H.C. de Heer

MSX Basic voor kinderen

deel 1



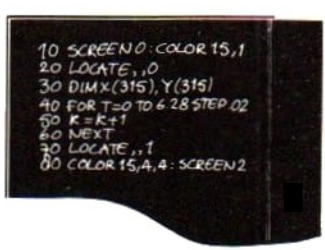
MSX Basic voor kinderen *deel 1* H.C. de Heer

MSX Basic voor kinderen

MSX Basic voor kinderen

deel 1

H.C. de Heer



uitgeverij STARK-TEXEL

postbus 302 - 1794 ZG Oosterend tel. 02223 - 661

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Heer, H.C. de

MSX Basic voor kinderen / H.C. de Heer. – Oosterend: Stark-Texel
ISBN 90 6398 084 1 (deel 1)
SISO 365.3 UDC 681.3.06
Trefw.: MSX-Basic (programmeertaal).

1e druk 1986
ISBN 90 6398 084 1

© uitgeverij Stark-Texel, Oosterend Nh.

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

No part of this book may be reproduced in any form, by print, photoprint, microfilm or any other means without written permission from the publisher.

Ondanks alle aan de samenstelling van de tekst bestede zorg kan noch de redaktie noch de uitgever aansprakelijkheid aanvaarden voor eventuele schade die zou kunnen voortvloeien uit enige fout die in deze uitgave zou kunnen voorkomen.

MSX is een handelsmerk van Microsoft

Inhoudsopgave

onderwerp	pagina
Voorwoord of inleiding	9
De eerste stappen naar je eigen programma	11
Wat is Basic en vooral wat is MSX Basic	12
En toen kwam MSX	13
Hoe werkt zo'n programma?	17
Kennismaken met het toetsenbord en het scherm	19
Je eigen programma's bewaren op cassette	33
Het gebruik van de FUNKTIE-toetsen	37
De rekenleraar	39
Variabelen	53
Het programma mooier maken	60
Het kan nog mooier	62
Een rekenprogramma	65
Schermindelingen	67
De FOR-NEXT loop	74
PRINT USING	81
Het rekenprogramma	94
Foutmeldingen	101
Belangrijke foutmeldingen op alfabetische volgorde	102
Waar staat wat en wat betekent het?	105

Voorwoord of inleiding

De bedoeling van dit boek is, je te vertellen wat programmeren inhoudt, hoe je een programma kunt begrijpen, maar ook hoe je zelf kunt leren programmeren.

Want dat je dat kunt is zeker.
Uiteraard moet er wel wat geleerd worden maar het blijft beslist eenvoudig te begrijpen.

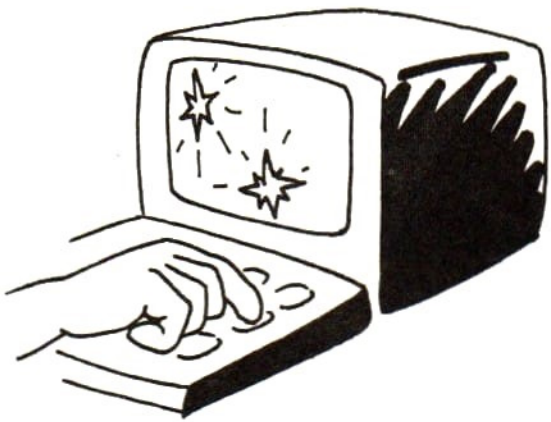
Alle onderdelen van de programmeer-technieken die je niet beslist nodig hebt, worden dan ook niet behandeld.

We laten je eerst kennis maken met de computer, vervolgens met zijn taal.
Daarna volgt een voorzichtig begin met het programmeren, en als je nu nog niet weet wat dat betekent kun je beter maar meteen beginnen te lezen.

Veel succes.

De eerste stappen naar je eigen programma

Je hebt vast wel eens vol verbazing gekeken hoe je vader of moeder bijna toverkunsten uithaalden op hun computer.
Als ze even wat ingetypt hadden en op de return knop drukten gebeurde er van alles en nog wat op het beeldscherm.



Misschien denk je dat dat voor jou nog veel te moeilijk is, maar dat is beslist niet waar. Heus, spelletjes op de computer doen is leuk, maar je eerste echte eigen programma maken is natuurlijk ook wel iets.
Daarmee gaan we dus nu beginnen, je eerste stappen op weg naar programmeren.

Om met een computer aan de gang te kunnen moet je eerst weten wat een computer eigenlijk is en wat hij allemaal kan. Nu, om met het tweede te beginnen, hij kan uit zichzelf helemaal niets. Zo, dat weet je dan ook weer, daar heb je dus niets aan.

Gewoon maar weer gaan spelen dan, of kan hij toch wel wat? Het lijkt er wel op, anders had je vader of moeder of je vriendje er toch zeker niet één gekocht?
Jazeker!

Je kunt hem namelijk heel wat laten doen door hem opdrachten te geven.

Want opdrachten uitvoeren kan een computer namelijk wel en hij doet dat ook heel goed. Alles wat je hem opdraagt doet hij niet alleen perfekt, hij doet het ook zo vaak als je wilt.

Alleen, je kunt een computer natuurlijk niets vertellen. Hij heeft tenslotte geen oren.
Maar hij heeft wél een toetsenbord.

Door nu op het toetsenbord bepaalde woorden in te typen kun je met de computer 'praten' en als de computer wat tegen jou wil zeggen doet hij dat via het beeldscherm.

Wat gebeurt er dan?

We 'praten' met de computer met behulp van een speciale taal die we computertaal noemen. Eén zo'n taal is BASIC. Onze MSX-computer verstaat echter een speciale versie van dat BASIC, dat we dan ook MSX-BASIC noemen.

Wat is BASIC en vooral wat is MSX-BASIC?

Om met het eerste te beginnen, BASIC is dus een taal waarmee je met de computer kunt praten.

Die taal bestaat uit een aantal woorden, statements, waarmee jij de computer bepaalde dingen kunt laten doen of waarmee de computer jou dingen kan laten zien of doen.

Echter, toen er computers gemaakt werden, ontstond er voor elk merk computer een eigen BASIC-taal.
Hierdoor konden programma's van de ene computer niet

door de computers van het andere merk worden gebruikt.

Je kunt dit een beetje vergelijken met de talen in de wereld. Een kind dat in Nederland is opgegroeid, verstaat een kind uit Spanje bijvoorbeeld niet.

Met die computers was dat dus eerst ook zo.
Een computer uit fabriek A leerde A-BASIC, de computer uit fabriek B leerde B-BASIC.

En toen kwam MSX

Een heleboel knappe koppen zeiden: „Dit kan zo niet langer doorgaan, de mensen met computer merk A moeten hun programma's ook op merk B kunnen laten werken en andersom natuurlijk. Daar gaan we wat aan doen."
En dat deden ze.

Alle BASIC-woorden werden heel goed bekeken.
De mooie werden gebruikt, de moeilijke werden gemakkelijker gemaakt, de vervelende verwijderd, kortom wat overbleef was een heel eenvoudige maar sterke taal die MSX-BASIC werd genoemd.

Dat was niet alles wat die heren besloten. Ze spraken ook nog het volgende af.

Iedere fabrikant die een computer wil maken volgens de MSX-norm, want zo heet dat, mag dat alleen doen als hij aan een hele lijst van verplichtingen voldoet.

Een van die verplichtingen is natuurlijk dat de computer het MSX-BASIC moet verstaan.
Maar ook het toetsenbord moet op een bepaalde manier worden opgebouwd.
Kijk maar eens naar de verschillende MSX-computers.

Ze hebben allemaal rechts naast de lettertoetsen die mooie pijltoetsen zitten, en boven de cijfertoetsen een aantal zoge-

naamde funktie-toetsen.
Ook alle aansluitingen voor bijvoorbeeld printers, joysticks en dergelijke moeten op een bepaalde manier worden gemaakt.

Zo kan jij dus op iedere MSX-computer van welk merk dan ook net zo gemakkelijk werken als op je eigen computer.

Voldoet een computer niet aan al de eisen die de heren aan MSX hebben gesteld, dan mag zo'n computer niet MSX worden genoemd.

Zo, dat weet je dus ook weer over je computer.
Nu weet je alleen nog niet precies wat MSX-BASIC is. Zoals gezegd, er zijn een heleboel woorden verwijderd of vervangen door andere, maar waarom?

Wel, de computer is nu zo gemaakt, dat hij met een paar woorden al heel wat kan doen.
Hij heeft aan een half woord genoeg, zou je kunnen zeggen.

Als je pas begint met programmeren is dat nog niet zo erg belangrijk, maar wanneer je programma's langer worden is elk woord dat je niet hoeft te typen mooi meegenomen.

Misschien heb je nog nooit met een andere computer gewerkt, dan valt het je ook niet op hoeveel eenvoudiger het programmeren met MSX eigenlijk is.

Hoe dan ook, we gaan gewoon iets meer vertellen over het MSX-BASIC.

Zoals gezegd, onze computer verstaat alleen maar de taal MSX-BASIC. Maar wat houdt dat in?

Die taal bestaat uit ongeveer 150 woorden, die elk een heel duidelijke betekenis hebben en waarmee je de computer dus van alles kunt laten doen.
Waarom maar die paar woorden terwijl wij er wel duizend kennen?

In de eerste plaats kent en kan de computer een heleboel dingen toch niet, dus woorden als lopen, vliegen, schreeuwen enzovoort hebben totaal geen zin.

Maar bijvoorbeeld wel schrijf, kleur, teken enzovoort.
Ook hoef je de computer niets beleefd te vragen.
Wanneer ik bijvoorbeeld tegen jou zeg: „Wil je even op een briefje schrijven het woord "computer" dan zijn dat nogal wat woorden. De computer heeft genoeg aan PRINT "computer".

Trouwens, zie je het bijzondere aan dat woord PRINT?
PRINT is een engels woord, net als die andere BASIC-woorden.

Dat is echt niet zo erg hoor, tenslotte zijn het er maar een goede 150, waarvan we er niet meer dan vijftig gebruiken, voorlopig.

Met die vijftig woorden kunnen we al aardig uit de voeten en zo leer je meteen wat engelse woorden.

Goed, met het commando

```
PRINT "computer"
```

schrijft de computer dus netjes 'computer' op het scherm. Dat zijn precies de woorden die tussen de aanhalingstekens staan.

En dat kan natuurlijk van alles zijn. Dat is geen BASIC. Dus,

```
PRINT "mijn naam is haas"
```

mag je best schrijven, maar

```
SCHRIJF "mijn naam is haas"
```

kan niet natuurlijk want 'schrijf' is geen BASIC.

Ook typfouten mag je tussen de aanhalingstekens zoveel maken als je zelf wilt, maar niet in de BASIC-woorden.

Dus,

```
PRINT "mein nederlans is tog wel goet"
```

vindt de computer best, maar bij

```
PRONT "dit is een test"
```

zal de computer hevig protesteren.



Als je deze zinnetjes eens op je computer wilt uitproberen kan dat wel, maar lees dan eerst het hoofdstuk over het gebruik van het toetsenbord.

Die vijftig woorden, waar we het zojuist over hadden, gaan we niet domweg uit ons hoofd leren.

In de loop van het boek komen ze vanzelf ter sprake. Dan worden ze ook uitvoerig besproken.

Nu weet je een beetje hoe jij de computer dingen vertelt. Wanneer je hem zegt dat er iets moet worden gedaan, geven we hem eigenlijk een opdracht.

Zo heet dat dan ook. Een opdracht of commando.
Wanneer je die opdracht geeft zoals in ons voorbeeld, dus

```
PRINT "dit is een opdracht"
```

wordt die opdracht direkt door de computer uitgevoerd.
Zo'n opdracht heet dan ook een *direkte opdracht* of *direkt commando.*

Net zoiets als, wanneer je opdracht krijgt van de leraar een som op het bord te schrijven.

Wanneer we meer dan één opdracht krijgen en wel in de vorm van een lijstje, bijvoorbeeld je huiswerk voor de komende week, dan noemen we dat een serie opdrachten.

Als we de computer een serie opdrachten geven dan heet dat een programma.

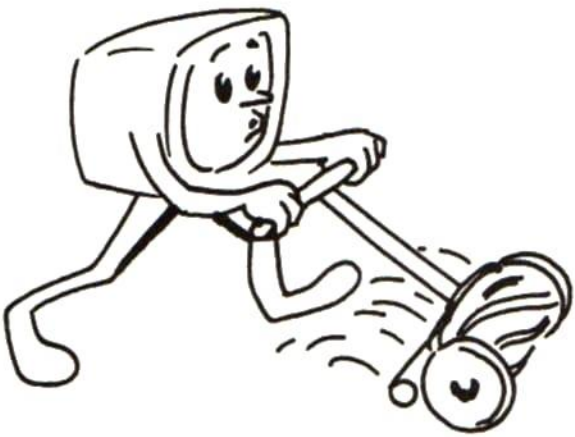
Hoe werkt zo'n programma?

Zoals gezegd, die opdrachten worden verzameld tot een serie opdrachten die elkaar netjes opvolgen.

Ze worden ook in de vorm van een lijstje in de computer ingetypt en wel, met een nummer voor elke opdracht.
Net als een lijstje met karweitjes bijvoorbeeld.

1. Maai het gras.
2. Ruim het losse gras op.
3. Berg de grasmaaier op.
   enz.

De computer zal dat ingetypte lijstje wel in zich opnemen maar er pas wat mee gaan doen als hij daarvoor apart opdracht krijgt.
En dat gebeurt dan... juist, met een direkt commando.



Voorlopig bewaart hij het programma alleen nog maar in zijn geheugen. (Heel voorzichtig mag je dat wel vergelijken met jouw geheugen.)

Wanneer jij het commando geeft om het programma te starten zal de computer het lijstje netjes regel voor regel gaan afwerken tot het einde.

Maar nu even genoeg gepraat. Je wilt nu vast weleens wat met je computer gaan doen, ja toch?

We beginnen natuurlijk met het aanzetten van de computer. Na enige tijd verschijnt er dan wat tekst op het scherm, het woord 'Ok' en een blokje.

De computer is kennelijk al heel tevreden over zichzelf omdat hij 'Ok' zegt.
Dat mag ook wel want er is al heel wat gebeurd.
Toen je hem aanzette heeft de computer al heel wat werk



verricht en door de 'Ok' melding en het blokje, (de cursor) laat de computer weten dat hij daarmee klaar is en jouw opdrachten wil uitvoeren.

We kunnen dus nu gaan beginnen en hem iets voor ons laten doen, maar hoe?
Dat is nog even moeilijk.
Laten we eerst maar eens kijken hoe alles zo'n beetje werkt.
Er zitten nogal wat toetsen op dat apparaat vind je niet?
Een beetje oefenen met het toetsenbord kan vast geen kwaad.
Dus, op naar het volgende hoofdstuk dan maar.

Kennismaken met het toetsenbord en het scherm

Dat blokje waar we het zojuist over hadden heet de cursor, of wel, vrij vertaald, het aanwijsblokje.

Hiermee geeft de computer aan waar een door jou getypte letter komt te staan.



Typ je naam maar eens in.

Zie je dat het precies zo werkt als een schrijfmachine? Met de verschillende toetsen kun je de cursor overal heen sturen. Druk maar eens op de verschillende pijltoetsen.
Zo kun je op elk plekje van het scherm schrijven. Eigenlijk zinloos voor de computer maar doe dit maar eens om jezelf wat bekend te maken met het toetsenbord.

Staat er nu door al je getyp wat erg veel onzin, druk dan op de SHIFT-toets en houd deze ingedrukt. Druk dan op de toets waarop staat 'CLS/HOME'. Zo, weer een schoon scherm.

Nu een andere oefening.

Typ eens precies in:

```
dit is eend stkjetekst
```

Ja echt, met al die fouten.
Die gaan we dus eventjes verbeteren.

Als het goed is staat de cursor netjes te wachten achter de 't' van 'tekst'.

Druk nu net zolang op de pijltoets die naar links wijst tot de cursor op de 'd' van 'eend' staat, want die moet weg.
Merk je trouwens dat de cursor doorloopt als je de pijltoets ingedrukt houdt?
Dit herhalen geldt uiteraard ook voor alle andere toetsen.
Dus, even indrukken geeft een aanslag, ingedrukt houden laat de toetsaanslag herhalen.

Goed, de cursor staat dus op de 'd' van 'eend'.
Druk nu op de DEL-toets.
Zoals je ziet verdwijnt de 'd' niet alleen, de tekst achter de cursor schuift ook netjes een stap terug.

Nu op naar de volgende fout.

