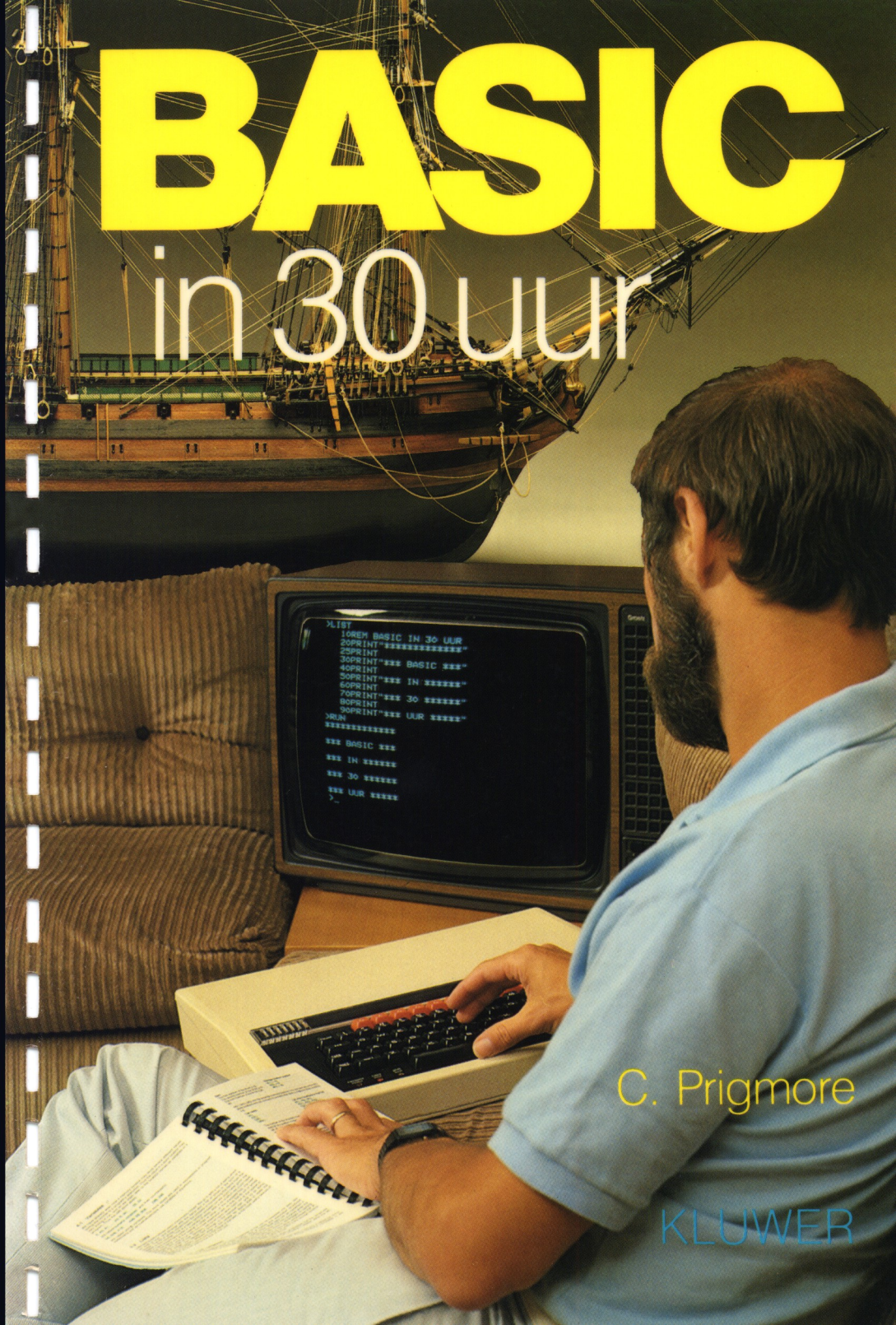
BASIC in 30 uur



C. Prigmore

KLUWER

BASIC in 30 uur

Clive Prigmore

BASIC in 30 uur

KLUWER TECHNISCHE BOEKEN B.V.
DEVENTER-ANTWERPEN

Vertaling: Gert Gremmen

ISBN 90 201 1680 0
D/1984/0108/168

Oorspronkelijke titel: 30 Hour BASIC
Uitgegeven bij: British Broadcasting Corporation

1e druk 1984
2e oplage 1984
3e oplage 1984

Inhoud

Hoe gebruik ik dit boek?

Doel van dit boek
Dit is nogal eenvoudig; u vertrouwd maken met het gebruik van uw (personal) microcomputer. Daartoe moet u drie dingen goed beheersen: (a) de taal BASIC; (b) opbouwen van gestructureerde programma's; (c) het gebruik van het toetsenbord. Dit boek leert u de eerste twee; uw computer wijst u de weg op het toetsenbord.

Heb ik een microcomputer nodig?
U kunt dit boek ook zonder computer bestuderen. Het boek is geschikt voor zelfstudie (a) met een microcomputer; doe alle oefeningen en beantwoord de Zelfstudie Opgaven en Problemen (ZOP's) en typ de programma's [K] in of (b) zonder microcomputer; in dit geval kunt u de gedeelten die zijn aangegeven met [K] overslaan.

Welke microcomputer heb ik nodig?
De programma's in dit boek kunnen op vrijwel elke computer worden uitgevoerd. U bent dus voor wat betreft uw keuze niet gebonden aan een bepaald type of merk. Als u een microcomputer gaat aanschaffen, laat uw keuze dan afhangen van factoren als geheugencapaciteit, mogelijkheden en uitbreidingsmogelijkheden. Om de programma's uit dit boek ook op een BBC-microcomputer goed te laten werken, treft u in een aantal programma's extra aanwijzingen aan.

Welke BASIC wordt in dit boek gebruikt?
Er zijn vele versies (dialecten) van BASIC in omloop, elke computerfabrikant heeft z'n eigen aangepaste versie. In dit boek wordt zoveel mogelijk gebruik gemaakt van MICROSOFT-BASIC. Soms bevatten de programma's instructies die niet op alle computers nodig zijn. Meestal worden die door deze computers genegeerd.

Opbouw van dit boek
Het boek is verdeeld in 9 hoofdstukken (zie Inhoud). Elk hoofdstuk bevat:
Voorbeelden: Problemen die in de tekst volledig worden opgelost en toegelicht.
Zelfstudie-opgaven en problemen (ZOP's): Vragen tussen de tekst, waarmee u kunt toetsen of u de zojuist behandelde stof hebt begrepen. Antwoorden op ZOP's vindt u steeds aan het einde van het hoofdstuk.
Oefeningen: Ingewikkelder problemen om uw tanden in te zetten. Antwoorden vindt u aan het einde van ieder hoofdstuk.
[K] : Staat voor een door u uit te voeren programma op uw eigen microcomputer. Vaak geeft dit een onverwacht grote steun.
Opgaven: Problemen met betrekking tot de toegepaste stof, vaak gestoeld op de praktijk. Antwoorden zijn in dit boek niet opgenomen.

1 Eenvoudige statements en commando's

1.1 Wat doet de computer precies?

In het algemeen kan worden gezegd dat een computer ons helpt met het oplossen van bepaalde problemen. De computer gebruikt daarbij symbolen of karakters die ons ook in het dagelijks gebruik bekend zijn namelijk de letters van het alfabet (hoofdletters en kleine letters), leestekens en enkele speciale symbolen als +, #, ★. De computer 'leest' een bepaalde groep symbolen of karakters. Dat lijkt allemaal nogal vaag, dus kijk eens naar de volgende voorbeelden:

Symbolen in	Symbolen uit
De maten van een kozijn.	De prijs van dubbel glas.
Lijst van geleende boeken.	Opgave van boeken die te laat zijn teruggebracht.
Een naam.	Een telefoonnummer.
Plaatscodering voor een schaakzet.	Een weergave van een schaakbord met de gedane zet.
Hypotheekbedrag en rentepercentage.	Maandlast.
Reeks codes.	Muzikale geluiden.

Afb. 1-1. Toepassingen van een computer.

Dit boek vertelt u niet hoe de computer dit doet maar vertelt u wel hoe u de computer zover krijgt. Hoewel we dus niet ingaan op de details van de computer als machine, geven we u wel een overzicht van de belangrijkste delen waaruit de computer is opgebouwd. Dit overzicht wordt gegeven in de volgende paragraaf.

1.2 Wat is een computer?

In afbeelding 1-2 is een eenvoudig model afgebeeld van een computer:

Afb. 1-2. Blokschema van een microcomputer.

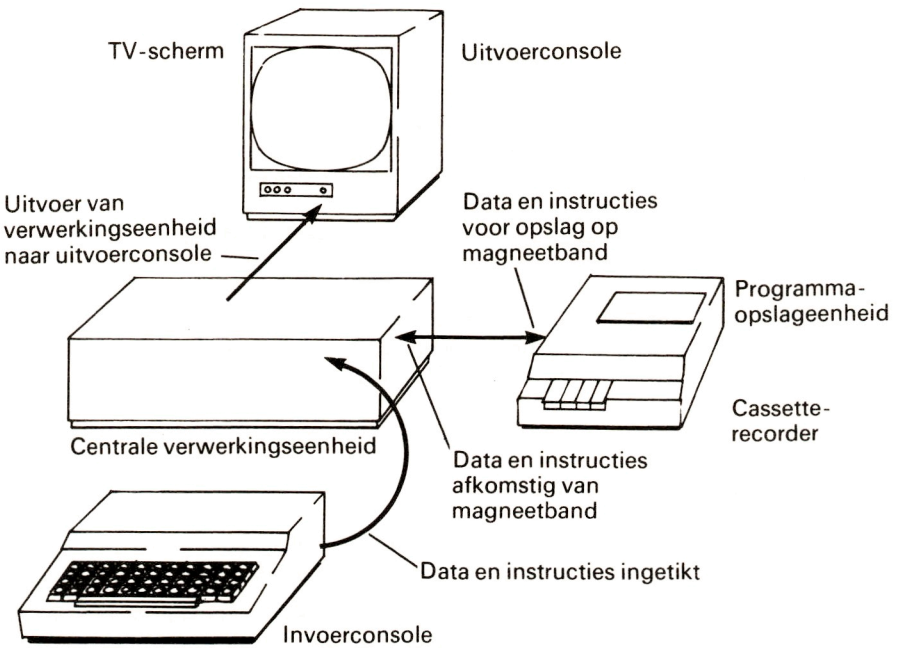
We kunnen drie belangrijke delen onderscheiden:

1 Een invoer-deel dat dient voor de ingave van instructies of data (gegevens) in de computer. Op een microcomputer ziet dat deel eruit als het toetsenbord van een schrijfmachine.
2 Een Centrale Verwerkings Eenheid of CVE (Eng. Central Processing Unit, CPU) die onder andere de instructies die via het toetsenbord zijn ingegeven, uitvoert. Deze instructies resulteren in een zodanige verwerking van de invoergegevens dat het 'antwoord' verschijnt.
3 Een uitvoer-deel dat de bewerkte informatie aan de gebruiker toont. Dit deel kan bijvoorbeeld een televisiescherm zijn, of een printer die de informatie op papier afdrukt.

Tot nu toe ziet het er allemaal nogal simpel uit. Dat zou ook zo zijn ware het niet dat de computer ook nog de volgende drie eigenschappen bezit. De computer kan (a) zeer grote hoeveelheden gegevens onthouden en deze (b) zeer snel verwerken en kan (c) een 'programma' onthouden dat de uit te voeren bewerkingen bevat en zo zijn eigen functie bepaalt. Deze laatste eigenschap is de belangrijkste en dit boek behandelt speciaal dit onderwerp.

Programma-opslag

Voordat we beginnen met programmeren, moeten we nog een technisch detail noemen. Als u dit boek, leest bent u hoogstwaarschijnlijk in het bezit van een personal computer met relatief klein geheugen dat bovendien alles 'vergeet' als de voedingsspanning wordt uitgeschakeld. Als u dus uw programma's wilt bewaren voor later



Afb. 1-3. Voorbeeld van een microcomputersysteem.

gebruik, dan heeft u een permanent geheugen nodig; een apart geheugensysteem dat naar behoefte aan de computer gekoppeld kan worden. Kleine systemen gebruiken voor dit doel meestal een audio-cassette en de wat grotere een magnetische schijfgeheugen.
In afb. 1-3 zijn de belangrijkste delen van een computersysteem afgebeeld:

1.3 Wat is BASIC?

Een computer is in feite een elektronisch apparaat dat patronen van elektrische signalen verwerkt. Als er al een te verwerken probleem zou zijn, zou u het niet kunnen invoeren als elektrisch signaal. Andersom levert de computer elektrische signalen die voor u, als gebruiker, onbegrijpelijk zijn.
Omdat dit erg lastig is, zijn er talen ontwikkeld die voor de gebruiker beter toegankelijk zijn. Zo'n programmeertaal is BASIC. Indien wij een programmaregel in de taal BASIC hebben ingetypt, zal deze 'vertaald' moeten worden in de taal die de microcomputer 'begrijpt': de machinetaal. Het programma dat voor deze vertaling zorgdraagt heet interpreter en is in vrijwel alle microcomputers standaard aanwezig.
Dit boek leert u BASIC, een hogere-orde-programmeertaal, waarmee iedere computer, die is uitgerust met een BASIC-interpreter, geprogrammeerd kan worden. BASIC staat, tussen haakjes, voor Beginners' All-purpose Symbolic Instruction Code. Vrij vertaald betekent dat: een symbolische instructiecode die door de beginner voor alle doeleinden gebruikt kan worden.

Het is belangrijk dat u de volgende stappen bij het programmeren van een micro-computer in BASIC goed in zich opneemt:

1 U heeft een probleem;
2 U deelt het probleem op in gedeelten die in BASIC geformuleerd kunnen worden;
3 U schrijft het probleem in een BASIC-programma uit;
4 U typt het programma op het toetsenbord van uw computer in;
5 De computer interpreteert uw BASIC-programma in zijn interne code en voert het uit;
6 De computer drukt de resultaten af zoals u dat gespecificeerd had in het programma.

Dit is alles wat u moet weten over het functioneren van uw computer. Vanaf nu is alles wat u van de computer wilt het ingeven van problemen en het ontvangen van de oplossing. Laten we eens naar een eerste, door de computer op te lossen probleem gaan kijken.

1.4 Een eenvoudig probleem

Het belangrijkste probleem bij het programmeren is het opdelen van het probleem in eenvoudige stappen, die door BASIC-instructies kunnen worden uitgevoerd.

Stelt u zich voor: u bent de computer en een kind vraagt u de som te berekenen van twee getallen. Al na korte tijd zal u gevraagd worden de som te bepalen van getallen, waarbij u niet zonder hulp van pen en papier de oplossing kunt vinden.

De volgende dialoog zou daarbij kunnen worden opgenomen:

KIND: 'START'
U: 'Geef mij het eerste getal'
KIND: '12157'
(U schrijft dit op een vel papier)
U: 'Geef mij het tweede getal'
KIND: '7896'
(U schrijft het tweede getal op het vel papier)
(U voert de optelling uit)
U: '20053'

We zouden het computerdeel in deze dialoog kunnen noteren als volgt:

1 Invoer: eerste getal;
2 Invoer: tweede getal;
3 Doe: optelling;
4 Uitvoer: resultaat.

Met behulp van een eenvoudige vergelijking hebben we een methode ter oplossing van het optelprobleem gevonden. Grofweg kunnen we stellen dat 1 en 2 te maken hebben met de invoer van getallen in de computer; dat 3 te maken heeft met een proces in de Centrale Verwerkings Eenheid en dat 4 de uitvoer van getallen vertegenwoordigt.

Hoewel we u nog geen woord 'BASIC' hebben geleerd, gaan we u toch de 'BASIC'-vertaling van het optel-probleem laten zien:

Voorbeeld 1
Schrijf een BASIC-programma om twee getallen in de computer in te voeren en de som ervan af te drukken.

Oplossing
De oplossing hebben we in feite al gevonden in de dialoog hierboven. In BASIC luidt deze als volgt:

```
10  INPUT EERSTE
20  INPUT TWEEDE
30  LET SOM = EERSTE + TWEEDE
40  PRINT SOM
50  END
```

Programma 1-1.

We hopen dat u, zonder al te veel voorstellingsvermogen, kunt aanvoelen hoe onze dialoog in een programma is omgezet. Een programma is een 'reeks van instructies samengesteld ter oplossing van een bepaald probleem door de computer'.

ZOP 1
We zijn nu aangeland bij het eerste punt in dit boek waar u uw begrip van de stof kunt testen. Deze ZOP's zijn zó opgesteld dat u onmiddellijk kunt vaststellen of u

de stof heeft begrepen. Heeft u alle antwoorden correct dan gaat u door met de volgende paragraaf. Heeft u er een fout, kijk dan in de laatste paragrafen nog eens na waar u de stof verkeerd heeft begrepen.

Maak de volgende zinnen af met de onder B gegeven antwoorden:

A
1 De CVE ...
2 De belangrijkste karakteristieken van een computersysteem zijn ...
3 Machinecode is ...
4 Machinecode is een voorbeeld van een ... programmeertaal.
5 BASIC is een .....programmeertaal.
6 Een BASIC-interpreter ...
7 Een computerprogramma is ...

B
a) lagere-orde
b) hogere-orde
c) verwerkt data volgens instructies, bestuurd zijn eigen systeem en stuurt de in- en uitvoerapparatuur.
d) een reeks van instructies of procedures voor de oplossing van een bepaald probleem.
e) dat deze grote hoeveelheden data kan opslaan, deze zeer snel kan bewerken volgens een in het geheugen opgeslagen methode.
f) vertaalt in BASIC geschreven code in machinecode.
g) een code die een directe relatie bezit met de in de computer voorkomende elektrische signalen.

1.5 Regelnummers

Laten we het optelprogramma nog eens wat nader bekijken:

```
10  INPUT EERSTE
20  INPUT TWEEDE
30  LET SOM = EERSTE + TWEEDE
40  PRINT SOM
50  END
```

Een programma is een reeks van instructies. In het bovenstaande programma is elke regel een instructie, dus:

```
10  INPUT EERSTE
```

is de eerste instructie van het programma, en

```
50  END
```

is de laatste instructie. Instructies in een programma worden ook wel 'statements' genoemd. Wij gebruiken beide begrippen door elkaar.

Instructies invoeren

Voorlopig gebruiken we maximaal één instructie of statement per regel. Bij het intikken van een instructie op een microcomputer wordt deze pas als beëindigd beschouwd na het intikken van de 'RETURN'-toets. Er gebeurt dus het volgende: U tikt 10 INPUT EERSTE en 'RETURN'. Daarna tikt u 20 INPUT TWEEDE en 'RETURN' enz.
Op het scherm ziet u:

```
10  INPUT EERSTE
20  INPUT TWEEDE
```

U zult hebben opgemerkt dat iedere regel begint met een nummer (1-9999). Deze nummers bepalen de volgorde waarin de instructies worden uitgevoerd. De uitvoering begint met de regel (instructie) met het laagste nummer, en daarna met het eerstvolgende hogere nummer enz.; totdat de computer anders wordt geïnstrueerd of totdat het einde van het programma is bereikt (hierover later meer).
Waarom, vraagt u zich nu af, was het programma niet als volgt geschreven?

```
1  INPUT EERSTE
2  INPUT TWEEDE
3  LET SOM = EERSTE + TWEEDE
4  PRINT SOM
5  END
```

Programma 1-2.

Dit zou inderdaad hebben gekund, want het programma zou uw wens identiek hebben vervuld!! U zult echter snel bemerken dat, als u begint te programmeren, u de behoefte heeft om hier en daar snel even een instructie tussen te voegen. Als de regels aansluitend genummerd zijn, is dat niet mogelijk zonder alle volgende regels aan te passen. Door het programma in stappen van 10 te nummeren beschikt u steeds over 9 (lege) regels tussen de instructies. De computer, zoals hiervoor beschreven, zoekt steeds naar het volgende hogere regelnummer en negeert niet aanwezige nummers. De aanwezigheid van 9 ongebruikte regelnummers beïnvloedt de uitvoersnelheid van het programma dan ook niet.

ZOP 2

Bestudeer de programma's 1-3 – 1-7 en geef aan welke programma's volgens u de juiste SOM van EERSTE en TWEEDE afdrukken.

(a)
```
11   INPUT EERSTE
59   INPUT TWEEDE
93   LET SOM = EERSTE + TWEEDE
401  PRINT SOM
500  END
```

(b)
```
23   INPUT EERSTE
32   INPUT TWEEDE
49   LET SOM = EERSTE + TWEEDE
40   PRINT SOM
50   END
```

(c)
```
10  INPUT EERSTE
20  INPUT TWEEDE
15  LET SOM = EERSTE + TWEEDE
40  PRINT SOM
50  END
```

(d)
```
100 INPUT EERSTE
200 INPUT TWEEDE
110 LET SOM = EERSTE + TWEEDE
190 PRINT SOM
220 END
```

(e)
```
100  INPUT EERSTE
50   INPUT TWEEDE
407  LET SOM = EERSTE + TWEEDE
902  PRINT SOM
1000 END
```

Programma's 1-3 – 1-7.

1.6 Programma-uitvoering en commando's

Het RUN-commando

Ons gemaakte programma willen we nu ook wel eens zien werken, dus:

```
10  INPUT EERSTE
20  INPUT TWEEDE
30  LET SOM = EERSTE + TWEEDE
40  PRINT SOM
50  END
```

en wat gebeurt er? NIETS!! De computer wacht nu op een commando om met de uitvoering van het programma te beginnen. Als u het programma wilt laten uitvoeren, geeft u het commando 'RUN'. Dit commando tikt u gewoon op de volgende regel:

```
10  INPUT EERSTE
20  INPUT TWEEDE
30  LET SOM = EERSTE + TWEEDE
40  PRINT SOM
50  END
RUN
```

(RUN heeft geen regelnummer: zie einde paragraaf.) Vervolgens drukt u op de 'RETURN'toets (ook wel met 'RET' of 'CR' aangegeven). Nu gaat uw programma van start. Er verschijnt een ? op het scherm; na een vraagteken verwacht de computer data. Tik uw eerste getal in en druk op 'RETURN'; een ? verschijnt op de volgende regel omdat de computer nu wacht op ingave van het tweede getal. Tik het tweede in en 'RETURN'. Uw computer drukt nu de som van beide getallen af. Onderstaand de volledige 'RUN' van uw programma:

```
10  INPUT EERSTE
20  INPUT TWEEDE
30  LET SOM = EERSTE + TWEEDE
40  PRINT SOM
50  END
RUN
? 12157
? 7896
20053
>READY
```

Afb. 1-4. Een volledige programma-'run'.

We gaan nu onderscheid maken tussen de invoer en de uitvoering van een programma. Terug naar de dialoog tussen u als computer met het kind. Een erg zorgvuldig kind zou u het volgende hebben verteld: 'Ik geef u twee nummers, die u moet opschrijven, optellen en dan vertelt u mij het antwoord'. Nu weet u precies wat u moet doen, hoewel u nog niets hebt gedaan. U kent de instructies, dus u bent geprogrammeerd. Het gesprek zou als volgt verder kunnen gaan:

KIND: 'START'
U: 'WAT IS HET EERSTE GETAL?'
KIND: '12157'
U: 'WAT IS HET TWEEDE GETAL?'
KIND: '7896'
U: '20053'

Nu heeft u de instructies uitgevoerd (RUN). Een programma is een reeks instructies voor de computer. Wanneer een programma wordt uitgevoerd, worden deze instructies opgevolgd.

Andere commando's: LIST, SAVE, LOAD
RUN is niet het enige commando dat u uw computer kunt geven met betrekking tot uw programma. LIST, SAVE en LOAD kunnen als volgt worden gebruikt.

LIST
U heeft bijvoorbeeld veel tijd besteed aan het intikken van uw programma en u heeft vele correcties gemaakt. Het zou nu handig zijn een volledig overzicht van uw programma te hebben. Na het intikken van LIST (en 'RETURN') verschijnt uw programma in de juiste volgorde op het scherm.

SAVE
Een volledig programma dat aan uw wensen voldoet, zult u zeker willen bewaren. Daartoe kunt u een kopie van het programma opslaan op cassette of op magneetschijf met het commando SAVE. (Daartoe moet op uw computer een schijveneenheid of cassetterecorder zijn aangesloten; zie ook de gebruiksaanwijzing bij uw computer).

LOAD
Een met SAVE op cassette of schijf gekopieerd programma kan met behulp van het commando LOAD worden 'teruggehaald' in uw computer.

Woorden als LIST, RUN, SAVE en LOAD besturen uw computer en worden commando's (Eng. Command) genoemd. Deze commando's worden door het BASIC-interpreter-programma 'begrepen' en direct uitgevoerd. Zij maken geen deel uit van BASIC als programmeertaal, hoewel ze wel min of meer zijn gestandaardiseerd. Een commando gebruikt een zelfstandige regel op het scherm en wordt niet voorafgegaan door een regelnummer. Het commando RUN na regel 50 uit het voorbeeld, doet het programma van start gaan.
Voorlopig weten we genoeg over commando's om verder te kunnen gaan met BASIC.

Computer- of Systeemantwoord
Nadat een commando succesvol is beëindigd, laat de interpreter ons dat weten met een boodschap op het scherm, meestal:
READY of
> of
OK of
K (niet te verwarren met [K] in dit boek)

BASIC-woorden
In het programma vindt u een aantal sleutelwoorden (Eng. Keywords), die een vaste betekenis hebben en de computer vertellen wat met die regel te doen (INPUT, LET, PRINT). Deze BASIC-woorden zijn afgeleid uit de Engelse taal en zó gekozen dat zij min of meer doen wat de normale betekenis aangeeft.

1.7 Programma-uitvoering en gegevens

U heeft ongetwijfeld bemerkt dat het optelprogramma geschikt is voor elk paar getallen. Voor het programma maakt het geen verschil welke getallen we intikken. Het is belangrijk dat u het onderscheid tussen een programma-als-een-reeks-algemene-instructies, en de getallen die tijdens de uitvoering van het programma moeten worden ingevoerd, goed leert kennen. Aan de hand van een voorbeeld kan dat nog iets verduidelijkt worden.
Een trainer roept voordat de wedstrijd begint nog even de hardlopers van zijn team bijeen om hen de laatste instructies te geven: 'langs de rechterkant van het veld tot in de verste hoek, het bruggetje over en dan linksaf de laan in'. Deze instructies zijn vergelijkbaar met een programma. Als de lopers hem verstaan en begrijpen, weten zij wat te doen; *maar ze staan nog op de startstreep*. Ze zijn nog niet gestart. Dit is te vergelijken met de invoer van een programma in de computer. De trainer roept vervolgens 'start' en de lopers gaan van start. Dit is te vergelijken met het RUN-commando dat de uitvoering van het programma doet beginnen.
Laten we de vergelijking nog even doortrekken en de wedstrijd als een speurtocht beschouwen. Veronderstel dat de trainer niet voldoende instructies had gegeven om de tocht af te maken maar iets had gezegd als: 'aan het eind van die laan vind je verdere instructies op een grote eik...' Deze instructies zouden op hun beurt de lopers de weg kunnen wijzen naar een volgende etappe, een nieuwe boodschap enzovoort, totdat de loop ten einde is. Deze boodschappen zijn vergelijkbaar met de invoer van data tijdens de loop van een programma. Dit voorbeeld verduidelijkt misschien uw inzicht in het onderscheid tussen het invoeren, het uitvoeren (runnen) en het ingeven van gegevens gedurende een programma.

ZOP 3
Onderstaand is een programma-run afgedrukt. Deze afdruk bevat BASIC-woorden, commando's, systeem-antwoorden en data. De verschillende delen kunnen bovendien benoemd worden als invoer van een programma, uitvoering daarvan en als listing

```
10  INPUT EERSTE
20  INPUT TWEEDE
30  LET SOM = EERSTE + TWEEDE
40  PRINT SOM
50  END
RUN
?  -37
?  -46
-83

LIST
10  INPUT EERSTE
20  INPUT TWEEDE
30  LET SOM = EERSTE + TWEEDE
40  PRINT SOM
50  END

RUN
?  12.83
?  48.95
61.78
```

Geef aan en benoem zo veel mogelijk van deze items.
BASIC-woorden met B
Commando's met C
Systeem-antw. met S
Data met D
Geef met accolades aan welke programmadelen betrekking hebben op invoer, uitvoering en listing.

1.8 INPUT, PRINT en LET

We hebben gezien dat een programma een lijst met instructies is, en we hebben u een indruk gegeven van enkele BASIC-statements. U had misschien al opgemerkt dat de drie statements INPUT, LET en PRINT corresponderen met de drie hoofdfuncties van een computer (invoer, CVE en uitvoer). Deze drie statements willen we nu wat nader bekijken.

INPUT
Het woord INPUT is een signalering aan de computer dat tijdens de uitvoering van een programma er behoefte zal bestaan aan een gegeven. Dat hebben we al gezien bij het optelprogramma toen de computer voorafgaand aan de optelling om EERSTE

en TWEEDE vroeg. Wat gebeurde er eigenlijk precies met EERSTE en TWEEDE. Het antwoord luidt dat deze zijn opgeslagen voor later gebruik in het programma in een opslagplaats genaamd EERSTE resp. TWEEDE.
Het woord EERSTE heeft in het programma twee functies: (a) het geeft aan dat de hier ingetikte gegevens het eerste ingevoerd zijn en (b) het wijst naar een plaats in het computergeheugen gemarkeerd met een label EERSTE, waarin de ingevoerde gegevens kunnen worden opgeslagen. Gedurende het gehele verloop van het programma blijft deze geheugenlocatie bestaan. Regel 10 in het optelprogramma betekent precies: wacht op intikken van een getal en sla dit op in een geheugenplaats; merk deze plaats vervolgens met 'EERSTE'

PRINT

Het statement 40 PRINT SOM doet ongeveer het omgekeerde als de statements 10 en 20. Het maakt het mogelijk gegevens die in de geheugenlocatie SOM zijn opgeslagen af te drukken op het scherm. De machine kopieert deze gegegevens op het scherm, maar vernietigt deze niet. Indien een printer op uw computer is aangesloten, resulteert dit statement ook werkelijk in een afdruk op papier; er bestaat echter geen verschil in gebruik voor beeldscherm of printer.

LET

30 LET SOM = EERSTE + TWEEDE is een voorbeeld van een toekenningsopdracht (Eng. assignment statement). Het is in dit soort opdracht dat de computer werkelijk aan het rekenen slaat. Zoals u ziet is dit een mengsel van geheugenlocatienamen (SOM, EERSTE en TWEEDE) en aritmetische operators (= en +). Van links naar rechts staat er:

'Laat de geheugenlocatie genaamd SOM gelijk worden aan de optelling van de waarden uit de locaties EERSTE en TWEEDE'.

Hoewel correct, zijn dergelijke uitdrukkingen makkelijker te lezen in omgekeerde richting:

'Tel de waarde van EERSTE bij die van TWEEDE op en berg het resultaat op in SOM'.

In het algemeen zijn toekenningsopdrachten van de volgende vorm:

LET geheugenlocatienaam = expressie
Bepaal de waarde van de uitdrukking (= expressie) aan de rechterzijde van '=' en sla deze op in de geheugenlocatie aan de linkerzijde van '='.

Een gemeen trekje van LET ...=... is dat het eenvoudig verward kan worden met in wiskundige vergelijkingen. Een voorbeeld. Veronderstel dat u een getal heeft opgeslagen in locatie L en dat u dat getal 5 groter wilt laten worden. In BASIC:

```
10  LET L = L + 5
```

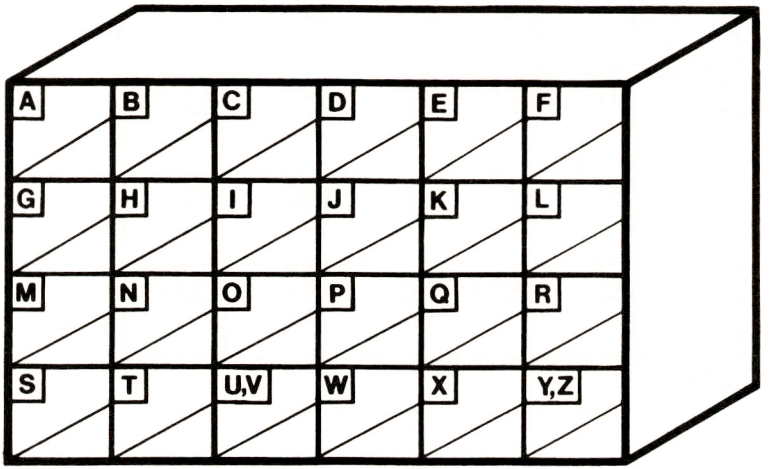
Dit betekent duidelijk *niet*:

```
L = L + 5
```

Er is geen L waarvoor dit geldig is. Wat het *wel* betekent is dat het LET-statement L gelijkmaakt aan de originele waarde van L plus 5.

1.9 Opslag van variabelen

Zoals we al eerder hebben opgemerkt: één van de belangrijkste karakteristieken van een computer is zijn vermogen grote hoeveelheden gegevens in het geheugen op te bergen. We zullen nu bekijken hoe BASIC zijn gegevens opslaat. Als u kijkt naar ons eerste programma en zich realiseert dat een computer slechts één ding tegelijk kan doen, dan is het logisch dat in regel 20 (waar een waarde voor TWEEDE wordt ingetikt) de waarde die in regel 10 voor EERSTE was ingetikt inmiddels ergens moet zijn opgeslagen. We kunnen ons dit geheugen met al zijn locaties voorstellen als een ladenkast waarbij elke lade voorzien is van een naam (label) en één geheugenlocatie voorstelt.



Afb. 1-5. Een model van de geheugenlocaties in een kleine computer.

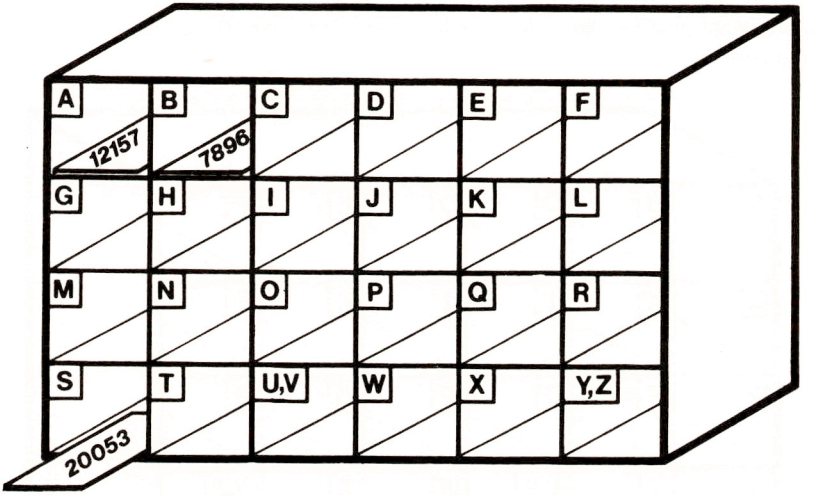
U ziet dat in dit model de namen van de geheugenlocaties zijn: A, B, C .... In het optelprogramma hadden we echter tot 6-letter-woorden gebruikt (TWEEDE). Dit leidt ons naar een van de verschillen tussen de BASIC-interpreters voor verschillende microcomputers. Niet alle laten zij een gelijk aantal karakters toe voor de benaming van geheugenlocaties. Sommige beperken het aantal karakters zelfs tot twee. De BBC-computer laat variabele-namen toe van onbegrensde lengte.

Benaming van geheugenlocaties
Het is duidelijk dat het voor de programmeur eenvoudiger is om locatienamen te gebruiken die hem herinneren aan de betekenis van de inhoud die daarin is opgeslagen. Daarom kozen wij in het optelvoorbeeld voor EERSTE, TWEEDE en SOM. Als we A, B en S hadden gebruikt had het voorbeeld er als volgt uitgezien:

```
10  INPUT A
20  INPUT B
30  LET S = A + B
40  PRINT S
50  END
```

Programma 1-8.

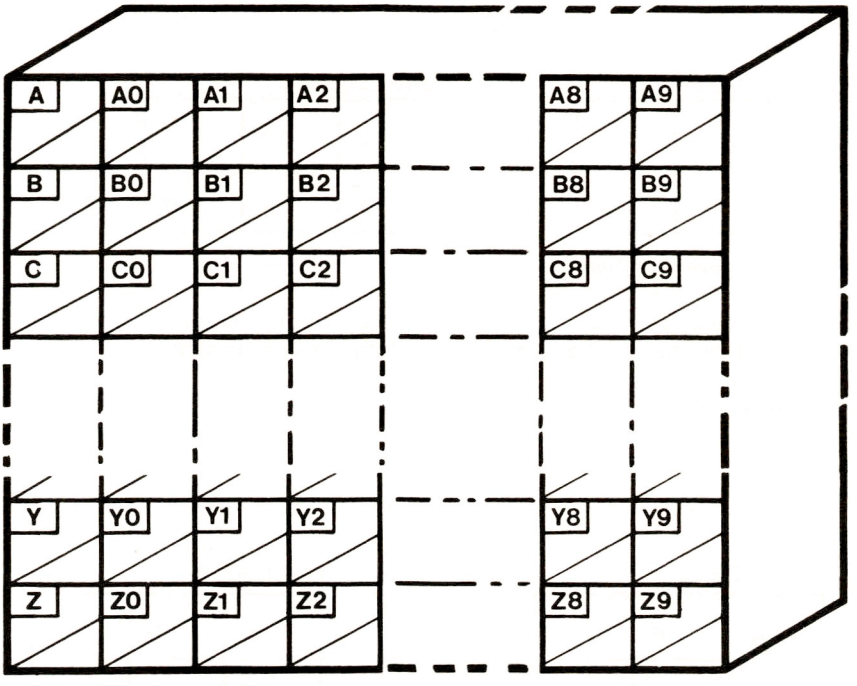
[K] Wanneer we bovenstaand symbool gebruiken, is dat om u erop te wijzen dat u bijgaand programma zelf op uw microcomputer kunt testen. Daartoe tikt u bovenstaande regels in, steeds gevolgd door RETURN en start het programma met RUN en RETURN. Uw computer vraagt u om een getal. Tik een getal in en RETURN. Nu vraagt uw computer om het tweede getal. Na dit te hebben gedaan, verschijnt het antwoord op uw scherm. Stel dat we de getallen 12157 en 7896 hadden ingevoerd dan waren de volgende geheugenlocaties als volgt gevuld:



Afb. 1-6. Geheugen na de uitvoering van programma 1-9.

Een universeel systeem van variabele namen.

Omdat op vele computers het gebruik van lange namen niet is toegestaan, wordt van nu af aan in dit boek gekozen voor namen met twee karakters nl. één letter gevolgd door één cijfer. Zodoende hebben we 286 verschillende locaties ter beschikking.

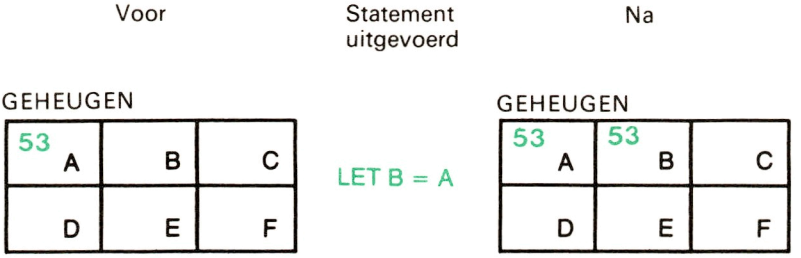


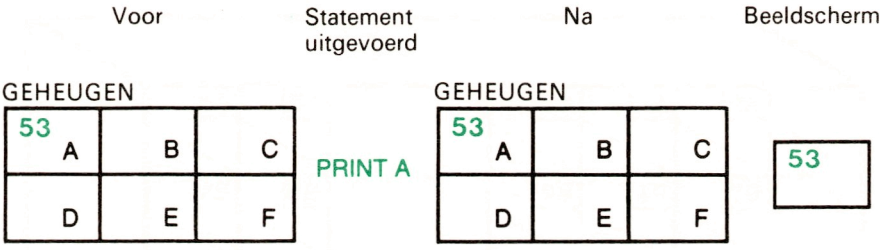
Afb. 1-7. 286 geheugenlocaties.

1.10 Kopiëren en overschrijven

BASIC-instructies kunnen een geheugenlocatie op twee manieren beïnvloeden: óf de inhoud daarvan verandert, óf deze verandert niet.

Kopiëren
Veronderstel dat het getal 53 is opgeslagen in locatie A. Wat gebeurt na LET B = A? En wat als PRINT A wordt uitgevoerd?



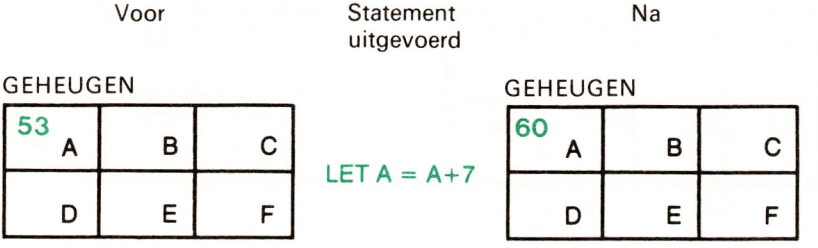


Afb. 1-8. Het gevolg van kopiëren.

In beide gevallen blijft de inhoud van A onveranderd. Net als een afschrift van uw girorekening: het afschrift kopieert uw tegoed maar beïnvloed dit niet.

Overschrijven

In dit geval bevat A nog steeds het getal 53, maar deze keer voeren we het statement LET A = A + 7 uit. Het resultaat is nu:



Afb. 1-9. Het gevolg van overschrijven.

Dit statement overschrijft de inhoud van geheugenlocatie A. Dat wil zeggen dat de originele inhoud verdwijnt en wordt vervangen door het resultaat van de berekening A + 7, in dit geval 60. Dit resultaat had natuurlijk op vele manieren bereikt kunnen worden b.v. LET A = 60.

ZOP 4

Geef aan welke van de volgende variabele-namen geldig zijn (volgens de regels op pag. 22).

(a) N3
(b) 3N
(c) W10
(d) B#
(e) QJ
(f) M
(g) M5
(h) M−5
(i) M+5
(j) U0

Geef, voorzover mogelijk, de reden aan van uw keus.

1.11 Aritmetische operators

Als u berekeningen uitvoert, gebruikt u verschillende operators. BASIC gebruikt dezelfde operators, maar zij worden iets anders geschreven:

Normaal symbool	Betekenis	BASIC-symbool
+	optellen	+
−	aftrekken	−
×	vermenigvuldigen	★
:	delen	/

ZOP 5

Schrijf de volgende acht uitdrukkingen met behulp van BASIC-symbolen. Waar haakjes gebruikt zijn, dient u deze ook in het antwoord te gebruiken.

(a) 3 + 7
(b) 3 × 7
(c) 8 : 4
(d) 5 × (2 + 8)
(e) 30 : (3 + 2)
(f) 24 − (4 × 3)
(g) 5 × 6 × 7
(h) 81 − (27 × 2)

ZOP 6

Als A de waarde 2 heeft, en B heeft de waarde 5 en C is 10; bereken dan de waarde van onderstaande uitdrukkingen:

(a) A + B + C
(b) A★B
(c) A★B★C
(d) C / A
(e) C / (B − A)
(f) A★A
(g) (B★C)/(B − A)
(h) (C − B) ★(C + B)

De berekening wordt feitelijk uitgevoerd in LET-statements, die de aritmetische eenheid (deel van de centrale verwerkingseenheid) opdraagt wat te doen. Dit kan met behulp van het volgende voorbeeld worden verduidelijkt:

Effect van Let A = B − C

LET A = B − C

GEHEUGEN BIJ AANVANG

A	15 B	10 C
D	E	F

Afb. 1-10.

Vergeet niet de uitdrukking van rechts naar links te 'lezen': neem het getal in locatie C; trek het van het getal in locatie B af en sla het resultaat op in A.

Resultaat van LET A = B − C

Merk op dat de geheugenlocaties B en C niet veranderd zijn.

GEHEUGEN NA BEREKENING

5 A	15 B	10 C
D	E	F

Afb. 1-11.

Effect van LET A = B ★ C

GEHEUGEN BIJ AANVANG

A	15 B	10 C
D	E	F

GEHEUGEN NA BEREKENING

150 A	15 B	10 C
D	E	F

Afb. 1-12.

ZOP 7

Vul de waarden van A, B en C in de hokjes in nadat elk van de volgende regels door de computer is uitgevoerd:

1. Programma

```
10  LET A = 12
20  LET B = 5
30  LET C = A★(A+B)
40  LET A = A+10
```

Waarde geheugenlocatie

A	B	C
☐	☐	☐
☐	☐	☐
☐	☐	☐
☐	☐	☐

Afb. 1-13.

2. Programma

```
10  LET A = 20
20  LET B = A★3
30  LET C = A/4
40  LET A = B+C
```

Waarde geheugenlocatie

A	B	C
☐	☐	☐
☐	☐	☐
☐	☐	☐
☐	☐	☐

Afb. 1-14.

Wat u zojuist heeft gedaan is niet moeilijk (hopen wij) en het wordt niet moeilijker als we verder gaan met het gebruik van ingewikkelder locatienamen. We moeten die namen ingewikkelder maken omdat het alfabet A, B, C ... Z ons slechts 26 geheugenlocaties zou toestaan. We voorzien de namen daarom van een index en we gebruiken nu dus als locatienamen A, A0, A1, A2 .....A9, B, B0.....Z9.

LET P4 = Q1 ★ R1

verschilt in geen enkel wezenlijk opzicht van

LET A = B ★ C.

P4, Q1 en R1 zijn gewone locatienamen. P4 is een locatienaam, net zoals 83-NJ-99 een autonummer is.

Voorbeeld 2
Schrijf een BASIC-programma dat twee getallen in de computer inleest en vervolgens de som, het produkt, het quotiënt en het verschil hiervan afdrukt.

Oplossing
Hoewel dit moeilijk mag lijken, hebben we de oplossing in feite al behandeld. We hadden het optelprogramma dat de som van twee getallen bepaalde al gemaakt. Het enige wat we dus hoeven te doen, is het veranderen van de aritmetische operator in regel 30:

```
30  LET SOM = EERSTE + TWEEDE
```

Voordat we dat doen, herschrijven we het programma met de nieuwe korte locatienamen.

Oude versie

```
10  INPUT EERSTE
20  INPUT TWEEDE
30  LET SOM = EERSTE + TWEEDE
40  PRINT SOM
50  END
```

Nieuwe versie

```
10  INPUT N1
20  INPUT N2
30  LET S = N1 + N2
40  PRINT S
50  END
```

Programma 1-9.

Het enige wat we nu nog moeten maken, zijn drie nieuwe versies voor regel 30:

10 INPUT N1	,,	,,	,,
20 INPUT N2	,,	,,	,,
30 LET S=N1+N2	LET V=N1−N2	LET P=N1★N2	LET Q=N1/N2
40 PRINT S	PRINT V	PRINT P	PRINT Q
50 END	,,	,,	,,

(We gebruiken S voor Som, V voor Verschil, P voor Produkt en Q voor Quotiënt).

Moeten we nu vier programma's schrijven? Gelukkig niet, want wanneer we berekeningen uitvoeren met N1 en N2 worden de inhouden niet overschreven, maar gekopieerd. Zodoende kunnen wij ze viermaal in één programma gebruiken.

```
10  INPUT N1
20  INPUT N2
30  LET S = N1 + N2
40  PRINT S
50  LET V = N1 − N2
60  PRINT V
70  LET P = N1 ★N2
80  PRINT P
```

oorspronkelijk somprogramma

extra regels voor verschil

extra regels voor produkt

```
90  LET Q = N1/N2
100 PRINT Q
110 END
```

extra regels voor quotiënt

Programma 1-10 Som, verschil, produkt en quotiënt van twee getallen.

[K] Tik programma 1-10 op uw computer in. Geef RUN en voer twee nummers in. U kunt het volgende verwachten:

```
RUN
? 57.82
? 19.11
76.93
38.71
1104.9402
3.02564102
```

1.12 Numerieke constanten

Eerder in dit hoofdstuk hebben we al gezien dat ons BASIC-programma kon rekenen met zowel gehele als decimale negatieve getallen. Op deze plaats willen we niet verder ingaan op hoe deze getallen door BASIC worden gerepresenteerd. We laten u daarentegen zien hoe ze te gebruiken.

Het statement LET P = 427 ★ R betekent: vermenigvuldig het getal in locatie R. met een nieuwe getal 427 en sla het resultaat op in locatie P. (Laat u niet afleiden door de wetenschap dat computers alleen met binaire getallen kunnen rekenen. De BASIC-interpreter zorgt ervoor dat we gewone decimale getallen kunnen gebruiken).

Het statement LET Y4 = 3.142 + Z8 telt het nieuwe getal 3.142 op bij de inhoud van Z8 en slaat dit in Y4 op. Op identieke wijze wordt in LET A = −49.93 / B het nieuwe getal −49.93 gedeeld door de inhoud van B en in A opgeslagen.

Oefening 1
Schrijf programma's in BASIC voor het uitvoeren van de volgende conversies:
(a) Voer de lengte van een spijker in inches als eerste getal in en druk vervolgens de lengte in centimeters af. Gegeven is dat één inch gelijk is aan 2.54 centimeter.
(b) De goudprijs wordt nog vaak in guldens per ounce uitgedrukt, één ounce is 28.375 gram. Dit programma moet het gewicht van een hoeveelheid goud in ounces omzetten in het gewicht in gram.
(Antwoorden bij oefeningen aan het eind van dit hoofdstuk).

Oefening 2
Elke conversie behelst: conversiefactor × te converteren getal. Schrijf een algemeen conversieprogramma dat eerst de conversiefactor inleest, vervolgens het te converteren getal en dat daarna de conversie uitvoert en het resultaat afdrukt.

1.13 Het REM-statement

Het statement REM is het 'REMARK'-statement. Het geeft ons de mogelijkheid een programma een titel te geven of om andere zinnige opmerkingen in het programma op te nemen. We kunnen bijvoorbeeld bij elke sectie van het programma opnemen wat dit deel precies doet. Het REM-statement wordt niet door de computer uitgevoerd; het wordt genegeerd evenals alles wat op die regel verder nog ingetikt was. Het volgende programma berekent percentages en maakt gebruik van het REM-statement om de titel van het programma te noteren; 10 REM ★★ PERCENTAGEBEREKENING ★★ (de ★★ hebben geen andere betekenis dan de tekst te benadrukken)

Voorbeeld 3
Schrijf een BASIC-programma dat twee getallen inleest en het tweede uitdrukt in een percentage van het eerste.
{percentage = (tweede : eerste) × 100}

Oplossing

```
10  REM ★★ PERCENTAGE BEREKENING ★★
20  INPUT E
30  INPUT T
40  LET P = (T/E) ★ 100
50  PRINT P
60  END
```

Programma 1-11 Percentageberekening.

```
RUN
? 57
? 74
129.824561

RUN
? 74
? 57
77.027027
```

[K] Programma 1-11

1.14 Ingewikkelder rekenwerk

We kunnen de computer nu gebruiken voor eenvoudig rekenwerk, ongeveer als een rekenmachine. We zullen eens bekijken hoe in BASIC ingewikkelder rekenwerk opgelost kan worden. Over het algemeen kunnen we in BASIC berekeningen net zo opschrijven als in algebraïsche notatie. We kunnen haakjes gebruiken om bepaalde waarden te groeperen en als BASIC deze tegenkomt in een expressie wordt het rekenwerk binnen de haakjes het eerst afgerond. Vervolgens worden vermenigvuldiging en deling en als laatste optellen en aftrekken uitgevoerd.

Deze volgorde is gelijk aan die welke we in het dagelijkse rekenwerk al gewend zijn. Later gaan we nog wat dieper op dit onderwerp in.

Voorbeeld 4
Schrijf de volgende expressies in BASIC-statements:

1. $AB + C$ 2. $A(B + C)$ 3. $\frac{A}{B + C}$

Oplossingen
1. A★B+C
   De volgorderegel leert ons dat eerst de vermenigvuldiging en pas daarna de optelling wordt uitgevoerd.
   (A★B)+C is ook goed, maar de haakjes zijn niet noodzakelijk.
2. A★(B+C)
   Merk op dat hier de haakjes wél nodig zijn.
3. A/(B+C)

ZOP 8
Schrijf de volgende uitdrukkingen als BASIC-expressies

1. ABC
2. $\frac{A\ B}{C}$
3. $\frac{A+B}{C}$

Oefening 3
Schrijf vervolgens een programma dat de waardes van A, B en C inleest en de expressies uit ZOP 8 uitrekent en afdrukt.

1.15 Tekst afdrukken

U heeft al gezien dat met behulp van het PRINT-statement de waarde van een geheugenlocatie kan worden afgedrukt. In de loop van dit boek zult u vaststellen dat dit PRINT-statement bijzonder handig is. Één van de functies van PRINT is het afdrukken van boodschappen of prompts op het scherm om de gebruiker aanwijzingen te geven. Iedere keer als het programma een INPUT-statement vermeldt, drukt de computer een ? op het scherm af. In lange programma's met veel invoer is een reeks vraagtekens op het scherm nogal verwarrend. U weet dan al gauw niet meer op welk INPUT-statement het ? betrekking heeft. Het afdrukken van een aanduiding met behulp van het PRINT-statement kan hier erg nuttig zijn.
Het is simpel om uw computer een boodschap of PROMPT op het scherm te laten afdrukken. Wat u nodig heeft is:

```
20 PRINT "BOODSCHAP"
```

Deze regel drukt:

```
BOODSCHAP
```

op een nieuwe regel op het scherm. Met andere woorden alles wat na het PRINT-

statement tussen aanhalingstekens " " staat, zal tijdens de uitvoering van het programma exact zo worden afgedrukt.
Let op:

```
20  PRINT "A + B"
```

geeft

```
A + B
```

op het scherm, maar de computer rekent de som van de waarden uit A en B niet voor u uit!!

Het volgende voorbeeld demonstreert het gebruik van PRINT " " als geheugensteun tijdens de uitvoer van het programma.

Voorbeeld 5
Schrijf een BASIC-programma dat de temperatuur in graden Celsius omzet in graden Fahrenheit.
(°F = 9/5 × °C + 32)

Oplossing

```
10  REM ★★ CELSIUS NAAR FAHRENHEIT ★★
20  PRINT "VOLGENDE TEMP. IN GR. CELSIUS"
30  INPUT C
40  LET F = (9/5) ★ C + 32
50  PRINT "DAT IS IN GRADEN FAHRENHEIT"
60  PRINT F
70  END
```

print boodschap gaat aan het input-statement vooraf zodat deze verschijnt voor de "?".

Programma 1-12 Temperatuurconversie.

```
RUN
VOLGENDE TEMP. IN GR. CELSIUS
? 16
DAT IS IN GRADEN FAHRENHEIT
60.8
```

[K] Programma 1-12.

Opgave 1

(Maak gebruik van REM en PRINT " " voor het documenteren en gebruiksvriendelijk maken van uw programma's)

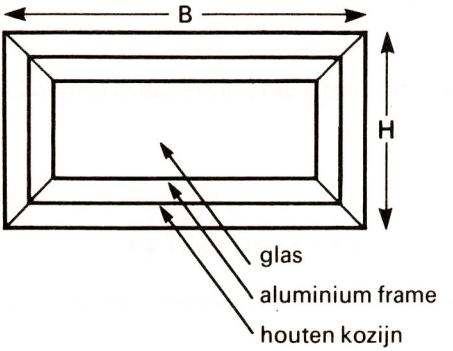
1. U belegt D gulden op een berekening met P procent rente per jaar. Uw rente kan dan worden berekend met de volgende formule:
   R = D × (P:100)
   (a) Schrijf een BASIC-programma ter berekening van de jaarlijkse rente.
   (b) Als de rente op uw rekening wordt bijgeschreven dan zal het volgend jaar de samengestelde rente zijn:

SR = (D + R) × (P : 100)

Breidt uw programma uit ter berekening van die samengestelde rente.

2. Beschouw het probleem van de berekening van de kosten van de installatie van aluminium ramen met dubbel glas. De ramen bestaan uit drie delen:
   (a) een hardhouten kozijn;
   (b) een aluminium frame;
   (c) glas.

   Als de hoogte van het raam H meter is en de breedte is B meter, dan is de totale lengte aan aluminium frame en hardhout kozijn 2 × (H × B) elk. De oppervlakte van het glas is dan (H × B) vierkante meter.



Afb. 1-15.

Schrijf drie afzonderlijke BASIC-programma's ter berekening van de kosten van:
(a) het kozijn (meterprijs hardhout f 12,–);
(b) het aluminium frame (prijs aluminium f 16,– per meter);
(c) het glas bij een vierkante-meterprijs van f 160,–.

Koppel vervolgens de drie programma's aaneen en bereken de totale prijs als het arbeidsloon per raam f 200,– bedraagt.

Samenvatting van hoofdstuk 1

Nu we het eerste hoofdstuk hebben afgerond, moet u met de opgedane kennis in staat zijn eenvoudige BASIC-programma's te schrijven met:

- Regelnummers ☐
- INPUT ☐
- LET ☐
- PRINT ☐
- Het gebruik van geheugenlocaties met een letter of een letter plus index als identificatie. ☐
- Kopiëren van een locatie in de andere ☐
- Overschrijven van een locatie
- ;, –, ★, / ☐

- ( ) ☐
- Numerieke constanten ☐
- REM ☐
- PRINT " " ☐

Het gebruik van:
- RETURN ☐
- RUN ☐

Hoe te reageren op:
- >READY ☐
- ? ☐

Antwoorden op ZOP's en oefeningen

ZOP 1

A	B
1	(c)
2	(e)
3	(g)
4	(a)
5	(b)
6	(f)
7	(d)

ZOP 2

(a) en (e) doen hetzelfde als programma 1.
(b) vraagt om SOM af te drukken voordat deze is berekend.
(c) en (d) berekenen SOM voordat TWEEDE is ingelezen.

