bij de PLAY opdracht) wordt bepaald door de registers 11 en 12. Dit gaat volgens de volgende formule:

magnitude = waarde van reg. 12 + (256 X de waarde van reg. 11)

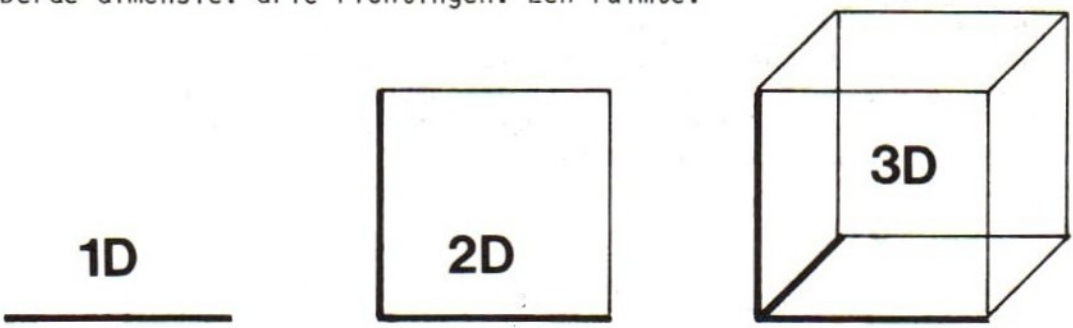
U kunt slechts een envelope per keer kiezen. U kunt wel van elk kanaal apart bepalen of het een gewoon constant volume heeft, of een volumeontwikkeling volgens het golfpatroon.



20. DE DERDE DIMENSIE

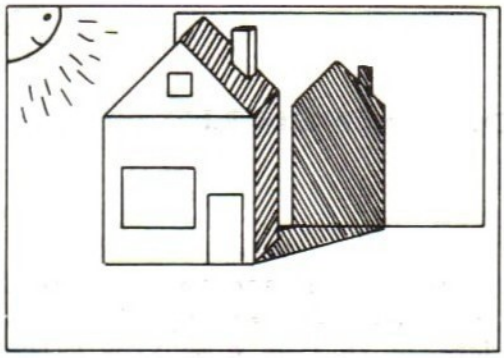
Een, twee of drie

Het beeldscherm van uw televisie of monitor is plat. Het papier waarop de tekst die u nu leest gedrukt is, is plat. Van deze voorwerpen zegt men dat zij in twee dimensies uitgedrukt kunnen worden.
De normale voorstelling van de verschillende dimensies is:
Eerste dimensie: een richting. Een lijn.
Tweede dimensie: twee richtingen. Een vlak.
Derde dimensie: drie richtingen. Een ruimte.



een dimensie, twee dimensies, drie dimensies

Een een-dimensionaal object of een twee-dimensionaal object kan gemakkelijk op papier getekend worden, zonder dat er ook maar iets aan de vorm veranderd hoeft te worden. Zo is het ook altijd duidelijk te zien welk een- of twee-dimensionaal object bedoeld wordt met een (twee-dimensionale) tekening.
Om niet al te langdradig te worden korten we een-dimensionaal af door 1D, twee-dimensionaal door 2D en drie-dimensionaal door 3D.
Zodra een object weergegeven moet worden op een voorwerp met een lagere dimensie krijgen we te maken met een probleem. Er moet dan geprojecteerd worden. Dit projecteren moet u heel letterlijk opvatten. Het werkt net zo als bij fotografie. Een 3D object, bijvoorbeeld een huis, wordt geprojecteerd op een 2D object, namelijk de film in het fototoestel. Een ander dagelijks voorkomend voorbeeld is de schaduw van een 3D object, bijvoorbeeld een huis, op een 2D object, bijvoorbeeld de grond of een muur.



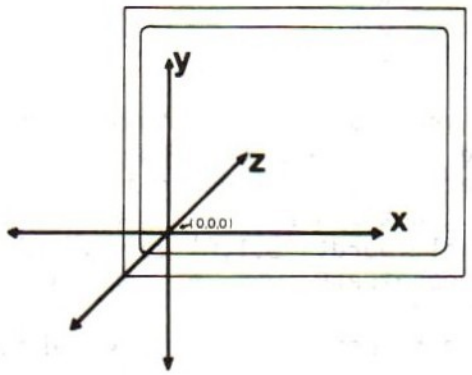
drie-dimensionaal naar twee-dimensionaal

Een van de problemen van projecteren is dat na de projectie de diepte van de oorspronkelijke vorm niet af te lezen is in de projectie.
Met deze problemen krijgt u te maken als u 3D tekeningen met de MSX wilt maken. Het beeldscherm waar de tekening uiteindelijk op komt te staan is immers plat. Er zal geprojecteerd moeten worden. De tekeningen in dit hoofdstuk geven alleen de buitenste lijnen van een voorwerp aan (de ribben).

Weergave op het beeldscherm

Het coordinatenstelsel om te tekenen in hoog oplossend vermogen bestaat uit een X-as (links-rechts) en een Y-as (boven-beneden). Door twee getallen op te geven, de X-coordinaat en de Y-coordinaat, kan een punt aangegegeven worden. De notatie daarvoor is (X,Y).
De derde dimensie kan natuurlijk niet weergegeven worden op een beeldscherm. Deze moet er dus telkens even bijgedacht worden. In dit boek is gekozen voor een systeem van denken waarbij de kijker het scherm als het ware 'inkijkt'.
De X-as blijft waar hij is, van links naar rechts. De Y-as blijft waar hij is, van boven naar beneden. De diepte wordt aangegeven door een derde as, de Z-as. Deze Z-as gaat van de oppervlakte van het scherm de diepte in. Dat wil zeggen, van de kijker weg, de televisie in. De X-as staat loodrecht op de Y-as, maar ook loodrecht op de Z-as. De Z-as staat loodrecht op de X-as en op de Y-as.
Een punt wordt in 3D aangegeven door drie coordinaten (X,Y,Z). Het eerste getal geeft de verplaatsing links-rechts aan. Positieve getallen rechts van de oorsprong (het punt (0,0,0)), negatieve getallen links van de oorsprong. Het tweede getal geeft de verplaatsing boven-beneden aan. Positieve getallen onder de oorsprong, negatieve getallen eronder. Let op: dit is anders dan

de normale wiskundige notatie. Het derde getal geeft de verplaatsing voor-achter (de diepte in) aan. Positieve getallen achter het oppervlak van het beeldscherm, van de kijker weg. Negatieve getallen voor het oppervlak van het beeldscherm, naar de kijker toe.



richtingen

Matrices

Dit boek beoogt allerminst een wiskundeboek te zijn. Bij het projecteren van 3D objecten is een beetje matrix-rekenkunde echter onontbeerlijk. In dit boek vindt u in hoofdlijnen aangegeven hoe de in de programma's gebruikte formules tot stand zijn gekomen. Als u al veel kennis van matrix-rekenkunde heeft kan het een hulp zijn voor het verder uitwerken van experimenten. Voor hen die de matrix-rekenkunde niet machtig zijn, is het een basis om de werking van de programma's te kunnen doorgronden.