In het woord 'stukje' ontbreekt de 'u'. Die gaan we er even tussen stoppen.
Zet de cursor op de 'k' van 'stkje' en druk op de INS-toets. De cursor zal nu kleiner worden wat betekent dat hij alles wat je nu typt tussen de 't' en de 'k' zal zetten. Typ nu de letter 'u'. Zie je wel? Druk nu op de RETURN-toets en ga dan met de pijltoets terug.

Probeer dit nu zelf met de ruimte die ontbreekt tussen de twee woorden 'stukje' en 'tekst'.

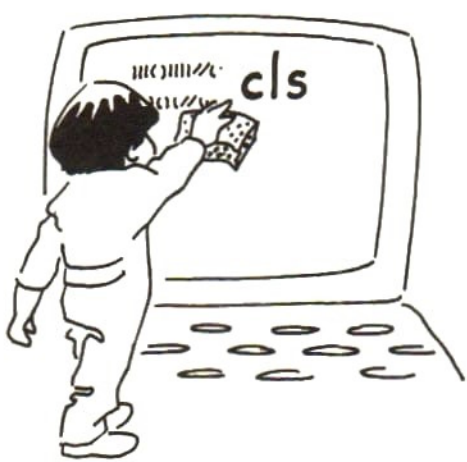
Er is nog een toets die je goed kunt gebruiken bij het verbeteren van woorden of zinnen, dat is de BS-toets.

'BS' betekent 'back space' en dat betekent dat je hiermee spaties typt, maar dan achteruit.

Typ maar eens in:

```
dit is een stukje tekstdefdder
```

De cursor staat nu achter de 'r' van dat gekke woord.
Druk nu net zolang op de BS-toets tot je het woord tekst weer netjes hebt.



Zo, oefen nog maar wat met het intypen en verbeteren van woorden, net zo lang tot je het zonder al te veel nadenken kunt doen.

Je weet toch nog, wanneer er teveel op het scherm staat, dat je het met de SHIFT-toets en CLS/HOME schoon krijgt.

Ook kan het zijn dat wanneer je op de RETURN-toets drukt, de computer het niet met je eens is en één of andere kreet op het scherm zet, bijvoorbeeld SYNTAX ERROR'. Daar moet je je nog niets van aantrekken. Gewoon SHIFT-CLS/HOME en alles is weer in orde.

Laten we maar eens een voorzichtig begin maken met de BASIC-woorden.
Ga je mee naar het volgende hoofdstuk?

De eerste BASIC sleutelwoorden

Typ maar eens in:

```
CLS
```

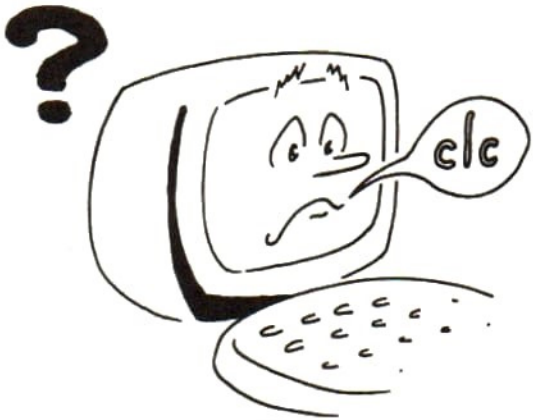
Er gebeurt nu nog niets, maar druk je op de grote toets waarop staat "RETURN" dan wordt, als alles goed gaat, het scherm schoongemaakt.

Typ nu in:

```
CLC
```

en druk op return.
Nu verschijnt de tekst "syntax error", wat zoveel wil zeggen als "wat bedoel je nu toch?'

De computer kent namelijk alleen de zogenaamde BASIC-sleutelwoorden en verder niets.
Wij zouden zeggen 'CLC' is vast een typfout, hij zal wel 'CLS' bedoelen, maar dat kan een computer niet.



Even opnieuw nu, typ:

```
CLS
```

en druk op return.

Wat gebeurde er nu eigenlijk allemaal?

Wel, je zei tegen de computer 'CLS' en er gebeurde niets. Daarna tikte je op de return toets. Toen maakte hij het scherm pas schoon.

'CLS' betekent 'clear screen' en dat is engels voor 'maak het scherm schoon'

Dat scherm schoonmaken, dat wil hij wel doen, maar hij weet niet dat jij al klaar bent met typen, dus doet hij nog niets.

Net zoals wanneer je moeder zegt: „Ga even naar de bakker", dan ga je natuurlijk niet meteen want je moeder is vast nog niet uitgepraat.

Je wacht dus tot je de volledige opdracht gehoord hebt. Die luidt bijvoorbeeld „Ga even naar de bakker en haal wat brood". Jij hebt genoeg verstand om te weten wanneer je

kunt gaan, dat is in dit geval wanneer je weet waarheen je moet gaan en wat je daar moet doen.

Dat geldt niet voor de computer, die weet dat niet en die moeten we vertellen wanneer we klaar zijn met ons verhaal en dat we nu iets van hem verwachten.
Dat doen we met behulp van de return-toets.

Inmiddels is dus het scherm schoon en staat de cursor weer trouw te wachten tot er iets gebeurt.

O ja, onderin het beeld staat nog steeds wat tekst.
Ook maar even wegdoen, vind je niet?

Typ maar:

```
KEY OFF
```

en druk op return.

Zo, daar zijn we voorlopig even vanaf.
Op dat commando "KEY OFF" komen we bij het behandelen van de zogenaamde FUNKTIE-toetsen nog terug.

Die opdracht trouwens die we zojuist gaven heet een direkt commando, omdat de computer deze opdracht direkt uitvoert, dat wil zeggen na return natuurlijk.

We kunnen de computer echter ook opdrachten geven die hij later pas mag uitvoeren, weet je nog?
Een serie van dat soort opdrachten heet een programma. De computer zal daaraan pas beginnen als we het hele programma hebben ingetypt en het commando 'RUN' gegeven hebben.
Moeilijk? Welnee, ik geef je een voorbeeld.

Typ maar eens:

```
PRINT "ik ben een computer"
```

Druk op return en je zult zien dat de computer het korte zinnetje "ik ben een computer" netjes neerzet op het scherm. Typ nu in:

```
10 PRINT "ik ben een computer"
```

Druk op return en je zult zien... nee hè, geen tekst op het scherm. De computer doet niets.
Door dat nummer voor het PRINT-commando ziet de computer deze regel niet als een direkte opdracht, maar zal hij deze regel alleen in zijn geheugen opnemen.

Je bent nu dus bezig een programma te maken en dat moet pas worden uitgevoerd als je klaar bent met het hele programma.
Typ nu in:

```
5 CLS
```

en druk op return.
Je hebt nu weer wat geleerd: 'PRINT' is engels voor 'schrijven' en 'RUN' betekent 'start'.
Er komt nog een commando bij namelijk 'LIST'. Typ maar eens in:

```
LIST
```

en druk op return.
Je ziet nu, en misschien tot je verbazing:

```
5 CLS
10 PRINT "ik ben een computer"
```

maar, we hadden het toch zo ingetypt?

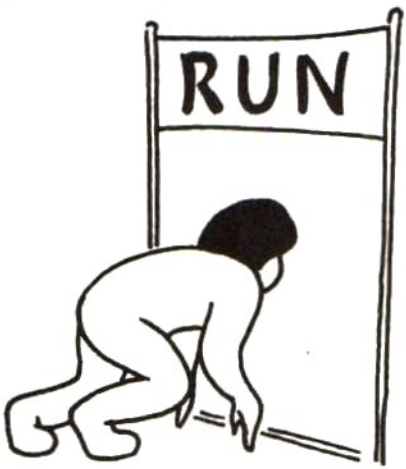
```
10 PRINT "ik ben een computer
5 CLS
```

Dat klopt, maar de computer houdt van orde en netheid en

zet alle regels dus op een rijtje.
Dat moest ook wel, want je wilt de zin 'ik ben een computer' toch op een schoon scherm zetten.

LIST betekent zoiets als 'lijst' ofwel, laat het programma maar eens zien.

Die nummers voor die zinnetjes zijn de regelnummers. Welke nummers je gebruikt is niet zo erg belangrijk als je er maar rekening mee houdt dat de computer bij het begin begint zodra je 'RUN' typt.



Bedenk wel als je nummert 1, 2, 3, 4 enz. en later blijkt dat er nog een regel tussen moet, dat het dan toch lastig wordt. Niet echt hoor, want je hebt daar ook heel slimme oplossingen voor.

Typ maar eens:

```
RENUM
```

en druk op return.

'RENUM' betekent 'hernummeren' en dat doet de computer, wanneer we achter renum verder niets zetten, met stappen van 10.

De computer laat het programma nu netjes beginnen bij 10 en telt er steeds 10 bij op. Nu staat er dus:

```
10 CLS
20 PRINT "ik ben een computer"
```

Het is dus beter om je programma zelf zo te nummeren. Wanneer je dan ontdekt dat er nog een regel tussen moet, kan dat gemakkelijk.

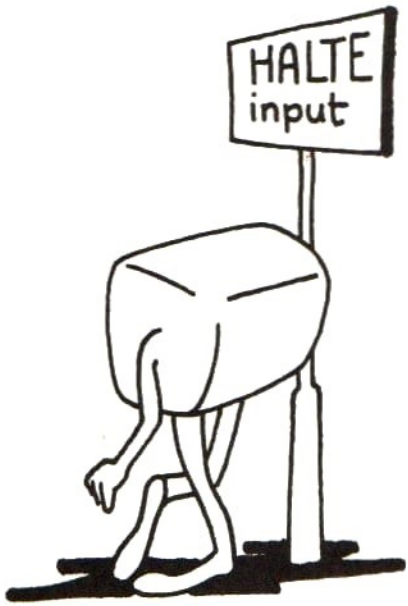
Laten we hiermee maar eens een programma maken

Stel dat je de computer een som wilt laten maken en wij geven hem de getallen voor deze som.
Dan zal hij eerst de getallen moeten krijgen, daarna uitrekenen en vervolgens het antwoord op het scherm zetten.

Daarvoor hebben we echter wel een nieuw commando nodig en wel: "INPUT" wat "Invoer" betekent en dat is een heel ander commando dan we gewend zijn.
Want, als je denkt dat de computer steeds doet wat we zeggen, doet de computer nu juist eigenlijk niets. Integendeel, hij verwacht nu iets van ons. Hij gaat namelijk net zo lang staan wachten tot we iets intypen.
Wat dat is bepalen we zelf in ons programma.

Alleen het woord INPUT is niet genoeg. Achter dat woord komen een of meer letters, waarin we de in te voeren getallen gaan bewaren. Die letter(s) noemen we een variabele.
Dat heet zo, omdat er iedere keer een ander getal in kan komen. (Variabel is afwisselend.)

Hoe moeten we ons dat voorstellen?



Wanneer je een of meer letters kiest voor een variabele, zal de computer deze de waarde 'Ø' geven.

Typ maar eens:

```
PRINT "a=";A
```

en druk op return.
De computer zal inderdaad 'a= 0' op het scherm zetten.
Probeer hetzelfde maar eens met:

```
PRINT "b=";B
PRINT "xa=";XA
PRINT "d3=";D3
```

Het is ook toegestaan komplete woorden als variabele te gebruiken.
Goede variabelen zijn:

```
antwoord, vraag, cijfer,naam enz.
```

Maar, bedenk wel dat de computer slechts naar de eerste twee tekens kijkt.

Dus, wanneer je als variabelen kiest:

```
antwoord en antenne
```

zal dit, al zien wij een levensgroot verschil, voor de computer hetzelfde zijn, namelijk 'an'.

En dan nog iets, als variabele mag je beslist geen BASIC-woord gebruiken.
De variabelen *formule, letter* enz. mogen dus niet worden gebruikt want hierin zitten de BASIC-woorden 'FOR' en 'LET'.

Zolang je alle BASIC-woorden nog niet kent moet je dus met het gebruik van dit soort variabelen voorzichtig zijn.

We gaan nu eerst weer verder met ons programma.

Typ maar eens onder de twee regels die er al staan:

```
30 PRINT "typ een getal";
40 INPUT A
50 PRINT "je typte getal";A
```

typ "RUN" en druk op return.

Als alles goed gaat komt er nu op het scherm te staan:

```
ik ben een computer
typ een getal?
```

Wat je nu ziet, is ook weer nieuw. Een vraagteken dat je niet had getikt.
Dat komt nu door het commando INPUT.

Hiermee geeft de computer aan dat hij wacht tot jij een getal intypt.

Doe dit maar. Typ bijvoorbeeld 365 en druk daarna op return.

Direkt hierna zal de computer melden:

```
je typte getal 365
```



Zoals je nu misschien wel ziet, hebben we de variabele A gevuld met 365. We noemen dit: A krijgt de waarde 365. Alleen, wat gek, je zegt tegen de computer:

```
PRINT "je typte getal";A
```

en dan komt er te staan:

```
je typte getal 365
```

Waarom niet:

```
je typte getal a
```

Let op, als er iets tussen aanhalingstekens staat, dus:

```
"je typte getal"
```

dan wordt, met het commando "PRINT", die tekst precies op het scherm gezet zoals die tussen de aanhalingstekens

staat.

Als je typt:

```
PRINT A
```

dan wordt de waarde van A op het scherm gezet.
In dit geval dus 365.

Met dit getal kunnen we ook rekenen.
Verander regel 50 maar eens als volgt:

```
50 PRINT "je typte getal";A-5
```

Wanneer je nu weer 365 intypt, zal de computer niet 365, maar 360 op het scherm schrijven, waarom?
De computer zal de waarde A, dus 365, verminderen met 5 en dan pas het resultaat op het scherm zetten.

Uiteraard kun je ook met twee of meer van die variabelen rekenen.

Laten we maar eens wat gaan veranderen en er een klein reken-programmaatje van maken.

Eerst moeten we een regel veranderen.
Dat is met een MSX computer wel erg eenvoudig.
We gaan regel 50 veranderen in:

```
50 PRINT "typ nog een getal";
```

Hoe doe je dat? Heel eenvoudig, typ precies wat hierboven staat, druk op return en hoewel je nu twee verschillende regels "50" hebt getypt, staat de laatste, deze dus, in het geheugen en is de andere regel 50 verdwenen.

List je programma maar eens.

Zo, we hebben nu:

```
10 CLS
20 PRINT "ik ben een computer"
30 PRINT "typ een getal"
40 INPUT A
50 PRINT "typ nog een getal"
```

Voeg daar nu aan toe:

```
60 INPUT B
70 PRINT "de som is";A+B
80 END
```

en laat dit programma maar eens lopen.
Weet je het nog? Typ "RUN" en druk op return.

Regel 70 zal nu ook geen problemen meer geven, of wel?

Nog een paar opmerkingen.
Iedere keer als je een cijfer intypt moet je ook op de return-toets drukken. Waarom?
Dat komt omdat de computer nooit zelf weet wanneer je klaar bent met typen.
Je kunt 7 maar ook 765 of 76583 typen.
Dus pas als je op de return-toets drukt weet de computer dat het ingetypte getal klaar is.

Is het je opgevallen dat de vraagtekens niet onder maar achter de zinnen "typ een getal" en "typ nog een getal" stonden?
Dit komt door de puntkomma(;) achter de regels.
Wanneer je die verandert in een komma, dan zal de computer de vraagtekens op het midden van de regels zetten.
Probeer dat maar eens.
Ook kun je de puntkomma helemaal weglaten, dan zullen de vraagtekens steeds onder de zinnetjes komen te staan. Uiteraard kun je dit allemaal eens oefenen.

Wanneer de computer de laatste zin op het scherm heeft gezet is het programma klaar en zal de computer op regel 80 stoppen.

Steeds, wanneer je een volgende som wilt maken moet je de computer opnieuw starten door het commando RUN.
Dit kan natuurlijk ook anders.
Er zijn alleen zo ontzettend veel manieren dat we maar beginnen met de eenvoudigste.

Weet je nog dat we met het commando "KEY OFF" (KEY betekent sleutel of sleutelwoord) die woorden onder het scherm hebben verwijderd?
Die kunnen we ook terug halen en dan zul je zien dat we op een eenvoudige manier het programma opnieuw kunnen starten.

Typ maar:

```
KEY ON
```



Zo, daar zijn ze weer, want
"OFF" is 'uit' en "ON" is aan.

Zoals je ziet, is het vijfde woord RUN.
Precies wat we nodig hebben.

Die woorden onderaan laten zien wat die grote toetsen (FUNKTIE-toetsen) linksboven op je toetsenbord allemaal voor je doen.
Wanneer je op de vijfde toets drukt zal je programma direkt starten, je hoeft zelfs niet op return te drukken.
We zullen die FUNKTIE-toetsen eens wat beter gaan bekijken.

Maar eerst nog even wat anders.