ZOP 3

```
10  INPUT EERSTE ---------- B
20  INPUT TWEEDE ---------- B   } invoer
30  LET SOM = EERSTE + TWEEDE - B
40  PRINT SOM ------------- B
50  END

RUN ------------------------ C
? -37 ----------------------- S,D  } uitvoering
? -46 ----------------------- S,D
-83 ------------------------ S

LIST ----------------------- C
10  INPUT EERSTE
20  INPUT TWEEDE
30  LET SOM = EERSTE + TWEEDE   } listing
40  PRINT SOM
50  END
```

```
RUN                    C
? 12.83                S,D
? 48.95                S,D
61.78                  S
                                 uitvoering
RUN                    C
? 17.0009              S,D
? -29.2629             S,D
-12.2629               S
```

(Heeft u bemerkt dat dit programma ook met negatieve getallen en cijfers achter de komma werkt?)

ZOP 4

(a) Goed
(b) Fout, begint met een cijfer in plaats van een letter.
(c) Fout, twee cijfers ('1' en '0') is te veel.
(d) Fout, want # is geen toegelaten symbool.
(e) Fout, twee letters (werkt op bijv. BBC-computer wel, maar is tegen de afspraak).
(f) Goed.
(g) Goed.
(h) Fout, – is geen toegelaten symbool.
(i) Fout, + is geen toegelaten symbool.
(j) Goed.

ZOP 5

(a) 3+7 (b) 3★7 (c) 8/4 (d) 5★(2+8) (e) 30/(3+2)
(f) 24–(4★3) (g) 5★6★7 (h) 81–(27★2)

ZOP 6

(a) 17 (b) 10 (c) 100 (d) 5
(e) 10/3 of 3.33 of 3 1/3 (f) 4
(g) 50/3 of 16.66 of 16 2/3 (h) 75

ZOP 7

1.

12		
12	5	
12	5	204
22	5	204

2.

20		
20	60	
20	60	5
65	60	5

Oefening 1

(a)

```
10  INPUT L1
20  LET L2 = 2.54 ★ L1
30  PRINT L2
40  END
```

```
RUN
? 12
30.48
```
eerste RUN

```
RUN
? 36
91.44
```
tweede RUN

K Programma 1-13

Programma 1-13 Van inch naar cm.

(b)

```
10  INPUT W1
20  LET W2 = W1 ★ 28.375
30  PRINT W2
40  END
```

```
RUN
? 10
283.75
```
eerste RUN

```
RUN
? 50
1418.75
```
tweede RUN

```
RUN
? 16
454
```
derde RUN

K Programma 1-14

Programma 1-14 Van ounce naar gram.

Oefening 2

```
10  INPUT V
20  INPUT F
30  LET N = F★V
40  PRINT N
50  END
```

```
RUN
? 16
? 28.375
454
```
van ounce naar grammen

```
RUN
? 36
? 2.54
91.44
```
van inches naar cm

K Programma 1-15

Programma 1-15 Algemeen conversieprogramma.

ZOP 8

1. A★B★C
2. A★B/C [(A★B)/C is ook goed]
3. (A+B)/C

Oefening 3

```
10  INPUT A
20  INPUT B
```

```
30  INPUT C
40  LET R = A★B★C
50  PRINT R
60  LET R = (A★B)/C
70  PRINT R
80  LET R = (A + B)/C
90  PRINT R
100 END
```

Expressie 1 en druk af

Expressie 2 en druk af

Expressie 3 en druk af

Programma 1-16 BASIC-expressies.

Merk op dat in alle drie antwoorden R wordt gebruikt voor de opslag van het antwoord; in dit geval geeft dat niet omdat het antwoord afgedrukt wordt voordat de inhoud wordt overschreven.

```
RUN
? 13
? -27
? 55.2
-19375.2
-6.35869565
-0.253623188
```

```
RUN
? 13
? 13
? 13
2197
13
2
```

[K] Programma 1-16

2 Beslissingen nemen

2.1 Introductie

De programma's uit hoofdstuk 1 waren recht-toe-recht-aan programma's. Zij begonnen op de regel met het laagste regelnummer en eindigden op de regel met hoogste regelnummer. Één van de dingen waar een computer echter zo goed in is, is het herhaald uitvoeren van steeds gelijke berekeningen. Een ander pluspunt is de capaciteit om beslissingen te nemen. Om deze punten in een programma te realiseren, is het noodzakelijk de volgorde van uitvoering van de programmaregels te wijzigen. In dit hoofdstuk behandelen we enkele van de statements die dat mogelijk maken. Voordat het echter zover is, leren we eerst nette afdrukken maken met een nieuw soort PRINT-statement.

2.2 PRINT...,

In hoofdstuk 1 schreven we een programma voor het berekenen van de verhoudingspercentages tussen twee getallen. Op het scherm zag de afdruk er als volgt uit:

```
RUN
? 57
? 74
129.825
```

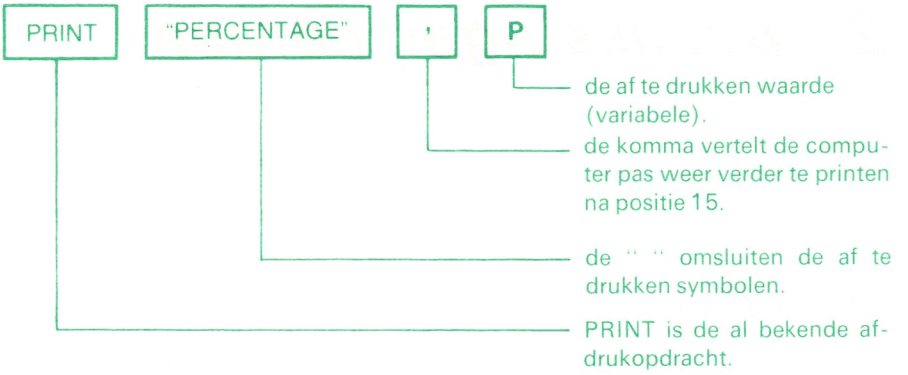
Het zou natuurlijk veel beter zijn als in het antwoord vermeld zou staan waar het om gaat; in dit geval PERCENTAGE. Daartoe vervangen we regel 50 (uit voorbeeld 3) door:

```
50  PRINT "PERCENTAGE", P
```

Het resultaat hiervan is

	programmaregel	**op scherm**	
oud:	50 PRINT P	129.825	
nieuw:	50 PRINT "PERCENTAGE", P	PERCENTAGE	129.825

Het PRINT-statement bestaat nu uit vier delen:



PRINT..., kan uitstekend worden gebruikt voor het verbeteren van de lay-out van programma-uitvoer. Zie bijvoorbeeld hoe dit systeem in het percentage-programma kan worden toegepast:

Oorspronkelijk programma:

```
10  REM ★★ PERCENTAGEBEREKENING ★
20  INPUT E
30  INPUT T
40  LET P = (T/E) ★100
50  PRINT P
60  END
```

RUN:

```
RUN
? 80
? 37
46.25
```

Verbeterd programma:

```
10  REM ★★ PERCENTAGEBEREKENING ★★
20  PRINT "UW EERSTE GETAL"
30  INPUT E
40  PRINT "UW TWEEDE GETAL"
50  INPUT T
60  LET P = (T/E) ★100
70  PRINT "PERCENTAGE", P
80  END
```

RUN:

```
RUN
UW EERSTE GETAL ———————— effect van regel 20
? 80
```

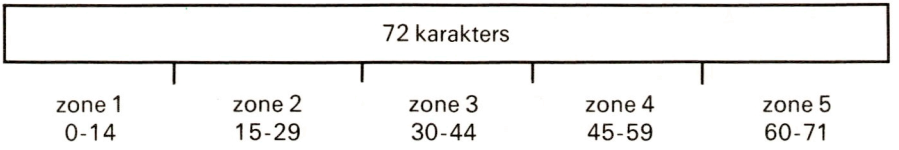
Programma 2-1. Verbeterd percentageprogramma.

Let op hoe het gebruik van teksten in regel 20 en 40 samen met het gebruik van PRINT..., niet alleen de uitvoer maar ook het programma duidelijker en begrijpelijker maakt.

[K] Programma 2-1

Komma's in PRINT-statements

U kunt meer dan één komma toepassen in een PRINT-statement. Basic kent 5 zones die corresponderen met 5 komma's in één PRINT-statement.*



Elke achtereenvolgende komma verplaatst het volgende printitem naar het begin van de volgende printzone.

```
PRINT "ZONE 1", "ZONE 2", "ZONE 3"
```

resulteert in:

```
ZONE 1          ZONE 2          ZONE 3
```

en:

```
PRINT "PERCENTAGE", P
```

geeft:

```
PERCENTAGE      46.25
```

terwijl

```
PRINT "PERCENTAGE "P**
```

resulteert in:

```
PERCENTAGE 46.25
```

Zie voor verklaring noten pag. 40

* De BBC-computer heeft 4 printzones van elk 10 karakters zodat voor deze computer het voorbeeld er iets anders uitziet.

** De BBC-computer laat deze manier niet toe. Om aan te geven dat er geen ruimte tussen beide items mag, gebruikt de BBC een ';'. (bijv. PRINT "PERCENTAGE";P)
Tevens dient u er rekening mee te houden dat de BBC-computer getallen tegen de rechterzijde van een zone print, terwijl woorden tegen de linkerzijde worden geplaatst. Wilt u de getallen ook tegen de linkerzijde geprint hebben dan dient u voor de variabele X een ';' aan te brengen.
Bijvoorbeeld:

```
10  LET X = 8
20  PRINT X, X, X
30  PRINT; X,; X,; X
40  END

RUN
         8        8        8        positie 9, 19, 29
8        8        8                 positie 1, 11, 21
```

Wij zullen u bij de programma's waar dit relevant is hierop attenderen. In vele gevallen is het echter niet belangrijk waar de uitkomst op het beeldscherm verschijnt.

ZOP 1

Wat zou er na de volgende BASIC-regels op het scherm staan:
(veronderstel A=48, B=8, C=6)

(a) PRINT "OPPERVLAK" A
(b) PRINT "LENGTE" B, "BREEDTE" C, "OPPERVLAK" A
(c) PRINT "LENGTE", "BREEDTE", "OPPERVLAK"

Geef aan met welk BASIC-statement de volgende regels op het scherm kunnen worden afgedrukt.

	zone 1	zone 2	zone 3	zone 4
(d)	LENGTE	8	BREEDTE	6
(e)	LENGTE	8		
	BREEDTE	6		
	OPPERVLAK	48		
(f)	LENGTE	BREEDTE		OPPERVLAK

2.3 Herhaalde RUN's met GOTO

Veronderstel nu dat u het programma wilt gebruiken om van een klas met schoolkinderen die een test hebben gemaakt het behaalde percentage van elk kind te berekenen. Daartoe zou u steeds het programma opnieuw moeten starten met RUN:

```
RUN
UW EERSTE GETAL
?80
UW TWEEDE GETAL
?42
PERCENTAGE        52.5
```

```
RUN
UW EERSTE GETAL
? 80
UW TWEEDE GETAL
? 19
PERCENTAGE        23.75
enz.
```

Afb. 2-1. Herhaald gebruik van het percentageprogramma.

Het zou veel eenvoudiger zijn als de computer nadat het eerste percentage is afgedrukt, direct opnieuw begon met de vraag om het eerste getal. Dit kan met het volgende statement.

GOTO regelnummer

Dit statement stuurt het programma naar ongeacht welk bestaand regelnummer, zowel in voorwaartse als terugwaartse richting. Hier volgt het percentageprogramma, waarin het GOTO-statement is verwerkt:

```
10  REM ★ SCORE NAAR PERCENTAGE ★
20  PRINT "MAXIMAAL AANTAL GOEDE VRAGEN"
30  INPUT T
40  PRINT "VOLGENDE SCORE"
50  INPUT M
60  LET P = (M/T) ★ 100
70  PRINT "PERCENTAGE", P
80  GOTO 40
```

Programma 2-2. Percentageprogramma voor herhaald gebruik.

Merk op dat hier het maximum aantal punten slechts éénmaal hoeft te worden ingetoetst. De waarde T wordt in de loop van het programma niet overschreven, noch aangepast. De berekening vindt plaats in regels 50-70 en in regel 80 wordt het programma opgedragen weer naar regel 40 te gaan. Merk op dat het programma nooit eindigt!! Daartoe moet u op de toets ESCAPE of BREAK drukken; deze toets beëindigt het programma ongeacht het stadium. Het aangepaste programma levert de volgende RUN:

```
RUN
MAXIMAAL AANTAL GOEDE VRAGEN
? 80
VOLGENDE SCORE
? 42
PERCENTAGE        52.5
VOLGENDE SCORE
? 67
PERCENTAGE        83.75
VOLGENDE SCORE
? 19
```

```
PERCENTAGE        23.75
VOLGENDE SCORE
? 55
PERCENTAGE        68.75
VOLGENDE SCORE
?
```

Het GOTO-statement onderbreekt het normale programmaverloop (in regelvolgorde). Zodra het programma GOTO tegenkomt, wordt de uitvoering onmiddellijk onderbroken en wordt verdergegaan met het aangegeven regelnummer (onvoorwaardelijke sprong, Eng. unconditional jump)

[K] Programma 2-2. Het programma kan worden beëindigd met "ESCAPE" of "BREAK".

ZOP 2

Het volgende programma kwadrateert een getal, dat wil zeggen het vermenigvuldigt het ingelezen getal met zichzelf. Voeg een regel toe zodanig dat het programma voor het kwadrateren van meer getallen kan worden gebruikt.

```
10  INPUT N
20  LET S = N★N
30  PRINT S
40  END
```

Programma 2-3 Kwadrateerprogramma.

[K] Tik het programma in en voeg de extra regel toe.

2.4 Programmeerstijl

Het gebruik van een GOTO-statement kan vaak nuttig zijn, maar het biedt geen volledige oplossing voor het herhalingsprobleem. Hier zitten we met het probleem van een niet-beëindigd programma. Steeds wordt in regel 80 het programma naar regel 40 teruggestuurd en wordt gevraagd om een nieuw getal '?'. Het programma zit vast in een oneindige kringloop, kortweg loop (zeg: loep) en kan daar niet uitkomen. De enige manier om te ontsnappen is de ESCAPE-toets. (De manier om uit een vastgelopen programma te ontsnappen varieert van computer tot computer, maar gaat meestal via een ESCAPE-toets (BBC) of BREAK-toets, vaak ook door CONTROL tegelijk met een letter in te drukken).

Het is geen mooie manier een programma met een noodstop te moeten afbreken. Daar komen we nog op terug. Veel belangrijker is het onoordeelkundig gebruik van GOTO. Zoals al eerder opgemerkt, kan dit statement uw programma op drastische wijze beïnvloeden. Doordat een sprong naar een willekeurige regel is toegestaan, wordt vaak de logische opbouw van een programma verstoord door 'gemakshalve' sprongen naar 'ergens' in het programma te maken. Beter is het de structuur van het probleem te volgen en op grond daarvan het programma op te bouwen. In dit boek zullen we daarom het gebruik van GOTO zo veel mogelijk vermijden, behalve

in die gevallen waar dit vanwege de duidelijkheid nuttig kan zijn. We hopen dat ook u het gebruik van het GOTO-statement zo veel mogelijk zult leren vermijden. Echter, in veel programmaleerboeken en computertijdschriften zult u het overvloedige gebruik van GOTO nog aantreffen.

2.5 IF ...THEN...

Het probleem bij het vorige programma had betrekking op het beëindigen daarvan. Als we met de hand een dergelijke berekening uitvoeren, zien we direct wanneer we aan het eind van de lijst gekomen zijn. De computer ziet dit echter niet. In de volgende paragrafen zullen we laten zien hoe de computer voor ons eenvoudig telwerk kan bijhouden als we deze de lengte van de lijst laten weten. Nu kijken we echter naar een eenvoudiger methode om het programma te laten weten wanneer het laatste lijstnummer is ingevoerd.

De toegepaste methode is gebaseerd op waarschijnlijkheid, of beter de onwaarschijnlijkheid dat een bepaald getal in de desbetreffende lijst zal voorkomen. Gaat het om de score antwoorden op een schooltest, dan ligt elke score tussen nul en het totaal aantal vragen. Kiezen we als 'score' een negatief getal dan is het logisch dat geen leerling dit kan hebben behaald. Een dergelijke waarde heet een dummywaarde (Eng. terminating value, rogue value). Wat we nu willen, is dat het programma normaal functioneert voor normale (= waarschijnlijke) scores en eindigt op de ingave van een negatief getal; b.v. –9. We willen een programma schrijven met de volgende logische structuur:

1 Start.
2 Voer het aantal vragen in.
3 Voer de volgende score in.
4 Als deze score gelijk is aan −9 ga dan naar regel 8
  ga anders naar regel 5.
5 Bereken het percentage.
6 Druk het percentage af.
7 Ga naar regel 3.
8 Stop.

Afb. 2-2. Logische structuur van een programma met een einde.

Gelukkig is er een BASIC-statement dat deze functie uit regel 4 voor ons vervult:

IF...THEN regelnummer [ELSE]

└── voorwaarde waarop naar regelnummer wordt gesprongen

Alles wat we moeten doen is statement 4 in BASIC-vorm te schrijven en in te voegen in het programma als:

```
40  IF M = -9 THEN 80
```

Dit statement betekent: Als de waarde in M gelijk is aan −9, ga dan naar regel 80, ga anders door met de volgende regel. Het gedeelte na ELSE is niet in het BASIC-statement opgenomen maar wordt impliciet verondersteld. Vele BASIC's kennen het ELSE-gedeelte wel expliciet en laten dan nog een kort extra statement toe (bijv. LET ...).

Het toevoegen van dit extra IF...THEN... kan erg gemakkelijk, omdat we erg royaal zijn geweest met het toedelen van regelnummers aan de diverse statements. Dit levert ons:

```
10  REM ★★ PERCENTAGES ★★
20  PRINT "HET MAXIMAAL AANTAL GOEDE VRAGEN"
25  INPUT T
30  PRINT "VOLGENDE SCORE"
35  INPUT M
40  IF M = -9 THEN 80
50  LET P = (M/T) ★ 100
60  PRINT "PERCENTAGE", P
70  GOTO 30
80  END
```

Programma 2-4 Percentage-berekening met een dummy-getal als eindwaarde.

Dit programma gebruikt u op dezelfde wijze als programma 2-2 totdat u de laatste score heeft ingevoerd. Nu tikt u de dummy-waarde −9 in en ziedaar, het programma eindigt.

```
RUN
HET MAXIMAAL AANTAL GOEDE VRAGEN
? 80
VOLGENDE SCORE
? 43
PERCENTAGE          53.75
VOLGENDE SCORE
? 29
PERCENTAGE          36.25
VOLGENDE SCORE
? 62
PERCENTAGE          77.5
VOLGENDE SCORE
? -9 ———————————— Dummy-waarde
```

[K] Programma 2-4. Gebruik nu een dummy-waarde die u handig vindt.

ZOP 3

Het programma uit ZOP 2:

```
10  INPUT N
20  LET S = N ★ N
30  PRINT S
```

```
35  GOTO 10
40  END
```

eindigde evenmin als het percentageprogramma. Verander het programma zó dat het op een geschikte dummy-waarde eindigt.

2.6 Ongelijkheden

Het is handig in een IF...THEN...-statement een expressie als M = −9 te kunnen gebruiken. Deze expressie maakt M niet gelijk aan −9, maar vergelijkt beide grootheden. Zijn deze gelijk dan treedt de THEN-conditie op; zijn zij ongelijk dan wordt de hele regel met IF... genegeerd.
Het IF-statement kijkt naar de voorwaarde die op IF volgt: is deze waar dan treedt de THEN-conditie op; is deze niet waar dan wordt de regel genegeerd. BASIC kent nog meer vergelijkingstekens.

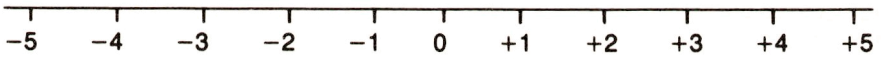
Vergelijkingsteken	Voorbeeld	Betekenis
=	A = −9	A is gelijk aan −9
>	A > −9	A is groter dan −9
<	A < −9	A is kleiner dan −9

Er zijn ook combinaties mogelijk:
>= betekent groter dan of gelijk aan
<= betekent kleiner dan of gelijk aan
<> betekent ongelijk aan (kleiner of groter dan)

Waar of niet waar?
Beschouw de expressie A < B. Als A = 2 en B = 5, dan is de expressie waar, want A is kleiner dan B.
Indien echter A = 2 en B = 1, dan is de expressie niet waar, want A is niet kleiner dan B.
En als A en B gelijk zijn, bijv. A = 2 en B = 2? De expressie is dan niet waar, want A is nog steeds niet kleiner dan B.
Aan een vergelijking van een dergelijke aard kan een waarde worden toegekend; deze waarde is óf waar óf niet waar.
Tussenliggende mogelijkheden zijn er niet. Dit noemt men de logische waarde van een vergelijking.

Vergelijking	Logische waarde
3 > 2	waar (Eng. true)
7 < 7	niet waar (Eng. false)

Vanaf dit punt gebruiken we voor waar de Engelse vertaling 'true' en voor niet waar het begrip 'false'.
Het gebruik van true en false in vergelijkingen is eenvoudig in het gebruik bij natuurlijke positieve getallen. Het is voor breuken echter niet altijd even duidelijk wat er gebeurt. Dit heeft te maken met de manier waarop de computer breuken in het geheugen opslaat. Hierop komen we later nog terug. Denkt u in geval van twijfel aan de getallenrechte:



Afb. 2-3. De getallenrechte.

Als een getal op deze rechte links ligt van een ander, dan is het kleiner; ligt het aan de rechterzijde, dan is het groter dan het andere getal.

Voorbeeld 1
Geef aan of de volgende vergelijkingen true of false zijn:

Waarde A	Waarde B	Expressie
2	5	A>B
2	−5	A>B
−2	−5	A>B
−2	−1	A>B
−5	2	A<B
5	−2	A<B
−3	3	A=B

Afb. 2-4.

Oplossing:
We berekenen de waarde van elke expressie door de waarden van de variabelen in te vullen. Vervolgens bepalen we de logische waarde van de vergelijking.

Waarde		Uitwerking		
A	B	Expressie	Ingevuld	Logische waarde
2	5	A>B	2> 5	F
2	−5	A>B	2>−5	T
−2	−5	A>B	−2>−5	T
−2	−1	A>B	−2>−1	F
−5	2	A<B	−5< 2	T
5	−2	A<B	5<−2	F
−3	3	A=B	−3= 3	F

F = false T = true

Afb. 2-5.

ZOP 4
Vul de volgende tabel in om te bepalen of een gegeven expressie true of false is.

Waarde		Uitwerking		
A	B	Expressie	Ingevuld	Logische waarde
3	7	A>B		
5	3	A>B		
−3	5	A>B		
8	5	A<B		
3	9	A<B		
8	−2	A<B		

Afb. 2-6.

Op dit moment zijn we in staat om het verloop van een programma te beïnvloeden, afhankelijk van de logische waarde van een bepaalde vergelijking; met andere woorden als aan een bepaalde vergelijking wordt voldaan.

Voorbeeld 2
Wordt bij uitvoering van het onderstaande programma na regel 30 naar regel 100 of naar regel 40 gesprongen?

```
10  LET A = -3
20  LET B = 2
30  IF A + B > 0 THEN 100
40  ....
```

Programma 2-5

Oplossing:
−3+2 is −1; en dat is niet groter dan 0. De vergelijking is dus false en het THEN-gedeelte wordt niet uitgevoerd. Het programma wordt vervolgd op de dichtstbijzijnde hogere regel (40).

ZOP 5
Geef in elk van de onderstaande voorbeelden aan of met regel 40 dan wel met regel 100 wordt verder gegaan nadat regel 30 is uitgevoerd?

(a)
```
10  LET A = 7
20  LET B = -8
30  IF A - B < 0 THEN 100
40  ....
```

(b)
```
10  LET X = 3
20  LET Y = -3
30  IF X/Y= -1 THEN 100
40  ....
```

(c)
```
10  LET P = -1
20  LET Q = 3
30  IF P + Q > Q THEN 100
40  ....
```

(d)
```
10  LET M = 3
15  LET N = -4
20  LET P = -2
30  IF M - N < N - P THEN 100
40  ....
```

(e)
```
10  LET R = 1
20  LET S = -2
30  IF R + S > -1 THEN 100
40  ....
```

Programma's 2-6 t/m 2-10.

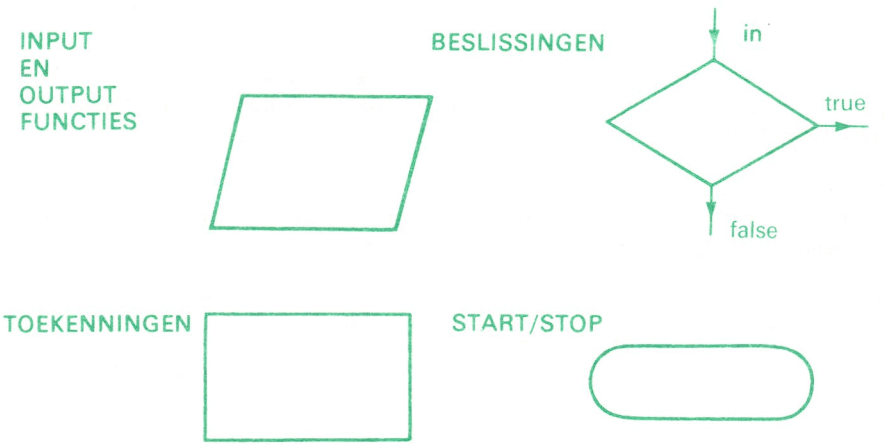
2.7 Stroomdiagrammen

Zoals we al eerder hebben opgemerkt, is de belangrijkste taak van de programmeur het vinden van een methode waarop, uitgaande van de beschikbare gegevens, het resultaat in geschikte vorm wordt verkregen. Daarmee komen we aan wat misschien het 'lelijkste' woord is uit de computerwereld: algorithme. Dit woord betekent de algemene beschrijving van een bepaalde oplossings-strategie. Meestal is een algorithme gedefinieerd als een reeks stappen of deelprocedures die gezamenlijk tot de gewenste oplossing leiden. Deze stappen zijn ieder voor zich zó eenvoudig dat zij geen nadere analyse behoeven (althans, in het desbetreffende stadium). Merk op dat in deze definitie geen enkele maal het woord computer is gevallen.
De definitie van een computerprogramma is vrijwel gelijk: een programma is een algorithme, geschreven voor een computer.

Een algorithme kan op drie manieren worden gerepresenteerd:

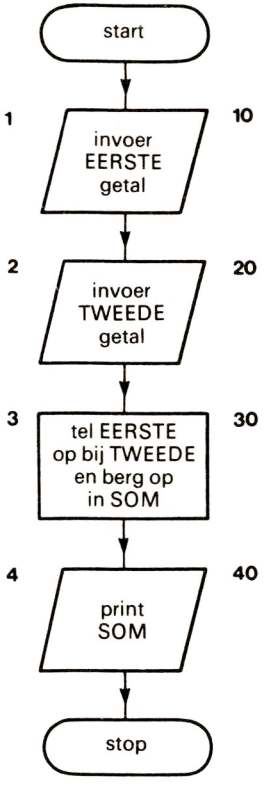
1. als een beschrijving;
2. in BASIC gecodeerd;
3. in een stroomdiagram (Eng. flowchart).

Stroomdiagrammen lijken wat op blauwdrukken en blokschema's en spreken tot de verbeelding van mensen die de zaken graag visueel in zich opnemen. De verschillende algorithme-stappen kunnen door een aantal standaardsymbolen worden weergegeven.



Afb. 2-7.

Een pijl geeft de leesvolgorde van het diagram aan.
De symbolen worden voorzien van passende inschriften. Het eerste programma uit dit boek heeft in stroomdiagram de volgende vorm:



Afb. 2-8. Stroomdiagram van programma 1-1.

De inschriften in de symbolen zijn wat cryptisch, maar kunnen door iedereen worden gevolgd, ongeacht de programmeerervaring. In dit kader noemen we de inschriften taal-onafhankelijk. De getallen aan de linkerzijde refereren aan de algorithmebeschrijving. De getallen ter rechterzijde met de regelnummers van programma 1-1.

ZOP 6
Teken een stroomdiagram voor het percentageprogramma (programma 2-1).

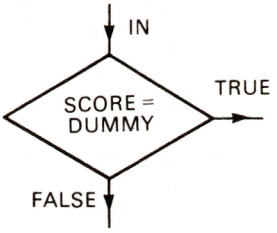
Het beslissingssymbool
U kent de methode voor het nemen van beslissingen in BASIC door middel van het IF...THEN...-statement:

IF <vergelijking> is true THEN ga naar regel X

Het principe van een zijsprong naar X of het negeren van de regel is afgebeeld in afb. 2-9a. De beslissing:

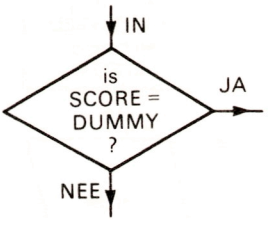
```
40 IF M = -9 THEN 80
```

zou worden afgebeeld als:



Afb. 2-9a.

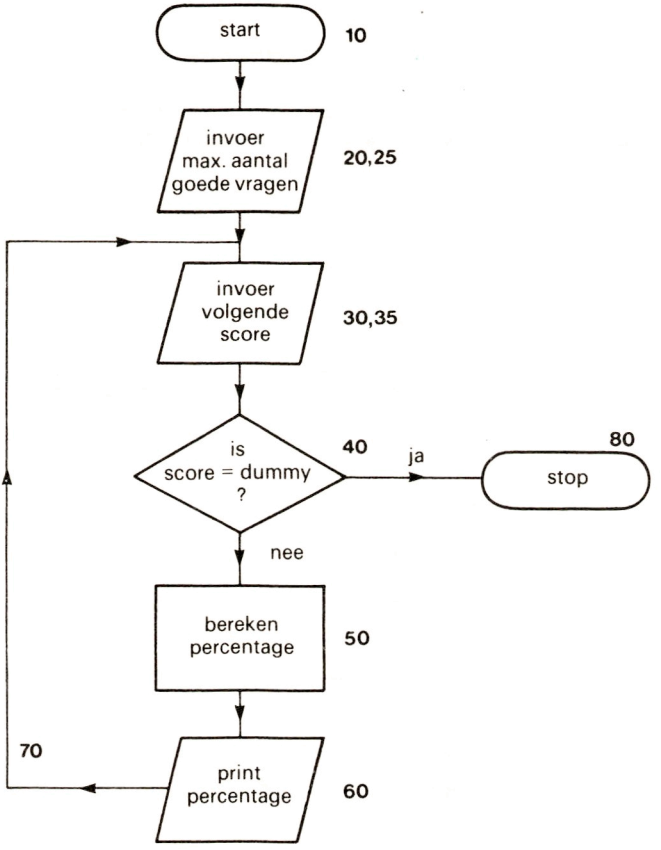
De vergelijking score = dummy kan worden uitgebeeld met een vraagteken:



Afb. 2-9b.

De stijl en opmaak van het diagram is aan u. Een goede test op de kwaliteit ervan is of u enige tijd later nog begrijpt waar het diagram over gaat. Of u laat het aan een met uw probleem onbekende lezen die dan min of meer de essentie moet kunnen volgen.

Bijgaand is afgebeeld een stroomdiagram van het scoreprogramma (programma 2-4):



Afb. 2-10. Stroomdiagram voor het scoreprogramma.

De nummers aan de rechterzijde refereren weer aan de programmaregelnummers. Het statement 70 GOTO 30 is hier met een doorgetrokken lijn naar blok 30 aangegeven.

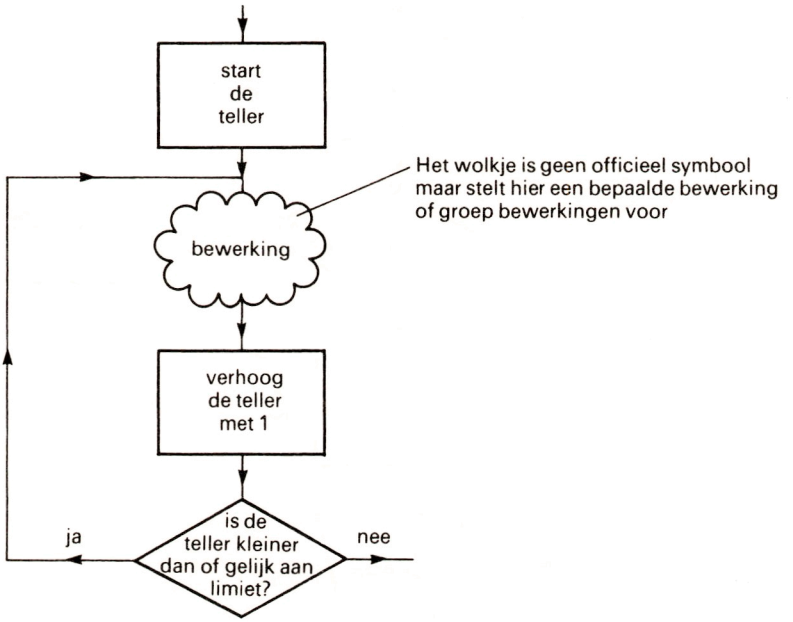
ZOP 7

Schrijf een stroomdiagram voor het programma dat u schreef als antwoord op ZOP 3.

Het belang van een goed stroomdiagram kan eigenlijk niet vaak genoeg worden benadrukt; na het tekenen van een goed stroomdiagram is het programmeren vaak niet zo moeilijk meer. We zullen het stroomdiagram nog vaak te hulp roepen bij het schrijven van de programma's in dit boek.

2.8 Tellen

Computers zijn goed in zich herhalende bewerkingen. Als we deze bewerkingen werkelijk willen beïnvloeden, in plaats van alleen maar stoppen en starten zoals in het vorige voorbeeld, dan moeten we de computer leren voor ons het aantal bewerkingen te tellen. Daartoe laten we de computer beginnen met tellen bij de eerste maal dat we een groep instructies (de bewerking) uitvoeren en vervolgens tellen we één verder voor iedere keer dat we die bewerking (in een lus) weer tegenkomen. Bij het bereiken van een bepaalde limiet wordt de lus doorbroken en eindigt het programma. In een stroomdiagram:



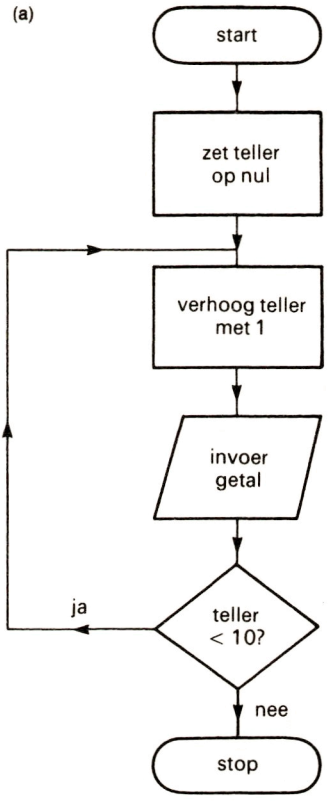
Afb. 2-11. Een teller in een stroomdiagram.

Merk op dat de 'teller' uit drie stroomdiagramsymbolen is opgebouwd:

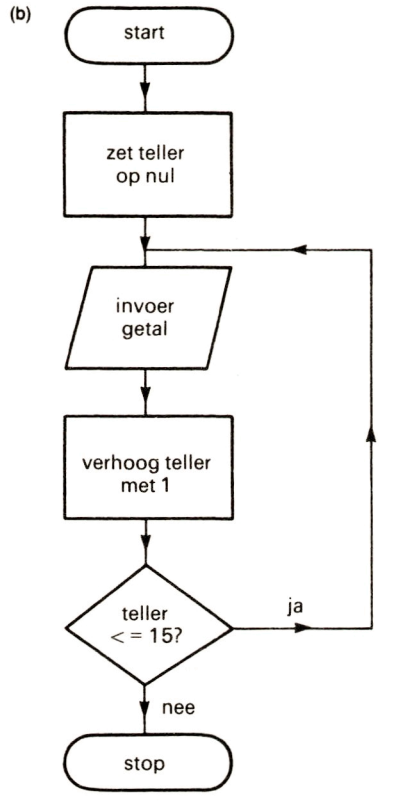
1. een procedure voor het initialiseren van de teller;
2. een procedure voor het bijtellen van één, iedere keer dat de bewerking doorlopen is;
3. een procedure voor het beëindigen van het programma als een bepaalde limiet bereikt is.

We moeten goed opletten de procedure na het juiste aantal keren te verlaten. Als een waarschuwing mag gelden dat, hoewel het bovenstaande voorbeeld 10x doorlopen is, de teller na afloop op 11 staat. Dit is iets om zeer goed op te letten, vooral als met het getal in de teller later nog iets gedaan moet worden.

ZOP 8
Hoeveel getallen kunnen met onderstaande stroomdiagrammen worden ingelezen?



Afb. 2-12a.



Afb. 2-12b.

ZOP 9
Maak het volgende programma af, zodanig dat vijf getallen worden ingelezen.

```
10  LET C = 0
20  INPUT N
30  IF C = ... THEN 60
40  LET C = C + 1
50  GOTO 20
60  END
```

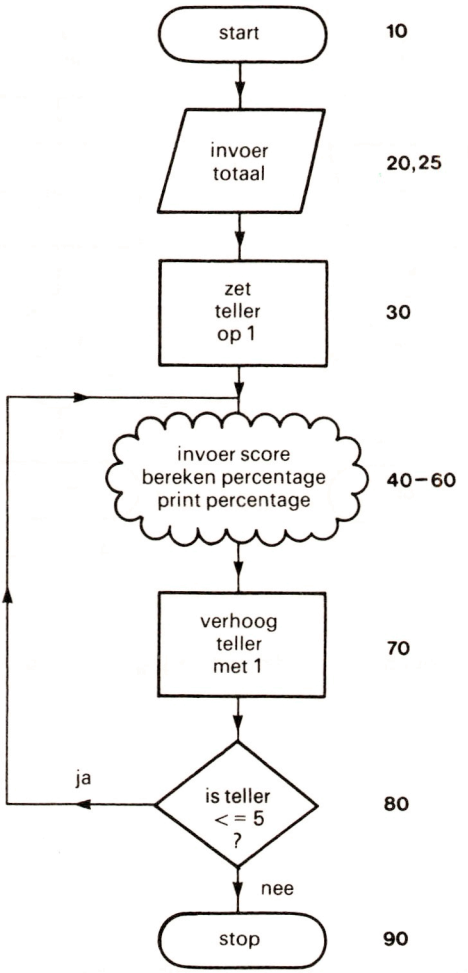
Programma 2-11.

Voorbeeld 3
Schrijf een BASIC-programma voor het berekenen van de scorepercentages voor vijf leerlingen.

Oplossing
We veronderstellen dat de bewerking hier inhoudt:

```
Voer de score van een leerling in.
Bereken het percentage.
Print het percentage.
```

Het algorithme van een dergelijk programma heeft omgezet in een stroomdiagram de volgende vorm:



Afb. 2-13. Stroomdiagram voor de percentageberekening van vijf scores.

Uit het diagram volgt snel de structuur van het programma, hetwelk eenvoudig uit programma 2-4 herschreven kan worden:

```
10  REM ★★ PERCENTAGES ★★
20  PRINT "HET MAXIMAAL AANTAL GOEDE VRAGEN"
25  INPUT T
30  LET C = 1 ____________________________ Start teller met 1
40  PRINT "VOLGENDE SCORE"
45  INPUT M
50  LET P = (M/T) ★ 100
60  PRINT "PERCENTAGE", P
70  LET C = C + 1 ________________________ Verhoog teller
75  PRINT "REGEL 75", "TELLER IS HIER:", C
80  IF C < = 5 THEN 40 ___________________ Telling beëindigd?
90  END
```

Programma 2-12 Uitbreiding van het percentageprogramma met een teller.

```
RUN
HET MAXIMAAL AANTAL GOEDE VRAGEN
? 75
VOLGENDE SCORE
? 57
PERCENTAGE      76
REGEL 75        TELLER IS HIER: 2
VOLGENDE SCORE
? 63
PERCENTAGE      84
REGEL 75        TELLER IS HIER: 3
VOLGENDE SCORE
? 42
PERCENTAGE      56
REGEL 75        TELLER IS HIER: 4
VOLGENDE SCORE
? 39
PERCENTAGE      52
REGEL 75        TELLER IS HIER: 5
VOLGENDE SCORE
? 69
PERCENTAGE      92
REGEL 75        TELLER IS HIER: 6      Bij het verlaten van de loop is de
                                       TELLER = 6
```

[K] Programma 2-12.

Oefening 1

In vraag 2 van opgave 1, schreef u een programma ter berekening van de kosten van dubbel glas. Indien u dit programma voor meer dan één raam wilt gebruiken,

moet u het programma steeds opnieuw starten. Pas het programma zó aan dat u meer dan één raam in één RUN kunt berekenen. De 'bewerking' in dit programma is:

```
voer hoogte en breedte in
bereken de kosten voor dit raam
print de kosten voor dit raam
```

a) Teken een stroomdiagram voor de berekening van zes ramen.
b) Codeer het algoritme in BASIC. [K] en test het op uw personal computer.
c) Breid het programma zó uit dat het voor een willekeurig aan het begin van het programma gekozen aantal ramen geschikt is.

Oefening 2
Breid het programma uit opgave 1b uit. De 'bewerking' in deze opgave was:

```
bereken de rente
print het jaar met de rente
bereken het tegoed voor het volgende jaar
```

(a) Teken een stroomdiagram om de achtereenvolgende rentes over zes jaar te berekenen. Controleer het antwoord.
(b) Schrijf aan de hand van het stroomdiagram een BASIC-programma. [K] Controleer het programma.
(c) Breid het programma nu zó uit dat het voor een willekeurig aan het begin gekozen aantal jaren de rentes berekent. [K] Controleer uw resultaten op uw microcomputer.

2.9 Vergelijkingen

We hebben al gezien dat het met BASIC mogelijk is op eenvoudige wijze getallen met elkaar te vergelijken. Vaak ontstaat de behoefte vast te stellen of een bepaalde waarde groter of kleiner is dan een andere. Een dergelijke beslissing is de basis voor sorteer-algorithmen. Er zullen nog verschillende soorten sorteerprogramma's aan de orde komen, dus laten we met het eenvoudigste geval beginnen.

Voorbeeld 4
Schrijf een eenvoudig algorithme dat van twee ingevoerde getallen het grootste afdrukt. Tot nu toe hebben we steeds de stroomdiagrammen gebruikt, deze keer gebruiken we de beschrijvende methode uit hoofdstuk 1.

Oplossing

1. Start.
2. Voer het eerste getal in.
3. Voer het tweede getal in.
4. Als eerste getal > tweede getal ga dan naar 7, ga anders naar 5.
5. Print tweede getal.
6. Ga naar 8.
7. Print eerste getal.
8. Stop.

Dit is niet de mooiste oplossing, maar een 'eerste poging'. Later zullen we mooiere oplossingen tegenkomen bij andere sorteermethoden.

Opgave 2

1. Teken een stroomdiagram en schrijf een programma voor het invoeren van twee getallen en het afdrukken van het kleinste van de twee. Verander het programma zó, dat het (a) 5 paar getallen en (b) elk gewenst aantal getallenparen kan vergelijken.
2. Verander het score-percentage-programma zodanig dat het:

(a) kan werken met elke willekeurige klassegrootte.
(b) het gemiddelde percentage kan berekenen.
(c) er het beste resultaat uit kan halen en afdrukken.

Maak eerst de stroomdiagrammen!!

Samenvatting van hoofdstuk 2

Aan het einde van dit hoofdstuk moet u de volgende zaken beheersen:

- Het combineren van tekst en variabelen in PRINT-statements
- Het gebruik van ',' in PRINT-statements.
- GOTO voor het herhalen en beïnvloeden van het programma.
- Het toepassen van een DUMMY-waarde om een programma te beëindigen.
- Het gebruik van IF ...THEN...
- Het bepalen van de logische waarden true en false in vergelijkingen met =, > en <.
- Tekenen van stroomdiagrammen.
- Het gebruik van tellers in het stroomdiagram en programma om de loop daarvan te beïnvloeden.

Antwoorden op ZOP's en oefeningen

ZOP 1

(a) OPPERVLAK 48
(b) LENGTE 8 BREEDTE 6 OPPERVLAK 48

(c) LENGTE BREEDTE OPPERVLAK
(d) PRINT "LENGTE", B, "BREEDTE", C
(e) PRINT "LENGTE", B
PRINT "BREEDTE", C
PRINT "OPPERVLAK", A
(f) PRINT "LENGTE", "BREEDTE", " ", "OPPERVLAK"

dit is een spatie.

ZOP 2

Voeg aan het programma toe:

```
35 GOTO 10
```

ZOP 3

```
10 INPUT N
15 IF N = -9 THEN 40
20 LET S = N★N
30 PRINT S
35 GOTO 10
40 END
```

Programma 2-13.

(Het gebruik van –9 als dummy-waarde is in dit voorbeeld minder goed gekozen als in het score-programma; een score van –9 is niet waarschijnlijk, maar het is mogelijk dat u –9 zou willen kwadrateren!).

ZOP 4

Waarde		Uitwerking		
A	B	Expressie	Ingevuld	Logische waarde
3	7	A>B	3>7	F
5	3	A>B	5>3	T
–3	5	A>B	–3>5	F
8	5	A<B	8<5	F
3	9	A<B	3<9	T
8	–2	A<B	8<–2	F

Afb. 2-14.

Als u een fout heeft gemaakt, kijk dan nog eens naar de getallenrechte:

A links van B betekent A < B is TRUE A B

A rechts van B betekent A > B is TRUE B A

ZOP 5

(a) 40 (b) 100 (c) 40
(d) 40 (e) 40

Uw computer had u daar als volgt bij kunnen helpen: voeg in

```
40  PRINT "40"
100 PRINT "100"
```

Afhankelijk van het resultaat wordt een van beide getallen voor u afgedrukt. Onderstaand is deze methode voor de vragen (a) en (b) aangegeven:

Programma ter oplossing van (a):

```
10  LET A = 7
20  LET B = -8
30  IF A-B < 0 THEN 100
40  PRINT "40"
50  GOTO 999
100 PRINT "100"
999 END
```

Programma 2-14.

Programma ter oplossing van (b):

```
10  LET X = 3
20  LET Y = -3
30  IF X/Y = -1 THEN 100
40  PRINT "40"
50  GOTO 999
100 PRINT "100"
999 END
```

Programma 2-15.

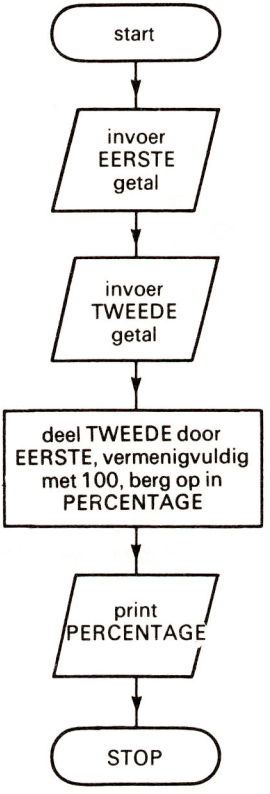
Programma-run 2-14

```
RUN
40
```

Programma-run 2-15

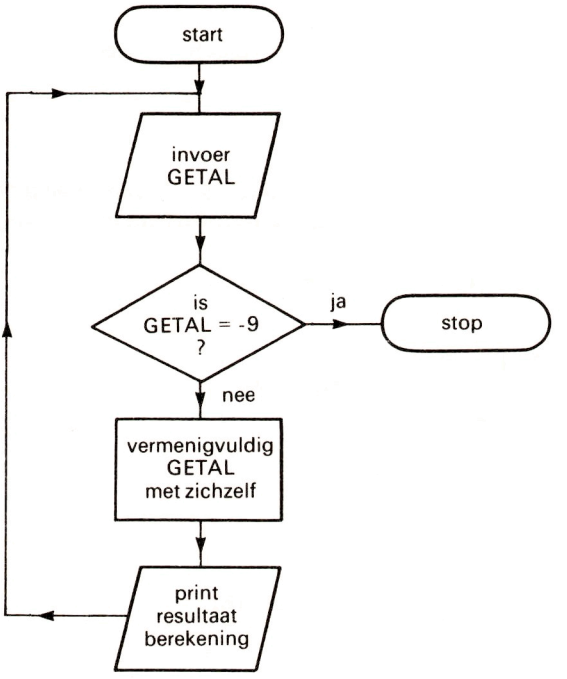
```
RUN
100
```

ZOP 6



Afb. 2-15.

ZOP 7



Afb. 2-16.

ZOP 8

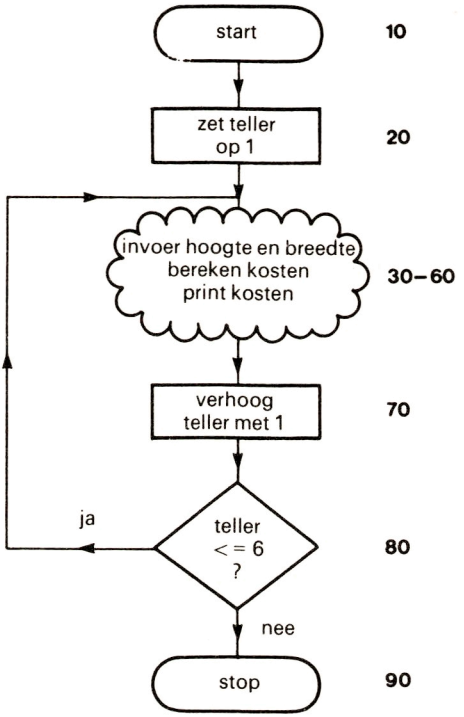
(a) 10 (b) 16

ZOP 9

```
30 IF C = 4 THEN 60
```

Oefening 1

1(a)



Afb. 2-17.

1(b)

```
10  REM ★★ KOSTEN VAN DUBBEL GLAS ★★
20  LET C = 1
30  PRINT "HOOGTE RAAM IN METERS"
35  INPUT H
40  PRINT "BREEDTE RAAM IN METERS"
45  INPUT B
50  LET K = 56 ★ (H + B) + 160 ★ (H ★ B) + 200
60  PRINT "RAAM", C, "KOSTEN", K
70  LET C = C + 1
80  IF C < = 6 THEN 30
90  END
```

Regels 20, 70 en 80 vormen de teller

Programma 2-16.

```
RUN
HOOGTE RAAM IN METERS
? 1.5
BREEDTE RAAM IN METERS
```

```
? 2
RAAM          1          KOSTEN     876
HOOGTE RAAM IN METERS
? 1.5
BREEDTE RAAM IN METERS
? 3
RAAM          2          KOSTEN     1172
HOOGTE RAAM IN METERS
? 1.5
BREEDTE RAAM IN METERS
? 4
RAAM          3          KOSTEN     1468
HOOGTE RAAM IN METERS
? 2.5
BREEDTE RAAM IN METERS
? 2
RAAM          4          KOSTEN     1252
HOOGTE RAAM IN METERS
?
etc.
```

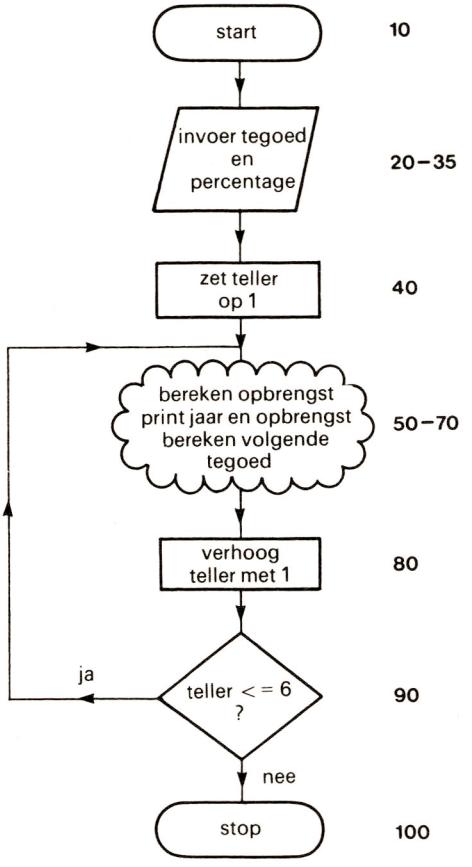
1(c)

```
10  REM ★★ KOSTEN VAN DUBBEL GLAS ★★
12  PRINT "AANTAL RAMEN"
14  INPUT N
20  LET C = 1
30  PRINT "HOOGTE RAAM IN METERS"
35  INPUT H
40  PRINT "BREEDTE RAAM IN METERS"
45  INPUT B
50  LET K = 56★(H + B) + 160★(H★B) + 200
60  PRINT "RAAM", C, "KOSTEN", K
70  LET C = C + 1
80  IF C < = N THEN 30
90  END
```

Programma 2-17.

Oefening 2

2(a)



Afb. 2-18.

2(b)

```
10  REM ★★ SAMENGESTELDE RENTEBEREKENING ★★
20  PRINT "UW REKENING TEGOED"
25  INPUT D
30  PRINT "RENTESTAND"
35  INPUT P
40  LET C = 1
50  LET Y = (P★D)/100
60  PRINT "JAAR"; C, "OPBRENGST", Y
70  LET D = D + Y
80  LET C = C + 1
90  IF C < = 6 THEN 50
100 END
```

Programma 2-18.

```
RUN
UW REKENING TEGOED
? 500
RENTESTAND
? 12.5
JAAR 1          OPBRENGST       62.5
JAAR 2          OPBRENGST       70.3125
JAAR 3          OPBRENGST       79.1015625
JAAR 4          OPBRENGST       88.9892578
JAAR 5          OPBRENGST       100.112915
JAAR 6          OPBRENGST       112.627029
```

2(c)

```
10  REM ★★ SAMENGESTELDE RENTEBEREKENING ★★
15  PRINT "BEREKENING OVER HOEVEEL JAREN"
16  INPUT N
20  PRINT "UW REKENING TEGOED"
25  INPUT D
30  PRINT "RENTESTAND"
35  INPUT P
40  LET C = 1
50  LET Y = (P ★ D)/100
60  PRINT "JAAR"; C, "OPBRENGST", Y
70  LET D = D + Y
80  LET C = C + 1
90  IF C < = N THEN 50
100 END
```

Programma 2-19.

```
RUN
BEREKENING OVER HOEVEEL JAREN
? 4
UW REKENING TEGOED
? 1000
RENTESTAND
? 13.75
JAAR 1          OPBRENGST       137.5
JAAR 2          OPBRENGST       156.40625
JAAR 3          OPBRENGST       177.912109
JAAR 4          OPBRENGST       202.375024
```

3 Strings

3.1 Wat is een string?

De eerste twee hoofdstukken uit dit boek gingen over het rekenen met getallen. De leek denkt dan ook dat dit de belangrijkste toepassing van een computer is. Dat is zeker niet waar. Vooral in commerciële programma's wordt het grootste deel van de tijd besteed aan andere mogelijkheden van de computer. In dit hoofdstuk behandelen we het gebruik van symbolen in BASIC in de vorm van strings.

Als we symbool schrijven, bedoelen we het alfabet in hoofdletters, de cijfers of digits van 0 ... 9, leestekens en nog enkele speciale symbolen.

A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,[,/,],↑,

←,0,1,2,3,4,5,6,7,8,9,:,;, <,=,>,?,spatie,!,komma,#,$,%,punt,(,),+,★,-,/,''.

We hebben geleerd programma's te schrijven met getallen (3,57, −92) en variabelen (A1, X, SOM enz.). Met BASIC is het ook mogelijk te manipuleren met groepen symbolen en deze in de computer in te lezen.
Dergelijke symbolengroepen worden strings genoemd. Enkele voorbeelden:

POES	(een woord)
JOOP DEN UYL	(een naam)
FXX45-★	(een willekeurige symbolenreeks)
PV-DA-99	(een autokenteken)
4470398	(een gironummer)

Een string mag uit elke willekeurige reeks symbolen opgebouwd zijn, ook een spatie is een belangrijk symbool. Wat BASIC betreft, wordt een getal als getal behandeld als het gaat om numerieke bewerkingen, in elk ander geval is een getal ook een reeks symbolen, en dus een string.
Als we een autokenteken of telefoonnummer tegenkomen, zien wij dat ook als een serie symbolen en niet als een getal waarmee een berekening moet worden uitgevoerd.