Voor de basis van de matrix-rekenkunde zoals die hier gebruikt wordt, moet u nog even het hiervoor behandelde 3D coordinatenstelsel in gedachten nemen. Elk punt in dit stelsel kan aangeduid worden met drie getallen; een X-, Y- en Z-coordinaat. Elk punt kan echter ook gezien worden als het resultaat van een stel verplaatsingen beginnend bij de oorsprong. Deze verplaatsingen worden vectoren genoemd. Zo is het punt (2,4,7) te beschouwen als het resultaat van een verplaatsing 2 maal een eenheid in de X-richting plus een verplaatsing van 4 maal een eenheid in de Y-richting plus een verplaatsing van 7 maal een eenheid in de Z-richting.
In een belangrijk deel van de wiskunde, de matrix-rekenkunde, worden meetkundige afbeeldingen (bijvoorbeeld rotaties) weergegeven met behulp van een matrix. Een matrix is een schema van een aantal getallen (of grootheden) die in de vorm van een

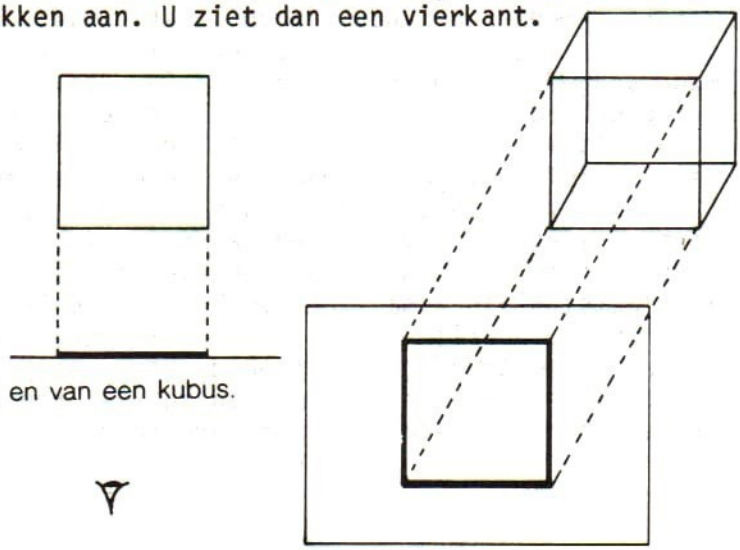
rechthoek geplaatst zijn. Een matrix is onder te verdelen in rijen (horizontaal) en kolommen (vertikaal). Een voorbeeld van een meetkundige afbeelding weergegeven met behulp van een matrix is het volgende:

$$\begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

In de eerste kolom van zo'n matrix staat genoteerd wat de meetkundige afbeelding doet met het punt (1,0,0). In ons voorbeeld wordt het punt (1,0,0) veranderd in het pnt (3,0,0). In de tweede kolom staat genoteerd wat er gebeurt met het punt (0,1,0). In ons voorbeeld wordt (0,1,0) veranderd in (0,2,0). In de derde kolom staat genoteerd wat er gebeurt met het punt (0,0,1). Dat punt wordt (0,0,4).
Wat gebeurt er nu met het punt (2,4,7) als we deze meetkundige afbeelding toepassen?
(2,4,7) is te schrijven als 2*(1,0,0)+4*(0,1,0)+7*(0,0,1).
(2,4,7) wordt veranderd in 2*(3,0,0)+4*(0,2,0)+7*(0,0,4).
Dat wordt (6,0,0)+(0,8,0)+(0,0,28). Dat is gelijk aan het punt (6,8,28).
Deze eigenschappen van het matrixrekenen worden straks gebruikt bij de rotatie in een 3D ruimte.

Projectie

Zoals al vermeld is het onmogelijk om te tekenen binnen de televisie. Om een 3D object op het beeldscherm weer te geven moet er geprojecteerd worden.
Eerst een voorbeeld van een eenvoudige projectie. Neem een vierkant en kijk recht van voren tegen de zijkant aan. U ziet dan alleen een lijnstuk. Of neem een kubus en kijk recht van voren tegen een van de vlakken aan. U ziet dan een vierkant.



projectie van een vierkant en van een kubus.

Maar het is niet de bedoeling om telkens recht tegen de voorkant van een object aan te kijken. Bij de hier gebruikte wijze van projecteren, de orthografische projectie, wordt van elk punt de Z-coordinaat nul gemakt. Daardoor is na het projecteren van een kubus alleen een plat figuur te zien. De matrixnotatie voor het nulmaken van de Z-coordinaat is:

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

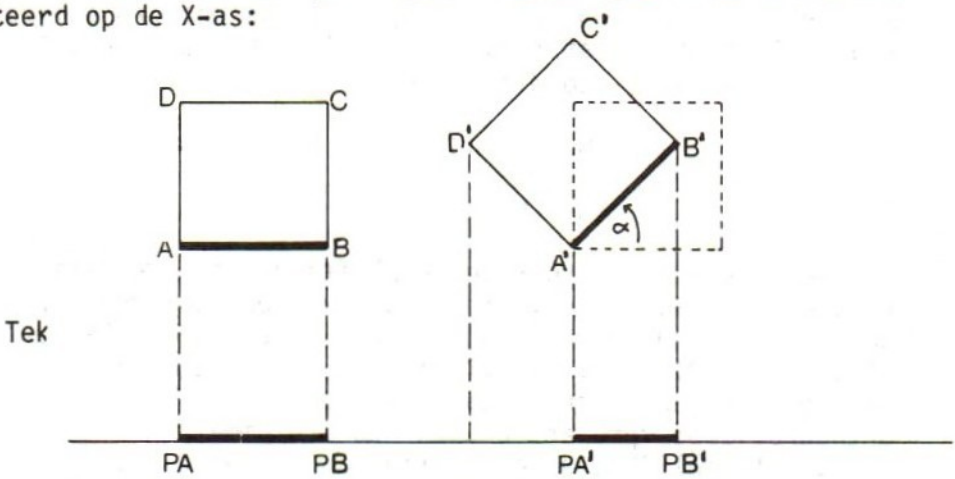
De vector die zorgt voor de verplaatsing in de Z-richting wordt (0,0,0) gemaakt. De vectoren die zorgen voor de verplaatsing in de X- en in de Y-richting blijven onveranderd. Het punt (1,0,0) blijft (1,0,0), (0,1,0) blijft (0,1,0) en (0,0,1) wordt (0,0,0). Als een bepaald punt geprojecteerd moet worden volgens de orthografische projectie kan de kolommatrix die het punt weergeeft (gelijk aan de notatie van de coordinaten, maar dan van boven naar beneden) vermenigvuldigd worden met de vierkante 3 bij 3 matrix die staat voor deze vorm van projectie. Het punt (5,8,9) geeft na projectie het punt (5,8,0). Dat klopt, want alle punten moeten uiteindelijk in een vlak terecht komen (het vlak waarop geprojecteerd wordt).

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 5 \\ 8 \\ 9 \end{pmatrix} = \begin{pmatrix} 5 \\ 8 \\ 0 \end{pmatrix}$$

Voor andere vormen van projectie (bijvoorbeeld de scheve projectie of de Marcatorprojectie) kunt u een wiskundeboek of een goede atlas naslaan.
Veel mooier wordt een tekening als we schuin tegen een object aankijken. Dat is ook altijd de wijze waarop een kubus weergegeven wordt. De kubus wordt getekend vanuit een gezichtspunt iets opzij en iets boven de kubus.
Om dit effect te bereiken zijn twee systemen mogelijk: De kijker verplaatst zich, of het object draait.
Er is geen wezenlijk verschil tussen deze twee, maar de tweede methode is wat eenvoudiger. Voor de wiskundige achtergrond van het volgende wordt u verwezen naar middelbare schoolboeken over vectormeetkunde, waarin de rotaties behandeld worden. Hier zal heel in het kort het principe uitgelegd worden.

