Szczepanowski

Het eerste boek voor de CPC-computers

CPC Bibliotheek 9

DATA BECKER
NEDERLANDS*

Het eerste boek voor de CPC-computers

Szczepanowski

Het eerste boek voor de CPC-computers

CPC Bibliotheek 9

Oorspronkelijke titel: CPC 464 für einsteiger

Vertaling: W.B. de Jong, Utrecht
Redaktie: A.W. Bruna & Zoon
Produktie: A.W. Bruna & Zoon
Zetwerk: Mat-zet, Beesd
ISBN 90 229 3356 3
D/1985/0939/234

Boeken en programma's van

DATA BECKER
NEDERLANDS*

worden in de handel gebracht door:
A.W. Bruna & Zoons Uitgeversmij. b.v.,
Postbus 8411, 3503 RK Utrecht
en
A.W. Bruna & Zoon n.v.
Antwerpsesteenweg 29a, 2630 Aartselaar.

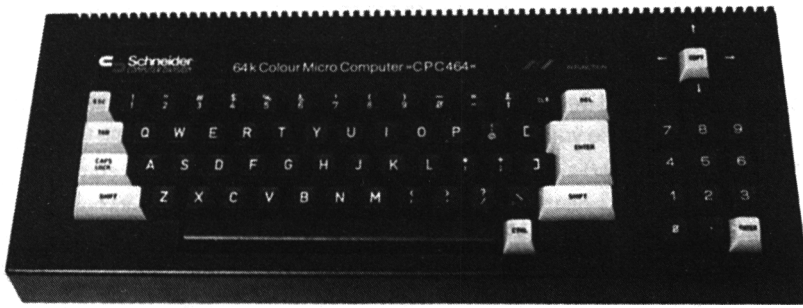
Belangrijke verwijzing.

De in dit boek weergegeven schakelingen, werkwijzen en programma's worden zonder de nodige inachtnemingen van de auteurs-rechten meegedeeld. Ze zijn uitsluitend voor amateur- en leerbestemmingen bedoeld en mogen derhalve geen andere bestemming hebben.

Alle schakelingen, technische aanwijzingen en programma's die in dit boek staan, zijn door de auteur met de grootste zorg omringt en samengesteld en door het inschakelen van een goed controlesysteem gereproduceert. Ondanks deze maatregelen is het niet te voorkomen dat er fouten zijn gemaakt. DATA BECKER kan derhalve niet verantwoordelijk gesteld worden voor eventuele fouten die in dit boek staan. Voor mededelingen over deze fouten is de uitgever echter zeer erkentelijk.

Inhoudsopgave

Het toetsenbord



In dit hoofdstuk leert U de toepassingen van de 74 toetsen van Uw CPC464. Met behulp van deze toetsen zult U alle verborgen mogelijkheden van Uw home-computer ontdekken. Hetzij bij het invoeren van teksten, het vervaardigen van beelden (grafieken), het versturen van aanwijzingen of het berekenen van wiskundige problemen, al deze zaken zullen U na het bestuderen van dit hoofdstuk nog weinig moeilijkheden opleveren. Als U eenmaal alle toetsen beheerst, zal de CPC464 U als een volgzame partner bij het oplossen van vele problemen terzijde staan.

Het beheren van het toetsenbord is niet slechts een zaak voor een echte computer-freak, die tenslotte van elke bit van zijn computer zelfs de voornaam wil weten, maar is het ook voor hen, die de computer alleen maar met bedrijfsklare programma's wil 'voeren'. Iemand, die tot de laatst genoemde gebruikersgroep behoort, moet niet vertwijfeld naar zijn handboek grijpen als er op het beeldscherm de standaarduitdrukking 'DRUK DE ENTER TOETS IN' verschijnt.

Kort gezegd, zelfs als men slechts af en toe van het gereedschap gebruik maakt, moet men op de hoogte zijn van de indeling van het toetsenbord. De vader moet tenminste weten, hoe het beste spelprogramma van zijn zoon geladen moet worden, als die zojuist de woning heeft verlaten.

Maar geen angst, U hoeft zich niet onmiddellijk bij een volkshogeschool aan te melden om een cursus schrijfmachine te volgen. De meeste ervaren amateur-programmeurs werken met het twee-vinger zoek systeem. Wie eenmaal met zijn toetsenbord vertrouwd is, zal zich erover verwonderen hoe snel hij over het toetsenbord suist.

In dit hoofdstuk worden in het bijzonder de witte hulptoetsen behandeld. Deze toetsen zijn erg belangrijk, omdat men daarmee o.a. teksten in kan voeren, programma's kunnen worden onderbroken en commando's naar de computer overgebracht.

ALGEMEENHEDEN OVER HET TOETSENBORD

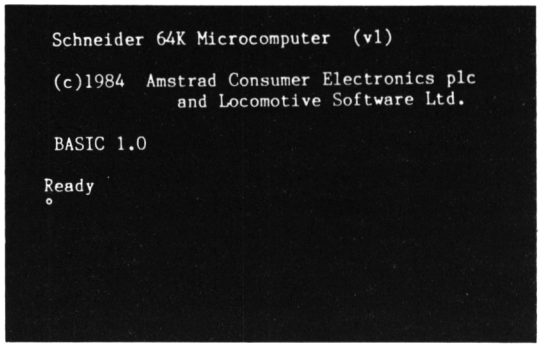
Op het eerste gezicht lijkt het toetsenbord van de CPC464 op het normale schrijfmachine-toetsenbord. Maar als men er wat nauwkeuriger naar kijkt ziet men al spoedig een aantal afwijkingen:

- Het toetsenbord kent niet de typisch Franse accenten, nog de Duitse Umlaut en ook niet het omgekeerde Spaanse vraagteken. De letters Z en Y staan op de plaats waar deze volgens de Amerikaanse ASCII-norm behoren te staan.

- Er zijn een aantal toegevoegde toetsen (hulptoetsen), waarvan de funktie later wordt verklaard.

Het is niet zo slim om er op los te experimenteren voordat men de betekenissen van de toetsen goed kent. Een verkeerd gebruik van het toetsenbord zal weliswaar geen rookwolken te voorschijn roepen, maar U kunt toch voor vreemde verrassingen komen te staan. Wacht dus geduldig op de praktische uiteenzetting op de volgende bladzijden.

EN DAN MOET HET MAAR



Nu wordt het tijd om Uw CPC464 startklaar te maken. Daarvoor hoeft U alleen maar de schakelaar aan te zetten. U vindt twee schakelaars, een aan de monitor en een aan de computer. Ze moeten alle twee aangezet worden. Nadat U de schakelaars hebt aangezet, verschijnt er op het beeldscherm een mededeling, waarmee tot uitdrukking wordt gebracht, dat de computer in staat is om gegevens te ontvangen. De eerste regel van deze mededeling zegt, dat U met de Schneider-computer werkt waarvan het hoofdgeheugen 64Kbyte groot is (dat zijn 65535 tekens). Van deze 65535 geheugenplaatsen kunt U voor Uw eigen BASIC programma's slechts 43533 gebruiken. De rest van deze geheugenruimte heeft de computer zelf nodig voor de besturingen en voor de computertaal BASIC. De daarop volgende regel deelt mee waar de computer eigenlijk vandaan komt,

dat is de firma AMSTRAD in Engeland. Van de firma LOCOMOTIVE SOFTWARE komt het geïntegreerde BASIC vandaan dat, zoals u later nog zult zien, tot grote prestaties in staat is. Dit BASIC heeft het volgnummer 1.0. Als er eens veranderingen of verbeteringen aangebracht worden, zal dit volgnummer ook veranderd worden.

In de laatste regel staat de mededeling 'READY', waarmee tot uitdrukking wordt gebracht dat de computer in staat is om commando's te ontvangen. Dit 'READY' signaleert dat de computer bedrijfsklaar is.

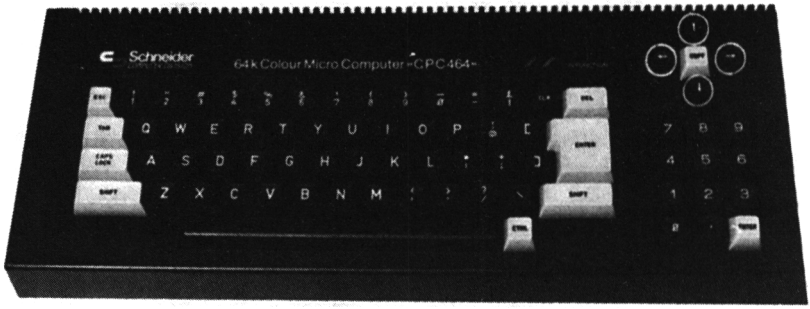
Nu naar het kleine vierkant dat onder de mededeling READY staat. Dit is een hulp voor het beeldscherm. Laten we aannemen, dat men iets op het beeldscherm wil schrijven, dan is het nodig dat men precies weet waar dit op het beeldscherm zal verschijnen. Dit merkteken, dat in vaktaal CURSOR heet, helpt hierbij door deze plaats aan te geven.

DE DRIE-VINGER-GREEP

In tal van situaties is het nodig, de computer in een andere toestand te brengen. Bij veel andere computers is het dan nodig om het apparaat uit en daarna weer aan te zetten. Dit geeft echter een overbodige belasting aan de hoogwaardige elektronika van Uw computer. Om dit te vermijden kan men de CPC464 door het bedienen van een aantal toetsen weer in de inschakel-toestand brengen. Hiervoor moet men drie toetsen indrukken. De SHIFT, CTRL en ESC toets. Men moet er echter op letten dat de ESC toets als laatste wordt ingedrukt terwijl de SHIFT en de CTRL toetsen ingedrukt blijven. Probeert U eens deze zogeheten drie-vinger-greep. U zult dan opmerken dat hetzelfde beeld verschijnt als bij het inschakelen van de computer.

Dit terugzetten van de computer wordt ook wel RESET of koude start genoemd. Dit moet men echter met de nodige zorgvuldigheid doen omdat bij een verkeerde handeling ook alle aanwezige BASIC programma's verdwijnen.

DE CURSOR TOETSEN



Deze toetsen bevinden zich rechts boven het cijferblok. De pijlen op deze toetsen geven de richtingen aan waarin de CURSOR zich moet bewegen. Druk nu eens 20 keer de 'rechts' toets in. De cursor gaat dan 20 plaatsen naar rechts en wacht daar geduldig op wat er gebeuren gaat.

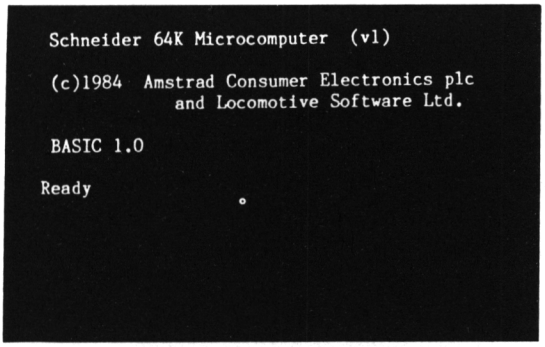
```
Schneider 64K Microcomputer  (v1)

(c)1984  Amstrad Consumer Electronics plc
           and Locomotive Software Ltd.

BASIC 1.0

Ready
```

Wat nu, als de cursor de uiterste rechterkant van het beeldscherm heeft bereikt? Probeer dat zelf eens uit, en druk daartoe nog eens 20 keer op de 'rechts' toets. Na de twintigste keer zal de cursor aan de linkerzijde van het beeldscherm verschijnen en wel op de volgende regel.

```
Schneider 64K Microcomputer  (v1)

(c)1984  Amstrad Consumer Electronics plc
           and Locomotive Software Ltd.

BASIC 1.0

Ready

```

Het is echter omslachtig om twintig keer op de cursor toets te drukken om in het midden van het beelscherm terecht te komen. Deze procedure kan men iets vereenvoudigen. Als U de toets ingedrukt houdt zal de cursor vanzelf naar rechts bewegen. Dit geldt ook voor alle andere toetsen. Probeer dat zelf eens uit, en houdt daarbij de cursor-rechts toets ingedrukt. U zult dan zien dat de cursor naar het rechterdeel van het beelscherm springt.

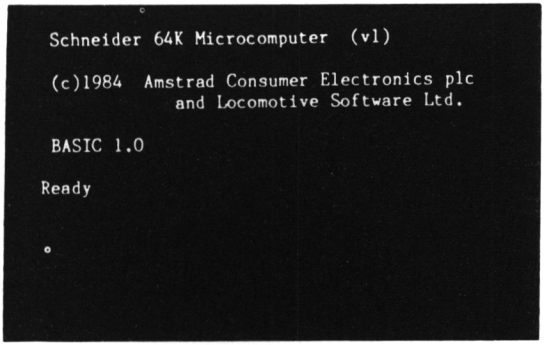
Om nu de cursor naar de andere kant te laten gaan gebruikt men de cursor 'links' toets. Probeer nu ook eens deze toets uit. De cursor beweegt zich nu dus in de andere richting.



Vanzelfsprekend beweegt de cursor ook hierbij naar links als men de cursor toets ingedrukt houdt, zonder dat men steeds opnieuw de toets hoeft te drukken.

Hoe kan men nu de cursor weer in de begin positie (onder het woord READY) brengen? Hiervoor kan men natuurlijk de cursor toets net zolang ingedrukt houden tot de gewenste plaats bereikt is. Maar dat is niet de bedoeling geweest van de uitvinder. De cursor toetsen BOVEN en ONDER werken op dezelfde manier als de toetsen RECHTS en LINKS.

Beweeg de cursor eens naar de meest linkse kant van het beeldscherm. Daar wacht de cursor geduldig tot U hem naar een andere plaats stuurt. Druk nu eens drie keer de toets CURSOR-ONDER in. U zult dan zien dat de cursor drie plaatsen naar beneden gaat.



Probeer nu eens de cursor in deze derde regel van het beeldscherm te bewegen. Gebruik hiervoor de toets CURSOR-BOVEN.

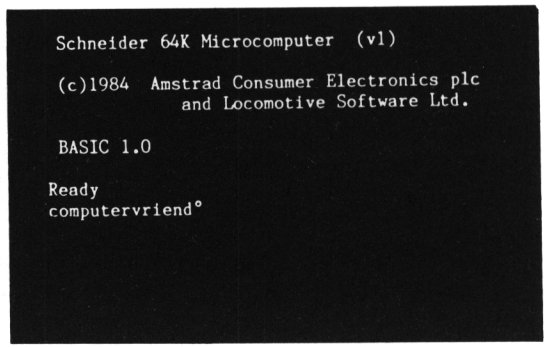
Ook hierbij zal de cursor vanzelf regel voor regel naar boven of naar beneden bewegen, zolang U de toets ingedrukt houdt.

Om een gevoel voor deze cursor toetsen te ontwikkelen, moet U eens proberen, met de cursor een onzichtbaar vierkant te tekenen. Dwz. U beweegt de cursor achtereenvolgens rechts/onder/links/boven of andersom dus links/onder/rechts/boven. Het beheersen van de cursor toetsen is erg belangrijk, omdat U later bv. correcties in een programma bliksemsnel moet kunnen uitvoeren. Maar ook hier geldt: oefening maakt de meester: en ook die is, voorzover we weten, nog niet uit de hemel gevallen.

EDITEREN MET DE CURSOR TOETSEN

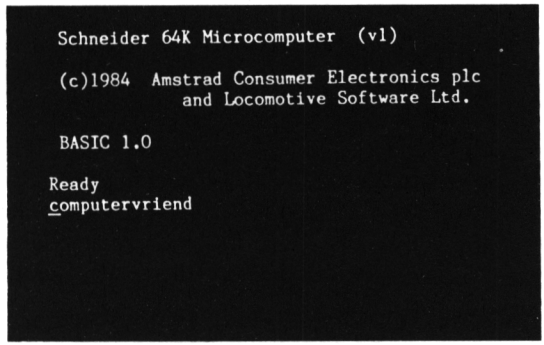
Hier duikt een vreemd woord uit de data-verwerking op, waarvan de betekenis aan sommige lezers, in het bijzonder bij de beginner, nog onbekend is. Met editeren wordt bedoeld, het bewerken van teksten, bv. teksten toevoegen of veranderen. Men spreekt al van editeren, als U een paar woorden op het beeldscherm af wilt drukken.

Bij vele computers kan men de cursor naar willekeur over het hele beeldscherm bewegen, teksten invoegen, en daarna weer verder bewegen. De CPC464 heeft echter een belangrijke eigenaardigheid. U kunt de cursor zolang willekeurig verplaatsen tot U een karaktertoets (lettertoets) hebt ingedrukt. Van nu af aan kunt U de cursor slechts binnen het gebied dat met karakters is gevuld, laten bewegen. Dit is een grijze theorie, die we nu in praktijk gaan brengen. Zet de computer om te beginnen in de begin stand (SHIFT-CTRL-ESC). Schrijf nu het woord 'computervriend' op het beeldscherm, ook als het zoeken naar de letters iets langer duurt.

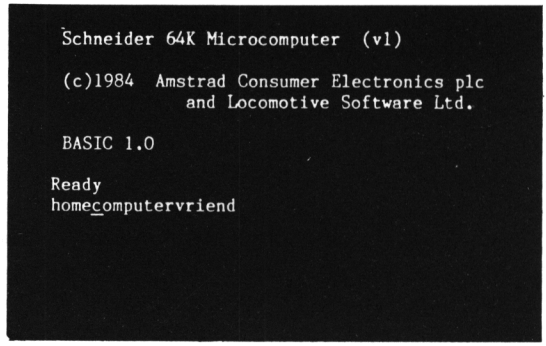


Nu bevindt de cursor zich achter de laatste letter van het woord en weigert een andere richting heen te gaan dan naar de linkerkant van het laatste karakter (letter). Probeert U dat eens ook als het geen resultaat heeft. De cursor kan nu alleen nog maar gebruikt worden om de tekst te bewerken. En hiervoor mag en kan men de tekst niet meer verlaten. Beweeg de cursor eens helemaal naar links, door de cursor toets LINKS ingedrukt te houden. Als de cursor aan de linkerkant van het beeldscherm is aangekomen zal hij daar stoppen alsof er een muur staat. Hetzelfde resultaat krijgt men als de cursor naar rechts bewogen wordt. Achter de laatste letter schijnt ook een muur te staan. Waarom deze beperking van de cursorbeweging? Om dit te begrijpen moet U weten waarom U eigenlijk iets op het beeldscherm schrijft. Hiervoor kunnen verschillende oorzaken zijn, zoals bv. het programmeren. Voor het schrijven van een programma, waarop we later terug zullen komen, moeten er regels ingevoerd worden. Elke programmaregel wordt door middel van een bepaalde toets (ENTER) aan de computer meegedeeld. Het is volkomen nutteloos om een regel op een andere plaats van het beeldscherm voort te zetten. Het invoeren van deze regel in de computer maakt de cursor weer vrij.

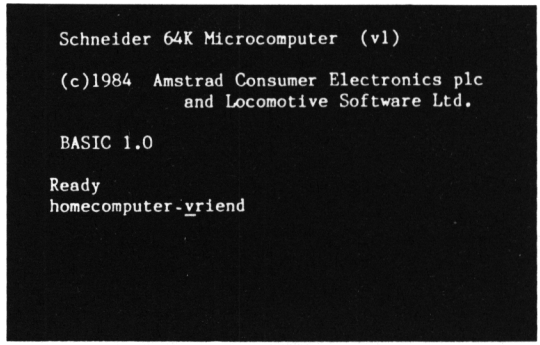
Gaan we nu weer verder met onze 'computervriend'. Op elke willekeurige plaats in dit woord kan men een letter toevoegen. Daartoe moet men de cursor op de letter plaatsen waar men een andere letter of een teken wil toevoegen. Deze zogenoemde 'toevoeg modus' is altijd ingeschakeld. Bij sommige andere computers moet deze modus eerst ingeschakeld worden. Veranderen we nu het woord 'computervriend' in 'home-computervriend', dan zetten we eerst de cursor naar de linkerkant van het beeldscherm.



De cursor bevindt zich nu op de plaats van de letter 'c'. Hier moet nu de lettergreep 'home' worden toegevoegd. Druk nu achter elkaar de letters 'h', 'o', 'm', en 'e' in.



Met elke letter wordt de tekst links van de cursor een plaats naar links geschoven. De cursor blijft echter op de 'c' staan. Nu moeten we nog het verbindings-streepje inbrengen. U weet zeker allang hoe men dat moet doen. U zet de cursor op de 'v' en drukt de toets '-' (rechts van de toets 0) in.



Als men de cursor nu naar de uiterste rechterkant van de tekst zet, kan het invoeren van de tekst worden voortgezet. Er is inmiddels al meegedeeld, dat het invoeren van de tekst met de toets 'ENTER' wordt afgesloten. Als U echter in dit voorbeeld deze toets indrukt, zal de CPC tegensputteren. Met de mededeling 'SYNTAX ERROR' zegt hij 'wat je nu doet begrijp ik niet'. Dat is ook duidelijk want de enige taal die Uw CPC464 verstaat is BASIC, en die moet U eerst nog leren. Tot U zover bent, zijn er nog maar een paar bladzijden te doen.

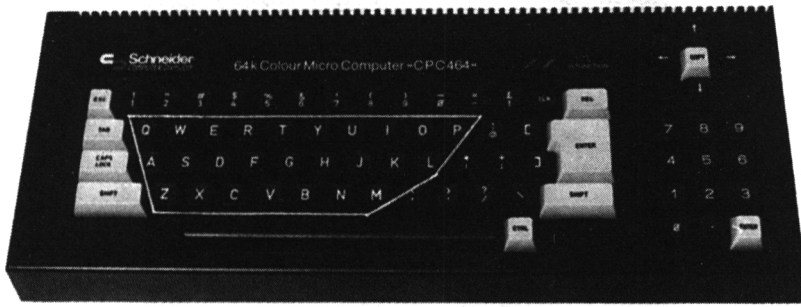
U hebt nu de 'invoegmodus' leren kennen. Als U een beetje oefent, zult U dit gauw onder de knie hebben.

WISSEN VAN HET BEELDSCHERM

Laten we aannemen dat alles wat zich op het beeldscherm bevindt, onbruikbare onzin is. Men veegt dan eenvoudig alles weg. HALT!! niet met spons en zeem, daarmee kan men

alleen de stof van het beeldscherm vegen, maar niet de ingetikte teksten en wat er in het geheugen van de computer staat. Tenslotte is het voor een computer die ca. 1 miljoen bewerkingen per seconde uit kan voeren, kinderspel om 'even' zo'n 1000 tekens van het beeldscherm af te halen. Veel computers hebben voor het wissen van het beeldscherm een aparte toets. Bij de CPC464 is dit niet zo comfortabel, omdat er hier vier toetsen moeten worden ingedrukt. Het BASIC van de CPC464 heeft hiervoor een opdracht die het hele beeldscherm schoon veegt. Deze opdracht luidt 'CLS', wat, zoals alle opdrachten uit het Engels (CLEAR SCREEN) komt. Zoals U inmiddels al weet, moet U deze opdracht ook met de toets 'ENTER' afsluiten. Geef dus de opdracht 'CLS' gevolgd door de toets 'ENTER'. Uw CPC464 gehoorzaamt onmiddellijk en wist het hele beeldscherm. De mededeling 'READY' betekent dat Uw computer weer op verdere instructies wacht. Dit commando moet U proberen te onthouden, omdat dat in de rest van dit boek regelmatig gebruikt wordt.

DE TOETSEN MET LETTERS



Zoals al eerder is meegedeeld, lijkt het toetsenbord, behoudens een enkele uitzondering, op het toetsenbord van een normale schrijfmachine. We concentreren ons allereerst op de letters. Als U een willekeurige toets indrukt dan verschijnt de bijbehorende letter op de plaats van de cursor op het beeldscherm. Voordat we deze toetsen gaan gebruiken is het aan te bevelen om het beeldscherm met 'CLS' te wissen.

Nu hebben we een 'schoon' beeldscherm waarop we kunnen editeren. Geef nu echter geen ongeordende teksten aan het geheugen van Uw computer in de hoop dat hij dat ter zijner tijd geordend kan weergeven. Zo eenvoudig is dat niet. Datgene wat U invoert, wordt voorlopig alleen in het beeldschermgeheugen vastgehouden. Hoe men deze teksten in het hoofdgeheugen kan bewaren, wordt behandeld als we BASIC bespreken.

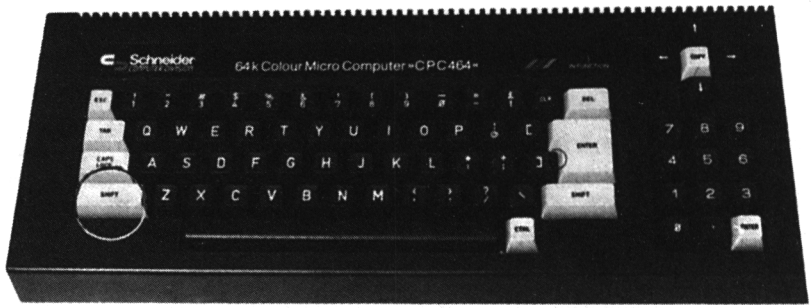
Schrijft U nu eens alle letters op het beeldscherm, in de alfabetische volgorde. U zult dan zien hoe moeilijk het is om de een of andere letter te vinden, of men moet toevallig secretaresse zijn. Op het beeldscherm bevinden zich nu alle kleine letters.

```
Schneider 64K Microcomputer  (v1)

(c)1984  Amstrad Consumer Electronics plc
            and Locomotive Software Ltd.

BASIC 1.0

Ready
abcdefghijklmnopqrstuvwxyz°
```

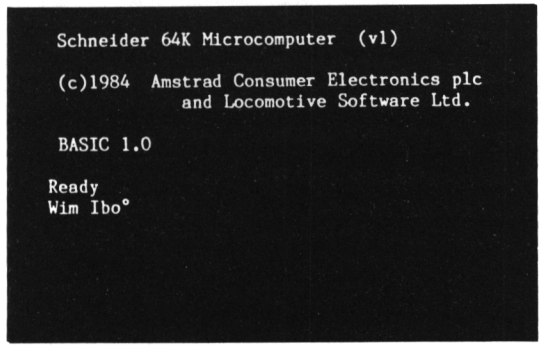
Wis nu nog een keer het beeldscherm schoon en geef nu nog eens alle letters weer. Maar nu niet in de alfabetische volgorde, maar op de manier waarop ze op het toetsenbord zijn aangebracht. Dus eerst de bovenste rij (Q tot P), dan de middelste rij (A tot L) en tenslotte de onderste rij (Z tot M). Deze volgorde intikken gaat veel eenvoudiger.

```
Schneider 64K Microcomputer  (v1)

(c)1984  Amstrad Consumer Electronics plc
            and Locomotive Software Ltd.

BASIC 1.0

Ready
qwertyuiopasdfghjklzxcvbnm°
```

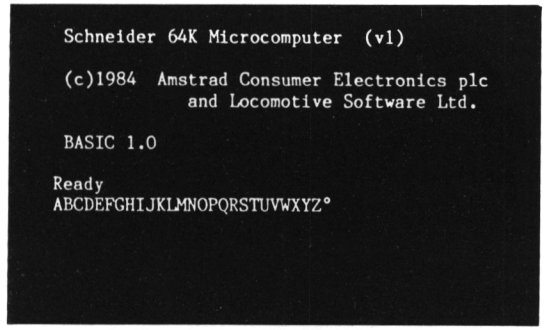
HOOFDLETTERS EN KLEINE LETTERS



U heeft inmiddels gezien dat elke letter die ingevoerd wordt op het beeldscherm, als kleine letter wordt afgedrukt. Vaak, maar vooral bij het verwerken van teksten, moeten er hoofdletters gebruikt worden. Op een schrijfmachine bevindt zich een toets die dit mogelijk maakt. Op de CPC is dat niet anders. De toets 'SHIFT', die U tweemaal op het toetsenbord ziet staan, is voor dit doel te gebruiken. Houd dus deze toets ingedrukt, als U een hoofdletter wenst. Schrijf de naam 'Wim Ibo' eens op het beeldscherm. Het is inmiddels bekent dat de hoofdletters met behulp van de SHIFT toets gemaakt kunnen worden.



Tot nu toe hebben we alle kleine letters op het beeldscherm geschreven. Laten we nu eens alle hoofdletters in alfabetische volgorde intikken. Om niet de SHIFT toets ingedrukt te moeten houden, is er een andere hulptoets (CAPS LOCK). Als U deze toets eenmaal indrukt verschijnen alle letters vanaf dat tijdstip als hoofdletters.



Als de toets CAPS LOCK nog een keer wordt ingedrukt verschijnen de letters weer als kleine letters.

DE SPATIEBALK

Ook deze toets kan men op elke schrijfmachine vinden. Ook bij Uw computer is het mogelijk om spaties tussen de verschillende woorden te zetten. Als er zich op de plaats waar de spatie moet komen te staan, reeds een ander karakter bevindt dan wordt die uiteraard uitgewist.

Maak het beeldscherm weer schoon en tik de volgende zin in:

Oefening maakt de meester

Hierbij moet tussen de enkele woorden op de spatiebalk worden gedrukt.

```
Schneider 64K Microcomputer  (v1)

(c)1984  Amstrad Consumer Electronics plc
           and.Locomotive Software Ltd.

BASIC 1.0

Ready
Oefening maakt de meester°
```

DE DEL TOETS

De afkorting van deze toets zegt al het een en ander over de funktie van deze toets. DEL staat voor DELETE (doorhalen). Met deze toets kan men het teken, dat links van de toets staat, wegnemen. Dit is erg gemakkelijk bij het corrigeren, als er een tikfout gemaakt is. Het indrukken van de toets DEL heeft tot gevolg dat een fout ingetikt karakter weggenomen wordt.

Wis het beeldscherm en tik de volgende zin in:

Tikfouten zijn niet te voorkomeb

Bij het intikken van het laatste woord is een tikfout ontstaan.

```
Schneider 64K Microcomputer  (v1)

(c)1984  Amstrad Consumer Electronics plc
           and Locomotive Software Ltd.

BASIC 1.0

Ready
Tikfouten zijn niet te voorkomeb
```

Tipp-Ex is niet nodig. Door het indrukken van een toets wordt deze fout verwijderd. Druk de toets DEL maar eens in.

```
Schneider 64K Microcomputer  (v1)

(c)1984  Amstrad Consumer Electronics plc
           and Locomotive Software Ltd.

BASIC 1.0

Ready
Tikfouten zijn niet te voorkome
```

Nu kunt U de juiste letter intikken en de tikfout verder vergeten.

```
Schneider 64K Microcomputer  (v1)

(c)1984  Amstrad Consumer Electronics plc
           and Locomotive Software Ltd.

BASIC 1.0

Ready
Tikfouten zijn niet te voorkomen
```

Maar dit voorbeeld laat toch niet de grote waarde van de toets DEL zien. Tik eens, nadat het beeldscherm weer gewist is, de volgende zin in:

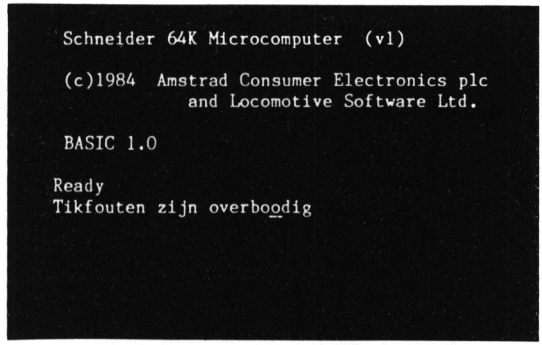
Tikfouten zijn overboodig

Ook hier is er een tikfout ingeslopen: Het is immers niet goed om tweemaal de letter 'O' in te tikken.

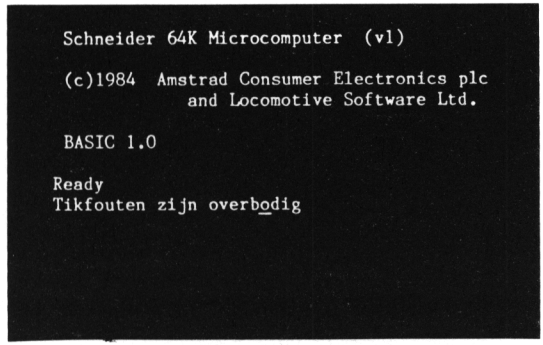
```
Schneider 64K Microcomputer  (v1)

(c)1984  Amstrad Consumer Electronics plc
           and Locomotive Software Ltd.

BASIC 1.0

Ready
Tikfouten zijn overboodig
```

Het is echter niet nodig om vanaf de gemaakte tikfout het hele woord over te schrijven. Wordt namelijk een teken dat links van de cursor staat met de DEL toets verwijderd dan schuiven alle tekens die rechts van de cursor staan automatisch een plaats naar links. Dit is niet gemakkelijk te verklaren, probeer het daarom zelf:

Zet de cursor op de plaats achter het teken dat verwijdert moet worden. Hiervoor heeft men de CURSOR-LINKS toets nodig.

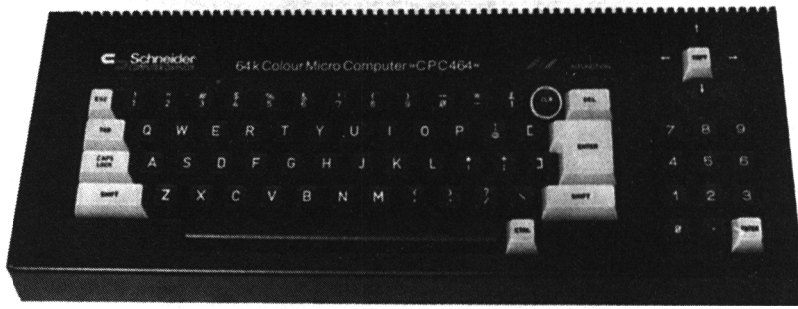


Als U nu op de toets DEL drukt, wordt de letter 'O' die links van de cursor staat weggenomen en wordt de rest, inclusief het teken onder de cursor, naar links opgeschoven.



Als U nu de cursor naar het einde van deze zin brengt, kan men met het schrijven verder gaan. Houdt daarom de CURSOR-RECHTS zolang ingedrukt tot de gewenste plaats bereikt is.

DE CLR-TOETS



De DEL toets heeft, zoals we gezien hebben de funktie, het teken links van de cursor weg te nemen en het teken dat rechts ervan staat, met alle andere tekens op die regel, naar links te schuiven. Dit is niet de enige hulp bij het editeren van de CPC464. Met de toets CLR wordt het teken dat onder de cursor staat, met alle daarop volgende tekens naar rechts geschoven. Hiertoe ook een voorbeeld. Geef nog eens een foute zin:

Het weer is heel mooi

```
Schneider 64K Microcomputer  (v1)

(c)1984  Amstrad Consumer Electronics plc
              and Locomotive Software Ltd.

BASIC 1.0

Ready
Het weer is heel mooi°
```

Nu kan het wel zijn dat het weer mooi is, maar toch niet zo mooi dat het woordje 'heel' op zijn plaats is. Het woordje 'heel' moet verwijderd worden. Natuurlijk kan dat met de toets DEL, maar we willen de funktie van de CLR-toets leren kennen. Omdat CLR de letter onder de cursor wegneemt en de tekens die rechts van de cursor staan naar links schuiven, moeten we de cursor op de plaats van de letter 'h' van 'heel' zetten.

```
Schneider 64K Microcomputer  (v1)

(c)1984  Amstrad Consumer Electronics plc
           and Locomotive Software Ltd.

BASIC 1.0

Ready
Het weer is heel mooi
```

Als U nu de toets CLR vijf maal indrukt, wordt het woord 'heel' gewist.

```
Schneider 64K Microcomputer  (v1)

(c)1984  Amstrad Consumer Electronics plc
           and Locomotive Software Ltd.

BASIC 1.0

Ready
Het weer is mooi
```

Met de toetsen DEL en CLR beschikt U over twee uitstekende editeerhulpen, waarvan U bij het intikken van teksten en programma's veel hulp zult hebben.

INSTELBARE LETTERGROOTTE

U zult zeker gemerkt hebben, dat er op een regel 40 tekens kunnen staan. Met de 25 regels kunnen er dus in het totaal 1000 tekens op het beeldscherm geplaatst worden. Bij veel andere computers van deze klasse wordt dit niet van harte meegedeeld. Daarbij wil men met alle mogelijke truukjes proberen het aantal tekens tot 80 uit te breiden. Dit vereist echter een technische uitbreiding die erg duur is.

Bij het ontwikkelen van de CPC464 kwam men op het idee om het aantal tekens dat op een regel kan staan, variabel te maken. Hierdoor kan de gebruiker kiezen of hij 20, 40 of zelfs 80 tekens op een regel wil hebben, en dat op een heel eenvoudige manier volgens BASIC. We hebben tot nu toe één BASIC opdracht leren kennen en wel de opdracht CLS. Met deze opdracht worden alle tekens van het beeldscherm afgeveegd. Het commando om naar een ander aantal tekens over te schakelen luidt 'MODE'. Omdat er een drietal

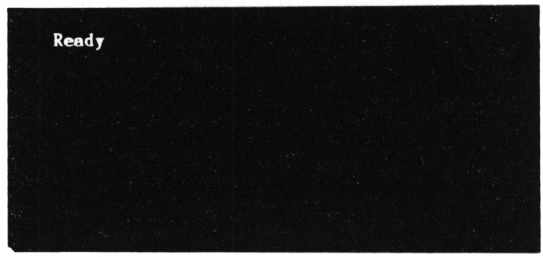
mogelijkheden zijn, moeten we aan het commando MODE nog een cijfer toevoegen. Dit cijfer bepaalt welke van de drie teken-typen we willen gebruiken. En daarom zijn er drie MODE-opdrachten:

MODE 0 – omschakelen naar 20 tekens op een regel.
MODE 1 – omschakelen naar 40 tekens op een regel.
MODE 2 – omschakelen naar 80 tekens op een regel.