Geheugenlocaties voor strings
Hoe kunnen we nu aan de computer kenbaar maken dat de symbolengroep die we aan het intikken zijn als string en niet als getal moet worden geïnterpreteerd? Dit onderscheid is in BASIC aangebracht door aan de variabelenaam die voor strings wordt gebruikt een dollarteken ($) toe te voegen. Daarmee beschikt een klein BASIC-systeem over 286 variabelenamen voor numerieke doeleinden:

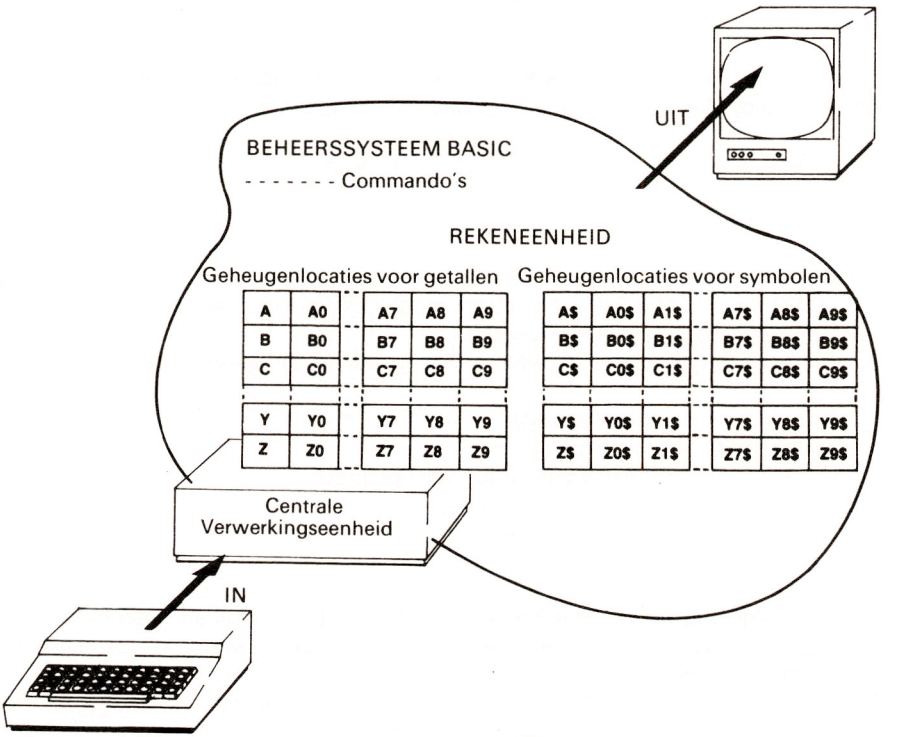
A, A0, A1..., A9
B, B0 enz.

en ook over 286 variabelenamen voor strings :
A$, A0$, A1$,...A9$
B$, B0$ enz.

We spreken in de praktijk over:

A$	"A-dollar"	of	"A-string"
B9$	"B-negen-dollar"	of	"B-negen-string"

U kunt een microcomputer nu dus opgebouwd denken uit o.a. twee geheugengebieden: één voor getallen en één voor strings. In figuur 3-1 is een en ander afgebeeld:



Afb. 3-1. Een eenvoudig microcomputersysteem.

Strings met " "
Het is gebruikelijk dat we de computer mededelen of een reeks karakters als string is bedoeld. Daartoe plaatsen we dubbele aanhalingstekens om de string:

```
10  LET Q$ = "HALLO"
en
30  IF Q$ = "HALLO" THEN 80
enz.
```

ZOP 1
Geef aan welke variabelenamen geldig zijn voor strings:

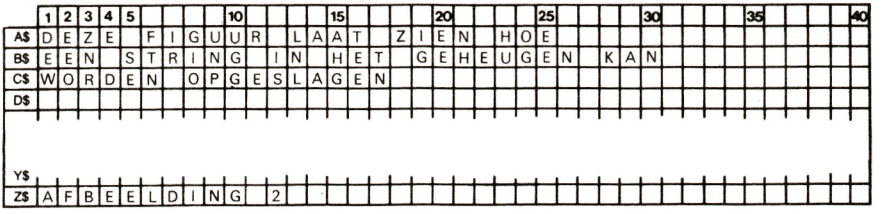
(a)A$ (b)M8 (c)T7$ (d)B9 (e)C$3 (f)8P$
(g)2$ (h)5$L

ZOP 2
Geef aan welke BASIC-statements correct zijn:

(a) LET A = 87
(b) LET B$ = "FRED"
(c) LET M$ = 9583
(d) LET K8 = "EEN FIETS"
(e) LET L17 = 38

3.2 Meer over strings

Hoe lang kan een string zijn? Hoeveel karakters mag u in een string plaatsen als een enkele groep? Dit is weer afhankelijk van de computer die u in uw bezit heeft. Voorlopig gaan we ervan uit dat een string maximaal 40 karakters (symbolen) lang mag zijn (incl. spaties), zowel voor in- als uitvoer van strings.
Dit is een logische beperking, omdat de regellengte op de gemiddelde computer toch niet meer bedraagt dan 40 tekens. Een geheugenlocatie voor strings moet u vanaf nu dus zien als een locatie met 40 posities, die elk een symbool kunnen bevatten.



Afb. 3-2. String-geheugenlocaties.

In dit hoofdstuk deden we alsof een string iets nieuws was. Dat is echter maar gedeeltelijk waar; in het eerste hoofdstuk hebben we al tekst laten afdrukken met behulp van het PRINT-statement. We namen impliciet aan dat de tekst tussen aanhalingstekens (Eng. stringquotes) letterlijk afgedrukt werd. In hoofdstuk 2 gebruikten we de komma om twee strings in een PRINT-statement tijdens het afdrukken van elkaar te scheiden.

3.3 PRINT ...;...

Uit wat we tot nu toe hebben gedaan, heeft u kunnen afleiden dat de lay-out van de informatie op het scherm erg belangrijk is. Dat geldt evenzeer voor getallen als voor strings. Als we teksten gaan bewerken, bijvoorbeeld reeksen karakters in de vorm van woorden of artikelnummers, willen we dat deze achter elkaar op het scherm worden afgedrukt en niet uitgespreid worden over verschillende printzones.

Het PRINT...;... statement kan dit voor ons verzorgen. PRINT H$;T$ draagt er zorg voor dat eerst alle karakters uit H$ van links af aan op de regel afgedrukt worden; onmiddellijk en aansluitend gevolgd door de karakters uit T$.

In de volgende paar pagina's stellen we ons een informatie centrum voor waarmee we kunnen communiceren in BASIC. Aan de hand hiervan zullen we het gebruik van strings in invoer en uitvoer behandelen.
We beginnen met een eenvoudige tele-antwoordservice.

CLEAR

Voordat we strings op wat grotere schaal in de computer gaan toepassen, moeten we wat geheugenruimte voor strings reserveren. Daarvoor wordt een algemeen stukje geheugen gebruikt dat wordt gevuld met allemaal spaties. CLEAR 100 in het volgende programma maakt honderd karakterlocaties voor ons leeg.*
We moeten de instructie in het begin van het programma plaatsen omdat anders onze met veel moeite ingetikte tekst uitgewist wordt.

```
10  REM ★★ TELE-ANTWOORD ★★
20  CLEAR 100
30  PRINT "HALLO"
40  PRINT "UW TELEFOONNUMMER ALSTUBLIEFT"
50  INPUT T$
60  PRINT                                    dit geeft een blanco regel
70  PRINT "HALLO", T$
80  PRINT
90  PRINT "HALLO"; T$
100 PRINT
110 PRINT "HALLO "; T$
120 PRINT
130 PRINT "HALLO"; " "; T$
140 END
```

Programma 3-1 Strings afdrukken.

Om u te helpen dit programma te begrijpen, hebben we naast de run de regelnummers van het programma vermeld waarop de desbetreffende uitvoer betrekking heeft. Dit wordt aangegeven met 'trace' (= spoor)

* Op de BBC-computer hoeft u CLEAR niet te gebruiken. Als u de programma's op een BBC-computer wilt draaien, laat u dan alle regels met CLEAR weg.

```
RUN                                                    Trace
HALLO                                                  ...30
UW TELEFOONNUMMER ALSTUBLIEFT                          ...40
? 58632                                                ...50
                                                       ...60
HALLO          58632                                   ...70
                                                       ...80
HALLO58632                                             ...90
                                                       ..100
HALLO 58632                                            ..110
                                                       ..120
HALLO 58632                                            ..130
```

Toelichting

Trace 50 INPUT T$ gaf een ? Ons antwoord was 58632 hetgeen door de computer als string wordt behandeld. (Als we INPUT T hadden geschreven was 58632 als getal behandeld.)

Trace 70 PRINT...,... op regel 70 drukt het eerste cijfer van het telefoonnummer af op de 15e positie van de regel, terwijl

Trace 90 PRINT...;... dit cijfer onmiddellijk achter de O van HALLO plaatst.

Geen van beide is echt bevredigend, maar:

Trace 110 Deze regel drukt evenals Trace 130 een extra

Trace 130 spatie af tussen beide strings, ofwel door HALLO te voorzien van een extra spatie ofwel door een " " spatie als aparte string af te drukken (is spatie).

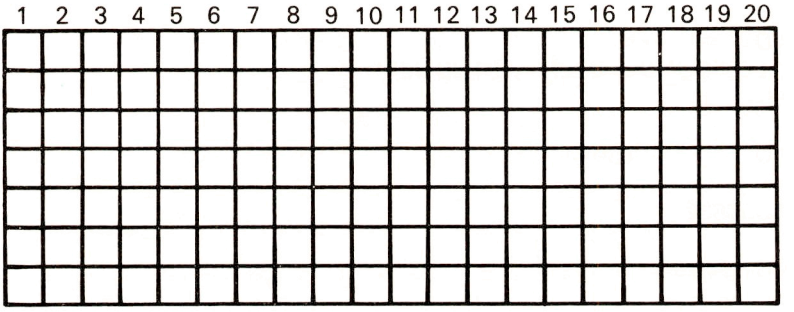
[K] Programma 3-1

ZOP 3

Bekijk het onderstaande programma en geef de uitvoer precies aan in het raster eronder. Elk vakje stelt een PRINT-positie voor.

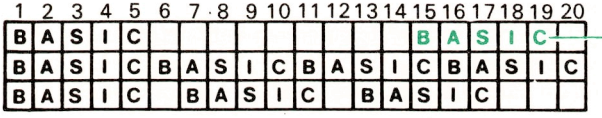
```
10  PRINT "PRINT LAYOUT"
20  LET B$ = "BASIC"
30  LET C$ = "CURSUS"
40  PRINT
50  PRINT B$, C$
60  PRINT
70  PRINT B$; C$
80  PRINT
90  PRINT B$; " "; C$
100 END
```

Programma 3-2.



Afb. 3-3.

ZOP 4
Schrijf een programma dat het volgende als uitvoer heeft:



De BBC-computer plaatst 2e maal 'BASIC' vanaf positie 11 regel 1; dat wil zeggen tweede printzone

Afb. 3-4.

3.4 INPUT "...";...

Tot nu toe hebben we PRINT"..." gebruikt als aanduiding voor ieder INPUT-statement. De meeste BASIC's kunnen echter deze "..."-string rechtstreeks in het INPUT-statement toepassen. Deze combinatie kan als volgt worden gerealiseerd:
in plaats van:

```
40  PRINT "UW TELEFOONNUMMER ALSTUBLIEFT"
50  INPUT T$
```

krijgen we:

```
40  INPUT "UW TELEFOONNUMMER ALSTUBLIEFT"; T$ *
```

Een vereenvoudiging van het programma en daardoor van de hoeveelheid gebruikte geheugenruimte, kan worden bereikt door 'HALLO' in een stringvariabele (H$) te plaatsen. Nu kan in plaats van het hele woord "HALLO" steeds H$ getikt worden, wat vijf letters scheelt.

```
10  REM ★★ TELE-ANTWOORD ★★
20  CLEAR 100
30  LET H$ = "HALLO"
40  PRINT H$
```

* Bij de BBC-computer dient u de puntkomma (;) weg te laten. In dit hoofdstuk zullen we u hieraan blijven herinneren, daarna moet u hier zelf aan denken.

```
50  INPUT "UW TELEFOONNUMMER A.U.B."; T$
60  PRINT
70  PRINT H$, T$
80  PRINT
90  PRINT H$; T$
100 PRINT
110 PRINT H$; " "; T$
120 END
```

hier geen vraagteken

Programma 3-3 De computer stelt de vragen.

```
RUN
HALLO
UW TELEFOONNUMMER A.U.B.? 58632

HALLO          58632

HALLO58632

HALLO 58632
```

computerreactie op regel 50

[K] Programma 3-3. BBC regel 50 :laat ; weg.

ZOP 5

Geef aan wat er op het scherm verschijnt als u het volgende programma uitvoert. Veronderstel dat uw naam Jan Haas is en uw leeftijd is 45.

```
10  LET T$ = "DANK U"
20  INPUT "WAT IS UW NAAM"; N$                    BBC regel 20: laat; weg
30  PRINT
40  INPUT "WAT IS UW LEEFTIJD"; A$                BBC regel 40: laat; weg
50  PRINT
60  PRINT T$, N$, A$
70  END
```

Programma 3-4.

3.5 Getallen en strings in PRINT-statements

We hadden het telefoonnummer uit het vorige programma ook in een numerieke variabele op kunnen slaan. Daarmee zouden echter al snel problemen ontstaan als het telefoonnummer te lang zou worden of als het spaties (of erger nog een streepje tussen kengetal en abonneenummer) zou bevatten. Laten we eens vergelijken hoe BASIC in dergelijke gevallen uw telefoonnummer zou uitvoeren. In onderstaand programma gebruiken we S$ om een schaalverdeling bovenaan het scherm af te drukken.

```
10  REM ★★ TELE-ANTWOORD ★★
20  CLEAR 100
30  LET H$ = "HALLO"
35  LET S$ = "123456789012345678901234 5"
```

```
40  PRINT H$
50  INPUT "UW TELEFOONNUMMER A.U.B."; T$
55  INPUT "WILT U DAT HERHALEN"; T
60  PRINT S$
70  PRINT H$, T$
75  PRINT H$, T
80  PRINT S$
90  PRINT H$; T$
95  PRINT H$; T
100 PRINT S$
110 PRINT H$; " "; T$
115 PRINT H$; " "; T
120 END
```

Programma 3-5 Strings en getallen afdrukken.

```
RUN
HALLO
UW TELEFOONNUMMER A.U.B.? 58632
WILT U DAT HERHALEN? 58632
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
HALLO                 58632
HALLO                   58632
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
HALLO58632



HALLO  58632
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5



HALLO  58632

HALLO   58632
```

Trace

60 S$ drukt een schaalverdeling af.

75 Merk op dat het eerste cijfer (5) op de zestiende positie wordt gezet, de vijftiende positie is gereserveerd voor

95 het teken van het getal in T. Als het getal positief is wordt het teken niet afgedrukt, maar

110 wordt er een spatie afgedrukt.

115 Regels 95 en 110 hebben beide hetzelfde effect.*

```
RUN
UW TELEFOONNUMMER A.U.B.?-58632
WILT U DAT HERHALEN?-58632
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
HALLO                -58632
HALLO                -58632
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
HALLO-58632
HALLO-58632
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
HALLO -58632
HALLO -58632
```

75

Zie wat er gebeurt als T$

95 en T beide - 58632 zijn.

115

* De BBC-computer geeft een ander resultaat. Probeer het maar eens en kijk wat er gebeurt.

[K] Programma 3-5. BBC regel 50 en 55: laat ; weg

ZOP 6

Schrijf een programma, om met uw naam en leeftijd als invoer een regel met : "Mijn naam is ... en ik ben ... jaar oud" af te drukken, met normale spatiëring.

Informatie-service

Onderstaande tekst is een voorbeeld van hoe een lay-out in BASIC kan worden gerealiseerd. We kunnen ons voorstellen dat in de niet al te verre toekomst onze TV, telefoon en computer onderling verbonden worden tot een intelligente terminal. Bij het zien van een interessante reclame of een artikel kunnen we een nummer 'draaien' en de volgende dialoog zou kunnen ontstaan:

```
RUN                                                          Trace
HALLO                                                          30
DIT IS UW INFORMATIE-SERVICE                                   40
                                                               50
BEANTWOORD S.V.P. DE VOLGENDE VRAGEN                           60

UW NAAM: J. JANSSEN                                            70
UW TELEFOONNUMMER: 015-897654                                  80
UW STRAATNAAM: VOORSTRAAT                                      90
UW HUISNUMMER: 888                                            100
UW POSTCODE: 4320 PD                                          110
UW WOONPLAATS: DELFT                                          120
                                                              130
                                                              140
                                                              150
                                                              160
DANK U VOOR UW BELANGSTELLING                                 170
                                                              180
UW PERSOONSGEGEVENS ZIJN ALS VOLGT DOOR ONS GENOTEERD:        190
NAAM: J. JANSSEN TELEFOON: 015-897654                         200
ADRES: VOORSTRAAT 888                                         210
       4320 PD DELFT                                          220
                                                              230
ONZE CATALOGUS                                                240
ZAL U SPOEDIG WORDEN TOEGESTUURD                              250
UW PERSOONSGEGEVENS WORDEN VERTROUWELIJK BEHANDELD            260
```

(Natuurlijk, in plaats van "zal u worden toegestuurd" kan ook "zal u overgeseind worden" worden gebruikt. In plaats van uw naam en adres kan dan volstaan worden met een inschrijf-antwoordnummer)

Deze dialoog werd verkregen met het volgende programma:

```
10  REM ★★ INFORMATIE-SERVICE ★★
20  CLEAR 100
30  PRINT "HALLO"
```

```
40  PRINT "DIT IS UW INFORMATIE-SERVICE"
50  PRINT
60  PRINT "BEANTWOORD S.V.P. DE VOLGENDE VRAGEN"
70  PRINT
80  INPUT "UW NAAM:"; N$
90  INPUT "UW TELEFOONNUMMER:"; T$
100 INPUT "UW STRAATNAAM:"; R$
110 INPUT "UW HUISNUMMER:"; H$
120 INPUT "UW POSTCODE:"; P$
130 INPUT "UW WOONPLAATS:"; C$
140 PRINT
150 PRINT
160 PRINT
170 PRINT "DANK U VOOR UW BELANGSTELLING"
180 PRINT
190 PRINT "UW PERSOONSGEGEVENS ZIJN ALS VOLGT DOOR ONS
    GENOTEERD:"
200 PRINT "NAAM:"; N$; "TELEFOON:"; T$
210 PRINT "ADRES:"; R$; " "; H$
220 PRINT "      "; P$; " "; C$
230 PRINT
240 PRINT "ONZE CATALOGUS"
250 PRINT "ZAL U SPOEDIG WORDEN TOEGESTUURD"
260 PRINT "UW PERSOONSGEGEVENS WORDEN VERTROUWELIJK
    BEHANDELD"
270 END
```

Programma 3-6 Informatie-service.

[K] Programma 3-6 (BBC regels 80,90,100,110,120,130: laat ; weg).

3.6 Standaardbrieven

Een informatie-service, zoals in de vorige paragraaf beschreven, zou in de nabije toekomst kunnen bestaan, maar "persoonlijke standaard"brieven zijn nu al gewoon. Dergelijke brieven worden gemaakt op een tekstverwerker, echter uw microcomputer heeft die faciliteiten niet. Desondanks kunnen, als we niet te hoge eisen stellen, ook in BASIC, programma's voor dergelijke brieven geschreven worden. Uw eigen brief kunt u straks in oefening 2 proberen.

Voorbeeld 1
De personeelsdienst van een bank ontvangt dagelijks vele verzoeken om werk. De dienst wil iedere sollicitant zo veel mogelijk in zijn toekomstige werkomgeving laten beoordelen. Daarvoor is een standaardbrief nodig die iedere sollicitant informeert over individuele details met betrekking tot de sollicitatie. Schrijf een programma dat dit voor u doet.

Oplossing
Het volgende programma:

```
10  REM ★★ STANDAARD BRIEF ★★
20  CLEAR 100
30  INPUT A$ ____________________________ naam sollicitant
40  INPUT B$ ____________________________ datum sollicitatie
50  INPUT C$ ____________________________ naam afdelingschef
60  INPUT D$ ____________________________ tijd gesprek
70  INPUT E$ ____________________________ datum gesprek
80  INPUT F$ ____________________________ vleugelnr. gebouw
90  INPUT G$ ____________________________ ondertekening
100 PRINT
110 PRINT
120 PRINT "GEACHTE HEER "; A$; ","
130 PRINT
140 PRINT "WIJ DANKEN U VOOR UW SCHRIJVEN D.D."; B$; "."
150 PRINT "WE NODIGEN U UIT VOOR EEN GESPREK MET"
160 PRINT C$; " OM "; D$; " DE "; E$
170 PRINT "IN VLEUGEL "; F$; " VAN ONS GEBOUW."
180 PRINT
190 PRINT "HOOGACHTEND,"
200 PRINT
210 PRINT
220 PRINT
230 PRINT G$
240 PRINT
250 PRINT
260 END
```

Programma 3-7 Standaard uitnodiging voor sollicitatie.

Dit programma levert het volgende resultaat op:

```
RUN                                                    Trace
? R. SMIT                                               30
? 12-02-1983                                            40
? MEVR. DE VRIES                                        50
? 9.30 UUR                                              60
? 17-2-1983                                             70
? DRIE                                                  80
? J. JANSSEN                                            90
                                                       100
                                                       110
GEACHTE HEER R. SMIT,                                  120
                                                       130
WIJ DANKEN U VOOR UW SCHRIJVEN D.D. 12-02-1983.        140
WE NODIGEN U UIT VOOR EEN GESPREK MET                  150
MEVR. DE VRIES OM 9.30 UUR DE 17-2-1983                160
IN VLEUGEL DRIE VAN ONS GEBOUW                         170
                                                       180
```

```
HOOGACHTEND,                                    190
                                                200
                                                210
                                                220
J. JANSSEN                                      230
```

De gebruiker van het programma zal nogal wat moeite hebben met het onthouden van de goede invoer op de goede plaats, daarom heeft het zin een geheugensteuntje te maken zoals hier afgebeeld:

```
RUN
? A$
? B$
? C$
? D$
? E$
? F$
? G$

GEACHTE A$,

WIJ DANKEN U VOOR UW SCHRIJVEN D.D. B$
WE NODIGEN U UIT VOOR EEN GESPREK MET
C$ OM D$ DE E$
IN VLEUGEL F$ VAN ONS GEBOUW.

HOOGACHTEND,


G$
```

[K] Programma 3-7.

Oefening 1
Een makelaar stuurt regelmatig een brief rond aan al zijn cliënten om na te gaan of zij nog belangstelling hebben voor onroerend goed en of de gegevens nog actueel zijn (soort huis, prijsindicatie enz.). Schrijf een BASIC-programma om zo'n brief te schrijven.

Oefening 2
We schrijven allemaal wel eens brieven waarin wij om informatie vragen bij een handelaar of postorderbedrijf. Schrijf een programma dat een dergelijke brief realiseert op een zo algemeen mogelijke wijze, waarbij u alleen nog maar de details behoeft in te vullen.

3.7 READ en DATA; files

READ en DATA

Tot nu toe hebben we alle data in de programma's alléén met het INPUT-statement ingevoerd. Een andere manier om gegevens in een programma in te voeren, is door deze in DATA-statements in het programma zelf op te nemen en dan door middel van een READ-statement in te lezen. Meestal worden deze DATA-statements aan het einde van het programma, of in een speciaal segment geplaatst.

Iedere keer dat de computer een READ-statement tegenkomt leest deze het volgende item uit de DATA-rij en plaatst deze in een door het READ-statement genoemde variabele.

In onderstaand voorbeeld gebeurt het volgende:

```
10  READ A$
20  READ B$
30  READ C$
100 DATA TOM, DICK
110 DATA HARRY
```

Programma 3-8.

Sommige computers vereisen " " om de strings in DATA-statements! bijv. 100 DATA "TOM", "DICK"

In regel 10 leest READ het eerste data-item en plaatst dit in variabele A$. Het eerste DATA-statement staat in regel 100 en bevat als eerste item TOM. De string TOM wordt nu in A$ opgeslagen. Het eerstvolgende READ-statement (regel 20) leest het tweede item uit regel 100: DICK. U kunt dit op uw eigen computer nagaan door de regels:

```
40  PRINT A$
50  PRINT B$
60  PRINT C$
```

in het bovenstaande programma op te nemen.

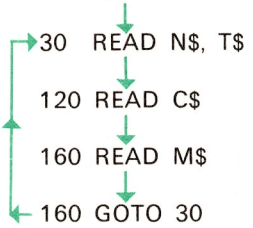
ZOP 7

Geef aan waarom het onderstaande programma niet werkt.

```
10  READ A$
20  READ B
30  READ C
40  READ D$
50  DATA PAUL, MARIE, 63
```

Programma 3-9.

In het volgende voorbeeld is het READ-statement in een lus opgenomen:




```
200 DATA ANTON, BENNIE, CEES
210 DATA DIRK, EDDIE
220 DATA GERARD
230 DATA HENDRIK, IZAAK, JANNES, KLAAS
240 DATA LEO, MARTIN
```

Dit gebeurt er:

Eerste lus

N$	leest	ANTON
T$	leest	BENNIE
C$	leest	CEES
M$	leest	DIRK

Geheugen na de eerste lus:

N$	ANTON
T$	BENNIE
C$	CEES
M$	DIRK

Afb. 3-5.

Tweede lus

N$	leest	EDDIE	(het eerstvolgende
T$	leest	GERARD	ongelezen DATA-item)
C$	leest	HENDRIK	
M$	leest	IZAAK	

Geheugen na de tweede lus:

N$	EDDIE
T$	GERARD
C$	HENDRIK
M$	IZAAK

Afb. 3-6.

ZOP 8
Geef aan hoe de geheugens in de volgende ronde gevuld zullen worden.

ZOP 9
Wat is de inhoud van A$, B$ en C$ nadat in het volgende programma alle DATA-items zijn gelezen?

```
10  READ A$
20  READ B$
30  READ C$
40  GOTO 20
50  DATA BEDELAAR, KLEERMAKER, SOLDAAT
60  DATA ZEEMAN, MILJONAIR
```

Programma 3-10.

Files en records

Vaak willen we grotere hoeveelheden data van dezelfde soort in een geheugen onderbrengen. Een telefoonboek is een goed voorbeeld van wat in computertaal een 'file' heet. Een file is een reeks gelijksoortige records. Elk record kan bijvoorbeeld van de volgende vorm zijn

NAAM	ADRES	TELEFOONNUMMER

Een record bestaat uit een aantal velden, in ons telefoonboek uit drie: naam, adres en telefoonnummer. Een record is een verzameling velden en een file is een verzameling records. Een telefoonboek is op alfabet geordend en heeft daardoor een eenvoudige structuur.

Strings vergelijken

Veronderstel dat u in uw microcomputer een eenvoudige telefoongids heeft geprogrammeerd, en u wilt weten of SMIT in uw file voorkomt. Daartoe moet de computer SMIT vergelijken met alle andere strings in het naam-veld van uw telefoon-file. Dat kan betrekkelijk eenvoudig omdat elke letter van een string in de computer wordt voorgesteld door een binair getal:

A = 100 0001
B = 100 0010
enz.
zie voor een volledige lijst met binaire codes de appendix.

Hierdoor worden woorden die wij op papier in een alfabetische volgorde hebben gezet, door de computer voorgesteld als binaire getallen in numerieke volgorde.
Bijvoorbeeld:

A$ = KAT
B$ = HOND
C$ = KAT
D$ = VIS
E$ = KATTEN

dan is
A$ = C$
maar
D$ > A$ (VIS komt later dan KAT)
en
E$ > C$ (de extra -TEN maakt dat E$ later komt)

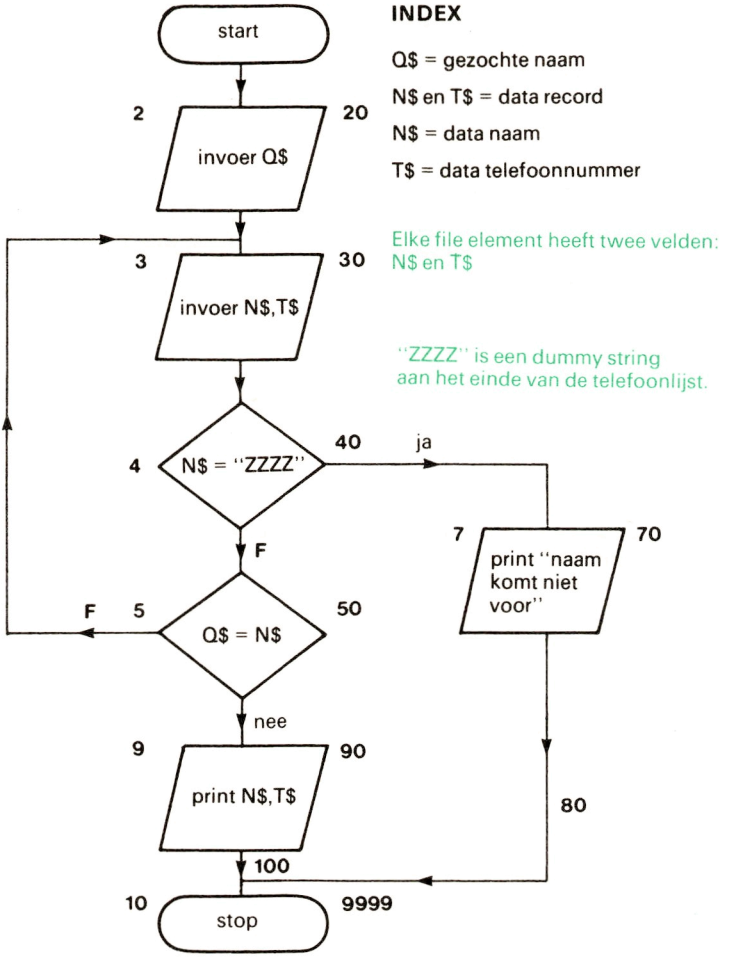
We zullen dit eens toepassen in enkele voorbeelden.

Voorbeeld 2
Creëer een datafile met namen en bijbehorende telefoonnummers. Schrijf vervolgens een programma dat in de file naar een bepaalde naam zoekt en indien gevonden, het bijbehorende telefoonnummer afdrukt.

Oplossing
We kunnen het programma beschrijven met het volgende algoritme:
1. Start.
2. Voer een naam in.
3. Lees het volgende record uit de datafile.
4. Als het einde van de file niet bereikt is (naam = ZZZ) print dan "naam komt niet voor" en ga naar 7, ga anders naar 5.
5. Als recordnaam = gezochte naam print dan naam en nummer en ga naar 7, ga anders naar 6.
6. Ga terug naar 3.
7. Stop.

BASIC laat echter geen complexe statements toe als in punt 4 en 5, daarom splitsen we deze nader uit:

1. Start.
2. Voer een naam in.
3. Lees het volgende record uit de datafile.
4. Als het einde van de file bereikt is, ga dan naar 7, ga anders naar 5.
5. Als record naam = gezochte naam ga dan naar 9, ga andersnaar 6.
6. Ga terug naar 3.
7. Print "naam komt niet voor".
8. Stop.
9. Print naam en nummer.
10. Stop.



Afb. 3-7. Zoeken in een telefoonlijst.

Ieder record bestaat uit twee velden: naam en telefoonnummer.

Afb. 3-8. bijv.

Veld 1	Veld 2
Naam N$	Telefoonnummer T$
HENRY	4539

Deze twee velden moeten beide gelezen worden om niet "uit de pas" te raken met het READ-statement. Daarom:

READ N$, T$ (lees twee datavelden)

en de bijbehorende DATA-statements zijn van de volgende vorm:

DATA ANTON, 1234

Merk op dat het einde van de file wordt gemarkeerd met een DATA-statement (regel 310 in het volgende programma) en dat dit data moet bevatten voor zowel T$ als N$. Ontbreekt de laatste dan wordt het programma beëindigd met een foutmelding. Daarom voorzien we het laatste DATA-statement van een dummy "telefoonnummer":

DAT ZZZZ, EINDE FILE

en niet alleen maar

DATA ZZZZ

(Sommige BASIC's vereisen dat de strings in DATA-statements ook tussen " " staan. Een niet-ingesloten string is eigenlijk een "open string", eindigend op een onbeperkt aantal spaties).

```
10  REM ★★ TELEFOONGIDS ★★
15  CLEAR 100
20  INPUT "NAAM VAN GEZOCHTE PERSOON:"; Q$
30  READ N$, T$                                    READ
40  IF N$ = "ZZZZ" THEN 70
50  IF Q$ = N$ THEN 90
60  GOTO 30
70  PRINT Q$; " KOMT NIET VOOR"
80  GOTO 9999
90  PRINT Q$; " 'S NUMMER IS: "; T$
100 GOTO 9999
200 DATA BERT, 1234
210 DATA CAREL, 9823
220 DATA DICK, 1850
230 DATA EDDIE, 7294
240 DATA GERRIT, 5821
250 DATA HENRY, 4539                               DATA
260 DATA MARIANNE, 7830
270 DATA PAUL, 1383
280 DATA SIMON, 1146
290 DATA WALTER, 5529
300 DATA WILLEM, 9936
310 DATA ZZZZ, EINDE LIJST
9999 END
```

Programma 3-11 Telefoongids.

```
RUN
NAAM VAN GEZOCHTE PERSOON: SIMON
SIMON'S NUMMER IS: 1146

RUN
NAAM VAN GEZOCHTE PERSOON: PAUL
PAUL'S NUMMER IS : 1383

RUN
NAAM VAN GEZOCHTE PERSOON: RUUD
RUUD KOMT NIET VOOR

RUN
NAAM VAN GEZOCHTE PERSOON: KAREL
KAREL KOMT NIET VOOR
```

Gedurende de vierde RUN vroegen we om de naam KAREL, terwijl alleen de naam CAREL in de lijst is opgenomen. Onze computer vergelijkt de naam met de lijst en constateert dat deze niet aanwezig is. Als we zelf de lijst met de hand hadden nagelopen was de vergissing in spelling ons onmiddellijk opgevallen. Bij het zoeken met de computer gebeurt dat nooit. Om alsnog de goede persoon te vinden, moeten we met verschillende spellingen het programma uitproberen.

[K] Programma 3-11 (regel 20 BBC: laat; weg)

ZOP 10
Welke veranderingen moet u in het programma aanbrengen om in omgekeerde volgorde te kunnen zoeken? (nummer in – naam uit).

3.8 Sorteren

Misschien heeft u opgemerkt dat de telefoonlijst uit voorbeeld 2 in alfabetische volgorde stond. Dat is heel gewoon en in de praktijk zou het zeer lastig zijn indien dit niet zo was. Echter, ons programma gebruikte deze informatie niet; het zocht recht toe recht aan tot de naam gevonden was. Ons algoritme zou even goed met een niet-alfabetische lijst hebben gefunctioneerd. Later gaan we verder in op sorteren en zoeken van data-items en zult u de voordelen van gesorteerde lijsten snel leren waarderen. Laten we daarmee eens beginnen.

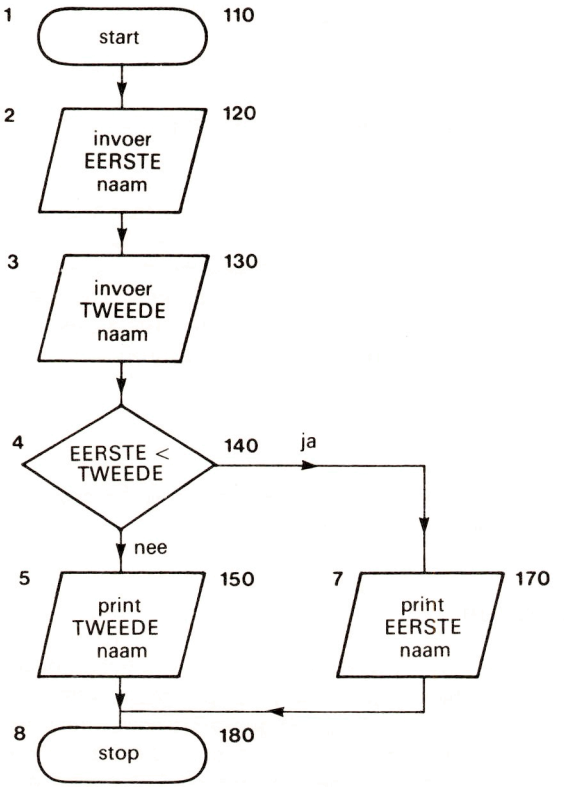
Voorbeeld 3
Schrijf een BASIC-programma dat twee namen inleest en dat de naam die volgens de alfabetische volgorde het eerst moet komen, afdrukt.

Oplossing
Beschrijvend algorithme:

1. Start.
2. Voer eerste naam in.

3. Voer tweede naam in.
4. Als eerste naam < tweede naam ga dan 7, ga anders naar 5.
5. Print tweede naam.
6. Ga naar 8.
7. Print eerste naam.
8. Stop.

Een stroomdiagram voor dit algoritme is afgebeeld in figuur 3-9.



Afb. 3-9. Bepalen van de alfabetische volgorde.

```
100 CLEAR 100
110 REM ★★ EERSTE IN HET ALFABET ★★
120 INPUT "EERSTE NAAM: "; A$
130 INPUT "TWEEDE NAAM: "; B$
140 IF A$ < B$ THEN 170
150 PRINT "IN HET ALFABET KOMT EERST: "; B$
160 GOTO 180
170 PRINT "IN HET ALFABET KOMT EERST: "; A$
180 END
```

Programma 3-12.

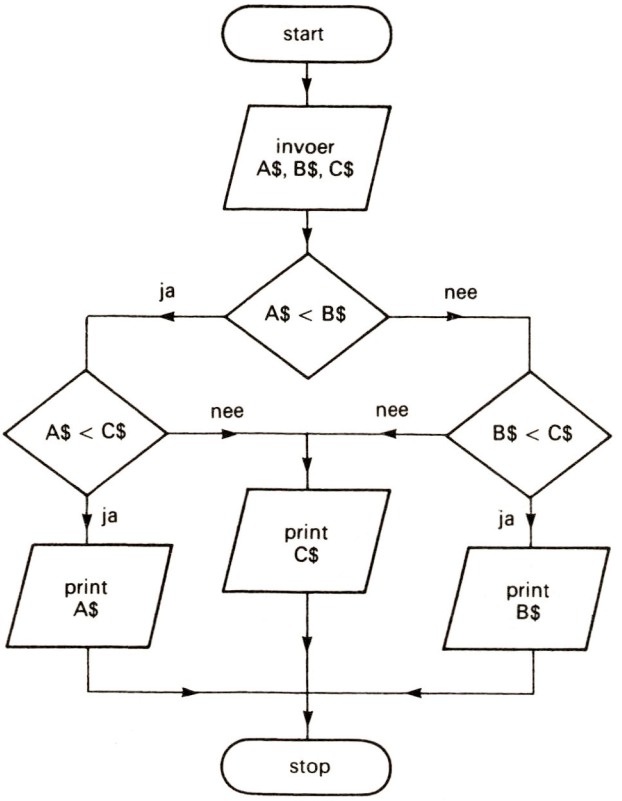
```
RUN
EERSTE NAAM: KUIPER
TWEEDE NAAM: PIETERSEN
IN HET ALFABET KOMT EERST: KUIPER
```

```
RUN
EERSTE NAAM: JANSSEN
TWEEDE NAAM: DEEN
IN HET ALFABET KOMT
EERST: DEEN
```

[K] Programma 3-12 (BBC regel 120, 130: laat; weg)

Drie namen

Veronderstel nu dat we drie namen willen invoeren. Een mogelijke oplossing daarvoor zou zijn het algoritme uit stroomdiagram 3-9 uit te breiden met een derde tak:



Afb. 3-10. Welk van de drie komt het eerst?

Als we aan een aantal studenten vragen dit probleem op te lossen komen de meesten aan met een soortgelijke oplossing. Het werkt perfect maar is niet geschikt voor uitbreiding. Stelt u zich een dergelijke aanpak eens voor met 4, 5 of laat staan 10 namen. De basis van het probleem is dat elke naam zijn eigen individuele variabelenaam heeft, en dat deze afzonderlijk moet worden afgedrukt.

Een eenvoudiger methode om dit probleem op te lossen, is de namen één voor één in te voeren en steeds de alfabetisch laagst-geplaatste in A$ op te bergen. Daarmee verliest het programma alle niet benodigde gegevens. In de volgende oefening gaat u zelf proberen met deze methode een sorteerprogramma te schrijven.

Oefening 3
Schrijf een BASIC-programma om drie namen in te lezen en de op alfabet eerstvoorkomende naam uit te printen. Gebruik daarvoor de zojuist geïntroduceerde methode.

Oefening 4
Hieronder is een lijst met landen en hun hoofdsteden afgedrukt. Schrijf een BASIC-programma met deze lijst als uitgangspunt dat op een quiz-achtige wijze de gebruiker confronteert met een landnaam, waarop deze met de juiste hoofdstad dient te antwoorden. Is het antwoord fout, druk dan het juiste antwoord af.

```
140 DATA FRANKRIJK, PARIJS
150 DATA WEST-DUITSLAND, BONN
160 DATA NEDERLAND, AMSTERDAM
170 DATA POLEN, WARSCHAU
180 DATA ITALIË, ROME
190 DATA SPANJE, MADRID
200 DATA PORTUGAL, LISSABON
210 DATA HONGARIJE, BOEDAPEST
220 DATA DENEMARKEN, KOPENHAGEN
230 DATA NOORWEGEN, OSLO
240 DATA ZZZZ, EINDE LIJST
```

Opgave 3

1. Schrijf een algoritme en teken een stroomdiagram bij het BASIC-programma dat u in oefening 3 heeft gemaakt.
2. Pas het programma van oefening 4 zó aan dat het aantal goede en foute antwoorden wordt bijgehouden en druk de resultaten na afloop af.
3. Schrijf een algorithme en BASIC-programma dat de volgende taken uitvoert: Sla een aantal woorden in individuele letters op in DATA-statements, bijv. de woorden "ALGORITHME" en "STROOMDIAGRAM":

```
900 DATA A,L,G,O,R,I,T,H,M,E
910 DATA S,T,R,O,O,M,D,I,A,G,R,A,M
```

Laat het aantal klinkers en medeklinkers uittellen en druk de aantallen af, tezamen met de verhouding klinkers/medeklinkers.

Samenvatting van hoofdstuk 3

Aan het einde van dit hoofdstuk dient u de volgende onderwerpen in eenvoudige BASIC-programma's te kunnen toepassen:

- Stringvariabelen
- CLEAR
- PRINT ...;...
- INPUT "...";...
- READ
- DATA met een enkel veld
- IF A$=B$ THEN...
- DATA met meer dan één veld
- Eenvoudige sorteerprocedures

Antwoorden op ZOP's en oefeningen

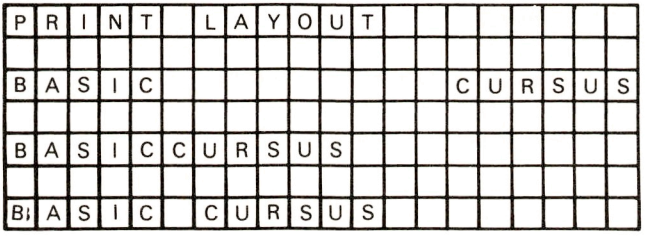
ZOP 1

Goede stringvariabelen zijn: A$ en T7$
Van de rest zijn M8 en B9 numerieke variabelen en de resterende namen zijn alle fout.

ZOP 2

(a), (b) en (e) zijn correct, maar (e) is niet toelaatbaar in een mini-BASIC computer.
(c) is fout: M$ is een stringvariabele en 9583 is een nummer.
(d) is fout: K8 is een numerieke variabele.

ZOP 3



Afb. 3-11.

(antwoord aangegeven voor standaard printzones)

ZOP 4

```
10  LET B$ = "BASIC"
20  PRINT B$, B$
30  PRINT B$; B$; B$; B$
40  PRINT B$; " "; B$; " "; B$
50  END
```

Programma 3-13.

ZOP 5

```
WAT IS UW NAAM? JAN HAAS
WAT IS UW LEEFTIJD? 45
DANK U          JAN HAAS          45
```

ZOP 6

```
10  INPUT "WAT IS UW NAAM"; N$
20  INPUT "WAT IS UW LEEFTIJD"; A
```

```
30  PRINT "MIJN NAAM IS"; N$; "EN IK BEN"; A; "JAAR OUD"
40  END
```

Programma 3-14.

Oefening 1
Omdat oefening 1 en 2 erg op elkaar lijken, hebben we het antwoord hier niet afgedrukt.

Oefening 2
(Laat voor de BBC-computer de ; uit de INPUT-regels weg)

```
10  REM ★★ INFORMATIE AANVRAAG ★★
20  CLEAR 500
30  PRINT "GEADRESSEERDE : "
40  PRINT
50  INPUT "NAAM : "; N$
60  INPUT "STRAAT : "; S$
70  INPUT "WOONPLAATS : "; T$
80  PRINT
90  INPUT "DATUM VOOR BRIEFHOOFD : "; D$
100 PRINT
110 PRINT "PRODUKTOMSCHRIJVING"
120 PRINT
130 PRINT "ONDERWERP"
140 INPUT I$
150 PRINT "INFORMATIEBRON SOORT/NAAM"
160 INPUT A$
170 INPUT "DATUM BRON : "; E$
180 PRINT
190 PRINT
200 PRINT
210 PRINT
220 PRINT N$
230 PRINT S$
240 PRINT T$
250 PRINT
260 PRINT
270 PRINT D$
280 PRINT
290 PRINT
300 PRINT "GEACHTE HEER,"
310 PRINT
320 PRINT "WILT U ZO VRIENDELIJK ZIJN MIJ NADER TE INFORMEREN
    BETREFFENDE :"
330 PRINT I$ ","
340 PRINT "ZOALS DOOR U GENOEMD IN :"
350 PRINT A$
360 PRINT "GEDATEERD : "; E$; "."
370 PRINT
```

```
380 PRINT
390 PRINT "HOOGACHTEND,"
400 PRINT
410 PRINT
420 PRINT
430 PRINT "J. JANSSEN"
```

Programma 3-15.

```
RUN
GEADRESSEERDE:

NAAM: SOFTWARE B.V.
STRAAT: RAMSTRAAT 88
WOONPLAATS: DELFT

DATUM VOOR BRIEFHOOFD: 16-02-1983

PRODUKTOMSCHRIJVING
ONDERWERP
? UW WOORDPROCESSOR "VULPEN"
INFORMATIEBRON SOORT/NAAM
? UW FOLDER
DATUM BRON: DEC. 1982


SOFTWARE B.V.
RAMSTRAAT 88
DELFT

16-02-1983

GEACHTE HEER,

WILT U ZO VRIENDELIJK ZIJN MIJ NADER TE INFORMEREN BETREFFENDE:
UW WOORDPROCESSOR "VULPEN",
ZOALS DOOR U GENOEMD IN: UW FOLDER
GEDATEERD: DEC. 1982.

HOOGACHTEND


J. JANSSEN
```

ZOP 7

Regel 20 probeert het tweede DATA-item "MARIE" in een numerieke variabele te lezen. Dat gaat niet en dus geeft de computer een foutmelding. Om dit te kunnen oplossen had regel 20 moeten luiden 20 READ B$.

Regel 40 vraagt om een vierde DATA-item, er zijn echter in regel 50 slechts drie items beschikbaar.

ZOP 8

N$	JANNES
T$	KLAAS
C$	LEO
M$	MARTIN

Afb. 3-12.

ZOP 9
A$ BEDELAAR
B$ ZEEMAN
C$ MILJONAIR

Als u dit programma uitvoert, krijgt u een "OUT OF DATA ERROR" of soortgelijke melding. Waarom?

ZOP 10
```
20  INPUT "TELEFOONNUMMER GEZOCHTE PERSOON : "; A$
50  IF A$ = T$ THEN 90
70  PRINT "HET NUMMER: "; A$; "STAAT NIET IN UW LIJST"
90  PRINT "HET NUMMER: "; A$; "BEHOORT BIJ : "; N$
```

Oefening 3
Een eenvoudige methode is in het onderstaande programma afgedrukt. De namen kunnen één voor één worden ingevoerd en de tot dan toe laatste naam wordt in variabele A$ opgeslagen. Alle andere voorafgaande gegevens gaan daarmee verloren.

```
10  REM ★★ EERSTE NAAM IN HET ALFABET ★★
15  CLEAR 100
20  INPUT "EERSTE NAAM: "; A$
30  INPUT "VOLGENDE NAAM: "; B$
40  IF B$ = "ZZZZ" THEN 90
50  IF A$ < B$ THEN 70
60  LET A$ = B$
70  PRINT "EERSTE TOT NU TOE IS: "; A$
80  GOTO 30
90  PRINT A$; " WAS DE ALLEREERSTE NAAM"
100 END
```

Programma 3-16.

```
RUN
EERSTE NAAM : TED
VOLGENDE NAAM : SIMON
EERSTE TOT NU TOE IS : SIMON
VOLGENDE NAAM : JULIEN
EERSTE TOT NU TOE IS : JULIEN
```

```
VOLGENDE NAAM : PETER
EERSTE TOT NU TOE IS : JULIEN
VOLGENDE NAAM : FRED
EERSTE TOT NU TOE IS : FRED
VOLGENDE NAAM : BERT
EERSTE TOT NU TOE IS : BERT
VOLGENDE NAAM : RON
EERSTE TOT NU TOE IS : BERT
VOLGENDE NAAM : AREND
EERSTE TOT NU TOE IS : AREND
VOLGENDE NAAM : ZZZZ
AREND WAS DE ALLEREERSTE NAAM
```

Oefening 4

```
10  REM ★★ EUROPESE HOOFDSTEDEN ★★
20  CLEAR 500
30  READ C$, T$
40  IF C$ = "ZZZZ" THEN 130
50  PRINT "WAT IS DE HOOFDSTAD VAN : "; C$
60  INPUT A$
70  IF A$ = T$ THEN 110
80  PRINT "NEE SORRY! DE HOOFDSTAD VAN "
90  PRINT C$; " IS "; T$
100 GOTO 30
110 PRINT "JA, DAT IS GOED!"
120 GOTO 30
130 PRINT "DIT IS HET EINDE VAN DE QUIZ"
140 DATA FRANKRIJK, PARIJS
150 DATA WEST DUITSLAND, BONN
160 DATA NEDERLAND, AMSTERDAM
170 DATA POLEN, WARSCHAU
180 DATA ITALIE, ROME
190 DATA SPANJE, MADRID
200 DATA PORTUGAL, LISSABON
210 DATA HONGARIJE, BOEDAPEST
220 DATA DENEMARKEN, KOPENHAGEN
230 DATA NOORWEGEN, OSLO
240 DATA ZZZZ, EINDE LIJST
```

Programma 3-17.

```
RUN
WAT IS DE HOOFDSTAD VAN : FRANKRIJK
? PARIJS
JA, DAT IS GOED!
WAT IS DE HOOFDSTAD VAN : WEST DUITSLAND
? BERLIJN
NEE SORRY! DE HOOFDSTAD VAN
WEST DUITSLAND IS BONN
```

```
WAT IS DE HOOFDSTAD VAN : NEDERLAND
? DEN HAAG
NEE SORRY! DE HOOFDSTAD VAN
NEDERLAND IS AMSTERDAM
WAT IS DE HOOFDSTAD VAN : POLEN
? WARSCHAU
JA, DAT IS GOED!
WAT IS DE HOOFDSTAD VAN : ITALIE
? ROME
JA, DAT IS GOED!
WAT IS DE HOOFDSTAD VAN : SPANJE
? MADRID
JA, DAT IS GOED!
WAT IS DE HOOFDSTAND VAN : PORTUGAL
? LISSABON
JA, DAT IS GOED!
WAT IS DE HOOFDSTAD VAN : HONGARIJE
? PRAAG
NEE SORRY! DE HOOFDSTAD VAN
HONGARIJE IS BOEDAPEST
WAT IS DE HOOFDSTAD VAN : DENEMARKEN
? COOPENHAGUE
NEE SORRY! DE HOOFDSTAD VAN
DENEMARKEN IS KOPENHAGEN
WAT IS DE HOOFDSTAD VAN : NOORWEGEN
? OSLO
JA, DAT IS GOED!
DIT IS HET EINDE VAN DE QUIZ
```

Appendix

American Standard Code for Information Interchange ofwel ASCII-code

Het gedeelte van de code dat voor ons van belang is, is hierbij afgedrukt:

@	100 0000	0	011 0000
A	100 0001	1	011 0001
B	100 0010	2	011 0010
C	100 0011	3	011 0011
D	100 0100	4	011 0100
E	100 0101	5	011 0101
F	100 0110	6	011 0110
G	100 0111	7	011 0111
H	100 1000	8	011 1000
I	100 1001	9	011 1001
J	100 1010	:	011 1010
K	100 1011	;	011 1011
L	100 1100	<	011 1100
M	100 1101	=	011 1101
N	100 1110	>	011 1110

O	100 1111	?	011 1111
P	101 0000	Spatie	010 0000
Q	101 0001	!	010 0001
R	101 0010	"	010 0010
S	101 0011	#	010 0011
T	101 0100	$	010 0100
U	101 0101	%	010 0101
V	101 0110	&	010 0110
W	101 0111	'	010 0111
X	101 1000	(	010 1000
Y	101 1001	)	010 1001
Z	101 1010	★	010 1010
[	101 1011	+	010 1011
\	101 1100	,	010 1100
]	101 1101	-	010 1101
↑	101 1110	.	010 1110
–	101 1111	/	010 1111

a) Met 7 elektronische schakelaars kan de aan (1) of uit (0) stand van elk van de karakters worden gerepresenteerd.

b) Zonder iets van binaire getallen te weten kunnen we deze codes toch gebruiken voor het vergelijken in sorteerroutines: door de decimale waarden van de getallen met elkaar te vergelijken.

A = "miljoen en een"
B = "miljoen en twee"
Z = "miljoen elfduizend en tien"

Hoewel dat rekenkundig incorrect is, voldoet de methode om vast te stellen dat A < B < C < D .... < Y < Z !!

4 Lijsten

4.1 Variabelen

In het begin van dit boek spraken we steeds over geheugenlocaties om getallen en strings in op te bergen. Deze geheugenlocaties kunnen gedurende het verloop van een programma van waarde veranderen; om die reden hebben we zo nu en dan al het woord variabele gebruikt in plaats van locatie. Van nu af aan heten alle geheugenlocaties variabelen. Dus de 286 namen:

A, A0, A1, A2.....,A9, B, B0,.....Z8, Z9

heten 'numerieke variabelen', en

A$, A0$, A1$.....,A9$, B$, B0$, B1$,.....Z8$, Z9$

heten 'string-variabelen'. Deze variabelen kunnen in een programma op dezelfde wijze worden gebruikt als variabelen in wiskundige vergelijkingen.

4.2 Lijsten

Lijsten kennen we allemaal. De meest bekende gebruiken we allemaal regelmatig bijvoorbeeld in de supermarkt. We gaan er in met een lijstje boodschappen en we verlaten de zaak met een lijstje met de prijzen van de boodschappen die we gekocht hebben. De samenstelling van beide lijstjes komt nogal toevallig tot stand, maar we zouden daar natuurlijk enige ordening in aan kunnen brengen. We zouden bijvoorbeeld alle produkten in volgorde van prijs aan de kassier(e) kunnen overhandigen en zodoende een prijslijst kunnen krijgen waarop de prijzen in volgorde zijn gesorteerd. We hadden ze ook kunnen ordenen in volgorde van gewicht of kwaliteit. Het feit dat een lijst geordend kan zijn, is de belangrijkste en nuttigste eigenschap ervan.

4.3 Lijstvariabelen of arrays

De meeste data die we tot nu toe hebben gebruikt, kon op de een of andere wijze worden ingedeeld of geordend. We keken naar scores, namen en bijbehorende telefoonnummers, landen en hoofdsteden enz. Vrijwel alle gegevens kunnen op een of andere wijze worden geklassificeerd. Als we voor een bepaald doel gegevens verzamelen, is dat doel het gemeenschappelijke kenmerk die de gegevens met elkaar verbindt.Het heeft duidelijke voordelen de variabelenaam voor een dergelijke groep gegevens zo te kiezen dat deze de verbondenheid ervan benadrukt. Beter nog zou het

zijn als de variabele op een of andere wijze in haar naam de positie van het lijstelement kon verwerken.

Laten we eens kijken naar de cijfers uit het rapportenboekje van een leraar. De cijfers vormen vanzelfsprekend een lijst en onze BASIC zou daar op de volgende wijze variabelen voor kunnen toewijzen:

Cijfers	**Variabele-symbool**
Cijfer van leerling 1	C(1) = 7
Cijfer van leerling 2	C(2) = 5
Cijfer van leerling 3	C(3) = 9

C(1), C(2) en C(3) zijn aparte geheugenlocaties en dus aparte variabelen. Elk van de 572 variabelenamen kan worden gebruikt als lijst door er een getal tussen haakjes achter te plaatsen.

Lijst-naam	**Geheugenlocaties in die lijst**
C(I)	C(1), C(2), C(3),....
A0(I)	A0(1), A0(2), A0(3)...
C$(I)	C$(1), C$(2), C$(3)...

Het getal tussen haakjes is de index die elke positieve en gehele waarde tussen 1 en een bepaalde limiet kan hebben. De naam van de lijst als geheel is C-lijst; van cijfer-lijst. Als uw BASIC het toelaat kunt u de lijst natuurlijk ook CIJFER (I) noemen. De index I kan ook als variabele gebruikt worden.

String-lijsten

Zoals u hierboven al heeft kunnen zien, kunnen ook strings in een lijst worden opgenomen op dezelfde wijze als numerieke waarden. Dus als C(I) een lijst van cijfers is, dan kan C$(I) een lijst zijn van namen.
Voorbeeld:

Index	Gegeven	Variabele naam
1	Jonas	N$(1) of NAAM $(1)
2	Alex	N$(2) of NAAM $(2)
3	Simon	N$(3) of NAAM $(3)
enz.	enz.	enz.

Afb. 4-1. String-lijst namen.