Rotatie

Het vierkant uit het vorige figuur wordt gedraaid en daarna geprojecteerd op de X-as:

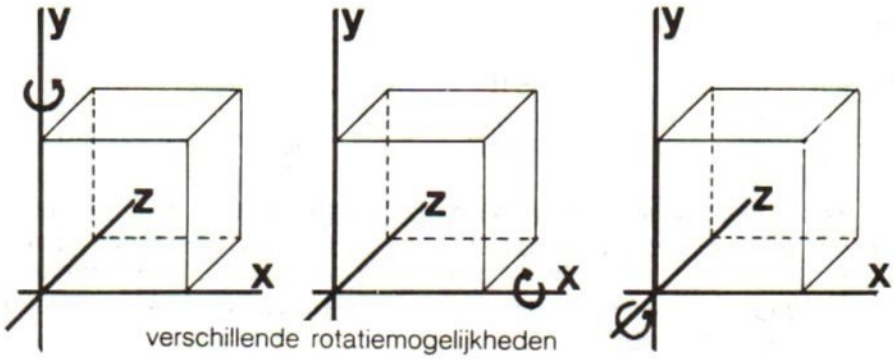


rotatie en projectie van een vierkant

Tek

De punten A, B, C en D zijn de hoekpunten van het vierkant zonder dat het gedraaid is. De punten A', B', C' en D' zijn de hoekpunten van het vierkant na draaiing. De draaiing vond plaats over een hoek van 45 graden. De afstand AB is 1. Het figuur is een vierkant dus alle andere zijden zijn ook precies 1 eenheid lang.
Bij een projectie zonder draaiing is de afstand PA-PB (de afstand tussen de projectiepunten van A en B) gelijk aan 1. De afstand tussen PD en PC is ook 1. De afstand tussen PA en PD en tussen PB en PC is 0.
Na het draaien ziet het er heel anders uit. De afstand PA' en PB' is kleiner dan 1. Hoeveel kleiner dat is, is afhankelijk van de hoek waarover gedraaid is. Bij een draaiing over een hoek van 45 graden is de afstand tussen PA' en PB' ongeveer 0.7. Voor de wiskundige lezers: deze afstand is precies gelijk aan 0.5*SQR(2), oftewel COS(RAD(45)). Hoe meer er gedraaid wordt, hoe kleiner de afstand. Als er meer dan 90 graden wordt gedraaid, wordt de afstand zelfs negatief (PB' ligt dan links van PA').
Op deze wijze zijn de verschillende afstanden op de projectielijn uit te rekenen en zijn de verschillende projectiepunten te berekenen.

Rotaties in 3D gaan niet wezenlijk anders. Er is nu echter de keuze uit drie verschillende rotaties: om de X-as, om de Y-as en om de Z-as.



verschillende rotatiemogelijkheden

De matrix van een rotatie over een hoek van A graden om de X-as ziet er als volgt uit:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & COS(A) & SIN(A) \\ 0 & -SIN(A) & COS(A) \end{pmatrix}$$

U ziet dat het punt (1,0,0) bij deze rotatie onveranderd blijft. Het punt(0,1,0) wordt veranderd in het punt (0,COS(A),-SIN(A)). Als we voor A bijvoorbeeld 30 graden nemen dan wordt het punt (0,1,0) dus veranderd in (0,0.866,-0.5) (0.866 is een benadering van 0.5*SQR(3), dat is de cosinus van 30 graden. 0.5 is de sinus van 30 graden).
De matrix van een rotatie over een hoek van B graden om de Y-as ziet er als volgt uit:

$$\begin{pmatrix} COS(B) & 0 & -SIN(B) \\ 0 & 1 & 0 \\ SIN(B) & 0 & COS(B) \end{pmatrix}$$

Bij deze rotatie blijft het punt (0,1,0) onveranderd.
De matrix voor een rotatie over een hoek van C graden om de Z-as is als volgt:

$$\begin{pmatrix} COS(C) & SIN(C) & 0 \\ -SIN(C) & COS(C) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Hier blijft het punt (0,0,1) onveranderd.

Het volgende programma tekent een kubus die niet recht van voren wordt bekeken. De kubus wordt eerst gedraaid om de X-as en daarna om de Y-as. De matrix die bij deze gecombineerde draaiing hoort verkrijgt men door de matrix van een rotatie om de X-as te vermenigvuldigen met de matrix van een rotatie om de Y-as.
Een rotatie om de X-as over een hoek van A graden gevolgd door

een rotatie om de Y-as over een hoek van B graden geeft:

$$\begin{pmatrix} COS(B) & 0 & -SIN(B) \\ 0 & 1 & 0 \\ SIN(B) & 0 & COS(B) \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 \\ 0 & COS(A) & SIN(A) \\ 0 & -SIN(A) & COS(A) \end{pmatrix}$$

Het resultaat van deze vermenigvuldiging moet nog vermenigvuldigd worden met de projectiematrix (de matrix die ervoor zorgt dat de Z-coordinaten gelijk aan 0 worden). Vervolgens moet vermenigvuldigd worden met de kolommatrix van het te roteren punt (X,Y,Z) van de kubus.

$$\begin{pmatrix} PX \\ PY \\ PZ \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} COS(B) & SIN(A)*SIN(B) & -COS(A)*SIN(B) \\ 0 & COS(A) & SIN(A) \\ SIN(B) & -SIN(A)*COS(B) & COS(A)*COS(B) \end{pmatrix} * \begin{pmatrix} X \\ Y \\ z \end{pmatrix}$$

Het eindresultaat:

```
PX = X*COS(B) + Y*SIN(A)*SIN(B) - Z*COS(A)*SIN(B)
PY = Y*COS(A) + Z*SIN(A)
PZ = 0
```

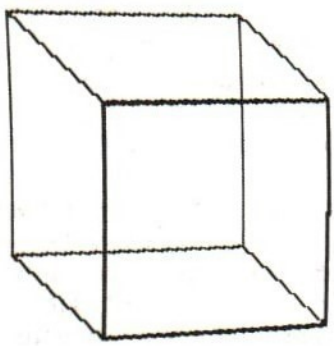
De waarden PX, PY en PZ staan voor de waarden van respectievelijk de X-coordinaat, de Y-coordinaat en de Z-coordinaat van het punt na projectie. Het punt heeft een Z-coordinaat die 0 is en ligt dus op het projectievlak (in ons geval het beeldscherm). Een rotatie die eerst om de Y-as gaat met een hoek van B graden en daarna om de X-as met een hoek van A graden en een projectie geeft als resultaat:

```
PX = X*COS(B) - Z*SIN(B)
PY = X*SIN(A)*SIN(B) + Y*COS(A) + Z*SIN(A)*COS(B)
PZ = 0
```

Dit programma laat een kubus in verschillende standen zien.

```
10 SCREEN 2
20 DIM X(21)
30 DIM Y(21)
40 DIM Z(21)
50 FOR D=1 TO 21
60 READ X(D),Y(D),Z(D)
70 NEXT D
80 INPUT "draaiing om y-as";A
90 INPUT "draaiing om x-as";B
100 A=(A*3.14)/180
110 B=(B*3.14)/180
120 SCREEN 2
```

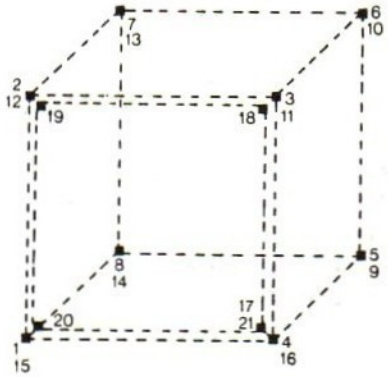
```
130 DRAW "bm100,130"
140 K=COS(A):L=SIN(A)*SIN(B):M=SIN(A)*COS(B):N=COS(B):J=SIN(B)
150 FOR D=1 TO 21
160 PX=X(D)*K+Y(D)*L-Z(D)*M
170 PY=Y(D)*N+Z(D)*J
180 Q=PX+100 : R=130-PY
190 DRAW "m=q;,=r;"
200 NEXT D
210 FOR PA=1 TO 2000:NEXT PA
220 SCREEN 1 : GOTO 80
230 DATA 80,0,0,80,80,0,0,80,0,0,0,0,1,1,1,79,1,0
240 DATA 79,79,0,1,79,0,1,1,0,80,0,0,80,0,80
250 DATA 80,80,80,80,80,0,80,80,80
260 DATA 0,80,80,0,80,0,0,0,0,0,0,0,80
270 DATA 80,0,80,0,0,80,0,80,80
```



een kubus getekend door bovenstaand programma

In de regels 20-40 wordt voldoende plaats gemaakt voor het tekenen van de kubus. In de regel 50-70 worden de verschillende gegevens voor de kubus in een array geplaatst. Daardoor gaat het tekenen sneller.
De gegevens die in de datalijst opgenomen zijn, zijn de punten waarnaar een lijn al dan niet getrokken moet worden (telkens een X-, Y- en Z-coordinaat). In het voorbeeldprogramma kon volstaan worden met het telkens trekken van een lijn. Een kubus kan

getekend worden door 16 lijnen te trekken. Om te zorgen dat de voorste lijnen ietsje dikker worden (hetgeen het 3D effect vergroot) zijn 21 lijnen nodig volgens onderstaand schema:



Als u andere figuren dan kubussen wilt tekenen, zult u soms te maken krijgen met tekeningen waarbij het veel eenvoudiger is om 'het potlood' even te verplaatsen zonder een lijn te trekken. U moet dan elk coordinatenstel voorzien van een vierde getal (0 of 1) voor het al dan niet tekenen van de lijn. Het is het handigste om daarbij altijd 0 aan te geven als er geen lijn getrokken moet worden en 'het potlood' alleen verplaatst moet worden. 1 staat dan voor het tekenen van een lijn. Een eenvoudige IF..THEN..opdracht zorgt ervoor dat op de juiste plaatsen de lijn getrokken wordt.
De regels 80 en 90 vragen de gebruiker om welke hoek de vorm gedraaid moet worden voordat hij geprojecteerd en getekend wordt.
De regels 100 en 110 zijn wat cryptisch. De meest gebruikte wijze om hoeken aan te geven is in graden. Er gaan 360 graden in een volle cirkel. Om een programma een beetje gebruikersvriendelijk te maken is het in elk geval nodig dat de gebruiker de hoeken in een voor hem/haar bekende grootheid op kan geven. In dit geval graden. De MSX werkt echter met radialen. In een volle cirkel gaan 2*PI radialen. PI is ongeveer gelijk aan 3.14. In de regel 100 en 110 rekent de MSX zelf de hoek in graden om in radialen.
Regel 120 schakelt om naar een grafisch scherm.
Regel 130 verplaatst de grafische cursor naar de beginpositie.
Regel 140 moet u in samenhang met de regel 160 en 170 lezen. De laatste twee regels worden voor elk punt herhaald. Regel 140 wordt maar een keer per tekening uitgevoerd. Door op deze wijze een deel van de berekeningen naar een programmaregel te verplaatsen waar zij maar een keer uitgevoerd hoeven te worden in plaats van vele keren, kan het tekenen vele malen sneller gaan. De berekening die de computer bij elk te tekenen punt moet maken is nu veel korter. Test u dit uit door in regel 160 en 170 de

oorspronkelijke berekening te plaatsen:

```
160 PX=X(D)*COS(A)+Y(D)*SIN(A)*SIN(B)-Z(D)*SIN(A)*COS(B)
170 PY=Y(D)*COS(B)+Z(D)*SIN(B)
```

Ziet u hoeveel langzamer het tekenen gaat? Bij het maken van uw eigen tekenprogramma's waar veel rekenwerk in voorkomt moet u hier goed op letten: laat zo veel mogelijk rekenwerk van te voren (dat wil zeggen buiten de tekenlus) doen. Een herhalingslus moet zo kort mogelijk zijn om snelle tekeningen te kunnen maken.
De regels 180 en 190 zorgen voor het uiteindelijke tekenwerk. Regel 180 zou eigenlijk overbodig moeten zijn. Door de coordinaten zorgvuldig te kiezen kan ervoor gezorgd worden dat een verschuiving niet meer nodig is. Anderzijds heeft een verschuivingsfactor het voordeel dat de vorm over het scherm heen en weer bewogen kan worden.
Regel 210 wacht een tijdje voordat regel 220 weer terugschakelt naar scherm 1 om de gebruiker een nieuw stel hoeken op te laten geven.

Een wijnglas

Door het vorige programma wat uit te breiden kunt u het uzelf nog gemakkelijker maken. Het onderstaande programma tekent niet alleen een door u bedacht voorwerp, het bedenkt zelfs hoe het voorwerp er uit moet zien. Het enige dat u hoeft in te voeren is een willekeurige gekromde lijn. De computer draait deze lijn telkens een paar graden rond. De rotatie vindt plaats rond de Y-as.

Na 360 graden is de lijn weer op zijn oude standpunt.

Omdat de kijker tijdens dit proces recht van voren tegen de draaiende lijn aankijkt, lijkt het of de computer een eenvoudige 2D tekening heeft gemaakt. Maar de computer heeft tijdens het draaien de verkregen gegevens opgeslagen in een array. Als u na het vormen van het figuur de computer vraagt het voorwerp te roteren rond de X-as kunt u het voorwerp van verschillende kanten bekijken.

```
10 SCREEN 2 : COLOR 4,15,5
20 CLS
30 DIM X(9)
40 DIM Y(9)
50 DIM Z(33,9,2)
60 FOR Q=1 TO 9
70 READ X(Q),Y(Q)
80 NEXT Q
90 FOR D=1 TO 32
100 DRAW"bm125,170"
110 S=SIN(D*.2)
120 C=COS(D*.2)
130 FOR A=1 TO 9
140 Z(D,A,1)=S*X(A)
150 Z(D,A,2)=C*X(A)
160 LINE - (Z(D,A,1)+125,Y(A)+170),1
170 NEXT A : PSET (125,170),15 : NEXT D
180 FOR B=.5 TO 1.8 STEP .3
190 CLS
200 PSET (125,170),15
210 N=COS(B):J=SIN(B)
220 FOR D=1 TO 32
230 PSET (125,170),15
240 FOR A=1 TO 9
250 PX=Z(D,A,1)
260 PY=Y(A)*N+Z(D,A,2)*J
270 IF Y(A)<-79 THEN LINE
    -(PX+125,PY+170),9 ELSE LINE -(PX+125,PY+170),4
280 NEXT A : NEXT D
290 NEXT B
300 DATA 40,0,4,6,4,-30,10,-45,4,-55,4,-70,30,-80,35,-106,55,-150
```

Dit programma laat als voorbeeld een wijnglas zien. Nadat eerst het glas in 2D getekend is, waarbij de gegevens voor de 3D weergave tegelijk berekend worden, wordt het glas vervolgens in verschillende 3D standen getoond. De hoeken waarover het glas om de X-as geroteerd wordt, kunt u (in radialen) aflezen in regel 180. Roteren rond de Y-as heeft niet zoveel zin, omdat het voorwerp verkregen is door rotatie rond de Y-as en perfect symetrisch is. Door de draaiing rond de X-as kunt u het glas van binnen bekijken.
De regels 30 en 40 reserveren ruimte voor de 2D weergave.
Regel 50 reserveert ruimte voor de 3D gegevens.
De regels 60-80 lezen de gegevens uit de datalijst. De gegevens bestaan uit telkens twee getallen die een punt van de hiervoor getekende lijn aangeven.
De regels 90-170 tekenen 32 keer alle negen punten van de lijn. Telkens wordt de lijn een beetje gedraaid. In de 2D weergave is dat te zien doordat de lijn verschuift.
De regels 140 en 150 plaatsen de 3D gegevens in de array.
Regel 160 zorgt voor het tekenen van de 2D uitvoering.
Regel 170 zorgt voor de lus en voor het verplaatsen van de grafische cursor.
Vanaf regel 180 wordt de 3D weergave verzorgd. Heeft u liever dat u zelf de hoek waarover het wijnglas draait kunt opgeven, dan kunt u regel 180, 190 en 290 veranderen in:

```
180 SCREEN 0 : INPUT "draaiing:";X
190 B=(X*3.14)/180 : SCREEN 2

290 FOR WA=1 TO 2000 : NEXT WA : GOTO 180
```

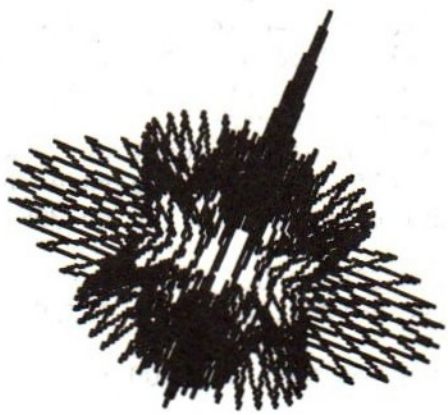
De regels 200-280 zijn vergelijkbaar met de regel 90-170. Nu hoeft er echter niets uitgerekend te worden, maar kan de computer direkt de gegevens uit de array gebruiken om te gaan tekenen. In regel 210 wordt vast een deel van het rekenwerk gedaan. Dit is eens stuk eenvoudiger dan in het kubusprogramma. Dat komt doordat er slechts over een as gedraaid hoeft te worden.
Regel 270 zorgt voor een twee-kleuren afbeelding. De bovenkant van het glas is rood, de onderkant is blauw.