We laten U niet langer op de praktische toepassing van deze opdracht wachten. Schakel het beeldscherm in op 'MODE 0'. Let er op dat ook deze opdracht met de ENTER toets moet worden afgesloten, omdat U anders tevergeefs op een reaktie wacht, pas het indrukken van ENTER laat de computer zijn werk doen.

'Dat is wat voor slechtzienden' zult U wellicht opmerken, als het overgedimensioneerde 'READY' op het beeldscherm verschijnt. Het is zeker duidelijker te lezen dan wat er tot nu toe op het beeldscherm verscheen. Men zal dit in het algemeen in spelprogramma's gebruiken. Als U er echter over denkt om bijvoorbeeld deze teksten als opschrift voor een programma te gebruiken en daarna voor de rest van die regel weer de normale tekens, moet ik U helaas teleurstellen. Op het beeldscherm kan men slechts 1 schrijftype toepassen, omdat na elke MODE opdracht het beeldscherm eerst gewist wordt.

Kijken we nu naar het tweede alternatief, dat is de afbeelding met 80 tekens op een regel. De opdracht hiertoe kennen we al uit het overzicht. Geef eens de opdracht MODE 2 en kijk wat er gebeurt.



Als U nu iets op het beeldscherm wilt schrijven, zult U zien dat de kleine tekens verrassend goed te lezen zijn. Als U het er niet mee eens bent, heeft U zeker een kleurenmonitor, want alleen een groene kleur op de monitor laat een goede oplossing van de 80 tekens toe. Dit is een voordeel van een groene monitor. Maar na het bewonderen van de grote kleurenpracht van de CPC464 zult U dit voordeel graag in willen ruilen voor een kleurenmonitor. Dit betekent echter niet dat U over twee monitors moet beschikken, ook al zou dat de optimale oplossing zijn. Een goed compromis is het inzetten van een groene monitor en een HF-modulator, waarmee men de prachtige kleuren van de CPC464 op de normale TV kan bewonderen

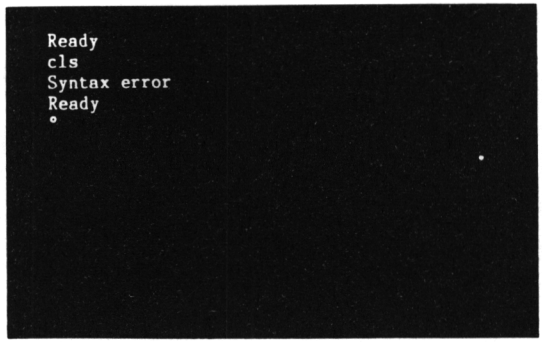
DE KOPIEER-CURSOR

Met elke computer kan men teksten, programma's en data vasthouden en te zijner tijd veranderen. Programma's worden in het algemeen niet slechts een keer geschreven, zij moeten regelmatig veranderd worden. Opdat u met de beste bedoelingen aan de eerste programmeerstappen kunt beginnen, leert U nu reeds hoe U de reeds ingevoerde volgorde kunt veranderen.

Laten we eens een voorbeeld uitvoeren. We nemen aan dat U de opdracht CLS wilt invoeren. Bij het intikken maakt U echter een fout en tikt U in 'CLD'. De laatste letter van deze opdracht is dus fout. Voer dit voorbeeld nu eens uit.

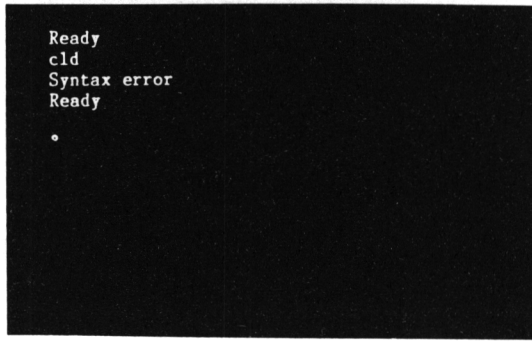
```
Ready
cld
Syntax error
Ready
```

We willen echter deze opdracht niet nog een keer intikken, maar datgene dat nog op het beeldscherm te zien is, veranderen. Andere vergelijkbare computers beschikken nog over een zogenoemde 'beeldscherm editor'. Met een dergelijke editor kan men deze regel met behulp van de cursor opzoeken, veranderen en met ENTER in het geheugen plaatsen. Dit is met de CPC niet mogelijk. Probeer het maar. Ga met de cursor naar de foutieve 'D' en vervang die door de letter 'S'. Links van de cursor staat nu de mededeling 'CLS'. Drukt U nu op de toets ENTER, dan wordt het beeldscherm niet gewist maar ontstaat de mededeling 'SYNTAX ERROR'.

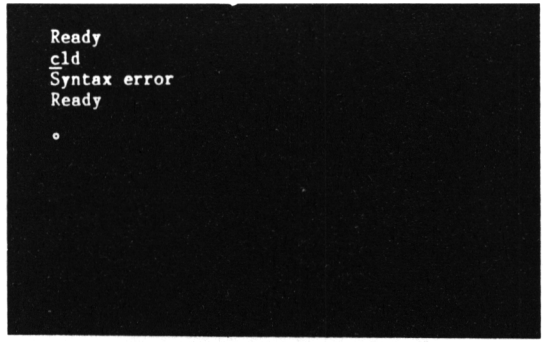


De reden hiervan is dat alle letters van een opdracht eerst moeten worden ingevoerd voor dat men dit met de toets ENTER beëindigt. De tekens die reeds op het beeldscherm staan kunnen niet meer veranderd of voltooid worden. Bij erg lange opdrachtregels, zoals we die later nog wel leren kennen, is dat een nadeel. Men moet dan immers de hele regel opnieuw intikken. De CPC heeft echter de mogelijkheid om U deze moeite uit handen te nemen. Daartoe moet men de 'oude' regel naar een andere plaats op het beeldscherm kopiëren. Het beheersen van deze methode kost weinig inspanning. U wordt stap voor stap met deze techniek vertrouwd gemaakt. Wis daarom eerst het beeldscherm met de juiste opdracht (dus met CLS) uit. Geef daarna opnieuw het verkeerde commando 'CLD'.

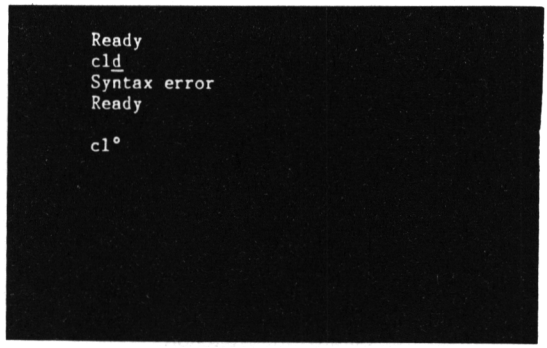
1. Stap – Cursor naar het begin van een 'nieuwe' regel.
Bekijk eerst eens waar de 'oude' regel heen gekopieerd moet worden. Ga dan met de cursor naar het begin van deze regel. In ons geval is dat de 5.de regel op het beeldscherm.



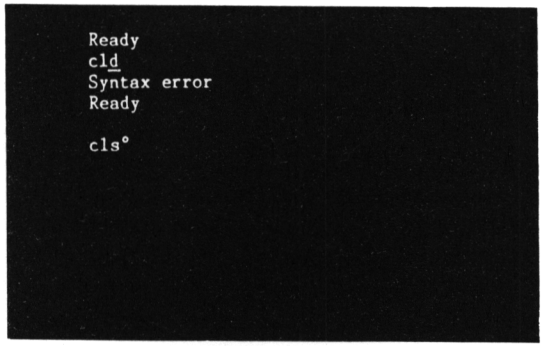
2. Stap – Kopieer-cursor naar begin van de 'oude' regel.
De tot nu toe beschreven cursor kunt U inmiddels goed beheersen. Wat is echter nu de kopieer-cursor en hoe moet men deze besturen? De kopieer-cursor is een tweede cursor die er precies eender uitziet als de eerste cursor. Hij wordt ook met de cursor toetsen gestuurd, maar dan tegelijk met de SHIFT toets. Houdt nu de SHIFT toets ingedrukt en stuur met de CURSOR BOVEN toets de kopieer-cursor tot de 'oude' regel bereikt is.



3. Stap – Met COPY-toets gewenst teken kopiëren.
De toets COPY beweegt de kopieer cursor op dezelfde manier als de normale cursor en zal daarbij de tekens onder de cursor kopiëren. Dit lijkt gecompliceerder dan het in werkelijkheid is. Probeert U dat zelf eens en druk twee keer op de COPY-toets.



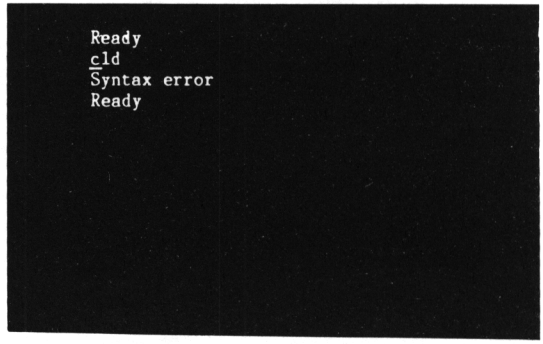
U hebt nu de eerste twee tekens van de opdracht CLS gekopieerd. De derde letter mag niet gekopieerd worden (die was immers fout), maar die moet opnieuw ingetikt worden. Druk dus daarom op de toets 'S'.



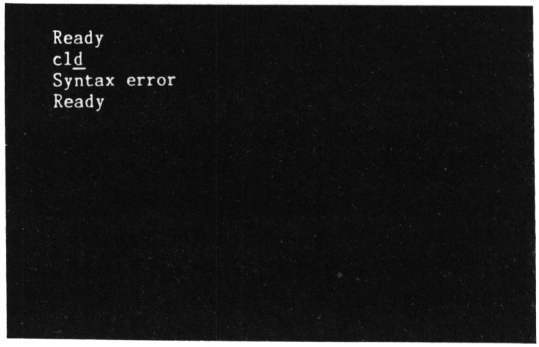
De oude opdracht is nu gekopieerd en veranderd. Druk nu op de toets ENTER en de opdracht zal worden uitgevoerd. Met ENTER verdwijnt ook de kopieer-cursor.

Naast deze methode om bestaande opdrachten te veranderen, is er nog een tweede. We hebben de 'oude' regel op een andere plaats van het beeldscherm gekopieerd en veranderd. De andere mogelijkheid is: de 'oude' regel op dezelfde plaats te veranderen. De poging om met de cursor deze plaats te veranderen liep uit op een mislukking. Laten we ook deze tweede mogelijkheid eens stap voor stap uitvoeren. Daartoe brengen we ook weer eerst een foutieve opdracht aan op het beeldscherm. We nemen hiervoor de reeds bekende opdracht 'CLD'.

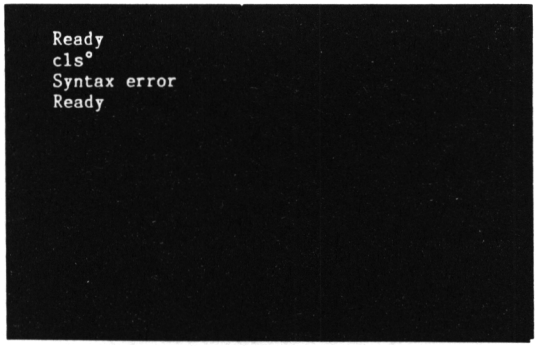
1. Stap – Cursor naar het begin van de 'oude' regel.
We zetten de cursor dus niet op een nieuwe plaats, maar naar het begin van de regel die we willen veranderen.



2. Stap – COPY-toets tot aan het eerste foutieve teken.
Druk nu de COPY-toets twee keer in. Daarbij worden de eerste twee tekens op deze plaats gekopieerd. Omdat de kopieer-cursor dezelfde is als de normale cursor moet U er op letten dat U de COPY-cursor gebruikt en niet de cursor RECHTS.



Het derde teken (de foutieve D) moet nu niet gekopieerd worden maar vervangen door de letter 'S'.



Omdat deze opdracht nu foutloos is kan hij met ENTER worden ingevoerd.

We hebben nu twee mogelijkheden leren kennen om een bestaand beeldschermteken te veranderen, waarbij de tweede methode er iets gemakkelijker uitziet als de eerste. Welk van deze twee U wilt gebruiken, zal van geval tot geval moeten worden bekeken.

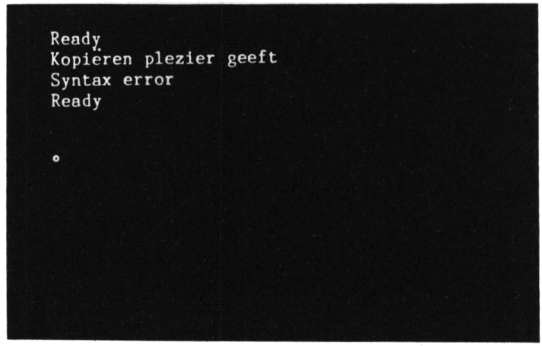
Misschien heeft U toch nog wat problemen met het werken met de COPY-cursor. We zullen daarom een wat omvangrijker voorbeeld behandelen. Nadat het beeldscherm gewist is, tikken we de volgende woorden:

Kopiëren plezier geeft

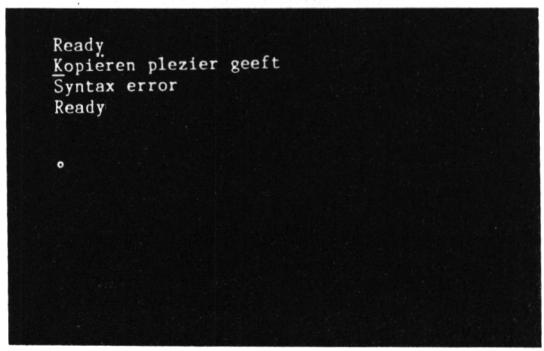
Daarna drukken we op de toets ENTER.

Van deze zin zullen we met behulp van de COPY-toets de zin 'Kopiëren geeft plezier' maken. Omdat deze woorden geen opdracht vormen moeten we de mededeling 'SYNTAX ERROR' op de koop toe moeten nemen. Maar dat zal ons verder niet irriteren bij het bereiken van ons doel.

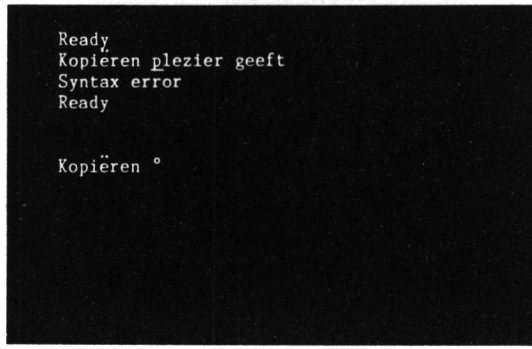
De goede zin moet in de 6.de regel van het beeldscherm verschijnen. Daarom zetten we de cursor naar het begin van deze regel.



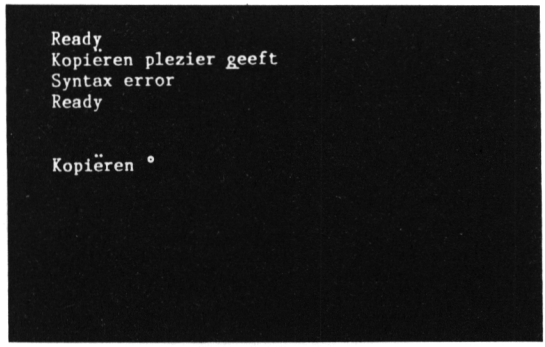
Nu moeten we de COPY-cursor naar het eerste woord brengen dat gekopieerd moet worden. Daartoe drukken we op de SHIFT toets en bewegen de cursor naar de gewenste plaats.



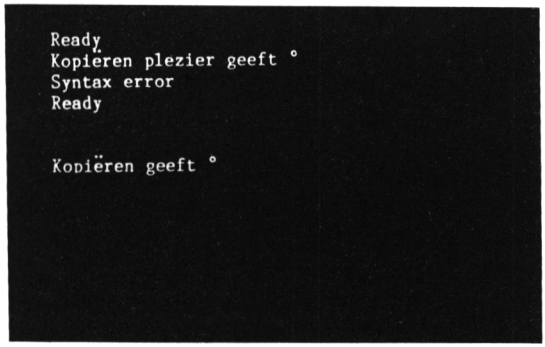
Nu kunnen we het eerste woord kopiëren. Druk daarom op de COPY-toets tot de letter 'p' van 'plezier' bereikt is. De spatie wordt dan ook gekopieerd.



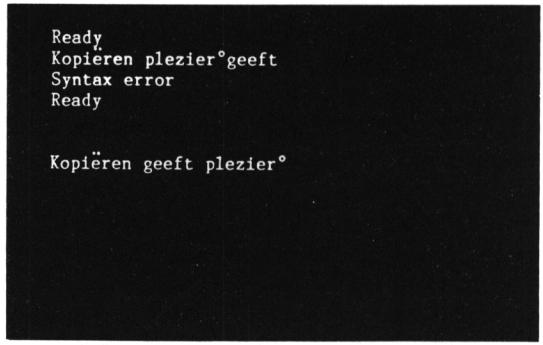
Nu moeten we het tweede woord kopiëren. Daarvoor moeten we de kopieer-cursor naar het begin van het woord 'geeft' brengen. Druk dus de SHIFT-toets in en druk zes keer op de CURSOR RECHTS.



Aansluitend wordt dit woord gekopieerd, door zes keer op de COPY-toets te drukken. Deze zesde keer is nodig om ook de spatie te kopiëren.



Probeer nu eens zelf het laatste woord 'plezier'. Dus de kopieer-cursor verplaatsen en de COPY-toets gebruiken! Als alles naar wens verlopen is, ziet Uw beeldscherm er uit als de volgende foto:



En is het gelukt? Wel, gefeliciteerd dan! Als U nu nog enige oefeningen maakt, zult U ook deze techniek spoedig beheersen. U moet er goed op letten dat de twee cursor toetsen niet worden verwisseld. Anders komt U in een situatie die U alleen met wat meer routine de baas kunt worden.

De eerste opdracht

Voor de verdere verklaringen van het toetsenbord is het niet meer voldoende om een willekeurig teken op het beeldscherm te plaatsen. Een enkele lezer zal zijn CPC464 voor dit doel gebruiken. Voor al diegenen die het vorige hoofdstuk vervelend vonden, wordt dit hoofdstuk aanmerkelijk interessanter. We willen onze computer nu gaan gebruiken waar hij in wezen voor ontworpen is: het uitvoeren van opdrachten.

DE ENTER TOETS

ENTER is verreweg de belangrijkste toets op Uw computer. Als U deze toets indrukt, wordt de ingetikte tekst in het geheugen van Uw computer opgeslagen. De computer interpreteert deze tekst als 'opdracht' en zal de noodzakelijke stappen inleiden. Na het afsluiten van deze opdracht zal de computer zich weer met de bekende mededeling READY (zoals ook na het aanzetten) melden. Hij is dus weer in staat om een andere opdracht in ontvangst te nemen. Probeert U eens de computer te begroeten met 'Goeden morgen' en druk daarna de toets ENTER in. Wat denkt U, zal de computer U antwoord geven? Maar kijkt U zelf.

```
Ready
Goeden morgen
Syntax error
Ready
```

Hier verschijnt de eerste foutmelding op Uw CPC464, de SYNTAX ERROR (syntax fout). De syntax, dat is het samenspel van de tekens, wordt door Uw computer niet begrepen.

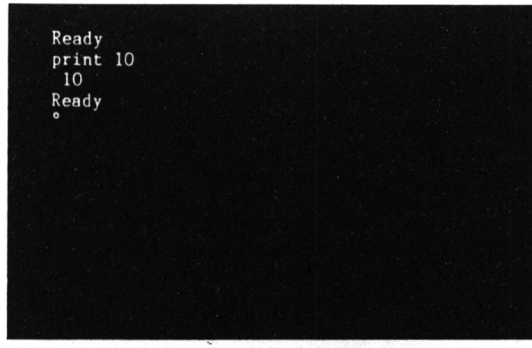
Er bestaan tegenwoordig weliswaar grote computers waarmee men zich over een uitgebreid gebied kan 'onderhouden'. Maar men moet hiervoor een veel hogere prijs betalen en men heeft een veel grotere geheugenruimte nodig. Datgene wat U met het toetsenbord invoert en met ENTER afsluit, wordt door de BASIC-interpreter geanalyseerd en, als dit zonder fouten is, uitgevoerd.

Uw CPC464 is van de programmeertaal BASIC voorzien en begrijpt dus alleen opdrachten uit deze taal. BASIC is de meest voorkomende computertaal, die juist bij de beginners optimale mogelijkheden biedt voor het programmeren. De opdrachten zijn ontleent aan de Engelse taal. Hoe snel kan ik deze taal leren? Dit is een veel voorkomende vraag, die men echter niet volledig kan beantwoorden. Veel criteria spelen hierbij een rol. Bij de een is het vermogen om te leren doorslaggevend, bij de ander is de tijd die men aan de computer besteedt, van betekenis. Er zijn mensen die elke vrije minuut aan hun computer wijden. Advertenties in dagbladen zoals 'Ik verkoop mijn computer omdat er een echtscheiding dreigt' zijn geen zeldzaamheid meer. In beginsel geldt; men moet een redelijke tijd aan zijn computer besteden, om de hobby van het programmeren in BASIC tot een ware vreugde te laten uitgroeien. Met een zekere voorkennis van de wiskunde, waartoe ook de beginselen van de algebra behoren, kan men met een studie van 10 uur per week al na drie maanden een redelijk resultaat verwachten. Weliswaar draagt het logische en abstrakte denkvermogen tot een resultaat bij, maar dit wordt van programma tot programma verder ontwikkeld. Voor het ontwikkelen van nuttige programma's, zoals bij voorbeeld een kleine tekstverwerker of een programma over het handelen met data, zijn behalve de kennis, ook ervaring met het programmeren noodzakelijk. Van programma tot programma zal de kwaliteit van de programma's toenemen. Een klein lichtpuntje voor U is: er zijn er velen die drie maanden nadat zij zich een computer hebben aangeschaft, pas toekomen aan het zelf ontwikkelen van een programma. Met een beetje doorzettingsvermogen, zult U ook spoedig een van de vele hobby-programmeurs zijn.

DE PRINTOPDRACHT

Zonder deze opdracht is er nauwelijks een programma denkbaar. Deze opdracht zorgt ervoor, dat er van een programma een uitvoer op het beeldscherm plaatsvindt. Of U nu een uitkomst van een berekening of een adres van de cassette op het beeldscherm wilt hebben, zonder deze opdracht gaat het niet. Ook het afdrukken gaat met behulp van deze opdracht.

Op dit moment gebruiken we alleen nog de 'direct-modus', we schrijven dus nog geen programma's. Direct-modus betekent dat een opdracht direkt uitgevoerd wordt. Dit willen we eerst zonder foutmeldingen laten zien. Geef daarom eerst de opdracht 'PRINT 10' gevolgd door ENTER.



Met PRINT kan dus alles gedaan worden wat de parameter aanwijst. Dit verschijnt dan op het beeldscherm, de uitvoer naar een ander apparaat gebeurt op een andere manier dan met PRINT. Een parameter is een bestanddeel van de PRINT opdracht, die nauwkeurig omschrijft, welke uitwerking de opdracht zal heben.

Hier was de parameter het getal '10'. 'PRINT 10' betekent dus 'laat het getal 10 op het beeldscherm zien'. Natuurlijk kan men ook andere tekens dan alleen maar getallen laten zien. Hoe veelzijdig deze opdracht is zal men in het verdere verloop van dit hoofdstuk wel zien.

REKENEN MET PRINT

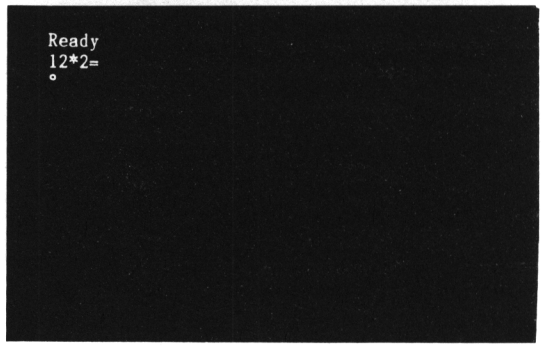
Als U wilt uitrekenen hoeveel loonbelasting U terug zult ontvangen en op dat moment toevallig geen rekenmachine bij de hand hebt, schakel dan de CPC464 in! Zelf schakel ik ook vaak de CPC464 in als er een berekening gemaakt moet worden. Veel mensen vragen dan ironisch 'wat is dat, kan men met dat ding ook nog rekenen?'. Dat zou nou ook wat zijn, als men met een computer, die ongeveer twintig keer zo duur is als een gewone rekenmachine, niet eens zou kunnen rekenen.

Maar nu ter zake. Laten we eerst eens de vier grondberekeningen bekijken. Hiervoor zijn er op het toetsenbord vier symbolen aangebracht:

+	voor het optellen
-	voor het aftrekken
*	voor het vermenigvuldigen
/	voor het delen

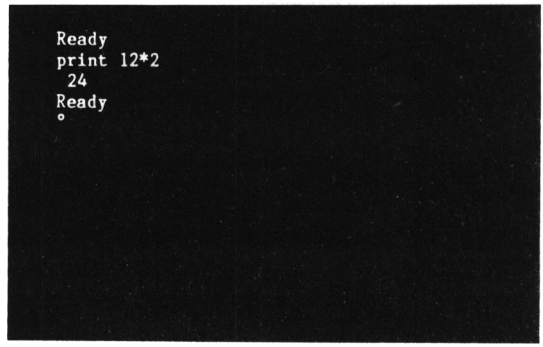
Nu is het rekenen met een computer niet te vergelijken met het rekenen met een zakrekenmachine. Men kan hier bijvoorbeeld niet invoeren '12*2=', omdat er, zoals inmiddels bekend is, zonder opdracht niets gebeurt. Er bestaan echter altijd lezers die dit door experi-

menten zelf willen onderzoeken. Wat er dan gebeurt is nogal verwarrend. Probeert U het maar eens. Voer de opdracht '12*2', gevolgd door ENTER, eens in.



Wat er nu gebeurt is zonder een toelichting niet te begrijpen. De computer laat niet de uitkomst zien, maar hij meldt zich weer met READY. De cursor springt naar het begin van een nieuwe regel. En wat gebeurt er intussen? De computer moet toch iets gedaan hebben, anders zou hij een foutmelding hebben gegeven. Ik zal hier niet al te diep op ingaan. Maar ik wil toch iets toelichten. Een programma bestaat uit meerdere regels die, in het algemeen, de een na de ander worden afgewerkt. Elk van deze regels heeft aan het begin een getal staan, dat de volgorde van de regels aangeeft. In dit geval krijgt de regel 12 als inhoud '*2'. Dit is weliswaar geen correcte opdracht, maar dat wordt bij het inlezen van deze regel niet opgemerkt. Dat gebeurt pas bij de uitvoering hiervan. Maar laten we onze opdracht eerst eens verbeteren.

Hoe moet men '12*2' dan wel correct invoeren en uit laten rekenen? De oplossing is eenvoudig: Geef de opdracht PRINT 12*2. Let er wel op dat er na een opdracht steeds de toets ENTER wordt ingedrukt. Wat U ook op het beeldscherm schrijft, dat zal de computer een zorg zijn. Hem interesseert alleen wat hij met de toets ENTER uit moet voeren.



De uitkomst ontlokt U misschien de uitroep 'AHA'. Uw computer heeft voor het eerst iets voor U gedaan. Hij volgde nauwkeurig Uw aanwijzing op om 12 met 2 te vermenigvuldigen, en de uitkomst hiervan op het beeldscherm mee te delen.

Intussen wil ik even opmerken, dat programmeertalen zo ingewikkeld lijken, omdat de opdrachten die aan de computer moeten worden verstrekt, nogal moeten worden ingekort. Elke programmeertaal moet tot in de puntjes nauwkeurig georganiseerd zijn. Een programmeertaal die een aanwijzing als 'bereken 12*2 en laat de uitkomst zien' bestaat niet, en zal ook nooit bestaan. Dit soort verwachtingen moet U maar gauw opzij zetten. Zo eenvoudig is het programmeren nu ook weer niet. Als U echter deze taal beheerst, kunt U zich rustig op de schouder kloppen.

In tegenstelling met andere computers moet de spatie achter de opdracht PRINT hier absoluut worden ingevoerd omdat de computer anders de mededeling SYNTAX ERROR geeft. Dat geldt niet alleen bij de opdracht PRINT, maar bij alle andere opdrachten geldt dit ook. Dit zult U in het verdere verloop van dit boek nog wel leren. Maar nu weer verder met onze basis berekeningen. Bereken eens achter elkaar '12+2', '12-2', '12*2' en '12/2'. Uw beeldscherm zal er dan als volgt uit gaan zien:

```
Ready
print 12+2
 14
Ready
print 12-2
 10
Ready
print 12*2
 24
Ready
print 12/2
 6
Ready
```

En, is het U gelukt? Goed, dan kunnen we nu weer een stapje hoger op onze ladder naar het programmeren gaan. Is het niet gelukt, lees dan het laatste onderdeel nog eens zorgvuldig door.

Naast zulke eenvoudige berekeningen kunt U ook lange berekeningen, tot 255 tekens toe, uit laten voeren. Bekijk daarvoor eerst de hierarchie van zulke berekeningen. Laat Uw computer eens de volgende berekening uitvoeren: 1+2+3+4+5+6+7+8+9.

```
Ready
print 1+2+3+4+5+6+7+8+9
 45
Ready
```

DE BEREKENINGEN MET HAAKJES

Er bestaat zelfs de mogelijkheid om een kettingberekening uit te laten voeren. Een praktisch voorbeeld: Er moet een totaalprijs van drie stukken tapijt uitgerekend worden. De prijs per vierkante meter is: *f* 23.90. Het eerste stuk is 2.45m * 2.80m, het tweede stuk 4.50m * 3.85m en het derde stuk 2.75m * 4.80m. Daar moet nog 19% BTW bijkomen. Hoe groot is nu de totaal prijs van deze drie stukken tapijt? Als U de haakjes op de juiste plaats zet, is er slechts een PRINT opdracht nodig:

PRINT (2.45*2.80+4.50*3.85+2.75*4.80)*23.90*1.19

Dit ziet er gecompliceerder uit dan het in werkelijkheid is. Tussen de haakjes wordt het aantal vierkante meters uitgerekend. Dit aantal wordt dan met de prijs per vierkante meter vermenigvuldigd en tenslotte wordt er nog de BTW bij gedaan. En dat alles met slechts een opdracht. Als alles correct is gedaan krijgt men als antwoord: *f* 1058.8179. Dus de prijs is *f* 1058,82.

Zoals U gezien hebt, wordt niet de komma maar de punt als decimaalteken gebruikt. Dit is weliswaar een Amerikaanse norm, maar die wordt ook in ons land gebruikt. Als U de komma op de plaats van de punt gebruikt, bijvoorbeeld in '(9*23,8)*2' zal er een foutmelding op het beeldscherm komen te staan.

```
Ready
print (9+23,8)*2
Syntax error
Ready
o
```

DE WETENSCHAPPELIJKE NOTATIE

In de vorige berekening kwam het antwoord met vijf plaatsen achter de komma op het beeldscherm te staan. Dit wordt de 'floating point' notatie genoemd. In beginsel wordt een getal in negen plaatsen nauwkeurig uitgerekend. Als een getal ontstaat met drie plaatsen voor de komma, dan wordt automatisch met zes cijfers achter de komma gerekend. Als een getal echter groter wordt dan 999999999, wordt er ook met negen cijfers gerekend maar dan komt er nog iets achter. Als U bijvoorbeeld de berekening '999999999+1' uit laat voeren, dan ontstaat er een getal van tien cijfers. Dit getal kan de computer niet meer als een 'normaal' getal opschrijven.

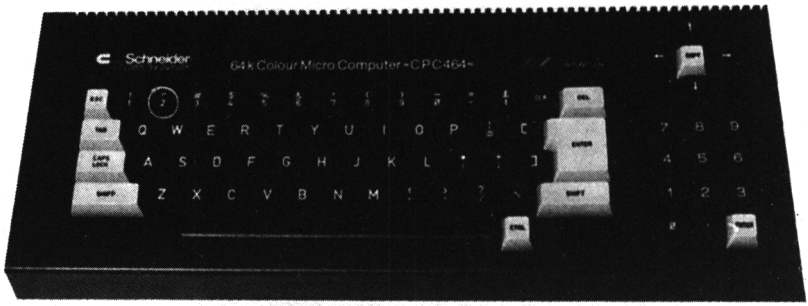
```
Ready
print 999999999+1
 1E+09
Ready
o
```

Twijfel echter niet aan deze uitkomst. Het betekent eenvoudig, dat de uitkomst '1 * 10 tot de macht 9' is, dus een 1 met negen nullen. 'E+09' betekent eenvoudig 'grondtal 10, exponent +9'. De exponent kan ook nog negatief zijn. De berekening 'PRINT 9/30000000000' laat dit zien.

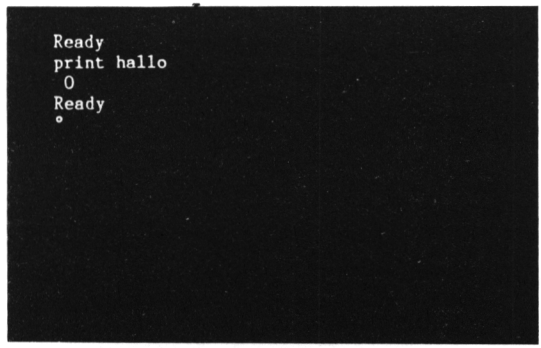
```
Ready
print 9/30000000000
 3E-10
Ready
```

De uitkomst moet eigenlijk zijn '0.0000000003'. De computer kiest hier echter voor de schrijfwijze met exponenten, ook wel genoemd de 'wetenschappelijke notatie'. '3E-10' betekent hetzelfde als '3 * (10 tot de macht -10)'. Het getal '3' bevindt zich dus op de tiende plaats achter de komma. Dit is voorlopig alles wat U over het rekenen in de direct modus moet weten. In Uw handboek van de CPC464 vindt U nog veel meer wiskundige funkties.

UITVOEREN VAN TEKSTEN MET PRINT



Behalve getallen kan men ook teksten, zogeheten 'strings' (teken-ketens), met de opdracht PRINT op het beeldscherm afdrukken. Vooruitlopend op een uiteenzetting, zal de poging 'PRINT HALLO' echter op een mislukking uitdraaien.



Terwijl de opdracht niet het gewenste resultaat heeft, verschijnt er toch geen foutmelding. De computer heeft de opdracht dus toch uitgevoerd. Maar wat heeft hij dan wel uitgevoerd? Waar vandaan het getal nul? Dat zijn de problemen die men bij de eerste pogingen om met BASIC te werken steeds opnieuw tegenkomt. Deze vragen zijn echter moeilijk te beantwoorden. Kort gezegd wordt de inhoud van een variabele (rekenkundig getalgeheugen) aangewezen. De betekenis van dit geheugen is 'HALLO'. Omdat de variabele 'HALLO' nog geen waarde heeft gekregen, ontstaat het getal 0. Als U dit nog niet helemaal begrepen heeft, is dat geen reden om hier ongerust over te worden. De variabelen worden later nog uitvoerig behandeld.

Maar hoe kan men dan een teken-keten (vanaf nu gebruiken wij de vakterm 'string'), hoe kan men een string nu met PRINT afdrukken? De oplossing is eenvoudig: strings worden tussen aanhalingstekens gezet. Zet dit nu om in een daad en verander de laatste opdracht. Zet nu de string 'HALLO' tussen aanhalingstekens:

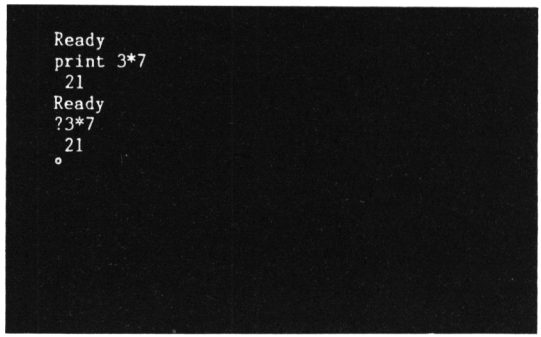
PRINT "HALLO"

En zie, nadat het eerst fout is gegaan, is het nu toch gelukt.
Overigens: als U tot nu toe erg voorzichtig met Uw computer bent omgegaan en U bij elke opdracht drie keer hebt bezonnen of het wel goed gaat, om vooral niets kapot te maken, dan kan ik U gerust stellen. Geen enkele opdracht, al is die dan ook nog zo fout, kan uit Uw computer rookwolken te voorschijn toveren. Bij geen enkele hobby is de wijsheid 'uit fouten leert men' zo welsprekend als bij deze hobby. De knutselaar radio-amateur, die voor *f* 45.- zijn eindversterker verkeerd in zijn zelf gebouwde versterker soldeert, kan zich wel voor zijn hoofd slaan. Maar als U een fout maakt in een 23K byte programma, dan kan het met vijf minuten alweer hersteld zijn. De enige en daarmee ook de grootste fout die U kunt veroorzaken, is als U bv. acht uur achter elkaar aan een programma hebt gewerkt en niets

daarvan in het extern geheugen hebt opgeslagen en Uw moeder schakelt de defecte stofzuiger aan. Een kortsluiting kan hier een groot gevolg hebben, namelijk acht uur werk naar de knoppen. Ook kunt U bv. een belangrijke cassette door onzorgvuldig handelen vernietigen. Maar als U de belangrijkste richtlijnen van het beschermen van Uw gegevens in acht neemt, kan er niets gebeuren.