Lijsten en arrays

Een tabel met gegevens, zoals in afbeelding 4-1 wordt vaak een array genoemd. Zoals de gegevens hier zijn geordend, in rijen en kolommen, kan zo'n tabel een twee-dimensionale array worden genoemd. Een lijst met 1 reeks met gegevens heet dan ook wel een één-dimensionale array. Hoe we in BASIC van twee-dimensionale arrays gebruik kunnen maken behandelen we in een volgend hoofdstuk.

Het statement DIM of: hoe lang is onze lijst?
Zo lang als u verkiest! We hadden al gezegd dat de index elke positieve en gehele waarde mag hebben binnen de geheugenlimieten van uw computer. Het is logisch dat we het systeem van te voren even waarschuwen hoeveel gegevens we van plan zijn in een bepaalde lijst of array op te slaan. Sommige computers hebben dat niet nodig voor lijsten korter dan 10, andere, waaronder de BBC-computer, vereisen een melding voor elke array. Daartoe gebruiken we het DIMENSION-statement.

regelnummer DIM A(X)

waarin A de naam van de lijst en X de lengte ervan voorstelt.

Dit DIM-statement moet in het programma voorkomen voordat de desbetreffende lijst voor het eerst wordt gebruikt. Ook als uw computer voor kleine lijsten niet behoeft te worden 'gewaarschuwd' is het een goede gewoonte dat toch te doen. Het aanmelden van een lijst aan het begin van een programma heet 'declareren'.

Items en indexnummers
Met het volgende pen-en-papier voorbeeld verkrijgt u een beter begrip in de termen item en index. Het legt ook de basis voor een sorteerprogramma dat later in dit hoofdstuk aan de orde komt.

Voorbeeld 1
Verhuis in de volgende lijst het item met de laagste waarde naar positie 1, door het eerste element met de achtereenvolgende elementen te vergelijken en om te ruilen indien de nieuwe waarde kleiner is dan de huidige eerste waarde. (Dit is makkelijker te doen dan te beschrijven) De lijst is : 3, 42, −8, 9, −11

start		stapsgewijs vergelijken en verwisselen			
positie of index	gegeven	1e stap vergelijk	2e stap verg. en verw.	3e stap vergelijk	4e stap verg.en verw.
1	3	3	−8	−8	−11
2	42	42	42	42	42
3	−8	−8	3	3	3
4	9	9	9	9	9
5	−11	−11	−11	−11	−8

Afb. 4-2. Elementair sorteren.

ZOP 1
Voer dezelfde procedure als in voorbeeld 1 uit met de volgende lijst getallen:
6, 8, 4, 7, 3, 9, 1.

4.4 In- en uitvoer van lijsten

Voordat we met items in een lijst kunnen manipuleren, moeten we deze aan de computer aanbieden. Nadat de lijst door het programma is bewerkt, moet deze meestal weer worden afgedrukt.

Voorbeeld 2
Schrijf een BASIC-programma dat drie getallen inleest en deze vervolgens in omgekeerde volgorde weer afdrukt.

Oplossing
We noemen de lijst A(I). Het programma wordt dan:

```
10  REM ★★ VOORBEELD ★★
15  DIM A(3)
20  INPUT A(1)
30  INPUT A(2)
40  INPUT A(3)
50  PRINT
60  PRINT
70  PRINT A(3), A(2), A(1)
80  END
```

Programma 4-1 Lijst in omgekeerde volgorde afdrukken.

```
RUN
? 29
? 32
? -17

-17          32          29
```

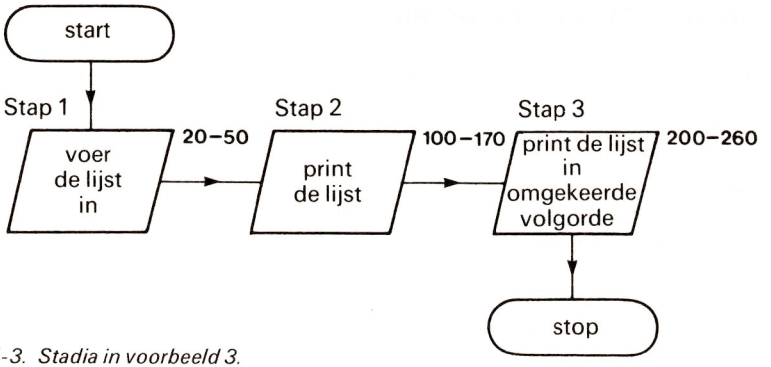
[K] Programma 4-1

In dit programma wordt gedaan wat is gevraagd, maar over de oplossing kunnen we niet naar huis schrijven. Hetzelfde hadden we ook met de technieken uit de vorige hoofdstukken kunnen oplossen, simpelweg met drie variabelen X,Y,Z. De toepassing van een teller maakt dat we de lijst op volgorde kunnen inlezen en door de teller terug te laten tellen, kunnen we de lijst in omgekeerde volgorde laten afdrukken. In het volgende voorbeeld wordt deze verfijning aangebracht.

Lijsten nummeren

Voorbeeld 3
Schrijf een programma dat vijf getallen inleest in een lijst, deze vervolgens genummerd afdrukt en tot slot de volgorde omkeert en weer afdrukt.

Oplossing
Zoals in de vraagstelling aangegeven, bestaat het probleem, en dus de oplossing uit drie delen. In een stroomdiagram afgebeeld:



Afb. 4-3. Stadia in voorbeeld 3.

Stap 1
We gebruiken de variabele C om de elementen in de lijst voor ons te tellen gedurende de invoerfase; tevens functioneert C als index voor de lijst L(C).

```
10  REM ★★ LIJST INLEZEN ★★
15  DIM L(5)
20  LET C = 1
30  INPUT "VOLGENDE NUMMER: "; L(C)
40  LET C = C + 1
50  IF C < = 5 THEN 30
```

Programma 4-2 Lijst aftellen tijdens invoer.

Stap 2
De tabel heeft de vorm uit ZOP 1. Een eenvoudige PRINT-lus met PRINT...,... kan de tabel afdrukken.

```
100 REM ★★ AFDRUKKEN LIJST ★★
110 PRINT
120 PRINT
130 PRINT "INDEX", "ITEM"
140 LET C = 1
150 PRINT C, L(C)
160 LET C = C + 1
170 IF C < = 5 THEN 150
```

print lijst

Programma 4-3 Printen van de lijst in volgorde van invoer.

Stap 3
Nu laten we C van 5 naar 1 terugtellen en printen de lijst tegelijkertijd uit.

```
200 REM ★★ LIJST IN OMGEKEERDE VOLGORDE ★★
210 PRINT
220 PRINT
```

```
230 LET C = 5
240 PRINT L(C)
250 LET C = C - 1
260 IF C > = 1 THEN 240
270 END
```

print lijst in omgekeerde volgorde

Programma 4-4 Printen in omgekeerde volgorde.

We hebben nu drie programmadelen die samen de oplossing voor ons probleem vormen. Alles wat ons nog rest, zijn de drie als volgt samen te voegen. De paar aanpassingen zijn in de kantlijn toegelicht.

```
10  REM ★★ LIJST INLEZEN ★★
15  DIM L(5)
20  LET C = 1
30  INPUT "VOLGENDE NUMMER: "; L(C)
40  LET C = C + 1
50  IF C < = 5 THEN 30
60  REM
70  REM
100 REM ★★ AFDRUKKEN LIJST ★★
110 PRINT
120 PRINT
130 PRINT "INDEX", "ITEM"
140 LET D = 1
150 PRINT D, L(D)
160 LET D = D + 1
170 IF D < = 5 THEN 150
180 REM
190 REM
200 REM ★★ LIJST IN OMGEKEERDE VOLGORDE ★★
210 PRINT
220 PRINT
230 LET E = 5
240 PRINT L(E)
250 LET E = E - 1
260 IF E > = 1 THEN 240
270 END
```

De REM-statements laten duidelijk de verschillende stappen zien

We hebben hier D gebruikt als index om te laten zien dat de naam onbelangrijk is; alleen de waarde telt.

Hier hebben we E als index gebruikt.

Programma 4-5 Het gehele omkeerprogramma.

```
RUN
VOLGENDE NUMMER: -8
VOLGENDE NUMMER: 15
VOLGENDE NUMMER: 23
VOLGENDE NUMMER: -4
VOLGENDE NUMMER: 19
```

```
INDEX          ITEM
1              -8
2              15
3              23
4              -4
5              19

19
-4
23
15
-8
```

K Programma 4-5 (BBC Regel 150: PRINT; D,L (D)

Om het probleem uit voorbeeld 3 op te lossen, hebben we heel wat moeite moeten doen. En waarvoor? Voor vijf getallen! Een matig resultaat, vindt u ook niet? Maar, als we de statements 20–50 die het aantal items tellen wat aanpassen, accepteert het programma opeens zoveel getallen als we maar willen (kunnen) invoeren.

Daartoe moeten we DIM afhankelijk maken van het aantal getallen. Omdat de teller C tot N+1 telt (zie regel 50) maken we DIM M$ (N+1).

```
10  REM ★★ INLEZEN N-LIJST ★★
20  INPUT "HOEVEEL ITEMS"; N ┐
25  DIM M$ (N + 1)           ┘   elk aantal items kan nu in
30  LET C = 1                     M$(C) worden ingelezen
40  INPUT "VOLGENDE ITEM"; M$(C)
50  LET C = C + 1
60  IF C < = N THEN 40
70  END
```

Programma 4-6.

De mogelijkheid om elk gewenst aantal items in een array te kunnen inlezen, is een goede stap voorwaarts. Maar we zijn natuurlijk nooit tevreden. Waarom zouden wij ons van te voren druk maken om na te tellen hoeveel items er in onze lijst staan, als de computer dat voor ons kan doen? In de volgende oefening wordt dit toegepast.

Oefening 1
(a) Schrijf een BASIC-programma dat een lijst van onbekende lengte kan inlezen. Beëindig de lijst met de dummy −9999. Noem de lijst P(C) en ga ervan uit dat deze niet langer wordt dan 30; DIM P(31) kan de lijst dan voor ons declareren. Controleer uw antwoord.
(b) Pas het programma zó aan dat alléén de oneven nummers van de lijst worden afgedrukt.

4.5 De FOR...NEXT-lus

We hebben al verschillende malen gesteld dat de computer erg goed is in grote aantallen herhaalde bewerkingen. Om in dergelijke gevallen niet de controle over het proces te verliezen, moet er geteld worden. Al in hoofdstuk 2 gebruikten we de telmethode om een programma te beëindigen, en we gebruikten daarvoor het IF...THEN-statement.

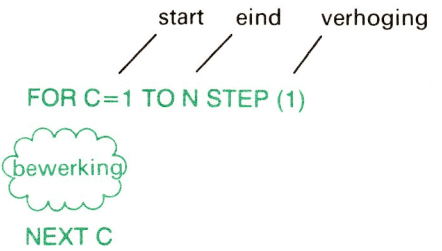
Programma 4-7.

Deze herhaalde bewerkingen zijn zó belangrijk dat er een speciale instructie voor gemaakt is. In BASIC is dat het FOR...NEXT-statement.

Programma 4-7 bestaat uit drie opeenvolgende elementen:

LET C = 1	definieert het beginpunt van de teller
LET C = C + 1	bepaalt de waarde waarmee de teller na elke bewerking wordt verhoogd (hier 1)
IF C < = N THEN	bepaalt de eindwaarde van de teller

Dezelfde drie constructies zijn opgenomen in de FOR...NEXT-instructie:



Programma 4-8.

Het NEXT C deel doet de programmateller terugspringen naar de instructie volgend op de FOR-instructie, zonder dat daarvoor het regelnummer opgegeven moet worden.

Arrays en FOR...NEXT-lussen behoren eigenlijk bij elkaar. Samen zijn ze de belangrijkste en krachtigste combinatie die BASIC ter beschikking heeft. Laten we eerst

eens in detail kijken hoe de FOR...NEXT-instructie gebruikt kan worden en deze instructie daarna in enkele vroegere programma's toepassen.

Voorbeelden van FOR...NEXT-lussen in de praktijk:

(a)
```
10  FOR I = 4 TO 10 STEP (2)
20  PRINT 1
30  NEXT I

RUN
4
6
8
10
```

(b)
```
10  FOR K = 11 TO 4 STEP (-2)
20  PRINT K
30  NEXT K

RUN
11
9
7
5
```

(c)
```
10  FOR J = -3 TO 10 STEP (3)
20  PRINT J
30  NEXT J

RUN
-3
0
3
6
9
```

(d)
```
10  FOR L = 4 TO -5 STEP (-2)
20  PRINT L
30  NEXT L

RUN
4
2
0
-2
-4
```

Programma's 4-9 t/m 4-12.

ZOP 2
Noteer de uitvoer van de volgende FOR...NEXT-lussen

(a)
```
10  FOR E = 1 TO 9 STEP (2)
20  PRINT E
30  NEXT E
```

(b)
```
10  FOR F = -30 TO -18 STEP (3)
20  PRINT F
30  NEXT F
```

(c)
```
10  FOR G = 8 TO -4 STEP (-5)
20  PRINT G
30  NEXT G
```

(d)
```
10  FOR H = -2 TO -11 STEP (-4)
20  PRINT H
30  NEXT H
```

Programma's 4-13 t/m 4-16.

FOR...NEXT...STEP(1)
Bovenstaande voorbeelden en ZOP's maakten gebruik van STEP-waarden van 2, 3, −2, −4 en −5. Vaak maken we echter alleen maar gebruik van een increment 1 (STEP 1). Als dat het geval is, kunt u STEP(1) gewoon weglaten.

Programma 4-17.

Dit programma wordt doorlopen met een STEP-waarde van 1.

Invoer/uitvoer-programma's met FOR...NEXT

De meeste programma's kunnen met behulp van de FOR...NEXT-instructie worden vereenvoudigd. We kunnen bijvoorbeeld programma 4-5 als volgt herschrijven: (Merk op dat in regel 20 en 150 de STEP (1) is weggelaten).

```
10  REM ★★ LEES 5 NAMEN IN EEN LIJST ★★
15  DIM L$(5)
20  FOR C = 1 TO 5                          ┐
30  INPUT "VOLGENDE NAAM : "; L$(C)         ├── Vervangt regel 20–50
40  NEXT C                                  ┘
50  REM
60  REM
100 REM ★★ AFDRUKKEN LIJST ★★
110 PRINT
120 PRINT
130 PRINT "INDEX", "NAAM"
140 PRINT
150 FOR D = 1 TO 5                          ┐
160 PRINT D, L$(D)                          ├── Vervangt regel 140–170
170 NEXT D                                  ┘
180 REM
190 REM
200 REM ★★ LIJST IN OMGEKEERDE VOLGORDE ★★
210 PRINT
220 PRINT
230 FOR E = 5 TO 1 STEP (-1)                ┐
240 PRINT E, L$(E)                          ├── Vervangt regel 230–260
250 NEXT E                                  ┘
260 END
```

Programma 4-18 Gebruik van FOR...NEXT om een lijst te keren.

```
RUN
VOLGENDE NAAM : CLAUS
VOLGENDE NAAM : WOLKERS
VOLGENDE NAAM : MULISCH
VOLGENDE NAAM : BORDEWIJK
VOLGENDE NAAM : VESTDIJK
```

```
INDEX          NAAM

1              CLAUS
2              WOLKERS
3              MULISCH
4              BORDEWIJK
5              VESTDIJK

5              VESTDIJK
4              BORDEWIJK
3              MULISCH
2              WOLKERS
1              CLAUS
```

[K] Programma 4-18.

De FOR...NEXT-lus kan in algemene vorm geschreven worden als:

```
FOR I = S TO F STEP (J)
   bewerking
NEXT I
```

indien S, F en J tenminste geschikte waarden hebben voordat de lus wordt begonnen. Ongeschikt zou bijvoorbeeld zijn:

FOR I = 2 TO 10 STEP −3

Omdat je 10 nooit zult bereiken wanneer je 3 van 2 aftrekt.

Oefening 2
In oefening 2 uit hoofdstuk 2 schreef u een programma voor het berekenen van de rente op uw rekeningtegoed. Herschrijf regel 40–90 van dit programma met behulp van een FOR...NEXT-lus.

Oefening 3
Als u wiskundig bent aangelegd, dan kunt u met het volgende probleem de kracht van het FOR...NEXT-statement uitproberen. Schrijf een programma om de inhoud van een kubus en de oppervlakte van één van de zijden ervan te berekenen voor alle kubussen met een oneven ribbelengte van 1 tot en met 21.

Schermuitvoer
Het is prettig een geschikte en esthetische lay-out op het scherm te hebben. De FOR-...NEXT-instructie kan daarvoor op grote schaal toegepast worden.

Lege regels
In hoofdstuk 3 bij het schrijven van brieven zag u het nut van lege regels met PRINT-statements. Met de volgende routine boekt u geheugenwinst bij meer dan 3 regels.

```
10  REM ★★ FOR...NEXT ★★
20  REM ★★ LEGE REGELS MET PRINT ★★
30  PRINT "HALLO"
40  FOR H = 1 TO 10
50  PRINT
60  NEXT H
70  PRINT "HALLO, HIER 11 REGELS LAGER !!"
80  END
```

instructies om 10 lege regels af te drukken

Programma 4-19.

```
RUN
HALLO



HALLO, HIER 11 REGELS LAGER !!
```

[K] Programma 4-19.

Lijnen tekenen

Het kan nuttig zijn een reeks symbolen af te drukken om een scheiding te bewerkstelligen tussen twee blokken afgedrukte gegevens of ter onderstreping van belangrijke gegevens. Het volgende programmadeel kan dit voor u doen:

```
10  REM ★★ FOR...NEXT ★★
20  REM ★★ LIJNEN TREKKEN ★★
30  FOR M = 1 TO 40
40  PRINT "★";
50  NEXT M
60  PRINT
70  END
```

instructies om 40 "★" af te drukken

Programma 4-20.

```
RUN
★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
```

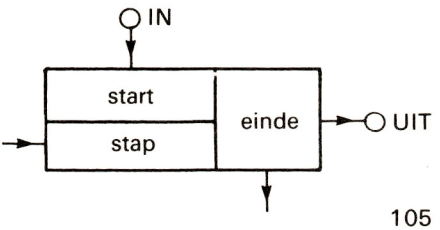
ZOP 3

Waarom heeft regel 40 een ';' aan het eind van het PRINT-statement? Wat gebeurt er als deze wordt weggelaten?

[K] Programma 4-20.

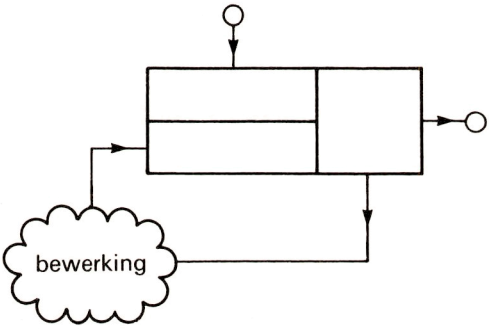
FOR...NEXT-lussen in stroomdiagrammen

Deze lussen zijn zó belangrijk dat er een speciaal symbool voor bestaat:



Afb. 4-4a. Stroomdiagramsymbool voor een FOR...NEXT-lus.

De loshangende einden zijn de verbindingen met de bewerking:



Afb. 4-4b. De relatie met het uit te voeren proces.

4.6 Nesten van lussen

Programma 4-20 drukte een regel met 40 sterretjes. Het is mogelijk het programma zó te herschrijven dat in de FOR...constructie een variabele het aantal sterretjes en dus de lengte van de regel aangeeft. Dan kunnen we in regel 20 het aantal specificeren:

```
10  REM ★★ LIJNEN VAN VERSCHILLENDE LENGTES ★★
20  LET N = 1 ┐──────────────────────aantal malen "★"
30  FOR M = 1 TO N
40  PRINT "★";
50  NEXT M
60  PRINT
70  END
```

Programma 4-21.

```
RUN
★
```

Om nu de regellengte te variëren tikken we in :

```
20  LET N = 'gevraagde aantal ★'
```

de gewenste lengte in.
Bijvoorbeeld :

```
20  LET N = 2

RUN
★★
```

```
20  LET N = 3

RUN
★★★

20  LET N = 4

RUN
★★★★

20  LET N = 8

RUN
★★★★★★★★

20  LET N = 16

RUN
★★★★★★★★★★★★★★★★

20  LET N = 32

RUN
★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
```

K Programma 4-21.

Nesten van FOR...NEXT-lussen

U heeft in programma 4-21 gezien hoe u de FOR...NEXT-lus kon beïnvloeden met een variabele. Nu u precies weet hoe dat in zijn werk gaat, hopen we dat bij u de volgende vraag naar boven komt: waarom laten we de waarde van N niet door een andere FOR...NEXT-lus beïnvloeden? En inderdaad, dat kan en programma 4-22 maakt daarvan gebruik. De M-lus in regel 30–50 wordt qua lengte bepaald door een N-lus in de regels 20–70. De M-lus is genest in de N-lus.

```
10  REM ★★ NESTED FOR...NEXT LUSSEN ★★
20  FOR N = 1 TO 16
30  FOR M = 1 TO N
40  PRINT "★";
50  NEXT M
60  PRINT
70  NEXT N
80  END
```

buitenste lus bepaalt aantal regels

binnenste lus bepaalt aantal ★

Programma 4-22 Geneste lussen.

```
RUN
★
★★
★★★
★★★★
★★★★★
★★★★★★
★★★★★★★
★★★★★★★★
★★★★★★★★★
★★★★★★★★★★
★★★★★★★★★★★
★★★★★★★★★★★★
★★★★★★★★★★★★★
★★★★★★★★★★★★★★
★★★★★★★★★★★★★★★
```

Het is essentieel dat u precies begrijpt hoe dat in zijn werk gaat. Bijvoorbeeld: het programma heeft zojuist 9 sterretjes afgedrukt en verlaat de M-lus in regel 50 en treedt de N-lus binnen in regel 60. Er wordt een lege regel geprint en N wordt één groter (10). Op regel 30 wordt de M-lus weer actief en worden er (nu) 10 sterretjes afgedrukt omdat de binnenste loop N (10) maal doorlopen wordt.

[K] Programma 4-22

Andere printpatronen met lussen

Als het printen van leuke patronen op het scherm u aanspreekt dan kunt u met het volgende programma en de beide oefeningen nog even vooruit.

```
10  REM ★★ GENESTE FOR...NEXT LUSSEN ★★
15  LET P = 1
20  FOR N = 1 TO 5
25  LET P = 2 ★ P ┐──────────── vermenigvuldigt P met 2 voor elke lus
30  FOR M = 1 TO P
40  PRINT "★";
50  NEXT M
60  PRINT
70  NEXT N
80  END
```

Programma 4-23.

```
RUN
★★                                  P =  2
★★★★                                P =  4
★★★★★★★★                            P =  8
★★★★★★★★★★★★★★★★                    P = 16
★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★    P = 32
```

[K] Programma 4-23

Oefening 4
Schrijf een programma om de tafels van vermenigvuldiging van 7, 8 en 9 uit te printen.

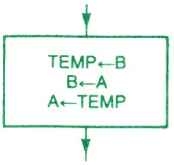
Oefening 5
Schrijf een programma met 'geneste'lussen om rechthoeken gevuld met sterretjes af te drukken. De maten daarvan moeten vrij te kiezen zijn.

4.7 Verwisselen

Ter voorbereiding van een verwisselprogramma, hebben we in hoofdstuk 2 en 3 al met de hand verschillende oefeningen gedaan. We vergeleken de elementen uit een lijst met het eerste element en verwisselden ze als element 1 groter was dan het volgende element uit de lijst. In ZOP 1 deden we dat met de hand en maakten we dus nog geen kennis met de problemen die ontstaan in de computer. In een programma mogen we niet zomaar stellen:

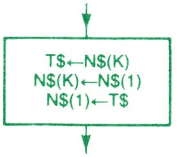
Kopieer A in B en B in A ,

want door het kopiëren van A wordt B overschreven en we hebben twee kopieën van A en géén van B. Nee, voor het verwisselen van A en B is een tussengeheugen vereist. We bergen de inhoud van B op in een tijdelijke locatie voordat we A in B kopiëren.
Vervolgens kopiëren we de tussenlocatie in A. In een diagram:



Afb. 4-5.

Veronderstel dat u een lijst met namen heeft (N$-lijst), en u wilt de eerste naam N$(1) verwisselen met naam N$(K) op nummer K.
Dit kan eenvoudig met behulp van een tijdelijke variabele T$:



Afb. 4-6.

Vergeet niet dat we de inhoud van N$(1) en N$(K) verwisselen. Als N$(1) = FRED en N$(K) = JIM, dan gebeurt er het volgende:

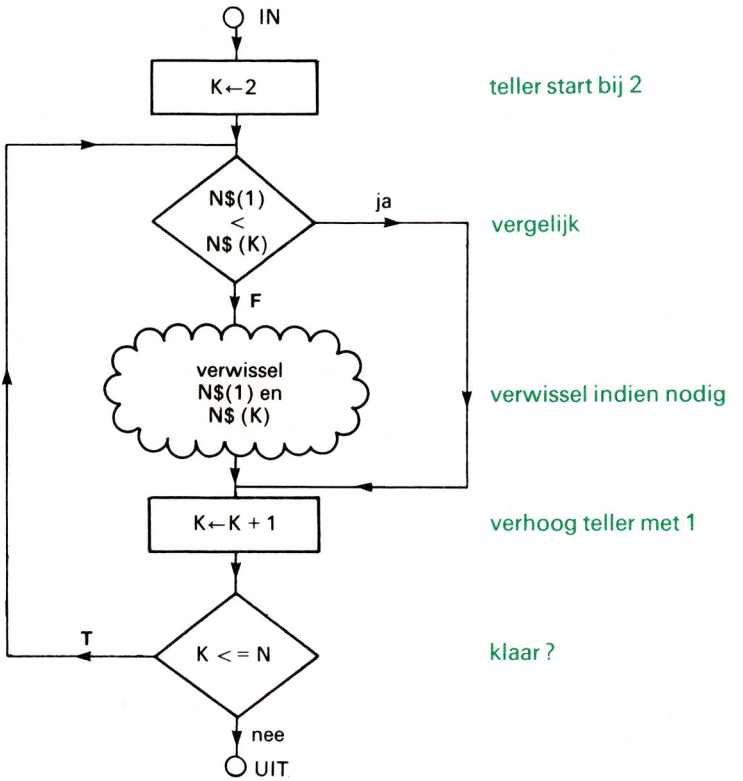
	Geheugenlocaties		
	N$(1)	N$(K)	T$
start	FRED	JIM	
1e stap	FRED	JIM	JIM
2e stap	FRED	FRED	JIM
einde	JIM	FRED	JIM

Afb. 4-7.

(Dat T$ na afloop nog steeds JIM bevat, doet in feite niet ter zake; de inhouden van N$(1) en N$(K) zijn verwisseld.)

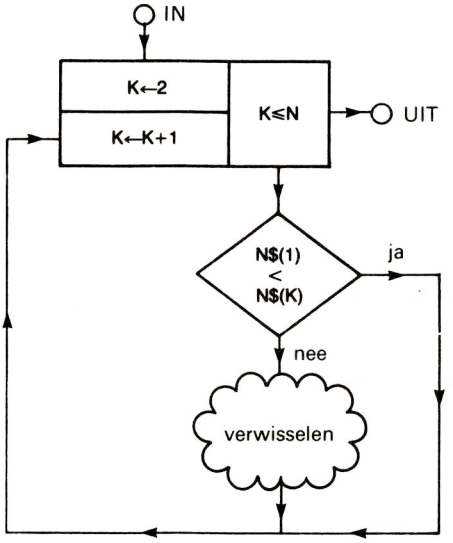
Stroomdiagram voor het sorteren van namen

In ZOP 1 maakten we kennis met het sorteren van getallen en we wilden het kleinste nummer aan het begin van de lijst hebben. Daarom verwisselen we nu op dezelfde manier elke naam met die aan het begin van de lijst, indien haar alfabetische waarde kleiner is. In stroomdiagram:



Afb. 4-8. Stroomdiagram voor verwisselen.

Of met gebruik van het speciale symbool voor de FOR...NEXT-lus zou het er zó uitzien:



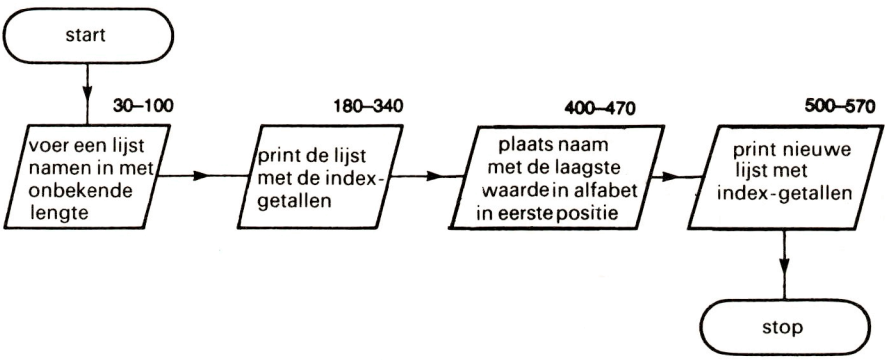
Afb. 4-9. Stroomdiagram voor verwisseling met FOR...NEXT-symbool.

Met dit programmadeel kunnen we nu een nieuw programma schrijven.

Voorbeeld 4

Schrijf een programma dat een lijst namen van onbekende lengte inleest in een array (lijst), laat deze uitprinten met index. Met behulp van de verwisselroutine plaatst u de naam met de laagste alfabetische waarde op de eerste plaats en print de lijst opnieuw uit.

Oplossing



Afb. 4-10. Stroomdiagram bij voorbeeld 4.

Verwisselprogramma

```
10  REM ★★ EERST OP ALFABETISCHE VOLGORDE ★★
15  CLEAR 100
20  DIM N$(30)
30  PRINT "GEEF NAMENLIJST EEN VOOR EEN"
40  PRINT "EINDIG MET ZZZZ"
50  PRINT
60  LET I = 1
70  INPUT "VOLGENDE NAAM: "; N$(I)
80  IF N$(I) = "ZZZZ" THEN 200
90  LET I = I + 1
100 GOTO 70
180 REM ★★★★★★★★★★★★
190 REM ★★ GEEN ZZZZ IN DE LIJST, DUS ★★
200 LET N = I - 1
210 REM ★★★★★★★★★★★★
300 PRINT
310 PRINT "INDEX", "ITEM"
320 FOR J = 1 TO N
330 PRINT J, N$(J)
340 NEXT J
400 REM ★★★★★★★★★★★★★★
410 REM ★★ VERWISSEL ROUTINE ★★
420 FOR K = 2 TO N
430 IF N$(1) < N$(K) THEN 470
440 LET T$ = N$(K)
450 LET N$(K) = N$(1)
460 LET N$(1) = T$
470 NEXT K
500 REM ★★★★★★★★★★★★★★★
510 PRINT
520 PRINT "★★ LIJST NA VERWISSELEN ★★"
530 PRINT
540 PRINT "INDEX", "ITEM"
550 FOR L = 1 TO N
560 PRINT L, N$(L)
570 NEXT L
580 END
```

invoeren lijst

aantal namen ingelezen = N

druk de lijst af

verwisselen

druk nieuwe lijst nogmaals af

Programma 4-24 Begin van een alfabetisch sorteerprogramma.

```
RUN
GEEF NAMENLIJST EEN VOOR EEN
EINDIG MET ZZZZ

VOLGENDE NAAM : JANSSEN
VOLGENDE NAAM : KLAASSEN
VOLGENDE NAAM : HENDRIKSEN
```

```
VOLGENDE NAAM : JONGSMA
VOLGENDE NAAM : ZZZZ

INDEX          ITEM
1              JANSSEN
2              KLAASSEN
3              HENDRIKSEN
4              JONGSMA

★★ LIJST NA VERWISSELEN ★★

INDEX          ITEM
1              HENDRIKSEN
2              KLAASSEN
3              JANSSEN
4              JONGSMA
```

Als we nu een printroutine inlassen tussen elke verwisselslag kunnen we het verloop goed volgen.

```
400 REM ★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
410 REM ★★ VERWISSEL ROUTINE ★★
420 FOR K = 2 TO N
430 IF N$(1) < N$(K) THEN 470
440 LET T$ = N$(K)
450 LET N$(K) = N$(1)
460 LET N$(1) = T$
470 PRINT
472 FOR L = 1 TO N
474 PRINT N$(L); " ";
476 NEXT L
478 PRINT
480 NEXT K
500 REM ★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
```

printroutine om het proces te illustreren

[K] Programma 4-24 met regels 400 tot 500 als hierboven.

Opgave 4

1. Hoogstwaarschijnlijk heeft u zich nu afgevraagd of u, nu het eerste element van de lijst gevonden is, het tweede op een vergelijkbare manier gevonden kan worden. Dat kan inderdaad. Door steeds één lager in de lijst te beginnen en vervolgens de rest van de lijst af te zoeken, verkrijgt u een volledige alfabetische lijst. Pas programma 4-24 zó aan dat dat voor u wordt gedaan. Daarvoor moet u geneste FOR...NEXT-lussen toepassen.

2. Schrijf een programma dat een lijst met namen en telefoonnummers gescheiden opbergt in twee lijsten N$(I) en T$(I). Gebruik de index I om de lijst na te zoeken op een gevraagde naam, en laat het telefoonnummer erbij afdrukken.

Samenvatting van hoofdstuk 4

Aan het einde van hoofdstuk 4 dient u de volgende onderwerpen in eenvoudige BASIC-programma's te kunnen toepassen.

Lijsten of array's
  invoeren van gegevens in een lijst.
  uitprinten van gegevens uit een lijst.

Tellers en hun gebruik bij lijsten

FOR...NEXT-lussen
  om een lijst in te lezen.
  om een lijst uit te printen.
  om ★ lay-outs te printen.

Geneste lussen
  om ★ lay-outs te printen.

Verwisselroutines
  eenvoudig sorteren.

Antwoorden op ZOP's en oefeningen

ZOP 1
In de vorm van een programma run:

```
RUN
VOER EEN LIJST IN
BEËINDIGEN MET ZZZZ

VOLGENDE NAAM : 6
VOLGENDE NAAM : 8
VOLGENDE NAAM : 4
VOLGENDE NAAM : 7
VOLGENDE NAAM : 3
VOLGENDE NAAM : 9
VOLGENDE NAAM : 1
VOLGENDE NAAM : ZZZZ

INDEX          ITEM
1              6
2              8
3              4
4              7
5              3
6              9
7              1
```

```
6  8  4  7  3  9  1
4  8  6  7  3  9  1
4  8  6  7  3  9  1
3  8  6  7  4  9  1
3  8  6  7  4  9  1
1  8  6  7  4  9  3
```

LIJST NA VERWISSELEN:

INDEX	ITEM
1	1
2	8
3	6
4	7
5	4
6	9
7	3

Oefening 1
We hebben veel REM-statements gebruikt om het één en ander uit te leggen.

```
5   DIM P(31)
10  REM ★★ NUMMERLIJST, ONBEPAALD LANG ★★
20  PRINT "VOER DE GETALLEN"
22  PRINT "EEN VOOR EEN IN."
24  PRINT "EINDIG MET '-9999' "
26  PRINT
30  LET C = 1                                          ┐
40  INPUT "VOLGENDE NUMMER : "; P(C)                   │
50  IF P(C) = -9999 THEN 100                           ├── invoervolgorde
60  LET C = C + 1                                      │
70  GOTO 40                                            ┘
80  REM ★★★★★★★★★★
90  REM ★★ VERGEET NIET DAT 'C' -9999 WEL NORMAAL GETELD
    HEEFT ★★
100 LET N = C - 1 ┐──────────────────────────────── correcte totaal
110 REM ★★★★★★★★★★
120 REM
130 REM
200 REM ★★ PRINT DE ONEVEN INDEXWAARDE MET LIJSTINHOUD ★★
210 REM ★★ 3 = 1 + 2..5 = 3 + 2..7 = 5 + 2.. ENZ..... ★★
220 LET C = 1                                          ┐
230 PRINT                                              │
240 PRINT                                              │
250 PRINT "ONEVEN INDEX", "ITEM"                       ├── uitvoervolgorde
260 PRINT C, P(C)                                      │
270 LET C = C + 2                                      │
280 IF C < = N THEN 260                                ┘
290 END
```

Programma 4-25.

```
RUN
VOER DE GETALLEN
EEN VOOR EEN IN
EINDIG MET '-9999'

VOLGENDE NUMMER : 42
VOLGENDE NUMMER : -12
VOLGENDE NUMMER : 37
VOLGENDE NUMMER : 92
VOLGENDE NUMMER : 11
VOLGENDE NUMMER : -3
VOLGENDE NUMMER : -9999

ONEVEN INDEX     ITEM
1                42
3                37
5                11
```

ZOP 2

(a) 1, 3, 5, 7, 9 (c) 8, 3, −2
(c) −30, −27, −24, −21, −18 (d) −2, −6, −10

Oefening 2

```
10  REM ★★ SAMENGESTELDE RENTE ★★
20  PRINT "VOER IN : JAREN, TEGOED EN % RENTE"
30  INPUT N, D, P
35  REM ★★★★★★★★★★
40  FOR C = 1 TO N
50  LET Y = (P ★ D)/100
60  PRINT "JAAR"; C, "OPBRENGST"; Y
70  LET D = D + Y
80  NEXT C
90  REM ★★★★★★★★★★
100 END
```

De FOR...NEXT-lus

Programma 4-26.

```
RUN
VOER IN: JAREN, TEGOED EN % RENTE
? 5, 500, 11.25
JAAR 1          OPBRENGST 56.25
JAAR 2          OPBRENGST 62.578125
JAAR 3          OPBRENGST 69.6181641
JAAR 4          OPBRENGST 77.4502075
JAAR 5          OPBRENGST 86.1633558
```

Oefening 3

```
10  REM ★★ KUBUS EN ZIJVLAK ★★
```

```
20  PRINT "NUMMERS", "ZIJVLAK", "INHOUD"
30  FOR I = 1 TO 21 STEP (2)
40  LET S = I★I
50  LET C = I★I★I
60  PRINT I, S, C
70  NEXT I
80  END
```

Programma 4-27.

```
RUN
NUMMER          ZIJVLAK         INHOUD
1               1               1
3               9               27
5               25              125
7               49              343
9               81              729
11              121             1331
13              169             2197
15              225             3375
17              289             4913
19              361             6859
21              441             9261
```

ZOP 3

';' aan het eind van een PRINT-statement onderdrukt de nieuwe regel na het printen van ★. Zonder dit teken zouden uw sterren elk op een aparte regel afgedrukt worden.

Oefening 4

```
20  REM ★★ TAFELS VAN VERMENIGVULDIGING ★★
30  FOR T = 7 TO 9
40  FOR K = 1 TO 12
50  LET P = K ★ T
60  PRINT K; " MAAL "; T; "="; P
70  NEXT K
80  PRINT
90  NEXT T
100 END
```

Programma 4-28.

```
RUN
 1 MAAL 7 = 7
 2 MAAL 7 = 14
 3 MAAL 7 = 21
 4 MAAL 7 = 28
 5 MAAL 7 = 35
 6 MAAL 7 = 42
 7 MAAL 7 = 49
 8 MAAL 7 = 56
```

9 MAAL 7 = 63
10 MAAL 7 = 70
11 MAAL 7 = 77
12 MAAL 7 = 84

1 MAAL 8 = 8
2 MAAL 8 = 16
3 MAAL 8 = 24
4 MAAL 8 = 32
5 MAAL 8 = 40
6 MAAL 8 = 48
7 MAAL 8 = 56
8 MAAL 8 = 64
9 MAAL 8 = 72
10 MAAL 8 = 80
11 MAAL 8 = 88
12 MAAL 8 = 96

1 MAAL 9 = 9
2 MAAL 9 = 18
3 MAAL 9 = 27
4 MAAL 9 = 36
5 MAAL 9 = 45
6 MAAL 9 = 54
7 MAAL 9 = 63
8 MAAL 9 = 72
9 MAAL 9 = 81
10 MAAL 9 = 90
11 MAAL 9 = 99
12 MAAL 9 = 108

Oefening 5

```
10  INPUT "LENGTE RECHTHOEK : "; L
20  INPUT "BREEDTE RECHTHOEK : "; W
30  FOR I = 1 TO W
40  FOR J = 1 TO L
50  PRINT "★";
60  NEXT J
70  PRINT
80  NEXT I
90  END
```

Programma 4-29.

5 Alles over strings en PRINT

5.1 Introductie

De vorige hoofdstukken werden voor een groot deel gevuld met de introductie van nieuwe begrippen en statements. Daar komt nu verandering in. Dit hoofdstuk gaat bijna geheel over strings. Toch introduceren we een nieuw machtig hulpmiddel voor het maken van lay-outs: het TAB-statement. De titel van dit hoofdstuk is misschien wat overdreven, maar aan het eind heeft u toch bijna alles met betrekking tot het printen "onder de knie".

5.2 De lengte van een string

In hoofdstuk 3 stelden we de vraag 'Hoe lang is een string?'. Op dat moment leek dat een onbelangrijke vraag, maar het aantal karakters dat een bepaalde string bevat is vaak een belangrijk gegeven. Vooral wanneer we het geheugen van onze computer zo efficiënt mogelijk willen benutten.

In BASIC kunnen we met de functie LEN (A$), de lengte van A$ in aantal karakters te weten komen.

Als A$ = "FRED" dan is LEN (A$) = 4 (numeriek!)
Als A$ = "I" dan is LEN (A$) = 1

ZOP 1
Geef de waarde van de volgende expressies aan:

(a) LEN (C$) als C$ = "ANN"
(b) LEN (D$) als D$ = "A"
(c) LEN (E$) als E$ = "72"
(d) LEN (F$) als F$ = "KAT 123"

Voorbeeld 1
Schrijf een BASIC-programma dat van een lijst woorden van elk woord de lengte afdrukt. Het laatste woord is dummy: ZZZZ.

Oplossing
In het onderstaande programma komen de woorden uit een READ-statement in plaats van een INPUT-statement om de invoertijd wat te verminderen. Elk woord uit DATA wordt in W$ (regel 100) opgeslagen en haar lengte in L (regel 120). Het resultaat wordt vervolgens op regel 140 afgedrukt.

```
10  REM ★★ WOORDLENGTE ★★
20  REM ★★★★★★★★★★
30  REM ★★ LEES WOORDEN EEN VOOR EEN UIT EEN DATA LIJST ★★
40  REM ★★ EN PRINT DE WOORDLENGTE ★★
90  REM ★★★★★★★★★★
100 READ W$
110 IF W$ = "ZZZZ" THEN 200
120 LET L = LEN (W$)
130 PRINT
140 PRINT W$; " HEEFT "; L; " LETTERS"
150 GOTO 100
200 END
900 DATA ONTWERP,EEN,ALGORITHME,EN,SCHRIJF,EEN,PROGRAMMA
910 DATA ZZZZ
```

READ-, LEN-, PRINT-cyclus

Programma 5-1 Woordlengte meten.

```
RUN
ONTWERP HEEFT 7 LETTERS

EEN HEEFT 3 LETTERS

ALGORITHME HEEFT 10 LETTERS

EN HEEFT 2 LETTERS

SCHRIJF HEEFT 7 LETTERS

EEN HEEFT 3 LETTERS

PROGRAMMA HEEFT 9 LETTERS
```

[K] Programma 5-1.

5.3 Frequentietabellen

Vaak is het nuttig te bepalen hoe vaak een bepaalde gebeurtenis in een tijdvak voorkomt. De frequentie van bepaalde letters in een codebericht is vaak een belangrijk gegeven voor het vinden van de sleutel. Voor het meten van frequenties gebruiken we eenvoudige technieken uit de statistiek waaronder het turven. Ons eerste pen-en-papier voorbeeld demonstreert dit.

Turven

Voorbeeld 2
Bepaal de frequentie waarmee de klinkers in de volgende DATA-statements voorkomen.

```
900  DATA ALS, EEN, LOSGELATEN, PIJL
```

```
910 DATA ERGENS, IN, EEN, GROTE, STAD, BELAND
920 DATA TUSSEN, AUTO'S, EN, FABRIEKEN
930 DATA TUSSEN, SNELWEGEN, EN FLATGEBOUWEN, ZZZZ
```

Oplossing:
Er zijn twee methoden mogelijk:
(a) Streep aan en turf eerst alle A's, vervolgens alle E's enz. Daartoe moet u vijfmaal door het hele gedicht heen voor relatief weinig resultaat (lage score-rate).
(b) Maak een tabel als onderstaand en neem hierin achtereenvolgens alle klinkers op, merk ze en turf ze in de tabel.

ÆLS: een turf in de A-kolom
ÆÆN: twee turven in de E-kolom
LØSGÆSCHØTÆN: twee turven in de O-kolom; twee in de E-kolom enz.

klinker	telling					frequentie
A	𝍸	11				7
E	𝍸	𝍸	𝍸	𝍸	1	21
I	111					3
O	1111					4
U	1111					4

```
900 DATA ÆLS ÆÆN LØSGÆLÆTÆN PIJL
910 DATA ÆRGÆNS IN ÆÆN GRØTÆ STÆD BÆLÆND
920 DATA TØSSÆN ÆØTØ'S ÆN FÆBRIÆKÆN
930 DATA TØSSÆN SNÆLWÆGÆN ÆN FLÆTGÆBØØWÆN
```

Afb. 5-1. Klinkeranalyse van een deel van een gedicht (auteur J. S. With).

ZOP 2

Maak een frequentietabel volgens de turfmethode van de woordlengten uit het gedicht in voorbeeld 2.

Laat de computer tellen

Het is wel aardig om met de hand te turven, maar laten we nu eens kijken of BASIC dat niet voor ons kan oplossen. In opgave 4 zagen we hoe twee verschillende lijsten met elkaar verbonden kunnen worden met behulp van een index; namelijk naam en telefoonnummerlijst. Het derde element van de namenlijst behoort bij het derde element uit de nummerlijst. Algemeen is het zo dat het I-de element uit de namenlijst behoort bij het I-de element uit de nummerlijst.

Veronderstel eens dat we het aantal malen, dat de cijfers 1 tot 9 in een reeks getallen voorkomen willen tellen. Daartoe kunnen we tien tellers definiëren C(0)...C(9) die

alle met nul beginnen. Om de cijfers in 473808 te tellen beschouwen we het eerste cijfer: de 4. Daartoe verhogen we C(4) met één en vervolgens beschouwen we het volgende cijfer enz.

Getallen ingevoerd	Stand van tellers									
	C(0)	C(1)	C(2)	C(3)	C(4)	C(5)	C(6)	C(7)	C(8)	C(9)
start	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	1	0	0	1	0	0
3	0	0	0	1	1	0	0	1	0	0
8	0	0	0	1	1	0	0	1	1	0
0	1	0	0	1	1	0	0	1	1	0
8	1	0	0	1	1	0	0	1	2	0

Afb. 5-2.

Het principe is dat als een cijfer I wordt gelezen, dat teller C(I) wordt opgehoogd. Dat kan worden bereikt met slechts twee BASIC-statements.

```
120  INPUT I
140  LET C(I) = C(I) + 1
```

teller in lijstvorm

ZOP 3
De statements:

```
10  INPUT N
20  LET C(N) = C(N) + 1
```

kunnen worden gebruikt om het aantal enen, tweeën enz. te tellen in een reeks ingevoerde cijfers:
3,1,0,5,9,9,6,6,6,0,4,4,2,4,1,2,1,3,0,2,1,3. Geef de waarde van de volgende tellers aan:

(a) C(3) nadat 3 cijfers zijn ingelezen
(b) C(9) nadat 12 cijfers zijn ingelezen
(c) C(1) nadat alle cijfers zijn ingelezen
(d) C(0) nadat alle cijfers zijn ingelezen.

Deze methode passen we nu toe om een lijst uit te tellen.

Voorbeeld 3
Schrijf een programma dat een reeks enkele cijfers inleest en dat de frequentie van elk cijfer afdrukt.

Oplossing:
Een cijfer is een van de getallen 0, 1, 2, 3,....,9. We zullen deze één voor één invoeren en de reeks beëindigen met −9999. Tot nu toe allemaal erg eenvoudig. De tellerlijst bevat 10 tellers:

C(0), C(1), C(2),...,C(9)

Het programma kan worden ingedeeld in twee delen: (a) een deel dat de cijfers één voor één inleest en de tellers ophoogt en (b) een afdrukroutine die door een FOR-...NEXT-lus wordt uitgevoerd. Het deel (a) wordt uitgevoerd door de statements 120 en 140. De FOR...NEXT-lus uit (b) heeft als index J, die gaat van 0 tot en met 9.

```
10  REM ★★ TEL HET AANTAL MALEN DAT IEDER CIJFER ★★
20  REM ★★ VOORKOMT EN TEL IN EEN TELLERLIJST C(I) ★★
30  REM ★★★★★★★★★★
40  DIM C(10)
100 PRINT "VOER EEN RIJ INDIVIDUELE CIJFERS IN"
110 PRINT "EINDIG MET -9999"
120 INPUT "VOLGENDE CIJFER"; I
130 IF I = -9999 THEN 200
140 LET C(I) = C(I) + 1
150 GOTO 120
190 REM ★★★★★★★★★★
200 PRINT
210 PRINT "CIJFER", "AANTAL"
220 PRINT
230 FOR J = 0 TO 9
240 PRINT J, C(J)
250 NEXT J
260 END
```

input en telroutine

printen tabel

Programma 5-2 Tellen met een tellerlijst (CI).

Nadat de volgende reeks getallen is ingevoerd ontstaat de frequentietabel als volgt:
(3,7,6,4,9,1,4,9,2,7,8,0,1,5,2,7,-9999)

```
VOER EEN RIJ INDIVIDUELE CIJFERS IN
EINDIG MET -9999
VOLGENDE CIJFER 3
VOLGENDE CIJFER 7
VOLGENDE CIJFER 6
VOLGENDE CIJFER 4
VOLGENDE CIJFER 9
VOLGENDE CIJFER 1
VOLGENDE CIJFER 4
VOLGENDE CIJFER 9
VOLGENDE CIJFER 2
VOLGENDE CIJFER 7
VOLGENDE CIJFER 8
VOLGENDE CIJFER 0
VOLGENDE CIJFER 1
VOLGENDE CIJFER 5
VOLGENDE CIJFER 2
VOLGENDE CIJFER 7
VOLGENDE CIJFER -9999
```

CIJFER	AANTAL
0	1
1	2
2	2
3	1
4	2
5	1
6	1
7	3
8	1
9	2

K Programma 5-2. BBC regel 240: PRINT; J,; C(J)

Frequentietabellen voor stringlengte
In dit hoofdstuk hebben we tot nu toe twee programma's geschreven, één voor het vaststellen van stringlengtes en één voor het maken van frequentietabellen.
In de volgende oefening willen we dat u beide programma's combineert om een programma te maken dat een frequentietabel samenstelt van de lengtes van de ingevoerde woorden. Eventueel kunt u de woorden uit voorbeeld 2 als invoer nemen. Veronderstel dat geen woord langer is dan 15 karakters, dan heeft de lengte-teller-lijst de volgende elementen:

L(1), L(2), L(3),...,L(15).

Oefening 1
Schrijf een programma dat een frequentietabel maakt van de lengtes van een willekeurig aantal woorden.

5.4 Frequentiediagrammen

Frequentiediagram voor het aantal klinkers
Het diagram met turven uit fig. 5-1 geeft een betere indruk van de verdeling van klinkers in een gedicht dan de kolom cijfers uit het bovenstaande voorbeeld. Dus waarom zou de computer dan niet een figuur voor ons maken? U heeft al kennis gemaakt met een programma voor het printen van sterretjes waarbij gebruik werd gemaakt van een FOR...NEXT-lus.

ZOP 4
Wat verschijnt er op het scherm met het volgende programma?

```
10  READ A
20  FOR I = 1 TO A
30  PRINT "★";
40  NEXT I
45  PRINT
50  GOTO 10
100 DATA 2, 5, 7, 8, 3, 1
```

Programma 5-3.

We kunnen hetzelfde doen met de frequenties uit afb. 5-1 om een diagram te maken.

Voorbeeld 4
Schrijf een programma om een diagram te printen van de verdeling van de klinkers uit voorbeeld 2.

Oplossing:
Dit programma maakt een diagram van frequenties die we al eerder hebben bepaald. Zij zijn opgenomen in het DATA-statement in regel 900. Met een teller F[K] worden de frequenties ingelezen waarin F(1) het aantal A's vertegenwoordigt, F(2) het aantal E's enz. (regels 50 tot 80).

Vervolgens printen we sterretjes overeenkomend met de waarden van F(K) (regels 220 tot 250).

```
10  REM ★★ FREQUENTIEVERDELING ★★
20  REM ★★ VOORBEELD DIAGRAM ★★
30  REM ★★ FREQUENTIELIJST IS F(K) ★★
35  DIM F(6)
40  LET K = 1
50  READ F(K)
60  IF F(K) = -9999 THEN 110
70  LET K = K + 1
80  GOTO 50
90  REM ★★★★★★★★★★
100 REM ★★ -9999 NIET IN DE LIJST PLAAT-
    SEN ★★
110 LET N = K - 1
120 REM ★★★★★★★★★★
200 REM ★★ PRINT ROUTINE ★★
210 PRINT
220 FOR X = 1 TO N
230 FOR Y = 1 TO F(X)
240 PRINT "★";
250 NEXT Y
260 PRINT
270 PRINT
280 NEXT X
300 REM ★★★★★★★★★★
310 END
900 DATA 9, 16, 10, 10, 6, -9999
```

lees de frequentie en sla ze op in F(1), F(2) enz.

print ★★★ op de regel

Programma 5-4 Frequentieverdeling in diagramvorm.

```
RUN

★★★★★★★★★

★★★★★★★★★★★★★★★

★★★★★★★★★★
```

★★★★★★★★★★

★★★★★★

K Programma 5-4.

Frequentiediagram voor woordlengten

Programma 5-4 is niet geschikt om de woordlengten uit ZOP 1 te bewerken. Daarvoor moeten twee aanpassingen geschieden. Ten eerste bevat de frequentielijst meer mogelijkheden; behalve dat de woordlengte varieert van 1 tot 15 is er ook nog de dummywaarde −9999. Daartoe vergroten we F tot 16 tellers met

```
35  DIM F(16)
```

Ten tweede ontstaan er problemen in de printroutine als een bepaalde frequentie nul blijkt te zijn. De FOR...NEXT-lus kan immers niet van 1 tot 0 gebruikt worden. Daartoe moeten we voorkomen dat de FOR...NEXT-lus wordt binnengegaan indien F(I) gelijk is aan nul. Een IF...THEN-statement is daarvoor zeer geschikt:

```
225  IF F(X) = 0 THEN 260
```

en regel 900 wordt nu:

```
900  DATA 1, 5, 4, 6, 9, 0, 1, 3, 1, 0, 0, 1, 0, 0, 0, −9999
```

Een run van het aangepaste programma ziet er dan als volgt uit:

```
RUN
★

★★★★★

★★★★

★★★★★★

★★★★★★★★★

★

★★★

★


★
```

Afb. 5-3. Frequentiediagram van aangepast programma 5-4.

[K] Programma 5-4 (aangepast).

5.5 De tabulator

De belangrijkste ingrediënten voor het maken van diagrammen kennen we nu, maar we zijn nog ver verwijderd van een leesbaar en bruikbaar diagram. Een handig hulpmiddel is de tabulator, in BASIC vertegenwoordigd door de functie TAB(X). TAB(X) maakt dat de cursor of printkop zich naar een gedefinieerde positie verder op de regel beweegt. Ter illustratie gebruiken we dezelfde techniek als in hoofdstuk 3, nl. een stukje demonstratieprogramma.

Kijk wat er gebeurt als we de posities op het scherm nummeren en als we printen met de TAB-functie.

```
50  PRINT "12345678901234567890123456789012345678901234567890"
60  PRINT "A"; TAB(5); "E"; TAB(7); "I"; TAB(19); "O"; TAB(31); "U"
```

Programma 5-5.

```
RUN
12345678901234567890123456789012345678901234567890
A    E I                 O                   U
```

U ziet dat TAB(5) ervoor zorgde dat de E op de zesde positie terecht kwam. Waarom? Omdat de tabulatorfunctie intern van nul af aan begint te tellen. Programma 5-6 heeft een nummerschaal vanaf 0:

```
50  PRINT "01234567890123456789012345678901234567890123456789"
60  PRINT "A"; TAB(5); "E"; TAB(7); "I"; TAB(19); "O"; TAB(31); "U"
```

Programma 5-6.

```
RUN
01234567890123456789012345678901234567890123456789
A    E I                 O                   U
```

Nu print TAB(5) de E op positie vijf, maar dat is nog steeds de zesde printpositie.

ZOP 5
Schrijf een programma om 'COL1', 'COL2' en 'COL3' op het scherm te printen op respectievelijk de posities 0, 10 en 20.