Om in plaats van een wijnglas een ander figuur te krijgen hoeft u alleen de gegevens te veranderen. Let u hierbij wel op: als u

meer gegevens plaatst, moet het aantal keren dat een gegeven gelezen wordt aangepast worden (regels 60, 130 en 240). Tevens moet de dimensionering van de arrays eventueel aangepast worden (regels 30, 40 en 50).
Verander in het vorige programma de volgende regels:

```
30 DIM X(13)
40 DIM Y(13)
50 DIM Z(33,13,2)
60 FOR Q=1 TO 13
130 FOR A=1 TO 13
240 FOR A=1 TO 13
270 LINE -(PX+125,PY+170),4
300 DATA 0,10,7,-30,15,0,22,-30,40,20,
  30,-40,75,-50,30,-60,40,-90,22,-70,
  15,-100,7,-70,0,-150
```

Voor andere voorwerpen kunt u de gegevens zelf veranderen. De gegevens horen twee aan twee bij elkaar. Telkens een X-coordinaat en een Y-coordinaat van een punt van de gekromde lijn die gedraaid moet worden. Let op de ruimtereservering voor de verschillende arrays en het aantal keren dat de lussen doorlopen moeten worden.



U kunt het uzelf nog gemakkelijker maken door zelfs de gekromde lijn niet punt voor punt op te geven, maar de computer te vragen om aan de hand van een door u opgegeven formule de punten te bepalen. Op deze wijze krijgt u mooie ribbelvlakken. Vervang regel 70 van het al volgens bovenstaande aanwijzingen veranderde programma in:

```
70 Y(Q)=-50-70*(SIN(Q-9.9)/Q):X(Q)=Q*20
```

De datalijst heeft u nu eigenlijk niet meer nodig.
Probeer ook:

```
70 Y(Q)=-90-50*(SIN(((Q/2.5)-.7)/Q-.9)) : X(Q)=30*Q
```

voor een knikkerputje.
Heel mooi is de volgende formule:

```
70 Y(Q)=-90-10*(Q^2)*(SIN(Q)/Q-.9) : X(Q)=30*Q
```

Een kubus

Het volgende programma laat zien hoe een object rond alle drie de assen gedraaid kan worden. Dit programma is opvallend door de snelheid ervan. Dit ondanks de vele berekeningen die voor elk punt opnieuw gemaakt moeten worden.
Bij een draaiing om drie assen worden de berekeningen die gemaakt moeten worden om het projectiepunt van elk punt te bepalen een stuk langer. In dit programma wordt altijd eerst gedraaid om de X-as, daarna om de Y-as en tot slot rond de Z-as. Deze volgorde kan veranderd worden, maar de volgorde maakt uit voor het uiteindelijke resultaat. Door echter de juiste draaiingshoeken te kiezen kan de gebruiker elke stand van het object verkrijgen.
De projectieformule is op de volgende wijze gevonden:
De matrix van een projectie op het X-Y vlak (Z=0) na een rotatie om de Z-as, na een rotatie om de Y-as, na een rotatie om de X-as is berekend. Deze matrix is gevonden door eerst de matrices voor rotatie om de Y-as en de X-as te vermenigvuldigen. Zie hiervoor. Vervolgens wordt het resultaat hiervan vermenigvuldigd met een matrix voor een rotatie om de Z-as (over hoek C). Het resultaat is:

```
⎛COS(B)*COS(C)    SIN(A)*SIN(B)*COS(C)     -COS(A)*SIN(B)*COS(C)+ ⎞
⎜                 +COS(A)*SIN(C)           SIN(A)*SIN(C)          ⎟
⎜                                                                 ⎟
⎜-COS(B)*SIN(C)   -SIN(A)*SIN(B)*SIN(C)+   COS(A)*SIN(B)*SIN(C)+  ⎟
⎜                 COS(A)*COS(C)            SIN(A)*COS(B)          ⎟
⎜                                                                 ⎟
⎝SIN(B)           -SIN(A)*COS(B)           COS(A)*COS(B)          ⎠
```

Deze matrix wordt vermenigvuldigd met de projectiematric (Z=0).

De gebruiker kan in het volgende programma kiezen uit twee mogelijkheden: Datalijst of datainvoer.
Kiest u voor de datalijst dan hoeft u verder niets meer te doen. Het programma heeft een stel gegevens klaar om een kubus te

tekenen waarmee getoond kan worden wat het programma allemaal kan. Wilt u een ander voorwerp, dan kan dat door de gegevens in te voeren door te kiezen voor datainvoer. De gegevens die u dan invoert blijven echter niet in het programma staan. Als u veel verschillende voorwerpen met veel gegevens wilt gebruiken bij dit of een van de volgende programma's dan kunt u deze gegevens het beste opslaan in aparte bestanden. Lees hiervoor het hoofdstuk over het werken met bestanden.

```
10 SCREEN 0 : CLS :KEY OFF
20 INPUT "datalijst (D) of datainvoer (I)";D$
30 IF LEFT$(D$,1)="I" THEN 140
40 RESTORE 360
50 READ L
60 DIM X(L)
70 DIM Y(L)
80 DIM Z(L)
90 DIM T(L)
100 FOR H=1 TO L
110 READ X(H),Y(H),Z(H),T(H)
120 NEXT H
130 GOTO 250
140 INPUT "aantal punten";L
150 DIM X(L)
160 DIM Y(L)
170 DIM Z(L)
180 DIM T(L)
190 FOR H=1 TO L
200 PRINT "X-coord. punt ";H;:INPUT X(H)
210 PRINT "y-coord. punt ";H;:INPUT Y(H)
220 PRINT "z-coord. punt ";H;:INPUT Z(H)
230 INPUT "lijn (1) of verplaats (0)";T(H)
240 NEXT H
250 CLS
260 PRINT " INVOER IN GRADEN"
270 INPUT "Rotatie om X-as ";A1
280 INPUT "Rotatie om Y-as ";B1
290 INPUT "Rotatie om z-as ";C1
300 SCREEN 2 : COLOR 1,15,15 : CLS
310 GOSUB 540 : REM hoekberekening
320 GOSUB 420 : REM tekenen
330 FOR WA=1 TO 3000 : NEXT WA
340 SCREEN 0: CLS: GOTO 260
350 GOTO 340
360 DATA 21
370 DATA 0,0,0,0,0,50,0,1,50,50,0,1,50,0,0,1,50,0,50,1,50,50,50,1
380 DATA 0,50,50,1,0,0,50,1,50,0,50,1,50,50,50,0,50,50,0,1,0,50,0,0
390 DATA 0,50,50,1,0,0,50,0,0,0,0,1,50,0,
```

```
    0,1,48,2,0,1,48,48,0,1,2,48,0,1
400 DATA 2,2,0,1,48,2,0,1
410 REM tekenen
420 PSET (125,125),15
430 FOR H=1 TO L
440 GOSUB 500 :REM projectie
450 IF T(H)=0 THEN DRAW"BM=px;,=py;"
460 IF T(H)=1 THEN LINE -(PX,PY),1
470 NEXT H
480 RETURN
490 REM projectie
500 PX=(X(H)*F+Y(H)*G+Z(H)*E)+125
510 PY=(X(H)*I+Y(H)*J+Z(H)*K)+125
520 RETURN
530 REM hoekberekening
540 A=(A1*3.14)/180 : B=(B1*3.14)/180 : C=(C1*3.14)/180
550 F=COS(B)*COS(C)
560 G=SIN(A)*SIN(B)*COS(C)+COS(A)*SIN(C)
570 E=SIN(A)*SIN(C)-COS(A)*SIN(B)*COS(C)
580 I=COS(B)*SIN(C)*-1
590 J=COS(A)*COS(C)-SIN(A)*SIN(B)*SIN(C)
600 K=COS(A)*SIN(B)*SIN(C)+SIN(A)*COS(C)
610 RETURN
```