VEREENVOUDIGDE PRINT-INVOER

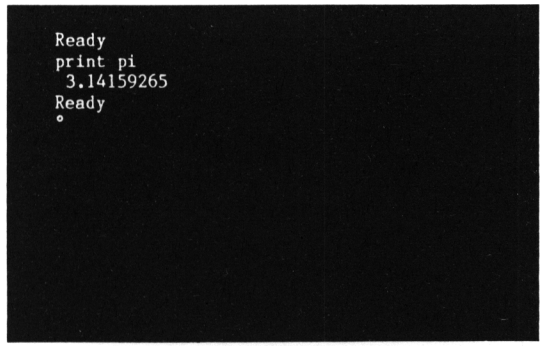
Inplaats van de opdracht PRINT kan men ook een vraagteken gebruiken. Voer inplaats van de opdracht PRINT eens het vraagteken in, daarmee bespaart U zich ook nog wat arbeid, temeer daar de opdracht PRINT de meest voorkomende opdracht is. Het maakt dus niets uit of men 'PRINT 3*7' dan wel '? 3*7' intikt. Probeer dat zelf maar eens. Achter het vraagteken is overigens geen spatie nodig.



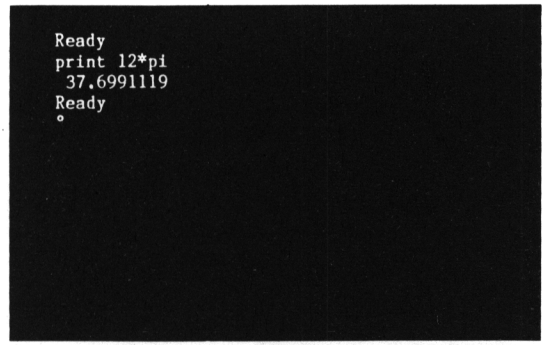
Het ligt overigens aan U, of U het woord PRINT geheel uit wilt schrijven, dan wel het veel kortere vraagteken wilt gebruiken. Bij berekeningen die men 'even snel' in wilt tikken, is het gebruik van het vraagteken erg praktisch. Men kan daarmee even snel rekenen als met een zakrekenmachine.

PI EN MACHTSVERHEFFEN

We gaan nu weer terug naar de wiskunde. Het voorkomen van de constante PI moet aan iedereen bekend zijn. Bij cirkel- en bolberekeningen is zij altijd nodig. Welke waarde heeft nu PI? Was het 3.1412 of 3.1214 of... Breekt U zich daar het hoofd maar niet over. Meestal wordt er met 3.14 gerekend, dus met twee plaatsen achter de komma. De CPC464 heeft echter deze constante ingebouwd. Roept U de waarde van PI maar eens op door 'PRINT PI'



PI met acht plaatsen achter de komma! Dat is meer dan voldoende. Bereken eens de omtrek van de cirkel met de diameter 12. De formule luidt 'Diameter * PI'.



De omtrek van deze cirkel is dus ongeveer 37.7 cm.

Nu naar het machtsverheffen. Machtsverheffen betekent een getal met zichzelf vermenigvuldigen. De exponent bepaalt hoe vaak het grondtal met zichzelf moet worden vermenigvuldigd.

2 tot de macht 3 betekent dus 2*2*2, dat is dus 8
10 tot de macht 4 betekent dus 10*10*10*10 en dat is 10000

In het eerste voorbeeld is '2' het grondtal, en '3' de exponent. Maar nu verder geen wiskunde-onderwijs. Bij de CPC464 wordt tussen het grondtal en de exponent een pijl naar boven gezet. Rekent U zelf eens 2 tot de macht 8 uit.

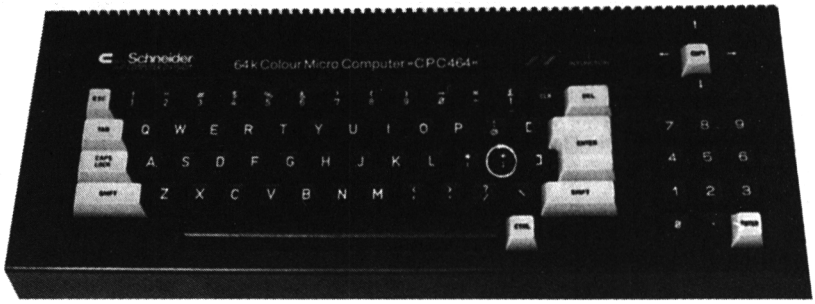
```
Ready
print 2^8
 256
Ready
°
```

Dat was niet moeilijk. Bereken nu eens de oppervlakte van een cirkel waarvan de straal 12 cm is. De formule hiervan is: Oppervlakte = Straal tot de macht 2 maal PI (De straal is de helft van de diameter).

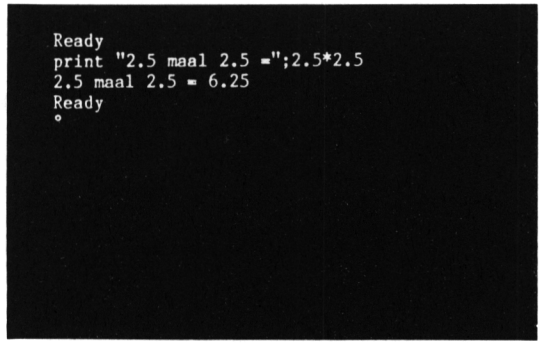
```
Ready
print 6^2*pi
 113.097336
Ready
°
```

De oppervlakte van deze cirkel is dus ongeveer 113 vierkante centimeter. Maar nu genoeg over deze droge wiskunde. Laten we eens wat interessantere zaken bekijken.

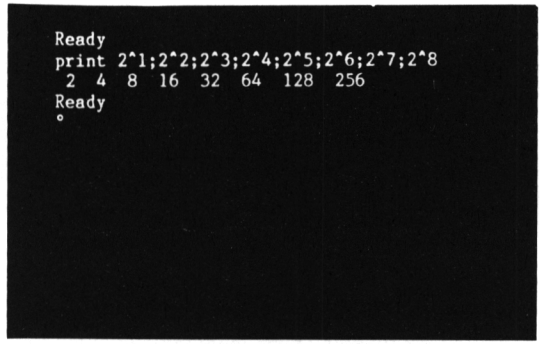
COMBINEREN VAN STRINGS MET GETALLEN



De toetsen ';' en ',' spelen een grote rol bij het combineren van strings met getallen. Maar welke? Als U bij voorbeeld een regel als '2.5 maal 2.5 = 6.25' op het beeldscherm wilt laten zien, en dan gelijktijdig de berekening uit wilt voeren, dan is er een string met een bijbehorende berekening nodig. De oplossing hiervan ziet er dan als volgt uit:



De punt-komma brengt dus een scheiding tot stand tussen de string en het getal. En niet alleen dat, ook berekeningen kunnen door een punt-komma van elkaar gescheiden worden. Zo kan men bijvoorbeeld meerdere berekeningen met 1 PRINT opdracht uitvoeren en de uitkomsten hiervan naast elkaar af laten drukken. Probeert U eens de machten van '2' met de exponenten van '1' tot en met '8' te berekenen en naast elkaar af te drukken.



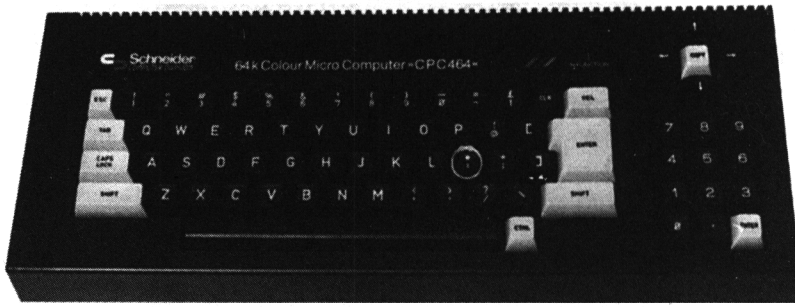
Een PRINT opdracht die met een punt-komma wordt afgesloten heeft tot gevolg dat de hierna volgende PRINT opdracht niet naar de volgende regel wordt doorgeschoven, maar op dezelfde regel wordt uitgevoerd. Maar hierover vertellen we later meer.

Geef nu nog eens de laatste opdracht, maar vervang hierbij de punt-komma door een komma.

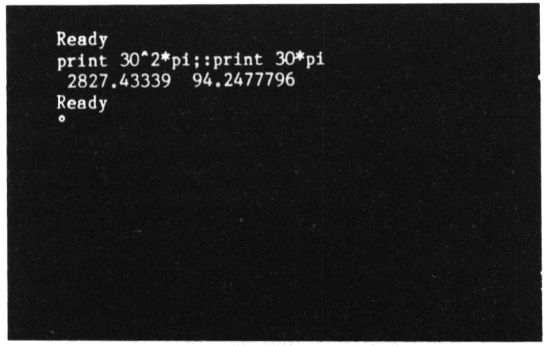
```
Ready
print 2^1,2^2,2^3,2^4,2^5,2^6,2^7,2^8
 2             4             8
 16            32            64
 128           256
Ready
```

Ook door de komma worden de waarden gescheiden, maar de afstand wordt dan wel groter. Als U de afstanden uittelt, dan zult U zien dat deze afstand 10 tekens groot is. Hierdoor is het op een eenvoudige wijze in te richten dat er getal kolommen ontstaan. Voor het scheiden van strings wordt echter zelden een komma gebruikt.

SCHEIDEN VAN OPDRACHTEN

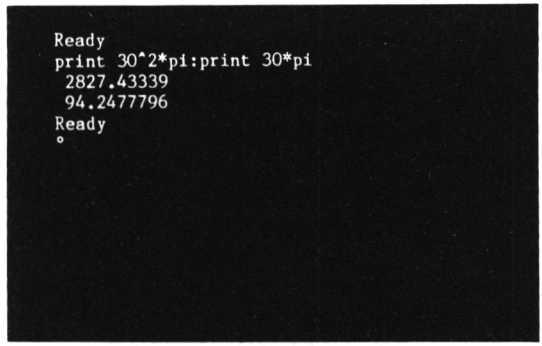


Een opdracht regel kan uit maximaal 255 tekens bestaan. Er zijn nauwelijks BASIC opdrachten te bedenken die deze ruimte nodig hebben. Met een dubbele punt kan men nu opdrachten van elkaar scheiden en zodoende meer opdrachten op een regel schrijven. Een voorbeeld zal dit duidelijk maken. U wilt twee berekeningen tegelijk uit voeren. De uitkomsten hiervan moeten onder elkaar verschijnen. In ons voorbeeld berekenen we een keer 30+2*PI en een keer 30*PI. Probeer nu deze twee PRINT opdrachten in een regel onder te brengen, gescheiden door een dubbele punt.



Op deze manier kan men ook vier of vijf opdrachten geven en deze door een dubbele punt scheiden. Men moet er alleen op letten dat het aantal van 255 tekens op een regel niet wordt overschreden.

Bij dit voorbeeld is het de bedoeling de werking van de punt-komma na de eerste PRINT opdracht te verduidelijken. Laten we het beeldscherm eerst weer eens wissen. Geef nu dezelfde PRINT opdracht nog eens, maar nu gescheiden door een punt-komma.



De punt-komma heeft hierbij als resultaat dat de regel niet wordt beëindigd. De tweede PRINT volgt onmiddellijk op de eerste. De lege plaats tussen deze getallen is nodig omdat er voor het eerste cijfer een plaats moet worden vrijgehouden voor een eventueel voorteken. Een positief getal krijgt weliswaar geen + teken, maar een negatief getal krijgt wel een - teken.

Opgemerkt wordt hier nog, dat een spatie na de punt-komma geen enkele betekenis heeft. Of er hier 1 spatie of 5 spaties of geen enkele spatie gebruikt wordt, dat heeft geen uitwerking op de opdracht.

Het eerste programma

Hiermee wordt niet het een of andere televisieprogramma bedoeld, maar Uw eerste stap om zelf een programma in BASIC te schrijven. Stap hierbij over Uw eventuele angsten heen, en pak dit 'hete ijzer' beet. BASIC is een programmeertaal die bedoeld is voor de beginners. Het feit dat men in een warenhuis of in een computerzaak vaak kinderen ziet die zich na de schooltijd met BASIC bezighouden, zal de laatste remmen bij U wegnemen. U wilt toch geen gebruiker zijn die alle tijd doorbrengt met programma's die kant en klaar in de winkel in de winkel liggen? Wees eens eerlijk, heeft U er niet vaak van gedroomd Uw eigen programma's te ontwikkelen? Dit, en het volgende hoofdstuk zal een kleine stap betekenen in de grote en interessante wereld van het programmeren.

EEN PROGRAMMA, WAT IS DAT?

In de vaktaal wordt een programma gedefinieerd als een 'volgorde van opdrachten waarmee een zekere vraagstelling opgelost kan worden'. Het is dus deze volgorde, die in een logisch verband een programma vormt. De weg naar een programma wordt dus bepaald door de vraag en door de logische volgorde, om hiermee het gestelde doel te bereiken. Hiermee bedoelen we dat we niet alleen maar alle BASIC opdrachten moeten kennen, maar ook hun samenspel, om tot een oplossing van het probleem te komen. Zo is het bijvoorbeeld zinloos om een groot aantal Engelstalige woorden te leren als men hiermee niet, door een zinvol samenspel, tot een communicatie kan komen.

HET NUMMEREN VAN REGELS

Een programma bestaat dus uit een volgorde van opdrachten. Maar waardoor wordt deze volgorde bepaald? Wel, bij de programmeertaal BASIC wordt elke opdracht (ook wel 'statement' genoemd) van een regelnummer voorzien. En deze regelnummers bepalen de volgorde waarmee het programma af moet lopen. Stelt U zich eens voor dat U met een zakrekenmachine de opgave '48/12' uit wilt rekenen. Dan is het duidelijk dat ook hiervoor een zekere volgorde van handelingen noodzakelijk is:

1. Zoek de zakrekenmachine
2. Zet het apparaat aan
3. Als de batterij leeg is ga dan naar 12
4. Druk op de toets '4'
5. Druk op de toets '8'
6. Druk op het teken dat de deling aan geeft
7. Druk op de toets '1'
8. Druk op de toets '2'

9. Druk op de toets '='
10. Lees de uitkomst af
11. Zet de uitkomst in het geheugen.
12. Schakel Uw rekenmachine uit

Verbazingwekkend, dat er voor zo een eenvoudige berekening zoveel stappen gedaan moeten worden. U ziet hierbij, dat alle regels genummerd zijn om de volgorde vast te leggen. Maar dit is niet de enige reden. Stap nummer 3 laat een belangrijke programmeerlogika zien, een sprong-opdracht. Als er aan een bepaalde voorwaarde voldaan is (batterij leeg), vertakt het programma zich naar stap 12. Dit getal is het zogenoemde adres van deze opdracht. Zonder regelnummer zou het niet mogelijk zijn om naar deze regel te springen.

We hebben in het vorige hoofdstuk inmiddels al een opdracht geleerd, de PRINT opdracht, waarmee men gegevens op het beeldscherm kan laten zien. Deze opdracht hebben we echter tot nu toe alleen in de direct mode gebruikt, dat wil zeggen, dat na het indrukken van de toets ENTER de opdracht direkt werd uitgevoerd. We willen nu een volgorde van drie PRINT-opdrachten geven. Wat moet men dan doen? Juist, elke opdracht krijgt een regelnummer, waarmee men de volgorde van de uit te voeren opdrachten aangeeft.

ALS REGELNUMMER KAN MEN BIJ DE CPC464 ALLE WAARDEN VAN 0 TOT 65535 KIEZEN. DE STAPGROOTTE HEEFT VERDER GEEN BETEKENIS.

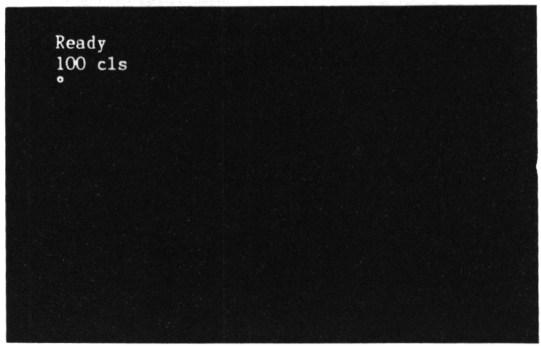
Met stapgrootte wordt bedoeld, met welke waarde de volgende stap groter is dan de regel waar men mee bezig is. Zo kan een programma van vier regels uit de regelnummers 1, 8, 10 en 20 bestaan. Maar ook de volgorde 100, 200, 300 en 400 is mogelijk. De stapgrootte mag men binnen een programma dan ook rustig veranderen.

Maar waarom zou men een stapgrootte groter kiezen dan 1. Wel, het antwoord is, als men tussen twee regels van een programma nog een andere regel wil tussenvoegen, dan moet deze stapgrootte immers groter dan 1 zijn, omdat de regelnummers altijd gehele getallen zijn. Tussen regel 11 en regel 12 kan men geen regel meer schuiven. Tussen de regels 11 en 15 zijn er echter veel meer mogelijkheden om een regel toe te voegen. Deze regel kan het regelnummer 12, 13 of 14 krijgen en wordt er automatisch in het programma tussen geplaatst. In de praktijk gebruikt men veelal een stapgrootte van 10.

Maar nu dan naar onze opdracht die we er als volgt uit willen laten zien:

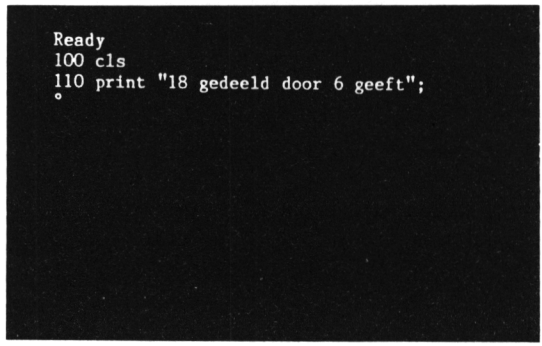
1. Wis het beeldscherm
2. Geef de tekst '18 gedeelt door 6 geeft:' uit
3. Geef de uitkomst onmiddellijk achter de tekst

Voor de oplossing van deze opdracht willen we slechts drie BASIC-regels gebruiken. Hoe gaat men bij deze opdracht dan te werk? Allereerst moet men een regelnummer kiezen. In ons voorbeeld kiezen we hier het getal 100 voor. We geven dus het getal 100 gevolgd door de opdracht voor het wissen van het beeldscherm. We herinneren ons deze opdracht als: PRINT CLS. De eerste regel zal er dan als volgt uitzien:



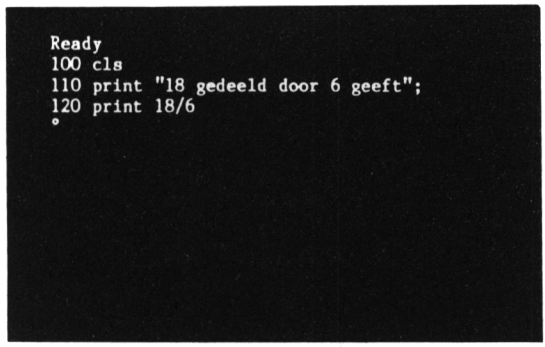
ACHTER ELK REGELNUMMER MOET EEN SPATIE GEZET WORDEN.

Dit is dan de eerste regel van ons programma. De tweede regel bestaat uit een STRING. Ook dit wordt, zoals bekend, met de opdracht PRINT gedaan. We werken met een stapgrootte van 10. De tweede regel begint dus met het getal 110. Tik dan nu deze regel in.



Achter de string wordt een punt-komma gezet, zodat de volgende PRINT-opdracht niet naar de volgende regel doorschuift, maar direkt achter de string komt te staan.

De derde regel moet U zelf bedenken. Er moet 18/6 uitgerekend worden.



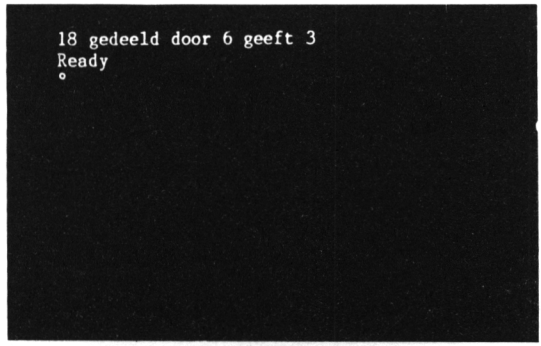
Nu is ons programma klaar en moet het alleen nog maar gestart worden.

HET STARTEN VAN EEN PROGRAMMA

Bij CPC464 staan de zojuist ingevoerde regels niet alleen op het beeldscherm, maar zij staan ook in het BASIC-geheugen opgeslagen. Ook als nu het beeldscherm gewist wordt, gaat het programma niet verloren. Het kan op elk gewenst moment gestart worden

HET COMMANDO 'RUN' START EEN BASIC-PROGRAMMA.

Geef nu eens het commando 'RUN' en kijk wat er op het beeldscherm gebeurt.



Met dit programma heeft de beginner zijn eerste resultaat geboekt.

U kunt dit programma nu zo vaak starten als U wilt. Overtuig U daar zelf eens van: geef nog eens het commando 'RUN' en nog een keer en nog een keer en...

HET VERANDEREN VAN EEN PROGRAMMA

Het kan gebeuren dat men in een programma een verandering aan wil brengen. Stelt U zich eens voor dat U het programma wilt gebruiken om de berekening 24/6 uit te voeren. Daarvoor is het niet nodig om alle drie de regels opnieuw in te tikken, maar het is voldoende om de betreffende regel te veranderen. Daarvoor moet men echter wel eerst het programma op het beeldscherm hebben.

DE OPDRACHT 'LIST' LAAT DE PROGRAMMA-REGELS OP HET BEELDSCHERM ZIEN.

'Al weer een nieuwe opdracht' zult U misschien denken. Maar dit zal, en dat is heel zeker, nog niet de laatste zijn. Om in BASIC te kunnen programmeren moet U met de belangrijkste opdrachten bekend zijn.

Geef dan nu eens het commando 'LIST', gevolgd door ENTER en kijk eens wat er gebeurt. Uw zojuist ingetikte programmaregels verschijnen op het beeldscherm. U hebt zeker opgemerkt dat Uw computer de programmaregels als hoofdletters op het beeldscherm laat verschijnen. Dit is voor de overzichtelijkheid van de 'LISTING'. Met 'LISTING' bedoelt men overigens het afdrukken van een programma, hetzij op het beeldscherm hetzij op een printer.

Wat U ook nog moet weten: als er een groter programma geLIST moet worden, dan zal het beeldscherm gaan 'scrollen'. Scrollen betekent dat er aan de onderkant van het beeldscherm steeds een nieuwe regel verschijnt, terwijl de bovenste regel dan van het beeldscherm verdwijnt. Om dit scrollen enige tijd te stoppen moet U de toets 'ESC' indrukken. Een willekeurige andere toets zal het scrollen dan weer vervolgen. Drukt U echter twee keer achter elkaar de toets 'ESC' in dan wordt de 'LISTING' afgebroken.

Komen we nu terug op het veranderen van het programma. Om de regel voor het berekenen van 24/6 te veranderen zijn er twee mogelijkheden:
1. De methode met de COPY-cursor.
2. De methode met de opdracht EDIT.

Met de COPY-cursor zijn we al eerder geconfronteerd. Maar dat mag ons niet verhinderen om dit nog een keer uit te voeren. De regel die we moeten veranderen staat nu op het beeldscherm. Ga dan nu met de COPY-cursor en met behulp van de shift toets naar het begin van die regel:

```
Ready
list
100 CLS
110 PRINT "18 gedeeld door 6 geeft";
120 PRINT 18/6
Ready
°
```

Ga vervolgens met de COPY toets naar het eerste teken dat veranderd moet worden:

```
Ready
list
100 CLS
110 PRINT "18 gedeeld door 6 geeft";
120 PRINT 18/6
Ready
110 PRINT "°
```

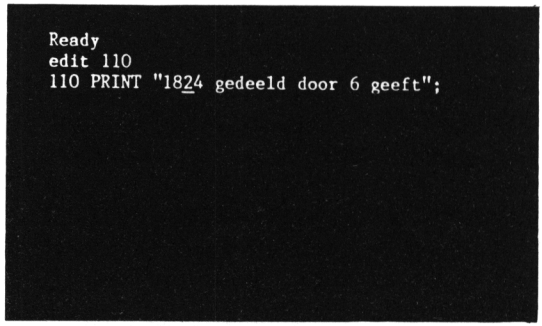
Op deze plaats wordt de eerste verandering van deze regel uitgevoerd. Voer nu het getal 24 in. Omdat bij het nogmaals indrukken van de COPY toets het getal '18' niet ook gekopieerd wordt, zet U de COPY-cursor op de spatie achter de '18' (vergeet hierbij de shift toets niet!).

```
Ready
100 CLS
110 PRINT "18 gedeeld door 6 geeft";
120 PRINT 18/6
Ready
110 PRINT "24°
```

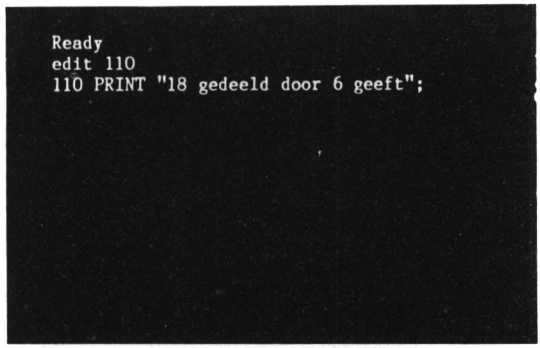
Nu wordt de rest van deze regel met de COPY toets gekopieerd. Druk nu op de toets ENTER en het veranderen vindt plaats. Opdat de uitkomst van de berekening in overeenstemming is met de afgedrukte tekst, moet ook regel 120 veranderd worden. Dit gebeurt op dezelfde manier als bij regel 110. Het is niet gemakkelijk om met de COPY-cursor om te gaan, maar ook hier geldt: oefening baart kunst. Zelf had ik in het begin met deze methode ook de nodige moeilijkheden, maar nu zou ik het niet anders meer willen.

Laten we nu de tweede methode bekijken. Met het commando 'EDIT' kan men een bepaalde regel uit een programma veranderen. De opgeroepen regel verschijnt dan op het beeldscherm. Geef nu eens het commando 'EDIT 110' (ENTER niet vergeten):

De regel 110 wordt opgeroepen en de cursor bevindt zich aan het begin van deze regel. Beweeg nu de cursor naar het teken dat als eerste veranderd moet worden. In dit voorbeeld willen we van de '24' weer een '18' maken. Als U nu het getal '18' intikt, zult U het volgende waarnemen:



Het getal '24' wordt niet met de waarde '18' overschreven, maar het getal '18' wordt gewoon toegevoegd. De EDIT opdracht werkt altijd in de invoegmodus. Het getal '24' moet nu nog verwijderd worden. Hiervoor gebruiken we de toets CLR, die, zoals inmiddels bekend is, het teken dat onder de cursor staat, wegneemt. Druk dus nu nog twee keer op deze toets:



Als alle gewenste veranderingen aangebracht zijn, drukken we op de ENTER toets. Overtuig U ervan dat de verandering ook werkelijk heeft plaatsgevonden. Geef daarom het commando 'LIST'.

U heeft nu twee manieren geleerd om veranderingen aan te brengen. Men kan zelf een keuze maken in het toepassen van de manier die men het plezierigste vindt.

AFTAKKINGEN

Programma's worden slechts zelden, zoals in het laatste voorbeeld, van voren naar achteren afgewerkt. Vaak wordt er, onder een bepaalde voorwaarde, een sprong naar een ander deel van het programma gemaakt. Ook hier is een commando voor nodig.

DE OPDRACHT 'GOTO' MAAKT EEN SPRONG NAAR EEN OPGEGEVEN REGEL.

In BASIC bestaat de opdracht 'GOTO' waarmee men naar een willekeurige regel kan springen.

Laten we dit gelijk in praktijk brengen. Zoals bekend, eindigt ons programma op regel 120. Wat gebeurt er als wij daarna, bijvoorbeeld in regel 130 naar regel 110 springen. Ik beweer dat er dan onafgebroken wordt meegedeeld '18 gedeeld door 6 geef 3'. Maar doe dat zelf eens. Tik de regel '130 GOTO 110' in en start het programma met 'RUN'.

In regel 130 is er een zogeheten eindeloze lus ontstaan, omdat ons programma steeds de regels van 100 tot 130 doorloopt. Als we nu niet ingrijpen zal het programma over een paar uur nog steeds lopen.

DE TOETS 'ESC' ONDERBREEKT HET PROGRAMMA.

Druk nu op de toets 'ESC' en het programma zal dan stoppen. Als U dan op een willekeurige andere toets drukt zal het programma weer verder gaan. Als U het programma af wilt breken, moet U een tweede keer op de toets 'ESC' drukken. De computer meldt zich dan met 'BREAK IN...', en laat daarmee zien in welke regel het programma is opgehouden. In welke regel het programma echt stopt, is toevallig.

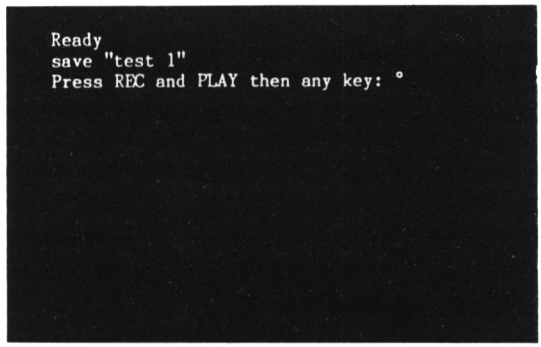
Als men het programma toch voort wilt zetten, is ook hiervoor een commando aanwezig. 'Doorgaan' is in het Engels 'continue'. Omdat de BASIC opdrachten uit het Engels komen, is dit commando 'CONT'.

DE OPDRACHT 'CONT' ZET EEN PROGRAMMA NA EEN ONDERBREKING VOORT.

Geef eens het commando 'CONT'. Het programma bevindt zich ook nu weer in de eindeloze lus en kan ook nu weer alleen maar met 'ESC' onderbroken of afgebroken worden.

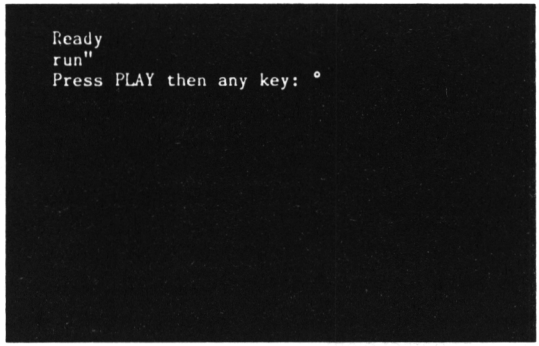
LEZEN EN SCHRIJVEN VAN PROGRAMMA'S

Als U de computer nu uitzet, bent U het programma kwijt. Omdat U zeker niet na het aanzetten van de computer steeds opnieuw het gewenste programma in wilt tikken, kan men de programma's op de ingebouwde cassette recorder bewaren. Leg een cassette in en spoel deze helemaal terug. Nu kiest U een naam voor het programma, die maximaal uit 16 tekens mag bestaan. Wij nemen als naam 'TEST 1'. Het commando om naar de recorder te schrijven is 'SAVE'. Geef nu het bijbehorende commando:



Als U nu volgens de aanwijzingen de toetsen 'REC' en 'PLAY' hebt ingedrukt, en daarop volgend een willekeurige toets, dan zal het programma naar de recorder worden weggeschreven.

Als U nu dit programma weer in wilt lezen en starten, zijn er verschillende mogelijkheden. Als het gewenste programma het eerste is dat er op de band staat, dan spoelen we eenvoudig de cassette terug en drukken we de toetsen 'CTRL' en 'ENTER' (op het cijfertoetsenblok) gelijktijdig in. Dan wordt de opdracht 'RUN' uitgevoerd. Deze opdracht zal het eerste programma dat op de band staat laden en automatisch starten. Deze opdracht kunt U natuurlijk ook zelf intikken en met de toets ENTER uitvoeren.



De computer vraagt nu aan U om op de toets 'PLAY' te drukken, en vervolgens op een willekeurige andere toets. Dan wordt deze opdracht ook uitgevoerd.

Als U echter alleen het eerste programma wilt laden maar nog niet wilt starten, geeft U de opdracht 'LOAD'. De computer meldt zich dan na het laden, met de mededeling 'READY'. Pas na het tikken van de opdracht 'RUN' zal het programma starten.

U heeft nu een tweede mogelijkheid van deze opdracht leren kennen. Het normale commando 'RUN' zal een programma dat zich reeds in het geheugen bevindt, starten. Wordt

er echter na het commando 'RUN' nog een programmanaam tussen aanhalingstekens toegevoegd, dan wordt het programma van de cassette gelezen en gestart.

De volgende voorbeelden laten een aantal verschillende mogelijkheden zien om een programma te laden:

Opdracht	Funktie
RUN (CTRL/ENTER)	Laadt het eerste programma en laat het aansluitend starten.
RUN "TEST 1"	Laadt het programma met de naam 'TEST 1' en laat het dan starten.
RUN	Start een programma dat zich reeds in het geheugen bevindt.
RUN 100	Laat een programma dat zich reeds in het geheugen bevindt op regel 100 beginnen.
LOAD"	Laadt het eerste programma in het geheugen.
LOAD "TEST 1"	Laadt het programma met de naam 'TEST 1' in het geheugen.

Een aantal van deze opdrachten ziet U hier voor het eerst. Zo kan men aan de opdrachten 'LOAD' en 'RUN' een naam van een programma toevoegen. Op de cassette wordt dan naar het programma met deze naam gezocht en na het vinden ingelezen. Hiermee heeft men dan de mogelijkheid om verschillende programma's op één cassette te zetten en weer terug te lezen.

Een tweede nieuwtje voor U is, dat er na de opdracht 'RUN' een regelnummer kan staan. Het programma wordt dan niet normaal van af het begin afgewerkt maar het start pas in de regel die achter 'RUN' staat.

Zo, dat was voorlopig alles wat men moet weten over het bewaren en teruglezen van informatie via de recorder. In het hoofdstuk 'De cassette recorder' wordt op alle mogelijkheden ingegaan die dit apparaat biedt.

VERWIJDEREN VAN EEN PROGRAMMA

Zoals inmiddels bekend is, wordt een programma bij het uitschakelen van de computer gewist. Toch is dit zeker niet de enige mogelijkheid.

DE OPDRACHT 'NEW' VERWIJDERT HET PROGRAMMA DAT IN HET GEHEUGEN STAAT.

Geef eens de opdracht 'NEW', en U zult zien dat het programma verdwenen is. Eerst moet natuurlijk dat programma op een cassette of diskette (daarover later) worden weggeschreven.

Voor alle sceptische lezers: als U na de opdracht 'NEW' de opdracht 'LIST' geeft zult U zien dat het programma werkelijk verdwenen is. Er worden geen regels meer afgedrukt. Ook de opdracht 'RUN' heeft geen effect meer.

U heeft nu de eerste beginselen voor de opbouw van een programma geleerd, dat voor U een begin kan zijn voor het verwerken van de volgende stap. Als de stof van dit hoofdstuk toch nog niet geheel verwerkt is, raad ik U aan om dit hoofdstuk nog eens door te nemen.

Hulp bij het programmeren

Het omvangrijke BASIC van Uw CPC464 is het uitgangspunt van dit hoofdstuk. Lang niet elke computer heeft in de standaardversie zoveel commando's, waarmee het werken met, en het opbouwen van programma's dermate vereenvoudigd wordt, als met Uw CPC464. Er bestaan opdrachten voor het wissen van bepaalde regels, voor het automatisch nummeren van regels, om er maar twee te noemen van de hierna te omschrijven programmeer-hulpen. Iedereen die met plezier wil omgaan met BASIC 1.0. moet dit hoofdstuk niet ongelezen laten.

AUTOMATISCHE REGELNUMMERING

Bij het opbouwen van een programma wordt als stapgrootte meestal 10 gekozen. Het intikken van het regelnummer bij het begin van elke regel kan iemand tamelijk vervelend vinden. Dat hebben ook de ontwerpers van het BASIC dat in Uw computer zit, bedacht. Daarom hebben ze een comfortabele opdracht ontwikkeld waarmee men automatisch de regels kan nummeren. Laten we eerst de beschrijving van deze opdracht bekijken:

Probleem:	Automatische regelnummering
Commando:	AUTO start, afstand
Parameter	start – beginregel afstand – stapgrootte
Voorbeeld:	AUTO 100,10 Te beginnen bij regel 100 worden alle volgende regels met de stapgrootte 10 vervolgd. (100, 110, 120, 130 ....)
Opmerking:	De automatische nummering kan met de toets 'ESC' beëindigd worden. Als een regel reeds bestaat, wordt dit kenbaar gemaakt doordat er een sterretje (*) achter het regelnummer wordt afgedrukt. Met ENTER blijft deze regel bestaan.

Hier ziet U voor het eerst een nauwkeurige omschrijving van een opdracht, zoals dat ook in het verdere verloop van dit boek bij elke nieuwe opdracht gedaan zal worden. Er duikt hier ook een vreemd woord op: Parameter. Met parameter bedoelen we alles wat achter een opdracht gezet MOET worden. Zo moet men bv. achter de opdracht 'LOAD' een parameter toevoegen en wel de naam van een programma.

