DRAGON 32

Richard Wadman

En innføring i Basic-programmering

Til norsk ved
Ivar Jardar Aasen

Informasjonssystemer as

Rettelser

s. 121

Linjenummer 30 i første programeksempel og ny linje nr. 45:

```
30 PRINT @ 128, "";: INPUT "NAVN▽";N$
45 PRINT @ 160, ""
```

s. 123

Linjenummer 20:

```
20 CLEAR 20, 31999:START = 32000:ANTALL = 8
```

s. 130

FM Gal filmodus. Forsøk på INPUT fra en fil som er OPEN for utskrift, eller PRINT av data til en fil som er OPEN for inndata, eller bruk av CLOAD der CLOADM skal brukes (og omvendt).

DRAGON 32

Richard Wadman

En innføring i Basic-programmering

Til norsk ved
Ivar Jardar Aasen

Informasjonssystemer as

Redaktør og grafisk formgiver: Thorkild Christensen

Grunnskrift: Helvetica mager 9/11 Metroset
Trykt i offset hos Emil Moestue as, Oslo.

ISBN 82-991032-0-7

Innhold

Innledning

Denne boken er laget for dem som ønsker å programmere Dragon 32 mikrodatamaskin ved hjelp av programmeringsspråket Basic.

Programmeringsspråket Basic er et meget kraftig programmeringsspråk, men samtidig er det svært enkelt å lære. Det består av omkring 100 forskjellige setningstyper, noe som er mindre en 1% av det normale ordforrådet til en person.

Selv om programmering kan virke vanskelig til å begynne med, som en hver ny ting, så består det i grunnen bare av å løse en oppgave på en logisk måte, trinn for trinn. Hemmeligheten er å ta tiden til hjelp og sørge for at du forstår det forrige trinnet før du starter på det neste. Bli ikke bekymret om du gjør feil, det er en del av læreprosessen. Du ødelegger ikke datamaskinen ved å gjøre programmeringsfeil. Finn feilen, rett den og fortsett. Forsøk dine egne ideer. Gjør dine egne eksperimenter, for på den måten vil du raskt finne ut hva datamaskinen er god for. Tast inn og kjør hvert enkelt eksempel, ikke bare for å se hva programmet gjør, men hvorfor det gjør det.

Utstyr og tilkopling

Hva trenger du?

En komplett pakke Dragon 32 består av:
1. Dragon 32 mikromaskin
2. transformator
3. tilkoblingskabel for TV-apparatet
4. tilkoblingskabel for kassettspiller
5. lærebok i Basic