De eerste regels zorgen voor een leeg scherm en de vraag aan de gebruiker of hij zelf gegevens wil invoeren of gebruik wil maken van de al in het programma aanwezige kubus. Als er gebruik gemaakt wordt van de kubus springt de computer na het dimensioneren en inlezen van de arrays over naar regel 250 (regel 130).
De regels 140-240 zorgen ervoor dat de gebruiker zelf een vorm kan opgeven waarmee de computer vervolgens aan het werk kan gaan. De regels 260-340 zorgen in door lus dat de gebruiker de draaihoeken op kan geven. Tevens worden door middel van twee subroutines het rekenen en tekenen verzorgd.
Regel 350 is een val. Hier hoort de computer nooit te komen. Deze regel mag u ook weglaten.
De regels 360-400 vormen de datalijst. Het gegeven achter de dataopdracht van regel 360 geeft alleen het aantal te lezen gegevens in de datalijst aan.
De regels 410-480 vormen de tekensubroutine. De grafische cursor wordt naar de juiste plaats gebracht (regel 420) en in een lus worden de gegevens een voor een afgewerkt. Eerst wordt het rekenwerk gedaan. Dit gebeurt in een aparte subroutine die op regel 490 begint. Aan deze rekenregels (500 en 510) kunt u zien

dat er al wat rekenwerk gedaan is. Als er niet van te voren al wat uitgerekend was had de computer voor elk punt moeten berekenen:

```
PX = X(H)*COS(B)*COS(C)+Y(H)*(SIN(A)*SIN(B)*COS(C)+COS(A)*SIN(C))
     +Z(H)*(SIN(A)*SIN(C)-COS(A)*SIN(B)*COS(C))
PY = X(H)*-(COS(B)*SIN(C))+Y(H)*(COS(A)*COS(C)-SIN(A)*SIN(B)*
     SIN(C))+Z(H)*(COS(A)*SIN(B)*SIN(A)+SIN(A)*COS(C))
```

De berekeningen hiervoor staan grotendeels in de subroutine die buiten de tekenlus aangeroepen wordt en de regel 530-610 beslaat. Na de projectiesubroutine wordt gekeken of het laatste van de vier gegevens een 0 of een 1 is. Is het een 0 dan wordt er geen lijn getrokken, is het een 1, dan wordt er wel een lijn getrokken.

Een piramide

De mogelijkheden van het programmeren 3D zijn nog lang niet uitgeput. Neemt u het volgende programma over. Gebruik als basis het vorige programma, want er zijn veel overeenkomsten.

```
10 SCREEN 0 : CLS :KEY OFF
20 INPUT "datalijst (D) of datainvoer (I)";D$
30 IF LEFT$(D$,1)="I" THEN 140
40 RESTORE 400
50 READ L
60 DIM X(L)
70 DIM Y(L)
80 DIM Z(L)
90 DIM T(L)
100 FOR H=1 TO L
110 READ X(H),Y(H),Z(H),T(H)
120 NEXT H
130 GOTO 250
140 INPUT "aantal punten";L
150 DIM X(L)
160 DIM Y(L)
170 DIM Z(L)
180 DIM T(L)
190 FOR H=1 TO L
200 PRINT "X-coord. punt ";H;:INPUT X(H)
210 PRINT "y-coord. punt ";H;:INPUT Y(H)
220 PRINT "z-coord. punt ";H;:INPUT Z(H)
230 INPUT "lijn (1) of verplaats (0)";T(H)
240 NEXT H
```

```
250 CLS
260 PRINT " INVOER IN GRADEN"
270 INPUT "Rotatie om X-as ";A1
280 INPUT "Rotatie om Y-as ";B1
290 INPUT "Rotatie om z-as ";C1
300 INPUT "Schaalfactor X-richting";SX
310 INPUT "Schaalfactor Y-richting";SY
320 INPUT "Schaalfactor Z-richting";SZ
330 INPUT "verplaatsing X-richting";VX
340 INPUT "verplaatsing Y-richting";VY
350 SCREEN 2 : COLOR 1,15,15 : CLS
360 GOSUB 560 : REM hoekberekening
370 GOSUB 440 : REM tekenen
380 FOR WA=1 TO 3000 : NEXT WA
390 SCREEN 0: CLS: GOTO 260
400 DATA 12
410 DATA 0,0,0,0,50,0,0,1,50,0,50,1,0,0,
    50,1,0,0,0,1,25,-50,25,1,50,0,50,1
420 DATA 0,0,0,1,0,0,50,0,25,-50,25,1,50,0,0,1,0,0,50,1
430 REM tekenen
440 PSET (125,125),15
450 FOR H=1 TO L
460 GOSUB 520 :REM projectie
470 IF T(H)=0 THEN DRAW "bm=px;,=py;"
480 IF T(H)=1 THEN LINE -(PX,PY),1
490 NEXT H
500 RETURN
510 REM projectie
520 PX=(X(H)*F*SZ+Y(H)*G*SY+Z(H)*E*SZ)+125+VX
530 PY=(X(H)*I*SX+Y(H)*J*SY+Z(H)*K*SZ)+125+VY
540 RETURN
550 REM hoekberekening
560 A=(A1*3.14)/180 : B=(B1*3.14)/180 : C=(C1*3.14)/180
570 F=COS(B)*COS(C)
580 G=SIN(A)*SIN(B)*COS(C)+COS(A)*SIN(C)
590 E=SIN(A)*SIN(C)-COS(A)*SIN(B)*COS(C)
600 I=COS(B)*SIN(C)*-1
610 J=COS(A)*COS(C)-SIN(A)*SIN(B)*SIN(C)
620 K=COS(A)*SIN(B)*SIN(C)+SIN(A)*COS(C)
630 RETURN
```

Met de kennis uit het voorgaande moet u dit programma helemaal kunnen volgen. De enige extra's zijn de mogelijkheden om een voorwerp te vervormen, te verkleinen of te vergroten. Verkleinen, vergroten of vervormen gebeurt allemaal op dezelfde manier. De opgegeven punten van het object worden gemanipuleerd door een vermenigvuldigingsfactor. Is deze 1 dan wordt het voorwerp in de opgegeven richting op ware grootte (de grootte uit de datalijst) getekend. Waarden groter dan 1 vergroten het voorwerp, waarden kleiner dan 1 verkleinen het.
Kiest u verschillende waarden voor de verschillende richtingen (X, Y en Z) dan vervormt u het voorwerp. De verplaatsing vindt plaats door de variabelen VX en VY.