Variabele TAB's
Als we regel 60 uit bovenstaand voorbeeld in een FOR...NEXT-lus opnemen, kunnen we een echte tabel printen:

```
50  FOR I = 1 TO 7
60  PRINT "A"; TAB(5); "E"; TAB(7); "I"; TAB(19); "O"; TAB(31); "U"
70  NEXT I
80  END
```

Programma 5-7.

```
RUN
A     E  I              O                U
A     E  I              O                U
A     E  I              O                U
A     E  I              O                U
A     E  I              O                U
A     E  I              O                U
A     E  I              O                U
```

Het volgende voorbeeld geeft een heel ander effect. We nemen nu een variabele op in de TAB-functie. Het gebruik van TAB(V) met V als variabele maakt dat we de cursor overal op de regel kunnen plaatsen.

```
30  FOR A = 1 TO 10
40  PRINT TAB(A); "HALLO"          de waarde van A in TAB(A) is
50  NEXT A                         gelijk aan de lusvariabele A
60  END
```

Programma 5-8.

```
RUN
HALLO
  HALLO
    HALLO
      HALLO
        HALLO
          HALLO
            HALLO
              HALLO
                HALLO
                  HALLO
```

Een stap verder kunnen we beide effecten in één programma onderbrengen:

```
50  FOR I = 1 TO 7
60  PRINT TAB(0 + I); "A"; TAB(5 + I); "E"; TAB(7 + I); "I";
65  PRINT TAB(19 + I); "O"; TAB(31 + I); "U"
70  NEXT I
80  END
```

Programma 5-9.

```
RUN
A         E  I          O          U
  A         E  I          O          U
    A         E  I          O          U
      A         E  I          O          U
        A         E  I          O          U
          A         E  I          O          U
            A         E  I          O          U
```

ZOP 6
Schrijf een programma dat drie willekeurige getallen inleest, waardoor het woord TABULATOR op de drie hiermee corresponderende posities op een regel wordt geplaatst.

Het gebruik van TAB in frequentiediagrammen
We weten nu voldoende van de tabulator om het frequentiediagram uit fig. 5-2 in een betere lay-out te laten printen. De laatste versie van de printroutine uit programma 5-4 (regel 200 tot 300):

```
200  REM ★★ PRINT ROUTINE ★★★★★★★★
210  PRINT
220  FOR X = 1 TO N
225  IF F(X) = 0 THEN 260
230  FOR Y = 1 TO F(X)
240  PRINT "★";
250  NEXT Y
260  PRINT
270  PRINT
280  NEXT X
300  REM ★★★
```

Programma 5-4 (aangepast)

We voegen toe:

regel 212 om een tekst boven de kolommen te printen
regel 214 om een lijn onder deze tekst te printen
regel 216 om de kolomverdeling aan te geven
regel 222 (in de lus) om zowel de kolomverdeling als de waarden X en F(X) aan te geven. De regel eindigt op ";" zodat het volgende PRINT-statement op dezelfde regel verder gaat.

Bestudeer het programma zorgvuldig zodat u in detail alles begrijpt. Voor de regels 10 tot 120 : zie programma 5-4.

```
200  REM ★★ PRINT ROUTINE ★★★★★★★★
210  PRINT
212  PRINT "LENGTE"; TAB(8); "FREQ"; TAB(18); "TURVEN"
214  PRINT "————————————————————————————————————"
216  PRINT TAB(7); "I"; TAB(12); "I"
220  FOR X = 1 TO N
222  PRINT TAB(2); X; TAB(7); "I"; TAB(9); F(X); TAB(12); "I"; TAB(18);
225  IF F(X) = 0 THEN 260
230  FOR Y = 1 TO F(X)
240  PRINT "★";
250  NEXT Y
260  PRINT
280  NEXT X
300  REM ★★★★★★★★★★★★★★★★★★
```

```
900 DATA 1, 5, 4, 6, 9, 0, 1, 3, 1, 0, 0, 1, 0, 0, 0,-9999
```

Programma 5-10.

```
RUN
LENGTE      FREQ        TURVEN ──────────────────────── 212
──────────────────────────────────────────────────────── 214
            I           I ──────────────────────────── 216
1           I 1         I ★
2           I 5         I ★★★★★
3           I 4         I ★★★★
4           I 6         I ★★★★★★
5           I 9         I ★★★★★★★★★
6           I 0         I
7           I 1         I ★
8           I 3         I ★★★
9           I 1         I ★
10          I 0         I
11          I 0         I
12          I 1         I ★
13          I 0         I      ── als 230-250 maar met TAB(14)
14          I 0         I         als uitgangspositie
15          I 0         I
└──────────────────────┘ └──── resultaat van regel 222
```

[K] Programma 5-10

Oefening 2
Pas programma 5-4 zo aan dat u een overeenkomstige printout verkrijgt als in programma 10 en voeg er de volgende dingen aan toe:

- een toepasselijke te veranderen koptitel;
- de toevoeging van de desbetreffende klinker links op de regel;
- een geschikte schaalverdeling als basis voor de tabel.

5.6 Manipuleren met strings

Strings kunnen meer dan een woord of item bevatten, terwijl deze toch als een eenheid wordt behandeld. 23-juli-1983 is bijvoorbeeld zo'n eenheid waar we soms een deel uit willen lichten, bijvoorbeeld de maand.

De datum
Hoe vaak bent u niet geconfronteerd met een dergelijke rij hokjes op een formulier?

DATUM						
	D	D	M	M	J	J

Afb. 5-4.

De DDMMJJ configuratie geeft nogal wat problemen. Vergelijk eens:

23 juni 1971 of 230671 en

14 september 1973 of 140973

met elkaar. De laatste datum geeft het kleinste getal terwijl het met 4 juli 1933 en 15 januari 1967 net andersom is. Het is duidelijk dat de DDMMJJ-methode niet geschikt is om data met elkaar te vergelijken. Een oplossing is het verwisselen van de volgorde van DDMMJJ in JJMMDD. Met de bovenstaande data levert dat de getallen

330704, 670115, 710623 en 730914

Nu zijn zowel de getallen als de data die zij voorstellen in oplopende volgorde gerangschikt. De datum wordt in de computer meestal als een numeriek gegeven behandeld, maar wordt als string ingelezen om eventuele foutcorrectie-procedures op de ingevoerde datum te laten uitvoeren.

Als wij geïnteresseerd zijn in salarisberekeningen dan zijn vooral de maand en jaarcijfers van belang. Een pianostemmer die zijn klanten elke drie maanden bezoekt zal vooral belang hebben bij de maandcijfers om zijn klanten op tijd voor een bezoek te informeren. Hoewel een gehele string op zich belangrijk kan zijn, kunnen er talloze redenen zijn om er een deel uit te lichten.

LEFT$(X$,I) en RIGHT$(X$,I)
Als we een gedeelte uit een string willen lezen hebben we een functie nodig die dat voor ons doet. We beginnen met twee van deze functies.

LEFT$(X$,I) geeft ons de I meest linkse karakters uit X$.

bijvoorbeeld als X$ = "WINDHOOS"
dan LEFT$(X$,4) = "WIND"

RIGHT$(X$,I) geeft daarentegen de I meest rechtse karakters uit X$.

bijvoorbeeld RIGHT$(X$,4) = "HOOS"

Uw computer licht een en ander nog eens voor u toe. Het volgende programma zal uit een string van 6 karakters met behulp van de index I uit de FOR...NEXT-lus een steeds langer deel uit de originele string halen. De schaalverdeling helpt u met tellen.

```
10  REM ★★ STRING TEST ★★
20  INPUT "VOER EEN 6-KARAKTER STRING IN"; X$
30  PRINT TAB(10); "1234567890"
40  FOR I = 1 TO 6
50  LET A$ = LEFT$(X$,I)
60  PRINT I; TAB(10); A$
70  NEXT I
```

Programma 5-11.

```
RUN
VOER EEN 6-KARAKTER STRING IN 123456
         1234567890
1        1
2        12
3        123
4        1234
5        12345
6        123456
```

Als we nu regel 50 van programma 5-11 aanpassen met:

```
50  LET A$ = RIGHT$(X$,I)
```

en de invoer is ABCDEF dan krijgen we:

```
RUN
VOER EEN 6-KARAKTER STRING IN ABCDEF
         1 2 3 4 5 6 7 8 9 0
1        F
2        E F
3        D E F
4        C D E F
5        B C D E F
6        A B C D E F
```

[K] Programma 5-11 met LEFT$ EN RIGHT$.
BBC regel 60: PRINT; I; TAB(10); A$

ZOP 7
Als A$ = "1A2B3C4D" wat zijn dan de volgende strings?

(a) LEFT$(A$,1)
(b) LEFT$(A$,4)
(c) RIGHT$(A$,3)
(d) RIGHT$(A$,4)

Andere strings 'verknippen'.
Programma 5-11 is een beetje onhandig, omdat we speciaal een string van 6 letters moeten invoeren. Daar is echter geen enkele reden voor; met langere strings werkt het programma ook. Daartoe behoeven we alleen maar de bovengrens van het FOR-...NEXT-statement aan te passen. Deze wordt gelijk gemaakt aan de stringlengte.

```
20  INPUT "VOER EEN STRING IN"; X$
40  FOR I = 1 TO LEN(X$)      ]---- LEN(X$) is de bovengrens van
50  LET A$ = LEFT$(X$, I)     ]     de lus
60  PRINT I; TAB(10); A$
70  NEXT I
80  END
```

Programma 5-12.

```
RUN
VOER EEN STRING IN VLINDER
1          V
2          VL
3          VLI
4          VLIN
5          VLIND
6          VLINDE
7          VLINDER
```

K Programma 5-12. BBC regel 60: PRINT; I; TAB(10); A$

Oefening 3
Schrijf een programma dat alleen de woorden uit oefening 1 uitprint die met een klinker beginnen.

Oefening 4
Schrijf een programma dat de uitvoer van programma 5-11b aanpast tot:

```
         F
        EF
       DEF
      CDEF
     BCDEF
    ABCDEF
```

MID$(X$,I,J)
We hebben LEFT$ en RIGHT$ gebruikt om gedeelten uit een string te 'knippen'. Het komt ook wel voor dat we een gedeelte uit het midden van een string willen halen. In plaats van moeilijk gecombineerde LEFT$ en RIGHT$ statements kunnen we rechtstreeks met MID$ elke gewenste string uit een andere halen. (bijvoorbeeld MM uit de datumstring JJMMDD). Het BASIC-statement is van de volgende vorm:
MID$(X$,I,J)

Het resultaat is een substring met een lengte J, te beginnen vanaf het I-de karakter.

```
MID$(X$,I,J)
        |└──── lengte van substring vanaf positie I
        └───── beginpositie in de originele string.
```

Als X$ = "POSITIE" dan

MID$(X$,5,2) = "TI"
MID$(X$,2,4) = "OSIT"

We laten de computer het demonstratievoorbeeld zelf afmaken met behulp van een aanpassing in programma 5-11.

```
10  REM ★★ STRING TEST ★★
20  INPUT "VOER EEN STRING IN"; X$
30  PRINT TAB(10); "1234567890"
40  FOR I = 1 TO LEN(X$)
50  LET A$ = LEFT$(X$,I)
60  PRINT I; TAB(10); A$
70  NEXT I
80  END
```

Programma 5-13.

Als we nu in regel 50 LEFT$ vervangen door MID$:

```
50  LET A$ = MID$(X$,I,1)
```

en we lezen de string "WINTERLANDSCHAP" in dan krijgen we:

```
RUN
VOER EEN STRING IN WINTERLANDSCHAP
           1 2 3 4 5 6 7 8 9 0
 1         W
 2          I
 3          N
 4          T
 5          E
 6          R
 7          L
 8          A
 9          N
10          D
11          S
12          C
13          H
14          A
15          P
```

In dit programma zoek MID$ naar alle mogelijke substrings van lengte 1.
Gebruiken we nu in regel 50:

```
50  LET A$ = MID$(X$,I,2)
```

met "LENTEDAUW" als invoerstring dan krijgen we:

```
RUN
VOER EEN STRING IN LENTEDAUW
           1 2 3 4 5 6 7 8 9 0
1          L E
2          E N
3          N T
```

```
4          T E
5          E D
6          D A
7          A U
8          U W
9          W _____ nul-string
```

[K] Programma 5-13. BBC regel 60: PRINT; I, TAB(10); A$

De laatste substring zorgde voor enige problemen. We kunnen immers geen 2 karakters die beginnen met het 9e karakter uit een string van 9 karakters halen. Daarom treedt hier een default-mechanisme in werking dat de substring afkapt tot 1 karakter. Er moeten altijd voldoende karakters in een string aanwezig zijn om alle substrings te kunnen lezen.

Als we J karakters uit een string willen halen mogen we niet beginnen voor de (LEN(X$) − J + 1)-de positie.

ZOP 8

Schrijf een programma dat van de Amsterdamse telefoonnummers (020-XXXXXXX) alleen het abonneenummer afdrukt.

MID$-string programma

U heeft waarschijnlijk al gezien dat MID$ ook linker- en rechter substrings uit een string kan halen. Het volgende programma demonstreert dat door uit een invoerstring alle mogelijke substrings in tabelvorm af te drukken.

```
10  REM ★★ STRING TEST ★★
20  INPUT "VOER EEN STRING IN"; X$
30  PRINT "J"; TAB(5); "I"; TAB(10); "1234567890"
35  FOR J = 1 TO LEN(X$)
40  FOR I = 1 TO (LEN(X$) - J + 1)
50  LET A$ = MID$(X$,I,J)
60  PRINT J; TAB(5); I; TAB(10); A$
70  NEXT I
72  PRINT " "
75  NEXT J
80  END
```

J is de lengte van de substring vanaf positie I

Programma 5-14.

```
RUN
VOER EEN STRING IN STRING
J     I      1 2 3 4 5 6 7 8 9 0
1     1      S
1     2      T
1     3      R
1     4      I
1     5      N
1     6      G
```

```
2     1     ST
2     2     TR
2     3     RI
2     4     IN
2     5     NG

3     1     STR
3     2     TRI
3     3     RIN
3     4     ING

4     1     STRI
4     2     TRIN
4     3     RING

5     1     STRIN
5     2     TRING

6     1     STRING
```

K Programma 5-14. BBC regel 60: PRINT;...enz.

5.7 De functie VAL

Nu we weten hoe we strings in stukken moeten knippen hebben we ook nog een methode nodig om te bekijken wat we gekregen hebben.
Een van deze methoden is de VAL(A$) functie. VAL(A$) bepaalt de numerieke waarde van de string.

VAL(A$) geeft de numerieke waarde van de string A$ indien de string A$ begint met een +, − of een digit. In alle andere gevallen is VAL(A$) gelijk aan nul (0).

Programma om VAL te demonstreren
In het volgende programma voeren we zeven strings in en bepalen de numerieke waarde daarvan met de VAL-functie.
(123456,12345A,...,ABCDEF)
We bepalen achtereenvolgens de VAL (van Eng. Value) van de gehele string, de LEFT$(N$,2) en van MID$(N$,3,2).

U zult zien dat als tenminste het meest linkse karakter een digit is, dat de VAL-functie een waarde oplevert. Ook als de rest van de string niet-numeriek is.

```
10  REM ★★ DE VAL-FUNCTIE ★★
20  INPUT "VOLGENDE STRING"; N$
25  IF N$ = "ZZZZ" THEN 999
30  LET N = VAL(N$)
40  LET P = VAL(LEFT$(N$,2))
50  LET Q = VAL(MID$(N$,3,2))
```

```
60  PRINT " "
70  PRINT N, P, Q
80  PRINT " "
90  GOTO 20
999 END
```

Programma 5-15.

```
RUN
VOLGENDE STRING 123456
123456              12          34

VOLGENDE STRING 12345A
12345               12          34

VOLGENDE STRING 1234AB
1234                12          34

VOLGENDE STRING 123ABC
123                 12          3

VOLGENDE STRING 12ABCD
12                  12          0

VOLGENDE STRING 1ABCDE
1                   1           0

VOLGENDE STRING ABCDEF
0                   0           0
```

VAL geeft de waarde van de linkse digits, ook als de string verder geen letters en tekens bevat.

'A' staat nu op positie 3 dus VAL(A$) is nu nul.

K Programma 5-15. BBC regel 70: PRINT; N,; P,; Q

ZOP 9

Wat zijn de numerieke waarden van de volgende statements:

(a) VAL(A$) als A$ = "54"
(b) VAL(B$) als B$ = "76XY"
(c) VAL(C$) als C$ = "A3"
(d) VAL(D$) als D$ = "−132"
(e) VAL(LEFT$(E$,2)) als E$ = "593"
(f) VAL(LEFT$(F$,1)) als F$ = "8AM"
(g) VAL(LEFT$(G$,2)) als G$ = "Z35"
(h) VAL(RIGHT$(H$,1)) als H$ = "593"
(i) VAL(RIGHT$(I$,2)) als I$ = "8AM"
(j) VAL(RIGHT$(J$,2)) als J$ = "Z35"

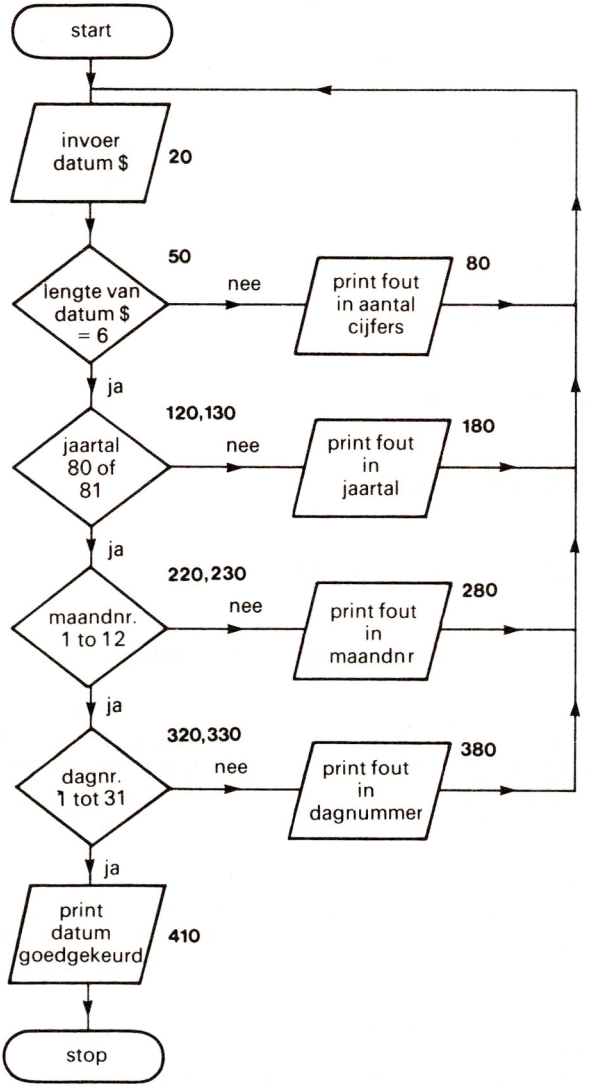
DATUM-stringcontrole

We zijn nu in staat om VAL in de praktijk toe te passen; in ons geval om de juistheid van data te controleren. Dit is precies wat er gebeurt in alle computers. We weten dat er vaak fouten worden gemaakt bij het intoetsen van data, dus laten we waar dat mogelijk is de computer zelf de fouten corrigeren.

Voorbeeld 5

Schrijf een programma voor datumcontrôle op de drie gegevens van 6-digit datum-string.

Oplossing
De datumstring kan worden onderscheiden in drie gedeelten:



Afb. 5-5. Stroomdiagram voor een datumcontroleprogramma.

We beschouwen alleen de jaren 1980 en 1981 dus:

VAL(LEFT$(D$,2)) moet een waarde van 80–81 hebben
VAL(MID$(D$,3,2)) moet een waarde van 1–12 hebben
VAL(RIGHT$(D$,2)) moet een waarde van 1–31 hebben.

Het controleren is een complexe procedure omdat we alle mogelijkheden en onmogelijkheden moeten nalopen. Het stroomdiagram is afgebeeld in afb. 5-5.

Het programma is recht-toe-recht-aan en passeert achtereenvolgens alle vier controles. Als één van de controles negatief is drukt het programma een foutmelding af, gevolgd door een herhaalde vraag om de datum. Is de datum goed bevonden dan gaat het programma naar regel 410 waar dit wordt bevestigd met een melding.

```
10  REM ★★ DATUM CONTROLE ★★
20  INPUT "VOLGENDE DATUM"; D$
30  IF D$ = "ZZZZ" THEN 900
50  IF LEN(D$) = 6 THEN 110                    ─ controle op lengte
80  PRINT "!!!!!!FOUT IN AANTAL CIJFERS!!!!!!"
90  GOTO 20
100 REM ★★★★★★★★
110 PRINT "AANTAL CIJFERS GOED"
120 IF VAL(LEFT$(D$,2)) = 80 THEN 210          ┐
130 IF VAL(LEFT$(D$,2)) = 81 THEN 210          ┘ controle op jaartal
180 PRINT "!!!!!!FOUT IN JAARTAL!!!!!!"
190 GOTO 20
200 REM ★★★★★★★★
210 PRINT "JAARTAL GOEDGEKEURD"
220 IF VAL (MID$(D$,3,2)) < 1 THEN 280         ┐
230 IF VAL (MID$(D$,3,2)) < = 12 THEN 310      ┘ controle op mndnr.
280 PRINT "!!!!!!FOUT IN MAANDNUMMER!!!!!!"
290 GOTO 20
300 REM ★★★★★★★★
310 PRINT "MAANDNUMMER GOEDGEKEURD"
320 IF VAL (RIGHT$(D$,2)) < 1 THEN 380         ┐
330 IF VAL (RIGHT$(D$,2)) < = 31 THEN 410      ┘ controle op dagnr.
380 PRINT "!!!!!!FOUT IN DAGNUMMER!!!!!!"
390 GOTO 20
400 REM ★★★★★★★★★★
410 PRINT "DATUM GOEDGEKEURD"
490 GOTO 20
900 PRINT "EINDE DATUM CONTROLE"
910 END
```

Programma 5-16.

```
RUN
VOLGENDE DATUM 1234567
!!!!!!FOUT IN AANTAL CIJFERS!!!!!!
VOLGENDE DATUM 123456
```

```
!!!!!!FOUT IN JAARTAL!!!!!!
VOLGENDE DATUM 803456
AANTAL CIJFERS GOED
JAARTAL GOEDGEKEURD
!!!!!!FOUT IN MAANDNUMMER!!!!!!
VOLGENDE DATUM 801256
AANTAL CIJFERS GOED
JAARTAL GOEDGEKEURD
MAANDNUMMER GOEDGEKEURD
!!!!!!FOUT IN DAGNUMMER!!!!!!
VOLGENDE DATUM 800131
AANTAL CIJFERS GOED
JAARTAL GOEDGEKEURD
MAANDNUMMER GOEDGEKEURD
DATUM GOEDGEKEURD
VOLGENDE DATUM ZZZZ
EINDE DATUM CONTROLE
```

[K] Programma 5-16.

Opgave 5

1. Schrijf een programma dat de frequentie telt waarmee elke klinker voorkomt en tevens een overzicht afdrukt van het totaal aantal klinkers en medeklinkers in het gedicht uit voorbeeld 2.
2. Schrijf een programma dat een string met karakters invoert en deze in omgekeerde volgorde weer afdrukt.

Samenvatting van hoofdstuk 5

Aan het eind van hoofdstuk 5 moet u de volgende zaken beheersen en kunnen toepassen in eenvoudige BASIC-programma's:

- LEN(A$)
- LET C(I) = C(I) + 1 om frequenties te tellen
- Het printen van frequentiediagrammen;
- Het gebruik van TAB(I) om kolommen te printen;
- Het gebruik van tekstkopjes en schaalverdelingen in diagrammen;
- LEFT$(X$,I) en RIGHT$(X$,I)
- MID$(X$,I,J)
- VAL(X$)

Antwoorden op ZOP's en oefeningen

ZOP 1

(a) 3 ; (b) 1 ; (c) 2 (geen 72, LEN telt het aantal tekens) (d) 7 (ook de spatie wordt als leesteken geteld)

ZOP 2
Uw antwoord moet zijn:

Woordlengte	Telling	Totaal
1		
2	111	3
3	111	3
4	11	2
5	1	1
6	1111	5
7		0
8		0
9	11	2
10	1	1
11		0
12	1	1
13		0
14		0
15		

Afb. 5-6.

ZOP 3
(a) C(3) = 1 (geen 3 want C(3) telt het aantal drieën)
(b) C(9) = 2
(c) C(1) = 4
(d) C(0) = 3

Oefening 1
Het antwoord vindt u in het vervolg van de paragraaf.

ZOP 4
★★
★★★★
★★★★★★★
★★★★★★★★
★★★
★

ZOP 5
```
10  PRINT "COL1"; TAB(9); "COL2"; TAB(19); "COL3"
```

ZOP 6
```
10  INPUT A, B, C
20  PRINT TAB(A); "TABULATOR"; TAB(B); "TABULATOR"; TAB(C); "TA-
    BULATOR"
```

Programma 5-17.

Oefening 2

```
10  REM ★★ FREQUENTIE DISTRIBUTIE ★★
20  REM ★★ FREQUENTIE LIJST IS F(K) ★★
25  DIM V$(5)
30  DIM F(6)
32  FOR I = 1 TO 5                    ┐
34  READ V$(I)                        ┘  leest de klinkers uit 800 in
36  NEXT I                               een klinkerlijst in.
40  LET K = 1                         ┐
50  READ F(K)                         │
60  IF F(K) = -9999 THEN 110          │  leest de frequenties uit 900
70  LET K = K + 1                     │
80  GOTO 50                           ┘
90  REM ★★★★★★★★
100 REM ★★ -9999 DOET NIET MEE ★★
110 LET N = K - 1
120 REM ★★★★★★★★
200 REM ★★ PRINT ROUTINE ★★
210 PRINT
212 PRINT "KLINKER"; TAB(8); "FREQ"; TAB(18); "TURVEN"
214 PRINT "=============================="
220 FOR X = 1 TO N
222 PRINT TAB(2); V$(X); TAB(7); "I"; TAB(10); F(X); TAB(14); "I"; TAB(18);
230 FOR Y = 1 TO F(X)  ──────────────  print woordlijst
240 PRINT "★";
250 NEXT Y
260 PRINT
280 NEXT X
290 PRINT "....SCHAAL...."; TAB(17); "0 . . . . 5 . . . . 0 . . . . 5 . . . . 0"
300 REM ★★★★★★★★
800 DATA A,E,I,O,U
900 DATA 9,16,10,10,6,-9999
910 END
```

Programma 5-18

```
RUN

KLINKER    FREQ          TURVEN
==============================
A          I  9   I    ★★★★★★★★★
E          I  16  I    ★★★★★★★★★★★★★★★★
I          I  10  I    ★★★★★★★★★★
O          I  10  I    ★★★★★★★★★★
U          I  6   I    ★★★★★★
....SCHAAL....          0 . . . . 5 . . . . 0 . . . . 5 . . . . 0
```

K Programma 5-18

ZOP 7

(a) 1;
(b) 1A2B;
(c) C4D;
(d) 3C4D

LEFT$ en RIGHT$ behandelen cijfers en letters op gelijke wijze.

Oefening 3

```
10  REM ★★ KLINKER-TEST: ★★
20  READ W$
30  IF W$ = "ZZZZ" THEN 9999
40  LET L$ = LEFT$ (W$,1)
50  IF L$ = "A" THEN 200
60  IF L$ = "E" THEN 200
70  IF L$ = "I" THEN 200
80  IF L$ = "O" THEN 200
90  IF L$ = "U" THEN 200
100 GOTO 20
190 REM ★★★★★★★★
200 PRINT
210 PRINT L$, W$
220 GOTO 20
230 REM ★★★★★★★★
900 DATA ALS,EEN,LOSGELATEN,PIJL
910 DATA ERGENS,IN,EEN,GROTE,STAD,BELAND
920 DATA TUSSEN,AUTO'S,EN,FABRIEKEN
930 DATA TUSSEN,SNELWEGEN,EN,FLATGEBOUWEN,ZZZZ
9999 END
```

Programma 5-19.

```
RUN
A           ALS

E           EEN

E           ERGENS

I           IN

E           EEN

A           AUTO'S

E           EN

E           EN
```

[K] Programma 5-19.

Oefening 4

```
10  REM ★★ STRING TEST ★★
20  INPUT "VOER EEN 6-KARAKTER STRING IN"; X$
30  PRINT TAB(10); "1234567890"
40  FOR I = 1 TO 6
50  LET A$ = RIGHT$(X$,I)
60  PRINT I; TAB(16-I); A$
70  NEXT I
80  END
```

BBC regel 60: PRINT;...enz.

Programma 5-20.

```
RUN
VOER EEN 6-KARAKTER STRING IN ABCDEF
          1234567890
1                   F
2                  EF
3                 DEF
4                CDEF
5               BCDEF
6              ABCDEF
```

K Programma 5-20.

ZOP 8

```
10  INPUT "VOLGENDE TELEFOONNUMMER"; N$
20  LET A$ = MID$(N$,4,LEN(N$)-3)
30  PRINT A$
40  GOTO 10
```

Programma 5-21.

K Programma 5-21

ZOP 9

(a) 54
(b) 76 (stopt bij 1e letter)
(c) 0 (begint met een letter)
(d) −132
(e) 59
(f) 8
(g) 0 (begint met een letter)
(h) 3
(i) 0
(j) 35

6 Over dobbelstenen en spelletjes

6.1 Random-getallen

De programmeerfunctie die wat speelsheid in een programma aanbrengt is de random-getallenfunctie. Deze functie vormt het hart van de meeste spelletjes en simulatieprogramma's op vrijwel alle gangbare microcomputers.

Bij het spelen van vele spelletjes bent u random-getallen tegengekomen. Spelletjes met het gooien van een munt of het werpen van een dobbelsteen of het trekken van lootjes uit een hoed. Dergelijke huis-tuin-keukenspelletjes zijn gemeengoed geworden in casino's, bingo-clubs, het trekken van de lottogetallen en het uitloten voor de voetbalcompetitie. Hoewel we een gevoelsmatig idee hebben van wat random-getallen zijn, is het zeer moeilijk een duidelijke definitie op te stellen. Laten we eens naar enige reeksen getallen kijken en proberen het begrip wat te verhelderen.
In gedachten werpen we een dobbelsteen met zes vlakken driemaal 15 keer. Veronderstel dat die driemaal hieronder als een respectievelijke A, B en C-reeks zijn voorgesteld.

Reeks A
5,1,2,4,5,3,2,1,6,3,5,4,3,4,2

Reeks B
6,6,6,6,6,6,6,6,6,6,6,6,6,6,6

Reeks C
1,2,3,4,5,6,1,2,3,4,5,6,1,2,3

Afb. 6-1. Random-getallen?

De meesten van ons zullen het ermee eens zijn dat reeks A een goede weergave is van een werkelijk gegooide dobbelsteen. Dit patroon is onregelmatig en zonder herhalingen en elk nummer lijkt even vaak voor te komen. Niemand verbaast zich over de korte reeksen cijfers die in volgorde als subreeks in A voorkomen.
Als contrast is reeks B afgedrukt die wel zeer onwaarschijnlijk voorkomt. Niemand verwacht vijftien maal 6 te werpen in 15 achtereenvolgende worpen. Als dit wel zo was, waren we hoogst verrast geweest en hadden we de dobbelsteen ervan verdacht niet goed te zijn. Zo gemakkelijk als reeks A door ons als echt wordt geaccepteerd, zo onwaarschijnlijk komt reeks B ons voor.

Een andere eigenschap van reeksen random-getallen is, en dat weten we ook uit

ervaring, dat toevallig voorkomende subreeksen zich in de loop van de tijd vanzelf uitschiften. Daarmee bedoelen we dat na verloop van bijv. honderd worpen we ongeveer 15 énen; 16 tweeën enz. kunnen aantreffen. Met andere woorden: na verloop van grotere aantallen worpen gaan de wetten der kansberekening gelden. Als we nu reeks C bekijken, dan zou over een langere periode aan de kansverdelingswetten zijn voldaan. Maar ook deze reeks is al op het eerste gezicht niet acceptabel als random, omdat het niet aannemelijk is dat de reeks 1,2,3,4,5,6 zich zo vaak zal herhalen.
De verdeling van worpen over een groot aantal metingen en de aannemelijkheid van de reeks zijn de voornaamste criteria voor een random-reeks. Er bestaan wel statistische methoden om deze eigenschappen te testen, maar deze vallen buiten het bestek van dit boek.

U zult niet verbaasd zijn dat onze computer ook random-getallen kan genereren. Hoe dat precies in zijn werk gaat, is op dit moment voor ons nog niet interessant. Het is voldoende dat we de RND-functie beschouwen alsof zij leest uit een tabel van grote lengte, gevuld met echte random-getallen. Na verloop van lange tijd herhaalt de reeks zich, doch dit is voor de meeste toepassingen van geen belang. Om een andere reeks getallen te krijgen volstaat het een ander beginpunt in de tabel te kiezen. Dit kiezen geschiedt met een getal dat over het algemeen 'seed' wordt genoemd. Random-getallenreeksen starten dan ook vanuit een bepaald 'seed'. Omdat de reeks zich na verloop van tijd herhaalt heten deze reeksen pseudo-random-getallenreeksen

6.2 De RND-functie

Als u in het bezit bent van een BBC-computer kunt u deze paragraaf overslaan en verder gaan met 6.3 RND op de BBC-computer. De BBC-functie is veel eenvoudiger te gebruiken dan de functie uit deze paragraaf.

Om random-getallen te genereren op een microcomputer kunt u de RND-functie gebruiken. RND verschilt van computer tot computer, zodat de RND-functie op uw microcomputer enigszins van de hier beschreven functie kan verschillen. Voor de juiste details verwijzen we u naar de gebruiksaanwijzing van uw computer.

De RND-functie heeft een argument nodig. Een argument is een numerieke expressie tussen haakjes.

```
RND(A)
      └── A is het argument van RND
```

A kan negatief zijn, nul of positief en heeft dan verschillende effecten tot gevolg. Het volgende programma demonstreert dat voor u:

```
10  REM ★★ DE RND-FUNCTIE ★★
20  INPUT "ARGUMENT VOOR RND : ";A
30  FOR I = 1 TO 10
40  LET B = RND(A)
50  PRINT B
```

```
60  NEXT I
70  END
```

Programma 6-1 Het effect van A op RND.

```
RUN
ARGUMENT VOOR RND: -1
0.32653747
0.32553747
0.32653747
0.32653747
0.32653747
0.32653747
0.32653747
0.32653747
0.32653747
0.32653747
0.32653747
```

Een negatief argument houdt het RND-resultaat op een vaste toevallige waarde.

```
RUN
ARGUMENT VOOR RND: 0
0.844109312
0.844109312
0.844109312
0.844109312
0.844109312
0.844109312
0.844109312
0.844109312
0.844109312
0.844109312
0.844109312
```

Als A = 0 blijft het resultaat eveneens constant

```
RUN
ARGUMENT VOOR RND: 1
0.996946841
0.155770048
0.123150311
0.55955522
0.722228633
1.81242754E-3 ★
0.596672254
0.247531421
0.722038242
0.34534098
```

Deze reeks lijkt meer op random

★ De E-notatie wordt behandeld in hoofdstuk 7.

```
RUN
ARGUMENT VOOR RND: 2
0.446516422
0.843781732
0.619249787
```

```
0.192015309
0.750175865
0.777580239
0.999965884
4.68787511E-2
0.499048179
0.758298454
```

En deze waarden ook. Zij zijn echter gelijk aan die bij A = 1

K Programma 6-1 om het effect van A op uw microcomputer uit te proberen.

De RANDOMIZE-functie

Wat het genereren van random-getallen betreft houden we ons verder niet bezig met argumenten ongelijk aan 1. De RND(1)-functie genereert steeds dezelfde reeks getallen als het programma opnieuw wordt gestart. (het lijkt misschien vreemd, maar het is heel handig bij het testen van programma's met steeds dezelfde reeks getallen) Om de 'seed' te vernieuwen, met andere woorden de random-generator op een andere plaats in de reeks te laten beginnen, gebruiken we het RANDOMIZE-statement. Het gebruik ervan ziet u in het volgende voorbeeld.

```
10  REM ★★ RANDOMIZE ★★
20  RANDOMIZE
30  FOR I = 1 TO 10
40  LET B = RND(1)
50  PRINT B
60  NEXT I
70  END
```

Programma 6-2 RANDOMIZE om de de volgorde te variëren.

```
RUN
0.40333726
0.259248312
2.74720457E-2
0.595798638
0.494966722
0.313430218
0.637142083
5.93551279E-2
0.849180383
0.127667247

RUN
0.345039618
0.862045728
0.722938438
0.896813678
0.230721752
0.412861442
0.747279305
9.32389824E-2
0.691675241
0.578488109
```

```
RUN
0.483629006
0.167380924
2.84756666E-2
1.34610012E-3
0.850183225
0.645714817
0.232323825
6.44814279E-2
0.809638755
3.36378207E-3
```

[K] Programma 6-2 (uw computer heeft RANDOMIZE misschien niet nodig)

RND(1)

Wat het bovenstaande voorbeeld goed demonstreert, is dat RND(1) random-getallen genereert in het bereik 0-1.

Hoe kunnen we nu andere random-getallen genereren? Heel eenvoudig door het resultaat met een constante te vermenigvuldigen.
Dus:

RND(1) geeft een random-getal tussen 0 en 1;
6★RND(1) geeft random-getallen tussen 0 en 6;
52★RND(1) geeft random-getallen tussen 0 en 52.

U kunt RND(1) beschouwen als een variabele 'conversiefactor' tussen nul en een, die door het toeval bepaald wordt.

Het volgende voorbeeld demonstreert dit idee:

```
10  REM ★★ RND ALS CONVERSIEFACTOR ★★
20  RANDOMIZE
30  PRINT "I", "RND(1)", "6★RND(1)", "52★RND(1)"
40  PRINT "---", "------", "------", "------"
50  FOR I = 1 TO 10
60  LET B = RND(1)
70  LET C = 6★B
80  LET D = 52★B
90  PRINT I,B,C,D
100 NEXT I
110 END
```

Programma 6-3 RND(1) als conversiefactor.

```
RUN
I           RND(1)          6★RND(1)        52★RND(1)
---         -------         --------        ---------
1           .58041          3.48246         30.1813
2           .128928         .773567         6.70424
3           .928324         5.56994         48.2728
4           .901162         5.40697         46.8604
```

5	.532818	3.19691	27.7065
6	.499882	2.99929	25.9939
7	.286114	1.71669	14.8779
8	.608704	3.65223	31.6526
9	.342298	2.05379	17.7995
10	.163376	.980258	8.49557

RUN

I	RND(1)	6★RND(1)	52★RND(1)
1	.438479	.263087	2.28009
2	.891465	5.34879	46.3652
3	.801263	4.80758	41.6657
4	.656113	3.93668	34.1179
5	.16401	.98457	8.52849
6	.835579	5.01347	43.4501
7	.238398	1.43039	12.3967
8	.0730044	.438026	3.79623
9	.581113	3.48668	30.2179
10	.0987764	.592658	5.13637

[K] Programma 6-3 Indien de print breder is dan uw scherm, pas dan in regel 90 de TAB-functie toe. Bijvoorbeeld:

```
90  PRINT I; TAB(4); B; TAB(18) ;C ;D
```

tevens dient u regel 30 en 40 hierop aan te passen.

ZOP 1
Schrijf een programma dat zes random-getallen uitprint in het bereik van 0-5.999999.

De RND+1-functie
Bekijkt u nog eens de getallen uit programma 3, die met 6★RND(1) werden gegenereerd; deze getallen waren:

3.48246
.773567
5.56994
5.40697
3.19619
2.99929
1.71669
3.65223
2.05379
.980258

De cijfers voor de decimale punt (in kleur) zijn:

3,0,5,5,3,2,1,3,2,0

Dit zijn elementen uit de verzameling:

(0,1,2,3,4,5)

Stel dat we de RND-functie willen gebruiken voor het genereren van de getallen op een dobbelsteen (1,2,3,4,5,6). Daartoe hoeven we slechts een constante waarde van 1 bij de gevonden resultaten op te tellen. In veel spellen willen we een worp van een dobbelsteen simuleren of het trekken van een kaart uit een kaartspel. We zijn daarom speciaal geïnteresseerd in de functies:

6★RND(1)+1 en 52★RND(1)+1

Het volgende programma maakt gebruik van deze beide functies:

```
10  REM ★★ RND+1 ALS CONVERSIEFACTOR ★★
20  RANDOMIZE
30  PRINT "I","RND(1)","6★RND(1)+1","52★RND(1)+1"
40  PRINT "---","------", "----------", "----------"
50  FOR I = 1 TO 10
60  LET B = RND(1)
70  LET C = 6★B+1
80  LET D = 52★B+1
90  PRINT I,B,C,D
100 NEXT I
110 END
```

Let op +1 op elke regel

Programma 6-4 6★RND(1)+1 en 52★RND(1)+1

```
RUN
I           RND(1)          6★RND(1)+1      52★RND(1)+1
---         ------          ----------      ----------
1           .58041          4.48246         31.1813
2           .128928         1.77357         7.70424
3           .928324         6.56994         49.2728
4           .901162         6.40697         47.8604
5           .532818         4.19691         28.7065
6           .499882         3.99929         26.9939
7           .286114         2.71669         15.8779
8           .608704         4.65223         32.6526
9           .342298         3.05379         18.7995
10          .163376         1.98026         9.49557

RUN
I           RND(1)          6★RND(1)+1      52★RND(1)+1
---         ------          ----------      ----------
1           .0438479        1.26309         3.28009
2           .891465         6.34879         47.3562
3           .801263         5.80758         42.6657
4           .656113         4.93668         35.1179
5           .16401          1.98406         9.52849
```

6	.835579	6.01347	44.4501
7	.238398	2.43039	13.3967
8	.0730044	1.43803	4.79623
9	.581113	4.48668	31.2179
10	.0987764	1.592666	6.13637

[K] Programma 6-4.

De INT-functie

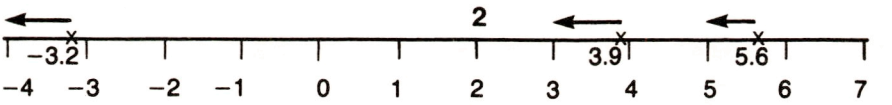
Zoals u in de bovenstaande kolommen ziet, hebben we nu precies de random-getallen gegenereerd die we wilden hebben. Maar wat doen we nu met het 'afval' rechts van de decimale punt?

Er bestaat een speciale functie om de decimalen van een getal af te knippen: de INT-functie (Engels: INTEGER)

De INT-functie bepaalt het grootste gehele getal dat kleiner dan of gelijk is aan het argument.

INT (5.6) = 5
INT (3.9) = 3
INT (−3.2) = −4
INT (2) = 2

Als het resultaat INT (−3.2) = −4 u verbaast, kijkt u dan eens naar de getallenrechte en herinnert u zich dat INT het gehele getal neemt dat niet groter is dan het argument. INT is geen afronding!



Afb. 6-2. Getallenrechte.

ZOP 2

Geef de waarden van de volgende functies aan:

(a) INT(4.5)
(b) INT(9.1)
(c) INT(−2.5)
(d) INT(−0.99)
(e) INT(1.01)

Met behulp van RND en INT kunnen we exact het werpen met een dobbelsteen of het trekken van een kaart simuleren, namelijk met:

INT(6★RND(1)+1 en
INT(52★RND(52)+1)

Het volgende programma geeft een simulatie van beide spellen in tabelvorm:

```
10  REM ★★ DE INT-FUNCTIE ★★
20  RANDOMIZE
30  PRINT "I","RND(1)","INT(6★...+1)","INT(52★...+1)"
40  PRINT "---", "------", "----------","----------"
50  FOR I = 1 TO 10
60  LET B = RND(1)
70  LET C = INT(6★B+1)
80  LET D = INT(52★B+1)
90  PRINT I,B,C,D
100 NEXT I
110 END
```

Programma 6-5 INT genereert gehele getallen

```
RUN
I          RND(1)          INT(6★...+1)    INT(52★...+1)
---        ------          ----------      ----------
1          .58041          4               31
2          .128928         1               7
3          .928324         6               49
4          .901162         6               47
5          .532828         4               28
6          .499882         3               26
7          .286114         2               15
8          .608704         4               32
9          .342298         3               18
10         .163376         1               9

RUN
I          RND(1)          INT(6★...+1)    INT(52★...+1)
---        ------          ----------      ----------
1          0.438479        1               3
2          .891465         6               47
3          .801263         5               42
4          .656113         4               35
5          .16401          1               9
6          .835579         6               44
7          .238398         2               13
8          .0730044        1               4
9          .581113         4               31
10         .0987764        1               6
```

[K] Programma 6-5.

6.3 RND op de BBC-computer

RND(1) op de BBC-computer geeft u direct random-getallen tussen 0 en .99999999.

Dus:

```
10  FOR I = 1 TO 10
20  PRINT RND(1)
30  NEXT I
```

Programma 6-6 RND(1) op de BBC-computer.

```
RUN
0.50939
0.94965
0.98707
0.5473
0.13987
0.92918
0.33882
2.6349E-2
0.92088
0.24963
```

Wilt u random-getallen groter dan 1, bijvoorbeeld tussen 0 en 100, dan vermenigvuldigt u het resultaat met een geschikte factor:

100★RND(1) in regel 20

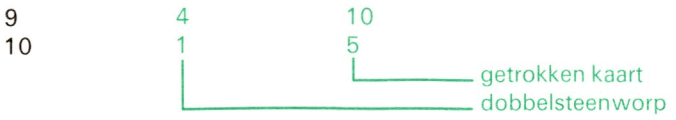
RND(n)
Voor dobbelsteen- en kaartspelen hebben we gehele getallen nodig (integers). Op de BBC-computer is dat wel heel eenvoudig:

RND(6) geeft RANDOM gehele getallen tussen 1 en 6
RND(52) geeft RANDOM gehele getallen tussen 1 en 52.

```
10  REM ★★ RND(n) – BBC ★★
20  PRINT; "I"; TAB(5) "RND(6)"; TAB(15) "RND(52)"
30  FOR I = 1 TO 10
40  LET A = RND(6)
50  LET B = RND(52)
60  PRINT; I; TAB(6) A; TAB(16) B
70  NEXT I
80  END
```

Programma 6-7 RND(n) - BBC.

```
RUN
I           RND(6)      RND(52)
1           1           31
2           6           32
3           6           23
4           2           6
5           4           17
6           3           45
7           4           44
8           1           36
```



K Programma 6-7.

In de programma's die hierna volgen zult u regelmatig een regel tegenkomen met RANDOMIZE. Indien u een BBC-computer gebruikt dient u deze regel weg te laten.

6.4 Random-getallen (slot)

Het volgende programma simuleert honderd maal de worp van een dobbelsteen:

```
10  REM ★★ 100 DOBBELSTEENWORPEN ★★
20  RANDOMIZE
30  FOR I = 1 TO 10
40  FOR J = 1 TO 10
50  LET X = INT(6★RND(1)+1)
60  PRINT X;
70  NEXT J
80  PRINT
90  NEXT I
100 END
```

Programma 6-8 Dobbelsteen.

```
RUN
4  1  6  6  4  3  2  4  3  1
6  6  2  2  5  3  6  5  2  2
3  3  6  3  4  2  6  1  2  6
6  1  4  3  5  2  2  2  1  6
6  3  1  1  6  3  4  5  2  5
6  1  4  2  2  4  5  1  2  4
6  6  5  4  3  4  5  5  1  5
3  5  4  3  5  3  6  6  6  6
1  5  3  4  3  1  1  4  2  1
6  3  6  1  6  1  6  5  3  1
```

K Programma 6-8
BBC: laat regel 20 weg
: regel 50 LET X = RND(6)
: regel 60 PRINT; X; " ";

Om er zeker van te zijn dat u de INT-functie heeft begrepen kunt u de volgende programma's uitproberen. (Niet geschikt voor de BBC-computer)

ZOP 3
Gegeven het volgende programma:

```
10  REM ★★ ZOP 3 ★★
20  FOR X = -3.8 TO -1.8 STEP(.2)
30  LET Y = INT(X)
40  PRINT X,Y
50  NEXT X
60  END
```

Programma 6-9

Dit programma print 10 getallen. Welke getallen zijn dit?.

ZOP 4
Gegeven het programma:

```
10  REM ★★ ZOP 4 ★★
20  FOR X = 1.6 TO 3.4 STEP(.2)
30  LET Y = INT(X)
40  PRINT X,Y
50  NEXT X
60  END
```

Programma 6-10

Welke 9 getallen worden hier geprint?

6.5 Twee voorbeelden

Deze paragraaf bestaat uit twee uitgewerkte lange voorbeelden. Probeert u de problemen eerst eens zelf op te lossen en vergelijk daarna uw oplossing met de in het boek gegeven uitwerking.

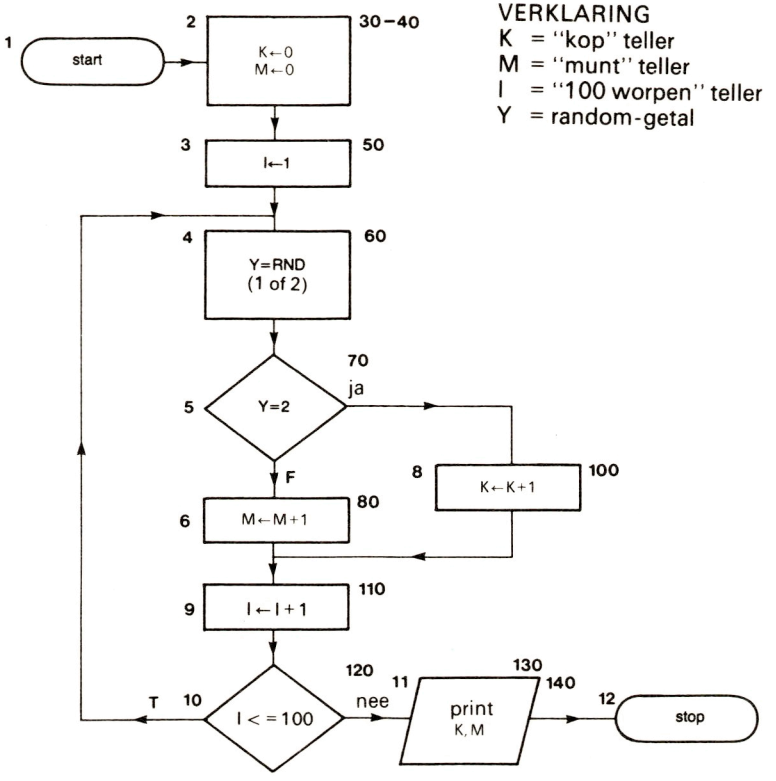
Voorbeeld 1
Schrijf een programma dat het tossen met een muntstuk simuleert. Werp 100 maal en geef een overzicht van het aantal malen kop en munt.

Oplossing 1
De kern van het probleem is een random-getallengenerator die alleen een 1 of 2 genereert. We gebruiken de 1 om de munt en de 2 om de kop te representeren. Hieronder volgt een beschrijvend algorithme voor dit probleem.

1. Start.
2. Maak tellers voor kop en munt nul.
3. Initialiseer lus-teller
4. Maak random getal van 1 of 2.
5. Als random getal = 1 ga dannaar 8, ga anders naar 6.
6. Verhoog kop-teller.
7. Ga naar 9.
8. Verhoog munt-teller.
9. Verhoog lus-teller.
10. Als lus-teller < = 100 ga dannaar 4, ga anders naar 11.
11. Print totaal aantal kop en munt
12. STOP.

Afb. 6-3. Beschrijvend algorithme van kop of munt.

Misschien geeft u de voorkeur aan een stroomdiagram:



Afb. 6-4. Stroomdiagram van kop of munt.

en tot slot het programma:

```
10  REM ★★ 100× KOP OF MUNT ★★
20  RANDOMIZE
30  LET K = 0
40  LET M = 0
50  LET I = 1
60  LET Y = INT(2★RND(1) + 1)
70  IF Y = 2 THEN 100
80  LET M = M + 1
90  GOTO 110
100 LET K = K + 1
110 LET I = I + 1
120 IF I < = 100 THEN 60
130 PRINT "KOP", "MUNT"
140 PRINT K, M
150 END
```

Programma 6-11 Kop of munt.

```
RUN
KOP          MUNT
51           49

RUN
KOP          MUNT
57           43

RUN
KOP          MUNT
55           45
```

[K] Programma 6-11.
BBC: laat regel 20 weg
regel 60 LET Y = RND (2)
regel 140 PRINT; K, M

Voorbeeld 2
Schrijf een programma dat het werpen van twee munten simuleert. Tel het aantal malen kop-kop, munt-munt en het aantal malen kop-munt of munt-kop.

Oplossing
We gebruiken dezelfde notaties en afspraken. 1 voor munt en 2 voor kop. De eerste worp gaat in variabele C1 en de tweede worp in C2. De som van beide geeft

S = C1 + C2

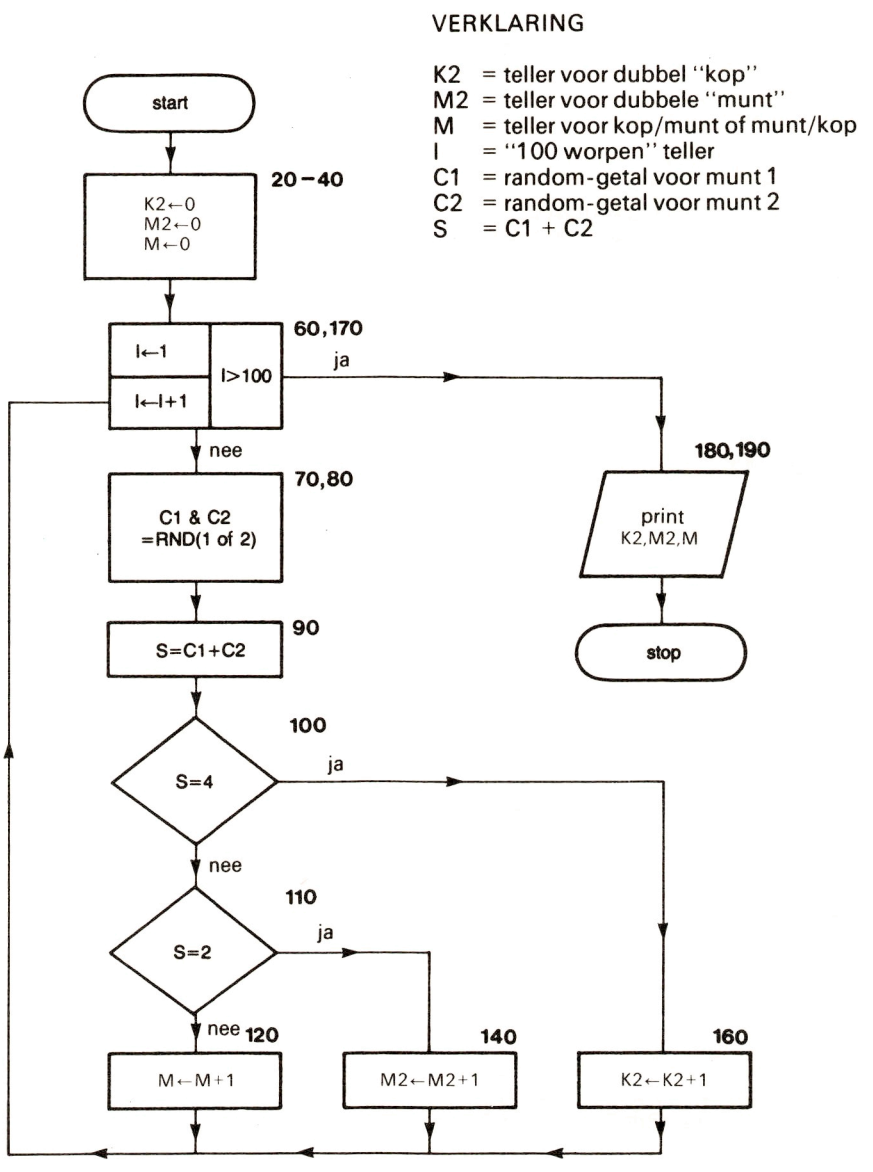
en S kan dus 2, 3 of 4 zijn voor respectievelijk MM, KM of MK en KK.

Resultaat	Score
MM	1+1=2
MK of KM	1+2=2+1=3
KK	2+2=4

Daarna behoeven we alleen nog maar het aantal malen 2, 3 of 4 te tellen.

Teller voor twee koppen: K2
Teller voor mix: M
Teller voor twee munten: M2

Het stroomdiagram is:



Afb. 6-5. Stroomdiagram voor tweemaal kop of munt.

en het programma is:

```
10  REM ★★ 100× TOSSEN MET 2 MUNTEN ★★
20  LET K2 = 0
30  LET M2 = 0
40  LET M = 0
50  RANDOMIZE
60  FOR I = 1 TO 100
70  LET C1 = INT(2★RND(1)+1)
80  LET C2 = INT(2★RND(1)+1)
90  LET S = C1 + C2
100 IF S = 4 THEN 160
110 IF S = 2 THEN 140
120 LET M = M + 1
130 GOTO 170
140 LET M2 = M2 + 1
150 GOTO 170
160 LET K2 = K2 + 1
170 NEXT I
180 PRINT "MM", "KM", "KK"
190 PRINT M2, M, K2
200 END
```

Programma 6-12 Tweemaal kop of munt.

```
RUN
MM          KM          KK
21          54          25

RUN
MM          KM          KK
26          48          26

RUN
MM          KM          KK
24          55          21
```

[K] Programma 6-12
BBC: laat regel 50 weg
regel 70 LET C1 = RND(2)
regel 80 LET C2 = RND(2)
regel 190 PRINT; M2, M, K2

6.6 Score bijhouden

U zult bemerkt hebben dat we vreemde variabelenamen voor de tellers hebben gebruikt. Kunnen we daarvoor geen lijst gebruiken op dezelfde manier als bij frequentietabellen? Jawel, laten we eens een lijst S(I) (van Score) toepassen.

Als de score 2 is, verhoog dan S(2)
Is de score 3, verhoog dan S(3)
Algemeen:
Is de score S verhoog dan teller S(S).