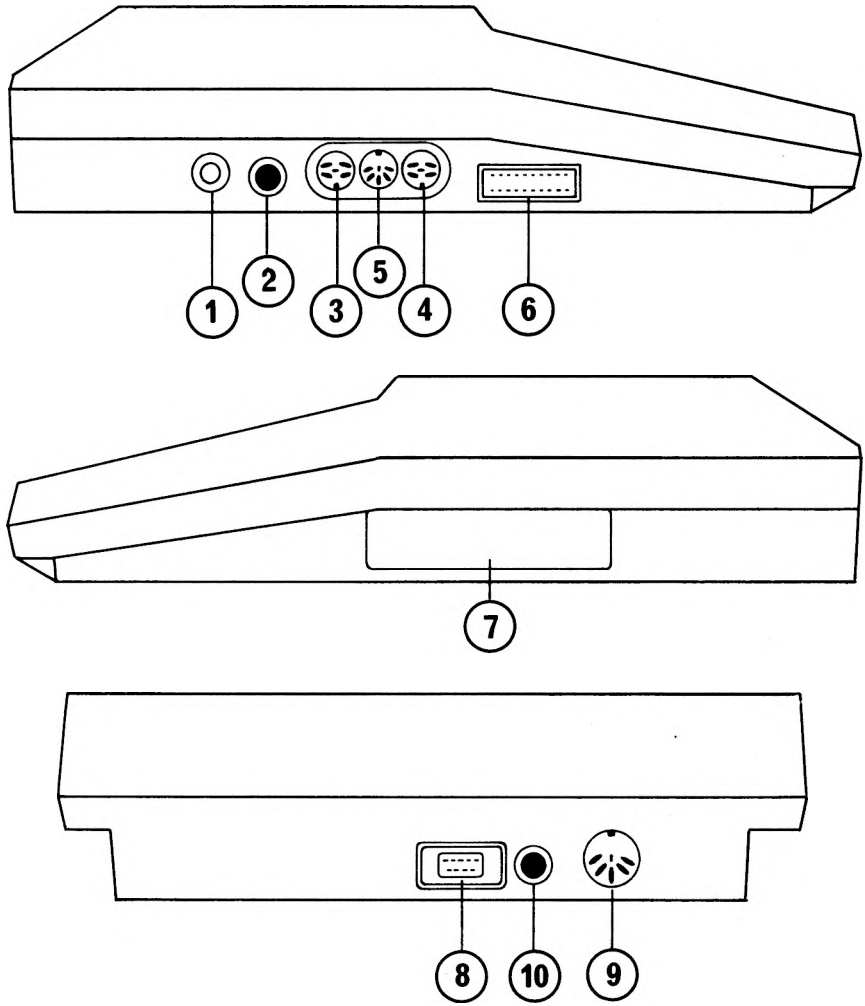
I tillegg til disse delene trenger du også et vanlig fjernsynsapparat. Dragon 32 mikrodatamaskin kan brukes enten sammen med et farge-TV eller et svart/hvitt TV. For å kunne dra nytte av den omfattende fargegrafikken til Dragon 32, må du imidlertid koble den til et farge-TV. Dette er alt som er nødvendig for å sette igang med Dragon 32.

Du kan forbedre mulighetene til datamaskinen ved å knytte til følgende enheter:
1. kassettspiller for å lagre program og data
2. skriver
3. styrespaker til bruk i spill
4. diskettenheter for lagring av program og data

Ingen av disse delene er nødvendige, men en kassettspiller vil spare deg for mye gjentatt skrivearbeid.

Oversikt over kontakter

1. TV-kontakten (TV) koples til fjernsynets standard antennekontakt med tilkoplingskabelen som følger med.
2. RESET-knappen stopper programmet som kjøres, og inn/ut-operasjoner umiddelbart. Et program som i øyeblikket er i datamaskinens lager vil fortsatt være til stede etter at Reset-knappen er trykket inn.
3. Venstre styrespak-kontakt (L.J.STK) 5 pinners DIN-kontakt brukt for å knytte til styrespak (joystick). Leveres som tilleggsutstyr.
4. Høyre styrespak-kontakt (R.J.STK) 5 pinners DIN-kontakt brukt for å knytte til styrespak (joystick). Leveres som tilleggsutstyr.



5. Kontakt for kassettspiller (TAPE) 5 pinners DIN-kontakt for tilknytting av kassettspiller. Kabel følger med Dragon som standard.
6. Parallell skriver-kontakt (P.I/O) for tilknytting av en skriver via en standard "centronics" kabel.
7. Åpning for program-modul brukes bl.a. for spillprogrammer. Modulen må *kun* stikkes inn og tas utnår datamaskinen er *avslått*.
8. Kontakt for transformator (TRANS.SUPPLY) for tilknytting av transformator som følger med Dragon 32.
9. Monitor-kontakt (MONITOR) for tilknytting av monitor.
10. På/av bryteren (ON/OFF) kontrollerer strømmen til datamaskinen.