Maar terug naar de opdracht AUTO. Laten we eens aannemen dat U de eerste regels van een programma in wilt voeren, te beginnen bij regel 10 en een stapgrootte van 5. Gebruikelijk is, dat men bij het intikken van een programma steeds begint met een regelnummer. Onze eerste regel begint dus met 10, de tweede regel krijgt dan het nummer 15 enz. Dit werk wordt U, zoals reeds gezegd, door de opdracht 'AUTO' afgenomen. Probeer deze opdracht nu zelf eerst eens, en geef het commando:

AUTO 10,5

Direkt hierna verschijnt het getal 10 op het beeldscherm. De computer wacht nu tot er een invoer op deze regel heeft plaatsgevonden. Geef hier bijvoorbeeld eens de opdracht 'PRINT "REGEL 10" ' en beëindig deze regel met ENTER. Nu verschijnt de tweede regel (15) en de computer wacht weer op een invoer. Geef hier eens de opdracht 'PRINT "REGEL 15" '. Hierna beëindigen we de automatische regelnummering met de toets 'ESC'. De mededeling 'READY' verschijnt op het beeldscherm waarmee aangegeven wordt dat de automatische regelnummering niet meer aktief is.

Geef nu nog eens de opdracht 'AUTO 10,5'. We leren hiermee een bijzonderheid van onze computer. De computer reageert als volgt

10*

Wat betekent nu dat sterretje? Als U de omschrijving van deze opdracht goed hebt begrepen, zult U zeker de oplossing weten. Het sterretje geeft aan dat deze regel reeds bestaat. Dit is ook erg belangrijk omdat anders het programma kan worden overschreven. Als U nu wilt dat deze regel onveranderd blijft, drukt U de toets ENTER in. Als deze regel echter veranderd moet worden, tikt U gewoon de nieuwe regel in en sluit U dit weer af met ENTER. Probeer dit nu een keer uit en druk bij de regel 10 de toets ENTER in. Geef dan bij regel 15 een nieuwe opdracht 'PRINT "NIEUWE REGEL 15" '. Hierna beëindigt U de automatische regelnummering met 'ESC', en bekijk eens de uitwerking met 'LIST'.

We hebben zojuist een mogelijkheid leren kennen om het invoeren van een programma te vereenvoudigen. Maak hiervan gebruik bij het intikken van programma's die in dit boek staan.

HERNUMMEREN VAN REGELS

Zoals U inmiddels al weet, heeft de stapgrootte van een programma niets uitstaande met het aflopen van dit programma. Of U nu de afstand 1, 5 of 10 maakt, dat ligt alleen maar aan Uw inschatting. U moet wel in de gaten houden dat een te kleine stapgrootte het tussenvoegen van programmaregels moeilijker (zo niet onmogelijk) maakt. Laten we dit

eens aan de hand van een praktisch voorbeeld duidelijk maken. Tik allereerst het volgende programma in:

```
1 CLS
2 PRINT "We vermenigvuldigen nu twee getallen"
3 PRINT "Het eerste getal is 15"
4 PRINT "Het tweede getal is 12"
5 PRINT "De uitkomst is";15*12
```

Zoals U ziet is de stapgrootte bij dit programma 1. Starten we dit programma dan krijgen we het volgende resultaat:

```
We vermenigvuldigen nu twee getallen
Het eerste getal is 15
Het tweede getal is 12
De uitkomst is 180
```

Om deze uitvoer optisch te verbeteren willen we de eerste regel onderstrepen. De uitvoer op het beeldscherm moet er dan als volgt uit zien:

```
We vermenigvuldigen nu twee getallen
Het eerste getal is 15
Het tweede getal is 12
De uitkomst is 180
```

En dus moet er na regel 2 van dit programma een nieuwe regel tussengevoegd worden. Hiervoor is het nodig om alle regels na regel 2 met 1 te verhogen, zodat we de nieuwe regel de waarde 3 kunnen geven. Het volledige programma ziet er dan als volgt uit:

```
1 CLS
2 PRINT "We vermenigvuldigen nu twee getallen"
3 PRINT "----------"
4 PRINT "Het eerste getal is 15"
5 PRINT "Het tweede getal is 12"
6 PRINT "De uitkomst is";15*12
```

Hoe zou U nu dit probleem oplossen? U zou zeker de regels 3 tot 6 opnieuw met het regelnummer intikken. Dit is echter niet de bedoeling van de uitvinder. Uw zeer comfortabele BASIC heeft hier een commando voor in petto, waarvan we eerst de omschrijving willen geven:

Probleem:	Hernummeren van regels
Opdracht:	RENUM nieuw, oud, afstand
Parameter:	Nieuw – nieuwe begingregel Oud – oude beginregel Afstand – nieuwe stapgrootte
Voorbeeld:	RENUM 100, 10, 5 Vanaf regel 10 van het programma wordt met als eerste regelnummer 100 en de stapgrootte 5 verder genummerd.
Opmerking:	De laatste regel van het programma mag niet groter zijn dan 65535 anders wordt het commando RENUM niet uitgevoerd en zal de foutmelding "Improper argument" op het beeldscherm verschijnen.

Deze opdracht is iets gecompliceerder dan de opdrachten die tot nu toe behandeld zijn. Maar dit zal ons niet ontmoedigen. U ziet dat er bij deze opdracht drie parameters nodig zijn. Allereerst is daar het regelnummer waar het hernummeren moet beginnen. Daarna volgt het regelnummer dat veranderd wordt. U hoeft dus niet alle regelnummers te veranderen. De derde parameter tenslotte is de nieuwe stapgrootte.
Hoe moet nu de opdracht RENUM opgebouwd worden om in ons voorbeeld de regels 3 tot 5 te veranderen in de regels 4 tot 6. We bouwen deze opdracht stap voor stap op. Om te beginnen moeten we het eerste nieuwe regelnummer invoeren. Dit is nummer 4. De tweede parameter staat voor het eerste oude regelnummer dat veranderd moet worden en dat is hier regelnummer 3. De nieuwe stapgrootte moet verder 1 zijn. Het commando moet er dus als volgt uit zien:

```
RENUM 4, 3, 1
```

Het resultaat van die commando is, zoals met LIST zichtbaar gemaakt kan worden:

```
1 CLS
2 PRINT "We vermenigvuldigen nu twee getallen"
4 PRINT "Het eerste getal is 15"
5 PRINT "Het tweede getal is 12"
6 PRINT "De uitkomst is";15*12
```

Met slechts een commando werd het programma zo ingericht dat er plaats kon ontstaan voor de regel die we nu toe kunnen voegen:

```
3 PRINT "----------"
```

Om soortgelijke problemen in de toekomst te vermijden werken we met de stapgrootte 10. Hoe moet nu de opdracht 'RENUM' er uit zien als we in ons voorbeeldprogramma vanaf regel 100 opnieuw willen nummeren met een stapgrootte 10. Dit kan voor U nu geen probleem meer zijn. Geef daarvoor de opdracht:

```
RENUM 100,1,10
```

De opdracht 'RENUM' hernummert niet alleen de regels maar doet daarnaast ook nog iets anders dat erg belangrijk is. Hij verandert ook nog de sprong-opdrachten mee. Wat betekent dat? Laten we dit eens proberen. Voeg er nog de volgende regel bij:

```
100 CLS
110 PRINT "We vermenigvuldigen nu twee getallen"
120 PRINT "----------"
130 PRINT "Het eerste getal is 15"
140 PRINT "Het tweede getal is 12"
150 PRINT "De uitkomst is"; 15*12
160 GOTO 110
```

We hebben met deze laatste regel een eindeloze lus ingebouwd. Als we nu dit programma hernummeren, krijgt regel 110 een andere waarde. De GOTO opdracht moet nu echter niet meer naar regel 110 uitgevoerd worden, omdat deze na het hernummeren waarschijnlijk niet meer bestaat. Om dit te voorkomen, verandert de opdracht RENUM niet alleen de regelnummers aan het begin, maar ook de sprong-adressen in deze regels. Dit is erg belangrijk, omdat een opdracht anders niet uitgevoerd kan worden. Overtuig U er zelf van, als U het commando 'RENUM 10,100,5' geeft. Als U nu het programma met 'LIST' bekijkt, zult U opmerken dat het sprong-adres achter de GOTO-opdracht ook veranderd is.

U beheerst nu een bevel waarvan U in de toekomst erg veel plezier zult beleven.

VERWIJDEREN VAN REGELS

U heeft inmiddels met de opdracht 'NEW' kennis gemaakt, waarmee het in het geheugen staande programma wordt gewist. Maar wat, als niet het gehele programma, maar slechts een deel ervan gewist moet worden? Het wissen van een enkele regel levert geen enkele moeilijkheid op. U geeft eenvoudig dat regelnummer en de opdracht ENTER. Om dit te testen, wissen we het vorige programma en voeren dan een nieuw programma in. Gebruik hiervoor het commando AUTO:

```
10 PRINT
20 PRINT
30 PRINT
40 PRINT
50 PRINT
60 PRINT
70 PRINT
80 PRINT
90 PRINT
```

In dit programma staan er weliswaar slechts 9 spaties, maar dat is voor de volgende oefeningen meer dan voldoende. We wissen nu regel 50 door dit getal, gevolgd door de toets ENTER in te tikken. Als U nu dit programma 'LIST' zult U het resultaat zien. Regel 50 bestaat niet meer. Wat echter als er meer regels, bv. van 20 tot 40 gewist moeten worden? Zeker, U kunt dan de regels 20, 30 en 40 apart wegwissen, maar bij grote aantallen is dat wel erg omslachtig. De volgende opdracht zal dit werk vereenvoudigen:

Probleem:	Wissen van groepen regels
Commando:	DELETE begin – einde
Parameter:	Begin – eerste regel die gewist moet worden Einde – laatste regel die gewist moet worden
Voorbeeld:	DELETE 40 – 130 wis regels 40 tot 130

We hebben nu de oplossing: om de regels 20 tot 40 te wissen moeten we als commando geven:

```
DELETE 20 – 40
```

Er bestaan meer mogelijkheden om deze parameters toe te passen, bekijk hiervoor de volgende tabel:

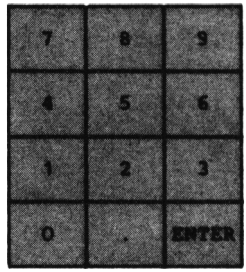
DELETE 10	Wist regel 10
DELETE 100-150	Wist regels 100 tot 150
DELETE 150-	Wist alle regels vanaf 150
DELETE -100	Wist alle regels tot en met 100

Deze mogelijkheid, om de getallen te variëren, bestaat ook bij het commando LIST. Ook hier geven we een tabel om dit te verduidelijken:

LIST 10	Geeft een LIST van regel 10
LIST 100-150	Geeft een LIST van regel 100 tot 150
LIST 150-	Geeft een LIST vanaf regel 150
LIST -100	Geeft een LIST tot en met regel 100

FUNKTIETOETSEN

Nu zult U kennis maken met een van de meest comfortabele hulpen bij het programmeren. Het is een vervelend gebeuren om commando's die men vaak gebruikt steeds opnieuw geheel in te tikken. Uw CPC464 biedt hiervoor de mogelijkheid, om opdrachten die men vaak nodig heeft, door het intikken van een getal, te verwezenlijken. Hiervoor kan men de toetsen op het cijferblok gebruiken:

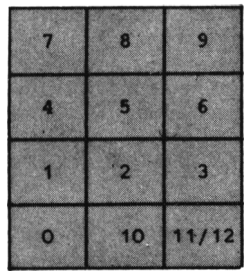


Op elk van deze toetsen kan men nu een rij van 32 tekens vastleggen, gezamenlijk echter niet meer dan 120 tekens. Bekijken we eerst de omschrijving van dit commando:

Probleem:	Bepalen van de funktie-toetsen
Commando:	KEY n, string
Parameter:	n – nummer van de funktie-toets string – max. 32 tekens
Voorbeeld:	KEY 0, "LIST" Bepaalt het commando "LIST" op de funktie-toets 0
Opmerking:	Het totaal aantal tekens mag de 120 niet overschrijden

Is dit commando niet uiterst interessant? Laten we dit onmiddellijk gebruiken. Er zijn 13 funktie-toetsen met de nummers 0 tot 12. Hierbij rijst de vraag hoe men 13 funktietoetsen met behulp van 12 toetsen kan realiseren. Dit is mogelijk omdat de toets ENTER van het cijferblok twee funkties bezit. De ene door het normaal intoetsen van ENTER en de andere

samen met de toets 'CTRL'. Het volgende diagram laat zien welke waarden de toetsen hebben.



De toets ',' heeft dus het nummer 10, de toets 'ENTER' de nummers 11 en 12. Laten we nu eens onder een aantal toetsen een zinvolle opdracht vastleggen. De opdracht 'LIST' wordt vaak gebruikt. We leggen dit vast onder de funktie toets 0. Dit wordt met de volgende opdracht gedaan:

```
KEY 0,"LIST"
```

Vergeet de spatie tussen de opdracht en het getal niet! De computer neemt U dat direkt kwalijk (SYNTAX ERROR). Als U nu op de funktie-toets '0' drukt dan verschijnt de opdracht 'LIST' op het beeldscherm. Maar deze opdracht wordt nog niet uitgevoerd. Daarvoor moet U eerst op de toets 'ENTER' drukken. Doch ook het 'ENTER' kan met dezelfde funktie-toets worden opgelost. Daarvoor moet U er echter de interne code voor dat 'ENTER', met een speciale opdracht, die we later nog wel leren, aanhangen. Deze code is 13. De hele opdracht ziet er dan als volgt uit:

```
KEY 0,"LIST"+CHR$(13)
```

Als U daar dan ook nog de 'ENTER' aan vast wilt knopen, ziet deze opdracht er al heel gecompliceerd uit:

```
KEY 1,"RUN"+CHR$(34)+CHR$(13)
```

U doet er verstandig aan om de twee codes 13 en 34 te onthouden!

Te zijner tijd zult U wel een standaard kiezen voor Uw funktie-toetsen. Om dit bij het inschakelen van Uw computer niet steeds opnieuw in te moeten tikken, kunt U daar het beste een programma voor maken. Dit programma kunt U dan op de cassette band zetten, waardoor het steeds voor U bereikbaar blijft. Een dergelijk programma kan er dan bijvoorbeeld als volgt uitzien:

```
10 KEY 0,"LIST"+CHR$(13)
20 KEY 1,"PRINT"
30 KEY 2,"NEW"
40 KEY 3,"RENUM"
50 KEY 4,"AUTO"
60 KEY 5,"DELETE"
```

Ontwikkel tijdens het verwerken van de volgende hoofdstukken Uw persoonlijk funktie-toetsen programma.

Nog een belangrijk gegeven tot slot: de drie-vinger-greep (SHIFT/CTRL/ESC), waarmee de computer in de beginstand komt, zal ook de funktie-toetsen uitschakelen!!

Het invoeren van BASIC

Met dit hoofdstuk zullen we de programmeertaal BASIC aan U voorstellen. Daarbij willen we niet alle BASIC opdrachten opsommen, maar wordt een adressenadministratie stap voor stap opgebouwd, waarin de nodige opdrachten dan stuk voor stuk zullen worden beschreven en ingevoerd. De lezer van dit boek zal hierdoor niet direkt een perfecte programmeur worden, maar wel intensief met de praktijk van het programmeren geconfronteerd worden. Deze confrontatie biedt hem de beste voorwaarden om zich met behulp van andere literatuur verder te ontwikkelen. Omdat niet alle commando's van de CPC464 worden behandeld, vindt U aan het einde nog een komplete lijst van alle commando's.

PROBLEEMOMSCHRIJVING EN ADRESSENBEHEER

Een adressenbeheer is een van de meest gewilde programma's op een home-computer. De brede toepasbaarheid draagt hier wezenlijk toe bij. Er bestaat nauwelijks iemand die dit programma niet kan gebruiken. Weliswaar is dit adressenbestand met slechts een paar adressen niet bepaald effectiever dan het gebruik van een adressenboekje. Maar als het om veel adressen gaat is de home-computer veel nuttiger.

Welke mogelijkheden worden er met een dergelijk programma dan geboden? Wel, in elk geval moet men de adressen kunnen in- en uitgeven.

Voordat men aanspraak kan maken op dit programma, moeten we nog eerst iets duidelijk maken. In dit programma, waar we gegevens gebruiken, moeten we een onderscheid maken tussen twee bestanddelen:

1. Het programma
2. De gegevens

De gegevens zijn geen onderdeel van het programma. Dit zou wel mogelijk zijn, maar deze gegevens kunnen dan alleen maar door de programmeur en niet door de gebruiker ingebracht worden. Een verandering van deze gegevens heeft dan gelijktijdig een verandering van het programma tot gevolg, die moeten we dus direkt weer vergeten.

Dit programma bevindt zich dus in een extern geheugen (cassette of diskette) en wordt ingelezen als dat nodig is. Maar hoe staat dat dan met de gegevens? Dit is de eerste grote vraag. Wat gebeurt daarmee?

U moet deze gegevens organiseren. Deze gegevens vormen een verzameling op een extern geheugenmedium. Er bestaan natuurlijk een aantal organisatievormen. Wij zullen hier de meest eenvoudige methode behandelen, de sequentiële data. Sequentieel betekent

dat deze data achter elkaar staan. De twee belangrijkste bestanddelen van ons programma voor het verwerken van een adressenbestand zijn:

1. Het programma
2. De data

ORGANISATIE VAN DATA

Als U nu zou vragen welk gedeelte het eerst georganiseerd zou moeten worden, zouden velen het antwoord geven 'het programma'. Dit antwoord is echter fout. Eerst moet er duidelijk zijn, wat, waar en hoe, er DATA moeten worden weggeschreven. De vraag 'hoe' is al beantwoordt, sequentieel. Het 'waar' is nog niet duidelijk. Hiervoor zijn er twee mogelijkheden:

1. De cassette
2. De diskette

We willen eerst onze data op de ingebouwde cassette inlezen. Omdat er tijdens het schrijven van dit boek nog geen disk-drive te verkrijgen was, wordt U vriendelijk verzocht om de bijbehorende delen van dit programma overeenkomstig te veranderen.

Blijft nu nog de vraag, wat moet er bewaard worden. 'Adressen natuurlijk' zal van vele kanten spontaan geroepen worden, maar weten we al wat er allemaal tot het 'adres' moet horen. Natuurlijk niet. Laten we eerst eens verzamelen wat er zoal tot een 'adres' gerekend kan worden:

- Titel
- Voornaam
- Naam
- Straat
- Postcode
- Plaats
- Telefoonnummer
- Opmerking

Deze aparte onderdelen van het adres noemt men een 'data-veld'. Zo spreekt men bijvoorbeeld over het veld 'Titel'. Het totaal aan data-velden noemt men een data-regel. Elke regel bestaat dus in ons voorbeeld uit 8 velden. Alle data-regels samen bepalen dan de DATA. Bekijken we deze struktuur eens in het volgende beeld:

Veld 1	Veld 2	Veld 3	..... Veld n	Dataregel 1
Veld 1	Veld 2	Veld 3	..... Veld n	Dataregel 2
Veld 1	Veld 2	Veld 3	..... Veld n	Dataregel 3
Veld 1	Veld 2	Veld 3	..... Veld n	Dataregel 4
Veld 1	Veld 2	Veld 3	..... Veld n	Dataregel n

De hiërarchie is dus DATA – Data regel – Data veld

INTERN GEHEUGEN VOOR DATA

Met de sequentiële data kan men weliswaar handig werken, maar er bestaan toch ook weer nadelen tegenover de andere vormen van organisatie. Laten we eens aannemen dat het gehele adressenbestand zich op een cassetteband bevindt. U zou nu een heel bepaald adres van dit bestand willen hebben. Hier is dan het probleem. U kunt uit een geheel data-bestand niet een aparte data-regel lezen. Een cassette-recorder en ook een diskdrive is niet in staat om uit een sequentieel bestand een enkele data-regel te lezen. Hoe kan men dan deze adressen wel bereiken?

Om de data-regel te kunnen gebruiken moet het gehele bestand eerst in het geheugen van de computer worden ingelezen. Een sequentieel bestand mag daarbij niet groter zijn dan de geheugenruimte van de computer. Hiervan is het voordeel dat als de gegevens eenmaal ingelezen zijn, de computer bliksemsnel deze data kan opzoeken. Voor dit deel van het programma gelden de volgende regels:

1. Data inlezen
2. Data-regels lezen, veranderen, wissen...
3. Data wegschrijven

Na het starten van het programma moet men dus eerst alle adresgegevens in het geheugen van de computer inlezen. Daarna kan men met de data-regels werken. Voordat het programma beëindigd wordt moet men echter deze datagegevens weer op de cassette of diskette schrijven, voorzover deze veranderd zijn. Als deze data-regels niet veranderd zijn, er geen data-regels zijn gewist en er ook geen nieuwe aan toegevoegd zijn, dan zijn de data nog dezelfde als de oorspronkelijke en hoeft dit bestand niet opnieuw naar de cassette of diskette weggeschreven te worden.

VARIABELEN

Zoals bekend is, worden de data-regels naar het geheugen van de computer geschreven. De vraag is alleen waar en hoe?

DE DATA WORDEN ALS VARIABELEN IN HET GEHEUGEN OPGENOMEN.

Vaiabele is dus een toverwoord. Een variabele is een geheugenbereik in de CPC464 waar een naam aan wordt toegevoegd. Door het oproepen van deze naam kan men dit deel van het geheugen gebruiken. We hebben al twee verschillende data-typen leren kennen. Zo zijn er numerieke data (getallen) en alfanumerieke data (strings en teksten). String-variabelen herkent men aan het teken '$' achter de naam van deze variabelen. Hoe ziet een variabele er nu uit? Allereerst bestaat de naam van een variabele uit ten hoogste 40 tekens. Het eerste teken MOET een letter zijn, de andere tekens mogen letters of cijfers zijn. Er zijn een paar uitzonderingen. Men mag geen BASIC-commando's of funkties gebruiken:

bijv. PRINT, PI, KEY

Voorbeelden van numerieke variabelen:

Teller
PLZ
Telefoon
Bedrag

Voorbeelden van alfanumerieke (string) variabelen:

Titel$
Artikelnaam$
Disconto$
X$

Bij de keuze van de naam van de variabele kan men het beste een zinvolle aanduiding nemen. Zo wordt de naam van de variabele waarmee men voornamen in het geheugen op wil slaan bv. 'Voornaam$' genoemd. De inkoopprijs van een artikel zou men 'INKOOPPRIJS$' kunnen noemen enz. Een zinvolle naam voor een variabele draagt bij tot het overzichtelijk lezen van een programma.

VERWERKEN VAN VARIABELEN

Na lange tijd gaan we nu weer met de computer verder. We gaan ons bezighouden met het opbouwen, het verwerken en het afdrukken van variabelen.

Het opslaan in het geheugen van data in de vorm van variabelen is eigenlijk kinderspel. Laten we eens aannemen dat we het getal 45.12 onder de naam 'INKOOP' in het geheugen op willen slaan. De bijbehorende opdracht luidt dan:

```
INKOOP=45.12
```

Dat is toch wel erg gemakkelijk, nietwaar? Als nu deze variabele op het beeldscherm moet verschijnen, tikken we eenvoudig in:

```
PRINT INKOOP
```

Tik deze beide opdrachten nu eens zelf in op Uw computer. Als deze variabele nu weer gewist moet worden, geven we eenvoudig de opdracht:

```
INKOOP=0
```

De variabelen kan men ook in berekeningen gebruiken. Geef eens de dubbele waarde van INKOOP. Als U inmiddels de variabele gewist heeft, geef dan eerst weer INKOOP=45.12.

```
PRINT INKOOP*2
```

De computer verdubbelt nu INKOOP en geeft deze nieuwe uitkomst (90.24). De variabele zelf behoudt zijn waarde 45.12. Wat nu echter als de variabele zelf verdubbeld moet worden? Omdat de variabele zelf een andere waarde moet krijgen, begint deze opdracht met INKOOP=. De totale opdracht voor het verdubbelen van de variabele INKOOP luidt dan:

```
INKOOP=INKOOP*2
```

Er wordt altijd eerst de waarde aan de rechterzijde van het is-gelijkteken uitgerekend, en daarna krijgt de variabele aan de linkerzijde deze waarde toegevoegd.

Stringvariabelen worden op dezelfde manier behandeld. Laten we eens de string "UTRECHT" als stringvariabele STAD$ in het geheugen van de computer opnemen. De opdracht hiervoor luidt:

```
STAD$="UTRECHT"
```

Ook stringvariabelen worden met de printopdracht op het beeldscherm gezet. Zie zelf:

```
PRINT STAD$
```

En wat gebeurt er nu? Natuurlijk, de string waarvan de variabele STAD$ luidt, wordt op het beeldscherm afgedrukt. Als deze string weer gewist moet worden, wordt de nieuwe string eenvoudig de zogenoemde 'lege string':

```
STAD$=""
```

Een lege string bestaat dus uit twee naast elkaar liggende aanhalingstekens. Het spreekt vanzelf dat men met strings niet kan rekenen. Ook niet als de string een getal voorstelt. Dat wil dus zeggen, als U bijvoorbeeld X$="123" noemt, kan daar niet mee gerekend worden.

Men kan strings echter wel achter elkaar zetten. de volgende rij opdrachten laat dit zien:

```
STAD1$="UTRECHT"
STAD2$="ZUID"
STAD$=STAD1$+STAD2$
PRINT STAD$
```

Allereerst worden de twee namen in STAD1$ en STAD2$ opgeslagen. Daarna ontstaat de derde string (STAD$), waarin de twee vorige strings zijn opgenomen. Met de vierde opdracht zal dan de string STAD$ op het beeldscherm verschijnen.

Het zou nuttig zijn om tussen deze twee strings nog een spatie te zetten. Om dit te bereiken moet de derde opdracht als volgt veranderd worden:

```
STAD$=STAD1$+" "+STAD2$
```

Er is dan een spatie tussen de twee namen onstaan.

We hebben nu de twee belangrijkste typen variabelen leren kennen. De numerieke variabele en de stringvariabele, waarbij deze tweede aan het dollar teken ($) te herkennen is. In zulke variabelen kan men ook adressen opslaan. Maar dit is niet zinvol. Voor het opslaan van meerdere gelijksoortige data-regels in het geheugen van Uw computer zijn er andere typen variabelen nodig.

TABELLEN

Bij het verwerken van data worden vaak tabellen gebruikt. Als U van een groep meerdere gegevens in het geheugen van uw computer op wilt slaan, is het erg omslachtig om elk element van deze groep een andere naam te geven. In een dergelijk geval krijgen alle elementen dezelfde naam echter met een toevoeging. Nemen we als voorbeeld een groep van vijf namen van steden:

BONN
PARIJS
LONDEN
AMSTERDAM
BRUSSEL

Om deze vijf namen in het geheugen op te slaan moeten we op de vorige manier vijf variabelen nemen. Als we deze groep echter als tabel (ook wel ARRAY genoemd) in het geheugen opslaan, dan is een stringvariabele voldoende. Omdat de elementen toch verschillend zijn, moet dit aangegeven worden. Hiervoor wordt er achter de string een getal tussen haakjes gezet. Dit getal noemt men de index. Het eerste element uit de groep krijgt dus de index 1, het tweede element krijgt de index 2 enz. De index begint weliswaar bij het getal 0 en niet bij 1, maar hierdoor gaat het overzicht verloren. Men kan de index 0 ook verwaarlozen. Dit kost wel wat geheugenruimte maar een foutenbron wordt ermee voorkomen.

Een tabel of ARRAY-variabele is bijvoorbeeld ADRES$(1). Ook XY(212) is een numerieke ARRAY-variabele. De index kan dus willekeurig groot genomen worden. Men moet echter op de twee volgende dingen letten:

1. Als de index groter wordt dan 10 dan moet men hiervoor geheugenruimte reserveren.
2. De ARRAY mag niet zo groot worden dat de geheugenruimte wordt overschreden.

De computer zal dus ARRAY's waarvan de index niet groter is dan 10 normaal kunnen verwerken. Maar hoe worden dan de grotere ARRAY's gedefinieerd? Hiervoor bestaat er een BASIC opdracht, de opdracht 'DIM'.

Probleem:	Dimensioneren van een ARRAY
Commando:	DIM v1 (n1,n2,n3...)
Parameter:	v1 = naam van de ARRAY-variabele n1, n2, n3... maximale index van de diverse variabelen

Voorbeeld:	DIM D$(20) Er moet een ARRAY met de naam D$ worden gemaakt, waarvan er 20 variabelen zijn.
Opmerking:	Elke ARRAY mag in een programma slechts één keer gedimensioneerd worden anders ontstaat de foutmelding 'ARRAY ALREADY DIMENSIONED'

Als de grootste index bv. 5 zal zijn, is er geen DIM-aanwijzing nodig, omdat tot index 10 de computer het werk wel kan. Deze zaak heeft een kleine moeilijkheid: als er slechts tot index 5 ruimte nodig is, dan wordt er ook ruimte gereserveerd voor de indexen 6 tot 10, wat eigenlijk niet nodig is. Als hier echter een DIM aanwijzing plaatsvindt, zal er maar ruimte gereserveerd worden tot index 5.

Maar nu terug naar de namen van onze vijf steden, waarvan we een ARRAY willen maken. Als we de naam STAD$ willen gebruiken, dan zullen we de steden met de volgende toevoegingen in het geheugen opslaan:

```
10 STAD$(1)="BONN"
20 STAD$(2)="PARIJS"
30 STAD$(3)="LONDEN"
40 STAD$(4)="AMSTERDAM"
50 STAD$(5)="BRUSSEL"
```

Men kan dit ook openen met 'DIM STAD$(5)' waardoor er geen geheugenruimte voor de indexen 6 tot 10 verloren gaat.

Als er een index gebruikt wordt die groter is dan de gereserveerde ruimte, ontstaat er de mededeling 'Subscript out of range'. De index heeft ook wel de naam 'Subscript'.

In het verdere deel van dit voorbeeld zullen we de landen die bij de genoemde hoofdsteden horen ook in een Array onderbrengen. De Array krijgt de naam LAND$. En we maken dan de volgende toevoegingen:

```
 60 LAND$(1)="DUITSLAND"
 70 LAND$(2)="FRANKRIJK"
 80 LAND$(3)="ENGELAND"
 90 LAND$(4)="NEDERLAND"
100 LAND$(5)="BELGIË"
```

We hebben nu twee tabellen opgebouwd, waarbij tussen de indexen een samenhang bestaat. Dat wil zeggen, STAD$(1) is de hoofdstad van LAND$(1). In het algemeen kunnen

we zeggen dat STAD$(X) de hoofdstad is van LAND$(X).

Het fraaie van het werken met indexen is, dat de index niet alleen maar een getal kan zijn, maar dat het evengoed een variabele, of zelfs de een of andere rekenuitdrukking kan zijn. Zo kan als index bijvoorbeeld gekozen worden: 'D$(X-2*1+13)'. De mogelijkheden van deze manier van indexeren zijn onuitputtelijk.

De zojuist genoemde tabellen noemt men 1-dimensionaal, omdat zij slechts 1 index bezitten. Men kan echter ook Array's met een willekeurig aantal indices construeren. Inplaats van het opbouwen van twee array's, zoals in het vorige voorbeeld, kan men ditzelfde ook met een 2-dimensionale array bereiken. Bekijk daarvoor eens de volgende programmastappen:

```
 10 DIM TABEL$(5,1)
 20 TABEL$(1,0)="BONN"
 30 TABEL$(2,0)="PARIJS"
 40 TABEL$(3,0)="LONDEN"
 50 TABEL$(4,0)="AMSTERDAM"
 60 TABEL$(5,0)="BRUSSEL"
 70 TABEL$(1,1)="DUITSLAND"
 80 TABEL$(2,1)="FRANKRIJK"
 90 TABEL$(3,1)="ENGELAND"
100 TABEL$(4,1)="NEDERLAND"
110 TABEL$(5,1)="BELGIË"
```

De definitie van DIM in regel 10 is weliswaar niet nodig omdat de computer tot DIM(10,10) zonder meer uit zal voeren, maar zonder de DIM definitie zal de computer 11*11 = 121 plaatsen reserveren, terwijl we er slechts 10 nodig hebben. Bij het toepassen van twee of meer dimensionale tabellen is het aan te bevelen om de grootte van de tabel zo nauwkeurig mogelijk te kiezen, omdat er anders erg veel geheugenruimte verloren gaat.

We beschikken nu over een twee-dimensionale tabel, waarvan de tweede index de groeps-aanduiding is (stad of land) en de eerste de plaats in deze tabel. Men kan dus deze tabel ook als matrix opschrijven:

	0	1
1	BONN	DUITSLAND
2	PARIJS	FRANKRIJK
3	LONDEN	ENGELAND

4	AMSTERDAM	NEDERLAND
5	BRUSSEL	BELGIË

Laten we nu weer teruggaan naar ons adressenbestand. Een twee-dimensionale tabel is ideaal voor het opslaan van sequentiële data. De eerste index nummert de data-regels, de tweede index nummert de velden in deze data-regel. Omdat er in onze data-regel 8 velden nodig zijn, is de tweede index maximaal 8. De eerste index, dat is dus het aantal adressen, kan men vrij kiezen. Laten we eens aannemen dat er 200 adressen in ons bestand moeten voorkomen. De naam van de tabel noemen we ADRES$. De DIM definitie wordt dan:

```
DIM ADRES$(200,8)
```

Hoe deze tabel gebruikt wordt, zullen we later nog wel zien.

INVOEREN VAN DATA VIA HET TOETSENBORD

In de oefeningen die we tot nu toe hebben gedaan, hebben we de data alleen maar ingevoerd en verwerkt. Interessant wordt het pas als we de data via het toetsenbord in kunnen voeren. Het commando hiervoor is INPUT, maar INPUT alleen heeft geen betekenis. Bij dit commando behoren twee parameters: een string die voor de eigenlijke INPUT staat en een variabele, waar de ingevoerde data in worden opgeslagen. Belangrijk: de INPUT opdracht kan niet in direct mode gebruikt worden, maar alleen in een programma. Maar bekijkt U eerst eens de omschrijving van dit commando:

Probleem:	Invoeren van data via het toetsenbord
Commando:	INPUT string,v1
Parameter:	string – Willekeurige tekens v1 – Variabele waarin de invoer wordt opgeslagen. INPUT "GETAL",X
Voorbeeld:	Hier wordt de string "GETAL" opgebouwd waarna er een numerieke waarde via het toetsenbord wordt ingelezen die dan de naam X krijgt.

Opmerking:	Dit commando kan niet in direct mode gebruikt worden.

Geef nu eerst eens het commando 'NEW' waardoor het eventueel aanwezige programma uit het geheugen gewist wordt. De volgende BASIC-regels laten de werking van de INPUT opdracht zien:

```
10 PRINT"CIRKEL BEREKENING"
20 PRINT"----------"
30 INPUT"DIAMETER ";DIAM
40 PRINT"OPPERVLAKTE:";(DIAM/2)+2*P1
```

Laten we dit programma eens met RUN starten. In regel 30 wordt er dan naar de diameter van de cirkel gevraagd. Hierbij moet een invoer altijd met ENTER worden afgesloten. Achter de string moet bij het commando INPUT altijd een punt-komma staan. De string kan echter ook wegggelaten worden, zoals men in het volgende programma kan zien:

```
10 PRINT"CIRKEL BEREKENING"
20 PRINT"----------"
30 INPUT DIAM
40 PRINT"OPPERVLAKTE:";(DIAM/2)+2*P1
```

In een dergelijk geval hoeft U natuurlijk niet het gehele programma opnieuw in te tikken, maar moet U alleen regel 30 veranderen. Het zal U opvallen dat de string die voor de INPUT staat alleen maar een mededeling bevat. Deze mededeling kan ook in de regel voor de INPUT staan:

```
10 PRINT"CIRKEL BEREKENING"
20 PRINT"----------"
25 PRINT"DIAMETER";
30 INPUT DIAM
40 PRINT"OPPERVLAKTE:";(DIAM/2)+2*PI
```

Voeg regel 25 eens toe aan het bestaande programma en start dit nog eens. U zult dan zien dat het precies hetzelfde loopt als het programma ervoor. Belangrijk is de punt-komma achter de PRINT opdracht in regel 25. Als U dit weglaat wordt de INPUT opdracht op de volgende regel uitgevoerd. Probeer dat maar eens.

Ook strings kunnen via het toetsenbord worden ingevoerd. Ook dit gebeurt met het commando INPUT. Ook hier eerst een voorbeeld. Wis eerst het oude programma met NEW.

```
10 PRINT''HOE HEET U?''
20 INPUT NAAM$
30 PRINT''GOEDEN DAG '';NAAM$;'', HOE GAAT HET MET U?''
```

Regels 10 en 20 kan men tot een regel samenvoegen. De string uit regel 10 komt dan voor de INPUT in regel 20 te staan:

```
10: INPUT''HOE HEET U?'';NAAM$
20 PRINT''GOEDEN DAG '';NAAM$'', HOE GAAT HET MET U?''
```

Deze voorbeelden demonstreren alles wat het commando INPUT te bieden heeft. Het programma zal ook doorlopen als er geen invoer gedaan wordt, maar als er alleen op de toets ENTER wordt gedrukt. De variabelen krijgen dan de waarde 0 of bij een string de 'waarde' '''' (de lege string).

LUSSEN

Laten we eerst maar eens een oefening maken waarmee we dan de besturingen van de lussen kunnen bekijken. Via het toetsenbord willen we vijf namen inlezen. Deze namen komen in de array NAAM$ te staan. Met de tot nu toe beschreven opdrachten kunnen we het volgende programma opstellen:

```
10 PRINT''INVOEREN VAN NAMEN''
20 PRINT''----------''
30 INPUT''NAAM '';NAAM$(1)
40 INPUT''NAAM '';NAAM$(2)
50 INPUT''NAAM '';NAAM£(3)
60 INPUT''NAAM '';NAAM$(4)
70 INPUT''NAAM '';NAAM$(5)
```

Met dit programma worden nu achter elkaar de vijf namen ingevoerd. Maar vindt U dit niet erg omslachtig? Stelt U zich eens voor dat er 100 namen moeten worden ingelezen!