Je eigen programma's bewaren op cassette

Natuurlijk wil je de programma's die je gemaakt hebt bewaren, maar als je de computer uitzet en daarna weer aan dan blijkt hij ineens alles wat je hebt getypt helemaal vergeten te zijn.

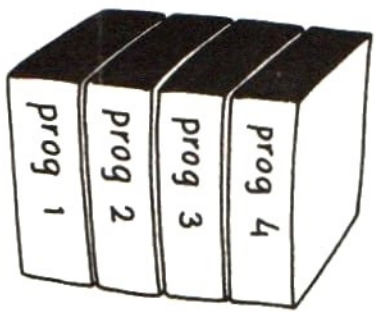
Het is natuurlijk niet de bedoeling dat je elke keer opnieuw

hetzelfde programma moet intypen.

Nee hoor, dat hoeft niet.
Als je namelijk klaar bent met intypen van een programma dan kun je dat programma op een cassettebandje bewaren, met behulp van een cassetterecorder.

Iedereen heeft natuurlijk wel eens met een cassetterecorder een plaatje van de radio opgenomen.

Wel dit is net zo iets.



We hebben eerst natuurlijk een programma nodig. Je kunt hiervoor het rekenprogramma gebruiken.

```
10 CLS
20 PRINT "ik ben een computer"
30 PRINT "typ een getal"
40 INPUT A
50 PRINT "typ nog een getal"
60 INPUT B
70 PRINT "de som is";a+b
80 END
```

Zo, dat programma gaan we op de band zetten.

Hoogstwaarschijnlijk is er al een cassetterecorder aan je computer aangesloten.

Mocht dit niet zo zijn dan staat in de gebruiksaanwijzing van de computer hoe dat moet.

Als je een cassetterecorder hebt met een zogenaamde remote-aansluiting dan zal de computer zelf de recorder starten en stoppen tijdens het opnemen, zo niet, dan moet je dat zelf doen.

Doe een bandje in de cassetterecorder maar denk eraan dat de aanloopstrook van het bandje wel doorgedraaid moet zijn. Dat doe je tenslotte ook als je muziek opneemt.

Zet, als dat kan, je opnameknop op ongeveer driekwart van het totaal, en zet je luidspreker uit.

Typ nu:

```
CSAVE"reken"
```

Druk de toetsen REC en PLAY samen in en druk op return.

Nu gaat als het goed is je bandje lopen. Na enige tijd stopt het bandje en komt het 'Ok' weer op het scherm. Je programma is nu geladen en staat op de band.

Zolang je de computer niet uitzet zit het natuurlijk ook nog in de computer zelf, je kunt dus gewoon nog verder met het programma.

Je kunt ook controleren of je programma wel goed op het bandje staat.
Want stel je voor dat er een klein foutje op het bandje is ontstaan, dan ben je heel je programma kwijt.

Zet je cassetterecorder op REW (dat wil zeggen terugspoelen) en typ:

```
CLOAD?
```

Druk op return en je bandje zal terugspoelen.

'CLOAD' betekent 'laden' maar door het vraagteken erachter wordt het programma niet geladen, maar vergeleken met het bestaande programma.

Als de cassetterecorder teruggespoeld is druk je op PLAY en je bandje gaat weer lopen. Na enige tijd verschijnt er op je beeldscherm:

```
FOUND:reken
```

'FOUND' betekent 'gevonden' en 'reken' was de naam die je zelf gegeven had. Als na nog een tijdje je bandje stopt en er 'Ok' op het scherm verschijnt is je programma goed overgenomen en kun je je computer met een gerust hart uitzetten.

Als het programma niet goed is opgenomen, bijvoorbeeld omdat je het te zacht hebt gedaan, dan komt er:

```
VERIFY ERROR
```

te staan en dat wil zeggen dat er een fout is opgetreden die je moet controleren; waarschijnlijk stond het volume te hard of juist te zacht.
Nu gaan we het programma weer terughalen van het bandje.

Zet eerst je computer uit en dan weer aan.
Typ:

```
CLOAD
```

en zet je recorder op PLAY, druk op return en als het allemaal weer goed gaat verschijnt na enige tijd

```
FOUND:reken
```

Als je weer even later 'Ok' ziet en je recorder stopt, dan zit je programma weer in je computer.

Het gebruik van de FUNKTIE-toetsen

Er zijn verschillende manieren waarop je een FUNKTIE-toets kunt laten werken.

1. Door op de toets te drukken wordt een kompleet woord op het scherm gezet.
   Dit gebeurt bijv. door op F-toets 3 te drukken.
   Het woord GOTO wordt dan geprint. Dit scheelt werk bij het intypen van de programma's.

2. Door op een toets te drukken wordt een commando op het scherm gezet (geprint).
   Een voorbeeld is F-toets 4.
   Hiermee wordt het woord LIST geprint.
   Door er nu een regelnummer achter te zetten en vervolgens op return te drukken wordt dan die regel gelist.

3. Door op een toets te drukken wordt een commando uitgevoerd. Een voorbeeld hiervoor is F-toets 5.
   Hier wordt het commando niet alleen geprint, maar ook uitgevoerd.

Het grote voordeel van deze toetsen is dat je, wanneer je er eenmaal aan gewend bent, er erg veel gemak van kunt hebben.

Als we de computer aanzetten zien we dat de volgende funkties aan de toetsen zijn toebedeeld.

F 1. COLOR	F 6. COLOR 15, 4, 4
F 2. AUTO	F 7. CLOAD'
F 3. GOTO	F 8. CONT
F 4. LIST	F 9. LIST.
F 5. RUN	F10. RUN (+CLS)

Allereerst beginnen we met de FUNKTIE-toetsen 4 en 9, die allebei voor het commando LIST worden gebruikt.

Bespreken F-toets 4 en 9

Wanneer we op F-toets 4 drukken gebeurt er nog niet veel. Alleen het woord LIST wordt uitgeprint. Waarom? Wel, er zijn verschillende mogelijkheden om een programma te listen dus de computer moet nog even weten wat je precies wilt.

Is je programma erg kort, bijvoorbeeld 20 regels, dan kun je direkt 'LIST' typen en op return drukken.

De computer geeft je dan het hele programma in een keer op het beeld.
Heb je een langer programma en je doet hetzelfde, dan zal de computer het hele programma eveneens listen, maar dan zie je het alleen maar snel voorbijgaan, tot hij het einde van het programma vindt.
Daarna stopt hij automatisch zodat je wel altijd de laatste paar regels ziet.
Meestal wil je echter op een andere manier naar het programma kijken. Dat kan.

Er zijn verschillende mogelijkheden.

1. Je weet nog niet waar staat wat je zoekt.
2. Je zoekt een bepaald gedeelte.
3. Je zoekt een regel.
4. Je zoekt de laatste regel waarmee je bezig was.
5. Je zoekt een stuk dat voor een bepaalde regel staat.
6. Je zoekt een stuk dat direkt na een bepaalde regel staat.

1. Als je niet weet wat je zoekt, dan laat je met het commando LIST en de return-toets het hele programma aan je voorbijgaan en druk je op de STOP-toets zodra er een klein gedeelte voorbij is gegaan. Als het gedeelte van het programma dat je zoekt niet op het scherm staat, druk je weer op de STOP-toets, totdat het door jou gezochte deel op het scherm verschijnt.
Als je in dat gedeelte wilt gaan wijzigen, druk dan op de CTRL-toets, houd hem ingedrukt en druk op de STOP-toets. De computer stopt dan met de listing, meldt BREAK en hierna kun je dan in dat gedeelte wijzigen.

2. Wanneer je een bepaald gedeelte zoekt, typ dan F-4 gevolgd door de twee regelnummers waartussen dat gedeelte staat. Dus bijvoorbeeld "LIST 100-160". De computer zal dan alle regels tussen 100 en 160 listen.

3. Wanneer je bijvoorbeeld alleen regel 145 zoekt, typ dan "LIST 145". Alleen regel 145 wordt dan weergegeven.

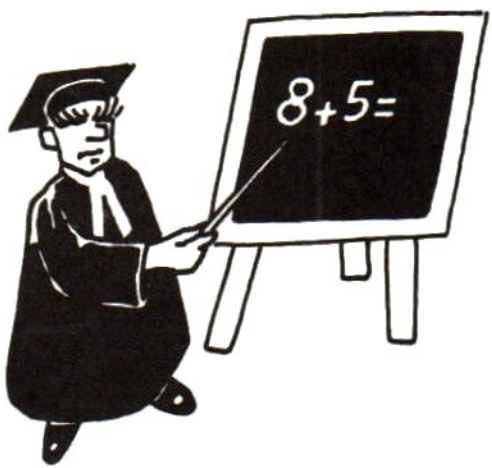
4. Zoek je de laatst behandelde regel, druk dan op de F-9-toets of typ zelf "LIST" (let op de punt).

5. Zoek je iets waarvan je zeker weet dat het voor bijvoorbeeld regel 170 staat, typ dan "LIST -170". De computer zal dan vanaf het begin listen tot regel 170.

6. Staat het daarentegen bijv. direkt na regel 170, typ dan "LIST 170-". De computer zal dan beginnen vanaf regel 170.

De rekenleraar

We hebben nu een programma gemaakt, waarbij de computer ons om twee getallen vroeg en die hij dan voor ons uitrekende, maar dat kan een rekenmachine ook.



Leuker is het natuurlijk, wanneer de computer ons twee getallen opgeeft, die wij dan moeten optellen, waarna de computer het antwoord controleert.

Laten we nu eens proberen zo'n programma te gaan maken.

Typ het eerst maar in en let even niet op de nieuwe commando's, die leggen we je straks uit.

Trouwens, nu we het er toch over hebben, dat woord commando klopt niet helemaal.
Een commando in een programma heet namelijk een statement. Dit woord zullen we dan ook gaan gebruiken, wanneer we het over commando's in een programmaregel hebben.

```
10 CLS
20 PRINT "5+8=";
30 INPUT ANT
40 IF ANT = 13 THEN GOTO 70
50 PRINT "fout"
60 GOTO 80
70 PRINT "goed"
80 END
```

De nieuwe statements.

'IF' betekent 'als' en 'THEN' betekent 'dan'.
Het heeft dezelfde betekenis als wanneer tegen jou gezegd wordt:

Als je je huiswerk af hebt, *dan* mag je naar buiten.
Als de som goed is, *dan* mag je er mee stoppen.

In het programma staat:

Als het ingetypte antwoord 13 is, *dan* mag je naar regel 70 gaan en het woord "goed" op het scherm zetten.

Er is nog een mooi vervolg op dit commando, dat is het

woord ELSE, dat "anders" betekent.

Het complete commando wordt dan:

IF . . . THEN . . . ELSE . . .

Dus *als* het antwoord 13 is, *dan* print je "goed", *anders* print je "fout".

Verander regel 40 maar eens als volgt:

```
40 IF ANT = 13 THEN PRINT "goed" ELSE PRINT
"fout"
```

Als je dan ook nog regel 50 verandert in:

```
50 END
```

Dan kunnen de regels 60 t/m 80 verdwijnen.

Hoe doen we dat?

Dat leren we dan ook maar meteen even, vind je niet?
Ook dit is weer heel eenvoudig met onze MSX-computer.

Hiervoor gebruiken we het commando: "DELETE".
'DELETE' betekent 'verwijderen'.

In ons geval willen we de regels 60, 70 en 80 kwijt.
Daarvoor moeten we dan het volgende direkte commando intypen:

```
DELETE 60-80
```

Druk op return en list je programma maar; ze zijn weg.

Wanneer je slechts één regel wilt verwijderen, kan je intypen:

```
DELETE 70
```

In dat geval is het echter veel eenvoudiger om alleen het regelnummer in te typen, want met:

```
70
```

en return is regel 70 ook verdwenen.

Zo, weer even terug naar het programma.

```
10 CLS
20 PRINT "5+8=";
30 INPUT ANT
40 IF ANT = 13 THEN PRINT "goed" ELSE PRINT
   "fout"
50 END
```

Zoals je ziet is dank zij het commando "ELSE" het programma ook een stuk korter geworden.
Natuurlijk is dat nu nog niet zo erg belangrijk, maar wanneer je langere programma's gaat maken scheelt dat ten eerste een heleboel typwerk, maar ook het geheugen van de computer is nu eenmaal niet zo groot dat je er maar van alles in kunt blijven stoppen.

Hoewel het commando 'GOTO' alweer uit het programma is verdwenen, moeten we het er toch nog wel even over hebben.



'GOTO' is engels voor 'ga naar' (let op GOTO moet als commando aan elkaar worden geschreven dus niet GO TO, maar altijd 'GOTO'). Achter dit commando komt dan een regelnummer waar we de computer naar toe sturen.

Dit is soms nodig wanneer we bijvoorbeeld over een aantal regels willen springen, of terug naar een lager regelnummer. Daarom wordt 'GOTO' ook wel een sprong-opdracht genoemd.

Kijk maar naar regel 40 in ons voorbeeld.
Als het antwoord gelijk is aan 13 dan moet het woord "fout" natuurlijk niet worden geprint.

Het programma zoals we het nu hebben, werkt nu wel, maar met dat ene sommetje kunnen we nog geen kant uit.

Het is de bedoeling dat de computer zelf getallen verzint en die dan aan ons opgeeft.

Ook hiervoor zijn er weer de nodige BASIC-statements die ons daarbij kunnen helpen.

In dit geval worden ze in een formule gezet.
Wanneer we die formule dan in ons programma opnemen, zal de computer hiermee voor steeds wisselende getallen zorgen.

De formule:

```
A = INT (10*RND (1) +1)
```

Hierin zitten twee nieuwe statements.
Waar het om gaat is 'RND', wat betekent 'random' of in goed nederlands 'willekeurig'.
Samen met de (1) achter 'RND' zorgt de computer voor een willekeurig getal, alleen dit is een getal tussen 0 en 1. Maar wel een getal met 14 cijfers achter de komma.

Typ maar eens:

```
PRINT RND(1)
```

Druk op return en je zult getallen zien die er ongeveer zo uitzien.

```
 .59521943994623
 .95434333958382
 .03439384829427
```

Zie je dat er geen nul voor de punt staat.
Bij getallen kleiner dan 1 zal de computer altijd de nul weglaten.
dus 0,3 wordt .3 maar 2,3 blijft natuurlijk 2.3

Als we deze getallen met 10 vermenigvuldigen, wordt het al wat beter.
Vandaar dat we in de formule 10*RND (1) hebben staan.
De getallen uit ons voorbeeld worden dan:

```
 5.9521943994623
 9.5434333958382
 0.3439384829427
```

Nu moeten we nog van al die cijfers achter de punt af.
Zie je trouwens dat getallen met een komma hier met een punt worden gezet.
Dat is bij de computer verplicht.
Een komma heeft namelijk in programma's een heel andere betekenis.
Maar hierop komen we later terug.
We gaan verder met het mooi maken van de drie voorbeeldgetallen.

Met het statement 'INT' wat een afkorting is van 'integer' en 'geheel getal' betekent, maken we dus van de drie getallen hele getallen.
Normaal gesproken worden getallen als volgt afgerond:

3,2 wordt 3
6,8 wordt 7
enz.

Bij 'INT' worden getallen gewoon afgekapt:

5.9521943994623 wordt 5
9.5434334958382 wordt 9
0.3439384829427 wordt 0

Mooie getallen, alleen nu worden ze hoogstens 9 en minstens 0.
Door nu bij het getal 1 op te tellen krijgen we een mooie reeks van 1 t/m 10.

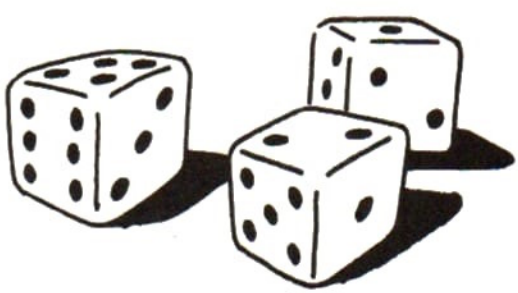
Met de formule:

A = INT (10*RND (1) +1)

krijgt A een waarde van 1 tot en met 10.

Als we de 10 veranderen in 6 krijg je dus...? juist, een dobbelsteen.

Wanneer je wat moeilijker sommen wilt, verander dan de 10 in 100 en de getallen zullen tussen 1 en 100 variëren.



We gaan weer verder met ons rekenprogramma.

Verander de regels 20 en 40 als volgt:

```
20 PRINT A;"+";B"=";
40 IF ANT = C THEN PRINT "goed" ELSE PRINT
   "fout"
```

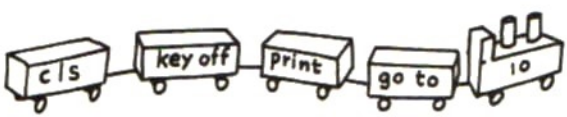
en typ de volgende regels erbij:

```
12 A = INT (10*RND(1)+1)
14 B = INT (10*RND(1)+1)
35 C = A + B
```

Verander ook de regels 10 en 50 als volgt:

```
10 CLS :KEY OFF
50 GOTO 12
```

Typ 'LIST' en druk op return.