Tilkopling av Dragon 32

1. Ta nå fjernsyns-kabelen og kople den til antenne-inngangen på fjernsynet og til kontakten (1) på Dragon 32.
2. Kople transformatoren til kontakten (8) på Dragon 32. Den andre ledningen fra hovedkrafttilførselen plugges til en kontakt i veggen.
3. Slå på fjernsynet og datamaskinen.
4. Du kan benytte datamaskinen med både farge- og svart/hvitt-fjernsyn, som har UHF-kanal (er). Noen eldre typer har ikke dette, og kan derfor ikke benyttes. Tilkopling skjer med den medfølgende ledningen. Den ene enden settes inn i kontakten merket TV på Dragon og den andre koples inn i fjernsynets antenne-inntak. Når maskinen og fjernsynet er slått på, kan du finne en ledig kanal på fjernsynet, og stille inn kanalvelgeren på UHF (ofte merket UHF eller U). Søk deretter fram til kanal 35, da skal skjermen vise en grønn firkant med en mørk bord rundt, eller en lysegrå firkant med en svart bord, hvis det er et svart/hvitt fjernsyn.
   I det grønne rektangelet vises følgende melding:

   (C) 1982 Dragon DATA LTD
   16K Basic INTERPRETER 1.0
   (C) 1982 BY MICROSOFT
   OK

Datamaskinen er nå klar til bruk.
Har du problem med dette, les igjennom fjernsynets bruksanvisning om stasjonsinnstilling (for UHF).

Bruk av program-moduler

Koble Dragon 32 til fjernsynet som ovenfor. Før du slår på datamaskinen stikker du modulen inn i åpningen for program-modulen (7) og sørger for at merkelappen på modulen vender opp. Pass på at *datamaskinen må være slått av* før du stikker inn eller fjerner program-modulen. Dersom du ikke har sørget for det, kan du skade både program-modulen og datamaskinen.

Bruk av styrespakene

Styrespaker er tilgjengelig for Dragon 32. Styrespakene er av potensiometertypen. De koples til styrespak-kontaktene (3 og 4).

Bruk av kassettspiller

Enhver kassettspiller av god kvalitet kan brukes for å lagre program og data fra Dragon 32. Kasserttspilleren må ha kontakter for fjernkontroll, høretelefon og mikrofon. Nødvendige kabler er tilgjengelig for mange standard kassettopptakere. Kassettspilleren koples til datamaskinen via kontakt (5). Tilkoplingene til kassettspilleren er avhengig av type kassettspiller. Se kapittel 4 for instruksjoner om hvordan en gjør bruk av kassettspilleren.

Hvordan ta vare på Dragon 32?

1. Sørg for at alle væsker holdes borte fra datamaskinen.
2. Sørg for at alle løse ledninger ligger ryddig og sikret.Snubling i en ledning kan bli kostbar.
3. Pass på at alle pluggene er godt festet i sine kontakter før du slår på maskinen.
4. Slå av alle brytere og kople fra maskinen når den ikke er i bruk.
5. Når du skal rengjøre maskinen og tastaturet, må du først kople maskinen fra strømtilførselen. Bruk en lett fuktet klut og tørk av maskinen og tastaturet. Bruk ikke noen form for rensevæske på spritbasis.

1. Sett i gang

Tastaturet

Du har nå koplet opp datamaskinen slik som instruksjonen sier og er klar til å starte. Slå på maskinen. Fjernsyns-skjermen skal nå ha en grønn firkant med melding (hvis ikke, slå av og kontroller alle koplingene igjen). Meldingen er avhengig av type maskin og om du har disketter tilknyttet osv. Her vil vi ikke bry oss nærmere om meldingen på skjermen. Den siste linjen er imidlertid den samme.

OK

OK er Dragons melding som forteller deg at den er klar til å motta instruksjoner. Du må vente på denne meldingen før du kan taste inn noe. Under OK er det en blinkende firkant. Denne kalles *markøren*. Den viser deg hvor du er på linjen.

Ta nå en titt på tastaturet Det ser ut som et skrivemaskintastatur med noen ekstra taster, og det oppfører seg som en skrivemaskin også. Trykk ned noen taster. Du ser tegnene komme fram på skjermen etter hvert som de blir tastet inn. Markøren beveger seg med og viser deg hvor langt du er kommet. Dersom du trykker SHIFT-tasten og 0 samtidig, og fortsetter å skrive, vil du se at bokstavene på skjermen har forandret seg til grønne på svart bakgrunn. Dette er små bokstaver (altså: a,b,c,d), men de vil bare opptre som små bokstaver på en *skriver*. Alle dine kommandoer eller instruksjoner til datamaskinen må være skrevet med store bokstaver.