In de praktijk zal men deze manier van programmeren dan ook nooit tegenkomen. Men neemt hiervoor een 'lus'. Om U te laten zien hoe een lus wordt opgebouwd, kijken we eerst naar de volgende omschrijving:

Probleem:	Besturing van een lus.
Commando:	FOR v1=n1 TO n2 STEP n3.

Parameter:	v1 – numerieke variabele. n1 – beginwaarde van v1. n2 – eindwaarde van v1. n3 – stapgrootte, waarde waarmee v1 verhoogd wordt na elke lusdoorgang.
Voorbeeld:	FOR A=1 TO 20 STEP 1 De waarde van de variabele A wordt na elke lusdoorgang met 1 verhoogd totdat de eindwaarde 20 bereikt is (20 lusdoorgangen).
Opmerking:	Als de stapgrootte 1 is kan men STEP weglaten. De opdracht in dit voorbeeld kan dan ook zijn: FOR A=1 TO 20

Een lus begint met de aanwijzing FOR. Alle hierna staande opdrachten behoren ook tot de lus en worden achtereenvolgens verschillende keren uitgevoerd. Maar hoe wordt dan het einde van die lus kenbaar gemaakt? Hiervoor bestaat er een andere opdracht, waarvan de taak bestaat uit: onderzoek of het einde van de lus bereikt is. Als de eindwaarde van de lus nog niet bereikt is, wordt de lusvariabele met de waarde STEP verhoogd en wordt de lus nogmaals doorlopen. Als de lus echter de eindwaarde heeft bereikt, wordt het programma achter het commando NEXT vervolgd.

Probleem:	Bepaal einde van de lus.
Commando:	NEXT v1.
Parameter:	v1 – lusvariabele.
Voorbeeld:	NEXT A Geeft het einde aan van de lus uit ons vorige voorbeeld.
Opmerking:	Als de variabele v1 niet wordt meegedeeld, dan heeft de opdracht NEXT betrekking op de laatst ingerichte lus.

Met behulp van deze twee opdrachten kunnen we nu willekeurige lussen programmeren. We gaan weer terug naar onze invoer van data zoals hiervoor werd begonnen. Met behulp van het commando INPUT moesten we vijf namen in een array met de index 1 tot 5 inlezen. Als we nu een lus gebruiken dan is er maar één INPUT opdracht nodig, waarbij de index optreedt als variabele, en wel als lusvariabele. De lus moet van 1 tot 5 lopen. Hoe ziet dan de bijbehorende FOR-opdracht er uit met de paramaters?

```
FOR I=1 TO 5
```

Het veranderde programma ziet er dan als volgt uit:

```
10 PRINT"INVOEREN VAN NAMEN"
20 PRINT"----------"
30 FOR I=1 TO 5
40 INPUT"NAAM";NAAM$(I)
50 NEXT I
```

Dit is toch wel een zeer elegante manier om 5 strings in te lezen nietwaar? Interessant is ook het aflopen van het programma na het starten. Allereerst worden de twee PRINT opdrachten uitgevoerd. Nu begint de lus met de variabele I, de begin waarde 1, de eindwaarde 5 en de stapgrootte 1. Herinnert U het zich nog; als er geen stapgrootte wordt meegedeeld dan is de stapgrootte automatisch 1. Bij het begin van de lus heeft I de waarde 1. In regel 40 krijgt dus de array variabele de waarde 1. In regel 50 wordt onderzocht of de lusvariabele al de eindwaarde 5 heeft bereikt. Als dat niet het geval is, wordt de variabele met de waarde 1 verhoogd en wordt het programma achter de opdracht FOR (dat is dus in regel 40) voortgezet. De variabele I heeft nu de waarde 2. De daarop volgende INPUT leest nu de naam in waarvan de index nu 2 is. En zo gaat het verder tot de opdracht NEXT de lus afsluit. Als de variabele nu de waarde 4 heeft, wordt dit nog eens met de waarde 1 verhoogd en wordt de lus met de waarde I=5 doorlopen. Aansluitend hierop wordt deze waarde door NEXT onderzocht. Daar dit nu het geval is, wordt het programma achter de opdracht NEXT vervolgd, dus beëindigd.

We kunnen het programma nog verder uitbouwen door de ingevoerde namen nu af te laten drukken. Daarvoor is er ook een lus nodig:

```
60 FOR I=1 TO 5
70 PRINT"NAAM ";NAAM$(1)
80 NEXT I
```

Het is allemaal niet zo gecompliceerd als het er in het begin uitzag.

Wat kan men allemaal nog meer met een lus doen? Men kan bijvoorbeeld een wachtlus inbouwen, die het aflopen van het programma wat vertraagt. Hiervoor bouwen we een lus met FOR op en sluiten deze lus weer onmiddellijk met NEXT af. Nu is het belangrijk om te weten hoe vaak deze lus moet worden doorlopen, om bijvoorbeeld een wachttijd van 1 seconde te hebben. Men kan ervan uitgaan dat een lus ca. 1000 keer per seconde wordt doorlopen. Een wachttijd van 3 seconden zal er dan als volgt uitzien:

```
FOR X=1 TO 3000:NEXT
```

We herinneren U er aan, dat op één regel meer dan één opdracht kan staan als deze door een dubbele punt wordt gescheiden. Laten we nu in ons programma eens een wachtlus, tussen het invoeren van de data en het afdrukken op het beeldscherm, opnemen. Als regel die daar tussen zit kunnen we 55 nemen. Hoe ziet deze regel er uit als de wachttijd ca. 2 seconden moet zijn?

```
55 FOR X=1 TO 2000:NEXT X
```

Als U nu met LIST een uitvoer van het programma op het beeldscherm krijgt, zult U zien dat de regels in de normale volgorde staan. Als U dan nu het programma start, dan zult U de uitwerking van regel 55 zelf waarnemen.

Tot nu toe hebben we alleen met de stapgrootte 1 gewerkt, maar daarvoor kunnen we ook elke andere waarde nemen. Als de namen uit het vorige programma niet van 1 naar 5 moeten worden ingevoerd, maar omgekeerd van 5 naar 1, dan is dat ook mogelijk. De beginwaarde van deze lus is dan 5, de eindwaarde is dan 1 en de stapgrootte -1. Hoe moeten nu de regels 30 en 60 veranderd worden? U heeft het antwoord zeker al lang klaar:

```
30 FOR I=5 TO 1 STEP -1
60 FOR I=5 TO 1 STEP -1
```

Laten we nu nog een paar oefeningen met de opdracht FOR maken. Hier moeten de FOR-opdrachten van de volgende lussen worden samengesteld:

	Lus-variabele	Begin-waarde	Eind-waarde	Stap-grootte
A).	INDEX	10	18	1
B).	GETAL	30	15.5	−0.5
C).	LUS	590	1800	0.25
D).	TIJD	1	10	0.3

De oplossingen zijn:

A). FOR INDEX=10 TO 18 STEP 1
of:
FOR INDEX=10 TO 18
B). FOR GETAL=30 TO 15.5 STEP -0.5
C). FOR LUS=590 TO 1800 STEP 0.25
D). FOR TIJD=1 TO 10 STEP 0.3

Tot slot van deze paragraaf nog een opmerking. Als U met FOR een lus opbouwt die logisch fout is (bv. FOR X=10 TO 0), dan wordt deze lus toch een keer doorlopen!

EERSTE REAKTIE VAN EEN PROGRAMMA

Nu we zover in BASIC doorgedrongen zijn, wordt het tijd om met het adressenprogramma te beginnen. Hoe moet men een dergelijk programma opbouwen? Als U al eerder met een dergelijk programma gewerkt heeft, weet U al wat zo'n programma op het beeldscherm laat zien. Het meldt zich meestal met een opschrift. Dit opschrift dat slechts uit een paar PRINT-opdrachten bestaat, zullen we eerst samenstellen. Commando's die U nog volledig onbekend zijn, moeten U niet hinderen om het programma in te tikken en te starten. Deze commando's worden later nog omschreven.

```
100 REM ==========
110 REM OPSCHRIFT VAN HET PROGRAMMA
120 REM ==========
130 CLS
140 PRINT STRING$(40,"=")
150 LOCATE 12,2
160 PRINT "ADRESSEN BESTAND"
170 PRINT STRING$(40,"=")
```

Nadat U dit deel van het programma hebt ingetikt en gestart, wilt U natuurlijk ook weten wat de nieuwe opdrachten betekenen. Beginnen we bij de regels 100 tot 120. Hier staat de dokumentatie van het programma. Om de computer deze regels niet te laten verwerken worden zij met REM aangeduid. REM betekent REMARKs, dus 'opmerkingen'. Zulke REM-regels kunnen in elk deel van een programma worden ingevoegd. Door deze commentaren worden de programma's duidelijker.

Het volgende nieuwe commando is eigenlijk geen commando maar een funktie, iets nauwkeuriger, het is een string-funktie. Zoals U weet, noemt men een keten van tekens in de vaktaal een 'string'. Met de funktie STRING$ kan men een teken van 1 tot 255 keer herhalen. Laten we ook hier eerst de omschrijving bekijken:

Probleem:	Produceren van een string
Funktie:	STRING$(n,"Z").
Parameter:	n – aantal tekens (0-255) "Z" – het teken dat verveelvoudigd moet worden.

Voorbeeld:	PRINT STRING$(80,":") hierdoor worden 80 dubbele punten op het beeldscherm gezet.

Deze funktie kunt U alleen maar in samenwerking met de PRINT opdracht proberen. Daar deze opdracht nogal eenvoudig is kunnen we direkt met de volgende verder gaan.

De opdracht PRINT zet de gegevens op het beeldscherm. Op welke plaats dit op het beldscherm komt te staan, hangt af van de laatste PRINT opdracht of van CLS. Om deze PRINT opdracht op een bepaalde plaats op het beeldscherm te krijgen, bestaat er in BASIC de volgende mogelijkheid:

Probleem:	Positie van de cursor
Commando:	LOCATE k,r
Parameter:	k – kolom r – regel
Voorbeeld:	LOCATE 5,10 zet de cursor op de 5-de kolom van de 10-de regel

Na elke PRINT opdracht wordt de positie van de cursor veranderd. Deze cursor kan men echter niet zien omdat de computer met het programma bezig is. Als men de positie van de cursor echter zelf wil bepalen dan moet men voor de PRINT opdracht eenvoudig de opdracht LOCATE zetten. Met deze opdracht geeft men, zoals hiervoor is omschreven, met de kolom en de regel de positie aan. Een voorbeeld zal dit verduidelijken: de tekst "HALLO" zal in het midden van het beeldscherm moeten komen te staan. Daarvoor hebben we de opdrachten LOCATE en PRINT nodig. De volgorde waarin dit moet worden gebruikt is:

LOCATE 18,13:PRINT"HALLO"

U herinnert zich toch nog wel dat er meer dan een opdracht op een regel mag staan, als U er maar aan denkt deze door een dubbele punt te scheiden.

Maar nu weer terug naar ons programma.

SUBROUTINES

In het verdere verloop van het adressenbestand zullen we de BASIC-regels voor het opschrift van het programma nog vaker nodig hebben. Om deze regels niet steeds opnieuw in te voeren, waardoor ook het programma onnodig langer wordt, maken we hiervoor een onder-programma (ook wel subroutine genoemd). Maar hoe wordt een dergelijke subroutine opgebouwd en hoe wordt dit uitgevoerd. Dat zijn twee vragen die, zoals altijd, eerst verklaard worden.

In de vorige paragraaf hebben we het commando GOTO leren kennen. Hiermee is het mogelijk om het programma op een andere plaats voort te zetten. Omdat na het beëindigen van een onderprogramma echter weer naar het vertrekpunt terug moet worden gegaan, en de computer bij de opdracht GOTO niet weet waar de sprong vandaan kwam, is de opdracht GOTO hier niet op zijn plaats. Voor het aanroepen van een onderprogramma is er een vergelijkbare opdracht, die we eerst willen omschrijven:

Probleem:	Aanroepen van een subroutine.
Commando:	GOSUB rn.
Parameter:	rn – regelnummer waar de subroutine begint.
Voorbeeld:	GOSUB 100 roept het onderprogramma op regel 100 aan.
Opmerking:	De met GOSUB aangeroepen subroutines moeten met de opdracht RETURN worden afgesloten. Komt het aflopen van het programma op een RETURN, zonder dat er een GOSUB aan vooraf ging, dan ontstaat de foutmelding: "UNEXPECTED RETURN"

Dit ziet er ook eenvoudig uit. Dat is het ook, vooral als men er een paar keer mee gewerkt heeft. We veranderen nu regels 100-170 in een subroutine, dat vanaf een willekeurige plaats uit het programma kan worden opgeroepen. Zoals uit de omschrijving van dit commando is gebleken, moet deze subroutine altijd met RETURN worden afgesloten. Daarom voegen we er nog de volgende regel aan toe:

```
180 RETURN
```

Nu kunnen we het programma echter niet meer starten. Doet U het toch, dan ontstaat de foutmelding "UNEXPECTED RETURN IN 180". Ook dit is duidelijk omdat we met RUN eenvoudig in dit onderprogramma zijn terecht gekomen, dat echter alleen met GOSUB

mag worden opgeroepen. Hoe gaan we dan nu verder?

Bekijken we eerst eens hoe de organisatie van de regelnummers van het toekomstige programma er uit zal zien:

0-99:	Hier willen we alle informatie voor het programma onderbrengen. Bijvoorbeeld: Inrichten van tabellen, toewijzen van variabelen enz.
100-999	In deze regels worden alle routines ondergebracht.
1000-	Vanaf regel 1000 begint het eigenlijke programma, het hoofdprogramma.

We zullen dus over de subroutines heen moeten springen om in het hoofdprogramma terecht te komen. Daarvoor zetten we in regel 99:

```
99 GOTO 1000
```

In regel 1000 begint dan het hoofdprogramma. Dit wordt duidelijk gemaakt met een aantal REM regels, die gevolgd worden door GOSUB 1000:

```
1000 REM ==========
1010 REM HOOFDPROGRAMMA
1020 REM ==========
1030 GOSUB 100
```

En nu loopt het programma weer. We maken nu eerst de subroutine voor het afdrukken van het opschrift.

HET MENU

Laten we eens op een rijtje zetten welke vorderingen we met ons programma gemaakt hebben:

```
99 GOTO 1000
100 REM ==========
110 REM OPSCHRIFT VAN HET PROGRAMMA
120 REM ==========
130 CLS
1240 PRINT STRING$(40,,"=")
```

```
150 LOCATE 12,2
160 PRINT''ADRESSENBESTAND''
170 PRINT STRING$(40,''='')
180 RETURN
1000 REM ==========
1010 REM HOOFDPROGRAMMA
1020 REM ==========
1030 GOSUB 100
```

Het programma meldt zich na het starten met zijn naam. Maar dit is zeker niet alles, het is slechts een begin. Omdat we met dit programma meerdere mogelijkheden willen hebben, moeten we deze ook aanwijzen. Denk eens na wat het programma allemaal moet kunnen. Allereerst moeten de gegevens op een extern geheugen kunnen worden opgeslagen. Men moet deze gegevens ook weer terug kunnen lezen. Dus moeten we hiervoor over de volgende twee keuzemogelijkheden beschikken:

-1- LEZEN VAN GEGEVENS
-2- OPSLAAN VAN GEGEVENS

Verder moeten we natuurlijk over de mogelijkheid beschikken adressen in te voeren. Daarvoor hebben we funktie 3 nodig:

-3- ADRESSEN INVOEREN

Het moet ook mogelijk zijn om deze adressen te veranderen. Punt 4 moet dit mogelijk maken:

-4- VERANDEREN VAN ADRESSEN

Als we van 'OME PIET' vanaf nu niets meer willen weten, moet hij uit ons adressenbestand verdwijnen. Daarvoor hebben we de funktie 'wis' nodig:

-5- ADRES WISSEN

En nu komt het belangrijkste: al deze gegevens leveren ons niets op als we er niets mee kunnen doen. Met het volgende programmadeel kunnen we de adressen op het beeldscherm afdrukken:

-6- ADRESSEN AFDRUKKEN

Omdat we het programma niet met de schakelaar willen beëindigen, komt nu het laatste deel van het programma:

-7- BEËINDIG HET PROGRAMMA

Als we alle programmastappen goed hebben doordacht breiden we ons programma zodanig uit dat al deze funkties als een menu op het beeldscherm verschijnt. Daarom voegen we de volgende regels aan ons programma toe:

```
1040 LOCATE 1,7
1050 PRINT" PROGRAMMA FUNKTIES "
1060 PRINT" ---------- "
1070 PRINT
1080 PRINT"-1- Lezen van gegevens"
1090 PRINT"-2- Opslaan van gegevens"
1100 PRINT"-3- Adressen invoeren"
1110 PRINT"-4- Veranderen van adressen"
1120 PRINT"-5- Adres wissen"
1130 PRINT"-6- Adres afdrukken"
1140 PRINT"-7- Beëindig het programma"
```

Als U het programma nu nog een keer start zult U zien dat het langzaamaan een professioneel karakter krijgt. Nu moet de gebruiker van het programma beslissen welke programma-funktie hij nodig heeft. We hebben daarvoor een numerieke variabele nodig die de waarde 1 tot 7 mag hebben, zoals het bij het menu staat aangegeven. Voordien wordt er nog een PRINT opdracht gegeven met de inhoud 'KEUZE':

```
1150 LOCATE 10,18:PRINT"KEUZE";
1160 INPUT funktie
```

DE VRAAGSTELLING MET IF

Wat echter als de gebruiker een getal kiest dat ongeldig is (bijvoorbeeld 9)? Dit moet in het programma voorkomen worden. De vraag is alleen, hoe? We moeten bepalen welke waarde er in ingevoerd mag worden en dan deze waarde op de geldigheid onderzoeken. We omschrijven eerst de opdracht hiervoor:

Probleem:	Voorwaarde onderzoek
Commando:	IF voorwaarde THEN aanwijzing.

Parameter:	voorwaarde – Vergelijkingsoperatie bv. A=10 of B=12 aanwijzing – Opdracht die uitgevoerd moet worden als de voorwaarde WAAR is.
Voorbeeld:	IF F=0 THEN GOTO 100 Als de variabele de waarde 0 heeft wordt het programma bij regel 100 voortgezet, indien dit niet zo is gaat het programma met de volgende regel verder.
Opmerking:	Als achter THEN een GOTO staat dan kan men dit ook weglaten. In het laatste voorbeeld mag men ook doen: IF F=0 THEN 100.

Een voorwaarde wordt niet alleen met de vergelijkingsoperator '=' vergeleken, maar ook met de operatoren '<' (kleiner dan) en '>' (groter dan). Ook combinaties van deze drie zijn toegestaan. Alles bij elkaar kan men de volgende vergelijkingsoperatoren inzetten:

Symbool	Betekenis
=	gelijk aan
<	kleiner dan
>	groter dan
<=,=<	kleiner dan of gelijk aan
>=,=>	groter dan of gelijk aan
<>,><	ongelijk aan

De opdracht IF onderzoekt dus eerst of aan de voorwaarde is voldaan. Als dit het geval is, dan wordt de opdracht achter THEN uitgevoerd. Als aan de voorwaarde niet wordt voldaan, gaat het programma verder op de regel die na IF staat.

Dit is echter niet alles wat men met IF kan doen. Er kunnen meerdere voorwaarden op een logische manier met elkaar verbonden worden. Dit lijkt gecompliceerder dan het in werkelijkheid is. Laten we ook hier eerst eens een voorbeeld ter hand nemen:

IF ANTW=1 OR ANTW2 THEN 100

Twee vergelijkingen die met elkaar verbonden zijn. OR staat voor 'of' en daarmee kan deze opdracht als volgt geïnterpreteerd worden: als ANTW gelijk is aan 1 of ANTW is groter dan 12, ga dan naar regel 100. Voor de sprong naar regel 100 moet dus of ANTW gelijk zijn aan 1 of ANTW moet groter zijn dan 12 of beide. Alleen als beide mogelijkheden niet waar zijn, wordt het programma op de regel, die achter de IF regel staat, vervolgd. De

OR verbinding is slechts een van de mogelijkheden. Een ander voorbeeld is:

IF ANTW=1 AND TEST2 THEN 100

Dit is een 'EN' verbinding, die slechts waar is als aan beide voorwaarden wordt voldaan, dus al zowel ANTW gelijk is aan 1 EN TEST groter is dan 12.

We stellen ons de vraag of de gebruiker een waarde heeft ingevoerd die buiten het gebied van 1 tot 7 ligt. Hoe moeten we nu de bijbehorende IF voorwaarde opstellen? Laten we de oplossing hiervan eens bekijken:

IF funktie=0 OR funktie7 THEN...

JA, wat dan? We geven dan eenvoudig een foutmelding en gaan weer terug naar regel 1000. Voor het afdrukken van deze foutmelding hebben we nog een subroutine nodig. Daarin komt de soort fout te staan die dan als string in regel 240 wordt afgedrukt. Laten we eerst eens dit programma bekijken:

```
200 REM ==========
210 REM FOUTMELDING
220 REM ==========
230 LOCATE 1,25
240 PRINT fout$
250 PRINT CHR$(7);
260 FOR 1=1 TO 1000:NEXT
270 LOCATE 1,25
280 PRINT STRING$(39," ")
290 RETURN
```

Allereerst wordt de cursor op de laatste regel gezet. Regel 240 geeft dan de string 'fout$', die door het programma wordt opgeroepen. Voordat men deze routine in het hoofdprogramma aan kan roepen, moet de string 'fout$' in de verzameling strings worden opgenomen.

ASCII-CODE

Nu naar de funktie in regel 250. U herinnert zich nog dat we de opdracht ENTER met CHR$(13) uit konden voeren, zoals dat in het hoofdstuk over de funktie-toetsen werd meegedeeld. We bekijken deze funktie nu eens wat nauwkeuriger. Elk teken wordt door de computer in een code gezet. Het krijgt een waarde tussen 0 en 255. Deze codering is genormeerd volgens de amerikaanse ASCII-norm. De code van de CPC464 wijkt slechts

weinig af van deze norm. Het zijn niet alleen de tekens die een code hebben. Ook speciale funkties, zoals in ons programma het oproepen van een toon, kan men met een ASCII-code oproepen. Om deze tekens naar de ASCII norm te coderen, zijn er twee funkties, die we hierna beschrijven:

Probleem:	Verandering van de ASCII-code in een teken.
Funktie:	CHR$(n).
Parameter:	n – ASCII-code (0-255).
Voorbeeld:	FOR I=32 TO 255:PRINT CHR$(I):NEKT.
Opmerking:	Wordt een ASCII-code gevraagd van een waarde die groter is dan 255 dan volgt er een foutmelding 'IMPROPER ARGUMENT'
Probleem:	Verandering van een teken in de ASCII-code.
Funktie:	ASC("Z"). "Z" – een teken, of een stringvariabel waar een teken in staat.
Parameter:	PRINT ASC("A") Laat van de letter A de ASCII-code op het beeldscherm zien ("A"=65).
Voorbeeld:	– Als de stringvariabele leeg is ontstaat de foutmelding "IMPROPER ARGUMENT"
Opmerking:	– Is de stringvariabele langer dan een teken, dan wordt de ASCII-code van het eerste teken meegedeeld.

Dat is een heleboel informatie, die we eerst eens moeten verteren. Zoals gezegd, de computer werkt intern alleen met de ASCII-code van een teken. Daarom begrijpt het apparaat alleen de ASCII-code en niet het teken zelf.

Als U weten wilt wat de ASCII-code van een bepaald teken is, dan kunt U dat met de funktie ASC("Z") doen. In de direct-mode moet U dan het commando PRINT ASC("Z") geven – voor Z kunt U dan het gewenste teken geven. De CPC464 antwoordt dan met de bijbehorende code.

In regel 240 werd met de opdracht 'PRINT CHR$(7)' een toon opgewekt, waarmee de fout ook door het gehoor werd opgemerkt. Overigens is ook deze code door ASCII genormeerd en bij de meeste computers te vinden.

Maar nu verder met onze fouten-routine. Na de toon onstaat er een wachtlus die ongeveer 1 seconde duurt, lang genoeg om het bericht op het beeldscherm te lezen. Daarop wordt de foutmelding in regel 280 weer gewist, eenvoudig door er een regel spaties overheen te schrijven. In regel 290 staat dan de terugsprong naar het hoofdprogramma.

We kunnen nu de IF-vraag achter de INPUT voltooien:

```
1170 IF funktie=0 OR funktie>7 THEN fout$="ONGELDIGE WAARDE":GOSUB
200:GOTO 1150
```

Tikt u nu eens deze regel in. Hierbij wordt dan opgemerkt dat de indeling van de twee regels zoals die hier boven staan, niet dezelfde is als de indeling op Uw beeldscherm. Als het getal dat nu ingevoerd wordt 0 of groter dan 7 is, ontstaat de foute bediening van de gebruiker, en wordt de routine in regel 200 aangeroepen. Allereerst wordt de foutmelding omgezet in een string, dan wordt de fout-routine aangeroepen, en tenslotte vraagt het programma in regel 1150 weer een nieuwe INPUT.

Start nu weer het programma en geef een waarde die niet toegestaan is. Het programma merkt deze foute invoer en geeft de foutmelding.

BEREKEND GOTO

Als er nu een geldige waarde wordt ingevoerd, moeten we de nodige stappen ondernemen. Dat wil zeggen we moeten nu naar de bijbehorende programma-regels springen.

Om, afhankelijk van de waarde van de variabele 'funktie', naar het betreffende programma-onderdeel te springen, bestaat er een opdracht, waarmee dit op de meest eenvoudige manier gedaan kan worden. Bekijk de omschrijving van dit commando eens:

Probleem:	Berekende sprong.
Commando:	ON v1 GOTO n1, n2, n3...
Parameter:	v1 – numerieke variabele n1, n2, n3... – regelnummers, waarheen afhankelijk van de waarde van v1 heen gesprongen wordt.

Voorbeeld:	ON F GOTO 100,200,300,400 Als F=1 dan wordt naar regel 100 gesprongen Als F=2 dan wordt naar regel 200 gesprongen enz. Als F4 dan wordt er niet gesprongen maar gaat het programma op de volgende regel verder.
Opmerking:	Als F geen geheel getal is (bv. 3.45) dan wordt het getal voor de komma gebruikt

Een ideaal commando voor ons doel. We bepalen eerst bij welke regelnummers onze programmadelen moeten beginnen. Onze volgende opdracht moet er dan als volgt uitzien:

```
1180 ON funktie GOTO 5000,10000,15000,
20000,25000,30000,35000
```

In welke regels nu de diverse programmadelen moeten worden ondergebracht, kunnen we in het volgende overzicht zien:

funktie

1	5000	GEGEVENS LEZEN
2	10000	GEGEVENS OPSLAAN
3	15000	ADRESSEN INVOEREN
4	20000	ADRESSEN VERANDEREN
5	25000	ADRESSEN WISSEN
6	30000	ADRESSEN AFDRUKKEN
7	35000	PROGRAMMA EINDE

We moeten nu het ene programmadeel na het andere ontwerpen. Zinvol beginnen we met funktie 3, het invoeren van adressen. Maar eerst moeten we nog de tabel voor deze adressen dimensioneren. Daarom voegen we regel 10 toe:

```
10 DIM adres$(200,7):REM ADRES ARRAY
```

We hebben nu plaats gereserveerd voor 200 adressen. Dat is in het algemeen meer dan voldoende. Wie heeft er meer dan 200 kennissen? Voor commerciële doeleinden is dit programma slechts beperkt te gebruiken. Een firma die meer dan 2000 klanten op deze manier in het bestand op wil nemen moet naar een ander geschikt programma uitkijken.

Laten we nu voor elk van deze programmadelen een routine schrijven, waarbij we bij elk programma met een mooi opschrift beginnen. Bij elke routine moet vooraf ook dit op-

schrift op het beeldscherm verschijnen. Elk deel van dit programma heeft een naam die ook door het menu wordt aangegeven. Deze namen slaan we in een tabel op, waarvan de index dezelfde waarde heeft als de variabele 'funktie'. Wat bedoelen we hier nu mee? In het overzicht op de vorige bladzijde kunt U zien, dat bijvoorbeeld het programmaonderdeel 'ADRESSEN INVOEREN' nummer 3 heeft. Dit nummer wordt dan de index van de plaats van de tabel. De tabel krijgt de naam 'funktie$', waardoor de variabele funktie$(3) de tekst 'ADRESSEN INVOEREN' krijgt. Het inrichten van deze tabel behoort tot de voorbereidingen en komt daarom in de regels 0-99 te staan. De volgende regels vormen deze tabel:

```
20 funktie$(1)="GEGEVENS LEZEN"
21 funktie$(2)="GEGEVENS OPSLAAN"
22 funktie$(3)="ADRESSEN INVOEREN"
23 funktie$(4)="ADRESSEN VERANDEREN"
24 funktie$(5)="ADRESSSEN WISSEN"
25 funktie$(6)="ADRESSEN AFDRUKKEN"
26 funktie$(7)="PROGRAMMA EINDE"
```

Let erop dat de strings allemaal dezelfde lengte hebben (tweede paar aanhalingstekens onder elkaar). In de subroutine vanaf regel 300 willen we nu het opschrift van het programmaonderdeel en de door de gebruiker gekozen funktie laten zien:

```
300 REM ==========
310 REM OPSCHRIFT VAN HET PROGRAMMA
320 REM ==========
330 GOSUB 100
340 LOCATE 10,5:PRINT STRING$(19,"*")
350 LOCATE 10,6:PRINT"*"+funktie$(funktie)+"*"
360 LOCATE 10,7:PRINT STRING$(19,"*")
370 RETURN
```

Hier ziet U dat ook een subroutine een andere subroutine kan aanroepen. Eerst wordt het algemene programma opschrift afgedrukt. Daarna de eerste regel van het 'kastje' waarin de naam van de funktie komt te staan. De daarop volgende regel ziet er gecompliceerder uit dan hij in werkelijkheid is. Allereerst komt de cursor op de 10-de kolom van de 6-de regel te staan. Daar komt allereerst een sterretje te staan gevolgd door de naam van de funktie. Deze naam komt uit de tabel 'funktie$' met de index 'funktie'. Achter deze string komt dan weer een sterretje te staan. In regel 360 wordt tenslotte het 'kastje' afgemaakt. De laatste regel behoeft geen toelichting meer.

Maar wanneer wordt deze routine opgeroepen? Het is logisch dat dit pas gebeuren kan, als de gebruiker al een keuze voor het menu gemaakt heeft. Dus voorbij de IF – vraag. We

gebruiken hiervoor regel 1175:

```
1175 GOSUB 300
```

We moeten deze regel wel nemen omdat regel 1180 al een andere bestemming heeft.

INVOEREN VAN ADRESSEN

En nu wordt het ernst. Het eerste belangrijke programmadeel moet gemaakt worden. Laten we met het meest eenvoudige beginnen:

```
15000 REM ==========
15010 REM ADRESSEN INVOEREN
15020 REM ==========
```

Tot zover gaat alles goed. Maar nu wordt het gecompliceerd (niet de moed verliezen). We moeten op de een of andere manier weten hoeveel adressen er zijn. Daartoe verhogen we in dit programma onderdeel een 'wijzer' met de waarde 1, waardoor we het aantal adressen weten. Noemen we deze variabele eenvoudig 'wijzer'. De volgende regel zal deze wijzer steeds met het getal 1 verhogen:

```
15030 wijzer=wijzer+1
```

Als bv. deze wijzer ongelijk is aan 0, dan zijn er al gegevens ingelezen. Hiermee wordt het adressenbestand automatisch verhoogd.

Nu moeten we 7 INPUT-regels geven om de 7 data-velden van een adres in te lezen. Maar dit gaat eenvoudiger. Allereerst zetten we vanaf regel 30 alle veldnamen in een tabel (array) vanaf 'veld$(1)' tot 'veld$(7)'.

```
30 veld$(1)="TITEL"
31 veld$(2)="VOORNAAM"
32 veld$(3)="NAAM"
33 veld$(4)="STRAAT"
34 veld$(5)="PLAATS"
35 veld$(6)="TELEFOON"
36 veld$(7)="OPMERKING"
```

De namen van deze array's komen aan het begin van het programma te staan en kunnen gedurende het gehele programma ingezet worden. Waarom we ze nu nodig hebben, zult U in het vervolg van dit programma zien:

```
15040 PRINT
15050 FOR index=1 TO 7
15060 PRINT veld$(index);
15070 INPUT adres$(wijzer, index)
15080 NEXT index
```

Start U nu eens dit programma, nadat U deze regels hebt toegevoegd, en kies dan funktie 3. Wat er nu allemaal gebeurt, wordt door deze lus veroorzaakt. Een hele adresregel wordt met deze vier opdrachten ingelezen. In deze lus wordt eerst de naam van het veld afgedrukt. De variabele 'index' wordt als index voor de array van dit veld gebruikt. Direct achter de naam van dit veld (punt-komma) wordt dan de inhoud van dit veld gelezen. Hier staan twee indices omdat de adressen in een twee dimensionaal veld worden opgeslagen. De eerste index is de variabele 'wijzer' die na het invoeren van een adres met de waarde 1 wordt verhoogd. De tweede index is de variabele 'index'. Er worden dus achter elkaar de 7 velden van een adres via het toetsenbord ingevoerd.

Nadat de velden op deze manier zijn ingelezen moeten we de gebruiker de mogelijkheid geven deze invoer te herhalen. Hij kan ergens een fout gemaakt hebben en moet dit dan kunnen corrigeren.

```
15090 LOCATE 1,18
15110 PRINT"ZIJN DE GEGEVENS JUIST(J/N)";
15120 INPUT antwoord$
15130 IF antwoord$="j" THEN 15150
15140 IF antwoord$="n" THEN wijzer=wijzer-1:GOSUB 300: GOTO 15000
15150 fout$=" 'J' of 'n' indrukken":GOSUB 300:GOTO 15090
```

Als alle gegevens op de juiste manier zijn ingevoerd, wordt het programma voortgezet. Als de gegevens niet juist zijn dan wordt dit programmadeel nog een keer gestart. Eerst wordt echter de teller (wijzer) de waarde met 1 verminderd, omdat anders in regel 15030 de volgende adressen-regel gevraagd wordt. Maar we willen deze laatste regel juist veranderen. Als er geen goed antwoord gegeven wordt dan ontstaat er eerst een foutmelding en daarna wordt de regel opnieuw opgeroepen.

Het programma wordt dan met de volgende vraag vervolgd:

```
15150 LOCATE 1,20
15160 PRINT"VERDERE INVOER (J/N)"
15170 INPUT antwoord$
15180 IF antwoord$="j" THEN GOSUB 300:GOTO 15000
15190 IF antwoord$="n" THEN 1000
15200 fout$=" 'j' of 'n' indrukken":GOSUB 200:GOTO 15150
```

Op deze plaats moet de gebruiker een keuze maken of er nog meer adressen moeten worden ingevoerd of niet. Als 'ja' dan wordt dit programmadeel opnieuw gestart. Als 'nee' dan gaat het programma terug naar het menu. Als er een foute invoer plaatsvindt, wordt er een foutmelding gegeven en gaat het programma terug naar dezelfde vraag.

Hiermee is het eerste programmadeel afgesloten. Men kan nu willekeurig veel adressen invoeren maar men kan er verder nog niets mee doen. Tenslotte nog het totale programmadeel 'ADRESSEN INLEZEN', die U met Uw eigen programma kunt vergelijken:

```
15000 REM ==========
15010 REM ADRESSEN INLEZEN
15020 REM ==========
15030 wijzer=wijzer+1
15040 PRINT
15050 FOR index=1 TO 7
15060 PRINT veld$(index);
15070 INPUT adres$(wijzer,index)
15080 NEXT index
15090 LOCATE 1,18
15100 PRINT"ZIJN DE GEGEVENS JUIST (J/N)((;
15110 INPUT antwoord$
15120 IF antwoord$="j" THEN 15150
15130 IF antwoord$="n" THEN wijzer=wijzer-1:GOSUB 300:GOTO 15000
15140 fout$=" 'j' of 'n' indrukken":GOSUB 200:GOTO 15090
15150 LOCATE 1,20
15160 PRINT"VERDERE INVOER (J/N)";
15170 INPUT antwoord$
15180 IF antwoord$="j" THEN GOSUB 300:GOTO 15000
15190 IF antwoord$="n" THEN 1000
15200 fout$=" 'j' of 'n' indrukken ":GOSUB 200: GOTO 15150
```

ADRESSEN VERANDEREN

Nu willen we de gebruiker van dit programma de mogelijkheid geven, bepaalde velden van een zekere data-regel te veranderen. We gebruiken hiervoor de funktie-toetsen. Met de toets 'v' willen we naar voren, en met de toest 'a' naar achteren 'bladeren'. Als een adres veranderd moet worden, moeten we de toets 'x' indrukken. Met de toets 'ENTER' gaan we weer terug in het menu waarbij de verandering is voltooid.

Dit klinkt allemaal erg aardig maar het moet eerst geprogrammeerd worden. Laten we eens met de eerste regels beginnen:

```
20000 REM ==========
20010 REM ADRESSEN VERANDEREN
20020 REM ==========
```

Deze regels hebben geen commentaar meer nodig. Maar nu wordt het kritiek. Hoe lossen we deze problemen op? We kiezen eerst een variabele waarmee we de eerste index van de tabel vast willen leggen. Deze variabele noemen we 'teller' die bij het begin de waarde 1 krijgt.

```
20030 teller=1
```

Nu kunnen we het eerste adres van ons bestand afdrukken. Dit bestaat, zoals bekend, uit 7 velden, waarvan de namen in een array zijn opgeslagen. We coderen weer een lus waarmee we het nummer van het veld, de naam en de inhoud af kunnen drukken.

```
20040 LOCATE 1,10
20050 FOR index=1 TO 7
20060 PRINT index; veld$(index);adres$(teller, index)
20070 NEXT index
```

Er worden dus steeds 7 velden afgedrukt, waarmee het nummer van dit veld, de betekenis en de inhoud wordt afgegeven. De inhoud van dit veld ontstaat uit de index 'teller' die zoals bekend naar de data regel wijst, en naar de eigenlijke veld index ('index').