Ook hier weer iets nieuws in regel 10.

Wanneer we na een regelnummer een statement zetten, gevolgd door een dubbele punt, dan kan daarachter nog een statement worden gezet.

Eigenlijk dus twee regels met één regelnummer.
Er kunnen er zelfs nog meer achter, tot 255 tekens, maar het wordt er dan natuurlijk niet overzichtelijker op.

We hebben ook het commando 'KEY OFF' nu in het programma opgenomen, waarom?
Wanneer we programmeren, gebruiken we de FUNKTIE-toetsen en zoals je weet geeft de regel onderaan het scherm

precies aan wat de verschillende toetsen voor ons doen.
Als we het programma gaan starten, willen we van die woorden onderaan het scherm af.
Nu we dat commando 'KEY OFF' in het programma hebben opgenomen zal dit dus automatisch gebeuren zodra het programma start.

Zie je dat het commando KEY OFF ook als KEYOFF mag worden geschreven?
Onze MSX-computer kijkt namelijk helemaal niet naar spaties. Voor de duidelijkheid houden we ze er echter wel zoveel mogelijk in, maar nodig is het niet.

```
40 IFANT=CTHENPRINT"goed"ELSEPRINT"fout"
```

mag dus wel, maar

```
40 IF ANT = C THEN PRINT "goed" ELSE PRINT "fout"
```

is natuurlijk wel duidelijker.

Bedenk echter wel dat alles wat tussen aanhalingstekens staat, ook als zodanig op het scherm verschijnt.
Hier zijn de spaties dus wel erg belangrijk.

```
135 PRINT "ik ben een computer"
```

mag natuurlijk niet zo getypt worden:

```
135 PRINT "ikbeneencomputer"
```

We gaan weer naar het programma kijken.

```
10 CLS:KEYOFF
12 A=INT(10*RND(1)+1)
14 B=INT(10*RND(1)+1)
20 PRINT A;"+";B"=";
30 INPUT ANT
35 C=A+B
40 IF ANT = C THEN PRINT "goed" ELSE PRINT
```

```
"fout"
50 goto 12
```

Zo, ons programma groeit al aardig, vind je niet?



We gaan het even doornemen.

Zoals je weet, start het programma door het commando RUN. Druk op FUNKTIE-toets 5 of typ "RUN" en druk op return.
Het programma wordt dan vanaf het begin, dus regel 10 gestart.

Het scherm wordt schoongemaakt door "CLS" en de onderste regel verdwijnt door "KEY OFF".

Dan gaat de computer in regel 12 en 14 voor twee willekeurige getallen zorgen, en die in de vorm van een som op het scherm zetten.
Hoe?

In regel 20 staat:

```
20 PRINT A
```

Dat wil zeggen PRINT de waarde van A, dus het eerste willekeurige getal, bijvoorbeeld 6.
Daarachter staat een puntkomma, waardoor het volgende teken achter het getal komt.
Dit is in ons geval: "+".

Weet je nog? Tussen aanhalingstekens wordt alles letterlijk op het scherm gezet.

Dus met:

```
20 PRINT A;"+"
```

komt er te staan:

```
6 +
```

Maar, zie je dat er een spatie staat tussen de "6" en de "+"? Dat komt omdat de computer achter een cijfer altijd een spatie zet.

De rest van regel 20 spreekt nu voor zich.

Er staat nu bijvoorbeeld

```
6 + 9 =?
```

Weet je nog, het vraagteken kwam door...? Juist, het INPUT-commando uit regel 30.

Alleen, deze zit wel vastgeplakt aan het "=" teken.
Dat kun je voorkomen door in regel 20 het volgende te veranderen.

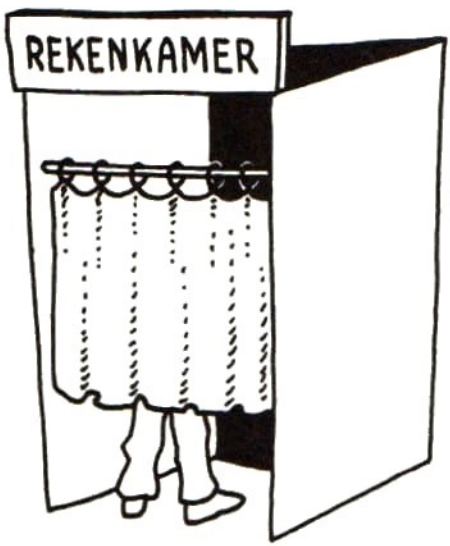
```
"=" in "= "
```

De som wordt dan:

```
6 + 9 = ?
```

Nu typ je het antwoord in. In ons voorbeeld is dat 15.

Hierna gaat de computer zelf in regel 35 het antwoord uitrekenen, maar daar zie je niets van.
Hij gaat daar A en B optellen, dus 6+9=15 en bewaart dit getal in de variabele C.



Vervolgens kijkt de computer in regel 40 of het door jou ingetypte antwoord gelijk is aan het door hem uitgerekende antwoord in variabele C.

Als dat zo is, dus:

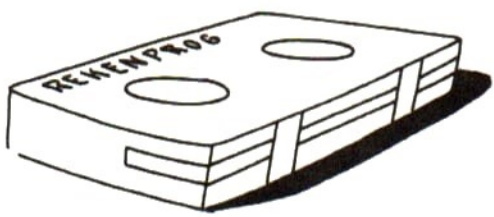
```
IF ANT = C
```

dan zal het woord "goed" op het scherm verschijnen, anders,

dus, wanneer ANT niet gelijk is aan C wordt het woord "fout" op het scherm gezet.
Hierna begint het programma weer opnieuw in regel 12.

Het beste doe je er aan dit programma goed in je op te nemen en er wat mee te stoeien.
Om te stoppen moet je wel op CTRL en STOP drukken.
Later zullen we dat ook mooier maken.

Als je het hierboven ingetypte programma eerst op een bandje zet, kun je het daarna naar hartelust wijzigen en er mee experimenteren.



Hoe kom je eigenlijk van je programma af?

Wanneer je je ingetypte programma op de band hebt gezet en het verder niet meer wilt gebruiken, moet je er nog wel vanaf.

Als je helemaal stopt is het geen enkel probleem.
Je zet de computer uit en het programma is vanzelf verdwenen.

Anders wordt het als je een ander programma wilt gaan intypen of van de band wilt halen.
Dan zijn er behalve het even uit- en weer aanzetten van de computer nog twee mogelijkheden.

Als je een programma van de band inlaadt, zal de computer namelijk het oude programma met het commando 'CLOAD' zelf uit het geheugen verwijderen.

Ga je een programma intypen, dan moet je het oude programma eerst uit het geheugen verwijderen.

Voor beide gevallen, dus inladen of intypen, is eigenlijk maar één manier goed en dat is het schoonmaken van het geheugen met behulp van het direkte commando:

```
NEW
```

'NEW' is engels voor 'nieuw' en hiermee wordt het programma volledig uit het geheugen verwijderd.

Het is zelfs mogelijk om het commando als statement, dus in een programma op te nemen.
Wees daar wel voorzichtig mee.
Als namelijk je programma 'runt' voordat je het op de band hebt gezet, zul je het weer helemaal opnieuw moeten intypen.



Voordat we ons programma verder gaan uitwerken tot een rekenprogramma met nog veel meer mogelijkheden gaan we

eerst nog wat nieuwe statements bespreken.
Hiervoor gaan we eerst even terug naar de variabelen.

Wat was ook alweer een variabele?

Je weet nog als het goed is wat een variabele is (zijn volledige naam is trouwens numerieke variabele, omdat er alleen cijfers in kunnen) en hoe je die kunt gebruiken.

Mocht je het niet helemaal meer weten, dan zullen we alles nog eens op een rijtje zetten en er nog iets aan toe voegen.

Elke variabele die je gaat gebruiken wordt door de computer altijd eerst op '0' gezet.

Ieder cijfer dat je er in wilt stoppen is toegestaan. Dus ook gebroken getallen.
Alleen, getallen met een komma moeten voor de computer worden getypt met een punt.
Dus 213,43 wordt 213.43 enz.



Je kunt met die getallen ook alle rekenkundige bewerkingen uitvoeren, zoals optellen, aftrekken enz.
Maar, er mag geen letter of woord in komen, dat accepteert de computer niet.
Als je het al eens gedaan hebt, dan zag je vervolgens de tekst:

```
REDO FROM START
```

Zo niet, probeer het dan maar eens.

Typ:

```
10 INPUT A
20 PRINT A
30 GOTO 10
```

en typ achtereenvolgens maar eens 4, 6, e, b2.

Wat betekent 'REDO FROM START'?

'Re' betekent *her*, 'do' betekent *doe*, dus 'herdoe', en dat wil in goed nederlands zoveel zeggen als: *doe nog eens*.
'From start' betekent vanaf start.
Met de zin 'doe nog eens vanaf start' zijn we er dus nu helemaal.

Dat vraagteken is weer gewoon het vraagteken van het programma, dus kunnen we direkt in plaats van de letter, het cijfer intypen.

Een nieuw commando

Behalve de hiervoor herhaalde numerieke variabele bestaat er nog een variabele en wel een alfanumerieke- of string-variabele.

Wat houdt dat in?

Een string-variabele is een variabele waarin alle karakters (tekens) van het toetsenbord geplaatst kunnen worden.

String betekent *reeks* of *serie*.
Hierin kunnen dus series tekens worden geplaatst.
De string-variabele is herkenbaar aan een dollarteken ($) achter zijn naam.
Bijvoorbeeld:

A$, B$, C4$, naam$, FT$ enz.

Zoals je ziet gelden hiervoor dezelfde voorwaarden als bij de gewone variabele.

Je begint met minstens een letter, eventueel gevolgd door één of meer letters en/of cijfers, maar wel afgesloten met het dollarteken.



Wat kunnen we er mee doen?

Je kunt er bijvoorbeeld namen, adressen en dergelijke in opnemen en bewaren, maar ook hele zinnen.

We zullen wat voorbeelden geven, dat spreekt misschien wat meer.

Typ:

```
10 A$="MSX computer"
20 PRINT "ik heb een ";A$
```

Als je het programmaatje start zal de computer

```
ik heb een MSX computer
```

op het scherm schrijven.

De variabele A$ is namelijk in regel 10 gevuld met "MSX-computer" en die wordt in regel 20 direkt achter de tekst "ik heb een " gezet.

Let er wel op, waar nodig, spaties te zetten.
Wat is het voordeel nu eigenlijk?
Je had het programma ook zo kunnen schrijven:

```
10 PRINT "ik heb een MSX computer"
```

Ook dan zal de computer, wanneer je hem start, op het scherm schrijven:

```
ik heb een MSX computer
```

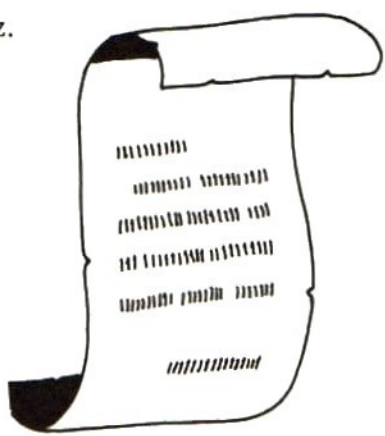
Inderdaad, in dit voorbeeld lijkt het alleen maar lastiger, het programma is met het tweede voorbeeld zelfs korter.

Maar heb je wel eens die zogenaamde persoonlijke brieven gezien, die sommige bedrijven aan mensen sturen?

Dat gaat ongeveer zo:

```
10 A$="Meneer Jansen"
20 PRINT "Beste ";A$
30 PRINT "Zoals U misschien weet, ";A$
40 PRINT "is deze brief voor u persoonlijk";
50 PRINT "en alleen u ";A$
60 PRINT "krijgt van ons bericht"
```

enz.



Als we dan regel 10 veranderen als volgt:

```
10 A$="Meneer Pietersen"
```

Dan krijgt ook meneer Pietersen een "persoonlijke brief". Zie je het bijzondere aan het gebruik van de string-variabele?

Behalve deze toepassing, kunnen we nog veel meer doen met deze variabele.

Net zoals we voor een numerieke variabele een INPUT-commando hebben, bestaat die ook voor de alfanumerieke variabele.

En niet alleen 'INPUT', maar ook nog 'LINE INPUT'.

We mogen ze allebei gebruiken, maar LINE INPUT heeft duidelijk voordelen en zal dan ook het meest worden gebruikt.

De verschillen met het gewone INPUT-commando zijn o.a.:

Er verschijnt geen vraagteken.
Alle tekens mogen worden gebruikt.
(Wanneer je bij een INPUT-commando een komma in de zin zet wordt alles achter die komma niet geprint, bij LINE INPUT wel.)

Even een voorbeeld met het LINE INPUT statement.
En natuurlijk ook weer iets nieuws.

```
10 CLS:KEYOFF
20 LINE INPUT "typ je naam ";N$
30 CLS
40 PRINT N$
50 PRINT "jou naam is ";
60 PRINT LEN(N$);
70 PRINT " tekens lang."
80 end
```

Twee nieuwe dingen

Zoals je ziet, mag je in plaats van het PRINT-commando in regel 20 ook 'LINE INPUT' gebruiken om tekst te printen.

Ditzelfde geldt ook voor 'INPUT'.

```
100 INPUT "typ een getal";A
```

mag dus ook.

Ten tweede zie je het woord 'LEN'.
Dit is een afkorting van "length" wat nederlands is voor het woord "lengte".

Hiermee kunnen we de lengte van een ingevoerd woord 'meten'. Dit kan belangrijk zijn bij het maken van een adressenbestand.
Je kunt dan bepalen hoe lang een naam maximaal mag zijn.

Het volgende voorbeeld hoef je niet in te typen. Het laat alleen zien hoe je 'LEN' kunt gebruiken in een programma.

```
100 LINE INPUT"typ je naam";N$
110 IF LEN(N$)>15 THEN 200
120 PRINT "je naam is: ";N$
      "          "
      "          "
      "          "
190 END
200 PRINT "te lang. Opnieuw s.v.p."
210 GOTO 100
```

Als de lengte van de ingevoerde naam groter is dan 15 tekens dan zal de computer naar regel 200 springen en daar de boodschap "te lang, opnieuw s.v.p." op het scherm zetten.

Hierna gaat hij weer terug naar regel 100 zodat de naam opnieuw, maar nu korter kan worden getypt.
Bedenk wel, dat wanneer je getallen invoert, je er niet mee

kunt rekenen.
Die getallen moet je zien als bijvoorbeeld de postcode of je huisnummer.

Wanneer je bijvoorbeeld probeert twee getallen op te tellen als volgt:

```
10 A$="2345"
20 B$="7623"
30 PRINT A$+B$
```

dan zet de computer op het scherm:

```
23457623
```

Zo kunnen we ook met namen goochelen:

```
10 A$="Jan "
20 B$="Jansen"
30 C$="Klaassen"
40 PRINT A$+B$
50 PRINT A$+C$
```



dan verschijnt er:

```
Jan Jansen
Jan Klaassen
```

Voorlopig laten we het bij deze uitleg.
In het volgende hoofdstuk passen we een string-variabele op een bijzondere manier toe, later komen we nog verder op de variabele terug.

We gaan ons programma mooier maken

Een programma is pas kompleet wanneer je niet zelf hoeft uit te leggen hoe het werkt.
Dat moeten we de computer kunnen laten doen. Een stukje tekst waarin staat wat de bedoeling is van het programma maakt het allemaal wat duidelijker.

Wanneer je een stukje tekst op het scherm hebt staan, is het natuurlijk heel belangrijk dat je die rustig kunt lezen.
Met andere woorden, het is niet de bedoeling dat de computer uitmaakt wanneer je klaar moet zijn met lezen.

In veel programma's gebeurt dat nog wel.
De programmeur heeft er dan voor gezorgd dat een stuk tekst een bepaalde tijd op het scherm blijft en laat het dan ineens verder gaan , zodat de één het nog niet heeft kunnen lezen terwijl een ander het stukje al lang heeft gezien en moet wachten tot de computer verder gaat.

Ook wordt wel eens met een zogenaamde tijdbalk gewerkt.
De balk (langs de tekst) wordt dan korter en geeft hiermee de tijd aan die je nog hebt voordat de tekst verdwijnt.

Maar ook hiervoor geldt hetzelfde als in het eerste geval.
Er zijn nu eenmaal snelle en minder snelle lezers.