Pass på at du ikke forveksler 0 (null) og O (bokstaven o). Trykk SHIFT-tasten og 0 samtidig en gang til og skriv litt mer. Du skal nå igjen være tilbake til svarte bokstaver på en grønn bakgrunn.

Prøv nå ←tasten. Dette er *tilbaketasten*. Når du trykker den vil markøren bevege seg tilbake langs linjen og bokstaven umiddelbart til venstre for den forsvinner. Dette er nyttig når man skal rette feil. Trykk tilbaketasten til du når den bokstaven som er feil og skriv på nytt.

For å blanke skjermen fullstendig trykker du CLEAR-tasten. Da blir alt fjernet fra skjermen og markøren beveger seg opp til venstre hjørne i øverste linje. CLEAR-tasten blanker bare skjermen og program som er lagret i datamaskinen blir ikke påvirket.

Øv deg litt på å bruke tastaturet. Finn ut hvor tastene er plassert og hvordan du retter feil. Blank så skjermen og forviss deg om at du er i modus for store bokstaver (svarte bokstaver på grønn bakgrunn).

Dragon som regnemaskin

Datamaskinen kan arbeide på to forskjellige måter:
Umiddelbart - den vil da adlyde kommandoen straks, og *utsatt* - den vil da lagre en mengde instruksjoner for så senere å utføre dem som et program. På den umiddelbare måten vil datamaskinen oppføre seg som en regnemaskin.

Datamaskinen forstår et språk som heter Basic. I Basic er det et antall spesielle ord som forteller datamaskinen hva den skal gjøre. F.eks. PRINT som betyr at den skal skrive det som følger på skjermen.

Forsøk selv, tast PRINT 12+7 og trykk ENTER-tasten. Skjermen skal da vise:

```
19
OK
```

Gjør et nytt forsøk. Tast PRINT 12+8/2 og trykk ENTER-tasten:

```
16
OK
```

Dersom du tastet noe feil har du sannsynligvis fått meldingen:

```
?SN ERROR
```

Dette betyr en *syntaksfeil.* Det vil si at datamaskinen ikke forstår det du har tastet. Vanligvis fordi det er blitt stavet feil. Datamaskinen gir feilmeldinger når den ikke forstår kommandoen, eller når den forstår kommandoen, men mener at den har blitt bedt om å gjøre noe som er ulogisk eller umulig.

Tast PRINT 3/0 og trykk så ENTER-tasten

Meldingen blir:

```
?/0 ERROR
```

Det betyr at du har forsøkt på å dividere med null, noe som er umulig.

Feilmeldingene er enkle for å spare plass i lageret, men en fullstendig liste med sannsynlige feilårsaker finnes i Tillegg C.

La oss gå tilbake til syntaksfeil. Datamaskinen er veldig nøye med stavemåten til sine spesielle ord i Basic. Dersom du ser en feil før du trykker ENTER-tasten, kan du bruke tilbaketasten (←) for å gå tilbake og rette den. Ellers er det foreløpig ikke noe annet alternativ enn å skrive om hele linjen korrekt.

Aritmetiske regler

Foreløpig i våre eksempler har vi bare bedt datamaskinen om å utføre to aritmetiske operasjoner. Disse er addisjon (+) og divisjon (/). Begrepet *operasjon* betyr noe vi ber datamaskinen om å gjøre. Det finnes seks enkle aritmetiske operasjoner som datamaskinen kan gjøre, og den har faste regler for hvordan disse utføres. I eksemplet vi så på tidligere:

PRINT 12 + 8/2

er det ikke klart om svaret skal være 16 eller 10.
12 pluss 8 dividert på to er 10, eller 8 dividert på 2 er 4 pluss 12 er 16.
Svaret gitt av datamaskinen er 16 på grunn av rekkefølgen den velger å utføre aritmetikken på. Operasjonene prioriteres slik:

1. Eksponensiering

Eksponensiere betyr opphøye i. 5 opphøyd i 4, (5^4), er 5 x 5 x 5 x 5.