Nadat het eerste adres op het beeldscherm verschenen is, moet de gebruiker opnieuw een keuze maken. Hij heeft nu vier mogelijkheden:

Toets 'v'	– vooruit (volgend adres)
Toets 'a'	– achteruit (vorig adres)
Toets 'x'	– veranderen
Toets 'ENTER'	– terug naar menu

Nu moet er aan het toetsenbord worden gevraagd of er een toets is ingedrukt. Het commando daarvoor is INKEY$.

Probleem:	Lezen van een teken van het toetsenbord.
Commando:	s=INKEY$
Parameter:	s – stringvariabele, waarin de ingedrukte toets wordt opgeslagen.

Voorbeeld:	Test of er een toets wordt ingedrukt en slaat dit op in X$. Daarna wordt het programma voortgezet.
Opmerking:	Als er geen toets wordt ingedrukt, wordt de stringvariabele gewist.

Als U het programma alleen voort wilt zetten als er een toets wordt ingedrukt, dan is de volgende regel noodzakelijk (deze regel svp. niet intikken):

```
10 X$=INEKY$ X$=" "THEN 10
```

En zo wordt steeds regel 10 uitgevoerd tot er een toets wordt ingedrukt. Een dergelijke aanwijzing volgt ook in ons programma:

```
20080 toets$=INKEY$:IF toets$=" " THEN 20080
```

Als deze regel bereikt wordt, moet de gebruiker een toets indrukken. We moeten ons nu afvragen of een van de volgende toetsen is ingedrukt: 'v', 'a', 'x' of 'ENTER'. Daarna moet er adequaat worden gereageerd. Allereerst onderzoeken we of de ENTER toets is ingedrukt; deze heeft, zoals bekend is, de ASCII code 13:

```
20090 IF toets$=CHR$(13) THEN 1000
```

We gaan dus terug naar het menu als de toets ENTER is ingedrukt. Als de toets 'v' wordt ingedrukt moeten we de volgende 'bladzijde' (dat is: het volgende adres) hebben. Daarvoor wordt de index 'teller' met de waarde 1 verhoogd, maar alleen dan als de 'teller' de laatste regel nog niet bereikt heeft (teller=wijzer). Met de volgende regel wordt deze opgave uitgevoerd:

```
20100 IF toets$="V" AND teller<wijzer THEN teller=teller+1:GOTO 20040
```

Als nu de toets 'v' wordt ingedrukt EN het laatste adres is nog niet bereikt, dan wordt de teller op het volgende adres gezet en wordt dit op het beeldscherm afgedrukt.

Bij het indrukken van de toets 'a' gaat het op eenzelfde manier, maar nu moet de 'teller' met de waarde 1 worden verminderd, als de 'teller' tenminste inmiddels niet al 1 is, omdat we dan al met het eerste adres te maken hebben.

```
20110 IF toets$="a" AND teller>1 THEN teller=teller-1:GOTO 20040
```

Bij regel 20040 wordt het adres met de index 'teller' afgedrukt. Deze 'teller' werd hier met de waarde 1 verminderd.

Nu moeten we ons nog afvragen of de toets 'x' is ingedrukt. Met deze toets moet er een verandering worden aangebracht.

```
20120 IF toets$="x" THEN 20140
20130 fout$="VERKEERDE INVOER!":SOBUB 200: GOTO 20080
```

Regel 20130 wordt uitgevoerd als de mogelijke waarden van 'teller' niet binnen het toegestane gebied liggen of als er een toets wordt ingedrukt die niet omschreven is. Dan wordt er een foutmelding afgegeven en springt het programma terug naar regel 20080. Op deze plaats wordt het toetsenbord opnieuw onderzocht.

Nu ontbreekt alleen nog maar het echte veranderen van de adressen vanaf regel 20140. Daar wordt eerst naar het nummer gevraagd en vervolgens naar de nieuwe inhoud van het veld dat veranderd moet worden.

```
20140 LOCATE 1,18
20150 INPUT"VELD NUMMER (1-7)";nummer
20160 IF nummer>1 AND nummer <8 THEN 20180
20170 fout$="NUMMER 1-7 INVOEREN":GOSUB 200:GOTO 20140
20180 INPUT"NIEUWE INHOUD:";adres$(teller, nummer)
20190 GOSUB 300:GOTO 20040
```

En dit was dan de rest van dit programmadeel. Allereerst wordt het nummer van het veld dat veranderd moet worden, ingelezen. Daarna wordt getest of 'nummer' een geldige waarde heeft. Als dat het geval is, wordt vanaf regel 20180 de nieuwe inhoud van dit veld ingelezen. Als er een ongeldige waarde wordt verstrekt, dan ontstaat er in regel 20170 een foutmelding. Dit programmadeel kan men alleen met de toets ENTER verlaten. Nu krijgen we nog een volledige LIST van dit programmadeel:

```
20000 REM ==========
20010 REM ADRESSEN VERANDEREN
20020 REM ==========
20030 teller=1
20040 LOCATIE 1,10
20050 FOR index=1 TO 7
20060 PRINT index;veld$(index);adres$(teller,index)
20070 NEXT index
20080 toets$=INKEY$
20090 IF toets$=CHR$(13) THEN 1000
20100 IF toets$="v" AND teller<wijzer THEN teller=teller+1:GOTO 20040
20110 IF toets$="a" AND teller>1 THEN teller=teller-1:GOTO 20040
20120 IF toets$="x" THEN 20140
```

```
20130 fout$="INVOER FOUT":GOSUB 200:GOTO 20180
20140 LOCATE 1,18
20150 INPUT"Veldnummer (1-7)";nummer
20160 IF nummer>1 AND nummer<8 THEN 20180
20170 fout$="Nummer 1-7 invoeren":GOSUB 200:GOTO 20140
20180 INPUT"NIEUWE INHOUD: ";adres$(teller,nummer)
20190 GOSUB 300:GOTO 20040
```

ADRESSEN WISSEN

Bij het wissen passen we een andere techniek toe dan bij het veranderen. U moet immers zo veelzijdig mogelijk leren programmeren. We moeten eerst de voornaam en de achternaam van het adres geven dat gewist moet worden. Is dit niet bekend dan kan men dit met het 'bladeren' uit het deel 'ADRESSEN VERANDEREN' opzoeken.

Maar voor men met dit deel van het programma begint, moet er eerst worden onderzocht of er wel iets in het geheugen van de computer staat. Voor het afdrukken van een foutmelding bouwen we eerst een routine vanaf regel 500 op. Laten we deze volledige routine eens bekijken:

```
500 REM ==========
510 REM GEEN GEGEVENS
520 REM ==========
530 fout$="GEEN GEGEVENS IN DE COMPUTER"
540 GOSUB 200
550 RETURN
```

Deze routine is weliswaar erg kort maar daarom nog niet zinloos. dit onderzoek zouden we ook in het programmadeel 'ADRESSEN VERANDEREN' moeten uitvoeren. Als er geen adressen zijn, kan men ze immers ook niet veranderen! Daarom voegen we de volgende regel aan ons programma toe:

```
20025 IF wijzer=0THEN GOSUB 500:GOTO 1000
```

Hiermee wordt onderzocht of er gegevens in de computer aanwezig zijn. De variabele 'wijzer' bevat immers de laatste data-regel en daarmee het aantal adressen dat in de computer aanwezig is. Als deze variabele de waarde 0 heeft, dan zijn er geen gegevens ingevoerd. We geven dan de foutmelding 'GEEN GEGEVENS IN DE COMPUTER' met behulp van de routine vanaf regel 500 en gaan dan terug naar het menu.

Gaan we nu weer verder met het programmadeel waar we mee bezig waren. De eerste regels zijn, zoals bekend, de eenvoudigste:

```
25000 REM ==========
25010 REM ADRESSSEN WISSEN
25020 REM ==========
25030 IF wijzer=0 THEN GOSUB 500:GOTO 1000
25040 LOCATIE 1,10
25050 INPUT"VOORNAAM: ";voornaam$
25060 INPUT"NAAM: ";naam$
```

Hier onderzoeken we eerst of er gegevens in de computer aanwezig zijn. Als dit zo is dan lezen we de voornaam en de naam van het adres in, dat gewist moet worden. Deze twee gegevens worden in het geheugen onder de naam 'voornaam$' en 'naam$' opgeslagen. Nu moeten we in de adressentabel naar deze namen zoeken. Hierbij kan men geen FOR-...Next lus gebruiken omdat deze zeker voor het beëndigen van de lus zal worden verlaten, namelijk als het adres gevonden is. FOR...Next lussen moet men echter niet voortijdig afbreken. De reden hiervoor is: het adres van het begin van de lus wordt in de computer opgeslagen. Pas als de lus tot een goed einde is gekomen wordt deze geheugenplaats weer gewist. Wordt de lus echter meerdere malen voortijdig afgebroken dan zal dit deel van het geheugen gevuld worden, waardoor het programma kan worden afgebroken.

We bouwen daarom een zelf gestuurde lus op:

```
25070 lus=1
25080 IF adres$(lus,2)=voornaam$ AND adres$(lus,3)=naam$ THEN 25110
25030 IF lus<wijzer THEN lus=lus+1:GOTO 25080
25100 fout$="ADRES IS AFWEZIG !":GOSUB 200:GOTO 1000
```

De beginwaarde van de variabele 'lus' krijgt allereerst de waarde 1. In deze lus wordt onderzocht of het met 'lus' geïndiceerde adres met de gezochte overeenstemt. De tweede index 2 en 3 zijn afkomstig van de 'voornaam' en van de 'naam' die dan met voornaam$ en met naam$ worden vergeleken. Komt de naam overeen dan wordt het programma op regel 25110 vervolgd. In regel 25090 wordt de variabele 'lus' met 1 verhoogd voorzover het einde van de adrestabel nog niet is bereikt.

Als de lus in zijn geheel wordt doorlopen, dus niet door het vinden van een naam wordt afgebroken, dan is deze naam ook niet gevonden. Dit wordt dan meegedeeld en we gaan terug naar het menu.

We moeten nu het programma voortzetten op de plaats waar een naam gevonden is. Dat is bij regel 25110. Hier moet het hele adres nog een keer op het beeldscherm verschijnen,

zodat de gebruiker kan beslissen of dit adres gewist moet worden.

```
25110 GOSUB 300:LOCATE 1,10
24120 FOR index=1 TO 7
24130 PRINT veld$(index);adres$(lus,index)
25140 NEXT index
25150 LOCATE 1,18
25160 INPUT"ADRES WISSEN (j/n)";antwoord$
25170 IF antwoord$="j" THEN 25200
25180 IF antwoord$="n" THEN 1000
25190 fout$=" 'j' of 'n' indrukken":GOSUB 200:GOTO 25150
```

Hier wordt eerst een nieuw beeldscherm opgebouwd en de cursor krijgt een plaats (regel 25110). Daarna wordt met een lus van regel 25120 tot 25140 de inhoud weergegeven van een adres-regel. Nu komt de vraag of er werkelijk gewist moet worden. Als het adres niet gewist moet worden (er bestaan altijd mensen die niet kunnen beslissen), springen we weer terug naar het menu. Als het adres echter wel gewist moet worden, gebeurt dat vanaf regel 25200. Laten we eerst een blik werpen op de laatste regels van dit programmadeel:

```
25200 FOR i1=lus TO wijzer-1
25210 FOR i2-1 TO 7
25220 adres$(i1,i2)=adres$(i1+1,i12)
25230 NEKT i2
25240 NEXT i1
25250 wijzer=wijzer=1
25260 GOTO 1000
```

Om een adres te verwijderen, is het niet genoeg om dit adres te wissen. Zouden we dat doen dan zou het adresbestand ondanks dat wissen toch even groot blijven. Dit was niet de bedoeling van de ontwerper. Alle adressen die achter het adres stonden dat nu verdwenen is, moeten een plaats opschuiven. Als bv. het adres met de index 5 gewist is en er 7 adressen in het geheugen staan, dan komt het adres met de index 6 op de plaats te staan van het adres met de index 5, dat daarmee gewist is. Het laatste, dus het 7-de, komt daarmee op de plaats te staan van nummer 6. En zo is het adres met de index 5 'echt' verdwenen.

Bij het programmeren volgens dit principe, duikt voor het eerst een in elkaar geschoven lus op (nesten). In de lus met de variabele 'i1' bevindt zich de lus met variabele 'i2'. De lus 'i1' doorloopt alle data regels van de adressen die gewist moeten worden. De velden worden daarbij stuk voor stuk opgeschoven. Als alle velden zijn opgeschoven kan met de volgende worden begonnen. De 'buitenste' lus loopt tot 'wijzer-1' omdat er in regel 25220

de waarde 'i1+1' staat. Als U de bijbehorende regels bestudeert, zult U deze techniek van doorschuiven snel bevatten.

Nu hebben we ook dit deel van ons, inmiddels omvangrijk, programma voltooid. Er volgt nu alleen nog maar, zoals ook bij de andere delen, een volledige LISTING van dit deel:

```
25000 REM ==========
25010 REM ADRESSEN WISSEN
25020 REM ==========
25030 IF wijzer=0 THEN GOSUB 500:GOTO 1000
25040 LOCATE 1,10
25050 INPUT"VOORNAAM:";voornaam$
25060 INPUT"NAAM      :";naam$
25070 lus=1
25080 IF adres$(lus,2)=voornaam$ AND adres$(lus,3)=naam$THEN 25110
25090 IF luswijzer THEN lus=lus+1:GOTO 25080
25100 fout$="ADRES IS AFWEZIG":GOSUB 200:GOTO 1000
25110 GOSUB 300:LOCATE 1,10
25120 FOR index=1 TO 7
25130 PRINT veld$(index);adres$(lus,index)
25140 NEXT index
25150 LOCATE 1,18
25160 INPUT"ADRES WISSEN (j/n) ";antwoord$
25170 IF antwoord$="j" THEN 25200
25180 IF antwoord$="n" THEN 1000
25190 fout$=" 'j' of 'n' indrukken":GOSUB 200:GOTO 25150
25200 FOR i1=lus TO wijzer-1
25210 FOR i2=1 TO 7
25220 adres$(i1,i2)=adres$(i1+1,i2)
25230 NEXT i2
25240 NEXT i1
25250 wijzer=wijzer-1
25260 GOTO 1000
```

ADRESSEN AFGEVEN

Dit deel van het programma behoort zeker niet tot het meest eenvoudige. Hier gaan we voor het eerst iets over de PRINTER meedelen. Maar ook het uitvoeren op het beeldscherm zal mogelijk worden gemaakt. Naar keuze, of U nu het beeldscherm of de PRINTER wilt gebruiken, we voeren eerst de zoekbegrippen in. Maar laten we beginnen met de eerste regels:

```
30000 REM ==========
30010 REM ADRESSEN AFDRUKKEN
30020 REM ==========
30030 IF wijzer=0 THEN GOSUB 500:GOTO 1000
30040 LOCATE 1,10
30050 INPUT"PRINTER OF BEELDSCHERM (p/b)";antwoord$
30060 IF antwoord$="p" THEN apparaat=8:GOTO 30090
30070 IF antwoord$="b" THEN apparaat=0:GOTO 30090
30080 fout$=" 'p' of 'b' indrukken":GOSUB 200:GOTO 30040
```

Ook hier wordt eerst onderzocht of er wel gegevens in de computer aanwezig zijn (regel 30030). Daarna wordt de vraag gesteld of de printer dan wel het beeldscherm gebruikt moet worden. Het antwoord 'p' of 'b' wordt aan de stringvariabele antwoord$ toegevoegd. In het geval dat de printer gekozen wordt, wordt de waarde van 'apparaat' op 8 gezet. En dan hebben we een nieuw type PRINT-opdracht nodig die we hier dan eerst uiteenzetten:

Probleem:	Uitvoer op verschillende apparaten.
Commando:	PRINT# 9,.....
Parameter:	g – nummer van het apparaat (0-9) 0 – beeldscherm 1 – 7 beeldschermvenster (wordt in dit boek niet beschreven) 8 – printer 9 – cassette-recorder
Voorbeeld:	PRINT#8,"ADRESSENLIJST" Drukt de tekst "ADRESSENLIJST" af op de printer.
Opmerking:	De toevoeging '#g' kan ook bij de bekende commando's LIST en INPUT gebruikt worden.

We kunnen onze gegevens met PRINT dus ook naar de printer of naar de cassette-recorder sturen. We hebben aan de variabele 'apparaat' het bijbehorende nummer gegeven. Als we nu voor een uitvoer altijd 'PRINT #apparaat,.....' gebruiken, dan is het afhankelijk van de variabele 'apparaat', of het beeldscherm dan wel de printer gebruikt wordt.

Maar waarom kan het nummer van het apparaat ook bij de opdrachten LIST en INPUT toegepast worden? Bij de opdracht LIST is de uitvoer op de printer erg interessant. Hierbij moeten we er wel op toezien dat de toevoeging #8 niet ontbreekt. Twee voorbeelden:

LIST #8 – Hiermee wordt het hele programma op de printer afgedrukt.

LIST 1000-,#8 – Drukt de regels vanaf 1000 op de printer af.

De toevoeging '#8' bij het commando INPUT leest de gegevens van de cassette-recorder, maar daar komen we later nog op terug. Nu weer terug naar ons programma.

We hebben nu het nummer van het apparaat gekozen. Omdat we in beginsel niet alle adressen op ons apparaat af willen drukken, willen we voor elk veld een zoekkriterium bepalen. Laten we daartoe eerst eens de volgende regels bekijken:

```
30090 GOSUB 300:LOCATE 1,10
30100 PRINT"Zoekbegrippen:"
30110 PRINT "----------"
30120 PRINT
30130 FOR index=1 TO 7
30140 zoek$(index)=" "
30150 PRINT veld$(index);
30160 INPUT zoek$(index)
30170 NEXT index
```

In regel 30090 wordt een nieuw beeldscherm opgezet, voordat er in de regels 30100-30110 gesignaleerd wordt dat er nu zoekbegrippen worden ingevoerd. Deze invoer wordt weer via een lus tot stand gebracht. Voordat er met het zoeken wordt begonnen, moeten eerst alle zoekbegrippen, die in de array 'zoek$' staan worden gewist. Er kan immers nog een waarde uit een vorige zoekroutine in staan. Deze begrippen worden dan in regel 30160 ingelezen. Als bv. alle adressen uit Utrecht worden gezocht, drukt men bij de eerste 4 velden alleen maar de toets ENTER in. Het 5-de veld, waarin de plaatsnaam staat wordt dan gevuld met 'Utrecht' en met ENTER afgesloten. De resterende 2 velden blijven dan met ENTER buiten beschouwing. Als U echter alle dames uit Utrecht zoekt, dan moet U bij titel 'Mevrouw' intikken (en bij plaats 'Utrecht'). U kunt dus naar verschillende begrippen tegelijkertijd zoeken.

Na het invoeren van de zoekbegrippen moet het zoeken zelf beginnen:

```
30180 FOR i1=1 TO wijzer
30190 gevonden=0
30200 FOR i2=1 TO 7
30210 IF zoek$(i2)=" " OR zoek$(i2)=adres$(i1,i2) THEN gevonden=gevonden+1
30220 NEXT i2
```

Ook hier hebben we een geneste lus nodig, waarbij binnen de data-regels alle velden worden doorzocht. Na het begin van de buitenste lus (regel 30180) wordt in regel 30190) de variabele 'gevonden' op 0 gezet. Met behulp van deze variabele worden alle velden geteld, waarvoor hetzij geen zoekbegrip (ENTER) is ingevoerd, hetzij er een zoekbegrip is ingevoerd dat met de inhoud van een veld overeenstemt. Als voor een veld geen zoekbegrip is ingevoerd dan is de inhoud van de variabele zoek$(i2) leeg (zoek$(i2)=" "). Dit onderzoek en het verhogen van de waarde van de variabele 'gevonden' gebeurt in regel 30210. Als na het aflopen van de binnenste lus (i2) de waarde van de variabele 7 is, dan voldoet het adres aan het zoekkriterium en kan worden afgedrukt. De volgende regels leiden dit afdrukken in:

```
30230 IF gevonden<>7 THEN 30300
30240 IF apparaat=0 THEN GOSUB 300
30250 PRINT #apparaat
```

Hier wordt het afdrukken van de gevonden data-regel voorbereid. Als er bij een data-regel niet een bijpassend kriterium gevonden wordt ('gevonden' ongelijk aan 7) wordt over het 'afdrukken' heen gesprongen. Als het beeldscherm gekozen is, verschijnt er eerst een nieuw opschrift. Aansluitend wordt er op het gekozen apparaat een spatieregel afgedrukt. Er volgen nu de resterende regels van dit programmadeel, die daarna weer worden besproken:

```
30260 FOR index=1 TO 7
30270 PRINT #apparaat,veld$(index);adres$(i1,index)
30280 NEXT index
30290 PRINT #apparaat
30295 IF apparaat=0 THEN INPUT"DRUK OP ENTER";antwoord$
30300 NEXT i1
30310 GOSUB 300:LOCATE 1,18
30320 PRINT"**DATA EINDE**"
30330 INPUT"DRUK OP ENTER";ANTWOORD$
30340 GOTO 1000
```

In dit gedeelte draait alles om het afdrukken van de gevonden data-regels. De regels 30260 tot 30200 bepalen een lus, waarin de 7 velden van de adressen met hun betekenis op het gekozen apparaat worden afgedrukt.

Als het beeldscherm gekozen is, wordt het zoeken pas weer voortgezet als op de toets ENTER is gedrukt (regel 30295). Verder zoeken betekent hier dat de variabele i1 met de waarde 1 wordt verhoogd door de opdracht NEXT i1. Als echter de printer is gekozen dan wordt door het indrukken van ENTER over deze regel heen gesprongen. En worden alle adressen achter elkaar afgedrukt.

Voorbij het einde van de lus 'i1' wordt het zoeken gestaakt. Dit wordt door de mededeling ''**DATA EINDE**'' aan de gebruiker meegedeeld. Als de gebruiker nu op de toets ENTER drukt, wordt dit programmadeel beëindigd en springt men terug naar het menu.

Dit was een enorme molensteen aan ons 'adressen bestand'. We hoeven nu alleen nog maar de delen voor het 'lezen' en 'schrijven' van de gegevens 'van' en 'naar' de cassette recorder te behandelen.

Met de volgende LISTING kunt U nog eens het programma deel ''Adressen afgeven'' controleren:

```
30000 REM ==========
30010 REM ADRESSEN AFGEVEN
30020 REM ==========
30030 IF wijzer=0 THEN GOSUB 500:GOTO 1000
30040 LOCATE 1,10
30050 INPUT''PRINTER OF BEELDSCHERM (p/b)'';antwoord$
30060 IF antwoord$=''p'' THEN apparaat=8:GOTO 30090
30070 IF antwoord$=''b'' THEN apparaat=0:GOTO 30090
30080 fout$='' 'p' of 'b' indrukken'';GOSUB 200:GOTO 30040
30090 GOSUB 300:LOCATE 1,10
30100 PRINT''ZOEK BEGRIPPEN:''
30110 PRINT''----------''
30120 PRINT
30130 FOR index=1 TO 7
30140 zoek$(index)='' ''
30150 PRINT veld$(index)
30160 INPUT zoek$(index)
30170 NEXT index
30180 FOR i1=1 TO wijzer
30190 gevonden=0
30200 FOR i2=1 TO 7
30210 IF zoek$(i2)='' '' OR zoek$(i2)=adres$(i1,i2) THEN gevonden=gevonden+1)
30220 NEXT i2
30230 IF gevonden7 THEN 30300
30240 IF apparaat=0 THEN GOSUB 300
30250 PRINT #apparaat
30260 FOR index=1 TO 7
30270 PRINT #apparaat,veld$(index);adres$(i1,index)
30280 NEXT index
30290 PRINT #apparaat
30295 IF apparaat=0 THEN INPUT ''DRUK NU OP ENTER'';antwoord$
```

```
30300 NEXT i1
30310 GOSUB 300:LOCATE 1,18
30320 PRINT"**DATA EINDE**"
30330 INPUT"DRUK NU OP ENTER";antwoord$
30340 GOTO 1000
```

DATA WEGSCHRIJVEN

We komen nu langzaam maar zeker aan het eind van ons programma 'adressenbestand'. De gegevens die inmiddels zijn ingevoerd zullen door het uitzetten van de computer echter weer verloren gaan. We willen onze adressen nu op de cassette-band opslaan. Waar moeten we dan allereerst op letten? We beginnen ook dit deel met het gebruikelijke opschrift en de vraag, of er wel gegevens in de computer staan.

```
10000 REM ==========
10010 REM DATA WEGSCHRIJVEN
10020 REM ==========
10030 IF wijzer=0 THEN GOSUB 500:GOTO 1000
```

Deze regels zijn ons niet meer onbekend. Voordat we onze gegevens op een cassetteband kunnen opslaan, moet deze eerst worden ingelegd en worden terug gespoeld. De volgende regels nodigen U daartoe uit:

```
10050 LOCATE 1,12
10060 PRINT"CASSETTE INLEGGEN AUB."
10070 PRINT"EN TERUG SPOELEN."
10080 PRINT
10090 INPUT"DRUK DAARNA OP DE TOETS ENTER";antwoord$
```

Ook deze regels zullen U geen moeilijkheden bezorgen. Maar nu gaat het gebeuren. Om de gegevens met de opdracht PRINT #,... op de cassetteband te kunnen schrijven, moeten we de recorder eerst daarop voorbereiden. Men zegt dan, het bestand moet eerst 'geopend worden'. Daartoe bestaat het volgende commando:

Probleem:	Openen van bestand om te schrijven.
Commando:	OPENOUT "naam" of OPENOUT "!naam".
Parameter:	naam – naam van de gegevens.

Voorbeeld:	OPENOUT "adressen" opent bestand "adressen" om te schrijven.
Opmerking:	Als voor de naam van de data een uitroepteken (!) staat (bv !ADRES) dan worden de mededelingen van de computer onderdrukt.

Met deze opdracht openen we dus ons bestand voordat deze gegevens kunnen worden overgezonden. We gebruiken het uitroepteken om mededelingen zoals 'Press REC and PLAY then any key:" te onderdrukken. We werken met onze eigen mededelingen. Tik nu dan de bijbehorende opdrachten in:

```
10100 PRINT"DRUK DE TOETSEN REC EN PLAY"
10110 PRINT"EN DAARNA ENTER –"
10120 INPUT antwoord$
10130 OPENOUT "!ADRESSEN"
```

We kunnen onze gegevens nu opslaan. Het eerste wat we op de band schrijven is het aantal data-regels dat, zoals bekend is, in de variabele 'wijzer' staat. Daarna sturen we alle aparte velden van de adressentabel met behulp van een geneste lus, die U ook niet meer onbekend voorkomt. Allereerst veld 1 tot 7 van de eerste data-regel, dan veld 1 tot 7 van de tweede data-regel enz.:

```
10140 PRINT #,wijzer
10150 FOR i1=1 TO wijzer
10160 FOR i2=1 TO 7
10170 PRINT #,adres$(i1,i2)
10180 NEXT i2
10190 NEXT i1
```

Na het opslaan van deze gegevens moet het bestand ook weer gesloten worden. De opdracht hiertoe heet 'CLOSEOUT'. Dit zetten we in de volgende regel:

```
10200 CLOSEOUT
```

Als de gegevens opgeslagen zijn willen we dat dit wordt meegedeeld. Hiervoor dienen de volgende regels:

```
10210 GOSUB 300
10220 PRINT"DE ADRESSEN ZIJN OPGESLAGEN!"
10230 FOR i=1 TO 2000:NEXT i
10240 GOTO 1000
```

Allereerst maken we weer een nieuw beeldscherm. Hierna krijgen we een wachtlus van ca. 2 seconden. Tenslotte gaan we weer terug naar het menu.

Zo eenvoudig is nu het opslaan van de gegevens op de cassette-recorder. Maar nu willen we deze gegevens weer terug halen.

DATA INLEZEN

Het teruglezen van de gegevens gaat bijna op dezelfde manier als het opslaan ervan. Hierbij wordt er niet onderzocht of er al gegevens in de computer staan. De gebruiker moet dus eerst de ingevoerde adressen opslaan voordat hij nieuwe adressen inleest:

```
5000 REM ==========
5010 REM DATA INLEZEN
5020 REM ==========
5030 LOCATE, 1,12
5040 PRINT"CASSETTEBAND INLEGGEN SVP."
5050 PRINT"EN TERUG SPOELEN."
5060 PRINT
5070 INPUT"DRUK DAARNA ENTER!";antwoord$
5080 PRINT
5090 PRINT"DRUK NU PLAY, DAARNA ENTER"
5100 INPUT antwoord$
```

Nadat alle voorbereidingen getroffen zijn, kan het bestand weer worden geopend. Nu echter om te lezen. Laten we daarom eerst maar weer eens de omschrijving bekijken:

Probleem:	Openen van een bestand om te lezen.
Commando:	OPENIN"NAAM" of OPENIN"!NAAM".
Parameter:	naam – naam van het bestand.
Voorbeeld:	OPENIN"Adressen" opent het bestand "Adressen" om te lezen.
Opmerking:	Als voor de naam van het bestand een uitroepteken(!) staat, worden de mededelingen van de computer onderdrukt.

Laten we dan nu deze opdracht in ons programma zetten:

```
5110 OPENIN''!ADRESSEN''
5120 INPUT #9,wijzer
5130 FOR i1=1 TO wijzer
5140 FOR i2=1 TO 7
5150 INPUT #9,adres(i1,i2)
5160 NEXT i2
5170 NEXT i1
5180 CLOSEIN
```

Hier wordt het bestand ''ADRES'' op de cassette geopend om te lezen. Dit lezen wordt door OPENIN mogelijk gemaakt. Daarna wordt het aantal gegevens dat in dit bestand staat ingelezen. Deze variabele wordt als eindwaarde voor de lus 'i1' gebruikt. Er worden dus evenveel gegevens ingelezen als er tevoren waren weggeschreven. Dit kanaal wordt in regel 5180 met het commando CLOSEIN weer gesloten.

```
5190 GOSUB 300:PRINT''GEGEVENS ZIJN GELADEN''
5200 FOR i=1 TO 2000:NEXT i
5210 GOTO 1000
```

De laatste regels van dit deel geven weer een melding en na een wachtlus gaan we weer terug naar het menu.

PROGRAMMA BEËINDIGEN

Zoals aan het begin reeds werd meegedeeld moet het programma niet met de netschakelaar worden beëindigd. Hoe een programma dan wel moet worden beëindigd, zullen we nu demonstreren. Laten we weer eerst naar de volgende regels kijken:

```
35000 REM ==========
35010 REM EINDE VAN HET PROGRAMMA
35020 REM ==========
35030 IF wijzer=0 THEN 35150
```

De eerste regels kennen we nu wel. Maar regel 35030 ziet er erg ongebruikelijk uit. Als 'wijzer=0' dan wordt het programma zonder meer afgebroken. Dit gebeurt in regel 35150.

```
35040 LOCATE 1,12
35050 PRINT''ZIJN ALLE GEGEVENS WEGGESCHREVEN (j/n)'';
```

```
35060 INPUT antwoord$
35070 IF antwoord$="n" THEN 1000
35080 IF antwoord$="j" THEN 35100
35090 fout$=" 'j' of 'n' indrukken":GOSUB 200:GOTO 35040
```

Langzaam maar zeker zien we de zin in van het programmadeel 'EIND VAN HET PROGRAMMA'. Er wordt eerst gevraagd of alle gegevens zijn opgeslagen. Men kan immers dit deel van het programma per ongeluk hebben aangeroepen. Als U nu het antwoord 'n' geeft komt U weer in het menu terecht. Als de adressen echter wel zijn verwerkt kan het geen kwaad om het programma te beëindigen.

```
35100 GOSUB 300:LOCATE 1,12
35110 PRINT"HET PROGRAMMA KAN MET"
35120 PRINT" 'GOTO 1000' WEER GESTART WORDEN"
35130 PRINT"ZONDER DAT ER GEGEVENS VERLOREN GAAN!"
35140 PRINT
35150 END
```

Voor het programma nu 'echt' wordt afgebroken wordt er nog een belangrijke mededeling gedaan: nadat het programma is beëindigd kan het met 'GOTO 2000' weer gestart worden ZONDER DAT ER GEGEVENS VERLOREN ZIJN GEGAAN. Het commando 'RUN' zal alle variabelen eerst wissen, en dat is in zo'n geval niet aan te raden.

En nu zijn we dan klaar. U bezit nu een, tot het laatste toe becommentarieerd, adressenbestand. Bovendien heeft U bij het ontstaan ervan, ongetwijfeld een hoop geleerd. Al wat U nu geleerd hebt, kunt U bij het tot stand brengen van Uw eigen programma's benutten. Ook kunt U dit programma naar eigen inzicht veranderen. Als U één of meer van de diverse onderdelen nog niet goed hebt begrepen, werk ze dan rustig nog een keer door. Als U 'BASIC' tot in alle details zult willen leren dan kan ik U het boek 'Het trainingsboek voor de CPC464' aanbevelen.

Kleuren en grafieken

Voor de vele kopers van de talrijke home-computers is een duidelijk kriterium de mogelijkheden van de kleuren en van de vaardigheden in de grafische toepassingen. De CPC464 biedt U een onvergelijkbare kleurenpracht en uitstekende grafische eigenschappen. Zo beschikt de gebruiker van de CPC464 over 27 kleuren, waarvan er, afhankelijk van de grafiekoplossing, 2, 4 of 16 kleuren kunnen worden gekozen. Het oplossende vermogen van de CPC464 gaat tot 640 x 200 beeldpunten, en dat is voor een computer van deze prijsklasse werkelijk sensationeel.

BEDRIJFSSOORTEN VOOR GRAFIEKEN

De CPC464 werkt met drie verschillende grafiekoplossingen, die we met het bekende commando 'MODE' kunnen kiezen. Daarmee bestaat er een onmiddellijk verband tussen de 'dichtheid' van de tekens en de oplossing van de grafiek. De volgende tabel zal U hiervoor een overzicht verschaffen.

Commando	Tekens per regel	Oplossing grafiek	Oplossing teken	Kleuren
MODE 0	20	160 x 200	32 x 8	16
MODE 1	40	320 x 200	16 x 8	4
MODE 2	80	640 x 200	8 x 8	2

De samenhang tussen de tekendichtheid en de grafiekoplossing wordt aan de hand van dit overzicht duidelijk. U ziet ook dat een teken in de MODE 2 vorm uit 8 x 8 punten is opgebouwd. In de MODE 1 vorm is dit aantal verdubbeld tot 16 x 8 punten. In de meeste vergelijkbare computers is een teken opgebouwd uit 8 x 8 punten en dat met 40 tekens op een regel. De CPC464 biedt dus in elk opzicht een dubbele oplossing in computers.

Op één geheugenplaats worden er steeds 8 punten vastgehouden. Het hele beeldscherm bestaat uit 640 x 200, dat is dus 128 000 punten. Voor het opslaan hiervan in het geheugen, heeft men derhalve 16 000 geheugenplaatsen nodig. Van het totale geheugen van 64K byte wordt dus ca. 16K byte gebruikt voor het grafiek-geheugen. Bij de andere vormen van de oplossing wordt dit aantal niet verminderd, omdat elke punt door een kleur wordt aangegeven. Ook de normale tekst wordt door het grafische geheugen vastgehouden. In vele andere computers worden de tekst en de grafische afbeeldingen gescheiden. Daar bestaat dus voor een grafische afbeelding een ander geheugen, dan voor de tekstafbeelding.

DE BEELDSCHERMKLEUREN

Na het inschakelen van de computer ziet U de standaardkleuren van de MODE 1 vorm, blauw en lichtgeel, aangenomen dat U een kleuren-monitor of een TV-toestel hebt aangesloten. Dit zijn slechts twee van de 27 mogelijke kleuren. In de volgende tabel kunt U alle kleuren zien:

Kleurnummer	Kleur	Kleurnummer	Kleur
0	zwart	13	wit
1	blauw	14	pastelblauw
2	lichtblauw	15	oranje
3	rood	16	rose
4	cyaan	17	pastelcyaan
5	lichtviolet	18	lichtgroen
6	lichtrood	19	zeegroen
7	purper	20	lichtturkoois
8	lichtcyaan	21	citroengroen
9	groen	22	pastelgroen
10	turkoois	23	pastelturkoois
11	hemelsblauw	24	lichtgeel
12	geel	25	pastelgeel
		26	lichtwit

Bij een dergelijke grote kleurenpracht is het niet eenvoudig om deze kleuren een naam te geven. Er duikt een kleur op als lichtwit, die behalve misschien in waspoeder-reclames, eigenlijk onzinnig is. Maar laten we daar nou niet op letten.

Elke kleur is door een eenduidig getal bepaald. De getallen die bij de standaardkleuren behoren zijn dus 1 voor blauw en 24 voor geel.