Wij doen dat dus anders.
Er is een heel mooi commando waarmee je de computer net

zo lang kunt laten wachten als je zelf wilt.
Dit commando heet 'INKEY$' en dat is een afkorting van 'INPUT KEY STRING'.
Moeilijk om letterlijk te vertalen, maar het betekent dat de computer met dit commando een ingetypt karakter opneemt.

Hiermee kunnen we dan een string-variabele vullen en zo werken met dit gegeven.

Een voorbeeld:

```
100 CLS:KEYOFF
110 PRINT "gebeurt er nu nog wat?"
120 PRINT "typ een toets"
130 K$=INKEY$:IF K$ = "" THEN 130
140 PRINT "zo, eindelijk"
150 END
```

Daar staan weer wat vreemde dingen, maar dat hebben we zo door.

In regel 110 staat:

```
PRINT "gebeurt er nu nog wat?"
```

En inderdaad als je de computer start zal er na deze zin verder niets gebeuren.
Waarom niet?
De computer zal in regel 130 de variabele K$ vullen met 'niets' want zolang we niets intypen blijft INKEY$ leeg.
In het tweede deel van regel 130 'kijkt' de computer wat er in K$ zit.
Als daar niets in zit dus als K$="" dan zal de computer verder gaan naar regel 130.

```
IF K$=""THEN 130
```

Maar, dat is de regel waarin we bezig zijn.
Inderdaad. Daarom zal de computer, zolang je niets intypt,

in deze regel blijven staan.
Al die tijd kun je dus rustig de tekst lezen.

Wanneer je de tekst hebt gelezen, druk je op een toets, het geeft niet welke en de computer zal verder gaan.
Waarom?

Zodra je een toets indrukt, bijvoorbeeld de letter 'g' dan zal 'INKEY$$' deze letter 'g' overbrengen naar K$.
Als we dan in het tweede deel van regel 130 kijken zal K$ niet langer leeg zijn, maar de letter 'g' bevatten.

Aan de voorwaarde K$="" (K$=leeg) voldoen we niet meer, dus zal de computer verder gaan naar het volgende regelnummer en daar zijn opdracht uitvoeren.

Dat is in dit geval het uitprinten van de tekst

```
"Zo, eindelijk"
```

IF betekende 'als', dus als K$ leeg is blijft de computer op regel 130, anders loopt hij door.

De computer komt nu op regel 140 en zal daar de tekst "Zo, eindelijk" op het scherm laten verschijnen.

Het kan nog mooier

Met een wat uitgebreider programma kun je de computer ook ergens anders heen sturen.
Typ dan het volgende programmaatje.

```
100 PRINT "Kan je al een beetje programmeren?"
110 PRINT" typ j of n"
120 K$= INKEY$;IF K$=""THEN 120
130 IF K$="j" THEN 200
140 IF K$="n" THEN 300
150 PRINT "je typte geen j of n"
160 GOTO 120
200 PRINT"dan heb je goed opgelet,ga zo door"
```

```
210 END
300 PRINT "neem het boek dan nog maar eens door"
310 END
```

Zo, weer heel wat, dat leggen we maar even snel uit.

Je weet nu dat de computer bij regel 120 wacht tot je iets typt.
Nu willen we dat je niet zo maar een toets, maar een bepáálde toets indrukt.
Zodra je een toets indrukt zal de variabele K$ zich daarmee vullen en verder gaan met het programma.

Het eerste wat hij dan tegen komt is regel 130 waarin hij kijkt of het ingetypte teken een 'j' is.

Als dat zo is dan wordt hij dus naar regel 200 gestuurd, zo niet dan zal de computer kijken in regel 140 of er dan misschien een 'n' was ingetypt.
Wanneer dat ook niet het geval is dan kan de computer dus met een gerust hart op het scherm zetten: "je typte geen j of n". Nu kun je alsnog 'j' of 'n' typen.

Kijk wel goed of je hoofd- of kleine letters intypt, voor de computer is de 'j' namelijk heel wat ander dan de hoofdletter 'J'.

Wil je de computer op allebei de tekens laten reageren dan kun je de regels veranderen als volgt:

```
120 IF K$="j" OR K$="J" THEN 200
130 IF K$="n" OR K$="N" THEN 300
```

Nu zal de computer dus kijken *of*, want dat betekent *or*, K$ gevuld is met een grote of een kleine letter en op beide hetzelfde reageren.

Er is nog een commando waar je programma een stuk mooier door wordt, niet voor de gebruiker of voor de computer alleen voor jou, de programmeur.

Zo heet het commando waarmee je in het programma alles kunt zetten wat jou wel, maar de computer niet interesseert.

Een voorbeeld?
Ieder programma heeft wel een naam.
Het eerste programma uit dit boek noemen we bijv. 'reken', maar dat is een naam die we gebruiken voor het laden van het programma.

Zodra we het programma listen, zien we wel een aantal regels die ons bekend voorkomen, maar vooral als je meer programma's hebt gemaakt, is het wel eens een beetje verwarrend.

Daarvoor is "REM" nu een uitstekend hulpmiddel.
REM betekent "remark" en dat is engels voor: opmerking.
Iedere programmaregel waar "REM" staat na het regelnummer wordt door de computer gewoon overgeslagen, maar jij kunt daar van alles in schrijven.

Zoals altijd zegt een voorbeeld veel meer:

```
10 REM dit is een rekenprogramma
20 REM gemaakt door W. Wortel
30 REM op een MSX computer.
40 REM ------------------------
50 PRINT "rekenen"
60
' '
' '    (rekenprogramma)
' '
' '
200 END
```

Alle regels die met een REM-commando beginnen worden dus niet door de computer bekeken.
Er mogen zelfs commando's in staan.
Typ het volgende maar eens:

```
10 REM A = 3
20 PRINT A
```

Zoals je ziet, reageert de computer totaal niet op de inhoud van regel 10. Hij print gewoon "0".

Het is dan ook een goede gewoonte ieder programma te beginnen met wat informatie over het programma, de naam van de programmeur, de naam van het programma, de computer enz.

Midden in een programma is het ook een handig hulpmiddel. Bijvoorbeeld als je wat in je programma wilt veranderen, vind je met behulp van de REM-regels gemakkelijk de diverse programma-delen terug.
Want vergeet niet, je programma's groeien voor je het weet tot hele lange lijsten en dan is het zoeken naar bepaalde programma-delen, net als het zoeken naar de bekende speld in de even bekende hooiberg. (In plaats van 'REM' mag je ook het aanhalingsteken " ' " typen.)

Een rekenprogramma

Met de tot nu toe geleerde commando's en statements kunnen we al heel wat doen.
Typ het onderstaande programma maar in.
Daarna kijken we naar de statements die je nog niet kent.

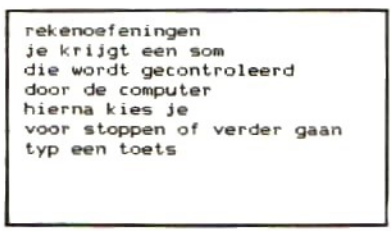
Bedenk echter één ding.
Zodra je het programma hebt ingetypt, moet je het eerst "SAVEN".
Dit is heel belangrijk want als je een grote fout hebt gemaakt, kan de computer wel eens heel raar gaan doen.
Het enige dat je dan kunt doen is de computer uitzetten en als je je programma dan niet op de band hebt staan, ben je het wel kwijt.

```
10 REM rekenprogramma
20 REM gemaakt door:
30 REM w. wortel
```

```
40 REM op een msx computer
50 CLS: KEY OFF
60 PRINT"rekenoefeningen"
70 PRINT"Je krijgt een som"
80 PRINT"die wordt gecontroleerd"
90 PRINT"door de computer"
100 PRINT"hierna kies je"
110 PRINT"voor stoppen of verder gaan"
120 PRINT"typ een toets"
130 K$=INKEY$:IF K$=""THEN 130
140 CLS
145 REM hier maakt de computer willekeurige getallen
150 A = INT (10 * RND (1)+1)
160 B = INT (10 * RND (1)+1)
165 REM afdrukken som op scherm
170 PRINT A;"+";B;"=";
180 INPUT ANT
185 REM controle antwoord
190 C=A+B
200 IF ANT = C THEN PRINT "goed" ELSE PRINT "fout"
205 REM keuze doorgaan of stoppen
210 PRINT "wil je nog een som?"
220 PRINT"typ j of n"
230 K$=INKEY$:IF K$=""THEN 230
240 IF K$="n" OR K$="N" THEN END
250 GOTO 140
```

Zie je het voordeel van de REM-regels?
Ze zijn er later ingetypt (kijk maar naar de regelnummers).
Hiermee zie je nog eens hoe belangrijk het nummeren met tussenruimte is.

Verder staat er nu wel een heleboel, maar toch eigenlijk niets nieuws.
Wanneer je het programma start zal echter de tekst als volgt op het scherm verschijnen.



Veel beter staat het wanneer de tekst midden op het scherm verschijnt.
Ook de regel "typ een toets" zal bijvoorbeeld onderin het scherm veel beter opvallen.
Om de tekst op iedere plaats te krijgen die we zelf willen zijn diverse commando's te gebruiken.
Eén daarvan is het commando:

```
LOCATE x,y
```

"LOCATE" betekent "plaatsen". Dat wil zeggen hiermee bepalen we op welke plaats de computer een PRINT-commando moet beginnen en met de twee variabelen x en y geven we die plaats aan.

Voor we hiermee gaan werken moet je eerst weten hoe de computer eigenlijk zijn tekst op het scherm zet.

Zet daarom eerst je rekenprogramma op de band, zodat je je computer vrij hebt voor de voorbeeldprogramma's.

Schermindelingen

Het scherm bestaat uit een aantal vakjes.
In elk vakje kan een leesteken, dat wil zeggen een letter, cijfer, spatie of ander teken van het toetsenbord staan.

Ieder vakje kan dus worden beschreven.
Alleen als we het niet aangeven, zal de computer altijd beginnen in het vakje, links-boven op het scherm.

Hoe tellen deze vakjes?

Elke plaats heeft een nummer, of eigenlijk twee.
Je kunt het vergelijken met een schaakbord.
Elk vak heeft daar een letter en een cijfer.
Met het noemen van die twee gegevens weet een schaker precies waar zijn schaakstuk staat.
Bijvoorbeeld bij het begin van het spel staat de witte dame

op E1, de zwarte op E8.

Bij onze computer werken we alleen met cijfers.
Maar hier moeten we even opletten. Een schaakbord is altijd even groot, namelijk 8 x 8 vakken.
Dat geldt niet voor de computer.
Als we de computer aanzetten is het scherm 37 vakjes breed.
Maar met het commando:

```
WIDTH x
```

kunnen we dat aantal veranderen, en wel van 0 tot en met 39. Door eenvoudig te typen:

```
WIDTH 40
```

kunnen er 40 tekens op elke regel.

Alleen, bij de meeste beeldschermen geeft dat aan de zijkanten wat problemen, er vallen dan wat tekens buiten beeld.

Dit commando "WIDTH" ("breedte") kan uiteraard ook in het programma worden verwerkt.

Terug weer naar het LOCATE-commando.

Als de onderste regel niet is beschreven door de FUNKTIE-toets-commando's hebben we de beschikking over 24 regels. Het aantal tekens naast elkaar wordt bepaald door het commando "WIDTH" maar is, zoals gezegd, bij het aanzetten van de computer 37 tekens.

Achter het commando "LOCATE" komt eerst het getal dat aangeeft waar de computer moet beginnen vanaf de linker kantlijn.

```
LOCATE 5 :PRINT "MSX"
```

Wil dus zeggen:

Plaats op de 6e plaats vanaf de linkerkant het woord "MSX" op regel 0.

Let op, de computer begint bij 0, dus LOCATE 5 is de 6e plaats.

```
LOCATE 5,10: PRINT "MSX"
```

wil dus zeggen: zet "MSX" op de 6e plaats op regel 11.

LOCATE 0,0 is dus helemaal linksboven.

Dat is ook de plaats waar de computer begint te schrijven als je geen LOCATE-commando typt.

```
PRINT "abcde"
```

zal op dezelfde plaats komen als:

```
LOCATE 0,0:PRINT "abcde"
```

Het midden van het scherm is (bij WIDTH 40), horizontaal 20 en verticaal 12.

Typ maar eens een kruisje midden op het scherm als volgt:

```
10 LOCATE 20,12
20 PRINT "x"
```

Een goede gewoonte is deze twee regels steeds achter elkaar te typen, gescheiden door een dubbele punt.

Je ziet dan precies waar het LOCATE-commando bij hoort.

```
10 LOCATE 20,12:PRINT "x"
```

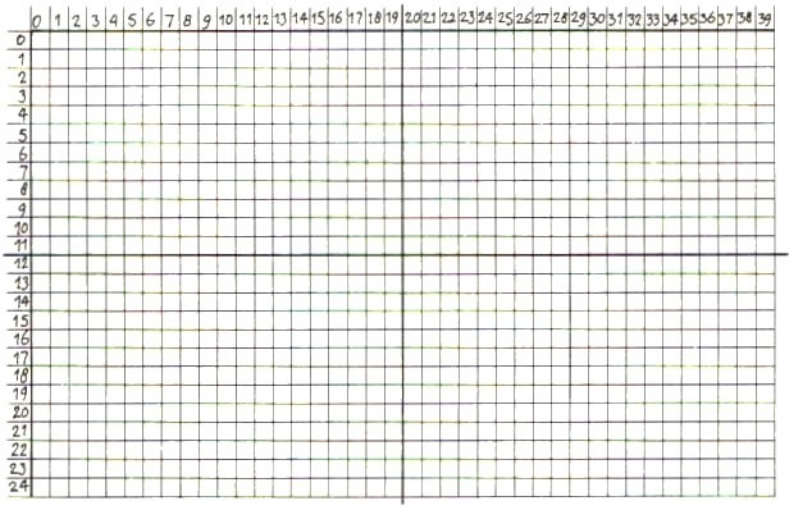
Met dit LOCATE-commando zijn nog een heleboel truuks uit te halen, maar we zullen hier alleen bespreken wat we echt gebruiken.

We zullen nu eens kijken hoe we ons programma "rekenoefeningen" hiermee wat kunnen verbeteren.

Het beste is, om de tekst die in de regels 60 t/m 120 staat eerst eens op papier te zetten om zo een mooie indeling te krijgen.

Teken hiervoor het onderstaande voorbeeld na, op bijv. ruitjes papier.

Beeldschermformulier



Nu kunnen we op deze tekening precies aangeven waar en hoe we de tekst willen hebben.

```
                    1111111112222222223333333333
  0123456789012345678901234567890123456789
0
1                 rekenoefeningen
2
3
4               je krijgt een som
5            die wordt gecontroleerd
6                door de computer
7
```

```
 8            hierna kies je
 9       voor stoppen of verder gaan
10
11
12
13
14
15
16
17
18
19              typ een toets
20
21
22
23
24
```

We gaan om te beginnen ons programma maken volgens dit voorbeeld.

Op de tekening kun je nu precies zien waar elke regel begint. We veranderen de regels waarin de tekst staat als volgt:

```
50 CLS:KEY OFF:WIDTH 40
60 LOCATE 12,1:PRINT"rekenoefeningen"
70 LOCATE 11,4:PRINT"je krijgt een som"
80 LOCATE  8,5:PRINT"die wordt gecontroleerd"
90 LOCATE 12,6:PRINT"door de computer"
100 LOCATE 13,8:PRINT"hierna kies je"
110 LOCATE  6,9:PRINT"voor stoppen of doorgaan"
120 LOCATE 14,19:PRINT"typ een toets"
```

Wanneer je nu het programma RUNT ziet het er al heel wat beter uit.

```
          rekenoefeningen

        je krijgt een som
      die wordt gecontroleerd
          door de computer


          hierna kies je
      voor stoppen of verder gaan



              typ een toets
```

Tot besluit van dit hoofdstuk moeten we nog de som en de uitslag een mooi plaatsje geven op het scherm.

Zet de teksten op het beeldschermformulier als volgt uit:



Ook al staan de vier teksten

1. de som
2. goed of fout
3. wil je nog een som?
4. typ j of n

niet tegelijkertijd op het scherm, toch kunnen we ze vast indelen.

Voor de teksten 1, 3 en 4 kunnen we de programma-regels als volgt veranderen:

```
170 LOCATE 14, 7: PRINT A;"+";B"= ";
210 LOCATE 11,18: PRINT"wil je nog een som?"
220 LOCATE 15,20: PRINT"typ j of n"
```

Voor de tekst nr. 2 maken we een uitzondering.

Na regel 190 zal de computer namelijk "goed" of "fout" op het scherm zetten.
Door nu direkt na regel 190 het LOCATE-commando te plaatsen zal dit voor zowel "goed" als "fout" gelden.