PRINT 4+3↑2

beregnes ved å kvadrere tre (3 x 3 = 9) og så legge til 4 som gir en total på 13. Dersom det er flere enn en eksponensiering, blir de beregnet fra venstre mot høyre. Forsøk et eksempel:

PRINT 2↑3↑2↑3 (↑-tasten til venstre på tastaturet brukes for eksponensieringsoperasjonen).

Uttrykket beregnes ved å multiplisere 2 med seg selv 3 ganger (2 x 2 x 2 = 8 så multipliseres resultatet med seg selv (8 x 8 = 64), så multipliseres resultatet med seg selv 3 ganger (64 x 64 x 64 = 262144).

2. Negativt fortegn

Negativt fortegn er et minustegn som brukes for å angi et negativt tall.

PRINT -3+2

Datamaskinen vil først gjøre bruk av minustegnet på tallet. Slik at -3+2 utregnet blir -1. Dersom datamaskinen utførte addisjonen først -3+2 ville resultatet blitt -5, men det gjør den altså ikke.

3. Multiplikasjon

Symbolet som datamaskinen bruker for multiplikasjon er en stjerne (*). Dermed blir det ikke noen forveksling med bokstaven x. Legg merke til at du må holde SHIFT-tasten nede for å få *.

PRINT 5*2+3

blir beregnet til 13 (5 x 2 = 10, pluss 3 = 13).

4. Divisjon

Symbolet datamaskinen bruker for divisjon er /.

PRINT 5/2+3

blir beregnet til 5.5 (5 : 2 = 2,5, pluss 3 = 5,5).

Multiplikasjon og divisjon har samme prioritet. Når aritmetiske operatorer har samme prioritet, blir de utført fra venstre mot høyre.

PRINT 5+2*3+4/2

blir beregnet til 13 (2 * 3 = 6, 4 / 2 = 2, 5 + 6 + 2 = 13)

5. Addisjon

Symbolet for addisjon er +.

6. Subtraksjon

Symbolet for subtraksjon er -.

Addisjon og subtraksjon har samme prioritet, slik at de utføres fra venstre mot høyre etter at operasjonene med høyere prioritet er blitt utført.

Kort oppsummert er datamaskinens rekkefølge for å utføre matematiske operasjoner følgende:

1. ↑ (eksponensiering, fra venstre mot høyre)
2. - (minustegn brukt for å angi negativt tall)
3. * / (multiplikasjon og divisjon, fra venstre mot høyre)
4. + - (addisjon og subtraksjon, fra venstre mot høyre)

Nedenfor finner du noen aritmetiske uttrykk som skal beregnes. Forsøk å gjøre hvert enkelt av dem i hodet (eller med papir og blyant), og deretter på datamaskinen. Dersom ditt svar er forskjellig fra datamaskinens, forsøk å finne ut hvorfor. Har du ikke erfaring fra tidligere med hvordan datamaskinen beregner uttrykk, bør du absolutt gjøre disse eksemplene. De fleste såkalte datamaskinfeil skyldes programmerere som bruker andre regler enn datamaskinen. Taster du setningene inn nøyaktig slik som de er skrevet, vil datamaskinen gi deg de korrekte svarene.

```
PRINT 3 + 2
PRINT 4 + 6 - 2 + 1
PRINT 8 * 4
```

```
PRINT 4 ↑ 2 + 1
PRINT 5/4 - 1
PRINT 5 - 4/2
PRINT 6 * - 2 + 6/3 + 8
PRINT 4 + - 2
PRINT 2 * 2 + 3 * 4
PRINT 8/2/2/4
PRINT 20/2 * 5
PRINT 8 * 2/2 + 5 * 3 * 2 ↑ 2
```

Det er ikke nødvendig å taste PRINT hver gang. Det finnes et kortere symbol: ?. Dersom du taster:

? 3 + 2 er det det samme som å taste PRINT 3 + 2.