DE RANDKLEUREN

Niet alleen de kleuren van de karakters en de kleur van het beeldscherm kunnen worden bepaald, ook de rand kan zijn eigen kleur krijgen. De rand is dat gedeelte van het beeldscherm dat men met de cursor niet kan bereiken. Bij het aanzetten van de computer kan men deze rand niet waarnemen, omdat hij dezelfde kleur heeft als de achtergrond. Met een bepaalde opdracht kan men deze kleur echter veranderen. Zoals altijd volgt nu eerst een omschrijving van deze opdracht:

Probleem:	Verandering van de randkleur.
Commando:	BORDER kleur 1, kleur 2
Parameter:	Kleur 1 – kleurnummer Kleur 2 – wordt alleen maar verstrekt als de kleuren regelmatig tussen kleur 1 en kleur 2 moeten wisselen.
Voorbeeld:	BORDER 0 verandert de kleur van de rand in zwart. BORDER 0,13 laat de rand tussen wit en zwart knipperen.
Opmerking:	In elke vorm (MODE 0 TOT MODE 2) kan men een van de 27 kleuren kiezen.

Nu kent U inmiddels het eerste commando om de 27 kleuren te kiezen. Wacht dus nu niet langer, maar verander de kleur van de rand in kleurnummer 24. De bijbehorende opdracht is:

BORDER 24

En nu ziet U de rand die totnutoe voor ons verborgen was. Deze rand kunt U nu in alle mogelijke kleuren laten zien. Maar omdat wij te lui zijn om dit allemaal op te schrijven, laten we dat door een programma doen. Dit programma laat alle kleuren achter elkaar zien. Dat wordt door het volgende programma gedaan:

```
10 FOR kleur=1 TO 26
20 BORDER kleur
30 x$=INKEY$:IF INKEY$=" " THEN 30
40 NEXT kleur
```

Wat nu, dit programma zegt U niets? Dan heeft U zeker hoofdstuk 5 overgeslagen en wilt U zeker dit plezierige hoofdstuk voorrang geven. Maar dat heeft geen enkele zin, omdat er hier steeds weer kleinere programma's opduiken waarmee de funkties van nieuwe grafiekopdrachten moeten worden duidelijk gemaakt.
Nu weer terug naar het programma. In een lus roepen we de afzonderlijke kleuren op, die dan in regel 20 een betekenis krijgen. Zoals met alle andere parameters kunt U ook dit kleurnummer als variabele invoeren. Regel 30 wacht tot er een toets door U wordt ingedrukt waarmee dan de volgende kleur verschijnt. Zo kunt U op de snelste manier alle kleuren van Uw CPC464 te zien krijgen. Als U echter alleen maar over een 'groene moni-

tor' beschikt, ziet U deze 'kleuren' alleen maar in diverse nuanceringen. Dit moet U echter niet tegenhouden om de verschillende grafiekmogelijkheden te onderzoeken.

Uit de omschrijving van het commando heeft U kunnen opmaken, dat U de beeldschermrand ook kunt laten knipperen. Om dit te bereiken, moeten er twee kleurnummers worden ingebracht, die dan door een komma moeten worden gescheiden. De kleur van de rand zal dan regelmatig tussen deze twee kleuren wisselen. Laten we dit eens proberen door twee kleuren te nemen waartussen een groot contrast bestaat, zwart en wit bv. Het commando hiervoor is:

BORDER 0,26

Het is zeker niet aan te bevelen om met deze twee kleuren verder te gaan als U Uw ogen wilt beschermen. Laten we dat knipperen maar weer uitschakelen.

BORDER 1

Dit knipperen kan een nuttige funktie hebben bij een foutmelding. De gebruiker zal op deze manier ook optisch opmerkzaam worden gemaakt op een fout.

INSTELLEN VAN DE KNIPPERSNELHEID

Met de CPC464 kunnen alle wensen worden vervuld. Zo kan men ook de knippersnelheid van het beeldscherm en van de rand instellen. Daarvoor gebruiken we het volgende commando:

Probleem:	Instellen van de knippersnelheid.
Commando:	SPEED INK tijd 1, tijd 2
Parameter:	tijd 1 – Tijd voor de eerste kleur tijd 2 – Tijd voor de tweede kleur (beide in eenheden van 0.02 sec.)
Voorbeeld:	SPEED INK 50, 100 Houdt de eerste kleur een seconde aan en de tweede kleur twee sec.
Opmerking:	Voor 'tijd 1' en 'tijd 2' kan men een getal invoeren van 0 tot 255. Daarmee kan men een tijd instellen van 0.02 tot 5.2 seconden.

Met deze opdracht kunt U dus het knipperritme van de randkleuren bepalen. Ook voor deze opdracht hebben we een programma:

```
10 INPUT "1. Kleur (0-26)      ";kleur 1
20 INPUT "Vertraging (1-255)";vert 1
30 INPUT "2. Kleur (0-26)      ";kleur 2
50 BORDER kleur1, kleur 2
60 SPEED INK vert1,vert2
```

In dit programma kunt U de beide kleuren en de vertraging zelf bepalen. De daarop volgende commando's laten dan de rand knipperen.

VERANDEREN VAN DE KARAKTERKLEUR

De standaardkleur van de karakters is, zoals bekend, lichtgeel (nummer 24). U kunt echter ook een andere kleur hiervoor nemen. Het volgende commando kan U hierbij helpen:

Probleem:	Veranderen van de kleur van het karakter.
Commando:	PEN #n, kleurreg
Parameter:	#n – nummer van beeldscherm venster (kan ook verwaarloosd worden) Kleurreg – nummer van kleurregister (0-15)
Voorbeeld:	PEN 1 Verandert de kleur van de karakters in de kleur die in register 1 staat.
Opmerking:	Kleurregisters worden met het commando INK veranderd.

Ik kan me voorstellen dat U door al deze kleurregisters een beetje in de war bent geraakt. Zoals U al weet, kunnen niet alle 27 kleuren tegelijkertijd gebruikt worden. Afhankelijk van de vorm (MODE) kunnen 2, 4 of 16 van de beschikbare kleuren worden gebruikt. In de 40-tekens modus kunt U uit 4 kleuren kiezen. Deze vier kleuren worden door het commando INK mogelijk gemaakt. Laten we ook deze opdracht eerst eens bekijken:

Probleem:	Bepalen van de keuze-kleuren.
Commando:	INK kleurreg, kleur1, kleur2

Parameter:	kleurreg – Kleurregister (0-15) Kleur 1 – Kleurnummer (0-26) Kleur 2 – wordt alleen aangeduid als er tussen de kleuren 'kleur 1' en 'kleur 2' regelmatig moet worden gewisseld (knipperen).
Voorbeeld:	INK 1,3 Zet de kleur 'rood' in kleurregister 1.
	INK 2,1,9 Laat de kleuren van register 2 tussen blauw en groen knipperen

Omdat we slechts uit 16 kleuren (MODE 0) kunnen kiezen, bestaan er ook 16 kleurregisters met de nummers 0 tot en met 15. Willen we nu bijvoorbeeld de kleuren van de karakters in rood veranderen, dan moeten we nummer 3 voor rood in een kleurregister zetten. Nemen we hiervoor register 1, dan luidt de bijbehorende opdracht:

INK 1,3

Nu kan de opdracht 'PEN' de kleuren van de tekens veranderen, indien dit kleurregister wordt aangegeven.

PEN 1

Deze opdracht zegt nu: 'verander de karakters in de kleur die in het kleurregister 1 staat'. In dit register hebben we tevoren de kleur 'rood' gezet. Bij het inschakelen van de computer heeft het register reeds een kleur. Welke kleuren dat zijn kunnen we in de volgende tabel zien:

Kleurregister	MODE 0	MODE 1	MODE 2
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24

10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	1,24	20	1
15	16,11	6	24

Omdat de kleurkeuze bij MODE 1 en MODE 2 beperkt is, worden de kleuren hier uiteraard herhaald. Met het kleurregister 0 bij MODE 1 bijvoorbeeld, worden tegelijkertijd de kleurregister 4, 8 en 12 bepaald. Daarom kunnen er in MODE 1 niet meer dan 4, en in MODE 2 niet meer dan 2 kleuren gekozen worden.

De opdracht INK zorgt ervoor dat alle karakters de kleur krijgen, die tevoren in het kleurregister is bepaald. Een voorbeeld: de standaardkleuren van de karakters worden na het inschakelen van de computer uit het register 1 gehaald. Als U na dit inschakelen de kleur van register 1 verandert, dan verandert tegelijkertijd de tekst die op het beeldscherm staat. Als U bijvoorbeeld de volgende opdracht geeft:

```
INK 1,13
```

dan worden alle tekens op het beeldscherm direct wit. Als U echter de karakters met de kleur uit register 2 op het beeldscherm zet, heeft dit geen verandering van kleur uit register 1 ten gevolge.

De kleur van de achtergrond wordt bepaald door het kleurregister 0. Als U dus nu de kleur van de karakters uit ditzelfde register haalt, dan zijn de kleuren van de karakters en van de achtergrond gelijk. Het gevolg is, dat men de karakters nu niet meer kan zien. Probeer dat maaar eens met:

```
PEN 0
```

Nu is de cursor verdwenen. Om dit weer ongedaan te maken, moet U 'blind' intikken:

```
PEN 1
```

Het kleurregister 1 behoort dus standaard bij de karakters. Omdat 1 register altijd voor de kleur van de achtergrond nodig is, kunnen we in MODE 1 slechts over drie verschillende karakter kleuren beschikken. Het volgende programma laat U de drie verschillende kleuren zien:

```
10 MODE 1
20 INK 1,12:INK 1,9:INK 3,13
```

```
30 PEN 1:PRINT''GEEL''
40 PEN 1:PRINT''GROEN''
50 PEN 3:PRINT''WIT'
```

We gebruiken voor de drie kleuren van de karakters, de kleurregisters 1, 2 en 3. In regel 20 bepalen we hiermee de kleuren 'geel', 'groen' en 'wit'. Hieropvolgend worden er drie regels afgedrukt in de drie verschillende kleuren.

In een kleurregister kan men echter ook twee verschillende kleuren onderbrengen, waardoor de karakters gaan knipperen. Zet eerst eens de computer in de beginstand (SHIFT-CTRL-ESC). Daarna veranderen we de kleur van de karakters (register 1) in 'zwart/wit'.

```
INK 1,0,13
```

En direct gaan alle tekens op het beeldscherm knipperen. Ook de tekens die U hierna invoert, gaan knipperen omdat ook deze in het kleurregister 1 staan. Om de volgende karakters onafhankelijk van de kleuren te maken die al op het beeldscherm worden gebruikt, is een volgende opdracht nodig:

```
PEN 2
```

Vanaf nu komen de kleuren van de karakters uit het register 2. En het knipperen kunt U weer uitzetten door kleur 24 in het register 1 te zetten.

```
INK 1,24
```

Er volgt hier weer een klein demonstratie-programma:

```
10 MODE 1
20 INK 2,1,26
30 INK 3,26,1
40 PEN 2:PRINT''HEEN EN WEER''
50 PEN 3:PRINT''HEEN EN WEER''
60 PEN 1
```

VERANDEREN VAN DE ACHTERGRONDKLEUR

We hebben tot nu toe de kleuren van de karakters en van de rand veranderd. Met de volgende opdracht kan men ook de kleur van de achtergrond veranderen:

Probleem:	Veranderen van de kleur van de achtergrond.
Commando:	PAPER #n,kleurreg
Parameter:	#n – nummer van het venster van het beeldscherm (kan ook weggelaten worden). Kleurreg – nummer van het kleurregister (0-15).
Voorbeeld:	PAPER 2 Verandert de kleur van de achtergrond in de kleur van kleurregister 2
Opmerking:	Om de hele achtergrond van kleur te laten veranderen, moet na de opdracht PAPER de opdracht CLS gegeven worden.

De opdracht 'PAPER' lijkt in zoverre op de opdracht 'PEN' dat er bij alle twee eerst een register moet worden aangegeven waarin de gewenste kleur wordt meegedeeld. Standaard voor de achtergrond is het register 0. Ook hier kunnen we niet het register 1 gebruiken, omdat dit al voor de kleuren van de karakters wordt gebruikt. Dit zou dan weer de tekens onzichtbaar maken.

Maar laten we nu eens de opdracht PAPER toepassen. We willen de achtergrondkleur lichtrood maken. Deze kleur staat standaard in register 3. We geven dus de volgende opdracht:

```
PAPER 3
```

Nu wordt echter niet de gehele achtergrond lichtrood, maar slechts die regels die hierna volgen. Om de hele achtergrond nu lichtrood te maken, moeten we eerst het beeldscherm van CLS wissen.

Ook de achtergrond kan tot knipperen worden gebracht, als we twee kleuren in het kleurregister brengen. We willen de achtergrond met de kleuren zwart en wit laten knipperen. Ook hiervoor maken we eerst het beeldscherm schoon door SHIFT-CTRL-ESC. We tikken nu achterelkaar de volgende opdrachten in:

```
INK 3,0,14
PAPER 3
CLS
```

De frequentie van dit knipperen, kan men met de inmiddels bekende opdracht 'SPEED INK' veranderen. Als U bijvoorbeeld elke seconde de kleur van de achtergrond wilt laten veranderen, dan geeft U de opdracht:

SPEED INK 50,50

Herinnert U het zich nog: de waarden bij deze opdracht stellen eenheden voor van 0.02 seconden (en 50 x 0.02 sec. = 1 sec.).

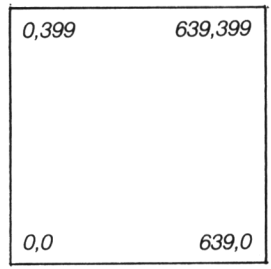
Tenslotte weer een klein, maar interessant programmavoorbeeld:

```
10 MODE 1
20 INK 2,0,13
30 INK 3,13,0
40 SPEED INK 50,50
50 PEN 2
60 PAPER 3
70 CLS
80 PRINT"POSITIEF EN NEGATIEF"
```

HIGH-RESOLUTION GRAFIEK

Zoals al eerder is meegedeeld, kunt U tussen drie verschillende grafiek-oplossingen met de bijbehorende kleuren kiezen. Om figuren te tekenen, bestaan er een paar opdrachten die we uitsluitend met de high-resolution grafiek zullen uiteenzetten.

Herinnert U zich nog: de high-resolution grafiek bestaat uit 640 x 200 beeldpunten en twee kleuren, een achtergrondkleur en een kleur voor de katakters. Om elke punt apart aan te kunnen roepen, moet men zich het beeldscherm als een coördinaten-systeem voorstellen. Vergelijkbaar met het commando LOCATE, waarbij de cursor op een willekeurige plaats gezet kan worden, wordt ook bij grafiek-opdrachten de plaats met twee waarden gegeven. Het eerste getal bepaalt de waarde van de X-positie (horizonttale as), het tweede getal bepaalt de waarde van de Y-positie (vertikale as). Omdat de oplossingsverhouding van 640 x 200 niet dezelfde is als van het beeldscherm, loopt de Y-waarde niet van 0 tot 199 maar van 0 tot 399, waarbij twee Y-waarden dezelfde positie bepalen (0 en 1, 2 en 3 enz). De vier hoekpunten hebben de volgende coördinaten:



Als U zich met dit systeem vertrouwd hebt gemaakt, kunnen we ons met de eerste commando's voor de grafische voorstellingen gaan bezighouden.

TEKENEN VAN PUNTEN

Probleem:	Plaatsen van een punt
Commando:	PLOT x,y
Parameter:	x – X-positie (0-639) y – Y-positie (0-399)
Voorbeeld:	PLOT 0,0 Zet een punt linksonder op het beeldscherm

De grafiek-opdrachten kan men in het algemeen goed begrijpen als men het coördinatensysteem beheerst. Het zetten van enkele punten wordt echter een vervelende zaak, als men niet meer opdrachten kan gebruiken. Zo kan het erg interessant zijn om punten door middel van een lus op het beeldscherm te zetten. Het is erg eenvoudig om bv. een rechte lijn te maken. Weliswaar bestaat er een speciale opdracht voor, maar voor het oefenen met de opdracht PLOT is het desondanks toch erg interessant.

Probeer eens om een rechte lijn aan de uiterste linker kant van het beeldscherm van onder naar boven te tekenen. Het volgende programma voert deze opdracht uit:

```
10 MODE 2
20 FOR y=0 TO 393
30 PLOT 0,y
40 NEXT y
```

Dit is echt een eenvoudig programma. Laten we daarom de moeilijkheidsgraad iets verhogen. Schrijf eens een programma, waarbij er een rand om het hele beeldscherm ontstaat. Hier is dan de oplossing:

```
10 MODE 2
20 FOR y=0 TO 399
30 PLOT 0,y:PLOT 639,y
40 NEXT y
50 FOR x=0 TO 639
60 PLOT x,0:PLOT x,399
70 NEXT x
```

U had toch zeker geen vier lussen nodig? De twee evenwijdige lijnen kunnen met een lus tegelijkertijd worden gemaakt. Maar dit is een kleine truc; met wat meer ervaring kan men dat zelf ook.

Het tekenen van een loodrechte lijn kan ook veel sneller gedaan worden. Herinnert U het zich nog: steeds twee op elkaar volgende y-waarden bepalen het zelfde punt. We kunnen de Y-lus dus halveren. Daartoe vergroten we de stapgrootte (STEP-parameter) tot 2. Verander daarom de volgende regel:

```
20 FOR y=0 TO 399 STEP 2
```

Dit 'sneller gaan' zal men bij het opnieuw starten van het programma wel waarnemen. Denk er steeds aan of het tekenen in de richting van de Y-as niet wat kan worden bespoedigd.

TEKENEN VAN EEN CIRKEL

Omdat de CPC464 geen aparte opdracht kent voor het tekenen van een cirkel, moeten we dat op een andere manier doen. Elk apart punt moet worden berekend en dan op het beeldscherm worden gezet. Daarbij heeft U de volgende formules nodig:

```
x = r * cos(a)
y = r * sin(a)
```

Waarbij 'r' de straal is en 'a' de hoek. Door gebruik te maken van een lus, waarbij de hoek met de waarde 1 toeneemt, kunnen wel elk punt van de cirkel berekenen. Bekijkt U eens het volgende programma:

```
10 MODE 2
20 x=320:y=200:r=100
30 DEG
40 FOR a=1 TO 360
50 PLOT x+r*cos(a),y+r*sin(a)
60 NEXT a
```

In regel 20 wordt het middelpunt en de straal van de cirkel bepaald. Het commando DEG deelt aan de computer mee dat voor de volgende hoek berekeningen in graden moeten plaatsvinden. Een alternatief hiervoor is de hoekmaat RAD. Hierbij geldt: '90 graden = ½ PI radialen'. Bij wetenschappelijke berekeningen wordt meestal RAD gebruikt. In regel 40 staat het begin van de lus. Als de begin- en eindwaarde veranderen, wordt er slechts een deel van de cirkel getekend. In regel 50 staat de eigenlijke formule van de cirkel. Bij de

berekende positie van de puntwaarde moeten nog de coördinaten van het middelpunt (x en y) worden opgeteld. Tenslotte volgt dan nog het einde van de lus.

Met deze routine kunt U nu een cirkel tekenen, ook als de wiskundige samenhang U niet duidelijk is. Als U ook nog een ellips wilt tekenen, kan ik U helaas niet verder helpen. Misschien kunt U dan nog wat oude boeken uit de kast halen en wat nascholing doen in de goniometrie. Maar alle gekheid op een stokje – voor alle mooie grafieken is het begrip 'algoritme' niet overbodig.

TEKENEN VAN RECHTE LIJNEN

Probleem:	Tekenen van een rechte
Commando:	DRAW x,y
Parameter:	2 – X-coördinaat y – Y-coördinaat
Voorbeeld:	DRAW 639,0 Tekent een rechte vanaf de plaats van de cursor tot de aangegeven x en y positie.

De enige vraag die hierbij gesteld kan worden, is: de cursor. Bij het begin, dus na het aanzetten van de computer, bevindt de cursor zich op de plaats '0,0'. Elke volgende grafiek-opdracht begint bij deze cursor. Zet men dus een punt op '100,200' dan staat daarna ook de cursor op dit punt. Een daarop volgende DRAW opdracht tekent een rechte vanuit dit punt. Het laatste punt van deze rechte is dan weer de nieuwe positie van de cursor.

Er bestaat echter ook de mogelijkheid om de cursor op een willekeurige plaats te zetten. De volgende opdracht doet dit:

Probleem:	Plaatsen van de cursor.
Commando:	MOVE x,y
Voorbeeld:	MOVE 200,250 Zet de grafiek cursor op de plaats '200,250'.

Het is U zeker opgevallen dat de parameter niet werd omschreven. Dat is opzet, omdat U intussen wel weet wat er met 'x,y' wordt bedoeld.

Laten we nu eens een rechte tekenen aan de linkerkant van het beeldscherm, zoals we dat al eerder met het commando PLOT hebben gedaan. Hiervoor zijn er slechts twee opdrachten nodig, kort na het inschakelen van de computer, zelfs maar een:

```
10 MODE 2
20 MOVE 0,0
30 DRAW 0,399
```

De opdracht 'MOVE 0,0' hoeft men na het aanzetten van de computer niet te gebruiken omdat de cursor dan al op de gewenste plaats staat.

We willen nu ook weer de rand, zoals dat ook bij de opdracht PLOT gedaan is, met DRAW tekenen:

```
10 MODE 2
20MOVE 0,0:REM hoek linksonder
30 DRAW 0,399:REM naar hoek linksboven
40 DRAW 639,399:REM naar hoek rechtsboven
50 DRAW 639,0:REM naar hoek rechtsonder
60 DRAW 0,0:REM naar hoek linksonder.
```

Dit programma verklaart zichzelf. Het verbinden van afzonderlijke punten is nu ook geen probleem meer. Interessant is het tekenen van rechten in verbinding met lussen. Overtuig U daar zelf van:

```
10 MODE 2
20 FOR i=0 TO 399 STEP 10
30 MOVE 399,i
40 DRAW 0,399-i
50 NEXT i
```

Een eenvoudig programma met een grote uitwerking. Omdat het nu zo lekker gaat, nog een programma:

```
10 MODE 2
20 MOVE 0,0
30 FOR i=0 TO 399 STEP 10
40 DRAW 399,i
50 DRAW 399-i,399
60 DRAW 0,399-i
70 DRAW i,0
80 NEXT i
```

Is het niet aardig, dat zulke beelden op het beeldscherm kunnen ontstaan. Wellicht probeert U zelf eens zulke fantasie-grafieken te ontwerpen. Men kan er veel plezier mee beleven, ook als het bij het begin nog niet helemaal goed gaat. Trouwens, ook ik heb me bij dit beeld het hoofd gebroken. De moeilijkheid hierbij is, om de beelden die men in zijn hoofd heeft zitten, om te rekenen in een coördinaten-systeem, dus om een algoritme te vinden. Dat lijkt op echte denksport.

RELATIEF TEKENEN

Ter afsluiting van dit hoofdstuk leert U nog een variant op de tot nu toe beschreven commando's. Bij de opdrachten PLOT en MOVE was de plaats van de cursor niet relevant, omdat we altijd een 'absoluut' punt hebben aangewezen. Voegen we echter aan de opdracht de letter 'R' toe, dan krijgt deze opdracht een andere uitwerking. De opdrachten zien er dan als volgt uit:

PLOTR x,y
DRAWR x,y
MOVER x,y

Met deze opdrachten worden geen absolute x en y coördinaten bepaald maar de relatieve waarde ervan. Kortom, bij het bepalen van de eindwaarde wordt de x-positie van de cursor erbij opgeteld. Dat zelfde geldt ook voor de y-waarde. Een voorbeeld: de grafiek-cursor bevindt zich op de plaats '200,100'. De opdracht 'PLOTR 50,50' zet nu een punt op de positie '250,150'.

Het volgende voorbeeld zal nog meer licht hierover laten schijnen. Er moet een vierkant van 10 bij 10 punten worden getekend. De coördinaten van de linker onderhoek van dit vierkant bevinden zich in de variabelen x en y.

```
10 MODE 2
20 x=200:y=100
30 MOVE x,y
40 DRAWR 10,0:REM 10 naar rechts
50 DRAWR 0,10:REM 10 naar boven
60 DRAWR -10,0:REM 10 naar links
70 DRAWR 0,-10:REM 10 naar onder
```

Zoals U hebt gezien, kunnen er hierbij ook negatieve getallen worden gebruikt. Dit is nodig om ook plaatsen links van de cursor en onder de cursor mogelijk te maken.

Dit was dan de afsluiting van het hoofdstuk over grafieken voor de beginners. Nog meer grafiekopdrachten vindt U in het handboek of in andere werken van DATA-BECKER-BOEKEN VOOR DE CPC464.

Oproepen van klanken

En computer is een zeer geschikt apparaat om geluiden mee te kunnen produceren. De processor, het eigenlijke werkpaard in de computer, wordt door een toongenerator gestuurd. Deze generator wekt een frequentie op, afhankelijk van de computer, van ca. 1-4 Hhz. De eenvoudigste manier is deze toon op de luidspreker te zetten. De toon die daarbij ontstaat, ligt echter buiten het bereik van ons gehoor en daarom moet deze frequentie aanmerkelijk lager liggen. Dit wordt bereikt door deze frequentie te delen. De frequentie die dan ontstaat, wordt door ons als toon waargenomen.

Dit werd door de eerste generatie computers zo gedaan. Omdat men niet tevreden was met deze tonen, die op het geluid leek dat de horloges maken, werd het opwekken van tonen in de computer steeds verder ontwikkeld. Het resultaat daarvan is dat vele homecomputers nu een zogenoemde synthesizer bezitten, die op een chip zijn ondergebracht. De hiermee opgewekte tonen lijken verbazend veel op de 'natuurlijk' muziekinstrumenten.

Deze chip van de CPC464 kan men eigenlijk niet een synthesizer noemen. Daarvoor ontbreken er te veel eigenschappen, zoals de verschillende golfvormen (driehoek, sinus, enz) en de filters die de klank moeten beïnvloeden. Ook is het eigenlijke principe van de synthesizer, de frequentie (dat is de toonhoogte) afhankelijk van de spanning te maken, niet aanwezig. De toongenerator van de CPC464 maakt de frequentie afhankelijk van een deelfaktor.

De toongenerator van de CPC464 beschikt over drie stemmen. Men kan dus drie tonen tegelijk horen. En verder is er ook een ruisgenerator aanwezig, waardoor men naast de tonen ook geruis kan laten ontstaan.

Alles wat door de toongenerator wordt opgewekt, wordt hoorbaar gemaakt door de ingebouwde luidspreker. Daarom moet het geluid van een eventueel aangesloten TV toestel worden uitgezet. De geluidssterkte van de ingebouwde luidspreker kan aan de rechterzijde van de CPC worden ingesteld.

Wie zijn eigen composities echter toch in stereo wil horen, heeft de mogelijkheid om de uitgang van zijn CPC met een stereo toestel te verbinden. Daarvoor bezit de CPC aan de linker achterzijde een uitgangsbus (I/O). Hiervoor moet men zelf een aansluiting maken, omdat deze (nog) niet bij de detailhandel te verkrijgen is. De uitgang van de CPC moet dan verbonden worden met de ingang van het stereo apparaat, waar men normaal een cassette-deck of een tuner op aansluit. Ook kan men de vaak aanwezige AUX-bus gebruiken.

In het volgende hoofdstuk leert U de belangrijkste BASIC commando's. Het gehele complexe opwekken van tonen is in dit boek niet beschreven. Verdere informatie vindt U in Uw handboek of in andere DATA BECKER boeken voor de CPC.

HET SOUND COMMANDO

Probleem:	Bepalen van toon-eigenschappen.
Commando:	SOUND a,b,c,d,e,f,g
Parameter:	a – Kanaal-status (in dit hoofdstuk 1). b – Deler, bepaald de frequentie (0-4095). Frequentie = 125000/deler c – Duur van de toon (-32768 tot 32767), in eenheden van 0.01 seconden. d – Geluidssterkte (0-15). e – Omhullingskromme voor geluidssterkte (0-15). f – Omhullingskromme voor de toon (0-15). g – Ruiskarakter (0-15).
Voorbeeld:	SOUND 1,1000,100,7,0,0,0 Roept een toon op met de frequentie van 125 Hz, duur 1 sec., geluidssterkte 7.

Omdat deze opdracht nogal omvangrijk is, zullen we niet alle parameters bespreken. Met name het programmeren met de omhullingskrommen, valt buiten het kader van dit boek. Maar de kennis die U in dit hoofdstuk op zult steken, is een goede basis voor een verder verdieping met behulp van Uw handboek.

DE TOONHOOGTE

We willen eerst de voor ons interessante parameters beschrijven. De eerste parameter 'a' geeft de status aan, waarmee het o.a. mogelijk is om drie stemmen tegelijk te laten klinken. In de volgende voorbeelden gebruiken we slechts één stem en wel stem 1.

De volgende parameter 'b' bepaalt de toonhoogte. Hier komt een deler te staan waar de grondfrequentie van 125000 door gedeeld moet worden. Omdat de deler een waarde tussen 0 en 4095 kan hebben zijn er frequenties tussen 30 en 125000 Hz mogelijk. Het

volgende programma laat U de tonen van 30000 Hz tot 300 Hz horen. Daardoor heeft U wat meer begrip voor de frequenties.

```
10 FOR deler=4.1 TO 416 STEP 0.1
20 PRINT INT(125000/deler)
30 SOUND i,deler,1,7,0,0,0
40 NEXT deler
```

De funktie INT snijdt van de waarde, die tussen de haakjes staat, alle plaatsen achter de komma af, omdat deze in dit voorbeeld overbodig zijn. U ziet ook dat we op de plaatsen van de parameters variabelen kunnen zetten, zoals bij alle andere commando's.

De verandering van de toonhoogte door middel van een lus, maakt talrijke effecten mogelijk. Zo kan men bv. eenvoudig een sirene simuleren:

```
10 FOR deler=120 TO 200
20 SOUND 1,deler,2,7,0,0,0
30 NEXT deler
40 FOR deler=200 TO 120 STEP-1
50 SOUND 1,deler,2,7,0,0,0
60 NEXT deler
70 GOTO 10
```

Deze sirene, waarin men een politieauto herkent, kan worden afgebroken met de toets 'ESC' (twee keer drukken). Voor diegenen onder U die nog wat met deze sirene willen spelen, hebben we de volgende variant van dit programma:

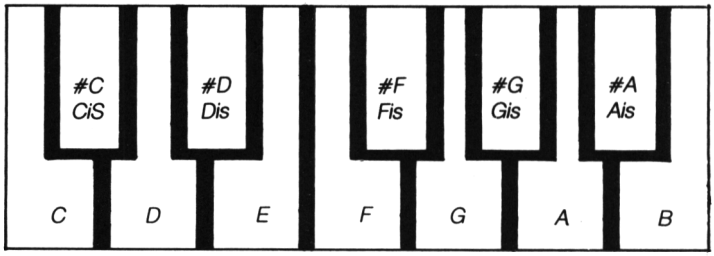
```
 5 INPUT"BEGIN DELER";a
 6 INPUT"EIND DELER";b
 7 INPUT'TEMPO";t
10 FOR deler=a TO b
20 DOUNT 1,deler,t,7,0,0,0
30 NEXT deler
40 FOR deler=b TO a STEP-1
50 SOUND 1,deler,t,7,0,0,0
60 NEXT deler
70 GOTO 10
```

Hier worden de belangrijkste eigenschappen weer in de variabelen gezet, die dan in het daarop volgende programma worden gebruikt. Ook dit kan weer met 'ESC' worden afgebroken.

MUZIEK UIT FREQUENTIES

Muziekinstrumenten wekken een toon traploos op. Een bepaalde volgorde van de tonen is voor ons een aangenaam gehoor. Voor verder begrip is het nodig om een aantal begrippen uiteen te zetten.

Een zogenoemde toonladder bestaat uit 12 tonen, zoals U in het volgende overzicht van een klavier kunt zien.



Een pianoklavier bestaat uit meerdere van deze delen. Elk deel noemt men een oktaaf. De eerste oktaven bevatten de lage tonen de laatste oktaven de hoogste.

In welke delen zijn de afzonderlijke tonen nu opgedeeld. De berekening van de frequenties van de aparte tonen gebeurt internationaal vanuit de toon 'A' van de middelste oktaaf. De frequentie van deze toon is 440 Hz. We zullen nu deze toon eens laten horen. Welke deler moeten we dan voor deze frequentie gebruiken? Herinnert U het zich nog: de deler wordt met de formule 'Deler=125000/frequentie' berekend. Bij een frequentie van 440 Herz hoort dan de deler 284.090909. De deler moet echter een geheel getal zijn zonder cijfers achter de komma. Daardoor ontstaat er een hele kleine afwijking in de frequentie, die zo klein is dat dit slechts specialisten zal opvallen. Laten we nu de middelste 'A' opwekken:

```
SOUND 1,284,100,7,0,0,0
```

Hoe moeten nu de andere tonen uit deze grond-frequentie worden berekend? De bijbehorende formule ziet er als volgt uit:

```
FREQUENTIE = 440 * (2 ↑ (OKTAAF+(N-10)/12))
```

OKTAAF – van -3 tot 4
N – plaats van de toon in de toonladder ('C'=1 'A'=10)

De mathematische oorsprong van deze formule doet hier niet ter zake. Belangrijk is dat U deze formule toe kunt passen. Als we bv. de toon 'C' uit de onderste oktaaf willen horen, wordt de frequentie als volgt berekend:

440 * (2 ↑ (3+(1-10)/12) = 32.7031957 Hz

De uitkomst hiervan moet worden omgerekend in een deler. De formule hiervan kent U inmiddels al.

125000/32.7031957 = 3822.25643

De deler 3822 wekt dus de toon 'C' op van de onderste oktaaf. Op deze manier kunnen we een programma schrijven dat alle tonen van de toonladder laat horen:

```
10 INPUT"OKTAAF (-3 tot 4)";oktaaf
20 FOR toon=1 TO 12
30 frequentie=440*(2 ↑ (oktaaf+(toon-10)/12))
40 deler=INT(125000/frequentie)
50 SOUND 1,deler,20,7,0,0,0
60 NEXT toon
```

Het wordt steeds interessanter om met de toongenerator te experimenteren. We zullen nu de kroon op dit hoofdstuk zetten. Het volgende programma 'KLAVIER VOOR BEGINNERS' verandert Uw CPC464 in een muziekinstrument:

```
10 MODE 0
20 DIM toon(90)
30 toon$="Q2W3ER5T6Y7UISO0P"
40 FOR i=1 TO 17
50 index=ASC(MID$(toon$,i,1))
60 toon(index)=i
70 NEXT i
80 CLS
90 PRINT " KLAVIER VOOR"
100 PRINT" BEGINNERS"
110 LOCATE 1,5
120 PRINT" 2 3   5 6 7   9 0"
130 PRINT
140 PRINT"Q W E R T Y U I O P"
150 a$=INKEY$:IF a$=" " THEN 150
160 IF toon(ASC(A$))=0 THEN 150
170 toon=toon(ASC(A$))
180 frequentie=440*(2 ↑ (toon/12))
190 deler=INT(125000/frequentie)
200 SOUND 1,deler,30,7,0,0,0
210 GOTO 150
```

Dit kleine programma heeft een grote uitwerking. Wellicht kunt U zelf een aantal verbeteringen aanbrengen, nadat U dit programma heeft bestudeerd.

RUIS

Bij de verklaring van het commando 'SOUND', hebben we al meegedeeld dat er een parameter is die het 'ruisen' veroorzaakt. Deze parameter staat op de laatste plaats en kan de waarde 0 tot 15 bezitten, waarmee het ruisen kan worden geaktiveerd. Met een en dezelfde SOUND-opdracht kan men dus 'tonen' en ook 'ruisen' laten ontstaan, afhankelijk van de waarde van de parameters 'b' en 'g'. Het volgende overzicht zal dat toelichten:

SOUND a,b,c,d,e,f,g

Parameter		Resultaat	
'b	'g'	Toon	Ruis
0	0	neen	neen
1-4095	0	ja	neen
0	1-15	neen	ja
1-4095	1-15	ja	ja

Hier ziet U dat er zowel een toon als ook een ruis tegelijkertijd kan ontstaan en dat met een SOUNDcommando.

Het ruisen kan door verandering van de parameter tussen 1 (helder ruisen) en 15 (gedempt ruisen) worden ingesteld. Het volgende programma maakt dit duidelijk:

```
10 FOR r=1 TO 15
20 SOUND 1,0,20,7,0,0,r
30 NEXT r
```

Dit ruisen wordt niet beïnvloed door andere tonen, omdat de tweede parameter op nul staat en daarmee de tonen worden uitgeschakeld.

Door deze ruisgenerator kunnen verschillende effecten worden bereikt. Geluiden zoals bv. wind, schoten, explosies ed. kunnen met deze ruisgenerator worden nagebootst. Het volgende programma bootst een lokomotief na:

```
10 SOUND 1,0,10,7,0,0,15
20 SOUND 1,0,5,7,0,0,0
30 GOTO 10
```

Deze lokomotief kunt U met de toets 'ESC' laten stoppen. Na het ruisen (0.1 sec.) volgt een pauze (0.05 sec.). Herinnert U zich nog dat de duur door de derde parameter wordt bepaald in eenheden van 0.01 seconden.

Met het laatste voorbeeld willen we een 'schot' laten horen. Daartoe moeten we de geluidssterkte laten verminderen.

```
1:a$=INKEY$:IF A$='' '' THEN 10
20 FOR v=7 TO 1 STEP -1
30 SOUND 1,0,10,v,0,0,15
40 NEXT v
50 GOTO 10
```

Deze 'schoten' ontstaan door het indrukken van een toets. Het programma kan met de toets 'ESC' worden afgebroken. Misschien wilt U nu zelf eens wat geluiden maken. De hiervoor benodigde commando's zijn voor U nu geen probleem meer.