We typen:

```
195 LOCATE 18,15
```

Wanneer we nu het programma starten, zal elke tekst op de afgesproken plaats komen.

Nog even dit.

Je zult vast wel eens het programma willen onderbreken terwijl de computer met het programma bezig is.
Bijvoorbeeld, wanneer je het met de loop van het programma niet helemaal eens bent.

Dat is mogelijk.
Hiervoor moeten dan wel *twee* toetsen worden ingedrukt.
Waarom?
Dit is bewust gedaan door de ontwerpers van de computer. Hiermee voorkom je het per ongeluk stoppen van een computerprogramma, want dat kan grote problemen geven.
In de meeste gevallen zijn namelijk alle ingevoerde gegevens uit het geheugen verdwenen als je het programma weer start.

Hoe heten de twee toetsen om het programma te stoppen?

De CTRL- toets en de STOP-toets.

Zodra deze samen ingedrukt worden zal de computer stoppen met de volgende melding:

```
BREAK IN XXX
```

XXX staat voor een regelnummer.

Het programma wordt met "RUN" weer gestart, maar dan staan wel alle variabelen op nul.

Alleen op de STOP-toets drukken kan ook.
De computer zal dan inderdaad stoppen met het uitvoeren

van het programma, maar er volgt geen melding.
Wanneer je dan weer op de STOP-toets drukt zal hij weer verder gaan waar hij gebleven was.
Alle variabelen behouden dan hun waarde.

Voor we verder gaan met het uitbreiden van het rekenprogramma zullen we in het volgende hoofdstuk eerst een paar nieuwe commando's bespreken met behulp van wat voorbeeldprogramma's.

Die commando's worden dan later weer in het rekenprogramma gebruikt.

De FOR-NEXT loop

De nieuwe statements die we nu gaan behandelen lijken misschien wat moeilijker dan de anderen, maar we hebben inmiddels al zoveel geleerd dat dit er ook nog wel bij kan.

Het gaat hier niet om één woord maar een combinatie van woorden, namelijk:

FOR-TO-STEP

En dan ook nog:

NEXT

Zo, dat zijn ze.

Wanneer ik zeg: "tel van één tot tien", dan zal ik, als het goed is, horen:

1 2 3 4 5 6 7 8 9 10

Je begint bij 1 en je stopt bij 10.

Nu de computer.

Als ik tegen hem zeg: „Tel van één tot tien", dan moet dat op de volgende manier gebeuren.

```
10 FOR X = 1 TO 10
20 PRINT X;
30 NEXT X
```

Nu zal de computer, wanneer ik het programma start,

```
1 2 3 4 5 6 7 8 9 10
```

op het scherm zetten.

Kennelijk telt hij net als wij.
Maar hoe?

"FOR" betekent letterlijk "voor" maar in dit geval moeten we FOR X = 1 meer zien als: "laat X 1 zijn".

"TO" betekent "tot".

Er staan in regel 10 eigenlijk twee aparte dingen.

FOR X = 1 TO 10

1. laat X 1 zijn
2. verhoog X van 1 tot 10

Dat verhogen doet de computer met "NEXT", wat "volgende" betekent.

Iedere keer wanneer de computer in de regel komt waar "NEXT X" staat, zal de computer kijken of de eindwaarde, in dit geval 10 al bereikt is.
Zo niet dan zal de computer X met 1 verhogen en teruggaan naar de regel die direkt na "FOR X = 1 TO 10" komt.

Als de waarde 10 wel bereikt is zal de computer het programma weer vervolgen met de eerstvolgende regel na

"NEXT".

Dit statement noemen we in z'n geheel ook wel een "FOR-NEXT-LOOP".
"LOOP" betekent "lus"

Wat je nu nog mist is de uitleg van het woordje "STEP".
"STEP" betekent "stap" en dat wil zeggen:
"vergroot het getal met stappen van: ..."

Verander regel 10 maar eens als volgt:

```
10 FOR X = 2 TO 10 STEP 2
```

Dan zal de computer op het scherm zetten:

```
2 4 6 8 10
```

Step 2 wil dus zeggen met stappen van twee.

Zie je ook dat je niet met 1 hoeft te beginnen?

Nog een mogelijkheid.

Van 20 naar 1 tellen, kan ook.
Verander regel 10 maar weer.

```
10 FOR X = 20 TO 1 STEP -1
```

Even goed onthouden, wanneer je aftelt moet altijd de STEP-waarde met een min-teken worden getypt. Ook als je met 1 wilt verminderen.

Nu heb je dit allemaal wel geleerd, maar we moeten er ook wat mee kunnen doen natuurlijk.

Stel dat je vijf namen wilt invoeren en die je later weer op het scherm zetten wilt.

Dat kan dan met een FOR-NEXT loop als volgt:

```
10 FOR I= 1 TO 5
20 LINE INPUT "typ een naam";N$(I)
30 NEXT
40 CLS
50 FOR U = 5 TO 1 STEP -1
60 PRINT N$(U)
70 NEXT
80 END
```

Zie je wat er gebeurt?

Een paar nieuwe dingen.
Ten eerste is achter NEXT de variabele-naam weggelaten.
Dat mag, omdat de computer toch weet om welke variabele het gaat.
Ten tweede, bij de invoer van de namen gebruiken we als variabele "I", en bij de uitvoer de "U", dat maakt dus blijkbaar ook niets uit.

Verder zie je nu achter N$ de variabele (I) staan.
Zodra het programma start, wordt I op 1 gezet.
In regel 20 wordt dus in N$(1) een naam ingevuld.
Als de computer na de NEXT in regel 30 teruggaat naar regel 20 dan zal nu dus N$(2) ingevuld worden.
Dit gaat door tot N$(5) is voorzien van een naam.

Hierna gaat de computer naar regel 40, maakt het scherm schoon en zal nu achtereenvolgens de vijf ingevoerde namen op het scherm zetten, maar dan in omgekeerde volgorde.
Zie je, dat je met die ingevoerde namen van alles kunt doen?

Een andere mogelijkheid met de FOR-NEXT loop is bijvoorbeeld het maken van de rekentafels.

Hiervoor kunnen we de FOR-NEXT loop heel goed gebruiken.

Typ maar in:

```
10 CLS:KEY OFF
```

```
20 INPUT"welke tafel wil je ";B
30 CLS
40 PRINT "de tafel van ";B
60 FOR A = 1 TO 10
70 PRINT A;"*";B;"=";A*B
90 NEXT
100 LINE INPUT "druk op return ";R$
120 GOTO 10
```

Voor je dit programma start even iets over regel 110.
Om een programma te onderbreken en door de gebruiker weer te laten starten gebruikten we bijvoorbeeld de volgende regels:

```
50 K$=INKEY$:IF K$= ""THEN 50
120 IF K$ ="j" THEN 200
130 IF K$ ="n" THEN 300
140 GOTO 500
```

En nu ineens dit.

Waarom?

Er is een duidelijk verschil.
Met het laatste voorbeeld kun je namelijk een keuze maken en wel tussen:

1. blijven staan
2. door "j" te typen naar 200 gaan
3. door "n" te typen naar 300 gaan
4. door iets anders te typen naar 500 gaan

Met regel 110 in ons voorbeeld-programma maken we geen gebruik van een keuze.

Het programma stopt totdat de gebruiker het programma weer verder laat gaan.
In dit geval mag de regel zoals in het tweede voorbeeld ook.

```
50 K$=INKEY$:IF K$=""THEN 50
```

Hier zal de computer net zo lang in regel 50 blijven tot er een toets ingedrukt wordt.

Het nadeel hiervan is dat het een beetje slordig staat, wanneer het niet uitmaakt welke toets je indrukt.

Met het voorbeeld in regel 110:

```
110 LINE INPUT "druk op return ";R$
```

wordt de gebruiker als het ware gedwongen op de return-toets te drukken.

Wat doet de computer als de return-toets wordt ingedrukt?

Normaal is het zo, dat als je een of meer tekens in een variabele invoert, je de invoer afsluit door op de return-toets te drukken.
Hierna vervolgt de computer dan het programma.

In ons geval, dus met variabele R$ wordt direkt op de return-toets gedrukt.
Ook hier gaat de computer dus verder met het programma.

Nu weer terug naar de rekentafels.
Met dit programma krijg je, als je om de tafel van 7 vraagt:

```
1 * 7 = 7
2 * 7 = 14
3 * 7 = 21
4 * 7 = 28
5 * 7 = 35
6 * 7 = 42
7 * 7 = 49
8 * 7 = 56
9 * 7 = 63
10 * 7 = 70
```

Maar wat zien we nu weer?
Een rare tafel, vind je niet?

Ten eerste staat alles weer links boven in het scherm, ten tweede wordt door de 10 onder in de tafel het geheel direkt een slordig aanzien gegeven.

We gaan hier eerst wat aan doen, voor we verder gaan met de FOR-NEXT loop.

Hiervoor moeten we dan wel weer twee nieuwe commando's leren.

Om teksten op het scherm te zetten waar wij willen, heb je al het LOCATE-commando geleerd.
Hiermee kan op elke plaats van het scherm een letter worden gezet.

Als we alleen van links naar rechts willen bepalen waar we wat neerzetten, kan dat met het volgende commando:

```
TAB(xx)
```

"TAB" is een afkorting van "tabulator" en dat komt je misschien bekend voor van een schrijfmachine.

Hiermee kun je op een regel bepalen waar je start met schrijven.
We veranderen de volgende regels.

```
40 PRINT TAB (8) "de tafel van";B
70 PRINT TAB (8) A;"*";B;"=";A*B
```

Typ ook de regels:

```
50 PRINT
80 PRINT
100 PRINT
```

Hiermee krijg je lege regels, dus tussenruimte.
Zo wordt het al een stuk beter, alleen die 10 zorgt nog voor de nodige moeilijkheden.

Zet dit programma maar even op cassette, dan gaan we eerst

even wat anders leren.

PRINT USING

We gaan een commando leren, dat eigenlijk voor gevorderde programmeurs bestemd is, maar daar schrikken we toch zeker niet meer van?
Tenslotte ben je toch geen beginner meer!

Het commando heet "USING" en wordt gebruikt achter het commando "PRINT" om cijfers op een bepaalde manier op het scherm te krijgen.

"USING" betekent "gebruikmaken".

"PRINT USING" betekent "printen met gebruikmaking van".

De plaats waar de cijfers moeten komen geven we aan met het symbool "#" dat ook wel het hekje genoemd wordt.

Typ het volgende programma:

```
10 FOR X = 1 TO 3
20 INPUT A(X)
30 NEXT X
40 FOR X = 1 TO 3
50 PRINT A(X)
60 NEXT X
70 END
```

Als je invoert: 45, 3, 275 dan zal de computer de cijfers als volgt neerzetten:

```
45
3
275
```

Wanneer je regel 50 als volgt verandert:

```
50 PRINT USING "###";A
```

dan zal het rijtje er zo uitzien:

```
 45
  3
275
```

Dat ziet er heel wat beter uit, vind je niet?

Zo kunnen we ook gebroken getallen netjes onder elkaar zetten. Verander regel 50 als volgt:

```
50 PRINT USING "###.##";A
```

Voeren we nu in: 34.25 435.7 12.76 2 dan krijgen we dit rijtje:

```
 34.25
435.70
 12.76
  2.00
```

Zie je dat de computer achter de punt de lege plaatsen aanvult met nullen?

Behalve de hekjes mogen we dus ook binnen de aanhalingstekens andere tekens zetten.

Hiermee kunnen we ons tafelprogramma ook een stukje mooier maken.
Laad het maar weer in.
We hadden eerst:

```
10 CLS:KEY OFF
20 INPUT"welke tafel wil je ";B
30 CLS
40 PRINT TAB(8) "de tafel van ";B
50 PRINT
60 FOR A = 1 TO 10
70 PRINT TAB(8) A;"*";B;"=";A*B
80 PRINT
```

```
90 NEXT
100 PRINT
110 LINE INPUT "druk op return ";R$
120 GOTO 10
```

Met USING wordt regel 70:

```
70 PRINT TAB(8) USING "## x ### = ####";A;B;A*B
```

Het is heel wat, dat is waar, maar als we die regel goed bekijken, dan is het echt niet zo moeilijk.

PRINT en TAB (8) hebben we al gehad.
Hiermee geven we aan dat de tekst op tabulator 8 moet beginnen.
Met using "## x ### = ####" geven we aan dat het eerste getal uit twee cijfers mag bestaan, het tweede uit drie, en het derde getal uit vier cijfers.
Waarom?

Het eerste getal wordt niet groter dan twee cijfers, want onze tafel gaat van 1 tot 10.
Als het tweede getal uit max. drie cijfers bestaat, bijv. 999 dan zal het derde getal (10 x 999 = 9990) nooit meer dan vier cijfers groot worden.

Verder staan de beide tekens "x" en "=" binnen de aanhalingstekens.

Dan nog de getallen zelf.
Die staan achter het USING "*masker*" zoals dat ook wel heet. A is de variabele die steeds opgehoogd wordt van 1 tot 10.
B wordt gevuld met de gewenste tafel en tot slot staat er nog A*B, dat is de uitkomst van de berekening.

Je mag ook meer dan één FOR-NEXT loop toepassen.
Hiermee kun je bijvoorbeeld twee getallen op laten lopen.
Verander in het tafel-programma de volgende regels maar

eens.

```
20 FOR B = 1 TO 10
120 NEXT B
```

Hiermee krijg je alle tafels van 1 tot 10, achter elkaar.

Hoewel met PRINT USING nog veel meer mogelijk is, heb je hier voorlopig wel genoeg aan.
Ook met de FOR-NEXT loop kunnen we nu aardig wat truuks uithalen.
In ons rekenprogramma en nog enkele kleine voorbeeldprogramma's zul je ze nog wel eens tegenkomen.

Probeer intussen zelf ook eens iets te maken waarbij je gebruik maakt van alle tot nu toe geleerde statements.

Een paar programma's

De computer wordt een getallenraadmachine.

```
10 REM getallen raden
20 REM door japie rademaar
30 REM voor msx
40 REM ------------------
50 KEY OFF:WIDTH 40
60 CLS:T=0
70 A=INT(100*RND(-TIME)+1)
80 LOCATE 10,6:PRINT"raadt het getal"
90 LOCATE 15,8:INPUT X:T=T+1
100 LOCATE 15,10
110 IF X = A THEN 150
120 IF X < A THEN PRINT "hoger" ELSE PRINT "lager"
130 FOR Y = 1 TO 500:NEXT Y
140 CLS :GOTO 80
150 LOCATE 10,15:PRINT"goed geraden"
160 LOCATE 10,17:PRINT"en dat in ";T;"keer"
170 LOCATE 8,22:PRINT"nog eens? typ j of n"
180 K$=INKEY$: IF K$="" THEN 180
190 IF K$ = "j" OR K$ = "J" THEN 60
200 CLS:END
```

Zo, daar staat weer wat.
Gelukkig al heel wat bekende kreten, maar ook nog genoeg problemen.
Die pakken we dus maar meteen even aan.

De eerste is het woordje TIME in regel 70.
We hadden hier eerst toch 1 staan?
Ja, dat is waar, maar dat geeft toch wel een klein probleem met dit spelletje.
De computer is namelijk zo ontzettend snel, dat hij elke keer wanneer hij start met hetzelfde getal begint.
En dan is het raden snel voorbij natuurlijk.
Nu heeft de computer ook een ingebouwde teller waarmee hij zelf werkt.
Wat hij precies doet en hoe, is nu even niet aan de orde, maar met behulp van de bijzondere variabele "TIME" wat "tijd" betekent, kunnen we die teller wel zichtbaar maken en gebruiken.

Bijzonder, waarom?

Wel, deze variabele is geen gewone maar een systeem-variabele. Dat heet zo, omdat de computer zelf deze variabele gebruikt. Het enige wat wij kunnen doen is de teller ermee zichtbaar maken en op nul zetten.

Typ:

```
PRINT TIME
```

en er zal een getal op het scherm komen.

Typ nu nog eens:

```
PRINT TIME
```

Zie je dat er nu een ander, hoger getal staat?

Typ nu:

```
10 TIME = 0
20 LOCATE 10,10: PRINT TIME
30 GOTO 20
```

Druk op F5 (RUN) en je ziet de teller tellen.

Dit programma stopt slechts door op CTRL/STOP te drukken, maar de teller loopt gewoon door.

Typ maar eens:

```
PRINT TIME
```

Zie je dat de laatste stand van de teller in je programma lager is dan het getal dat nu op je scherm verschijnt?

Door nu deze teller in onze formule te verwerken zal het willekeurige getal dus echt willekeurig worden.

Het tweede dat je tegenkomt is in regel 120 het teken ⟨.

IF X ⟨ A THEN PRINT enz.

Wat betekent dit?