De waarden die in de datalijst staan zorgen voor het tekenen van de omtrekken van een piramide. De waarden worden in groepen van 4 opgegeven. De eerste drie waarden geven een X-, Y- en Z-coordinaat. De vierde waarde geeft aan of er een lijn getrokken moet worden of dat naar een ander punt gesprongen moet worden. Op deze manier kunnen eenvoudig verschillende, ook ingewikkelde, voorwerpen getekend worden, zonder dat deze met een enkele ononderbroken lijn getekend hoeven te worden.
Het vinden van de benodigde gegevens is ook bij een niet symetrisch figuur niet moeilijk. U legt het voorwerp (of een maquette van het voorwerp) op een rasterpapiertje. De richting links-rechts noemt u de X-richting. Voor elk punt moet eerst de afstand van links naar rechts opgegeven worden.
De richting voor-achter noemt u de Z-richting.
Vlak voor het voorwerp (bij voorkeur op de X-as) plaatst u een stukje glas of een doorzichtige film/plastiek, met daarop een even groot rooster getekend. De richting boven-onder noemt u de Y-richting.
Op deze wijze kunt u van alle ter zake doende punten van het voorwerp de coordinaten bepalen. Bedenk bij het invoeren dat na de coordinaten van elk punt, telkens een 1 moet komen als er een lijn naar dat punt getrokken moet worden en een 0 als er alleen naar dat punt gesprongen moet worden.
Heeft u heel ingewikkelde figuren dan kunt u gebruik maken van een bochtenmeter. Dit voorwerp bestaat uit een stel ijzeren staafjes vlak naast elkaar. Door de verschuifbare staafjes allemaal tegen het voorwerp aan te leggen, de staafjes vast te zetten en de stand van de staafjes vervolgens op een roosterpapier af te lezen, zijn ook moeilijke vormen snel te bepalen. Dit stuk gereedschap is bij elke ijzerhandel verkrijgbaar.
Met behulp van dit programma en uw eigen fantasie kunt u veel voorwerpen die u goed kent, bekijken op het beeldscherm en onderzoeken of ze mooier zouden zijn als ze bijvoorbeeld wat langer, breder of hoger waren geweest.

BIJLAGEN

ASCII-Codes

Alle in deze tabel genoemde karakters kunt u op het scherm plaatsen door middel van het toetsenbord. U drukt daarvoor een toets of een combinatie van toetsen in.
Tevens kunnen deze karakters verkregen worden door de CHR$ opdracht. U plaats de juiste code tussen haakjes achter de opdracht:

```
PRINT CHR$(205)
```

zorgt voor het afdrukken van een klein driehoekje.
Alleen de codes 1 tot en met 31 vragen een andere behandeling. Deze codes zijn in de officiele ASCII-set niet bedoeld voor het weergeven van symbolen of letters. Om deze codes daar toch voor te kunnen gebruiken heeft de MSX van deze codes een dubbele code gemaakt. U moet voor elke code van 1 tot en met 31 eerst de code 1 invoeren en vervolgens de code met 64 verhogen.

```
PRINT CHR$(1);CHR$(65)
```

zorgt voor een lachend gezichtje op uw scherm.

CODE	TEKEN	CODE	TEKEN	CODE	TEKEN	CODE	TEKEN
0	(nul)	15	☼	30	\	45	—
1	☺	16	┼	31	+	46	.
2	☻	17	┴	32	(spatie)	47	/
3	♥	18	┬	33	!	48	0
4	♦	19	┤	34	"	49	1
5	♣	20	├	35	#	50	2
6	♠	21	┼	36	$	51	3
7	•	22	│	37	%	52	4
8	■	23	—	38	&	53	5
9	○	24	┌	39	'	54	6
10	◙	25	┐	40	(	55	7
11	♂	26	└	41	)	56	8
12	♀	27	┘	42	*	57	9
13	♪	28	×	43	+	58	:
14	♫	29	/	44	,	59	;

CODE	TEKEN	CODE	TEKEN	CODE	TEKEN	CODE	TEKEN
60	<	109	m	158	₧	207	▐
61	=	110	n	159	ƒ	208	▌
62	>	111	o	160	á	209	▀
63	?	112	p	161	í	210	▄
64	@	113	q	162	ó	211	▖
65	A	114	r	163	ú	212	▗
66	B	115	s	164	ñ	213	▘
67	C	116	t	165	Ñ	214	▝
68	D	117	u	166	ª	215	▚
69	E	118	v	167	º	216	△
70	F	119	w	168	¿	217	┼
71	G	120	x	169	⌐	218	ω
72	H	121	y	170	¬	219	▉
73	I	122	z	171	½	220	▊
74	J	123	{	172	¼	221	▋
75	K	124	¦	173	¡	222	▌
76	L	125	}	174	«	223	▍
77	M	126	~	175	»	224	α
78	N	127		176	Ã	225	β
79	O	128	Ç	177	ã	226	Γ
80	P	129	ü	178	Ĩ	227	Π
81	Q	130	é	179	ĩ	228	Σ
82	R	131	â	180	Õ	229	σ
83	S	132	ä	181	õ	230	µ
84	T	133	à	182	Ũ	231	γ
85	U	134	å	183	ũ	232	Φ
86	V	135	ç	184	Ij	233	Θ
87	W	136	ê	185	ij	234	Ω
88	X	137	ë	186	³/₄	235	δ
89	Y	138	è	187	∽	236	∞
90	Z	139	ï	188	◇	237	Ø
91	[	140	î	189	‰	238	∈
92	\	141	ì	190	¶	239	∩
93	]	142	Ä	191	§	240	■
94	^	143	Å	192	▬	241	±
95	_	144	É	193	▙	242	≥
96	`	145	æ	194	▟	243	≤
97	a	146	Æ	195	▬	244	⌠
98	b	147	ô	196	▪	245	⌡
99	c	148	ö	197	▟	246	÷
100	d	149	ò	198	▌	247	≈
101	e	150	û	199	▞	248	°
102	f	151	ù	200	▙	249	•
103	g	152	ÿ	201	▌	250	·
104	h	153	Ö	202	▙	251	√
105	i	154	Ü	203		252	ⁿ
106	j	155	¢	204	▒	253	²
107	k	156	£	205	▼	254	■
108	l	157	¥	206	▲	255	

FOUTMELDINGEN

Wanneer de MSX in een programma een fout ontdekt stopt hij het programma en plaatst een bericht op het scherm. Aan de hand van dit bericht kan de gebruiker sneller zien waar welke fout zit. Elke foutmelding heeft een eigen nummer. Dit nummer kunt u gebruiken om het programma bij het ontdekken van een fout naar een bepaalde subroutine te laten springen. U gebruikt daarvoor de opdracht

ON ERROR GOTO ...

Achter GOTO plaatst u het regelnummer waar de computer naartoe moet als er een fout geconstateerd wordt.

De foutmeldingen staan in alphabetische volgorde.

56 - Bad file name (slechte bestandsnaam)

De naam van het bestand is niet toegestaan.
De naam van het apparaat is bij deze OPEN, SAVE of LOAD opdracht niet toegestaan.

52 - Bad file number (slecht bestandsnummer)

U heeft een opdracht of statement gebruikt dat naar een niet bestaand bestand verwijst, of buiten het bereik van MAXFILES ligt.
U heeft PRINT# gebruikt met een bestandsnummer dat nog niet geopend is.

17 - Can't CONTINUE (kan niet doorgaan)

U probeert het programma te CONTinueren terwijl het programma gestopt is met een fout, of gestopt is met END of verbeterd is na de onderbreking.
Het door u gevraagde programma bestaat niet.
Het programma bevatte al een CONT opdracht.

19 - Device I/O error (fout in de in- of uitvoer)

Er is iets mis met de invoer vanaf de cassetterecorder. Of de recorder werkt niet goed, of de cassette is niet goed. Het programma kan niet geladen worden.
Het laden is ergens onderbroken.
U gebruikt een onjuiste I/O poort.
De volumeregeling van de cassetterecorder is niet goed.
U kunt deze fout niet herstellen. U zult de cassette moeten terugspoelen, de oorzaak eventueel weghalen en opnieuw proberen te laden.

57 - Direct statement in file (direkte opdracht in bestand)

Binnen het bestand dat geladen wordt, komt de computer een direkte opdracht tegen (zonder regelnummer). Het laden wordt gestopt.
U probeert een bestand als BASIC programma te laden terwijl het dit niet is.

11 - Division by zero (deling door nul)

Het is onmogelijk door 0 te delen.
Let op! Variabelen die niet gedefinieerd zijn, worden door de MSX altijd beschouwd als nul. Als u door een niet gedefinieerde variabele deelt, deelt u door nul. Een foutmelding zal het gevolg zijn.

54 - File already open (bestand is al open)

U probeert een bestand te openen dat al open is.