In voorbeeld 2 kunnen we regel 10 tot 90 ongewijzigd laten en ons nieuw score-systeem op regel 100 toevoegen:

Het gehele programma wordt dan:

```
10  REM ★★ 100× TOSSEN MET 2 MUNTEN ★★
15  DIM S(4)
20  LET S(4) = 0
30  LET S(3) = 0
40  LET S(2) = 0
50  RANDOMIZE
60  FOR I = 1 TO 100
70  LET C1 = INT(2★RND(1)+1)
80  LET C2 = INT(2★RND(1)+1)
90  LET S = C1 + C2

110 NEXT I
120 PRINT "MM", "KM", "KK"
130 PRINT S(2), S(3), S(4)
140 END
RUN                                          Programma 6-13.
MM          KM          KK
21          58          21
RUN
MM          KM          KK
27          47          26
RUN
MM          KM          KK
31          52          17
```

[K] Programma 6-13
BBC: laat regel 50 weg
regel 70 LET C1 = RND(2)
regel 80 LET C2 = RND(2)
regel 130 PRINT; S(2), S(3), S(4)

Score-lijsten voor dobbelstenen
De score-lijst voor een worp met een dobbelsteen zou er als volgt uit kunnen zien:

S(1), S(2), S(3), S(4), S(5), S(6)

en die voor het gooien met twee dobbelstenen:

S(2), S(3), S(4),...,S(12).

Oefening 1
Schrijf een programma dat 100 dobbelsteenworpen simuleert. Tel het aantal malen dat elk cijfer van 1–6 voorkomt.

Oefening 2
Pas het programma zo aan dat het programma twee dobbelstenen simuleert en de som van beide als worp beschouwt.

Oefening 3
Schrijf een programma dat de gegevens uit oefening 2 in een frequentiediagram afbeeldt.

6.7 Afkortingen in BASIC

Onze programma's beginnen zo langzamerhand behoorlijk lang te worden en hoe langer ze zijn, des te meer tikwerk gaat er in het invoeren zitten. Er zijn een aantal trucs om dat wat in te perken. Tot nu toe hebben we deze nog niet gebruikt omdat de leesbaarheid van het programma onze prioriteit heeft. Hier en daar zullen we wat afkortingen gebruiken bij wijze van voorbeeld, maar over het algemeen blijven we liever te duidelijk dan onoverzichtelijk.

De belangrijkste afkortingen zijn:

1. Het woord LET mag meestal worden weggelaten, bijvoorbeeld:

   ```
   20  LET A = B
   ```

   mag worden geschreven als:

   ```
   20  A = B
   ```

2. Meer dan één statement per regel is toegestaan. Deze moeten door een ':' worden gescheiden, bijvoorbeeld:

   ```
   10  LET A = 7
   20  LET B = 8
   30  PRINT A + B
   ```

   kan vervangen worden door:

   ```
   10  LET A=7: LET B=8: PRINT A + B
   ```

   en met gebruik van de eerste afkortmogelijkheid:

   ```
   10  A=7: B=8: PRINT A + B
   ```

   Het onderbrengen van verschillende statements in een regel bespaart zowel tikwerk als rekentijd. Hoe minder regelnummers hoe minder tijd er aan het lezen daarvan verloren gaat.

3. Het woord PRINT mag in veel computers worden vervangen door '?'. (P. op de BBC-computer). Bij het listen van het programma zult u merken dat het vraagteken is vervangen door PRINT. Als u intoetst:

```
10  A=7: B=8: ? A+B
```

en u vraagt vervolgens om een LIST, dan krijgt u

```
10  A=7: B=8: PRINT A+B
```

4. In een PRINT-statement mag u expressies gebruiken. Deze worden uitgerekend voordat ze worden uitgeprint. We hebben dat hierboven al toegepast in PRINT A+B.

Maar LET OP!!!

- Schrijf niet willekeurige statements op een regel, alleen maar omdat het is toegestaan.
- Wees zorgvuldig met GOTO, IF...THEN en later ook met GOSUB... De regelnummers die in het statement worden genoemd moeten ook werkelijk bestaan en het programma begint aan het éérste statement op die regel.
- De meeste programma's besteden 80% van hun rekentijd in 20% van de regels. Concentreert u zich op die regels als u inkort. In oefening 3 waren dat de regel 70-120.
- Regels met een REM-statement worden door de computer gelezen. Daarom is het beter de REM's op te nemen aan het einde van programma-statements. In programma's die later in dit hoofdstuk voorkomen zullen we dat demonstreren.
- Bij het intoetsen van programma's uit dit boek dient u veel aandacht te besteden aan ';' en ':'. Vooral in tijdschriften wordt door een slordige druktechniek dit verschil niet duidelijk genoeg. Ook tekens als (, < en), > kunnen daardoor snel verkeerd gelezen worden.

Enkele voorbeelden

Uit programma 6-3 kunnen zes regels worden herschreven in drie regels:

```
10  REM ★★ RANDOMIZE ★★
20  RANDOMIZE
30  FOR I = 1 TO 10: PRINT RND(1): NEXT I
```

3 statements gescheiden door ':'

RND is bepaald in PRINT-statement

Programma 6-14.

Programma 6-13 was 14 regels lang en kan worden ingekort tot 10:

```
10  REM ★★ 100× TOSSEN MET 2 MUNTEN ★★
20  DIM S(4): S(4) = 0: S(3) = 0: S(2) = 0
50  RANDOMIZE
60  FOR I = 1 TO 100
70  C1=INT(2★RND(1)+1):C2=INT(2★RND(1)+1)
90  S = C1 + C2: S(S) = S(S)+1
110 NEXT I
```

3× LET vervalt

hier wordt het rekenwerk gedaan, regel 70-90 worden 100 × uitgevoerd

```
120 PRINT "MM", "KM", "KK"
130 PRINT S(2),S(3),S(4)
140 END
```

hier geen verkorte schrijfwijze omdat dit slechts éénmaal per run wordt uitgevoerd.

Programma 6-15.

ZOP 5
Probeer programma 6-23 in verkorte schrijfwijze weer te geven.

6.8 Het koppelen van strings

Nadat we in hoofdstuk 5 met veel pijn en moeite er in geslaagd zijn strings in stukjes te knippen, gaan we ons nu bezig houden met het aaneenkoppelen van strings. Programma 6-16 laat zien wat we daarmee bedoelen.

```
20  INPUT A$
30  INPUT B$
40  PRINT A$ + B$
50  END
```

Programma 6-16 Koppelen van strings.

```
RUN
? AAN
? KOPPELEN
AANKOPPELEN

RUN
? SAMEN
? STELLEN
SAMENSTELLEN
```

ZOP 6
Schrijf een programma om woorden in het meervoud af te drukken. Ga ervan uit dat het meervoud van een zelfstandig naamwoord altijd verkregen wordt met het toevoegsel 'en'

Programma 6-17 laat zien hoe we een aantal losse karakters tot een string aaneen kunnen rijgen. In regel 40 zijn de afzonderlijke karakters opgeslagen en in de regels 110-140 wordt er steeds één letter aan de totale string toegevoegd.

```
10  REM ★★ SAMENSTELLEN VAN STRINGS ★★
20  REM ★★ VOORBEREIDING ★★
25  DIM A$(10)
30  FOR I = 1 TO 10: READ A$(I): NEXT I
40  DATA A,B,C,D,E,F,G,H,I,J
50  REM ★★★★★★★★★★★★★★★★
100 C$ = " ": REM ★★ C$ = LEEG ★★
110 FOR J = 1 TO 10
120 C$ = C$ + A$(J)
130 PRINT J, C$
140 NEXT J
150 END
```

Programma 6-17.

```
RUN
1             A
2             AB
3             ABC
4             ABCD
5             ABCDE
6             ABCDEF
7             ABCDEFG
8             ABCDEFGH
9             ABCDEFGHI
10            ABCDEFGHIJ
```

K Programma 6-17. BBC: regel 130 PRINT; J, C$

Dit proces is erg belangrijk in tekstanalyse, maar wij zullen het voornamelijk gebruiken bij spelletjes.

6.9 STR$

Deze functie heeft het omgekeerde effect van de VAL-functie. STR$ geeft de string representatie van het argument van X.

STR$(N) lijkt veel op N als het in een PRINT-statement wordt opgenomen.

```
10  REM ★★ DE STR$-FUNCTIE ★★
20  INPUT "VOLGENDE GETAL"; N
25  PRINT "012345678901234567890"
30  PRINT N, STR$(N)
40  END
```

Programma 6-18.

```
RUN
VOLGENDE GETAL  17
01234567890123456789O
 17              17

RUN
VOLGENDE GETAL -17
01234567890123456789O
-17             -17

RUN
VOLGENDE GETAL  99.34
01234567890123456789O
 99.34           99.34

RUN
VOLGENDE GETAL -99.34
01234567890123456789O
-99.34          -99.34
```

K Programma 6-18. BBC: regel 20 laat; weg
regel 30 PRINT; N, STR$(N)

Toch wordt in de tweede helft van de printregel een string geprint en geen getal. De BBC-computer laat dit wel heel duidelijk zien als we het volgende statement toevoegen:

```
35 PRINT; N,N
```

Nu wordt in beide helften een getal afgedrukt.

In het volgende programma (programma 6-19) maken we gebruik van STR$ om bijvoorbeeld het karakter 8 (in tegenstelling tot de waarde 8) aan het einde van een string te koppelen.

```
50 REM ★★★★★★★★★★★★★★★★
100 C$ = " ": REM ★★ C$ = LEEG ★★
110 FOR J = 1 TO 10
120 C$ = C$ + STR$ (J)
130 PRINT J, C$
140 NEXT J
150 END
```

Programma 6-19.

Uitvoer op sommige microcomputers

```
RUN
1             1
2             1 2
3             1 2 3
4             1 2 3 4
5             1 2 3 4 5
6             1 2 3 4 5 6
7             1 2 3 4 5 6 7
8             1 2 3 4 5 6 7 8
9             1 2 3 4 5 6 7 8 9
10            1 2 3 4 5 6 7 8 9 1 '0
```

Uitvoer op BBC-computer

```
RUN
1             1
2             12
3             123
4             1234
5             12345
6             123456
7             1234567
8             12345678
9             123456789
10            12345678910
```

Dus, behalve op de BBC-computer kunnen we nog geen karakters op opeenvolgende plaatsen aan elkaar koppelen. De extra spatie wordt tussengevoegd als plaats voor het teken.

[K] Programma 6-19. BBC: regel 130 PRINT; J, C$

Oefening 4
Schrijf een programma om een willekeurig woord van het toetsenbord in te lezen, de afzonderlijke letters te coderen en de verkregen code af te drukken.
(tip: maak een lijst als uit programma 6-17 voor het gehele alfabet. Denk aan het DIM statement. Neem iedere letter van het woord en vergelijk deze met de lijst. Indien gevonden laat de index dan toegevoegd worden aan de codestring.

Oefening 5
Schrijf een programma dat 20 random 3-letter woorden genereert. (Het is interessant te zien hoe vaak u het programma moet herstarten voordat u een bestaand woord aantreft).

Opgave 6

Schrijf een programma dat een spel kaarten genereert. Bedenk zelf een geschikte codering voor de kaarten. (Bijvoorbeeld twee karakters; SA voor schoppen-aas). Vergeet niet dat éénzelfde kaart niet tweemaal voor mag komen.

tips:
- stel een set kaarten samen in een lijst;
- kies (met behulp van de RND generator 1-52);
- druk de kaarten af.

Als een kaart getrokken is, plaats dan een merkteken (b.v. '★') in de lijst zodat u weet wanneer een kaart niet meer beschikbaar is.

Samenvatting van hoofdstuk 6

Op dit punt in het boek aangekomen dient u de volgende onderwerpen in eenvoudige BASIC-programma's te kunnen toepassen.

- Het gebruik van het RND-statement voor het genereren van random getallen tussen nul en één.
- Het gebruik van INT met RND om random integers te genereren.
- Het simuleren van dobbelstenen.
- Het simuleren van het werpen van één en twee munten.
- Het toepassen van scorelijsten.
- Afkorten en van statements en programmadelen.
- Het gebruik van meer statements op een regel.
- Het koppelen van strings.
- Het gebruik van STR$(X).

Antwoorden op ZOP's en oefeningen

ZOP 1

```
10  RANDOMIZE
20  FOR I = 1 TO 6
30  LET N = 6★RND(1)
40  PRINT N
50  END
```

Programma 6-20.

ZOP 2

(a) 4; (b) 9; (c) −3; (d) −1; (e) 1.

ZOP 3

−3.8	−4
−3.6	−4
−3.4	−4
−3.2	−4
−3	−3
−2.8	−3
−2.6	−3
−2.6	−3
−2.4	−3
−2.2	−3
−2	−2

ZOP 4

1.6	1
1.8	1
2	2
2.2	2
2.4	2
2.6	2
2.8	2
3	3
3.2	3

Oefening 1

Beschrijf algorithme:

1. Start.
2. Maak de zes lijstelementen voor totaalscore gelijk aan nul.
3. Start de 100 worpen lus.
4. Genereer een random-getal uit de verzameling (1,2,3,4,5,6).
5. Verhoog de bijbehorende scoreteller.
6. Als lusteller < = 100 ga dannaar 4, ga anders naar 7.
7. Print score en frequentie.
8. Stop.

```
10  REM ★★ 100× DOBBELSTEEN ★★
15  DIM S(6)
20  FOR J = 1 TO 6
30  LET S(J) = 0
40  NEXT J
50  RANDOMIZE
60  FOR I = 1 TO 100
70  LET S = INT (6★RND(1) + 1)
80  LET S(S) = S(S) + 1
90  NEXT I
100 PRINT
110 PRINT "SCORE", "FREQUENTIE"
120 FOR K = 1 TO 6
130 PRINT K, S(K)
140 NEXT K
150 END
```

Programma 6-21.

```
RUN
SCORE        FREQUENTIE
1            16
2            13
3            23
4            16
5            18
6            14

RUN
SCORE        FREQUENTIE
1            16
2            13
3            16
4            17
5            15
6            23

RUN
SCORE        FREQUENTIE
1            23
2            14
3            14
4            15
5            12
6            22

RUN
SCORE        FREQUENTIE
1            14
2            19
```

```
3           21
4           14
5           20
6           12
```

K Programma 6-21. BBC: regel 50 laat; weg
regel 70 LET S = RND(6)
regel 130 PRINT;....

Oefening 2

```
10  REM ★★ GOOI 2 DOBBELSTENEN 100× ★★
20  DIM S(15)
30  FOR J = 2 TO 12
40  LET S(J) = 0
50  NEXT J
60  RANDOMIZE
70  FOR I = 1 TO 100
80  LET S1 = INT (6★RND(1)+1)
90  LET S2= INT (6★RND(1)+1)
100 LET S = S1 + S2
110 LET S(S) = S(S)+1
120 NEXT I
130 PRINT
140 PRINT "SCORE", "FREQUENTIE"
150 FOR K = 2 TO 12
160 PRINT K, S(K)
170 NEXT K
180 END
```

Programma 6-22.

```
RUN
SCORE       FREQUENTIE
2           9
3           5
4           8
5           11
6           11
7           13
8           11
9           11
10          14
11          4
12          3
```

K Programma 6-22.

BBC: regel 60 laat; weg
regel 80 LET S1 = RND(6)
regel 90 LET S2 = RND(6)
regel 160 PRINT;...

Wat dacht u van 1000 worpen? Verander regel 70 in:

```
70  FOR I = 1  TO  1000
```

en u krijgt ongeveer onderstaand resultaat:

```
RUN
SCORE          FREQUENTIE
2              24
3              56
4              80
5              103
6              149
7              170
8              125
9              110
10             86
11             73
12             24
```

Suggesties voor enkele variëteiten

1. Wat dacht u van het printen van een frequentiediagram? En hoe zou u het probleem oplossen van regels die buiten het scherm respectievelijk papier vallen?
2. Maak een algemeen schaalprogramma dat een variabele schaal toelaat en de regellengte optimaal benut.

Oefening 3

```
10  REM ★★ FREQUENTIEDIAGRAM ★★
20  DIM S(15)
30  FOR J = 2 TO 12
40  LET S(J) = 0
50  NEXT J
60  RANDOMIZE
70  FOR I = 1 TO 100
80  LET S1 = INT(6★RND(1)+1)
90  LET S2 = INT(6★RND(1)+1)
100 LET S = S1 + S2
110 LET S(S) = S(S) + 1
120 NEXT I
130 PRINT
140 PRINT "FREQUENTIEDIAGRAM"
150 PRINT
160 FOR K = 2 TO 12
170 PRINT K; TAB(5); S(K); TAB(10);
175 IF S(K) = 0 THEN 210
180 FOR L = 1 TO S(K)
190 PRINT "★";
200 NEXT L
210 PRINT
220 NEXT K
230 END
```

Programma 6-23.

RUN

```
FREQUENTIEDIAGRAM

2             1             ★
3             4             ★★★★
4             7             ★★★★★★★
5             11            ★★★★★★★★★★★
6             15            ★★★★★★★★★★★★★★★
7             16            ★★★★★★★★★★★★★★★★
8             19            ★★★★★★★★★★★★★★★★★★★
9             8             ★★★★★★★★
10            9             ★★★★★★★★★
11            7             ★★★★★★★
12            3             ★★★
```

K Programma 6-23.
BBC: regel 60 laat ; weg
: regel 80 INT(6★RND(1)+1)
: regel 90 INT(6★RND(1)+1)
: regel 170 PRINT; K....

ZOP 5
De volgende listing geeft slechts één van de vele mogelijkheden ter verkorting aan.

```
10  REM ★★ FREQUENTIEDIAGRAM ★★
20  DIM S(15)
30  FOR J = 2 TO 12: S(J) = 0: NEXT J
60  RANDOMIZE
70  FOR I = 1 TO 100
80  S1 = INT(6★RND(1)+1): S2 = INT(6★RND(1)+1)
100 S = S1 + S2: S(S) = S(S)+1
120 NEXT I
130 ?:? "FREQUENTIE DIAGRAM":?
160 FOR K = 2 TO 12
170 ?K; TAB(5); S(K); TAB(10);
175 IF S(K) = 0 THEN 210
180 FOR L = 1 TO S(K):?"★";:NEXT L
210 ?
220 NEXT K
230 END
RUN
```

Programma 6-24.

```
FREQUENTIEDIAGRAM
2             4             ★★★★
3             3             ★★★
4             3             ★★★
5             18            ★★★★★★★★★★★★★★★★★★
6             11            ★★★★★★★★★★★
7             13            ★★★★★★★★★★★★★
8             11            ★★★★★★★★★★★
```

```
9           13          ★★★★★★★★★★★★
10          9           ★★★★★★★★★
11          11          ★★★★★★★★★★★
12          4           ★★★★
```

ZOP 6

```
10  REM ★★ MEERVOUDEN ★★
20  LET A$ = "EN"
30  INPUT B$
40  PRINT B$ + A$
50  END
```

Oefening 4

```
10  REM ★★ CODEREN ★★
20  CLEAR 500
30  DIM A$(26)
40  FOR I = 1 TO 26
50  READ A$(I)
60  NEXT I
70  DATA A,B,C,D,E,F,G,H,I,J,K,L,M,N,
80  DATA O,P,Q,R,S,T,U,V,W,X,Y,Z
100 INPUT "VOLGENDE WOORD : "; W$
110 L = LEN (W$)
120 FOR J = 1 TO L
130 I = 1
140 IF MID$ (W$, J, 1) = A$(I) THEN 200
150 I = I + 1
160 GOTO 140
200 C$ = C$ + STR$(I) + " "
210 NEXT J
220 ?: ? C$
230 END
```

maak een lijst met het alfabet

vergelijk iedere letter uit W$ met elke letter uit het alfabet todat deze gelijk zijn

Programma 6-25.

```
RUN
VOLGENDE WOORD: BEREKENEN
2  5  19  5  11  5  14  5  14

RUN
VOLGENDE WOORD: PARLEMENT
16  1  19  12  5  13  5  14  21

RUN
VOLGENDE WOORD: PROFESSIONEEL
16  19  15  6  5  20  20  9  15  14  5  5  12
```

[K] Programma 6-25

Oefening 5

BBC: let op! Laat DIM A$(26) op regel 20 wel staan.

```
10  REM ★★ WILLEKEURIGE 3-LETTER WOORDEN ★★
20  CLEAR 100: DIM A$(26)
30  FOR I = 1 TO 26: REM ★★ ALFABETISCHE LIJST ★★
40  READ A$(I): NEXT I
50  DATA A,B,C,D,E,F,G,H,I,J,K,L,M
60  DATA N,O,P,Q,R,S,T,U,V,W,X,Y,Z
70  RANDOMIZE
80  INPUT "NIEUWE SERIE? (JA/NEE)"; R$
90  IF R$ < > "JA" THEN 190
100 FOR K = 1 TO 20: REM ★★ 20 WOORDEN IN EEN LIJST ★★
110 W$ = " ":REM ★★ LEGE STRING OM TE BEGINNEN ★★
120 FOR J = 1 TO 3 : REM ★★ START VAN EEN WOORD ★★
130 X = INT (26★RND(1)+1)
140 W$ = W$ + A$(X)
150 NEXT J: REM ★★ EINDE WOORD ★★
160 PRINT W$: REM ★★ PRINT WOORD ★★
170 NEXT K: REM ★★ NIEUW WOORD ★★
180 GOTO 80
190 END
```

Programma 6-26.

Teveel statements maken een programma verwarrend; voegen zij iets toe aan de leesbaarheid?

```
RUN
NIEUWE SERIE? (JA/NEE) JA
HAI
ASN
DVN
RGT
VZU
BZY
XGC
HAK
ZIL
DFL
TNZ
STJ
ZHL
JUT
RDH
PNX
FNH
LSR
NJQ
RDE
```

NIEUWE SERIE? (JA/NEE) NEE

Hoeveel maal moet u het programma 'runnen' voordat u een bestaand woord verkrijgt?

[K] Programma 6-26

7 Getallen

7.1 Introductie

We hebben al laten zien dat de computer meer kan dan alleen berekeningen uitvoeren, maar in dit hoofdstuk gaan we juist verder in op de rekenkundige capaciteiten van de computer. We gaan ons op eenvoudig recht-toe-recht-aan rekenwerk toeleggen dus erg moeilijk zal het niet worden. We zijn er zeker van dat u dit hoofdstuk zult kunnen volgen. We gaan door met de "kijk eens wat er gebeurt"-aanpak met behulp van uw computer.

7.2 Gemiddelde en rekenkundig gemiddelde

Hoe lang heeft u tot nu toe over elk hoofdstuk gedaan? Gemiddeld drie uur! Als we u hadden gevraagd hoeveel tijd u nodig heeft 's morgens naar uw werk te gaan had u op een overeenkomstige wijze geantwoord. Sportenthousiasten spreken over doelgemiddelden en slaggemiddelde. Atlassen geven een overzicht van de gemiddelde regenval per regio over de maanden van het jaar en we spreken over de gemiddelde leeftijd van de personen in een groep.

Als we het rekenkundig gemiddelde van de personen uit een groep willen berekenen, dan tellen we de afzonderlijke leeftijden bij elkaar op en delen dat door het aantal personen in de groep. De berekening van het rekenkundig gemiddelde omvat: optellen, tellen en deling van de som door het aantal.

Voorbeeld 1
Bereken het rekenkundig gemiddelde van de volgende reeks getallen:

6,7,2,5,4,4,9,8

Oplossing:
De som is: $6+7+2+5+4+4+9+8 = 45$.
Er zijn 8 getallen en hun rekenkundig gemiddelde is: $45/8 = 5.625$.

ZOP 1
Bereken het rekenkundig gemiddelde uit de volgende reeks getallen:

8,4,2,6,1,7,6,1,4

Voorbeeld 2
Ontwerp een algorithme en schrijf een programma ter berekening van het rekenkundig gemiddelde van de getallen uit de volgende DATA-statements.

```
900  DATA 56,47,52,65,24,34,59,37,49,66
910  DATA 38,24,62,76,31,47,66,61,74,45
920  DATA 66,44,55,67,36,56,54,54,50,43
930  DATA 18,83,23,79,29,-9999
```

Waarbij −9999 natuurlijk slechts dient ter afsluiting.

Oplossing:
We geven het algorithme eerst in beschrijvende vorm.

1. Start
2. Zet teller op 1.
3. Zet som op 0.
4. Voer het volgende getal in.
5. Als getal = −9999 ga dan naar 9, ga anders naar 6.
6. Tel getal bij som op.
7. Verhoog de teller.
8. Ga naar 4.
9. Maak de teller één kleiner.
10. Bereken gemiddelde = som/teller
11. Print: gemiddelde
12. Stop

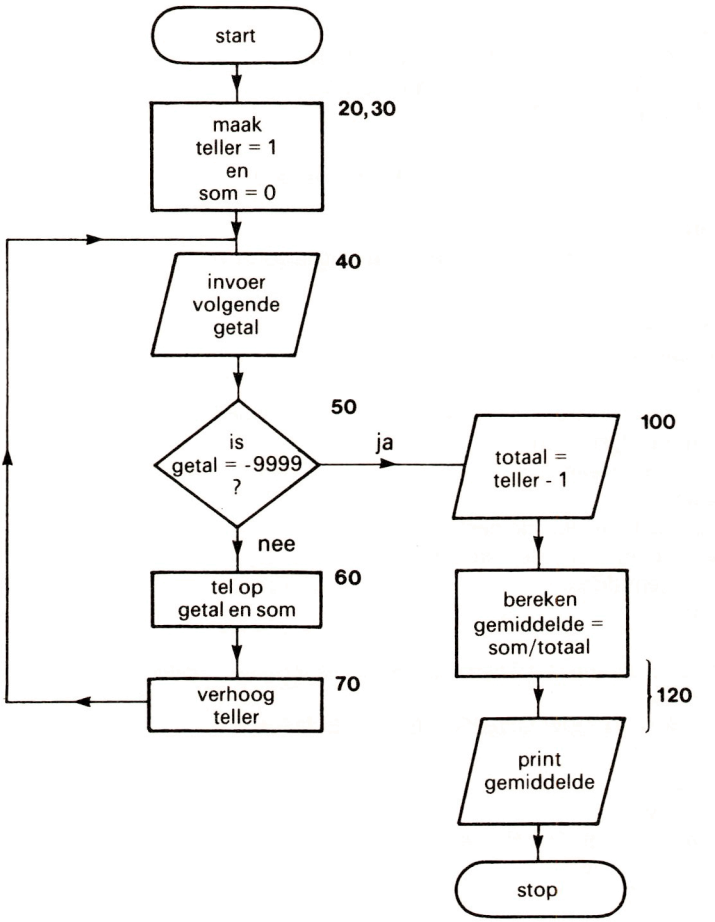
Afb. 7-1. Beschrijvend algorithme voor het rekenkundig gemiddelde.

```
10  REM ★★ REKENKUNDIG GEMIDDELDE ★★
20  LET C = 1
30  LET S = 0
40  READ M
50  IF M = -9999 THEN 100
60  LET S = S + M
70  LET C = C + 1
80  GOTO 40
90  REM ★★★★★★★★★★
100 LET N = C - 1
110 REM ★★★★★★★★★★
120 PRINT "GEMIDDELDE = "; S/N
130 REM ★★★★★★★★★★
140 END
900 DATA 56,47,52,65,24,34,59,37,49,66
910 DATA 38,24,62,76,31,47,66,61,74,45
920 DATA 66,44,55,67,36,56,54,54,50,43
930 DATA 18,83,23,79,29,-9999
```

Programma 7-1 Rekenkundig gemiddelde.

RUN
GEMIDDELDE = 50.5714286

K Programma 7-1



Afb. 7-2. Stroomdiagram voor het rekenkundig gemiddelde.

Oefening 1
Schrijf een programma om de gemiddelde woordlengte in het gedicht uit voorbeeld 2 van hoofdstuk 5 te bepalen. Daartoe kunt u gebruik maken van programma 5-1 voor het tellen van woordlengten.

Oefening 2
Simuleer 100 dobbelsteenworpen en bereken het gemiddelde van de worpen.

Simulatie:
De te verwachten gemiddelde score voor het groot aantal malen werpen van een dobbelsteen is:

$$\frac{1+2+3+4+5+6}{6} = \frac{21}{6} = 3.5$$

Als u echter het resultaat van oefening 2 bekijkt, zult u merken dat u maar zelden precies 3.5 zult meten. Pas bij zeer grote aantallen komt het resultaat bij 3.5 in de buurt. U zult zich afvragen of de random-getallengenerator wel goed is (we noemden deze al 'pseudo'). Hoeveel experimenten moeten we doen om ons te overtuigen dat deze wel goed is?

Om dat te weten te komen moeten we wat in de statistische theorie duiken en dat valt buiten het bestek van dit boek. Maar we kunnen wel met behulp van het bovenstaande programma het gemiddelde berekenen van 30 uitkomsten van elk 100 worpen. In de DATA-regels van programma 1 zijn de decimale gedeelten van 30 runs opgenomen. Dat mag, omdat alle getallen met 3. beginnen.

```
130  REM ★★★★★★★★★★
140  END
900  DATA 56,47,52,65,24,34,59,37,49,66
910  DATA 38,24,62,76,31,47,66,61,74,45
920  DATA 66,44,55,67,36,56,54,54,50,43
930  DATA -9999

RUN
GEMIDDELDE = 51.2666667
```

Afb. 7-3.

Het totale gemiddelde der decimalen is 51.2267 en dat staat voor een worpgemiddelde van 3.51226. Dat resultaat is al veel overtuigender als het voorafgaande.

Simulatie
Simulatie is een groot woord voor wat we zojuist hebben gedaan. Het is echter belangrijk te benadrukken dat we allerlei processen uit de werkelijkheid op dergelijke wijze kunnen nabootsen: simuleren. In de huiskamer beginnen we niet aan het gooien en administreren van 3000 dobbelsteenworpen: met de computer is dat zó gedaan.

Oefening 3
Schrijf een programma om de gemiddelde score te berekenen van honderd worpen met twee dobbelstenen.
Wat verwacht u ten aanzien van het gemiddelde resultaat als u met 3 of 4 ... dobbelstenen gaat werpen? Worden uw verwachtingen door de experimenten gestaafd?

7.3 Getalbereik

Tijdens het bespreken van de resultaten van oefening 2 in de vorige paragraaf spraken we al terloops over het bereik van getallen. We constateerden dat de uitkomsten lie-

pen van 3.24 tot 3.76. Het bepalen van het bereik omvat het bepalen van zowel de grootste als de kleinste waarde uit een verzameling.

Voorbeeld 3
Ontwerp een algorithme en schrijf een programma om de maximale en minimale waarde van de getallen uit de DATA-statements van voorbeeld 2 te bepalen. Voeg het programma aan dat uit voorbeeld 2 toe.

(Als u het aandurft, bekijk dit dan eerst als oefening voordat u naar de oplossing kijkt.)

Oplossing
U zult merken dat we dit probleem al eerder hebben opgelost in hoofdstuk 4 bij het bepalen van het kleinste lijstelement. Als we diezelfde aanpak opnieuw willen volgen zijn we echter verplicht alle data eerst in een lijst onder te brengen en vervolgens tweemaal het programma te doorlopen. Dat is veel werk en daarom volgen we nu een snellere aanpak: we proberen al tijdens het inlezen de grootste en de kleinste waarde te bepalen.

We weten al hoe data wordt ingelezen (regel 10-50 in programma 7-1), maar wat doen we met de ingelezen gegevens?

1. We creëren twee variabelen:
   M voor de laatste maximale waarde en
   B voor de laatste minimale waarde.

2. We lezen het eerste getal in en plaatsen dit in M én B. Tot hier toe is dit zowel het grootste als kleinste getal.

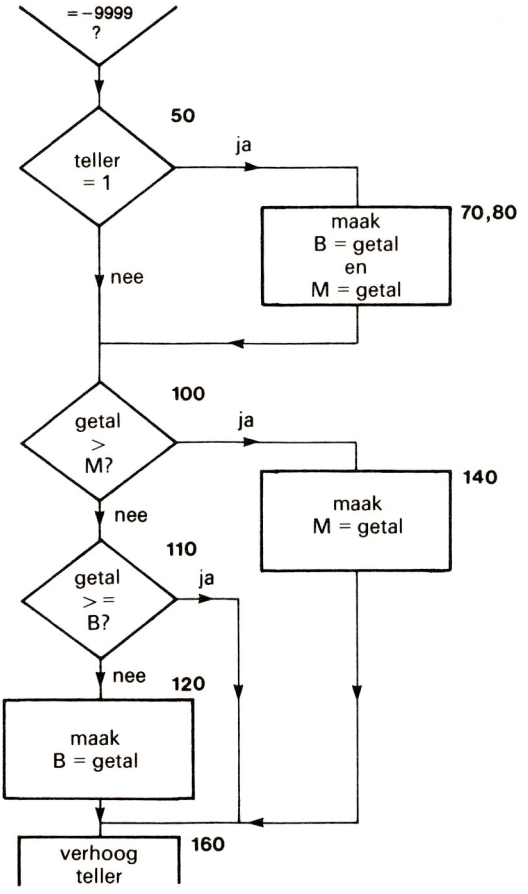
3. Dan lezen we het volgende getal en plaatsen dit in M als dit groter is dan M en in B als het kleiner is dan B.

Een beschrijvend algorithme is:

1. Start routine
2. Als teller = 1 plaats dan getal in M en B en ga naar 6, ga anders naar 3.
3. Als getal > M plaatsdan getal in M en ga naar 6, gaanders naar 4
4. Als getal < = B ga dan naar 6, ga anders naar 5.
5. Plaats getal in B
6. Einde routine.

Afb. 7-4.

In een stroomdiagram:



Afb. 7-5.

Welk van de twee beschrijvingen prefereert u?

Als er meer dan één sprong in het programma voorkomt kan het beschrijvend algorithme verwarrend zijn. Het twee-dimensionale plaatje van het stroomdiagram is dan duidelijker. Het is maar waar u de voorkeur aan geeft!

```
10  REM ★★ MAX EN MIN ★★
20  LET C = 1
25  LET S = 0
30  READ M
40  IF M = −9999 THEN 190
50  IF C > 1 THEN 100
60  REM ★★★★★★★★★★
```

```
70  LET B = M
80  LET T = M
90  REM ★★★★★★★★★★
100 IF M > T THEN 140
110 IF M > = B THEN 160
120 LET B = M
130 GOTO 160
140 LET T = M
150 REM ★★★★★★★★★★
160 LET C = C + 1
165 LET S = S + M
170 GOTO 30
180 REM ★★★★★★★★★★
190 PRINT "MAX = "; T, "MIN = "; B
200 REM ★★★★★★★★★★
210 LET N = C - 1
220 PRINT "GEMIDDELDE = ";S/N
230 REM ★★★★★★★★★★
900 DATA 56,47,52,65,24,34,59,37,49,66
910 DATA 38,24,62,76,31,47,66,61,74,45
920 DATA 66,44,55,67,36,56,54,54,50,43
930 DATA 18,83,23,79,29,-9999
```

het programma passeert dit deel slechts éénmaal; de eerste maal.

Hier wordt beslist!

Programma 7-2.

```
RUN
MAX = 83                MIN = 18
GEMIDDELDE = 50.5714286
```

[K] Programma 7-2

Oefening 4
Schrijf een programma om een frequentiediagram te maken van de data uit programma 7-2. Deel de getallen in categoriën in van elk 10 eenheden groot:

0-9, 10-19, 20-29,....90-99

Suggestie:
Gebruik een scorelijst van de vorm:

S(0), S(10), S(20),....S(100)

Deel elk getal bij die serie in, waarvoor geldt dat het getal groter of gelijk is aan de index, maar kleiner is dan de index van de volgende serie. Deze aanpak is ook toegepast in het antwoord.

7.4 Rekenkunde (aritmetiek)

Tot nu toe hebben we alle rekenkunde zoveel mogelijk vermeden, behalve waar het zeer eenvoudige berekeningen betrof. Daar zullen we verder mee door gaan. Maar het kan geen kwaad om ook eens een blik te werpen op de capaciteiten van de com-

puter op rekenkundig gebied. Als u de moed in de schoenen zinkt, dan gaat u gewoon verder met de volgende paragrafen, want u mist niets wezenlijks aan programmeertechniek. Echter, we hopen dat u er toch eens aan begint. De computer haalt in ieder geval het geestdodende werk uit de rekenkunde.

Hier is een eenvoudig programma dat voor de getallen 1–10 hun kwadraten, 3e machten en reciproken berekent.

```
20  REM ★★ BEREKEN DE KWADRATEN, 3E MACHTEN EN ★★
30  REM ★★ RECIPROKEN VOOR DE EERSTE 10 ★★
40  REM ★★ NATUURLIJKE GETALLEN ★★
50  PRINT "N","N★N","N★N★N","1/N"
60  PRINT
70  FOR I = 1 TO 10
80  LET N = I
90  LET S = I★I
100 LET C = I★I★I
110 LET R = 1/I
120 PRINT N,S,C,R
130 NEXT I
140 END
```

Programma 7-3.

```
RUN
N          N★N        N★N★N      1/N

1          1          1          1
2          4          8          0.5
3          9          27         0.333333333
4          16         64         0.25
5          25         125        0.2
6          36         216        0.166666667
7          49         343        0.142857143
8          64         512        0.125
9          81         729        0.111111111
10         100        1000       0.1
```

[K] Programma 7-3

```
BBC: regel 10 PRINT "N"; TAB(7); "N★N"; TAB(13); "N★N★N";
       TAB(21); "1/N"
     regel 80 PRINT; N; TAB(7); S; TAB(13); C; TAB(21); R
```

Machtsverheffen

We gaan er van uit dat u min of meer vertrouwd geraakt bent met de notatie:

$4 \times 4 = 4^2$ (vier in het kwadraat of vier tot de tweede macht)
$7 \times 7 \times 7 = 7^3$ (zeven tot de derde macht)
$10 \times 10 \times 10 \times 10 \times 10 = 10^5$ (tien tot de vijfde macht)

In BASIC gebruiken we voor het aangeven een extra symbool. Meestal wordt daarvoor het teken '↑' gebruikt. Een getal P tot de macht K noteert men in BASIC zo: P↑K

P heet de exponent en de techniek heet machtsverheffen. Negatieve machten worden op dezelfde manier behandeld.

	N	P	N↑P
$\frac{1}{4}\times\frac{1}{4}=\frac{1}{4\times4}=4^{-2}$	4	−2	4↑(−2)
$\frac{1}{7}\times\frac{1}{7}\times\frac{1}{7}=\frac{1}{7\times7\times7}=7^{-3}$	7	−3	7↑(−3)
$\frac{1}{10}\times\frac{1}{10}\times\frac{1}{10}\times\frac{1}{10}\times\frac{1}{10}\times\frac{1}{10}=\frac{1}{10\times10\times10\times10\times10}=10^{-5}$	10	−5	10↑(−5)

Afb. 7-6.

Gebroken machten zijn geen probleem, maar in dit boek zullen we ze verder niet gebruiken.
We kunnen de ↑-notatie gebruiken inplaats van de ★ in programma 7-3. Programma 7-3 aldus herschreven wordt:

```
10  PRINT "N", "N↑2", "N↑3", "N↑(−1)"
20  FOR N = 1 TO 10
30  PRINT N, N↑2, N↑3, N↑(−1)
40  NEXT N
50  END
```

Programma 7-4.

Reeksen en hun som
De individuele termen van een reeks bij elkaar optellen, dat is een aardige klus. Hoeveel tijd verwacht u nodig te hebben voor de uitwerking van:

$$\frac{1}{1^2},\ \frac{1}{2^2},\ \frac{1}{3^2},\ \cdots\ \frac{1}{N^2}\ ?$$

Met programma 7-5 gaat het heel gemakkelijk.

```
10  PRINT "N", "N↑(−2)"
20  FOR N = N TO 10
30  PRINT N, N↑(−2)
40  NEXT N
50  END
```

Programma 7-5.

```
RUN
N             N↑(−2)
1             1
2             0.25
3             0.111111111
4             6.25E−2
```

```
5          4E-2
6          2.77777778E-2
7          2.04081633E-2
8          1.5625E-2
9          1.2345679E-2
10         1E-2
```

(De notatie met E, die u zo hier en daar kunt tegenkomen wordt in paragraaf 7.6 nader toegelicht)

[K] Programma 7-5.

Oefening 5
Maak een programma dat uitrekent hoeveel termen van de volgende reeks bij elkaar moeten worden opgeteld voordat de som de 2.4 overschrijdt.

$$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$$

Oefening 6
Pas het programma uit oefening 5 zo aan dat het berekent wanneer de som van de volgende reeks 1.5 overschrijdt.

$$1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\ldots$$

Oefening 7
Faculteiten zijn interessante getallen. 4-faculteit wordt genoteerd als 4! en is: $4! = 4 \times 3 \times 2 \times 1$

$7\text{-faculteit} = 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 7!$
$N\text{-faculteit} = N \times (N-1) \times \ldots \times 2 \times 1 = N!$

Schrijf een programma dat de faculteit berekent voor positieve gehele getallen.

Oefening 8
We schreven al eerder een programma om de rente R over ons tegoed D bij een rentepercentage van P te berekenen.
Dat programma was niet erg geavanceerd. Een betere methode is met de volgende formule:

$$Y = D \times (1 + P/100)^T$$

waarin T het aantal jaren is.

Schrijf een programma waarin met behulp van deze formule, voor variabele tegoeden, percentages en looptijden de totale rente berekend kan worden.

7.5 'Droog-runnen'

Vaak blijkt dat een geschreven programma in het geheel niet werkt, of niet naar tevredenheid werkt. Als we tijd en geduld hebben kunnen we achter de computer blijven zitten tot we de fout gevonden hebben, maar als dat niet kan of lukt, blijft er niets anders over dan een pen en papier te pakken en heel hard na te denken. Het doorlopen van een algorithme met de hand wordt 'droog-runnen' genoemd.

We zullen dit proces illustreren aan de hand van programma 7-18, dat was de oplossing van oefening 5.

```
10  REM ★★ SOM DER RECIPROKEN ★★
20  LET S = 0
30  LET N = 1
40  LET S = S + 1/N
50  IF S > 2.4 THEN 80
60  LET N = N + 1
70  GOTO 40
80  PRINT "SOM = "; S; " HET AANTAL TERMEN IS: ";N
100 END
```

Programma 7-18 (oplossing van oefening 5).

```
RUN
SOM = 2.45   HET AANTAL TERMEN IS : 6
```

Met de hand gaan we nu alle programmaregels die de variabelen beïnvloeden nalopen, en schrijven de resultaten op in een tabel. Daarbij letten we goed op de variabele waarden te noteren nadat in gedachten de bewerking uitgevoerd is. Deze methode heeft een veel gebruikte Engelse benaming: 'tracing'

Stap	Regel	N	S
1			
2			
etc			

Afb. 7-7.

Alle regels die de variabele waarden niet beïnvloeden laten we uit de tabel weg, bijvoorbeeld:

```
10  REM...
70  GOTO...
80  PRINT...
```

De volgende tabel bevat alle bewerkingsstappen totdat de eindwaarde is bereikt.

Stap	Regel	N	S	
1	20	0	0	
2	30	1	0	
3	40	1	1	
4	50	1	1	
5	60	2	1	Let op het effect van GOTO 40 in regel 60
6	40	2	1.5	
7	50	2	1.5	
8	60	3	1.5	GOTO
9	40	3	1.83	
10	50	3	1.83	
11	60	4	1.83	GOTO
12	40	4	2.08	
13	50	4	2.08	
14	60	5	2.08	GOTO
15	40	5	2.28	
16	50	5	2.28	
17	60	6	2.28	GOTO
18	40	6	2.45	

Afb. 7-8.

Tracing

Veel BASIC's kennen een TRACE-commando, maar deze leveren vaak zoveel informatie tegelijk dat u door de bomen het bos niet meer ziet. Een zorgvuldig ontworpen TRACE-routine van uzelf werkt vaak nog het beste. We zullen u iets verder in dit hoofdstuk leren uw eigen TRACE-routine te schrijven.

ZOP 2

Maak een complete 'droog-run' van het volgende programma, compleet met regelnummers en variabelen, zoals in bovenstaande tabel aangegeven. Begin juist nadat regel 20 is uitgevoerd en stop nadat de conditie in regel 50 true is geworden.

```
10  REM ★★ SOM DER KWADRATEN ★★
20  S = 0
```

```
30  N = 1
40  S = S + N↑2
50  IF S > 50 THEN 80
60  N = N + 1
70  GOTO 40
80  PRINT "SOM = "; "HET AANTAL TERMEN IS: "; N
100 END
```

Programma 7-6.

7.6 De representatie van getallen

Tot nu toe hebben we ons nauwelijks bezig gehouden met de representatie van getallen in de computer. We zijn niet van plan ons daarmee bezig te gaan houden, maar iets meer kennis omtrent getallen in de computer is wel op z'n plaats. In het algemeen gebruiken we in de computer de volgende soorten getallen:

- positieve en negatieve gehele getallen (integers of gehele getallen);
- breuken en getallen welke een geheel en gebroken deel bezitten (reële getallen);
- zeer grote en zeer kleine getallen;
- het getal nul.

Het eerste belangrijke feit dat u dient te weten is dat getallen in de computer niet allemaal exact kunnen worden gerepresenteerd. Er is altijd een zekere mate van onnauwkeurigheid. Dit komt omdat de computer rekent in het binaire stelsel. De getallen die ons (in het tientallig) stelsel mooi voorkomen, zijn dat vaak in het binaire stelsel niet. Ter vergelijking: het getal 1/3, hetgeen in het decimale stelsel kan worden genoteerd als 0,33333... wordt daarin nooit exact gerepresenteerd; hoeveel drieën we ook toevoegen. In de praktijk is dat echter geen probleem omdat we voor elk probleem voldoende nauwkeurigheid kunnen bereiken. Het getal 1/10 kan in het decimale stelsel wel exact worden genoteerd nl. 0,1. In het binaire stelsel kan dit echter niet.

De meeste BASIC-interpreters kunnen getallen in 6 decimalen representeren. Het getal 3 1/10 in decimale vorm kan worden gerepresenteerd als 3.10000 in zes cijfers nauwkeurig. Als dit getal was ontstaan als resultaat van een berekening in de computer, dan was het resultaat waarschijnlijk 3.09999 of 3.10001 geweest. In het algemeen mag u stellen dat u nooit vertrouwen mag hebben in het laatste digit van een getal (het meest rechtse cijfer).

Programma 7-7 geeft een demonstratie van hoe deze onnauwkeurigheid ontstaat. De FOR...NEXT-lus telt 1000 maal de waarden 4.0, 4.1 en 4.25 in de variabelen S, T en U bij elkaar op. Nu kunnen 4.0 en 4.25 in binaire vorm exact gerepresenteerd worden en 4.1 kan dat niet. In het resultaat zien we daarom dat het antwoord voor 4.0 en 4.25 exact goed is terwijl de som van 1000 × 4.1 een afwijking vertoont van .00016. Het is lang niet altijd mogelijk dergelijke afwijkingen te vermijden, beschouwt u dit programma daarom als een waarschuwing.

```
10  REM ★★ NAUWKEURIGHEIDSDEMONSTRATIE ★★
20  LET S = 0
30  LET T = 0
```

```
40  LET U = 0
50  FOR I = 1 TO 1000
60  LET S = S + 4.0
70  LET T = T + 4.1
80  LET U = U + 4.25
90  NEXT I
100 PRINT S,T,U
120 END
```

BBC: 100 PRINT; S, T, U

Programma 7-7.

```
RUN
4000          4100.00016          4250
```

K Programma 7-7

Grote en kleine getallen

Zes decimale digits laten natuurlijk niet toe met zeer grote en zeer kleine getallen te rekenen. Daarom representeert BASIC deze in exponentiële vorm.

Kleine getallen

0.000586321 betekent $\frac{586321}{1.000.000.000} = \frac{586321}{10^9}$

Dit zou in exponentiële vorm kunnen worden gerepresenteerd door 586321×10^{-9}

We zouden ook kunnen zeggen dat 0.000586321 is $\frac{0.586321}{1000}$; hetgeen 0.586321×10^{-3} betekent.

In BASIC wordt de notatie met $\times 10^{-3}$ genoteerd als E−3 of E−03. Het getal hierboven wordt in BASIC genoteerd als:

0.586321E−3 of 0.586321E−03.

Overeenkomstig is:

$0.0234539 = 2.34539 \times 10^{-2} =$ 2.34539E−2

en

$0.00000000959734 = 0.959734 \times 10^{-8} =$ 0.959734E−8

E staat voor exponent van het grondtal 10. E−4 betekent dus dat het voorgaande getal door 10000 moet worden gedeeld (komma 4 plaatsen naar links) en E+9 betekent dat het getal met 1.000.000.000 moet worden vermenigvuldigd.

Grote getallen

$12368500 = 1.23685 \times 10^7 = 1.23685E+7$
$935.432 = 0.935432 \times 10^3 = 0.935432E+3$
$959734000000000000000 = 0.959734E+21$

De meeste BASIC interpreters kunnen 'floating point' getallen verwerken met exponenten tussen −32 en +32. Uw eigen systeem kan daarop natuurlijk een uitzondering vormen; zie daartoe de handleiding.

Bestudeer hoe uw computer grote en kleine getallen afdrukt met behulp van het volgende programma.

```
10  REM ★★ GETALLEN DEMONSTRATIE ★★
20  PRINT "GETAL", "REPRESENTATIE"
30  FOR I = -10 TO 10
40  PRINT "10↑";I,10↑I
50  NEXT I
100 END
```

Programma 7-8.

[K] Programma 7-8

7.7 De INT-functie en afronden

Vaak is het nodig de resultaten van een berekening af te ronden naar het dichtstbijzijnde gehele getal:

bijv. 6.6 afronden naar 7
7.4 afronden naar 7

Vooral als we niet zeker zijn van het laatste cijfer in een getal kan afronding belangrijk zijn.

bijv. 6.99999 of 7.00001 voor 7.

De functie INT(X+.5) kan deze afronding voor ons realiseren.

```
10  REM ★★ INT MET AFRONDING ★★
20  PRINT "X", "INT(X)", "INT(X + 0.5)"
30  FOR I = -1.4 TO -2.6 STEP (-.1)
40  PRINT I, INT(I), INT(I + .5)
50  NEXT I
60  END
```

Programma 7-9a.

```
RUN
X .          INT(X)        INT(X + 0.5)
-1.4         -2            -1
-1.5         -2            -1
-1.6         -2            -2
-1.7         -2            -2
-1.8         -2            -2
-1.9         -2            -2
-2           -2            -2
-2.1         -3            -2
-2.2         -3            -2
-2.3         -3            -2
-2.4         -3            -2
-2.5         -3            -2
-2.6         -3            -3
```

Veranderen we regel 30 in:

```
30  FOR I = 1.4 TO 2.6 STEP .1
```

dan krijgen we:

```
RUN
X           INT(X)      INT(X + 0.5)
1.4         1           1
1.5         1           2
1.6         1           2
1.7         1           2
1.8         1           2
1.9         1           2
2           2           2
2.1         2           2
2.2         2           2
2.3         2           2
2.4         2           2
2.5         2           3
2.6         2           3
```

Programma 7-9b.

[K] Programma 9a en 9b.
BBC: regel 40 PRINT; I,; INT(1); INT(1 + .5)

ZOP 3
Wat wordt het resultaat van het bovenstaande programma als we regel 30 veranderen in:

(a) 30 FOR I = .4 TO 2.2 STEP (.2)
(b) 30 FOR I = .4 TO −.6 STEP (−.2)

7.8 De ABS-functie

Een rekenkundige functie die te maken heeft met bovenstaande ideeën is de ABS-functie. De ABS-functie bepaalt de modulus, of absolute waarde van een getal. Dat klinkt ingewikkeld, maar het valt mee.

ABS(X) geeft ons de positieve waarde van X.

dus: ABS(23) = 23 en ABS(−23) = 23.

Het volgende programma demonstreert deze functie:

```
10  REM ★★ DE ABS-FUNCTIE ★★
20  PRINT "X", "Y", "X + Y", "ABS(X + Y)"
30  FOR I = 1 TO 4
40  READ X, Y
50  PRINT X,Y,X + Y, ABS(X + Y)
60  NEXT I
```

```
100 DATA 5,7,5,-7,-5,7,-5,-7
110 END
```

Programma 7-10.

```
RUN
X          Y          X + Y      ABS(X + Y)
5          7          12         12
5          -7         -2         2
-5         7          2          2
-5         -7         -12        12
```

K Programma 7-10. BBC: 50 PRINT; X,; Y,; X+Y,; ABS(X+Y)

ZOP 4
Geef aan wat de print-out van het vorige programma wordt als we regel 100 veranderen in:

```
100  DATA 9,14,11,-2,-4,13,-7,-8
```

7.9 Iteratie

Het mathematisch proces waarin we voor de oplossing van een probleem een waarde gissen, verifiëren en opnieuw (beter) gissen en opnieuw testen enz. totdat de oplossing voldoende nauwkeurig bekend is, wordt iteratie genoemd.
De essentie van itereren is:

- maak een willekeurige start.
- bepaal hoe nauwkeurig deze start was.
- verfijn de startwaarde in een reeks berekeningen.

Worteltrekken door iteratie
BASIC kan rechtstreeks worteltrekken met de functie SQR(X), zoals het volgende programma demonstreert:

```
10  FOR X = 33 TO 63 STEP(10)
20  PRINT X, X↑(.5), SQR(X)   BBC: 20 PRINT; X; TAB(5); X↑(.5), SQR(X).
30  NEXT X
40  END
```

Programma 7-11.

(SQR(X) geeft de wortel uit X als deze niet negatief is).

```
RUN
33          5.74456265      5.74456265
43          6.55743853      6.55743853
53          7.28010989      7.28010989
63          7.93725394      7.93725393
```

Toch willen we u demonstreren hoe u de wortel met een iteratief proces kunt berekenen; niet omdat dat beter gaat maar omdat het worteltrekken een geschikt demonstratieprogramma levert.

De methode
Als u de wortel wilt trekken uit N ga dan als volgt te werk:

- maak een gissing naar de wortel, bijv. G.
- bereken N/G.
- het gemiddelde van G en N/G is een betere benadering voor de wortel uit N, bepaal G = (G/N+G)/2.
- herhaal het proces totdat N/G − G kleiner is dan de gewenste nauwkeurigheid.

We zullen hiervoor geen bewijs leveren (dat u in veel wiskundeboeken kunt vinden) maar zullen dit in een eenvoudig voorbeeld aantonen:

Wortel uit N = 12
gissing G = 2
bereken N/G = 6
gemiddelde (6+2)/2 = 4

gissing G = 4 (tweede iteratiestap)
bereken N/G = 3
gemiddelde (4+3)/2 = 3.5

Nu wordt 3.5 een nieuwe benadering voor de wortel.

In tabelvorm kunt u één en ander zó handig noteren:

G	N/G	$\frac{G+N/G}{2}$
2	12/2=6	(2+6)/2=4
4	12/4=3	(4+3)/2=3.5
3.5	12/3.5=. . .	. . . .

Afb. 7-9.

ZOP 5
Om te verifiëren dat u het iteratieproces begrijpt, kunt u een tabel maken voor beginwaarde 1 in plaats van 2.

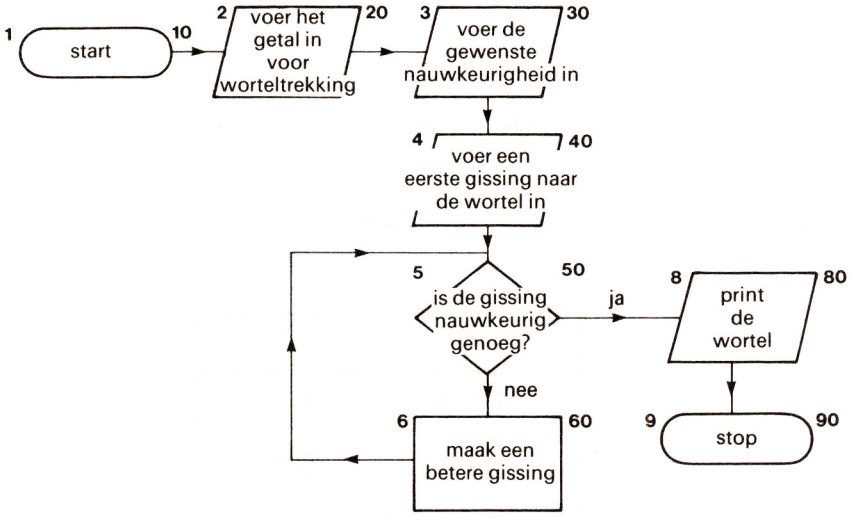
Nauwkeurigheid
De vraag is nu: 'wanneer is het iteratieproces ten einde?'.
Wel, dat is als het kwadraat van de iteratiewaarde zo dicht mogelijk bij N gekomen is. Daar iedere stap ons dichterbij de werkelijke wortel brengt (het proces 'convergeert') maar deze nooit echt bereikt, moeten we een keuze maken. Wij moeten bepalen welke afwijking voor ons acceptabel is. Als we bijvoorbeeld de wortel willen weten met een nauwkeurigheid van twee cijfers achter de komma dan moet het verschil tussen G★G en N kleiner zijn dan 0.005. Omdat we niet geïnteresseerd zijn in de polariteit van het verschil (of de wortel iets te groot of iets te klein is) moeten we de absolute waarde van dit verschil berekenen:

```
IF ABS(N-G★G) < 0.005 THEN einde iteratie
```

Beschrijvend algorithme:

1. Start.
2. Inlezen getal waaruit wortel getrokken moet worden.
3. Inlezen gewenste nauwkeurigheid.
4. Inlezen eerste gissing.
5. Als gissing < nauwkeurigheid ga dan naar 8, ga anders naar 6.
6. Bereken nieuwe gissing.
7. Ga naar 5.
8. Print gevonden waarde.
9. Stop.