De cassette-recorder

U zult zeker niet over het hoofd gezien hebben, dat er zich in Uw CPC464 een cassette-recorder bevindt. De bedoeling van dit apparaat is niet om U wat muziek te bezorgen bij het programmeren, maar om de programma's en de gegevens te kunnen bewaren. Het eenvoudige opslaan en weer teruglezen van programma's heeft U al geleerd. In het programma over het 'adressenbestand' heeft U ook al gegevens op de cassette verwerkt. Dit alles zal in dit hoofdstuk dan ook niet meer behandeld worden. Andere bijzondere toepassingen zullen hier worden uiteengezet en met voorbeelden worden verduidelijkt.

SCHRIJFSNELHEID

Het BASIC van de CPC464 kan de snelheid van de recorder beïnvloeden. In de normale toestand wordt er met 1000 baud gewerkt. Dat betekent dat er 1000 signalen per seconde naar de recorder gestuurd worden. Een teken bestaat uit 8 van zulke signalen, die men in de vaktaal 'bits' noemt. Elke toestand van een bits (aan of uit) wordt als een toon gecodeerd. Daarmee bestaan er twee tonen, waarmee de totale informatie kan worden overgestuurd. Bij het opslaan (programma's of data) wordt ook nog een hoge toon opgeroepen, waarmee het begin van een programma of het begin van data wordt aangeduid. De hierop volgende bits zijn, als de volumeregelaar voldoende is open gezet, alleen nog maar als gekrijs te identificeren.

Voor een optimale zekerheid worden de gegevens dubbel opgetekend. Beide delen worden dan met elkaar vergeleken. Op deze manier kan eenvoudig vastgesteld worden of het opslaan foutloos is gegaan. De computer meldt dit met 'READ ERROR' (laad fout).

Bij het gebruik van hoogwaardige cassettes (chroom dioxide) is dit dubbele optekenen niet nodig. Daardoor wordt de laadsnelheid verdubbeld. BASIC heeft daar een speciale opdracht voor:

SPEED WRITE 1

Als deze opdracht wordt ingevoerd, zal de cassette-recorder met een dubbele snelheid werken. Het is echter aan te bevelen om van zulke programma's een copy te maken die dan met de enkelvoudige snelheid is opgenomen. De praktijk heeft uitgewezen dat programma's die met de dubbele snelheid zijn opgenomen, na enige tijd toch niet goed meer kunnen worden teruggelezen.

Om weer om te schakelen naar de enkelvoudige snelheid moet men de volgende opdracht geven:

SPEED WRITE 0

Wat echter als een programma moet worden teruggelezen dat met 2000 baud is opgeslagen? De CPC464 schakelt intern automatisch op de snelheid over, waarmee het programma is opgeslagen. En zo hoeft U er alleen maar op te letten met welke snelheid een programma moet worden opgeslagen. Bij het teruglezen gebeurt dit automatisch.

PROGRAMMA'S ALS ASCII DATA.

Gedurende het verloop van dit boek doken er begrippen op als 'programma' en 'bestand', die we steeds zorgvuldig uit elkaar moesten houden. Maar wat is nu eigenlijk het verschil? De programma's bevinden zich zowel in het geheugen als op de cassetteband of in een ander medium in een gecomprimeerde vorm. Zo worden bv. de commando's als code opgeslagen, daar anders bv. het commando 'PRINT' vijf geheugenplaatsen nodig zou hebben. Om dit te voorkomen bestaat elk commando uit een code (0-255), waarmee dan slechts één geheugenplaats nodig is. Dit is te vergelijken met de karakters van de CPC464, die immers ook uit een code bestaan ('ASC' en 'CHR$').

De CPC biedt de mogelijkheid om de programma's als uitgeschreven tekst op de cassette op te slaan. Voor dit commando volgt dan eerst een omschrijving:

Probleem:	Opslaan van programma's in uitgeschreven tekst.
Commando:	SAVE"NAAM",A
Parameter:	"NAAM" – naam van het programma.
Voorbeeld:	SAVE"ADRESSEN",A Slaat het programma met de naam "ADRESSEN" in uitgeschreven vorm op.
Opmerking:	Dit opgeslagen programma kan men wel als bestand maar niet als programma teruglezen.

Op dit moment heeft U er natuurlijk geen idee van waarvoor deze manier van opslaan gebruikt kan worden. Pas bij het toepassen van een tekstverwerker zult U dit voordeel kunnen inschatten. De als ASCII-bestand opgeslagen programma's kunnen door een tekstverwerker ingelezen en verwerkt worden. Dit wordt interessant als een LISTING van een programma in een tekst moet worden overgenomen, zoals in dit boek ook in praktijk gebracht werd.

Met het volgende voorbeeld kunt U zich er van overtuigen, dat deze programma's in de vorm van een bestand kunnen worden ingelezen. Voer daarom eens het volgende demonstratie-programma in:

```
10 REM ==========
20 REM PROGRAMMA ALS BESTAND
30 REM ==========
40 PRINT"DIT PROGRAMMA IS BEËINDIGD"
```

En zet nu dit programma met de volgende opdracht op de band:

```
SAVE"DEMO,A
```

Nu willen we dit programma als een bestand inlezen en op het beeldscherm laten zien. Daarvoor hebben we het volgende programma nodig:

```
10 OPENIN"DEMO"
20 FOR i=1 TO 4
30 INPUT #9,A#
40 PRINT A#
50 NEXT i
60 CLOSEOUT
```

Als U nu dit programma intikt en start zult U tot Uw verbazing waarnemen dat het zojuist opgeslagen programma "DEMO" nu op het beeldscherm verschijnt.

BESCHERMDE PROGRAMMA'S

Misschien heeft U wel eens geprobeerd om een programma dat U kant en klaar in de winkel hebt gekocht, in de computer in te lezen en daarna te LISTEN. Als dat niet gelukt is, komt dat omdat deze programma's beschermd zijn. Naar wens kunt U ook Uw eigen programma's beschermen. De volgende omschrijving laat dit zien:

Probleem:	Beschermen van cassette-programma's.
Commando:	SAVE"NAAM",P
Parameter:	"NAAM" naam van het programma.

Voorbeeld:	SAVE"TOP SECRET",P Schrijft het programma "TOP SECRET" zodanig naar de cassette-recorder dat dit niet meer te LISTen is.
Opmerking:	Men moet een versie van het programma normaal opslaan omdat deze bescherming niet meer ongedaan kan worden gemaakt.

Deze omschrijving zegt voldoende voor het beschermen van programma's. We willen dit commando ook direct uitproberen. Tik daarom de volgende BASIC regels in:

```
10 REM ==========
20 REM TOP SECRET
30 REM ==========
40 PRINT"Deel de code mee";
50 INPUT a4
60 IF a4<>"xy12" THEN NEW
70 PRINT"O.K."
```

Dit programma willen we nu beschermd opslaan. Dat doen we met het commando:

```
SAVE"TOP SECRET",P
```

Nu spoelen we de band terug tot het begin en laden het programma met het commando:

```
LOAD"TOP SECRET"
```

Daarna starten we het programma met 'RUN'. Wat nu, het programma doet het niet. U hoeft echter niet bang te zijn dat iets fout is gegaan. Een beschermd programma kan alleen met 'RUN' geladen en gestart worden en niet met 'LOAD' en 'RUN'. Spoelen we onze cassette dus nog een keer terug en geef het volgende commando:

```
RUN"TOP SECRET"
```

Na het laden van dit programma wordt het automatisch gestart. Kan men deze code misschien nog kraken? U kunt proberen om het programma met de toets 'ESC' af te breken. Druk deze toets dan twee keer in. En zie, het programma wordt beëindigd. Maar juich niet te vroeg, want als U nu het commando 'LIST' geeft, gebeurt er niets, omdat het programma uit het geheugen verdwenen is. Bij het afbreken met de toets 'ESC' werd het gewist. Dit betekent dat het programma niet gekraakt kan worden. Maar dat wil nog niet zeggen dat ervaren kenners van de CPC464 ook zonder resultaat zullen blijven. Met het kennen

van de machine is elke code, al is zij nog zo goed, te kraken. Tot eind nog een samenvatting:

- De toevoeging 'P', achter de opdacht SAVE beschermt het programma tegen onbevoegd listen.
- Het programma kan alleen met 'RUN' geladen en gestart worden.
- Als het programma met 'ESC' wordt gestopt dan wordt het geheugen gewist.
- De bescherming kan niet meer ongedaan worden gemaakt.

BEVEILIGEN VAN EEN GEHEUGENDEEL

Het totale geheugen van de CPC464 bedraagt 64 Kbyte, dat zijn 65535 bytes (tekens). Als een deel van dit bereik moet worden opgeslagen, is daar het volgende commando voor nodig:

Probleem:	Opslaan van een deel van het geheugen.
Commando:	SAVE''NAAM'',B,van,tot,start.
Parameter:	''NAAM'' – naam van het programma van – Beginadres (0-65535 tot – Eindadres start – Startadres bij machinetaalprogramma's.
Voorbeeld:	SAVE''ROUTINE'',B,49152,65535 Slaat de gehele beeldscherm inhoud op.
Opmerking:	Een dergelijk gedeelte van het geheugen moet met 'RUN' gestart worden.

Dit commando is vooral voor het opslaan van machinetaal programma's interessant. Maar dat is alleen voor ervaren programmeurs bestemd, die met de snelheid (traagheid) van BASIC niet tevreden zijn.

Verder kan deze opdracht ook gebruikt worden voor het opslaan van gemaakte grafieken. Daartoe moet men de hele beeldscherminhoud (49152-65535) naar de cassette schrijven. Bij het gebruik van 2000 Baud duurt dit ongeveer 7 minuten. Dit is vooral bij driedimensionale grafieken het geval.

INHOUDSOPGAVE VAN EEN CASSETTE

Een effectief registreren van programma's kan gedaan worden met kleine cassettes die voor dit doel normaal te verkrijgen zijn en die speciaal voor home-computers worden gemaakt. Omdat deze cassettes een looptijd hebben van 5 tot 10 minuten per kant, kan men op elke kant een programma zetten. Dit vereenvoudigd het terugvinden van aparte programma's.

Hoe moet men dit echter doen bij een 'gewone' cassette waar meerdere programma's achter elkaar staan. Dan moet men de stand van de bandteller steeds bijhouden om een programma snel terug te kunnen vinden. Wie echter alleen maar wil weten welke programma's of bestanden zich op een band bevinden, biedt de CPC464 nog een commando aan:

Probleem:	Weergeven van de inhoudsopgave van een cassette band.
Commando:	CAT
Voorbeeld:	CAT Geeft een overzicht van de inhoud van de ingelegde band. Het programma type wordt als volgt meegedeeld: $ – BASIC programma's. % – Beschermde programma's. * – ASCII-bestanden. & – Deel van de geheugeninhoud.
Opmerking:	De gehele cassette moet worden ingelezen.

Als U nu dit commando gaat proberen zult U zien dat niet alleen maar het type, maar ook de lengte van het programma resp. het bestand wordt meegedeeld. Hiermee heeft men altijd een goed overzicht van de programma's die op de cassette staan.

HET AAN ELKAAR KNOPEN VAN PROGRAMMA'S

Elke programmeur ontwikkelt in de loop der tijd een aantal uitgekiende onderprogramma's, die in vele programma's kunnen worden ondergebracht. Door een omvangrijke bibliotheek van onderprogramma's duurt het ontwikkelen van nieuwe programma's veel korter. Maar hoe moet men dan deze onderprogramma's in het nieuwe hoofdprogramma

onderbrengen? De omvangrijke taalschat van de CPC464 heeft ook hier een oplossing voor:

Probleem:	Verbinden van programma's.
Commando:	MERGE"NAAM"
Parameter:	"NAAM" – naam van het programma.
Voorbeeld:	MERGE"UPRO 1" Voegt het programma met de naam "URPO 1" toe aan het programma dat zich reeds in het geheugen van de computer bevindt.
Opmerking:	Als er gelijke regelnummers in beide programma's voorkomen, wordt er over de regels die al in het geheugen staan geschreven.

De funktie van deze opdracht zal in het volgende overzicht duidelijk worden:

Progr. in geheug.	Progr. op cass.	Uitkomst
10 REM P1 20 REM P1	5 REM P2 15 REM P2	5 REM P2 10 REM P1 15 REM P2 20 REM P1
10 REM P1 20 REM P1	5 REM P2 10 REMP2	5 REM P2 10 REM P2 20 REM P1

Hier wordt duidelijk dat de regels van beide programma's zodanig worden geordend, dat het in overeenstemming is met de voorschriften die bij het ordenen van de regels gelden. Als er echter twee regels met hetzelfde nummer worden gebruikt, dan zal het regelnummer dat zich al in de computer bevindt worden overschreven. Dat is ook nodig omdat een regelnummer slechts een keer in een programma voor kan komen.

Als de regelnummers van de twee programma's, die we met elkaar willen verbinden, niet met elkaar in overeenstemming zijn, moeten ze eerst worden aangepast. Daarvoor kunnen we bv. het reeds genoemde 'RENUMBER' gebruiken.

Afsluitend nog een voorbeeld voor de toepassing van het commando 'MERGE'. Voer daarvoor eerst het volgende programma in:

```
10 REM programma 2
25 REM programma 2
30 REM programma 2
45 REM programma 2
```

Omdat we dit programma later weer nodig hebben, moet dit eerst op een cassette-band worden weggeschreven. Dat doen we met de opdracht:

```
SAVE"PROG 2"
```

Nu wissen we het geheugen met 'NEW'. En voeren het programma in dat we met het vorige willen verbinden:

```
10 REM programma 1
20 REM programma 1
30 REM programma 1
40 REM programma 1
50 REM programma 1
```

Nu gaan we deze twee programma's met elkaar verbinden, nadat we eerst de cassette-band naar het begin hebben terug gespoeld. Daarna geven we het commando:

```
MERGE "PROG 2"
```

De computer gedraagt zich als bij het normale laden van een programma. De aansluitende opdracht 'LIST' laat zien wat er gebeurd is:

```
10 REM programma 2
20 REM programma 1
25 REM programma 2
30 REM programma 2
40 REM programma 1
45 REM programma 2
50 REM programma 1
```

De twee regels 10 en 30 werden door de regels van het tweede programma overschreven. Alle andere regels zijn op een juiste manier geordend. Het commando 'MERGE' zal u bij het ontwikkelen van nieuwe programma's erg behulpzaam zijn.

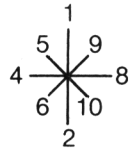
Nog meer opdrachten

Als U dit hoofdstuk van het boek met goed resultaat heeft bereikt, zult U zeker niet verwachten dat U nog meer commando's van Uw CPC464 zult leren. Maar er zal U niets in de weg worden gelegd. U wordt met alle commando's geconfronteerd waarvan de complexiteit binnen de grenzen blijft. Bij alle opdrachten vindt U zoals gewoonlijk een uitvoerige beschrijving evenals verklarende voorbeeld-programma's.

JOYSTICK

Als U Uw CPC464 eens van de achterzijde bekijkt, zult U aan de rechterzijde een aansluiting zien die men bij elke home-computer heeft. Hier kan men de zogenoemde Joysticks op aansluiten, die er hoofdzakelijk toe dienen om bij spelen optimaal te kunnen sturen.

Omdat het bepalen van de stand van een joystick bij de CPC erg eenvoudig is, zullen we de bijbehorende opdracht geven:

Probleem:	Stand van de Joystick.
Funktie:	JOY(n).
Parameter:	n – nummer van de joystick.
Voorbeeld:	PRINT JOY(0) Laat de code van de joystick zien als deze een zekere stand heeft.
Opmerking:	Er gelden de volgende codes.  Vuurknop 1=32 Vuurknop 2=16

U zult gemerkt hebben dat er twee joysticks kunnen worden toegepast, terwijl er slechts een poort is. Er kunnen echter slechts twee joysticks aangesloten worden, als U de originele joysticks van SCHNEIDER (of joysticks van een andere firma die speciaal voor de CPC gemaakt worden) gebruikt. Als U echter genoeg heeft aan 1 joystick, kunt U elke andere joystick gebruiken (bv. voor ATARI, COMMODORE 64 ed.). De code hiervan is dan 'JOY(0)'.

Laten we eerst eens een programma schrijven waarmee we de uitwerking van het gebruik van een joystick optisch kunnen laten zien.

```
10 PRINT JOY(0)
20 GOTO 10
```

Als U nu dit programma intikt en start zult U de code zien van de stand die bij de joystick hoort. Op deze waarde kan dan het bijbehorende programma reageren.

Met de funktie JOY(0) kunnen we een indrukwekkend programma schrijven, waarmee we op het beeldscherm kunnen tekenen. De 'vuurknop' wordt gebruikt om het beeldscherm weer te kunnen wissen.

```
 10 MODE 2
 20 CLS
 30 x=320:y=200
 40 PLOT x,y
 50 a=JOY(0)
 60 IF a=4 AND x0    THEN x=x+1:GOTO 40
 70 IF a=8 AND x639 THEN x=x-1:GOTO 40
 80 IF a=1 AND y399 THEN y=y+1:GOTO 40
 90 IF a=2 AND y0    THEN y=y+1:GOTO 40
100 IF a=16 THEN CLS
110 GOTO 40
```

Dit programma ziet er eenvoudig uit. U zult hierbij zeker geen problemen hebben om de werkwijze te begrijpen en kleine verbeteringen aan te brengen. Zo is het heel goed mogelijk om de bewegingen ook diagonaal uit te voeren. Een tip: U heeft daarvoor nog vier IF statements nodig waarbij de variabelen 'x' en 'y' op een geldige waarde worden onderzocht, om dit daarna te kunnen aktualiseren.

TOEVALSGETALLEN

Vele BASIC beginners schrijven hun eerste programma's met behulp van toevalsgetallen. Zonder toevalsgetallen zou het programmeren van een geluks-spel zelfs onmogelijk zijn. De CPC roept met deze funktie getallen op van 0.000000001 tot 0.999999999:

Probleem:	Maken van toevalsgetallen.
Funktie:	RND(0)

Voorbeeld:	PRINT RND(0) Maakt een toevalsgetal en laat dit op het beeldscherm zien.
Opmerking:	Dit getal ligt tussen 0.000000001 en 0.999999999.

Deze opdracht is eenvoudig toe te passen. Moeilijker wordt het als dat getal in een ander gebied moet liggen. Hiervoor moeten we de toevalswaarde aanpassen. Willen we bv. een getal tussen 0 en 999 hebben dan moet het programma er als volgt uitzien:

```
10 REM **********
20 REM toevalsgetallen
30 REM **********
40 getal=RND(0)
50 PRINT INT(getal*1000)
```

Het toevalsgetal wordt dus met 1000 vermenigvuldigd. Hierdoor komt het bereik dan tussen 0.000001 en 999.999999 te liggen. Omdat we de cijfers achter de komma niet nodig hebben, worden deze met de funktie INT eenvoudig 'weggesneden'.

Een andere moeilijkheid is als het gewenste getal niet bij nul maar bij een ander getal moet beginnen. Met de volgende formule kan men elk gewenst bereik krijgen:

Berekenen van toevalsgetallen:

INT (RND(1) * ((B+1)-A))+A

Waarin A= eerste getal
en B = laatste getal

Als we bijvoorbeeld de getallen van 1 tot 6 willen hebben, wordt deze formule als volgt opgebouwd:

INT(RND(1) * ((6-1)+1))+1
dat geeft dan:
INT(RND(1) * (5+1))+1
en tenslotte:
INT(RND(1) * 6)+1

Het afsluitende programma geeft een aantal rijen lotto-getallen (6 van de 41):

```
10 REM **********
20 REM LOTTO GETALLEN
30 REM **********
40 INPUT"HOEVEEL RIJEN ";rijen
50 FOR i1=1 TO rijen
60 FOR i2=1 TO 6
70 PRINT INT(RND(0)*41+1;"-";
80 NEXT 12
90 PRINT
100 NEXT i1
```

Ook dit programma zal U geen problemen opleveren. Maar ook dit programma heeft haken en ogen: in een rij kunnen een of zelfs meer gelijke getallen ontstaan. Het vermijden hiervan is een opdracht aan U. Een tip, sla de getallen die inmiddels gevonden zijn op in een array, vergelijk deze tabel dan met het getal dat nieuw gevonden wordt. Als dit laatste getal nog niet aanwezig is, wordt het geaccepteerd. Voor de volgende rij moet dit array eerst weer gewist worden. Veel plezier hiermee!

LEFT$

De volgende funkties die behandeld worden voor de bewerkingen met string's, zullen U vele nieuwe mogelijkheden geven. Zo kunnen er bijvoorbeeld gedeeltes van een string worden genomen om er verder mee te werken of kan een string in een andere string geplaatst worden. Het is zeker de moeite waard deze funkties te bestuderen. Elke funktie begint met een omschrijving:

Probleem:	Linker deelstring afsplitsen.
Funktie:	LEFT$(string,aant)
Parameter:	string – stringvariabele of string tussen aanhalingstekens. aant – aantal tekens dat men vanaf de linkerzijde van de string wil hebben.
Voorbeeld:	PRINT LEFT$("ABCDEFG",3) zet de drie linkse tekens van de string "ABCDEFG" op het beeldscherm (dat zijn A, B en C).
Opmerking:	Als 'aant' groter is dan het aantal tekens van de string, zullen alle tekens worden gebruikt.

Met deze interessante funktie kan men het linkerdeel van een string 'afsnijden'. Het volgende programma zegt meer, dan men met woorden kan doen:

```
10 PRINT"WAT IS UW NAAM ";naam$
20 aant=1
30 deel$=LEFT$(naam$,aant)
40 PRINT deel$
50 IF deel$=naam$ THEN 80
60 aant=aant+1
70 GOTO 30
80 PRINT"EIND"
```

Omdat deze funktie erg eenvoudig is, gaan we direct verder met de volgende:

RIGHT$

Probleem:	Rechter deelstring afsplitsen.
Funktie:	RIGHT$(string,aant)
Parameter:	string – stringvariabele of string tussen aanhalingstekens. aant – aantal tekens dat men vanaf de rechterzijde van de string wil hebben.
Voorbeeld:	PRINT RIGHT$("ABCDEFG",3) zet de drie rechtse tekens van de string "ABCDEFG" op het beeldscherm (dat zijn dan E, F, EN G).
Opmerking:	Als 'aant' groter is dan het aantal tekens van de string zullen alle tekens worden gebruikt.

Deze funktie lijkt erg veel op de vorige en hoeft ook niet verder niet verder worden toegelicht. U kunt in het programma van de vorige bladzijde eenvoudig het commando 'LEFT$' vervangen door 'RIGHT$'. Het verschil tussen deze twee funkties zal dan duidelijker worden.

We hebben nu een linker- en een rechterdeel van een string. Met de volgende funktie kunnen we zelfs een deel uit het midden halen:

MID$

Probleem:	Deel uit het midden van een string afsplitsen.
Funktie:	MID$(string,pos,aant)
Parameter:	string – stringvariabele of string tussen aanhalingstekens. pos – positie in de string waar het afsplitsen beginnen moet. aant – aantal tekens dat vanaf 'pos' moet worden afgesplitst.
Voorbeeld:	PRINT MID$("ABCDEFG",2,4) Zet vanaf positie 2 vier tekens van "ABCDEFG" op het beeldscherm (dat zijn dan B, C, D en E).
Opmerking:	Als 'aant' groter is dan de rest van de string, worden alle resterende tekens afgedrukt.

Tot nu toe hebben we het linker- en het rechterdeel van een string af kunnen snijden. Deze funktie geeft U nu de mogelijkheid om een deel UIT een string weg te nemen. Deze funktie heeft een parameter meer nodig dan de vorige twee, want niet alleen het aantal tekens maar ook de beginpositie moet worden meegedeeld.

Deze funktie wordt vaak ingezet als men een afzonderlijk teken uit een string wil hebben. Zo kan men bijvoorbeeld een aantal tekens loodrecht op het beeldscherm neerzetten. Het volgende programma demonstreert dat:

```
10 a$="LOODRECHT"
20 FOR p=1 TO 9:REM lengte van de string
30 PRINT MID$(a$,p,1)
40 NEXT p
```

De beginpositie wordt door de lus steeds een plaats naar rechts geschoven. De lengte van het stringdeel is constant 1, omdat we slechts een letter tegelijkertijd willen hebben. Maar wat nu als de lengte van de string niet bekend is? Ook dit probleem is opgelost als U de volgende funktie hebt leren kennen:

BEPALEN VAN DE LENGTE VAN EEN STRING

Probleem:	Bepalen van de stringlengte.
Funktie:	LEN(string)
Parameter:	string – stringvariabele of string tussen aanhalingstekens.
Voorbeeld:	Dit drukt de lengte van de string a$ af op het beeldscherm (7).
Opmerking:	Als de string 'leeg' is geeft deze funktie de waarde 0.

Als we nu deze funktie in het programma van de vorige bladzijde inbouwen, wordt ook een willekeurige string loodrecht afgedrukt:

```
10 INPUT "STRING :";a$
20 FOR p=1 TO LEN(1$)
30 PRINT MID$(a$,p,1)
40 NEXT p
```

Deze oplossing is eleganter dan de vorige, omdat hierbij de lengte van de string onbekend mag zijn.

Nadat U nu de belangrijkste stringfunkties hebt leren kennen, volgt nu nog een klein, maar zeer indrukwekkend, programma:

```
10 REM **********
20 REM LICHTKRANT
30 REM **********
40 CLS
50 INPUT"TEKST: ";tekst$
60 INPUT"LENGTE:";lengte
70 INPUT"TEMPO: ";tempo
80 1=LEN(tekst$)
90 tekst$=STRING$(1,"-")+tekst$+STRING$(1,"-")
100 For i=1 TO 1
110 LOCATE 12,20
120 PRINT MID$(tekst$,i,lengte)
130 FOR wacht =1 TO tempo:NEXT wacht
```

```
140 NEXT i
150 GOTO 100
```

De tekst die in onze lichtkrant moet verschijnen wordt eerst in de variabele tekst$ ingelezen. Daarna wordt de lengte van het deel, wat we willen laten zien, bepaald en een 'lengte' toegewezen. Nadat we de snelheidsfaktor 'tempo' hebben ingevoerd, krijgt de variabele '1' de lengte van de tekst. Omdat we de tekst met een begin-streepje en een eind-streepje willen afsluiten, hebben we regel 90 toegevoegd. Hier hebben we het commando 'STRING$' gebruikt, dat we ook al in het programma 'adressenbestand' hebben toegepast, om een keten van strings te construeren. De daarop volgende lus geeft steeds een string, die de lengte heeft van 'lengte', dat bij het begin van het programma werd gevraagd. Er ontstaat hierbij een tekst die na iedere lusdoorgang 1 plaats naar links verschoven wordt. De wachtlus in regel 130 duurt, afhankelijk van de variabele 'tempo', tussen 50 en 350 eenheden. Dit programma wordt steeds weer herhaald, tot het met de toets 'ESC' wordt afgebroken.

INSTR

Vaak is het gewenst, om een string naar zijn inhoud te onderzoeken, in het bijzonder bij een bestandverwerking. Ook hier geven we eerst een omschrijving van de funktie die dit mogelijk moet maken:

Probleem:	Zoeken naar een string in een andere string.
Funktie:	INSTR(pos,string1,string2)
Parameter:	pos – positie van waaruit gezocht moet worden. string 1 – string waarin gezocht moet worden. string 2 – string waarnaar gezocht moet worden.
Voorbeeld:	PRINT INSTR(3,"ABCDEFG","FGH") Zoekt in de eerst genoemde string vanaf positie 3 naar de laatst genoemde string en deelt de uitkomst van de positie mee (6).
Opmerking:	Als de positie niet gevonden wordt, ontstaat de waarde 0. De variabele 'pos' kan worden weggelaten als de gehele string moet worden onderzocht.

Deze comfortabele funktie ondersteunt het zoeken naar bepaalde data in een bestand. Tot nu toe konden we alleen maar strings vanaf een bepaalde positie onderzoeken, maar met het commando INSTR wordt de gehele string op een bepaalde tekenvolgorde onderzocht. Een programma zal ook hier duidelijkheid verschaffen:

```
 5 CLS
10 INPUT "TEKST" ";tekst$
20 INPUT "ZOEKBEGRIP";zoek$
30 p=INSTR(tekst$,zoek$)
40 IF p=0 THEN 70
50 PRINT "HET ZOEKBEGRIP BEVINDT ZICH OP DE";p;"PLAATS IN DE TEKST"
60 GOTO 80
70 PRINT "ZOEKBEGRIP NIET GEVONDEN"
80 PRINT "ZOEKEN BEËINDIGD."
```

Hier geeft U eerst de tekst in en aansluitend het zoekbegrip. De INSTR-funktie in regel 30 slaat de uitkomst van het zoeken op, in 'pos'. Er werd hier geen beginpositie meegedeeld omdat we de hele string 'tekst$' willen doorzoeken. Alleen als de uitkomst van het zoeken, dus de variabele 'pos', ongelijk is aan nul, is 'zoek$' ook gevonden. De positie die hierbij hoort, wordt dan meegedeeld.

Als afsluiting krijgt U nog een programma waarmee het mogelijk is om een gevonden tekst door een andere te vervangen:

```
 5 CLS
10 INPUT "tekst:";tekst$
20 INPUT "zoeken naar:";zoek$
30 INPUT "vervangen door:";verv$
40 p=INSTR(tekst$,zoek$)
50 IF p=0 THEN 90
60     tekst$=LEFT$(tekst$,p-1)+verv$+MID$(tekst$,p      +LEN(zoek$),LEN(tekst$)-
LEN(zoek$)-(p-1)
70 PRINT "NIEUWE TEKST:    ";tekst$
80 GOTO 100
90 PRINT "NIET GEVONDEN"
100 PRINT "ZOEKEN BEËINDIGD"
```

Regel 60 zal U misschien in verwarring brengen, omdat er hier een groot aantal stringfunkties zijn ondergebracht. Dit is nodig omdat de nieuwe string uit de volgende delen moet worden opgebouwd:

1. Het deel dat voor de plaats staat van de gevonden data-regel (p-1).
2. De nieuwe tekst (verv$).
3. De rest, te beginnen bij de positie achter het zoekbegrip (p+LEN(zoek$) tot het einde van de tekst (LEN(tekst4)-LEN(zoek$)-(p-1).

Een dergelijke routine om teksten te zoeken en dan te kunnen veranderen, vindt U in elke tekstverwerker. Met de CPC464 kan dit op een heel eenvoudige manier gerealiseerd worden.

Appendix

GERESERVEERDE WOORDEN

Men kan de namen van de variabelen weliswaar vrij kiezen, maar er zijn toch een aantal beperkingen. Alle woorden die de taal BASIC zelf nodig heeft, mag men niet gebruiken. Als U zich niet aan deze regel houdt, zal de CPC464 U daar wel aan herinneren met een 'SYNTAX ERROR'. Als U deze foutmelding in een BASIC regel ziet verschijnen, en U kunt dit van geen kant verklaren, kijk dan eerst eens naar de namen van de variabelen. Vaak zijn woorden uit de volgende lijst er de oorzaak van.

ABS
AFTER
AND
ASC
ATN
AUTO

BIN$
BORDER

CALL
CAT
CHAIN
CHR$
CINT
CLEAR
CLG
CLOSEIN
CLOSEOUT
CLS
CONT
COS
CREAL

DATA
DEF
DEFINT
DEFREAL
DEFSTR
DEG
DELETE
DEG
DELETE
DI
DIM
DRAW
DRWAR

EDIT
EI
ELSE
END
ENT
ENV
EOF
ERASE
ERL
ERR
ERROR
EVERY
EXP

FIX
FN
FOR
FRE

GOSUB
GOTO

HEX$
HIMEM

IF
INK
INKEY
INKEY$
INP
INPUT
INSTR
INT

JOY

KEY

LEFT$
LEN
LET
LINE
LIST
LOAD
LOCATE
LOG
LOG10
LOWER$

MAX
MEMORY
MERGE
MID$
MIN
MOD
MODE
MOVE
MOVER

NEXT
NEW
NOT

ON
ON BREAK
ON ERROR GOTO
ON SQ
OPENIN
OPENOUT
OR
ORIGIN
OUT
PAPER
PEEK
PEN
PI
PLOT
PLOTR
POKE
POS
PRINT

RAD
RANDOMIZE
READ
RELEASE
REM
REMAIN
RENUM
RESTORE
RESUME
RETURN
RIGHT$
RND
ROUND
RUN

SAVE
SGN
SIN
SOUND
SPACES
SPC
SPEED
SQ
SQR
STEP
STOP
STR$
STRING$
SWAP
SYMBOL

TAB
TAG
TAGOFF
TAN
TEST
TESTR
THEN
TIME
TO
TROFF
TRON

UNT
UPPER$
USING

VAL
VPOS

WAIT
WEND
WHILE
WEDTH
WENDOW
WRITE

XOR
XPOS

YPOS

ZONE

FOUTMELDINGEN

Hoe vreemd het ook lijkt: foutmeldingen zijn voor een computer net zo belangrijk als de commando's. Hoe meer foutmeldingen een computer geeft, des te eenvoudiger is het voor de gebruiker om de fout te kunnen lokaliseren. Stelt U zich eens voor dat de computer op elke fout zou reageren met 'SYNTAX ERROR'. U zou dan de fouten zonder veel moeite kunnen opsporen. Omdat de CPC464 veel fouten kan onderscheiden worden deze nu beschreven. U kunt dit dan altijd opzoeken als een foutmelding U niet duidelijk is.

Het getal dat voor de foutmelding staat, is voor het opvangen van een fout in een programma (ON ERROR GOTO) nodig. Dit commando is in dit boek niet beschreven, zodat U dit getal eerst later zult kunnen gebruiken.

1. Unexpected NEXT
Er wordt een 'NEXT' in een programma gevonden waar niet een 'FOR aan vooraf is gegaan, of de 'NEXT' variabele is niet in overeenstemming met de 'FOR' variabele.

2. Syntax ERROR
Deze foutmelding zal U niet meer onbekend zijn. Zij ontstaat altijd als de computer de ingevoerde regel niet begrijpt.

3. Unexpected RETURN
Het programma wordt bij een RETURN beëindigd omdat er geen GOSUB aan vooraf is gegaan.

4. Data exhausted
Met het commando 'READ' moeten er meer DATA worden ingelezen dan er in de DATA regels staan.

5. Improper argument
Dit is een algemene foutmelding die altijd optreedt, als de waarde van een funktie of een parameter niet geldig is.

6. Overflow
De uitkomst van een berekening is groter dan in het geheugen kan worden opgeslagen, of een integere variabele kan niet als een 16-bits getal worden gecodeerd.

7. Memory full
Het programma of de variabelen hebben een grootte die boven het ter beschikking staande geheugen uitstijgt.
De MEMORY-opdracht veroorzaakt deze foutmelding als men probeert het begin van BASIC naar een plaats te brengen die niet is toegestaan.

8. Line does not exist
Het regelnummer komt niet in het programma voor.

9. Subscript out of range
De index van een array is te klein of groter dan met DIM gereserveerd is.

10. Array alleready dimensioned
Een array is reeds eerder gedimensioneerd.

11. Division by zero
Er wordt geprobeerd om de rekenkundig niet toegestane deling door nul uit te voeren.

12. Invalid direct command
Er wordt getracht om een commando in de direct-mode te geven, die men alleen maar in een programma mag gebruiken (bv. INPUT).

13. Type mismatch
Er wordt een numerieke waarde gegeven als er een string wordt verwacht of omgekeerd.

14. String space full
Er is geen ruimte meer voor strings.

15. String too long
De grens van een string van 255 tekens werd bv. door het optellen van strings overschreden.

16. String expression too long
Door gebruikmaking van de vele stringuitdrukkingen werden er strings tussen geschoven. Als de geheugenruimte dit niet toestaat verschijnt deze foutmelding.

17. Cannot continue
Het programma kan met het commando CONT niet worden voortgezet.

18. Unknown user function
Een funktie (FN) werd niet met DEF FN gedefiniëerd.

19 RESUME missing
Het programma werd in een ON ERROR GOTO routine afgebroken.

20. Unexpected RESUME
Een RESUME kan niet worden uitgevoerd, omdat het niet met ON ERROR GOTO werd opgeroepen.

21. Direct command found
Bij het inlezen van een programma werd een regel gevonden zonder regelnummer.

22. Operand missing
Een noodzakelijke parameter werd niet aangegeven.

23. Line too long
Een BASIC regel overschrijdt de maximale grootte (255 tekens).

24. EOF met
Er wordt getracht om een bestand over het einde heen te lezen.

25. File type error
Een programma of een bestand wordt op de verkeerde manier ingelezen. Zo kan men bv. een bestand niet met LOAD inlezen.

26. Next missing
Er is een FOR-lus zonder het bijbehorende NEXT geprogrammeerd.

27. File already open
Men probeert een bestand te openen dat niet met CLOSE is gesloten.

28. Unknown command
Er is een commando meegedeeld dat niet in de woordenschat van de CPC voorkomt.

29. WEND missing
Een WHILE-lus werd zonder het bijbehorende WEND ingezet.

30. Unexpected WEND
Het programma komt een WEND tegen waar geen bijbehorende WHIL-lus aan vooraf is gegaan.

Register

R

S

T

V

W

Het beginnersboek voor de CPC-computers zou eigenlijk voor iedereen daadwerkelijk het eerste boek bij de CPC-computers moeten zijn. Dit boek is opgezet als een eenvoudig te hanteren naslagwerk dat het werken met deze computer heel toegankelijk en plezierig maakt.

Uit de inhoud:
* de bediening van de editor en het toetsenbord * het eerste commando (print) * het eerste programma * BASIC stap voor stap; tegelijkertijd wordt zo een adressenbestand gemaakt * GRAFISCHE commando's van de CPC-computers * 'sound'-commando's en het gebruik; dit commando wordt onder andere verduidelijkt door het programma 'orgel' * het werken met en de werking van de ingebouwde cassetterecorder * verdere nuttige mogelijkheden van deze veelzijdige computer.

ISBN 90 229 3356 3

CPC Bibliotheek 9

DATA BECKER
NEDERLANDS*