In het rekenprogramma dat we al eerder maakten stond:

IF A = C THEN enz.

Weet je nog?
Dat betekende dat wanneer A gelijk was aan C dan moest het woord "goed" op het scherm komen.

Behalve dit "=" teken, waarmee we kunnen kijken of twee waarden gelijk zijn, zijn er nog een aantal mogelijkheden om te vergelijken.

We zullen ze eens allemaal op een rijtje zetten.

=	gelijk aan.
⟨	kleiner dan
⟩	groter dan
⟨⟩	ongelijk aan
⟨=	kleiner of gelijk aan
⟩=	groter of gelijk aan

Dat zijn er nogal wat.
Maar we gaan ze dan ook even rustig bekijken.

Laten we als voorbeeld eens nemen:

Jan = 1,50 m lang
Piet = 1,34 m lang
Annie = 1,50 m lang
Wilma = 1,48 m lang

Wat kunnen we hierover allemaal zeggen?

Jan is groter dan Piet	dus, Jan	⟩ Piet
Wilma is kleiner dan Annie	dus, Wilma	⟨ Annie
Annie is net zo groot als Jan	dus, Annie	= Jan
Jan is niet gelijk aan Piet	dus, Jan	⟨⟩ Piet

En zo kun je nog wel even doorgaan.

Zie je wat de bedoeling is van die tekens?

Weet je trouwens hoe je ze heel eenvoudig kunt onthouden?

Dat teken voor kleiner ⟨ is net de k van "kleiner".
Als je dat onthoudt weet je de andere vanzelf.

Dan regel 130.

Hier staat:

```
FOR Y = 1 TO 500 :NEXT Y
```

De manier waarop de commando's "FOR" en "NEXT" hier worden gebruikt is eigenlijk een manier om de computer even te laten wachten.

Hoe werkt het?

Met FOR wordt de variabele Y op 1 gezet, en met NEXT direkt opgehoogd.

De computer zal dit dus blijven doen tot 500, echter zonder dat hij iets hoeft uit te voeren.
Maar dat ophogen kost wel tijd, en dat is nu precies wat we willen.

Al de tijd dat de computer staat te tellen, kunnen wij lezen wat er op het scherm staat.
Wanneer de computer klaar is, zal hij verder gaan met het programma en dus in regel 140 het scherm schoonmaken en verder gaan naar regel 80.

De lengte van de pauze is in te stellen door eenvoudig het laatste getal, hier dus 500 te vergroten of te verkleinen.

In ons voorbeeld duurt de pauze ongeveer één seconde.

```
FOR Y = 1 to  500: NEXT (1 seconde)
FOR Y = 1 to 1000: NEXT (2 seconden)
FOR Y = 1 to 1500: NEXT (3 seconden)

enz.
```

De variabele-naam achter NEXT mag weggelaten worden.

```
NEXT Y of NEXT
```

is voor de computer gelijk.

Dan in regel 90 weer iets nieuws.
Wat gebeurt er in regel 90, achter "INPUT X"?

Hier staat:

```
T = T + 1
```

En niet te vergeten in regel 60:

```
T = 0
```

Iedere keer wanneer er een getal in de computer wordt ingevoerd moeten we dat tellen.

Wanneer het getal geraden is, kan de computer vertellen in hoeveel keer je het getal geraden hebt.

Hiervoor gebruiken we de variabele T.
Allereerst wordt die in regel 60 op nul gezet.
Nu hoeft dat niet direkt, want wanneer we beginnen is elke variabele die we kiezen automatisch nul.
Maar vergeet niet, dat de teller iedere keer nadat het getal geraden is, weer op nul moet worden gezet.

Die teller zelf staat in regel 90.
Zoals gezegd, direkt achter het INPUT-commando.
Elke keer als deze regel gepasseerd wordt zal de variabele T met 1 worden verhoogd.

Hoe?

We beginnen met te zeggen dat T nul is:

```
T = 0
```

Als je nu zegt:

```
T = T + 1
```

dan staat hier eigenlijk:

```
T = 0 + 1
```

Dus wordt T 1 want 0 + 1 is nog altijd 1.

Als we nu weer een getal invoeren dus als we nog een keer moeten raden, zal T weer met 1 opgehoogd worden:

```
T = T + 1
```

wordt dan:

```
T = 1 + 1
```

En hierdoor wordt T dus 2.

Als het getal nu geraden is, zal de computer dit dus netjes melden aan de speler en wel in regel 160.

Nog een tip voor ons, programmeurs.

Wanneer je zo'n programma als dit hebt ingetypt, moet het natuurlijk eerst getest worden.
Maar stel dat je niet zo goed bent in raden, hoe kun je dan die laatste paar regels bekijken waarin staat dat je het getal geraden hebt?

Voeg deze regel maar eens aan het programma toe:

```
85 LOCATE 30,6:PRINT A
```

Als je nu het programma start, zie je direkt het te raden getal.
Zo kun je dus testen of het programma goed werkt.
Dat moet je ook altijd doen.

Na de start zal de computer bijv. 56 in gedachten nemen.
Dank zij de ingevoegde regel 85 weten we dit nummer ook.
Je raadt nu 55.
De computer zal nu als het goed is "hoger" melden.
Zo niet, dan zal waarschijnlijk ")" in plaats van "(" gebruikt zijn in regel 20.
Vervolgens raad je 57.
Als je dan "lager" op je scherm krijgt, zit je dus helemaal goed.
Nu raad je 56 en als alles precies zo ging als hiervoor is gezegd, zal de computer melden:

```
    goed geraden
en dat in 3 keer
```

Als alles goed werkt, vergeet dan niet regel 85 weer te verwijderen.

Weet je nog?

Typ 85, druk op return, en de regel is weer weg.

Een heel kort programma is het volgende:

```
10 REM dobbelsteen
20 REM door w. dobbel
30 REM op MSX
40 KEY OFF
50 CLS
60 A = INT (6*RND(1)+1)
70 LOCATE 10,10:PRINT"ik gooide een ";A
80 LOCATE  5,20:PRINT"typ een toets"
90 K$=INKEY$:IF K$=""THEN 90
100 GOTO 50
```

Het staat nu wel mooi op het scherm: "Ik gooide een 5".
Maar echt de stippen tellen is leuker, vind je niet?

Typ het volgende programma maar eens in.
Vergeet niet het eerst te "SAVEN", weet je nog?

```
10 REM    dobbelsteen
11 REM
12 REM      op MSX
13 REM
14 REM door d.dobbelaar
15 REM
16 REM--------------------
20 KEY OFF :WIDTH 37
30 CLS
40 LOCATE 0,10
50 A = INT (6*RND(1)+1)
60 ON A GOTO 100,200,300,400,500,600
70 LOCATE 0,20
```

```
80 LINE INPUT "druk op return";R$
90 GOTO 30
100 PRINT "    "
110 PRINT " O "
120 PRINT "    "
190 GOTO 70
200 PRINT "O   "
210 PRINT "    "
220 PRINT "  O"
290 GOTO 70
300 PRINT "O   "
310 PRINT " O "
320 PRINT "  O"
390 GOTO 70
400 PRINT "O O"
410 PRINT "    "
420 PRINT "O O"
490 GOTO 70
500 PRINT "O O"
510 PRINT " O "
520 PRINT "O O"
590 GOTO 70
600 PRINT "O O"
610 PRINT "O O"
620 PRINT "O O"
690 GOTO 70
```

Ook in dit programma weer iets nieuws.

```
ON A GOTO 100,200,300,400,500,600
```

" ON-GOTO" (want zo heet het commando) betekent "ga" bij. . . naar. . .

Hiermee kunnen we dus een uitgebreide keuze maken, waar het programma verder moet gaan.

Het is eigenlijk net zoiets als IF. . . THEN. . .

Kijk maar.

```
60 IF A = 1 THEN 100
61 IF A = 2 THEN 200
62 IF A = 3 THEN 300
63 IF A = 4 THEN 400
64 IF A = 5 THEN 500
65 IF A = 6 THEN 600
```

Alleen wordt het programma dan wel erg lang, vind je niet?

Beter dan toch maar

```
ON A GOTO 100,200,300,400,500,600
```

In bovenstaand voorbeeld gaat de computer, als A 3 is naar regel 300, als A 5 is naar regel 500 enz.

Die regelnummers 100, 200, 300, 400, 500, en 600 zijn bewust gekozen omdat je hiermee direkt de waarde van de dobbelsteen in het programma kunt herkennen.

Heb je bijvoorbeeld een foutje gemaakt in de "5" dan moet je dus zoeken in de regels 500, 510, 520, en 590.

Verder spreekt dit programmaatje als het goed is vanzelf.

Het rekenprogramma

Met wat we nu geleerd hebben kunnen we ons rekenprogramma een heel stuk uitbreiden.

Behalve optelsommen, kunnen we de computer bijvoorbeeld ook vermenigvuldigingen, delingen en aftreksommen laten maken.

Hiervoor moeten er dan echter nogal wat wijzigingen in ons programma worden aangebracht.
Het beste is daarom het hierna geplaatste programma in zijn geheel in te typen.

Maar wat moeten we precies gaan doen?

Ten eerste moeten we diegene die met het programma gaat werken, bijv. je vriendje, laten kiezen tussen optellen, aftrekken, delen of vermenigvuldigen.

Ten tweede moeten we zorgen dat er geen fouten ontstaan wanneer hij verkeerde toetsen typt.

Om met het eerste te beginnen, we kunnen hem laten zien door middel van een tekst die op het scherm komt, welke mogelijkheden er in ons programma aanwezig zijn.

Zo'n tekst heet een menu.

Een menu is dus een stukje tekst dat op het scherm verschijnt en waarop de gebruiker kan zien waaruit hij kan kiezen.
Net als een menu-kaart in een restaurant.

Jij, de programmeur, moet nu zorgen dat zo'n menu op de juiste manier op het scherm komt te staan.

Ons menu gaat er als volgt uitzien:

```
Rekenoefeningen

Je kunt kiezen tussen


1..optellen
2..aftrekken
3..vermenigvuldigen
4..delen

5..stoppen


maak je keuze

typ 1,2,3,4 of 5
```

De gebruiker kan niet alleen kiezen uit optellen, aftrekken vermenigvuldigen of delen, maar hij kan ook stoppen.

En zo hoort het ook.
Je programma is pas volledig als er ook een eind aan zit.

Voor alle duidelijkheid geven we ook nog aan wat de gebruiker precies moet doen:

```
maak je keuze

typ 1,2,3,4 of 5
```

Als je namelijk niet laat weten wat hij moet doen zou hij bijvoorbeeld intypen:

```
ik wil optelsommen maken
```

en dat is echt niet nodig.

Je begrijpt misschien al met welk commando we de computer met het ingevoerde cijfer 1, 2, 3, 4 of 5 laten werken.

Juist, met ON. . GOTO.

Als de gebruiker een 1 kiest, gaan we naar de regels waar de optelsommen worden verzorgd.
Bij een 2 gaan we naar de regels waar de aftreksommen worden gemaakt en bij de 3 krijgen we de vermenigvuldigingsregels. 4 laat de computer delen en bij een 5 tenslotte, zal de computer het programma beëindigen.

Maar stel dat je vriendje leuk wil zijn en hij typt een letter in plaats van één van de getallen 1, 2, 3, 4, of 5.
Wat dan?
Ook daarvoor hebben we een oplossing.
We maken het programma zo, dat iedere toets anders dan 1, 2, 3, 4 of 5 door de computer wordt geweigerd.

Hoe doen we dat?

Eerst laten we de computer kijken naar het ingetoetste teken. Dat wordt met INKEY$ overgebracht naar K$.
Vervolgens geeft de computer het getal dat in K$ zit door aan de variabele K.
Dat kan echter niet zomaar.

K$ is een stringvariabele en K is een gewone variabele.
Zoals je misschien nog weet is een cijfer in een stringvariabele eigenlijk geen getal maar een leesteken, en in een variabele mogen nu eenmaal alleen maar cijfers staan.

```
K$ = "1234"
K  =  1234
```

We moeten dus het leesteken, bijvoorbeeld 3 omzetten in een cijfer 3. Dat kan met de funktie "VAL", dat "value" = "waarde" betekent.

Ook "funktie" is een nieuw woord.

Funktie wil zeggen dat met dit woord of gegeven de computer kan rekenen.

Een funktie is eigenlijk voluit een funktionele bewerking. Ook INT en RND zijn funkties.

Met deze funktie "VAL" kijkt de computer of hij het ingetoetste teken of karakter kan "vertalen" naar een getal. Als dat zo is dan krijgt de variabele K dat getal.
Zo ziet het er uit:

```
240 K$=INKEY$:IF K$ = "" THEN 240
255 K= VAL(K$)
```

Alleen, als onze vriend nu bijv. een 7 intypt dan zal de computer de variabele K met 7 vullen want dat is tenslotte een cijfer.
Maar, er mag alleen een 1, 2, 3, 4 of 5 worden ingetypt, dus zijn we er nog niet.

Alles groter dan 5 is verboden, maar ook alles kleiner dan 1. Dat betekent dat, wanneer we het volgende typen, ons programma volledig is beveiligd tegen verkeerd intypen.

```
IF K <1 OR K >5 THEN 230
```

Wanneer je nu iets anders intypt dan 1, 2, 3, 4 of 5, zal de computer terug stappen naar regel 230 en weer wachten tot er iets wordt ingetypt.

Probeer altijd je programma's zo te maken dat de gebruiker geen fouten kan veroorzaken, waardoor je programma stopt met een "BREAK IN ...".

Door dit soort "beveiligingen" maak je je programma's net iets beter dan van diegenen die dat niet doen.

Bedenk, dat als er iemand achter jouw computer zit, hij misschien niet weet hoe een computer werkt en dus snel dit soort fouten maakt.

Dan is er nog een probleem.

Zoals je weet maakt de computer twee willekeurige getallen. Deze getallen worden bewaard in de variabelen A en B.

Bij optellen en vermenigvuldigen is er niets aan de hand maar wat gebeurt er als je de volgende aftreksom krijgt:

```
3 - 8 = ?
```

Dat kan niet, natuurlijk,
Als we echter de twee getallen verwisselen is alles opgelost.

Hoe doen we dat?

Allereerst kijken we hoe het met elke computer kan.

Zoals je ziet, is in ons voorbeeld, A, 3 en B, 8.
Dus A is kleiner dan B, of A ⟨ B.

Als A kleiner is dan B, willen we de getallen omdraaien.

De regel waarin we A en B verwisselen zou er dan als volgt uitzien:

```
IF A < B THEN X = A : A = B : B = X
```

Waarom?

Om de twee variabelen te verwisselen moeten we eerst variabele B in een hulpvariabele opbergen.
In ons voorbeeld X.
Dat moet omdat anders de ene variabele de waarde van de andere krijgt en zo zijn eigen waarde verliest.
Moeilijk?
Even een voorbeeld dan, waardoor het allemaal wat duidelijker wordt.

Stel, je wilt een glas melk verwisselen met een beker water.
Dat kan alleen als je eerst de melk of het water in iets anders bewaart.

Je doet eerst bijv. het water in een kopje, vervolgens de melk in de beker, dan het water in het glas.
Zo heb je de twee vloeistoffen verwisseld met behulp van een kopje (je hulpvariabele).

Gelukkig is er met onze MSX computer een veel eenvoudiger oplossing voor dat verwisselen, en wel het commando "SWAP" dat "verwisselen" betekent.
De volgende regel is al voldoende om dat hele verhaal verder maar gewoon te vergeten.

```
IF A < B THEN SWAP A,B
```

Voor wat het delen betreft, hier passen we een truukje toe om steeds netjes uit te komen, want, als we zonder meer de variabelen A en B zouden gebruiken dan krijgen we wel erg moeilijke delingen te verwerken.

Wat dacht je bijvoorbeeld van deze delingen?

```
3 : 5 = ?
9 : 7 = ?
4 : 3 = ?
```

Als we nu eerst de twee variabelen vermenigvuldigen en we stoppen de uitkomst in een hulpvariabele Q dan wordt de zaak ineens een stuk beter.

Een voorbeeld:

```
Q = A * B
```

Als A = 4 en B = 6 dan wordt Q: A x B = 4 x 6 = 24

Wanneer we nu als som op het scherm zetten:

Q:B, dan komt er te staan, 24 : 6 = ?

We laten de computer gewoon uitrekenen wat Q gedeeld

door B is en bergen dit antwoord op in de variabele C.
Hierna controleert de computer of ANT gelijk is aan C en geeft vervolgens de uitslag.

Zoals je ziet krijg je nu steeds mooi uitkomende deelsommen.

Met al deze lessen moet het rekenprogramma eigenlijk geen problemen meer opleveren.