Afb. 7-10.



Afb. 7-11. Stroomdiagram voor het worteltrekken-iteratie-proces.

```
10  REM ★★ WORTELTREKKEN DOOR ITERATIE ★★
20  INPUT "DE WORTEL UIT: "; N
30  INPUT "GEWENSTE ABS. NAUWKEURIGHEID: ";A
40  INPUT "UW BENADERING: "; G
50  IF ABS(N - G★G) < A THEN 80
60  LET G = .5★(G + (N/G))
70  GOTO 50
80  PRINT
81  PRINT "DE WORTEL UIT "; N; " IS "; G
90  END
```

Programma 7-12 Vierkantswortel iteratieprogramma.

```
RUN
DE WORTEL UIT : 12
GEWENSTE ABS. NAUWKEURIGHEID : .005
UW BENADERING : 2
```

```
DE WORTEL UIT 12 IS 3.46428571

RUN
DE WORTEL UIT : 12
GEWENSTE ABS. NAUWKEURIGHEID : .00005
UW BENADERING : 2

DE WORTEL UIT 12 IS 3.46410162
```

[K] Programma 7-12.

Tracen van het iteratieve proces

Het resultaat van programma 12 is niet direct opwindend. We kunnen de wortel immers ook rechtstreeks bepalen. Maar we wilden een eenvoudig voorbeeldprogramma om iteratie te demonstreren. We gaan het proces eens wat nader beschouwen met behulp van een eenvoudig trace-programmadeel:

```
47  PRINT "VORIG", "NIEUWE", "ABS(N-G★G)"
55  PRINT G
65  PRINT G, ABS (N-G★G)
```

Bij elke iteratieslag drukken bovenstaande regels de waarden van G, nieuwe G en de fout af. Zo heeft u een goed overzicht in het gehele proces.

```
10  REM ★★ WORTELTREKKEN DOOR ITERATIE ★★
20  INPUT "DE WORTEL UIT : ";N
30  INPUT "GEWENSTE ABS. NAUWKEURIGHEID :";A
40  INPUT "UW BENADERING : ";G
47  PRINT "OUD", "NIEUW", "ABS(N - G★G)"
50  IF ABS (N - G★G) < A THEN 80
55  PRINT G
60  LET G = .5★(G + (N/G)
65  PRINT G, ABS(N - G★G)
70  GOTO 50
80  PRINT
81  PRINT "DE WORTEL UIT"; N; "IS"; G
90  END
```

Programma 7-13 Iteratie met trace.

```
RUN
DE WORTEL UIT : 12
GEWENSTE ABS. NAUWKEURIGHEID :.005
UW BENADERING : 2
OUD             NIEUW           ABS(N - G★G)
2               4               4
4               3.5             0.25
3.5             3.46428571      1.27550587E-3

DE WORTEL UIT 12 IS 3.46428571

RUN
DE WORTEL UIT : 12
```

3 iteratieslagen volstaan hier

```
GEWENSTE ABS. NAUWKEURIGHEID : .0005
UW BENADERING : 2
OUD             NIEUW          ABS(N - G★G)
2               4              4
4               3.5            0.25
3.5             3.46428571     1.27550587E-3
3.46428571      3.464106162    3.35276127E-8

DE WORTEL UIT 12 IS 3.46410162

RUN
DE WORTEL UIT : -67
GEWENSTE ABS. NAUWKEURIGHEID : .005
UW BENADERING : 2
OUD             NIEUW          ABS(N - G★G)
2               -15.75         315.0625
-15.25          -5.74801587    100.039686
-5.74801587     2.9540901      75.7266483
2.9540901       -9.86316426    164.282009
-9.86316426     -1.5351062     69.356551
-1.5351062      21.0550414     510.314767
21.0550414      8.9364528      146.860189
8.9364528       0.719535419    67.5177312
0.719535419     -46.19801516   2201.25997
-46.1980516     -22.373887     567.590819
-22.373887      -1.38753833    68.9252626
-1.38753833     23.449708      616.888807
23.449708       10.2962648     173.013069
```

10-maal nauwkeuriger kan in 4 slagen

als we de wortel willen bepalen uit een negatief getal

komt er geen zinnig resultaat en eindigt het programma alléén met ESCAPE

[K] Programma 7-13 BBC:
```
55 PRINT; G; TAB(12);
65 PRINT; G; TAB(25); ABS (N-G★G)
```

Oefening 9

Pas het iteratieprogramma aan zodat u er de derde-machtswortel uit een getal mee kunt trekken. Als G de gissing is dan is .5★(G + N/(G★G)) een betere benadering.

Opgave 7

1. Als G een gissing is naar de oplossing van de vergelijking X★X + B★X + C = 0 dan wordt een betere benadering gevonden met −C/(B+G). Ontwerp een algorithme en schrijf een programma dat met iteratie een oplossing vindt voor elke kwadratische vergelijking.

Opmerking:
- De oplossing van een vergelijking is die waarde van X waarvoor het linkerlid gelijk is aan nul.
- Kies de waarden van B en C zodanig dat B★B > 4★C

2. De stelling van Pythagoras gaat over rechthoekige driehoeken. Er zijn echter enkele speciale driehoeken waarvoor alle zijden een gehele lengte hebben:

$3^2 + 4^2 = 5^2$
$5^2 + 12^2 = 13^2$

Ontwerp een algorithme en schrijf een programma dat uitrekent hoeveel van dergelijke driehoeken er zijn voor zijden kleiner dan 100.
TIP: U heeft hierbij de afrondingsfunctie INT(X + .5) nodig.

Samenvatting van hoofdstuk 7

Aan het eind van hoofdstuk 7 aangekomen dient u de volgende onderwerpen te beheersen en in eenvoudige BASIC-programma's te kunnen toepassen:

- Berekenen van het rekenkundig gemiddelde;
- Idem met behulp van een BASIC-programma;
- Schrijven van een programma om het grootste en kleinste getal uit een DATA-lijst te bepalen;
- Het gebruik van ★, / en ↑ in programma's
- 'Droog runnen' in tabelvorm van een programma;
- Interpreteren van E-notatie;
- Gebruiken van INT(X + .5) ter afronding van X;
- Gebruiken van ABS(X);
- Het ontwerpen van iteratieve programma's inclusief het beëindigingscriterium;
- Het invoegen van geschikte 'trace-regels' in een programma.

Antwoorden op ZOP's en oefeningen

ZOP 1
SOM = 8 + 4 + 2 + 6 + 1 + 7 + 6 + 1 + 4 = 39
Er zijn 9 getallen.
Het rekenkundige gemiddelde is: 39/9 = 4.3333...

Oefening 1

```
10  REM ★★ GEMIDDELDE WOORDLENGTE ★★
20  LET S = 0
30  LET C = 1
40  READ W$
50  IF W$ = "ZZZZ" THEN 160
60  LET L = LEN(W$)
70  LET S = S + L
80  LET C = C + 1
90  GOTO 40
100 REM ★★★★★★★★★★★★★★★★★★
110 DATA ALS, EEN, LOSGELATEN, PIJL
120 DATA ERGENS, IN, EEN, GROTE, STAD, BELAND
```

```
130 DATA TUSSEN, AUTO'S, EN, FABRIEKEN
140 DATA TUSSEN, SNELWEGEN, EN, FLATGEBOUWEN, ZZZZ
150 REM ★★★★★★★★★★★★★★★★★★★
160 LET N = C - 1
170 LET A = S/N
180 PRINT "DE GEMIDDELDE WOORDLENGTE IS "; A
190 END
```

Programma 7-14.

```
RUN
DE GEMIDDELDE WOORDLENGTE IS 5.5
```

[K] Programma 7-14

Oefening 2

```
10  REM ★★ GEMIDDELDE UIT 100 WORPEN ★★
20  REM ★★ MET EEN DOBBELSTEEN ★★
30  RANDOMIZE
50  LET S = 0
60  FOR I = 1 TO 100
70  LET X = INT(6★RND(1)+1)
80  LET S = S + X
90  NEXT I
100 PRINT "GEMIDDELDE SCORE = "; S/100
110 END
```

Programma 7-15.

[K] Programma 7-15 BBC: regel 70 RND(6) gebruiken

Oefening 3

```
10  REM ★★ GEMIDDELDE UIT 100 WORPEN ★★
20  REM ★★ MET TWEE DOBBELSTENEN ★★
50  LET S = 0
60  FOR I = 1 TO 100
70  LET X = INT(6★RND(1)+1)
75  LET Y = INT(6★RND(1)+1)
80  LET S = S + X + Y
90  NEXT I
100 PRINT "GEMIDDELDE SCORE = "; S/100
110 END
```

Programma 7-16.

[K] Programma 7-16 BBC: RND(6) gebruiken

Oefening 4

```
10  REM ★★ HISTOGRAM ★★
20  DIM S(100)
30  FOR K = 0 TO 100 STEP(10) : S(K) = 0 : NEXT K
40  REM ★★★★★★★★★★
50  READ M
60  IF M = -9999 THEN 150
70  REM ★★★★★★★★★★
```

```
80  K = 10
90  IF M < K THEN 120
100 K = K + 10
110 GOTO 90
120 S(K-10) = S(K-10)+1
130 GOTO 50
140 REM **********
150 FOR K = 0 TO 100 STEP (10)
160 PRINT K, S(K)
170 NEXT K
180 REM **********
190 DATA 56,47,52,65,24,34,59,37,49,66
200 DATA 38,24,62,76,31,47,66,61,74,45
210 DATA 66,44,55,67,36,56,54,54,50,43
220 DATA 18,83,23,79,29,-9999
230 END
```

Programma 7-17.

```
RUN
0           0
10          1
20          4
30          5
40          6
50          8
60          7
70          3
80          1
90          0
100         0
```

[K] Programma 7-17

Oefening 5

```
10  REM ** SOM DER RECIPROKEN / **
20  LET S = 0
30  LET N = 1
40  LET S = S + 1/N
50  IF S > 2.4 THEN 90
60  LET N = N + 1
70  GOTO 40
90  PRINT "SOM = "; S; "HET AANTAL TERMEN IS: ";N
100 END
```

Programma 7-18.

```
RUN
SOM = 2.45 HET AANTAL TERMEN IS : 6
```

[K] Programma 7-18

Oefening 6

```
10  REM ★★ MACHTREEKSONTWIKKELING ★★
20  LET S = 0
30  LET N = 1
40  LET S = S + N↑(-2)
50  IF S > 1.5 THEN 90
60  LET N = N + 1
70  GOTO 40
90  PRINT "SOM = "; "HET AANTAL TERMEN IS : ";N
100 END
```

Programma 7-19.

```
RUN
SOM = 1.51179705  HET AANTAL TERMEN IS :  7
```

U kunt natuurlijk regel 50 aanpassen om het aantal termen te bepalen dat voor andere eindwaarden nodig is.

```
50  IF S > 1.6 THEN 90
RUN
SOM = 1.60049693  HET AANTAL TERMEN IS:  22          voor 1,60

50  IF S > 1.61 THEN 90
RUN
SOM = 1.61103901  HET AANTAL TERMEN IS:  29          1,61

50  IF S > 1.62 THEN 90
RUN
SOM = 1.62024396  HET AANTAL TERMEN IS:  40          1,62

50  IF S > 1.63 THEN 90
RUN
SOM = 1.63011952  HET AANTAL TERMEN IS:  67          1,63
```

Dit zou een goed startpunt zijn voor een studie met betrekking tot de convergentie en limieten van reeksen.

[K] Programma 7-19.

Oefening 7

```
10  REM ★★ FACULTEITEN ★★
20  PRINT "RUN BEEINDIGEN MET -9999"
30  PRINT
40  INPUT "VOLGENDE FACULTEIT"; N
50  IF N = -9999 THEN 130
60  LET F = 1
70  FOR I = 1 TO N
80  LET F = F★I
90  NEXT I
100 PRINT N, F
110 PRINT
```

```
120 GOTO 40
130 END
```

Programma 7-20.

```
RUN
RUN BEEINDIGEN MET -9999

VOLGENDE FACULTEIT 1
1           1

VOLGENDE FACULTEIT 3
3           6

VOLGENDE FACULTEIT 5
5           120

VOLGENDE FACULTEIT 7
7           5040

VOLGENDE FACULTEIT 9
9           362880

VOLGENDE FACULTEIT 11
11          39916800

VOLGENDE FACULTEIT -9999
```

[K] Programma 7-20.

Oefening 8

```
10  REM ★★ SAMENGESTELDE RENTE ★★
20  PRINT "VOER IN: TEGOED, RENTE, LOOPTIJD IN JAREN"
30  INPUT D, P, T
40  PRINT "JAAR", "OPBRENGST"
50  FOR I = 1 TO T
60  PRINT I, D★(1 + P/100)↑I
70  NEXT I
80  END
```

Programma 7-21.

```
RUN
VOER IN: TEGOED, RENTE, LOOPTIJD IN JAREN
? 500, 11.5, 5
JAAR            OPBRENGST
1               557.5
2               621.6125
3               693.097937
4               772.804199
5               861.676682
```

[K] Programma 7-21.

ZOP 2

stap	regel	N	S
1	20	0	0
2	30	1	0
3	40	1	1
4	50	1	1
5	60	2	1
6	40	2	5
7	50	2	5
8	60	3	5
9	40	3	14
10	50	3	14
11	60	4	14
12	40	4	30
13	50	4	30
14	60	5	30
15	40	5	55

ZOP 3

(a)

X	INT(X)	INT(X+.5)
.4	0	0
.6	0	1
.8	0	1
1.0	1	1
1.2	1	1
1.4	1	1
1.6	1	2
1.8	1	2
2.0	2	2

(b)

X	INT(X)	INT(X+.5)
.2	0	0
0	0	0
−.2	−1	0
−.4	−1	0
−.6	−1	−1
−.8	−1	−1

ZOP 4

```
RUN
X          Y          X+Y        ABS(X+Y)
9          14         23         23
11         -2         9          9
-4         13         9          9
-7         -8         -15        15
```

ZOP 5

9	N/G	$\frac{G+N/G}{2}$
1	12	6.5
6.5	1.8	4.15
4.15	2.89	3.52
3.52	3.41	3.46 etc

Oefening 9

```
10  REM ★★ 3E MACHT WORTEL ITERATIE ★★
20  INPUT "DE 3E-MACHT UIT : ";N
30  INPUT "GEWENSTE ABS. NAUWKEURIGHEID: ";A
40  LET G = N/2
43  LET C = 1
50  IF ABS(N - G★G★G) < A THEN 80
60  LET G = 0,5★(G + N-/(G★G))
67  LET C = C + 1
70  GOTO 50
80  PRINT "HET AANTAL ITERATIESTAPPEN = ";C
81  PRINT "DE 3E-MACHT WORTEL UIT "; N;"= ";G
90  END
```

Programma 7-22.

```
RUN
DE 3E-MACHT UIT : 28
GEWENSTE ABS. NAUWKEURIGHEID : .0005
HET AANTAL ITERATIESTAPPEN = 18
DE 3E-MACHT WORTEL UIT 28 = 3.03657776

RUN
DE 3E-MACHT UIT : 10101
GEWENSTE ABS. NAUWKEURIGHEID : .005
HET AANTAL ITERATIESTAPPEN = 28
DE 3E-MACHT WORTEL UIT 10101 = 21.616634
```

```
RUN
DE 3E-MACHT UIT : -937
GEWENSTE ABS. NAUWKEURIGHEID : .005
HET AANTAL ITERATIESTAPPEN = 21
DE 3E-MACHT WORTEL UIT -937 = -978541982
```

[K] Programma 7-22.

8 Een introductie in data-processing

8.1 Introductie

In dit hoofdstuk gaan we opnieuw in op het belang van een ordening in data-gegevens. Met name één bepaalde methode van ordening, het sorteren, en de verwisselmethoden die daarbij behoren. Als de data eenmaal gesorteerd is, gaan we in op zoekprocedures om een gesorteerd element te kunnen terugvinden. Tot slot behandelen we data in tabelvorm.

Deze technieken geven ons meteen de kans om de subroutine als universeel instrument voor het oplossen van herhaalde structuren te introduceren.

8.2 Sorteren

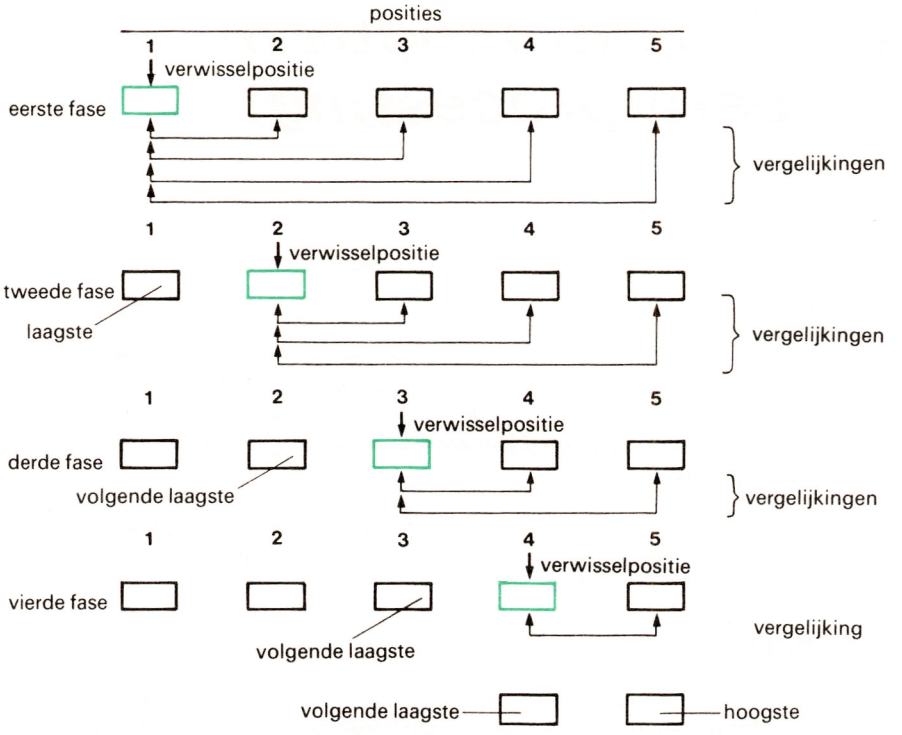
In hoofdstuk 4 hebben we aandacht besteed aan het vinden van de laagste waarde in een lijst. Dat deden we met behulp van een verwisselprocedure die het gevonden element verwisselde met het eerste op de lijst, afhankelijk van het resultaat van een vergelijkende test.
We stelden dat we de gehele lijst zo konden sorteren door achtereenvolgens alle lijstelementen te vergelijken met respectievelijk de elementen op positie 2, 3,..... N.
Omdat de verwisselroutines in sorteerroutines zo belangrijk zijn, bekijken we deze nu wat nader.

Afb. 8-1 laat de procedure zien voor het sorteren naar oplopende grootte in de volgorde 1 tot 5.

Eerste fase:
In de eerste fase worden alle elementen vergeleken met het element in positie 1 en verwisseld indien kleiner. Positie 1 bevat nu het kleinste element.

Tweede fase:
De tweede fase negeert positie 1 en vergelijkt alle elementen met het element in positie 2. De elementen worden verwisseld indien het gevonden element kleiner is dan dat in positie 2. Dit bevat nu het op één na kleinste element.

Derde fase:
De derde fase begint op positie 3 omdat de posities 1 en 2 respectievelijk de laagste en de op één na laagste waarde bevatten.



Afb. 8-1. Sorteerprocedure voor 5 elementen.

Vierde fase:
De vierde fase heeft alleen betrekking op de posities 4 en 5 en na afloop is element 5 automatische het hoogste van de totale lijst.

De sorteerprocedure kan als volgt worden samengevat:

fase	verwissel-positie	resterende elementen	
		begin	einde
1	1	2	5
2	2	3	5
3	3	4	5
4	4	5	5

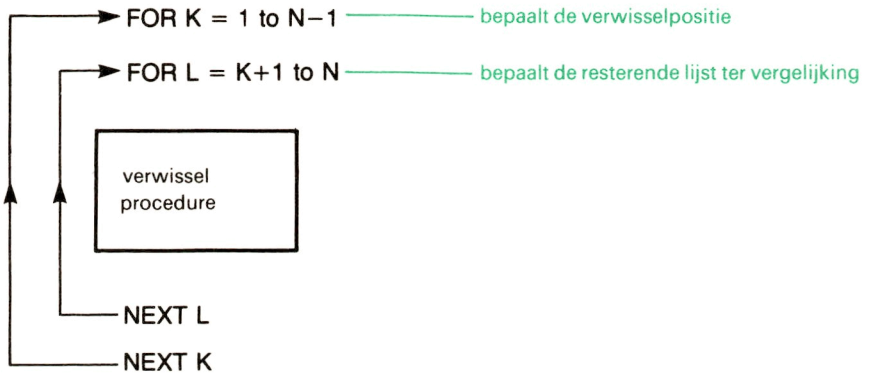
Afb. 8-2. Vier sorteerfasen in een 5-elementslijst.

Of in het algemeen:

fase	verwissel-positie	resterende elementen	
		begin	einde
1	1	2	N
2	2	3	.
3	3	4	.
.	.	.	.
.	.	.	N
.	.	.	.
K	K	K+1	.
.	.	.	.
.	.	.	N
.	.	.	N
N−1	N−1	N	N

Afb. 8-3. (n-1) sorteerfasen in een n-elementslijst.

Elke fase bestaat uit een reeks herhaalde bewerkingen: dat vráágt om een FOR-...NEXT-lus. Het sorteerproces bestaat uit (n–1)fases: dat vraagt om nog een FOR-...NEXT-lus.



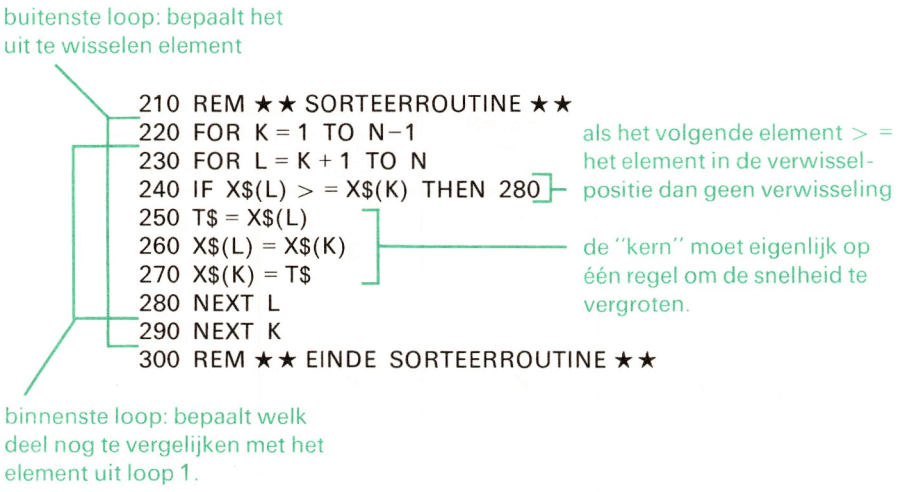
Afb. 8-4. Geneste lussen voor het verwissel-sorteer-probleem.

ZOP 1

Gebruik de bovenstaande aanpak om de volgende reeks getallen in volgorde van grootte te plaatsen. Laat de reeks zien na iedere sorteerfase.

6,1,4,0,2,3,7,8

Het programma is:



Programma 8-10 Het verwissel-sorteerprogramma.

Het gebruik van het sorteerprogramma

Dit sorteerprogramma kan overal toegepast worden waar iets gesorteerd moet worden. Hier is een bepaalde toepassing: het sorteren van een lijst met namen op alfabetische volgorde.

regel 50-80: lees data in
regel 210-300: sorteerproces
regel 410-450: print gesorteerde lijst.
De data bevindt zich in regel 900

```
10  REM ★★ SORTEERROUTINE ★★
20  CLEAR 100
30  DIM X$(100)
50  I = 1
60  READ X$
70  IF X$ = "ZZZZ" THEN 190
80  X$(I) = X$ : I + 1 : GOTO 60
180 REM ★★★★★★★★★★
190 N = I - 1 : REM ★ LENGTE LIJST ★
200 REM ★★★★★★★★★★
210 REM ★★ SORTEERROUTINE ★★
220 FOR K = 1 TO N-1
230 FOR L = K + 1 TO N
240 IF X$(L) > = X$(K) THEN 280
250 T$ = X$(L)
260 X$(L) = X$(K)
270 X$(K) = T$
280 NEXT L
290 NEXT K
```

inlezen data

sorteerroutines

```
300 REM ★★ EINDE SORTEERROUTINE ★★
400 REM ★★★★★★★★★★
410 PRINT "GESORTEERDE LIJST"
420 FOR P = 1 TO N
430 PRINT X$(P); TAB(6★P);
440 NEXT P
450 PRINT
500 REM ★★★★★★★★★★
900 DATA JAN, PIET, BERT, TOM, KEES, ZZZZ
910 END
```

print het resultaat

Programma 8-2 Toepassing voor een sorteerprogramma.

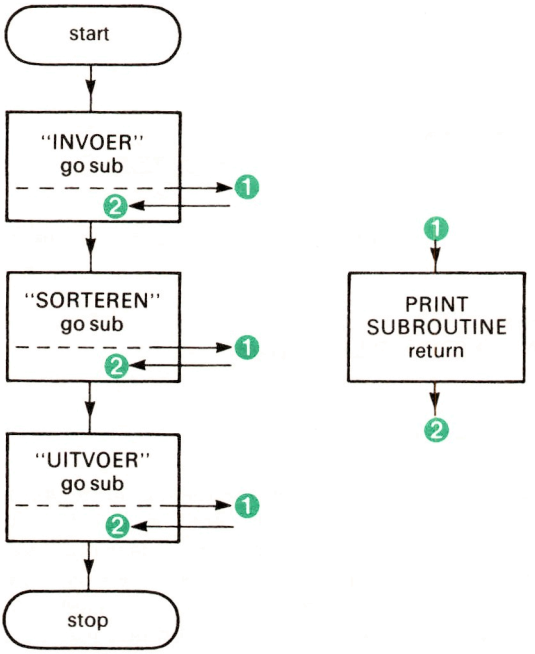
```
RUN
GESORTEERDE LIJST
BERT   JAN   KEES   PIET    TOM
```

8.3 Subroutines

Als u zo ver gekomen bent dat u het programma uit de vorige paragraaf begrepen en geanalyseerd heeft, zult u onderhand 'door het bos de bomen beginnen te zien'. U zult hebben opgemerkt dat programma's uit deelstructuren zijn opgebouwd tot een groter geheel, net als de paragrafen uit een boek. Het is een goed gebruik een programma in gedeelten op te splitsen, te schrijven en te testen. Bepaalde gedeelten vindt u door het gehele programma terug. De lengte van het programma en de structuur ervan kunnen worden verbeterd door deze gedeelten in een 'subroutine' op te nemen. Daartoe moet een dergelijk deel steeds betrekking hebben op dezelfde variabelen en deze op dezelfde manier bewerken.
Het gebruik van subroutines zullen we aan de hand van het sorteerprogramma nader beschouwen. We voegen twee extra Trace-regels toe om het sorteerproces te verduidelijken:

Sorteerroutine	**Trace-regel**
1. Invoer	De ingevoerde lijst
2. Sorteren	De lijst na iedere sorteerfase
3. Uitvoer	De uiteindelijke, gesorteerde lijst

Afb. 8-5 laat de structuur van het programma zien met de onderverdeling en het gebruik van een subroutine voor de PRINT-functie:



Afb. 8-5. Print-subroutine in sorteerprogramma's.

GOSUB

Om in BASIC een subroutine aan te roepen gebruiken we het statement:

GOSUB

gevolgd door het nummer van de regel waar de subroutine zich bevindt. Elke subroutine eindigt met het statement:

RETURN

zonder argument. Return zorgt ervoor dat het programma hervat wordt op de regel volgend op die waarin het desbetreffende GOSUB-statement stond. In het volgende programmasegment wordt in regel 30 de subroutine op regel 200 aangeroepen. Na de uitvoering van de regels 200 en 210 doet regel 220 het programma terugkeren zodat regel 40 uitgevoerd gaat worden.

```
10  INPUT A
20  INPUT B
30  GOSUB 200
40  S = A + B
50  .....

200 PRINT "A", "B"
210 PRINT A, B
220 RETURN
```

"SUBROUTINE" op regel 200 e.v.

ZOP 2

Wat is de waarde van B nadat het volgende programma is uitgevoerd? : (a) met A = 5 of (b) met A = 3 als aanvangswaarde.

```
10  INPUT A
20  IF A < 5 THEN 40
30  GOSUB 70
40  B = A★A
50  PRINT B
60  END
70  A = 1/A
80  RETURN
```

Programma 8-3.

Hieronder volgt een sorteerprogramma met daaraan een print-routine toegevoegd. Deze wordt aangeroepen vanuit de regels 194, 280 en 420.

```
10  REM ★★ SORTEERROUTINE ★★
20  CLEAR 100
30  DIM X$(100)
50  I = 1
60  READ X$
70  IF X$ = "ZZZZ" THEN 190
80  X$(I) = X$ : I = I + 1 : GOTO 60
180 REMREM ★★★★★★★★★★
190 N = I - 1 : REM ★LENGTE LIJST★
192 PRINT "LIJST VOOR SORTEREN"
194 GOSUB 510 ------------------> heen
196 PRINT <------------------ terug
200 REM ★★★★★★★★★★
210 REM ★★ SORTEERROUTINE ★★
220 FOR K = 1 TO N-1
225 PRINT "CYCLUS NO. : "; K
230 FOR L = K + 1 TO N
240 IF X$(L) > = X$(K) THEN 280
250 T$ = X$(L) : X$(L) = X$(K) : X$(K) = T$
280 GOSUB 510 ------------------> heen
285 NEXT L <------------------ terug
287 PRINT
```

eerste Trace

tweede Trace

```
290 NEXT K
300 REM ★★ EINDE SORTEERROUTINE ★★
400 REM ★★★★★★★★★★
410 PRINT "GESORTEERDE LIJST"
420 GOSUB 510 ──────────→ heen
450 END ←──────────────── terug
500 REM ★★ PRINT SUBROUTINE ★★
510 FOR P = 1 TO N
520 PRINT X$(P); TAB(6★P);
530 NEXT P
540 PRINT
550 RETURN
900 DATA JAN, PIET, BERT, TOM, KEES, ZZZZ
```

derde Trace

subroutine

Programma 8-4 Print-subroutine ter illustratie van het sorteerprogramma.

```
RUN
LIJST VOOR SORTEREN
JAN     PIET    BERT    TOM     KEES

CYCLUS NO. : 1
JAN     PIET    BERT    TOM     KEES
BERT    PIET    JAN     TOM     KEES
BERT    PIET    JAN     TOM     KEES
BERT    PIET    JAN     TOM     KEES

CYCLUS NO. : 2
BERT    JAN     PIET    TOM     KEES
BERT    JAN     PIET    TOM     KEES
BERT    JAN     PIET    TOM     KEES

CYCLUS NO. : 3
BERT    JAN     PIET    TOM     KEES
BERT    JAN     KEES    TOM     PIET

CYCLUS NO. : 4
BERT    JAN     KEES    PIET    TOM

GESORTEERDE LIJST
BERT    JAN     KEES    PIET    TOM
```

afgedrukt door GOSUB op regel 194

afgedrukt door GOSUB op regel 280, telkens nadat het programma de lus heeft uitgevoerd (gecontroleerd door K)

afgedrukt door GOSUB op regel 420

Voorbeelden met subroutines

Het doel van een subroutine is het vereenvoudigen en verduidelijken van een programma. Daardoor is het vaak moeilijk om een eenvoudig programma te construeren ter demonstratie van het gebruik van subroutines zonder dat dit wat tegenstrijdig aandoet. We hebben een programma nodig dat dezelfde dingen vaak uitvoert op verschillende punten in het hoofdprogramma.

Voorbeeld 1

Een spelletje met dobbelstenen geeft een goed voorbeeld: Twee dobbelstenen worden geworpen. Als er een 'dubbel' optreedt is het spel afgelopen. Is dit niet het geval dan wordt er opnieuw geworpen. Schrijf nu een programma dat het aantal worpen telt dat nodig is om een dubbel te gooien en tevens aangeeft welk dubbel gegooid is.

```
10  REM ★★ TWEE GELIJKE WORPEN ★★
20  RANDOMIZE
30  C = 1
40  GOSUB 130 : REM ★★ EERSTE WORP ★★
50  S1 = S
60  GOSUB 130 : REM ★★ TWEEDE WORP ★★
70  S2 = S
80  IF S1 = S2 THEN 100
90  C = C + 1 : GOTO 40
100 PRINT "DUBBEL "; S1; " IN "; C;" WORPEN"
110 END
120 REM ★★ DOBBELSTEEN SUBROUTINE ★★
130 D1 = INT(6★RND(1)+1):D2 = INT(6★RND(1)+1)
140 S = D1 + D2: RETURN
```

het resultaat S is uit de subroutine wordt in het hoofdprogramma gebruikt

subroutine

Programma 8-5.

```
RUN
DUBBEL 8 IN 8 WORPEN
RUN
DUBBEL 5 IN 11 WORPEN
RUN
DUBBEL 9 IN 5 WORPEN
RUN
DUBBEL 6 IN 1 WORPEN
RUN
DUBBEL 7 IN 3 WORPEN
RUN
DUBBEL 6 IN 2 WORPEN
```

Oefening 1

(a) Schrijf een programmasegment dat een regel met 40 streepjes ('----') kan afdrukken op het scherm.

(b) Schrijf een programmaregel die een 'onderzeeër' (< = >) afdrukt op een willekeurige positie op de regel.

(c) Schrijf een programma dat in op elkaar volgende regels:
1. een regel met 40 streepjes afdrukt;
2. een onderzeeboot op positie S afdrukt;
3. nog een regel met 40 streepjes afdrukt.

waarbij 1. en 3. door een subroutine worden uitgevoerd.

Oefening 2

Vervolgens schrijft u een programma dat van dit eenvoudig spelbord een onderzeeër in een kanaal maakt en deze kan printen op een door het toeval bepaalde plaats in

het kanaal. Daarbij gaat het om de printposities 1–38 voor een scherm met 40 tekens per regel. Gebruik de random-getallengenerator om de plaats te bepalen.

Oefening 3
De essentie van dit spel is natuurlijk al duidelijker. De computer neemt een locatie 'in gedachten' en de speler raadt waar de onderzeeër zich bevindt door een getal van 1 tot 40 in te toetsen. Is de geraden positie goed; dus het getal is tussen S en S+2, dan print de computer 'raak' en eindigt het spel. Is het schot mis, dan print de computer 'mis' en nodigt u uit voor een nieuw spel. Als het spel eindigt, geeft de computer natuurlijk wel nog even aan waar de onderzeeër zich precies in het kanaal bevond.
(We adviseren u al bij het ontwerp rekening te houden met een commando om het spel te beëindigen; het is geen mooie methode een spel door 'ESCAPE' of door eindeloos proberen te moeten beëindigen.)

8.4 Zoeken

Het onderzeeërprobleem is een goede aanleiding ons bezig te gaan houden met zoekproblemen. De enige methode om de onderzeeër te vinden was om systematisch vanaf het begin van de regel tot het einde daarvan alle posities te proberen. Hoeveel gemakkelijker zou het niet zijn als de computer steeds antwoordde met 'te hoog' of 'te laag' na iedere poging. Zonder twijfel zou u dan in staat zijn een geschikte methode te vinden voor het vinden van de onderzeeër.
Als inhoudsopgaven, woordenboeken, telefoonboeken en bibliotheekcatalogi niet in alfabetische volgorde waren opgezet, had u heel wat meer moeite met het vinden van de gewenste informatie.

Nu we een heleboel moeite hebben gedaan om gegevens te sorteren op numerieke of alfabetische volgorde, hebben we ook een techniek nodig om de gesorteerde gegevens weer snel terug te vinden. Als we een telefoonboek openslaan beginnen we niet op de eerste pagina; we doen een eerste gok naar de beginletter van de gezochte naam en verfijnen die gok in successievelijke pogingen. Pas dan beginnen we echt met het zoeken naar de naam.

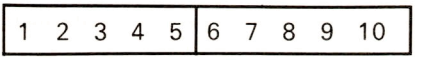
Zoeken door bisectie
Een 'ruwe benadering' is voor een computer geen werkbaar begrip; we kunnen wel op de volgende manier een eerste gok maken:
- splits het bereik in tweeën en vraag of het gezochte item boven of onder dit punt ligt.
- als het gezochte eronder ligt, splitsen we het bereik opnieuw in tweeën met het middelste gegeven als nieuwe bovengrens en ondergrens blijft gelijk.
- ligt het gezochte boven de helft, dan wordt het middelste gegeven de ondergrens.
- we herhalen de twee-deel-procedure totdat het element is gevonden.

Schematisch:

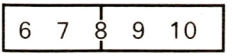
Stel de 7 wordt gezocht in een lijst met:

1	2	3	4	5	6	7	8	9	10

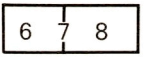
deel dan de lijst:



is 7 het middelste gegeven? Nee.
is 7 < het middelste gegeven? Nee.
7 is dus in de hoogste helft.



is 7 het middelste gegeven? Nee.
is 7 < het middelste gegeven? Ja.
7 bevindt zich dus in de onderste helft.



is 7 het middelste gegeven? Ja.
7 bevindt zich dus in de lijst.

Dit schema illustreert goed hoe bisectie in de praktijk werkt, maar we moeten nog wel onderscheid leren maken tussen de waarde van het gegeven en dat van de index. De index gebruiken we om te delen, en het gegeven om de waarden te vergelijken.

Voorbeeld 2
Een gesorteerde lijst bevat de gegevens A,F,I,M,P,T,U,Z. Gebruik de bisectie-zoekmethode om te bepalen of P al dan niet in de lijst voorkomt.

Index:	1	2	3	4	5	6	7	8
Gegeven:	A	F	I	M	P	T	U	Z
Start-index:	laag (1)							hoog (8)

Middelste index = INT $\frac{(\text{laag} + \text{hoog})}{2}$ = 4 (M)

Vergelijkingen

P is 4e gegeven? NEE P < 4e gegeven? NEE maak index(4) nieuwe gegevens

Index:	4	5	6	7	8
Gegeven:	M	P	T	U	Z
Start-index:	laag (4)				hoog (8)

Middelste index = INT $\frac{(\text{laag} + \text{hoog})}{2}$ = 6 (T)

Vergelijkingen

P is 6e gegeven? NEE P < 6e gegeven? JA maak index(6) tot nieuwe bovengrens

Index:	4	5	6
Gegeven:	M	P	T
Start-index:	laag (4)		hoog (6)

Middelste index = INT $\frac{(4+6)}{2}$ = 5 (P)

Vergelijking

P is 5e gegeven? JA

Afb. 8-6. Zoeken door bisectie.

Oefening 4
Voer dezelfde procedure uit als in voorbeeld 2 om de letter I te zoeken.

Om aan te tonen dat P in de lijst voorkomt, hebben we slechts 3 testen hoeven uit te voeren. Hoewel dat in dit voorbeeld misschien wat overdreven veel lijkt en de methode niet veel voordeel lijkt te bezitten ten opzichte van direct zoeken komt dit pas duidelijker tot uiting bij het zoeken in echt lange lijsten.

In dergelijke korte zoekprocedures is het voordeel van de gestructureerde methode nihil. Voordat we de prestaties van de bisectiemethode gaan testen, moeten we eerst nog wat losse eindjes aan elkaar knopen.

Problemen met bisectie-zoekmethoden

(a) Hoe te stoppen

Voorbeeld 3
Voer de zoekprocedure nog eens uit als in voorbeeld 2 maar nu voor de letter Q.

Index:	4	5	6
Gegeven:	M	P	T
Start-index:	laag(4)		hoog(6)

Middelste index = $\frac{(4 + 6)}{2} = 5$

Vergelijkingen

Q = 5e gegeven? NEE
Q < P ? NEE
maak index(5) nieuwe ondergrens

Index:	4	5	6
Gegeven:		P	T
Start-index:		laag(5)	hoog(6)

Middelste index = INT $\frac{(5 + 6)}{2} = 5$

Vergelijkingen

Q = 5e gegeven? NEE
Q < P ? ---
hebben wij dit niet eens eerder getest?

Afb. 8-7.

Het lijkt erop dat we niet kunnen stoppen. Q is er niet maar de methode zit vast en blijft maar zoeken tussen P en T. Als de index van boven- en ondergrens elkaar zo dicht zijn genaderd dat ze naast elkaar liggen en het gezochte is niet gevonden dan moet geconcludeerd worden dat het gezochte niet in de lijst aanwezig is en kan de zoekprocedure beëindigd worden. Dus de bisectie zoekmethode eindigt als of het object gevonden is, of als boven- en ondergrens opéénvolgende indices hebben gekregen.

(b) Hoe te starten
Het begin lijkt nogal duidelijk. Maak de bovengrens gelijk aan het bovenste element

en de ondergrens aan het onderste element. Problemen ontstaan er dan als het object een waarde heeft die boven of onder de beide grenzen ligt.
Bekijk de volgende lijst met de letters C...S:

Laagste				Hoogste
1	2	3	4	5
C	F	G	P	S

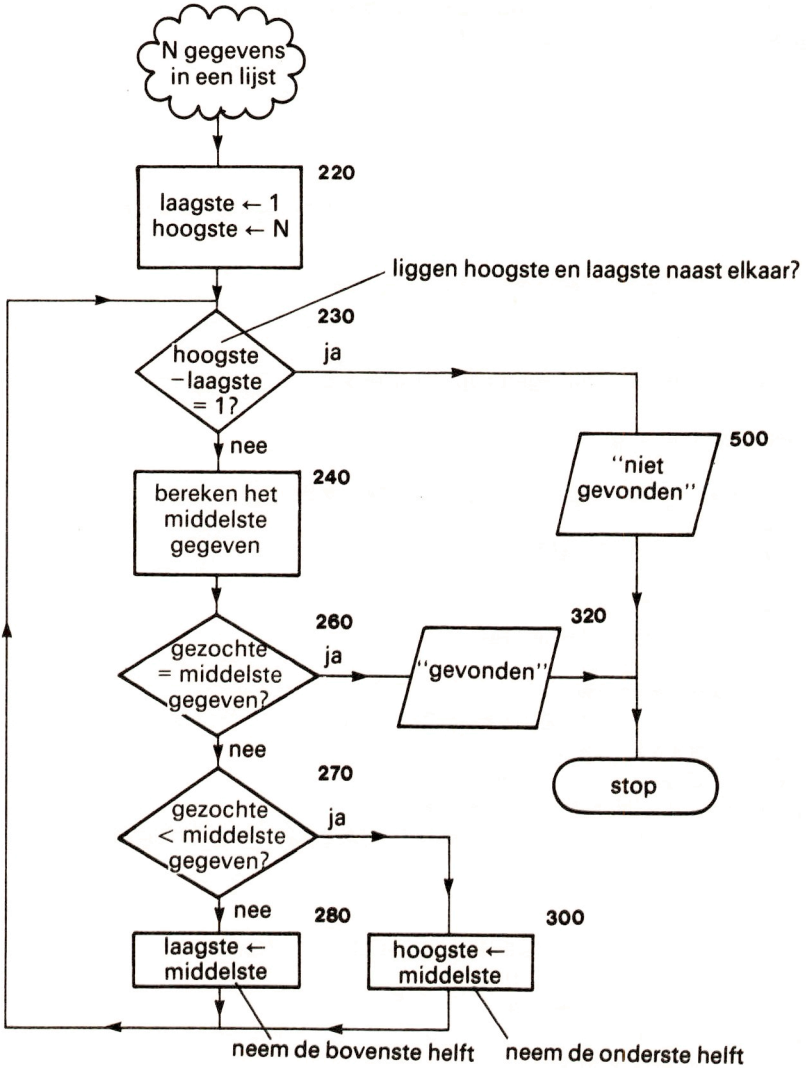
Als het object nu A of B of hoger dan S zou zijn, zou het zoekproces niet eens kunnen beginnen.
De eenvoudigste methode om aan dit probleem te ontkomen, is ervoor te zorgen dat de hoogst toegestane objectwaarde en de laagst toegestane objectwaarde altijd in de lijst aanwezig zijn. De gegeven lijst begint dan dus met gegeven (1) = "AAAA" en eindigt met gegeven (N) = "ZZZZ".
We zetten eerst een stroomdiagram op (zie afb. 8-8)

Nu moeten we het programma nog schrijven

```
10  REM ★★ BISECTIE ZOEKPROGRAMMA ★★
20  CLEAR 100
30  DIM N$(20): DIM T$(20)
40  I = 1
50  READ N$(I), T$(I)
60  IF N$(I) = "ZZZZ" THEN 100
70  I = I + 1 : GOTO 50
90  REM ★★★★★★★★★
100 N = I : REM ★WE GEBRUIKEN ZZZZ HIER★
110 REM ★★★★★★★★★★
150 INPUT "GEZOCHTE NAAM?"; Q$
200 REM ★★ BEGIN MET ZOEKEN ★★
210 PRINT "L"; TAB(5); "H"; TAB(10); "M"; TAB(15); "N$(M)"
220 L = 1 : H = N
230 IF H - L = 1 THEN 500
240 M = INT ((L + H)/2)
250 PRINT L; TAB(5); H; TAB(10); M; TAB(15); N$(M)
260 IF Q$ = N$(M) THEN 320
270 IF Q$ < N$(M) THEN 300
280 L = M : GOTO 230
300 H = M : GOTO 230
320 REM ★★ EINDE ZOEKEN ★★
330 PRINT Q$; "'S TELEFOONNUMMER IS"; T$(M)
350 GOTO 600
500 PRINT Q$; "STAAT NIET IN DE LIJST"
600 INPUT "ZOEKT U NOG EEN ANDER NUMMER?"; R$
610 IF R$ = "JA" THEN 150
620 END
900 DATA AAAA, 0000
910 DATA BERT, 1234
920 DATA CEES, 5678
```

programma gaat verder op pag. 218/219

In stroomdiagram:



Afb. 8-8. Stroomdiagram voor bisectie-zoeken.

```
930 DATA DICK, 9012
940 DATA ERIC, 3456
950 DATA GERT, 7890
960 DATA HENK, 1357
970 DATA MARK, 3567
980 DATA PAUL, 8979
990 DATA SIMON, 5436
```

```
1000 DATA WALTER, 8654
1010 DATA WILLEM, 1212
1020 DATA ZZZZ, 9999
```

Programma 8-6 Bisectie zoekprogramma.

```
RUN
GEZOCHTE NAAM? SIMON
L     H     M     N$(M)
1     13    7     HENK
7     13    10    SIMON
SIMON'S TELEFOONNUMMER IS 5436
ZOEKT U NOG EEN ANDER NUMMER? JA
GEZOCHT NAAM? PAUL
L     H     M     N$(M)
1     13    7     HENK
7     13    10    SIMON
7     10    8     MARK
8     10    9     PAUL
PAUL'S TELEFOONNUMMER IS 8979
ZOEKT U NOG EEN ANDER NUMMER? JA
GEZOCHTE NAAM? RUUD
L     H     M     N$(M)
1     13    7     HENK
7     13    10    SIMON
7     10    8     MARK
8     10    9     PAUL
RUUD STAAT NIET IN DE LIJST
```

8.5 Tabellen

Als we grote hoeveelheden informatie moeten verwerken, moeten we deze ook in het geheugen kunnen opslaan. Daartoe staan ons verschillende manieren ter beschikking. De eerste is het gebruik van een lijst, ook wel één-dimensionaal array genoemd. Een tweede methode gebruikt twee-dimensionale arrays. Stel u heeft de volgende data:

	1e kwartaal	2e kwartaal	3e kwartaal	4e kwartaal
autoverkoop	20	70	80	40
service	10	14	18	11
benzine	30	45	50	30

Afb. 8-9. Inkomsten van een benzinestation.

Nu kunt u deze informatie in een lijst plaatsen, maar deze vorm van organisatie is onhandig en lastig te gebruiken. De eerste vier gegevens hebben betrekking op de autoverkoop, en de tweede vier op service-artikelen enz. Een alternatief is het gebruik van drie lijsten, één voor elke bedrijfstak. BASIC kent het gebruik van meer-dimensionale lijsten met een van de 286 variabelenamen bijvoorbeeld:
XX(I,J)

Lijsten en tabellen.
Lijsten gebruiken een index om de plaats van de verschillende lijstelementen aan te geven; tabellen gebruiken twee indices, meestal naar het Engelse voorbeeld 'sub-scripts' genoemd.

Lijst
L(1), L(2), L(3), ... L(I).
De index van dit element is 3.

Array

A(1,1)	A(1,2)	A(1,3)
A(2,1)	A(2,2)	A(2,3)
A(3,1)	A(3,2)	A(3,3)

Dit element heeft twee sub-scripts:
3 voor rij drie en
2 voor kolom twee

Tabellen
- Een tabel bestaat uit een of alleen string-variabelen of alleen numerieke variabelen. (Een getal kan natuurlijk als string worden opgeslagen en de waarde kan dan met behulp van VAL worden teruggevonden).

- Een tabel kan variabelenamen gebruiken als alle andere variabelen, dus ook één- en tweeletternamen en stringnamen bijvoorbeeld A(I,J) of B$(I,J) of M3(I,J).

Algemeen:

	kolom 1	kolom 2	kolom 3	kolom 4
rij 1	r1k1	r1k2	r1k3	....
rij 2	r2k1	r2k2	r2k3	....
rij 3	r3k1	r3k2	r3k3	....
enz.	....	....	....	....

Afb. 8-10. De rijen en kolommen van een tabel.

Voor het benzinestation hebben we 3 rijen en 4 kolommen nodig, dat zijn totaal 12 elementen:

T(1,1)	T(1,2)	T(1,3)	T(1,4)
T(2,1)	T(2,2)	T(2,3)	T(2,4)
T(3,1)	T(3,2)	T(3,3)	T(3,4)

Dan is T(2,1) = 10
T(3,3) = 50 enz.

Dit komt geheel overeen met de constructie die we al eerder hebben ontmoet bij de bespreking van files met een serie records onderverdeeld in velden. Een file in tabelvorm ziet er als volgt uit:

	Veld 1	Veld 2
	Naam	Telefoonnummer
Record 1	BERT	1234
Record 2	CEES	5678
Record 3	DICK	9012
Record 4	ERIC	3456

Afb. 8-11.

Of in het algemeen:

	Veld 1	Veld 2	Veld 3	enz.
Record 1	R1V1	R1V2	R1V3	....
Record 2	R2V1	R2V2	R2V3	....
Record 3	R3V1	R3V2	R3V3	....
enz.	....	....	....	....

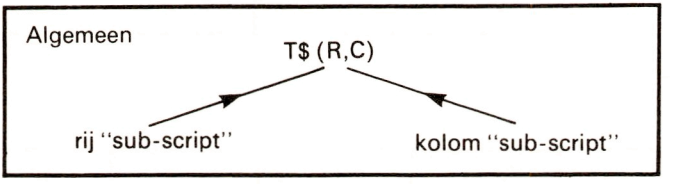
Afb. 8-12.

De individuele elementen uit een tabel telefoonnummers hebben de volgende gedaante:

	Veld 1	Veld 2
	Naam	Telefoonnummer
Record 1	T$(1,1) = BERT	T$(1,2) = 1234
Record 2	T$(2,1) = CEES	T$(2,2) = 5678
Record 3	T$(3,1) = DICK	T$(3,2) = 9012
Record 4	T$(4,1) = ERIC	T$(4,2) = 3456

Afb. 8-13.

- De gehele tabel heet T$;
- Elk element uit de tabel wordt gedefinieerd door twee 'sub-scripts', dus 9012 (derde rij, tweede kolom) wordt genoemd T$(3,2).
- De 'drie' en de 'twee' karakteriseren de positie van het element T$(3,2), en niet zijn waarde. De waarde is 9012 dus:
  T$(3,2) = 9012



Afb. 8-14.

Voorbeeld 4
De tabel N$ heeft 9 elementen met onderstaande waarden. Wat zijn de variabelenamen?

	R \ K	1	2	3
	1	KEES	KAREL	HENK
N$(,)	2	WIM	JAN	DIRK
	3	INA	MIES	JOKE

Afb. 8-15.

Oplossing:

KEES = N$(1,1)	KAREL = N$(1,2)	HENK = N$(1,3)
WIM = N$(2,1)	JAN = N$(2,2)	DIRK = N$(2,3)
INA = N$(3,1)	MIES = N$(3,2)	JOKE = N$(3,3)

ZOP 3
In de volgende tabel A$ moeten de variabelen met hun sub-scripts benoemd worden.

SJAAN	LENIE	GERDA
MARJAN	TRUDIE	WILMA
CARLA	INGRID	PETER
RUUD	JOHN	JOHAN
BERT	BAS	GERRIT

Afb. 8-16.

Tabellen en geneste lussen
Zoals FOR...NEXT-lussen en lijsten al voor elkaar gemaakt leken te zijn, zo zijn de tabellen complementair aan geneste FOR...NEXT-lussen.

Stel dat u de volgende namen in een tabel wilt inlezen:

BAS, BERT, CARLA, GERDA, GERRIT, INGRID, JOHAN, JOHN, LENIE, MARJAN, PETER, ROB, RUUD, SJAAN, TRUDIE, WILMA

U kunt dat op ideale wijze realiseren met een READ-statement in twee geneste FOR-...NEXT-lussen.

```
60  FOR I = 1 TO 4
70  FOR J = 1 TO 4
80  READ N$(I,J)
90  NEXT J
100 NEXT I
```

In programma 8-7 wordt deze methode toegepast in de regels 10 tot 100.

Het is natuurlijk wel aardig een tabel met namen te vullen, maar we willen natuurlijk ook iets zien, daartoe is de printroutine toegevoegd die de lijst in een tabelvorm afdrukt met I en J erbij.

```
10  REM ★★ TABEL READ EN PRINT ★★
20  CLEAR 200
30  DIM N$(20,5)
40  REM ★★ READ ROUTINE ★★
60  FOR I = 1 TO 4
70  FOR J = 1 TO 4
80  READ N$(I,J)
90  NEXT J
100 NEXT I
110 REM ★★ PRINT ROUTINE ★★
115 PRINT "I", "J", "N$(I,J)"
```

DIM-statement voor een 2-dimensionaal array van 20 rijen en 5 kolommen

```
130 FOR I = 1 TO 4 ——————————— rijen
140 FOR J = 1 TO 4 ——————————— kolommen
150 PRINT I, J, N$(I,J)
160 NEXT J
180 NEXT I
190 GOTO 270
240 REM ★★★★★★★★★★
250 DATA BAS, BERT, CARLA, GERDA, GERRIT
260 DATA INGRID, JOHAN, JOHN, LENIE
270 DATA MARJAN, PETER, ROB, RUUD, SJAAN
280 DATA TRUDIE, WILMA
290 END
```

Programma 8-7 Data inlezen in een 4 x 4 array.

```
RUN
I          J          N$(I,J)
1          1          BAS
1          2          BERT
1          3          CARLA
1          4          GERDA
2          1          GERRIT
2          2          INGRID
2          3          JOHAN
2          4          JOHN
3          1          LENIE
3          2          MARJAN
3          3          PETER
3          4          ROB
4          1          RUUD
4          2          SJAAN
4          3          TRUDIE
4          4          WILMA
```

[K] Programma 8-7
BBC regel 150 PRINT; I,;J,; N$(I,J)

ZOP 4
Als de volgende wijzigingen in programma 8-7 worden aangebracht, geef dan aan hoe de tabel eruit ziet nadat een run is uitgevoerd.

```
60  FOR I = 1 TO 3
70  FOR J = 1 TO 5
130 FOR I = 1 TO 3
140 FOR J = 1 TO 5
```

Uitvoer in tabelvorm
De uitvoer van programma 8-7 is niet geheel bevredigend. Een afdruk in twee dimensies in plaats van een lange rij zou een beter overzicht van de tabel geven:

```
130 FOR I = 1 TO 4
140 FOR J = 1 TO 4
150 PRINT TAB(10★(J-1)); N$(I,J);
160 NEXT J
170 PRINT
180 NEXT I
190 GOTO 270
```

de eerste kolom begint met de indexwaarde 0

kolommen 10 karakters breed

Programma 8-8.

De uitvoer wordt dan:

```
RUN
BAS         BERT        CARLA       GERDA
GERRIT      INGRID      JOHAN       JOHN
LENIE       MARJAN      PETER       ROB
RUUD        SJAAN       TRUDIE      WILMA
```

K Programma 8-8

Opgave 8

1. Een vertegenwoordiger heeft 4 produktlijnen. Zijn verkoopcijfers zijn in de onder staande tabel weergegeven:

dag \ produkt	1	2	3	4	totalen
1	500	300	20	25	e
2	600	700	40	0	f
3	200	550	60	20	g
4	250	450	100	5	h
5	400	200	100	11	i
totalen	a	b	c	d	t

Afb. 8-17.

Schrijf een programma voor hem dat zijn resultaten voor hem verduidelijkt door:
(a) Zijn dagtotalen af te drukken (e,f,g,h,i,j)
(b) De produkttotalen af te drukken (a,b,c,d)
(c) Zijn weektotaal voor hem af te drukken (t).