53 - File not found (bestand niet gevonden)

Een LOAD of OPEN opdracht verwijst naar een niet bestaand bestand.

59 - File not OPEN (bestand is niet open)

U probeert gegevens weg te schrijven naar of te lezen uit een bestand dat nog niet geopend is.

12 - Illegal direct (niet toegestaan direkt)

U probeert een opdracht direkt aan de computer te geven, terwijl deze opdracht alleen in een programma opgenomen kan worden.

5 - Illegal function call (niet toegestane funktieaanroep)

U geeft een verkeerde parameter aan een wiskundige of stringfunktie. Dat wil zeggen dat bij de gebruikte opdracht deze waarde niet opgegeven mag worden.
De waarde van een funktie ligt buiten het beschikbare bereik.

55 - Input past end (invoer voorbij het einde)

U tracht gegevens uit een bestand te lezen terwijl alle gegevens al gelezen zijn. Deze foutmelding komt ook voor als het bestand door een fout in de programmatuur helemaal geen gegevens bevat.

51 - Internal error (interne fout)

Iets in de computer werkt niet goed. Meestal werkt het BASIC vertaalprogramma niet goed.

25 - Line buffer overflow (regelbuffer vol)

De buffer voor regels is vol. De ingevoerde regel bevat teveel karakters.

24 - Missing operand (parameter ontbreekt)

U heeft te weinig parameters bij een opdracht gebruikt.

1 - NEXT without FOR (NEXT zonder FOR)

De computer komt een NEXT opdracht tegen zonder de FOR opdracht eerder gezien te hebben. Dit kan ook gebeuren als u door een GOTO of GOSUB naar een regel binnen een FOR...NEXT... lus verwijst.
De variabele achter de NEXT opdracht komt niet overeen met de variabele achter de FOR opdracht.

21 - No RESUME (geen RESUME)

In de foutherstellingsroutine staat geen RESUME waar de computer dit verwacht. U moet elke foutherstelingsroutine (zie begin van deze paragraaf over ON ERROR GOTO) laten eindigen met END, RESUME of ON ERROR 0.

4 - Out of DATA (geen DATA meer)

U probeert een gegeven te lezen door middel van READ terwijl alle gegevens al gelezen zijn.
U bent de DATA opdrachten vergeten.

7 - Out of memory (geen geheugen genoeg)

De computer heeft geen geheugen genoeg om uw opdracht uit te voeren. Dit kan veroorzaakt worden door een te lang programma, teveel FORs, teveel GOSUBs, teveel variabelen, een te lange array, teveel arrays of teveel funkties.

14 - Out of string space (geen stringruimte meer)

Het gebied in het geheugen dat gereserveerd is voor het werken met strings is te klein voor het door u beoogde doel. U kunt dit gebied groter maken met CLEAR.

6 - Overflow (overstroming)

De variabele, funktie of statement kan dit getal niet aan, het is te groot.

22 - RESUME without error (RESUME zonder fout)

De computer komt een RESUME opdracht tegen zonder een bijbehorende ON ERROR opdracht.
Dit kan ook gebeuren als het hoofdprogramma geen END opdracht heeft of als er met een GOTO naar de RESUME verwezen wordt.

3 - RETURN without GOSUB (RETURN zonder GOSUB)

De computer komt een RETURN opdracht tegen zonder eerst een GOSUB opdracht gehad te hebben. Dit kan gebeuren door het ontbreken van een END aan het einde van het hoofdprogramma of door een GOTO naar de subroutine.

10 - Redimensioned array (array wordt weer gedimensioneerd)

U probeert een al bestaande array opnieuw te definieren.
U gebruikt ongedefinieerde arrayvariabelen die achteraf gedimensioneerd worden.

16 - String formula too complex (stringformule te complex)

Een stringvariabele is te gecompliceerd.

15 - String too long (string te lang)

De door u gebruikte string is langer dan 255 lettertekens.

9 - Subscript out of range (index buiten het bereik)

U geeft een arrayvariabele een hoger nummer dan volgens de dimensionering van de array mogelijk is.
U kunt zonder dimensionering arrays gebruiken met een maximale lengte van 10. Als u een arrayvariabele een nummer groter dan 10 geeft, moet u eerst de array dimensioneren.

2 - Syntax error (grammatika fout)

U heeft een fout in de BASIC grammatika gemaakt.

13 - Type mismatch (type klopt niet)

U gebruikt de verkeerde waarden. In een LET opdracht staan aan de twee zijden van het is-gelijk-teken verschillende grootheden (bijvoorbeeld een numerieke variabele met een stringwaarde).
U probeert een rekenkundige of logische bewerking uit te voeren op een string.

8 - Undefined line number (ongedefinieerd regelnummer)

In een GOTO, GOSUB, IF..THEN.., DELETE of RESUME opdracht gebruikt u een regelnummer dat niet bestaat. Dit kan soms pas tot uiting komen bij het hernummeren van een programma (RENUM).

18 - Undefined user function (ongedefinieerde gebruikersfunktie)

U roept een FN funktie aan zonder deze eerst door middel van DEF FN gedefinieerd te hebben.

23 - Unprintable error (niet afdrukbare fout)

Er is een fout ontdekt, maar er is geen foutmelding.

26/49 - Unprintable error (niet afdrukbare fout)

Gereserveerd voor latere uitbreidingen.

60/255 - Unprintable error (niet afdrukbare fout)

Deze foutmeldingen zijn gereserveerd voor definities door de gebruiker.

20 - Verify error (controle fout)

Het programma in het geheugen van de MSX is niet gelijk aan het programma op de cassette.

INDEX

ANDERE UITGAVEN VAN WOLFKAMP:

* APPLE HANDBOEK *
Van Kampen / De Kuiper / Huyser
ISBN 90-70556-01-4 Fl. 97,50

* MACHINETAAL PROGRAMMEREN APPLE *
Inman / Inman
ISBN 90-70556-03-0 Fl. 49,50

* OP AVONTUUR MET DE BBC *
Van Engelen
ISBN 90-70556-09-X Fl. 49,50

* USURPATOR SCHAAK 6502 / 6800 *
Muller
ISBN 90-70556-06-5 Fl. 19,50

* WERKEN MET ASSEMBLERS *
De Vries
ISBN 90-70556-08-1 Fl. 59,50

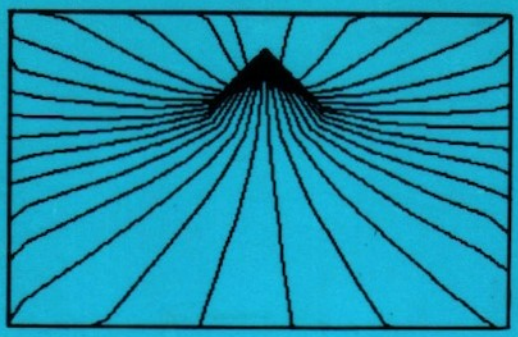
* VOOR GALG EN RAD, ZX SPECTRUM *
Van Engelen
ISBN 90-70556-07-1 Fl. 27,50

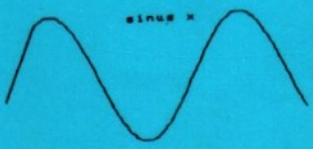
* PRODOS WERKBOEK *
Van Kampen / Huyser
ISBN 90-70556-11-1 Fl. 24,50

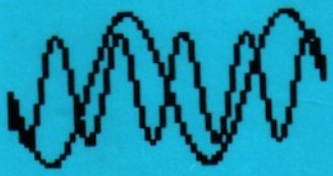
* INTRODUCTIE IN CP/M *
Meerman e.a.
ISBN 90-70556-10-3 Fl. 24,50

* INTRODUCTIE IN MS-DOS *
Paijmans
ISBN 90-70556-13-8 Fl. 24,50

* COMMODORE 64 BINNENSTE BUITEN *
De Jong
ISBN 90-70556-12-X Fl. 39,50







WOLFKAMP