Sterker nog, je moet in staat zijn dit programma naar eigen ideeën uit te breiden.
Denk bijvoorbeeld maar eens aan puntentelling of een leuk begin van het programma.

Houd echter steeds het volgende in de gaten. SAVE je programma voor je het de eerste keer runt.
Als je je programma verandert, best, maar zet het dan op een ander bandje dan je origineel, zodat je altijd nog terug kunt als blijkt dat je veranderingen geen verbeteringen waren.

```
100 ' rekenprogramma
110 ' gemaakt door:
120 ' w.wortel
130 ' op een msx computer
140 CLS:KEY OFF
150 LOCATE 8, 1:PRINT "rekenoefeningen"
160 LOCATE 6, 4:PRINT "je kunt kiezen tussen:"
170 LOCATE 6, 8:PRINT "1..optellen"
180 LOCATE 6, 9:PRINT "2..aftrekken"
190 LOCATE 6,10:PRINT "3..vermenigvuldigen"
200 LOCATE 6,11:PRINT "4..delen"
210 LOCATE 6,13:PRINT "5..stoppen"
220 LOCATE 6,17:PRINT "maak je keuze"
230 LOCATE 6,19:PRINT "typ 1,2,3,4 of 5"
240 K$=INKEY$:IF K$=""THEN 240
250 K=VAL(K$):IF K<1 OR K>5 THEN 230
260 CLS
270 'maken getallen
```

```
280 A=INT(10*RND(-TIME)+1)
290 B=INT(10*RND(-TIME)+1)
300 IF A<B THEN SWAP A,B
310 LOCATE 10,7
320 ON K GOTO 330,370,410,450,560
330 PRINT A;"+";B;"=";
340 INPUT ANT
350 C=A+B
360 GOTO 490
370 PRINT A;"-";B;"=";
380 INPUT ANT
390 C=A-B
400 GOTO 490
410 PRINT A;"*";B;"=";
420 INPUT ANT
430 C=A*B
440 GOTO 490
450 Q=A*B
460 PRINT Q;":";B;"=";
470 INPUT ANT
480 C=Q/B
490 LOCATE 13,15
500 IF ANT = C THEN PRINT "goed"ELSE PRINT "fout"
510 LOCATE 4,18:PRINT"wil je nog een zelfde som?"
520 LOCATE 12,20:PRINT"j of n"
530 K$=INKEY$:IF K$=""THEN 530
540 IF K$="n" OR K$="N"THEN RUN
550 GOTO 260
560 LOCATE10,10:PRINT"tot ziens."
570 FORX=1 TO 10000:NEXT
580 CLS
```

Foutmeldingen

Tot slot nog even dit.

Wanneer je zo met je computer bezig bent zul je vast wel eens dingen op het scherm gezien hebben waarvan je zegt: "wat is dat nu weer?"

Wanneer je iets doet wat niet mag van de computer zal hij op zijn manier protesteren.

Een van de meest voorkomende kreten waar we het in het begin al eens over gehad hebben is wel:

SYNTAX ERROR

Die heb je zo op het scherm.
Iedere spelfout wordt bestraft met deze tekst.
Het betekent dan ook: "spelfout".

Bij het laden van een programma heb je vast weleens:

VERIFY ERROR

gezien.
Dat wil zeggen: "kontroleer fout".

In totaal zijn er om precies te zijn 59 foutmeldingen mogelijk. Zoveel fouten kun je onmogelijk nu al maken dus zullen we ons in dit boek beperken tot de fouten die je redelijkerwijs kunt veroorzaken.

Iedere foutmelding heeft een nummer.
In een aantal BASIC- handboeken staan alle foutmeldingen met dit nummer vermeld.
Sommige op nummervolgorde, andere alfabetisch.

Hoewel je thuis misschien al zo'n boek hebt is het toch handig in dit boekje de meest voorkomende foutmeldingen te plaatsen.

Belangrijke MSX-foutmeldingen op alfabetische volgorde

17 can't continue	Wanneer je probeerde met "cont" het programma te vervolgen terwijl de computer dit niet kan.
01 next without for	geprobeerd werd een NEXT zonder FOR uit te voeren.

07 out of memory	er werd geprobeerd een commmando uit te voeren waardoor er geen geheugenruimte overbleef.
14 out of string- space	hetzelfde als hierboven maar dan voor het string-geheugen.
03 return without gosub	geprobeerd werd return uit te voeren zonder door gosub te zijn gestart. (gosub wordt in deel twee behandeld)
02 syntax error	spelfout.
13 type mismatch	een letter of cijfer werd verwisseld.
08 undefined line-number	een niet bestaand regelnummer werd aangeroepen.
20 verify error	het programma op de band is niet gelijk aan het programma in de computer.

Waar staat wat en wat betekent het?

naam	uitleg	pagina
PRINT	Schrijven op het scherm	15
CLS	Maakt het scherm schoon	22
KEY OFF/ON	Tekst functie-toetsen uit/aan	24
LIST	Programma bekijken	25
RENUM	Hernummeren	26
INPUT	Invoeren via toetsenbord	29
CSAVE	Programma bewaren op band	35
CLOAD	Programma laden vanf band	35
CLOAD?	Programma controleren	36
IF-THEN-ELSE	Als-dan-anders	40
DELETE	Verwijderen	41
GOTO	Ga naar	42
RND	Willekeurig getal maken	43
INT	Afkappen van een getal	44
NEW	Haalt alles uit het geheugen	52
LINE INPUT	Invoeren van gegevens	57
LEN	Lengte	58
INKEY$	Opnemen ingetoetst karakter	61
REM	Opmerking	64
LOCATE	Plaats bepalen	67
WIDTH	Breedte scherm	68
FOR-NEXT-	Doe tot een eindwaarde is bereikt	74
STEP	met stappen van ...	76
TAB	Tabulator	80
PRINT USING	Schrijf op het scherm op bepaalde manier	81
TIME	Variabele voor teller	85
ON-GOTO	Ga bij bepaalde waarde naar ...	92
VAL	Waarde	98
SWAP	Verwissel	99

Nederlandstalige MSX handboeken

MSX BASIC handboek voor iedereen, door A.C.J. Groeneveld
Een kompleet nederlandstalig handboek voor iedere MSX computergebruiker. Dit handboek omvat een volledige behandeling van het MSX-basic in het Nederlands. Het handboek geeft een antwoord op elke vraag die een programmeur, van welke scholing ook, over het MSX-basic zou kunnen stellen. De volledige syntaxisbehandeling rekent af met onzekerheden of een bepaalde schrijfwijze nu wel of niet is toegestaan. De duidelijke beschrijving geeft per sleutelwoord aan, welke de funktie hiervan is. De laatste mogelijk nog aanwezig onduidelijkheden worden vervolgens door de opgenomen, zinvolle voorbeelden weggenomen ISBN 90 6398 1007

MSX ZAKBOEKJE door Wessel Akkermans
Een vlot geschreven naslagwerk na of naast het handboek. U vindt er o.a. in: niet computergerichte tabellen; de MSX-BASIC instructieset; diverse tabellen die het BASIC-programmeren kunnen versnellen; de Z80 instructieset; hardware-gegevens (connectoren) en een aantal programmaatjes ISBN 90 6398 888 5

MSX DISK handboek voor iedereen, door A.C.J. Groeneveld
Handboek voor diskdrivebezitters om naast het grote handboek te gebruiken. Een zeer volledige behandeling van het disk-gebeuren zelf en de specifieke disk kommando's, uitgebreid met voorbeelden, tabellen en overzichten. Het handboek is aangevuld met interessante programma's, waaronder een tekentafelprogramma en een basisprogramma voor basisonderhoud ISBN 90 6398 407 3

MSX PRAKTIJKPROGRAMMA'S door Wessel Akkermans
Praktische programma's met waar nodig eerst een stukje theorie. Erg handig bij het maken van uw programma's. Een greep uit de onderwerpen: priemgetallen; zoeken en sorteren; trefwoordenlijsten; converteren van getallen; enz. ISBN 90 6398 437 5

MSX QUICK DISK handboek voor iedereen, door A.C.J. Groeneveld
Hèt handboek voor iedere QUICK DISK gebruiker. Uitvoerige behandeling van de sleutelwoorden aangevuld met duidelijke voorbeelden met listing ISBN 90 6398 254 2

MSX DOS handboek voor iedereen, door A.C.J. Groeneveld
Dit handboek geeft u op een heldere wijze een totaalbeeld van de mogelijkheden van het MSX-DOS. Ook is dit handboek voorzien van een inleiding op het begrip 'operating system' en dus echt een handboek voor iedereen ISBN 90 6398 674 2

MSX LEERBOEKEN
door Wessel Akkermans en Piet den Heijer
De serie MSX leerboeken geeft een komplete kursus MSX-BASIC programmeren, in drie delen. Deze leerboeken zijn gericht op de beginnende programmeur. De moeilijkheidsgraad van de leerstof wordt dan ook slechts geleidelijk hoger. De gebruikte voorbeelden zijn zo praktisch mogelijk gekozen. Hierdoor kunnen al in een vroeg stadium bruikbare programma's worden gemaakt. Dit zal de lezer/leerling er toe aansporen om verder te gaan. Aan het eind van ieder deel is een groot voorbeeldprogramma opgenomen. Dit programma laat zien waartoe de lezer/leerling na bestudering van het betreffende leerboek in staat zal zijn.
Bij ieder leerboek is een afzonderlijk –Opdrachten en uitwerkingen– boekje te verkrijgen. In deze boekjes staan, in volgorde van de hoofdstukken uit het leerboek, vragen en opdrachten met antwoorden en uitwerkingen. Een unieke serie leerboeken voor een ieder die meer over MSX wil weten en het betere werk met zijn computer wil maken.

MSX Basic leerboek deel 1 - ISBN 90 6398 649 1
Opdrachten bij deel 1 - ISBN 90 6398 596 7
MSX Basic leerboek deel 2 - ISBN 90 6398 769 2
Opdrachten bij deel 2 - ISBN 90 6398 556 8
MSX DOS leerboek deel 3 - ISBN 90 6398 519 3
Opdrachten bij deel 3 - ISBN 90 6398 516 9

MSX Verder uitgediept door H. Klopper
Eindelijk een Nederlandstalig boek over het altijd in de mist gehulde onderwerp – PEEKS EN POKES. In dit boek staan alle belangrijke RAM en VRAM adressen. De video chip en zijn registers worden volledig uitgelegd. Maar ook hoe men een machinetaal programma van cassette naar disk kan schrijven. Bovendien een diskloader utility en een uiterst geavanceerde programma beveiliging. Tenslotte zijn er een aantal interessante programma's opgenomen, waaronder een wereldkaart, waarmee verder kan worden geëxperimenteerd. Elke MSX gebruiker kan in dit boek iets van zijn gading vinden en nieuws leren.

ISBN 90 6398 447 2

MSX Machinetaal handboek door H. Klopper en M. Le Belle
Hoewel een MSX computer over een krachtig Basic beschikt, is het toch handig tijdens het programmeren de grondbeginselen van machinetaal te kennen. Daarvoor is dit boek een goede gids. De zaken worden niet puur theoretisch maar ook aan de hand van duidelijke voorbeelden, die direkt bruikbaar zijn, uitvoerig uitgelegd. Enkele onderwerpen zijn verder – scroll routine –machinetaal software (ook in disk Basic) op casette zetten –disassembler –Z80 essembler instrukties –lijst van ROM-routines –alle hook-adressen –bespreking van Basic tokens en een kompleet token-overzicht. Hèt handboek voor iedere MSX programmeur die zijn computer ten volle wil benutten.

ISBN 90 6398 735 8

MSX TRUUKS EN TIPS
door A.C.J. Groeneveld
Hoe laat ik de computer een cirkel arceren, hoe tover ik mijn computer om in een elektronisch orgeltje, hoe maak ik een mooie intro voor een spelletje. Allemaal vraagstukken die zich lastig laten programmeren maar die iedere MSX-er toch graag opgelost wil zien.
Dit boekje staat boordevol truuks en tips, allemaal in gewoon MSX basic geschreven. Bladerend door dit boek komt u tot de ontdekking dat er voornamelijk korte maar uiterst krachtige en bijzonder goed bruikbare routines zijn opgenomen. Dit boekje geeft kort maar krachtig een antwoord op al uw programmeervragen.

deel 1 ISBN 90 6398 900 8
deel 2 ISBN 90 6398 340 9
deel 3 ISBN 90 6398 910 5

SOFTWARE PLUS IN MSX

INTROTAPE MSX door A.C.J. Groeneveld
Heeft u nog maar net een MSX computer gekocht en wilt u graag weten wat de computer kan en hoe u hem kunt leren programmeren? Deze cassette introduceert MSX op een uiterst vriendelijke en onderwijzende manier. U krijgt instructies hoe u de computer aan moet sluiten en de tape laden. Daarna volgt een demonstratie van de mogelijkheden in MSX, zoals het tekenen van sprites en het werken met de driestemmige toongenerator. Het geheel wordt afgesloten met twee 'les' gedeeltes. In anderhalf à drie uur weet u wat de MSX computer is, wat hij kan, en heeft u haast ongemerkt al wat regels geprogrammeerd.

ISBN 90 6398 148 1

MSX SCRIPT door Ton Weijters
Een menugestuurde nederlandstalige tekstverwerker. Het programma is geschikt om efficiënt grotere of kleinere teksten te bewerken. Paginaindeling (regellengte, paginalengte, marge, inspringen, centreren, enz.) wordt door het programma verzorgd. Dit geldt ook voor de paginatelling, toptitel en het eventueel uitvullen van de regels. Ook corrigeren, zoeken, string-substitutie, blokken tekst verplaatsen, copiëren of verwijderen, onderstrepen en vet zetten, is mogelijk met dit programma.

ISBN 90 6398 189 9

MSX DRAWS door A.C.J. Groeneveld
Een tekenprogramma in MSX basic, waarmee u al binnen 10 minuten uw eerste tekening kunt maken. Draws werkt erg vriendelijk en maakt gebruik van alle grafische mogelijkheden van de MSX computer. U kunt met Draws zowel technisch als creatieve tekeningen maken. Het programma heeft een effectief bereik van ruim 30.000 bij 30.000 puntjes met mogelijkheden als lijnen, cirkels, krommen, inkleuren, vergroten, verkleinen, verschuiven, verdraaien en andere tekeningen invoegen

ISBN 90 6398 754 4

MSX2 Basic handboek
A.C.J. Groeneveld, ISBN 90 6398 221 6, 507 pagina's, prijs f 56,50
Wie denkt over de aanschaf van een MSX2 computer, of er al een heeft, kan niet zonder het 507 pagina's tellende MSX2 Basic handboek. Alles over MSX2 Basic, de grafische- en geluidsmogelijkheden en de computer zelf. Met 288 voorbeeldprogramma's.

MSX2 Disk/Dos Uitbreidingshandboek
A.C.J. Groeneveld, ISBN 90 6398 222 4, 172 pagina's, prijs f 37,50
Omvat een volledige behandeling van het MSX2 Disk Basic en het MSX DOS operating system, voorafgegaan door een zeer duidelijke inleiding tot de fenomenen disk en operating system. Verder praktische tabellen, duidelijke afbeeldingen en zinvolle voorbeelden. Een standaardwerk dat naast elke MSX2 Disk computer zou moeten liggen.

MSX2 Utility-Toepassingshandboek
A.C.J. Groeneveld, ISBN 90 6398 223 2, 144 pagina's, prijs f 29,75
Een verzameling programma's die voor elke MSX-er onontbeerlijk zijn. Een aantal van de mogelijkheden met deze programma's: bestandsonderhoud met lijstwerk in iedere vorm op schijf en tape, staaf- en taartdiagrammen, programma's samenstellen met sprites en geluidseffekten, binair manipuleren binnen blokken op schijf. Alle programma's in dit boek zijn geschikt voor zowel MSX als MSX2 computers.

De bedoeling van deze serie boekjes is je te laten zien wat programmeren is, maar vooral hoe je het zelf heel eenvoudig kunt leren.

Want dat je dat kunt is zeker.
Uiteraard moet er wel wat geleerd worden, maar de benodigde stof wordt stap voor stap behandeld.

Eerst maak je kennis met de computer, vervolgens met zijn taal, het MSX Basic.

Dan volgt het programmeren.
In de loop van dit boekje wordt een rekenprogramma opgebouwd met de geleerde Basic woorden.
Om bepaalde woorden te verduidelijken zijn een paar kleine programma's opgenomen met die betreffende woorden.

Niet alleen kinderen, maarook volwassenen die de weg naar zelf programmeren nog niet hebben gevonden, nodigen we uit dit boekje door te nemen en zo tot de konklusie te komen dat ook zij echte programmeurs kunnen worden.

ISBN 90 6398 084 1 uitgeverij Stark-Texel