2. Schrijf een programma dat het onderzeeërspel uitbreidt tot een 10 × 10 veld. Als een peiling vlakbij de onderzeeër plaatsvindt, moet het programma een 'bijna gevonden' boodschap geven. U beslist wat precies 'dichtbij' is.

Samenvatting van hoofdstuk 8

Aan het einde van hoofdstuk 8 dient u de volgende onderwerpen te beheersen en in eenvoudige BASIC-programma's te kunnen toepassen:

- Handmatig sorteren met een verwisselroutine.
- Sorteren met verwisselroutine in twee geneste programmalussen.
- Toepassen van GOSUB.
- Handmatig zoeken met de bisectiemethode in de dataregels.
- Programmatisch zoeken met de bisectiemethode.
- Data in tweedimensionale arrays inlezen.
- Met een programma een tweedimensionale array inlezen.
- De data uit een tweedimensionaal array uitprinten.
- De hoeveelheid rijen en kolommen kunnen bepalen in twee-dimensionale arrays.

Antwoorden op ZOP's en oefeningen

ZOP 1

Na de eerste run:	0	1	4	6	2	3	7	8
Na de tweede run:	0	1	4	6	2	3	7	8
Na de derde run:	0	1	2	6	4	3	7	8
Na de vierde run:	0	1	2	3	4	6	7	8

ZOP 2

(a) B = 1/25
(b) B = 9 (GOSUB werd hier niet gebruikt)

Oefening 1

(a)
```
10 FOR I=1 TO 40
20 PRINT "-";
30 NEXT I
40 PRINT
```
Programma 8-9.

of in één regel:

```
10 FOR I=1 TO 40:PRINT "-";:NEXT I: PRINT
```

(b)
```
10 PRINT TAB(S); "<=>"
```

(c)
```
10 INPUT S
20 GOSUB 100
30 PRINT TAB(S); "<=>"
40 GOSUB 100
50 END
100 FOR I=1 TO 40: PRINT "-";: NEXT I: PRINT
110 RETURN
```
Programma 8-10.

De waarde van S bepaalt de positie van de onderzeeër in het kanaal, en dat levert het volgende plaatje:

```
--------------------------------
                <=>
--------------------------------
```

Oefening 2

```
10  REM ★★ ONDERZEEËR ★★
20  RANDOMIZE
50  S = INT(38★RND(1)+1)
60  GOSUB 510
70  PRINT TAB(S); "<=>"
80  GOSUB 510
90  END
500 REM ★★ PRINT SUBROUTINE ★★
510 FOR I = 1 TO 39: PRINT "-"; : NEXT I : PRINT
520 RETURN
```

Programma 8-11.

```
RUN
--------------------------------
                <=>
--------------------------------
RUN
--------------------------------
  <=>
--------------------------------
RUN
--------------------------------
                              <=>
--------------------------------
```

Oefening 3

```
10  REM ★★ ONDERZEEËR ★★
20  RANDOMIZE
30  REM ★★ ZOEK SPEL ★★
40  GOSUB 300 ____________________________________ 1e
50  PRINT "EEN GETAL VAN 1 TOT 39 OM MIJ TE VINDEN"
60  GOSUB 300 ____________________________________ 2e
70  REM ★★ WILLEKEURIGE POSITIE ★★
80  S = INT(38★RND(1)+1)
90  PRINT
100 INPUT "PROBEER EEN ANDER GETAL!";X
110 IF X < S THEN 190
120 IF X > S + 2 THEN 190
130 REM ★★ GEVONDEN ★★
140 GOSUB 300 ____________________________________ 3e
150 PRINT TAB(S); "GEVONDEN"
160 GOSUB 300 ____________________________________ 4e
170 GOTO 320
180 REM ★★ !!MIS!! ★★
```

```
190 GOSUB 300 ____________________________________ 5e
200 PRINT "MIS GERADEN"
210 GOSUB 300 ____________________________________ 6e
220 INPUT "WILT U NOG DOORSPELEN?"; R$
230 IF R$ = "JA" THEN 90
240 PRINT "IK ZAT HIER!!!" : PRINT
250 GOSUB 300 ____________________________________ 7e
260 PRINT TAB(S); "<=>"
270 GOSUB 300 ____________________________________ 8e
280 GOTO 320
290 REM ★★ PRINT SUBROUTINE ★★
300 FOR I = 1 TO 39 : PRINT "-"; :NEXT I : PRINT
310 RETURN
320 END
```

SUBROUTINE print een regel met '-'.

Programma 8-12.

```
RUN
----------------------------------
EEN GETAL VAN 1 TOT 39 OM MIJ TE VINDEN
----------------------------------
PROBEER EEN ANDER GETAL! 27
----------------------------------
MIS GERADEN
----------------------------------
WILT U NOG DOORSPELEN? JA

PROBEER EEN ANDER GETAL! 25
----------------------------------
MIS GERADEN
----------------------------------
WILT U NOG DOORSPELEN? NEE
IK ZAT HIER!!!
----------------------------------
   <=>
----------------------------------
```

Oefening 4

Index:	1	2	3	4	5	6	7	8
Gegeven:	A	F	I	M	P	T	U	Z

Middelste index = INT $\frac{(8 + 1)}{2}$ = 4 (M)

Vergelijkingen

I = 4e gegeven? NEE
I < 4e gegeven? JA
4 wordt nieuwe bovengrens

Index: 1 2 3 4
Gegeven: A F I M

Middelste index = INT $\frac{(4+1)}{2}$ = 2 (F)

Vergelijkingen

I = 2e gegeven? NEE
I < 2e gegeven? NEE
2 wordt nieuwe ondergrens

Index: 2 3 4
Gegeven: F I M

Middelste index = INT $\frac{(4+2)}{2}$ = 3 (I)

Vergelijking

I = 3e gegeven? JA
I bevindt zich dus in de lijst

Afb. 8-18.

ZOP 3

A$(1,1) = SJAAN	A$(1,2) = LENIE	A$(1,3) = GERDA
A$(2,1) = MARJAN	A$(2,2) = TRUDIE	A$(2,3) = WILMA
A$(3,1) = CARLA	A$(3,2) = INGRID	A$(3,3) = PETER
A$(4,1) = RUUD	A$(4,2) = JOHN	A$(4,3) = JOHAN
A$(5,1) = BERT	A$(5,2) = BAS	A$(5,3) = GERRIT

ZOP 4

RUN

I	J	N$(I,J)
1	1	BAS
1	2	BERT
1	3	CARLA
1	4	GERDA
2	1	GERRIT
2	2	INGRID
2	3	JOHAN
2	4	JOHN
3	1	LENIE
3	2	MARJAN
3	3	PETER
3	4	ROB
4	1	RUUD
4	2	SJAAN
4	3	TRUDIE

(merk op dat de laatste naam: 'WILMA' niet in deze tabel voorkomt. Een 5 × 3 tabel leest alleen de eerste 15 namen)

9 Manipuleren met files

Waarschuwing voor BBC-computer
In dit hoofdstuk gaan we in op eenvoudige manipulaties met files. Daarbij wordt één belangrijk aspect misschien niet voldoende duidelijk.

Op de meeste computers betekent OPENOUT het openen van een bestaande file om informatie in te schrijven. Op de BBC betekent OPENOUT het creëren van een nieuwe file. Gebruikt u OPENOUT op een bestaande file, dan wordt deze vernietigd.

Als u wilt lezen of schrijven op een bestaande file van een BBC-computer dan gebruikt u OPENIN.

OPENOUT creëert nieuwe files
OPENIN lezen en schrijven op bestaande files.

9.1 Programma-opslag

Dit boek heeft zich tot nu toe beziggehouden met een computer bestaande uit drie delen: toetsenbord, centrale verwerkingseenheid en een beeldmonitor. Een dergelijk systeem is, zoals u heeft kunnen vaststellen, volledig afhankelijk van de netspanning. Als u de spanning uitschakelt vergeet de computer letterlijk alles wat u er aan programmeermoeite heeft ingestoken. Als u een bepaald programma opnieuw zoudt willen uitvoeren moet u het geheel opnieuw intoetsen. Gelukkig zijn alle microcomputers voorzien van een schakeling om programma's op gewone audiocassettes op te nemen. Als uw programma eenmaal op een cassette is opgenomen, kunt u het elk ogenblik opnieuw in uw computer laden en gebruiken.

Er kunnen twee redenen zijn waarom u iets op cassette wilt bewaren:
(a) U heeft een compleet programma geschreven dat u niet wilt verliezen of
(b) U heeft een gedeelte van een programma ingebracht dat u aan het ontwikkelen bent.
Als u iedere keer nadat u een scherm vol programmatekst heeft geschreven, deze tekst op een cassette registreert, dan verliest u nooit te veel als er op de één of andere wijze iets misgaat. Als het complete programma klaar is en eenmaal op cassette is vastgelegd, kunnen de deelprogramma's gewist worden.

De opneemprocedure
Als eerste moet u het programma een naam geven waaronder de computer het programma kan opslaan. Elk systeem stelt zijn eigen eisen aan de lengte en karakterset

waaruit de programmanaam mag zijn samengesteld. Meestal is een minimum van zes karakters toegestaan en moet de naam met een letter uit het alfabet beginnen. Uw handleiding kan u verder informeren.

Veronderstel dat uw programma PRONAAM heet en u wilt deze op een cassette vastleggen. Daartoe sluit u een audiocassetterecorder op uw computer aan en plaatst een cassette in uw recorder. De verdere handelwijze is als volgt:

- Toets 'SAVE "PRONAAM"' gevolgd door RETURN;
- Start uw recorder in de stand 'opname';
- Druk opnieuw op RETURN;
- Na afloop van het opneemproces stopt u de recorder.
  (Dit is de procedure op de BBC-computer)

Of:
- Start uw recorder in de stand opname;
- Toets SAVE gevolgd door RETURN;
- Uw computer vraagt nu om FILENAME? Toets PRONAAM (of de naam naar uw keuze) zonder " " gevolgd door RETURN.

De laadprocedure
Een eenmaal opgeslagen programma kan worden teruggelezen met een laadcommando. Sluit de cassetterecorder op uw computer aan en plaats de gewenste cassette in de recorder. Hierna handelt u als volgt:

- Toets LOAD "PRONAAM" en toets RETURN
- Start de recorder in de stand 'weergave';
- Schakel de recorder af als het programma geladen is.
  (Dit is de procedure op de BBC-computer)

of
- Toets LOAD gevolgd door RETURN;
- Op de vraag FILENAME? antwoordt u met PRONAAM (zonder " ") gevolgd door RETURN;
- Schakel de recorder in in de stand 'weergave';
- Als het programma geladen is, schakelt u de recorder af.

9.2 LPRINT

Vroeger of later wilt u een kopie van uw programma op papier. In computertaal, zoals u inmiddels al weet, sterk op de Engelse taal gericht, noemt men dat een 'hard-copy'. Alle tot nu toe gebruikte programma's waren geschikt om ook op de printer dezelfde tekst te verkrijgen als op uw scherm. Het is echter niet altijd wenselijk om deze tekst zowel op het scherm als op de printer te hebben. Bijvoorbeeld in run 13 uit hoofdstuk 7 (iteratief worteltrekken) zijn we wel geïnteresseerd in het resultaat, maar niet zozeer in de conversatie vooraf:

GEWENSTE NAUWKEURIGHEID?... enz.

Deze gegevens zijn wel zinvol op het scherm, maar we hebben ze niet direct op papier nodig. Om onderscheid te maken tussen de schermtekst en de printertekst is een apart printstatement in veel BASIC's geïmplementeerd:

LPRINT stuurt de te printen tekst rechtstreeks naar de printer en
LLIST maakt een listing van het programma op papier.

Sommige BASIC's kennen ook het LRUN statement, dat alle tekst gedurende een bepaalde run naar de printer stuurt.

De BBC-computer kent geen van de bovenstaande statements. In plaats daarvan zijn een aantal speciale commando's opgenomen.

- Vanaf het moment dat u het commando CTRL B heeft gegeven verschijnt alle tekst op zowel scherm als printer; na het commando CRTL C is de printer weer uitgeschakeld.
- In een programma kunt u met de statements VDU2 EN VDU3 de printer resp. in- en uitschakelen.

9.3 Sequentieel toegankelijke files

We hebben het begrip 'file' al verschillende malen genoemd. Daarbij hadden we het over een verzameling data (gegevens). Een programma dat op cassette is opgeslagen heeft echter ook een file-naam en zowel data als programma's heten dan files.

Data

Alle tot nu toe gebruikte gegevens in de programma's hebben we via het toetsenbord of met behulp van DATA-statements in het programma gebracht. Dat gaat uitstekend zolang de hoeveelheid data beperkt is. Zodra we echter met grotere hoeveelheden data gaan werken, wordt het noodzakelijk deze op cassette of floppy-disk op te slaan. Daartoe moeten we speciale commando's in BASIC gebruiken die ons in staat stellen gegevens op cassette of disk op te slaan en weer terug te lezen.

Incompatibiliteit van systemen

De meeste microcomputersystemen verschillen in detail van elkaar van wat de methode van registratie en file-behandeling betreft. We concentreren ons daarom op algemene principes die op de meeste systemen gelijk zijn. De voorbeelden zijn zo eenvoudig mogelijk gehouden. Het is echter zeer waarschijnlijk dat u op uw systeem kleine wijzigingen in de voorbeelden zult moeten aanbrengen. De BBC-computer wijkt op verschillende punten af; waar nodig zijn de aanpassingen voor de BBC-computer vermeld.

Sequentiële files

Als vereenvoudiging op de file-manipulatietechnieken zullen we ons beperken tot sequentieel toegankelijke files.
Sequentiële files zijn files waaruit ieder gegeven alléén kan worden gelezen in de volgorde waarin ze zijn geschreven toen de file werd gecreëerd. Dit is het enige type file dat u kunt gebruiken als u een cassetterecorder als geheugenmedium gebruikt omdat de cassettenband nu eenmaal sequentieel geschreven en gelezen moet worden.

9.4 Werken met files

Bij het maken van een file wordt data vanuit het geheugen van de computer geschreven naar het randapparaat (in ons geval cassette of disk). Tijdens het lezen wordt data vanuit het randapparaat naar het programmageheugen van de computer gebracht.

Omdat het niet mogelijk is de juistheid van de informatie op de drager vast te stellen, moeten de schrijf- en leesoperaties complementair zijn. U kunt niet met zekerheid vaststellen of een schrijfoperatie gelukt is totdat de informatie teruggelezen is en op het beeldscherm of printer is gecontroleerd.

Het is mogelijk om gelijktijdig een file te maken en te lezen door gebruik te maken van twee cassetterecorders. Hierop zullen we verder niet ingaan en het maken en het lezen van een file zoveel mogelijk gescheiden behandelen. Het maken van een file (Eng. create) vereist de volgende acties:

- de opgave van een symbolische naam (filenaam);
- het 'openen' van de file om informatie er naar toe te voeren;
- het daadwerkelijk schrijven van de informatie in de file;
- het 'sluiten' van de file.

Merk op dat:

- gedurende het maken van een file, de computer een tijdelijk nummer voor de desbetreffende file nodig heeft om deze gedurende de uitvoering te identificeren.
- tussen het openen en sluiten van de file de computer wordt gecontroleerd van de recorder of disk-drive.

Het maken van een file geschiedt op de meeste computers als volgt

```
1000 REM ★★ CREEER EEN DATAFILE ★★
1010 OPEN "O", £2, "DATA 1"      BBC: regel 1010 A = OPENOUT "DATA 1"
```


```
1060 CLOSE £2                    : regel 1060 CLOSE #A
```

Programma 9-1.

Toelichting

Open "O" of OPENOUT	geeft de computer aan een file voor output te openen ("O" voor output)
£2 of #A	een tijdelijk intern filenummer dat we, gedurende de periode dat de file geopend is, gebruiken. Op sommige computers is het gebruik van een hele reeks nummers toegestaan. Maar wij zullen ons houden aan £2.

"DATA 1" symbolische naam waaronder de file op cassette of disk wordt opgeslagen.

CLOSE £2 of CLOSE #A dit statement sluit de file af en plaatst een 'end-of-file' kenmerk aan het einde daarvan. Na CLOSE kunnen op het interne filenummer geen gegevens meer worden geschreven.

Voorbeeld 1a
Schrijf een programma dat een DATA-file aanmaakt van 10 namen.

Oplossing:

```
1000 REM ★★ CREEER EEN DATA FILE ★★
1010 OPEN "O", £2, "DATA 1"
1020 FOR I = 1 TO 24
1030 READ N$
1040 PRINT £2, N$
1050 NEXT I
1060 CLOSE £2
1090 PRINT "FILE DATA 1 IS OPGESLAGEN"
9000 DATA RUUD, PETER, JAAP, JOHAN, BERT, ADA, JULIA, KEES,
     CARLA, PAULINE, JOOS, JANNY
9010 DATA HENK, HENRY, GERT, GERDA, FRITS, TED, ROB, BEN, NICO,
     WILLY, LEO, BAS
9020 END
```

schrijft in de file

hoge regelnummers, want het aanmaken van een file komt nadat een programma al informatie heeft aangemaakt

Programma 9-2 Creëer een data-file.

```
RUN
RECORD then RETURN
FILE DATA 1 IS OPGESLAGEN
```

voordat u 'RETURN' indrukt moet u de recorder hebben ingeschakeld... en hier weer uitschakelen

```
BBC: 1010 A = OPENOUT "DATA 1"
     1040 PRINT #A, N$
     1060 CLOSE #A
```

Als de file eenmaal is geopend, kan erin geschreven worden:

```
1020 FOR I = 1 TO 24
1030 READ N$
1040 PRINT £2, N$                    BBC: 1040 PRINT #A, N$
1050 NEXT I
```

Programma 9-3.

Het enige nieuwe in deze routine is PRINT £2,N$. Dit statement plaatst het element N$ (string) in de file met het interne filenummer £2. (Nu weet u waarom we een intern nummer aan de file toekennen.)

Na afloop van het programma zijn alle 24 namen op cassette geregistreerd in een

file met de symbolische naam "DATA 1". Ook na uitschakeling van uw computer verandert er niets, de file blijft ongewijzigd.

Lezen uit files

Nu wilt u de informatie natuurlijk ook weer teruglezen. Daartoe heeft u een leesprogramma nodig met de volgende structuur:

```
10  REM ★★ LEZEN UIT EEN DATAFILE ★★
20  OPEN "I", £1, "DATA 1"          BBC: 20  B = OPENIN "DATA 1"
30  IF EOF(1) THEN 100              BBC: 30  IF EOF#B = -1 THEN 100
```


```
100 CLOSE £1                        BBC: 100 CLOSE #B
```

Programma 9-4.

Toelichting

OPEN "1" of OPENIN	draagt de computer op een file te openen in de 'input'-mode.
£1 of #B	intern filenummer (als bij output).
"DATA 1"	de filenaam waar de computer naar zoekt op cassette of disk.
EOF(1) of EOF	vraagt de computer of het 'end-of-file'-kenmerk inmiddels is gelezen.
CLOSE £1 of CLOSE #B	sluit de file af voor verdere informatie.

Voorbeeld 1b

Schrijf een programma dat de file DATA 1 uit voorbeeld 1a inleest en de 24 namen weer afdrukt.

```
10  REM ★★ LEZEN UIT EEN DATA FILE ★★
20  OPEN "I", £1, "DATA 1"          BBC: 20  B = OPEN "DATA 1"
30  IF EOF(1) THEN 100                   :30  IF EOF#B = -1 THEN 100
40  INPUT £1, A$                         :40  INPUT #B,A$
50  PRINT A$
60  GOTO 30
100 CLOSE £1                             :100 CLOSE #B
110 END
```

Programma 9-5 Lezen uit een data-file.

```
RUN
RUUD
PETER
JAAP
```

voordat u na RUN op de RETURN-toets drukt, schakelt u de recorder op weergave

JOHAN
BERT
ADA
JULIA
KEES
CARLA
PAULINE
JOOS
JANNY
HENK
HENRY
GERT
GERDA
FRITS
TED
ROB
BEN
NICO
WILLY
LEO
BAS

Als de file eenmaal geopend is, kunnen we data inlezen uit de file met het INPUT-statement:

```
40 INPUT £1,A$
```

BBC: 40 INPUT #B,A$

Het eerste gegeven (hier string) is na dit statement bekend in A$ en kan naar keuze verder verwerkt worden. In dit geval zal het gegeven worden afgedrukt:

```
50 PRINT A$
```

Inleiding bij oefeningen
Een belangrijke handicap van de gegeven schrijf- en leesprogramma's is het feit dat zij beide op een bepaalde vaste filenaam opereren namelijk DATA 1. Een programma voor DATA-verwerking moet echter met willekeurige filenamen kunnen functioneren. Daartoe vervangen we de string in de OPEN-statements door een stringvariabele F$:

```
INPUT "NAAM VAN DE DATA-FILE"; F$
OPEN "O", £2, F$
```

BBC: A = OPENOUT (F$)

Het is natuurlijk ook niet erg zinvol de gegevens uit DATA-statements te moeten inlezen. In de volgende oefeningen gaan we daar wat aan doen.

Oefening 1
Pas programma 9-2 zo aan dat de reeks namen rechtstreeks van het toetsenbord kan worden ingelezen en dat tevens een filenaam naar keuze kan worden ingetoetst.

Oefening 2
Pas programma 9-5 zo aan dat het de aangepaste file uit oefening 1 kan lezen en uitprinten.

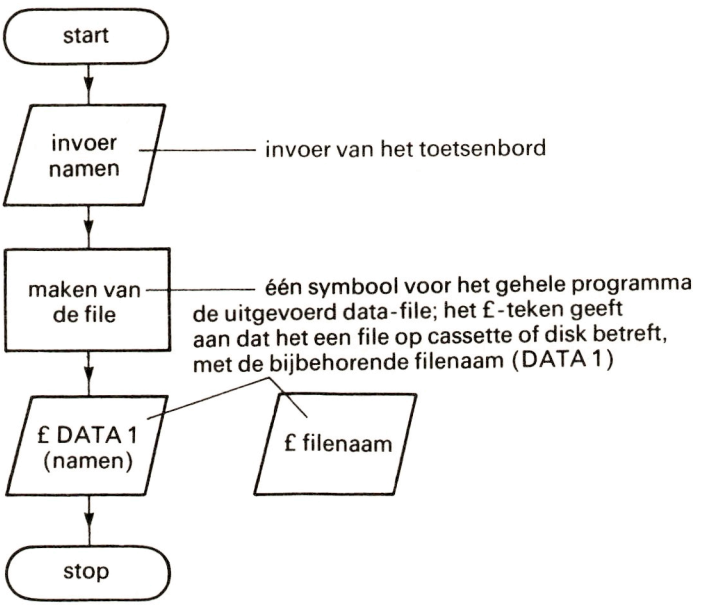
Oefening 3
Schrijf een programma dat de file uit oefening 1 inleest en nazoekt op alle namen beginnend met een N. Druk de lijst vervolgens af.

Oefening 4
Schrijf een programma dat de file uit oefening 1 inleest in een lijst en die vervolgens met de indices afdrukt.

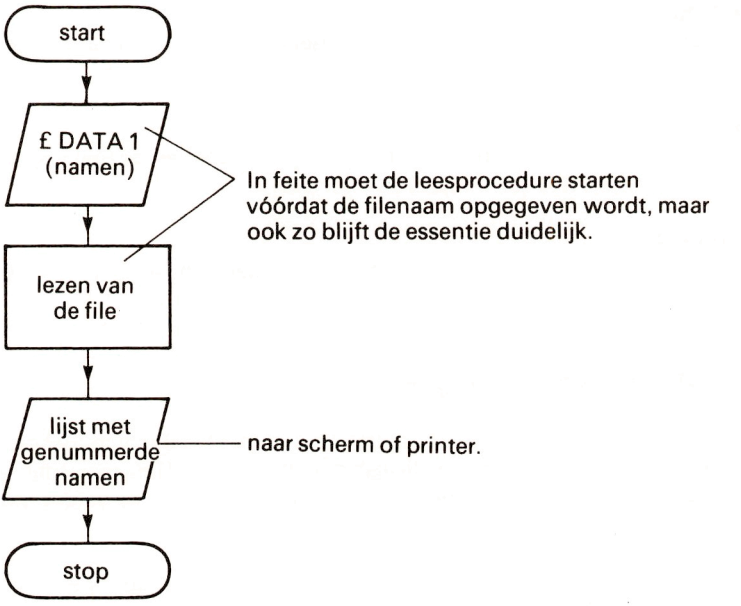
9.5 Files in stroomdiagrammen

De oplossing van oefening 4 is de sleutel tot de ontwikkeling van toekomstige programmatuur. Onze oude bekende, de lijst, doet 't em weer. Data in een lijst verzameld is veel handiger om mee te manipuleren. Voordat we daarmee verder gaan, vatten we nog even kort samen tot waar we gekomen zijn.

De oplossing van oefening 1 kan worden samengevat als in figuur 9-1 en de oplossing van oefening 4 is afgebeeld in figuur 9-2.



Afb. 9-1. Stroomdiagram voor file-creatie.



Afb. 9-2. Inlezen van een file in een lijst.

9.6 Sorteren van files

De lijst met namen uit oefening 2 vraagt erom gesorteerd te worden! Daartoe koppelen we de sorteerroutine uit hoofdstuk 8 aan ons file-inleesprogramma. Voordat we u het programma presenteren, beelden we het algorithme in een stroomdiagram af.

Het programma is eenvoudig van structuur en bestaat uit drie routines (inlezen, sorteren en printen) die u al eerder bent tegengekomen.

```
10  REM ★★ INLEZEN EN SORTEREN ★★
20  DIM X$(50)
30  C = 0
40  INPUT "NAAM VAN DE DATA FILE"; G$
50  OPEN "I",£1, G$
60  IF EOF (1) THEN 100
70  C = C + 1
80  INPUT £1, X$(C)
90  GOTO 60
100 CLOSE £1
110 REM ★★ EINDE INLEZEN ★★
130 REM ★★ N IS DE LIJSTLENGTE ★★
140 N = C
```

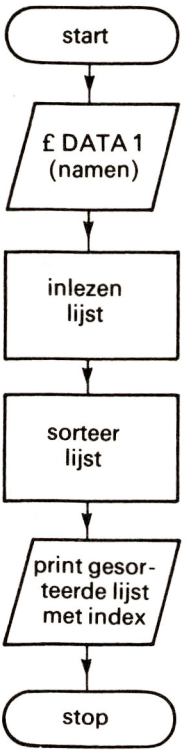
```
BBC: 50  A = OPENIN(G$)
60  IF EOF#A = -1  THEN
inlezen
80  INPUT  #A, X$(C)

100 CLOSE  #A
```

```
200 REM ★★★★★★★★★★
210 REM ★★ SORTEERROUTINE ★★
220 FOR K=1 TO N-1
230 FOR L=K+1 TO N
240 IF X$(L) >=X$(K) THEN 280
250 T$=X$(L)
260 X$(L)=X$(K)
270 X$(K)=T$
280 NEXT L
290 NEXT K
300 REM ★★ EINDE SORTEERROUTINE ★★
400 REM ★★★★★★★★★★
410 PRINT "GESORTEERDE LIJST"
420 FOR P=1 TO N
430 PRINT P, X$(P)
440 NEXT P
450 PRINT
500 REM ★★★★★★★★★★
510 END
```

sorteren

printen

Programma 9-6 Inlezen en sorteren van een file.



Afb. 9-3. Inlezen en sorteren van een file.

```
RUN
NAAM VAN DE DATA FILE DATA 1
GESORTEERDE LIJST
 1          ADA
 2          BAS
 3          BEN
 4          BERT
 5          CARLA
 6          FRITS
 7          GERDA
 8          GERT
 9          HENK
10          HENRY
11          JAAP
12          JANNY
13          JOHAN
14          JOOS
15          JULIA
16          KEES
17          LEO
18          NICO
19          PAULINE
20          PETER
21          ROB
22          RUUD
23          TED
24          WILLY
```

Oefening 5

We hebben niet direct behoefte aan een print-out van de gesorteerde lijst zoals in programma 9-6, maar willen de gesorteerde lijst in een file bewaren voor toekomstig gebruik.
Pas programma 9-6 zo aan dat de gesorteerde data naar een nieuwe file bijvoorbeeld SDATA wordt geschreven.
Hoe kunt u testen of dat is geslaagd?

Oefening 6

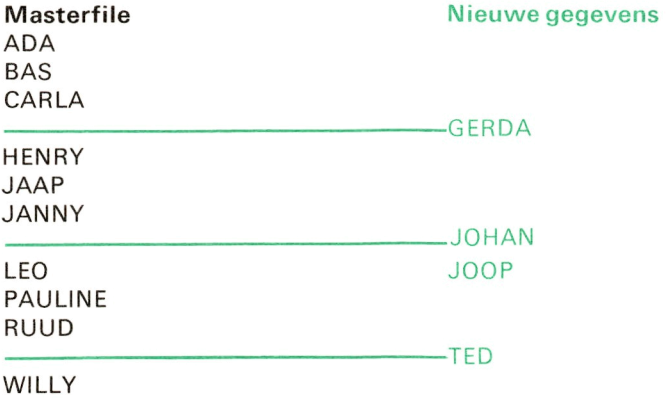
Nu u een file met gesorteerde data heeft gemaakt, zult u behoefte hebben aan een zoekprogramma om snel informatie te kunnen terugvinden.
Combineer programma 9-6 met de bisectie-zoekmethode uit hoofdstuk 8 om dit te realiseren.

Oefening 7

Maak een programma om een file in te lezen en de desbetreffende data en een tabel met lay-out naar keuze te printen.
Gebruik data naar eigen inzicht.

9.7 Samenvoegen van files (Eng. Merge)

We hebben nu leren werken met files. We kunnen files maken, teruglezen, sorteren en doorzoeken. Wat kunnen we nog meer met files doen?
Een file met veel gebruikte informatie blijft zelden steeds gelijk. We willen erin veranderen, toevoegen of verwijderen. De rest van dit hoofdstuk besteden we aan de vraag hoe we gegevens kunnen toevoegen aan en verwijderen uit files.
Zonder twijfel heeft u reeds verschillende oplossingsmethoden in gedachten voor deze problemen. We hopen dat u die zeker zult uitproberen. Toch willen we u een algemene aanpak aanraden. Om gegevens aan een file toe te voegen, voegen we een file aan de file toe. U heeft dit ongetwijfeld al vaak gedaan terwijl u kaart speelde. Gewonnen kaarten (nieuwe gegevens) die u als een stapel in handen krijgt (file) voegt u aan de gesorteerde set (gesorteerde file) in uw handen toe.



Afb. 9-4.

Voordat we hiervoor een routine ontwikkelen, moeten we een oplossing vinden voor het extremen-probleem uit de bisectie-routine in hoofdstuk 8. Daartoe ontwikkelen we eerst een routine.

De extremen uit de masterfile
De volgende routine neemt een gesorteerde file (G$) en plaatste "AAAA" als eerste element en "ZZZZ" als laatste element in de lijst waarin deze file gelezen wordt.

```
10  REM ★★ LEZEN UIT EEN DATA FILE ★★
20  DIM M$(50)
30  C = 1 : M$(C) = "AAAA"          eerste item in de lijst: "AAAA"
40  INPUT "NAAM VAN DE DATA FILE";G$
50  OPEN "I", £1, G$
60  IF EOF(1) THEN 100
70  C = C + 1
80  INPUT £1, M$(C)
90  GOTO 60
100 CLOSE £1
```

```
BBC: 50  A = OPENIN(G$)
     60  IF EOF#A = -1 THEN 100
     80  INPUT #A, M$(C)
     100 CLOSE #A
```

```
110 REM ★★★★★★★★★★
120 N = C + 1 : M$(N) = "ZZZZ"
130 REM ★★★★★★★★★★
140 FOR I = 1 TO N
150 PRINT I, M$(I)
160 NEXT I
170 END
```

laatste item in de lijst: "ZZZZ"

Programma 9-7 Oplossing voor het extremenprobleem in bisectieroutines.

```
RUN
NAAM VAN DE DATA FILE DATA 1
 1          AAAA
 2          RUUD
 3          PETER
 4          JAAP
 5          JOHAN
 6          BERT
 7          ADA
 8          JULIA
 9          KEES
10          CARLA
11          PAULINE
12          JOOS
13          JANNY
14          HENK
15          HENRY
16          GERT
17          GERDA
18          FRITS
19          TED
20          ROB
21          BEN
22          NICO
23          WILLY
24          LEO
25          BAS
26          ZZZZ
```

eerste item

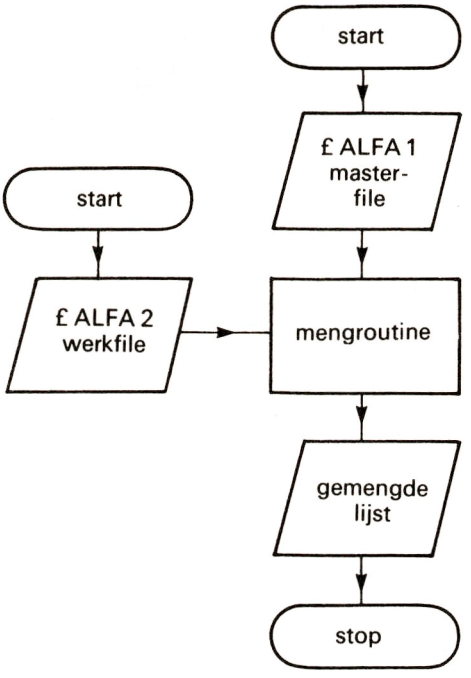
originele DATA 1-file

laatste item

Het mengprogramma

Een stroomdiagram in blokvorm voor het mengen van twee files is gegeven in afb. 9-5. De twee startblokken bestaan niet werkelijk, maar er zijn twee gescheiden beginpunten voor de twee files.

Essentieel voor de routine is dat beide files ALFA 1 EN ALFA 2 vooraf gesorteerd dienen te zijn, bijvoorbeeld met de eerder in dit hoofdstuk beschreven routines.



Afb. 9-5. Mengroutine in een blokschema.

De mengroutine
Het principe is:

Als het gegeven in de 'master-file' kleiner is dan het gegeven in de werkfile:
- schrijf het gegeven uit de 'master-file' in de nieuwe file.
- zo niet, schrijf het gegeven uit de werkfile in de nieuwe file.

```
10  REM ★★ LEZEN UIT EEN DATAFILE ★★
20  DIM M$(50) : DIM C$(50)
30  C = 1 : M$(C) = "AAAA"
40  INPUT "NAAM VAN DE DATA FI-
LE";G$
50  OPEN "I", £1, G$
60  IF EOF(1) THEN 100
70  C = C + 1
80  INPUT £1, M$(C)
90  GOTO 60
100 CLOSE £1
110 REM ★★★★★★★★★★
120 N = C + 1 : M$(N) = "ZZZZ"
130 REM ★★★★★★★★★★
140 FOR I = 1 TO N
150 PRINT I, M$(I)
160 NEXT I
```

C$ wordt nieuwe lijst

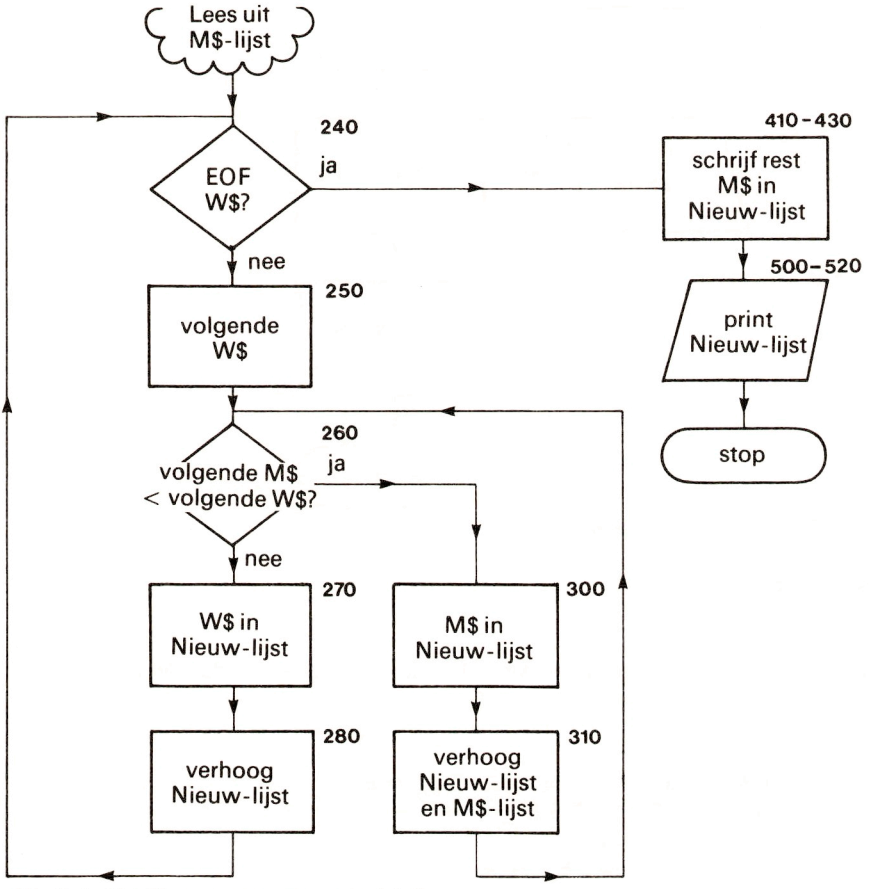
inlezen masterfile en toevoegen lijstextremen "AAAA" en "ZZZZ"

BBC: 80 INPUT #A, M$(C)
100 CLOSE #A

print masterfile

VERKLARING:

W$	gegeven uit werk-file
M$	gegeven uit master-file
Nieuw-lijst	nieuwe lijst waarin het geheel opgeslagen wordt (C$)



Afb. 9-6. Het file-mengprogramma in detail.

```
200 REM ★★ MENGEN ★★
210 I = 1 : K = 1
220 INPUT "NAAM VAN DE WERKFILE"; H$
230 OPEN "I", £1, H$
240 IF EOF(1) THEN 400
250 INPUT £1, W$
260 IF M$(I) < W$ THEN 300
270 C$(K) = W$
280 K = K + 1
```

BBC:

```
230 A = OPENIN(H$)
240 IF EOF#A = -1 THEN 400
250 INPUT #A, W$
```

mengroutine

```
290 GOTO 240
300 C$(K) = M$(I)
310 K = K + 1 : I = I + 1
320 GOTO 260
400 CLOSE £1
410 C$(K) = M$(I)
420 K = K + 1 : I = I + 1
430 IF I < = N THEN 410
440 REM ★★★★★★★★★★
500 FOR L = 2 TO K-2
510 PRINT L, C$(L)
520 NEXT L
530 END
```

400 CLOSE #A

k is één te hoog en we willen ZZZZ niet op de nieuwe file; daarom K-2

Programma 9-8 Het complete file-mengprogramma.

```
RUN
NAAM VAN DE DATA FILE ALFA 2
 1            AAAA
 2            BART
 3            ELLY
 4            LAMMIE
 5            LENIE
 6            TOM
 7            ZZZZ

NAAM VAN DE WERKFILE ALFA 1
 2            ADA
 3            BART
 4            BAS
 5            BEN
 6            BERT
 7            CARLA
 8            ELLY
 9            FRITS
10            GERDA
11            GERT
12            HENK
13            HENRY
14            JAAP
15            JANNY
16            JOHAN
17            JOOS
18            JULIA
19            KEES
20            LAMMIE
21            LENIE
22            LEO
23            NICO
24            PAULINE
25            PETER
```

```
26              ROB
27              RUUD
28              TED
29              TOM
30              WILLY

RUN
NAAM VAN DE DATA FILE ALFA 1
 1              AAAA
 2              ADA
 3              BAS
 4              BEN
 5              BERT
 6              CARLA
 7              FRITS
 8              GERDA
 9              GERT
10              HENK
11              HENRY
12              JAAP
13              JANNY
14              JOHAN
15              JOOS
16              JULIA
17              KEES
18              LEO
19              NICO
20              PAULINE
21              PETER
22              ROB
23              RUUD
24              TED
25              WILLY
26              ZZZZ
NAAM VAN DE WERKFILE ALFA 2
 2              ADA
 3              BART
 4              BAS
 5              BEN
 6              BERT
 7              CARLA
 8              ELLY
 9              FRITS
10              GERDA
11              GERT
12              HENK
13              HENRY
14              JAAP
```

15 JANNY
16 JOHAN
17 JOOS
18 JULIA
19 KEES
20 LAMMIE
21 LENIE
22 LEO
23 NICO
24 PAULINE
25 PETER
26 ROB
27 RUUD
28 TED
29 TOM
30 WILLY

We hebben het programma tweemaal laten draaien met beide te mengen files als masterfile. Zoals u ziet maakt dat geen werkelijk verschil. (Behalve dat als de masterfile de kleinste is, het minste beslag wordt gelegd op het geheugen.)

Als we ook de nieuwe lijst in een datafile willen plaatsen, kunnen we een routine aan het mengprogramma toevoegen in plaats van de regels 510-520.

9.8 Verwijderen uit files

Het verwijderen van elementen uit een file lijkt een totaal ander proces. Toch is dit met een geringe wijziging van het mengprogramma te verwezenlijken.

We gaan er weer van uit dat de te verwijderen gegevens alfabetisch zijn gerangschikt in een werkfile.

Het proces gaat als volgt:
- Als het gegeven in de masterfile gelijk is aan dat in de werkfile, kopieer het masterfile-gegeven dan NIET in de nieuw aan te maken file.
- Als het gegeven in de masterfile ongelijk is aan dat in de werkfile, kopieer het masterfile-gegeven dan wel in de nieuwe file.

Voor deze eenvoudige wijziging geven we geen stroomdiagram, omdat dit veel overeenkomst vertoont met het mengprogramma. We hopen dat u de wijzigingen kunt volgen aan de hand van de programmalisting. Een kleine gedachtenoefening zal u bevestigen dat het hier niet meer mogelijk is beide files te verwisselen.

Master-file (ALFA 1)	**Werkfile (ALFA 3)** (te wissen gegevens)
ADA	BEN
BAS	GERDA
BEN	JAAP

```
BERT                                    JULIA
CARLA                                   NICO
FRITS                                   RUUD
GERDA                                   TED
GERT
HENK
HENRY
JAAP
JANNY
JOHAN
JOOS
JULIA
KEES
LEO
NICO
PAULINE
PETER
ROB
RUUD
TED
WILLY
```

Afb. 9-7.

```
10  REM ★★ VERWIJDEREN UIT EEN DATAFILE ★★
20  DIM M$(50) : DIM C$(50)
30  C = 1 : M$(C) = "AAAA"
40  INPUT "NAAM VAN DE DATAFILE : "; G$
50  OPEN "I", £1, G$                    BBC:  50 B = OPENIN(G$)
60  IF EOF(1) THEN 100                        60 IF EOF#B = -1 THEN 100
70  C = C + 1
80  INPUT £1, M$(C)                           80 INPUT #B, M$(C)
90  GOTO 60
100 CLOSE £1                                 100 CLOSE #B
110 REM ★★★★★★★★★★
120 N = C + 1 : M$(N) = "ZZZZ"
130 REM ★★★★★★★★★★
140 FOR I = 1 TO N
150 PRINT I, M$(I)
160 NEXT I
200 REM ★★ WISSEN ★★
210 I = 1 : K = 1
220 INPUT "NAAM VAN DE WERKFILE: "; H$
230 OPEN "I", £1, H$                         230 B = OPENIN(H$)
240 IF EOF(1) THEN 400                       240 IF EOF#B = -1 THEN 400
250 INPUT £1, W$                             250 INPUT #B, W$
260 IF M$(I) = W$ THEN 330 ──────────────────► als mastergegeven =
                                               werkgegeven, doe niets
                                               en ga verder met
```

```
300 C$(K) = M$(I)
310 K = K + 1 :I = I + 1
320 GOTO 260
330 I = I + 1
340 GOTO 240
400 CLOSE £1
410 C$(K) = M$(I)
420 K = K + 1 :I = I + 1
430 IF I < = N THEN 410
440 REM **********
500 FOR L = 2 TO K-2
510 PRINT L, C$(L)
520 NEXT L
530 END
```

volgend mastergegeven

schrijf anders mastergegeven in nieuwe lijst

```
400 CLOSE #B
```

Programma 9-9 Gegevens uit een file vernietigen.

```
RUN
NAAM VAN DE DATAFILE: ALFA 1
 1            AAAA
 2            ADA
 3            BAS
 4            BEN
 5            BERT
 6            CARLA
 7            FRITS
 8            GERDA
 9            GERT
10            HENK
11            HENRY
12            JAAP
13            JANNY
14            JOHAN
15            JOOS
16            JULIA
17            KEES
18            LEO
19            NICO
20            PAULINE
21            PETER
22            ROB
23            RUUD
24            TED
25            WILLY
26            ZZZZ
NAAM VAN DE WERKFILE: ALFA 3
 2            ADA
 3            BAS
 4            BERT
 5            CARLA
```

originele file: ALFA 1

Naschrift

De behandelde technieken voor het verwerken van informatie in files waren eenvoudig en nogal grof. We gaven de voorkeur aan eenvoud boven elegantie.

U zult hebben opgemerkt dat we bij het lezen uit en het schrijven in files steeds één gegeven tegelijk verwerkten. Zowel het PRINT£ als INPUT£ statement kan méér dan één gegeven schrijven en lezen.

We hadden ook steeds maximaal één file open. Dat betekent dat alle file-informatie in het geheugen moet worden gelezen voordat een nieuwe file gecreëerd kan worden. Dit beperkt de maximale hoeveelheid gegevens tot dat wat in het geheugen past. Voor een grotere file moeten we het geheugen aanpassen. Een oplossing daartoe is het openen van twee files tegelijk. Terwijl de ene file geleegd wordt in het geheugen wordt de andere met bewerkte informatie uit het geheugen gevuld. Daartoe moeten we de beschikking hebben over twee cassetterecorders.

Ondanks deze beperkingen kunt u toch overweg met de belangrijkste basismethoden voor het werken met files.

Opgave 9

1. Schrijf een programma om een boekenuitleenfile te creëren met de namen der uitleners in alfabetische volgorde:

LENER LEENDATUM BOEKTITEL

(a) Schrijf een programma om nieuwe titels aan het bestand toe te voegen.
(b) Schrijf een programma om een listing te maken van de boeken die 'over tijd' zijn.
2. Schrijf een programma dat uit een tabel de som van de rijen en kolommen berekent en het totaal in een nieuwe file plaatst.

Samenvatting van hoofdstuk 9

Aan het einde van hoofdstuk 9 dient u de volgende onderwerpen te beheersen en in eenvoudige programma's te kunnen toepassen.

- Het maken van een datafile.
- Het lezen van een datafile.
- Het sorteren van een datafile.
- Het toevoegen of mengen van twee datafiles.
- Het verwijderen van gegevens uit een datafile.

Antwoorden op oefeningen

Oefening 1

```
1000 REM ★★ CREEER EEN DATAFILE ★★
1010 INPUT "NAAM VAN DE DATAFILE"; F$
1020 OPEN "O", £2, F$                          BBC: 1020 A = OPENOUT(F$)
1030 INPUT "VOLGENDE NAAM";N$
1040 IF N$ = "ZZZZ" THEN 1070
1050 PRINT £2, N$                                   1050 PRINT #A, N$
1060 GOTO 1030
1070 CLOSE £2                                       1070 CLOSE #A
1080 PRINT F$; "IS OPGESLAGEN"
1090 END
```

Programma 9-10.

```
RUN
NAAM DE DATAFILE DATA 1 ———— cassetterecorder op
VOLGENDE NAAM BERT            opname dan 'RETURN'
VOLGENDE NAAM GERT
VOLGENDE NAAM PETER
VOLGENDE NAAM HENK
VOLGENDE NAAM HENRY
VOLGENDE NAAM GERDA
VOLGENDE NAAM CARLA
VOLGENDE NAAM FRITS
VOLGENDE NAAM BEN
VOLGENDE NAAM ROB
VOLGENDE NAAM LEO
VOLGENDE NAAM ADA
VOLGENDE NAAM NICO
VOLGENDE NAAM PAULINE
VOLGENDE NAAM KEES
VOLGENDE NAAM JOOS
VOLGENDE NAAM JOHAN
VOLGENDE NAAM JAAP
VOLGENDE NAAM BAS
VOLGENDE NAAM JANNY
VOLGENDE NAAM JULIA
VOLGENDE NAAM ZZZZ
DATA 1 IS OPGESLAGEN
```

Oefening 2

```
10  REM ★★ LEZEN UIT EEN DATAFILE ★★
20  INPUT "NAAM VAN DE DATAFILE";G$
30  OPEN "O", £1, G$
40  IF EOF(1) THEN 80
50  INPUT £1, A$
60  PRINT A$
70  GOTO 40
80  CLOSE £1
90  END
```

```
BBC: 30 B = OPENIN(G$)
     40 IF EOF#B = -1 THEN 80
     50 INPUT #B, A$

     80 CLOSE #B
```

Programma 9-11.

```
RUN
NAAM VAN DE DATAFILE DATA 1
ROB
PETER
PAULINE
LEO
JOOS
JULIA
KEES
NICO
JAAP
HENK
HENRY
JOHAN
JANNY
FRITS
BERT
BAS
ADA
CARLA
BEN
GERDA
GERT
```

Oefening 3

```
10  REM ★★ LEZEN UIT EEN DATAFILE ★★
20  INPUT "NAAM VAN DE DATAFILE"; G$
30  OPEN "I", £1, G$
40  IF EOF(1) THEN 90
50  INPUT £1, A$
60  IF LEFT$(A$,1) <>"J" THEN 40
70  PRINT A$
80  GOTO 40
90  CLOSE £1
100 END
```

```
BBC: 30 B = OPENIN(G$)
     40 IF EOF#B = -1 THEN 90
     50 INPUT #B, A$

     80 CLOSE #B
```

Programma 9-12.

```
RUN
NAAM VAN DE DATAFILE DATA 1
JOOS
JULIA
JAAP
JOHAN
JANNY
```

Oefening 4

```
10  REM ★★ LEZEN UIT EEN DATAFILE ★★
20  DIM A$(50) ——————————————————— dimensioneer een lijst
30  C = 0 ———————————————————————— initialiseer index
40  INPUT "NAAM VAN DE DATAFILE"; G$    BBC: 50  B = OPENIN(G$)
50  OPEN "I", £1, G$                    60  IF EOF#B = -1 THEN 100
60  IF EOF(1) THEN 100                  70  INPUT #B, A$(C)
70  C = C + 1 ———————————————————— alléén als EOF niet
80  INPUT £1, A$(C)                         gevonden is
90  GOTO 60
100 CLOSE £1                            100 CLOSE #B
110 REM ★★ N IS HET AANTAL NAMEN IN DE LIJST ★★
120 N = C : REM ★★★★★★ ———————————— C is de totale lengte
130 FOR I = 1 TO N van de lijst
140 PRINT I, A$(I)
150 NEXT I
```

Programma 9-13.

```
RUN
NAAM VAN DE DATAFILE DATA 1

 1          ROB
 2          PETER
 3          PAULINE
 4          LEO
 5          JOOS
 6          JULIA
 7          KEES
 8          NICO
 9          JAAP
10          HENK
11          HENRY
12          JOHAN
13          JANNY
14          FRITS
15          BERT
16          BAS
17          ADA
18          CARLA
19          BEN
20          GERDA
21          GERT
```

Oefening 5

In plaats van de print-routine uit de regels 410-500 van programma 9-6, schrijven we een creatieroutine als volgt:

```
300 REM ★★ EINDE SORTEERROUTINE ★★
400 REM ★★★★★★★★
410 REM ★★ MAAK EEN DATAFILE ★★
420 INPUT "NAAM VAN DE DATAFILE"; F$
430 OPEN "O", £2, F$                          BBC: 430 A = OPENOUT(F$)
440 FOR P = 1 TO N
450 PRINT £2, X$(P)                                450 PRINT #A, X$(P)
460 NEXT P
470 CLOSE £2                                       470 CLOSE #A
480 PRINT F$; "IS OPGESLAGEN"
500 REM ★★★★★★
510 END
```

Programma 9-14.

```
RUN
NAAM VAN DE DATAFILE DATA 1
NAAM VAN DE DATAFILE ALFA 1 ———————— cassetterecorder op
ALFA 1 IS OPGESLAGEN                 opname dan 'RETURN'
```

Om te testen of het een en ander is geslaagd, draaien we het inleesprogramma voor ALFA 1:

```
RUN
NAAM VAN DE DATAFILE ALFA 1
GESORTEERDE LIJST
 1            ADA
 2            BAS
 3            BEN
 4            BERT
 5            CARLA
 6            FRITS
 7            GERDA
 8            GERT
 9            HENK
10            HENRY
11            JAAP
12            JANNY
13            JOHAN
14            JOOS
15            JULIA
16            KEES
17            LEO
18            NICO
19            PAULINE
20            PETER
```

```
21          ROB
22          RUUD
23          TED
24          WILLY
```

Oefening 6

```
10  REM ★★ ZOEKEN IN EEN DATAFILE ★★
20  DIM N$(50)
30  C = 0
40  INPUT "NAAM VAN DE DATAFILE"; G$
50  OPEN "I", £1, G$
60  IF EOF(1) THEN 100
70  C = C + 1
80  INPUT £1, N$(C)
90  GOTO 60
100 CLOSE £1
110 REM ★★ IS DE LENGTE VAN DE LIJST ★★
120 N = C : REM ★★★★★★
150 INPUT "GEZOCHTE NAAM"; Q$
200 REM ★★ START ZOEKEN ★★
220 L = 1 : H = N
230 IF H - L = 1 THEN 500
240 M = INT((L + H)/2)
260 IF Q$ = N$(M) THEN 230
270 IF Q$ < N$(M) THEN 300
280 L = M : GOTO 230
300 H = M : GOTO 230
320 REM ★★ EINDE ZOEKEN ★★
330 PRINT "JA"; Q$; "STAAT IN DE LIJST"
350 GOTO 600
500 PRINT Q$; "STAAT NIET IN DE LIJST"
600 PRINT "EINDE ZOEK PROGRAMMA"
610 END
```

inlezen

bisectie-zoekprogramma

printroutine

Programma 9-15.

```
BBC: 50   B = OPENIN(G$)
     60   IF EOF#B = -1 THEN 100
     80   INPUT #B, N$(C)
     100  CLOSE #B

RUN
NAAM VAN DE DATAFILE ALFA 1
GEZOCHT NAAM GERT
JA GERT STAAT IN DE LIJST
EINDE ZOEK PROGRAMMA

RUN
NAAM VAN DE DATAFILE ALFA 1
```

de gesorteerde datafile (we mogen hier i.v.m. de bisectie-routine geen ongesorteerde datafile gebruiken)

```
GEZOCHTE NAAM NICOLETTE
NICOLETTE STAAT NIET IN DE LIJST
EINDE ZOEK PROGRAMMA
```

Opmerking:
We hebben hier het inleesprogramma gecombineerd met het bisectie-zoekprogramma uit hoofdstuk 8 en dat functioneert. We moeten echter oppassen met de begin- en eindlimieten voor het bisectieprobleem en moeten van te voren de elementen "AAAA"en "ZZZZ"aanbrengen. Zie daartoe ook programma 9-8.

Oefening 7

```
10  REM ★★ LEES EEN DATAFILE ★★
20  DIM A$(30,30)
30  INPUT "AANTAL RIJEN EN KOLOMMEN : ";R,C
40  INPUT "NAAM VAN DE DATAFILE"; G$
50  OPEN "I", £1, G$                    BBC: 50  B = OPENIN(G$)
60  IF EOF(1) THEN 100                       60  IF EOF#B = -1 THEN 100
70  FOR I = 1 TO R                           inlezen
72  FOR J = 1 TO C                           in een tabel
80  INPUT £1, A$(I,J)                        80  INPUT #B, A$(I,J)
85  NEXT J
90  NEXT I
95  GOTO 60
100 CLOSE £1                                 100 CLOSE #B
110 REM ★★★★★★
120 REM ★★ PRINT DE TABEL ★★
130 FOR I = 1 TO R
140 FOR J = 1 TO C
150 PRINT TAB(10★(J-1)); A$(I,J);            printen van de tabel
160 NEXT J
170 PRINT
180 NEXT I
190 END
```

Programma 9-16.

```
RUN
AANTAL RIJEN EN KOLOMMEN: 6,4
NAAM VAN DE DATAFILE ALFA 1
ADA         BAS         BEN         BERT
CARLA       FRITS       GERDA       GERT
HENK        HENRY       JAAP        JANNY
JOHAN       JOOS        JULIA       KEES
LEO         NICO        PAULINE     PETER
ROB         RUUD        TED         WILLY
```

Trefwoordenlijst

O

P

R

S

T

U

V

W

Z

De BASIC-cursus in dit boek is gebaseerd op de door de Engelse BBC gegeven televisiecursus BASIC. Deze cursus sluit ook erg goed aan op de algemene computerinformatie die in de zo succesvolle uitgave 'het Computerboek' wordt gegeven.
De cursus is bestemd voor iedereen die de programmeertaal BASIC goed wil leren beheersen.
Daarvoor is het niet voldoende de instructies te kennen, men moet ze ook op de juiste gestructureerde wijze weten toe te passen.
Alleen dán zal men in staat zijn de meer ingewikkelde programma's te schrijven zonder daarbij het overzicht te verliezen.
De talloze zelfstudie-opgaven stellen ook hen die geen computer hebben in staat de cursus te volgen en te controleren of zij de stof hebben begrepen.
Wie wel over een computer beschikt, zal veel praktische ervaring opdoen met de oefeningen die ieder stukje theorie afsluiten.

ISBN 90 201 1680 0