Nå bør du være fortrolig med å bruke ENTER-tasten til slutt på hver linje. OK forteller deg at datamaskinen er klar. ENTER-tasten forteller datamaskinen at du er ferdig.

Vi har nå sett på reglene for behandlingsrekkefølge, men det er også mulig å endre denne rekkefølgen. Det gjøres ved hjelp av parenteser. La oss anta at du ønsker å dividere 14 med 4 pluss 3. Skriver du 14/4 + 3 vil du få svaret 6.5, fordi du får 14 dividert med 4 og så lagt til 3. Men dette er ikke hva du ønsket. For å få til det du ønsket må du skrive:

14/ (4 + 3) og svaret blir 2.

Parentesene endrer behandlingsrekkefølgen. Regelen er enkel: Utfør alt inne i parentesen først. Er det flere parenteser inne i hverandre, utføres de innerste først.

12/ (3 + (1 + 2) ↑ 2) blir beregnet slik:
1. (1 + 2) utføres først:12/ (3 + 3 ↑ 2)
2. 3 ↑ 2 deretter:12/ (3 + 9)
3. (3 + 9) deretter: 12/12
4. Divisjonen til slutt: 1

Her er noen flere uttrykk å beregne. Dersom du ikke er vant med hvordan datamaskinen arbeider, bør du bruke de få minuttene som skal til for å arbeide gjennom eksemplene. Du kan da bruke datamaskinen mer effektivt senere.

```
? 44/ (2 + 2)
? (44/2) + 2
? 4 + (-5*2)
? 100/ (200/ (2* (9-5)))
? 42/ ( (9/3) + 1.75+ (5/4))
```

Skrive tekst

Så langt har vi bare brukt tall i våre PRINT-setninger. Vi ser ofte på datamaskinen som en *tallknuser*, men det er bare en av de funksjoner datamaskinen kan ha. En datamaskin kan også brukes for å behandle tegn. Med tegn mener vi bokstavene, tallene, puktum, komma og andre spesialtegn som vi vil møte senere. Basic tillater oss å behandle grupper av tegn, som vi kaller *strenger*.

En streng er en hvilken som helst blanding av tegn, selv blank, eller mellomrom (markert med ∇), kan være et viktig tegn i en streng.

For å fortelle datamaskinen at dette er en streng og ikke et tall, så er strengen omsluttet av anførselstegn. Noen eksempler på strenger er:

"TITTELEN", "Z+*?!", "PEDER AAS"
"DC34567", "02-466800"

De siste to strengene kan være et bilnummer og et telefonnummer. Selv om de begge inneholder numeriske tegn, kan vi ikke bruke disse sifrene til å gjøre noen ordentlige former for aritmetikk. Basic oppfatter ikke en samling sifre som er omsluttet av anførselstegn som tall. Strengen 12345 er fullstendig forskjellig fra tallet 12345. Tall og strenger lagres på forskjellige måter i datamaskinens lager.Forsøk å skrive:

```
PRINT"2 + 5 = " ; 2 + 5.
```

Den første delen av PRINT-setningen kommer fram på skjermen nøyaktig som skrevet i strengen. Anførselstegnene som omslutter strengen er ikke del av den. Den andre delen av setningen blir et numerisk uttrykk, slik at skjermen skulle vise:

```
2 + 5 = 7
```

Som vi allerede har sett, kan blank eller mellomrom være et viktig tegn i en streng. I Basic-setninger har mellomrom ingen betydning, de gjør kun den enkelte linje lettere å lese. I strenger derimot har de betydning. Når en streng skrives på skjermen, blir den kopiert nøyaktig som den gruppen tegn som finnes mellom anførselstegnene.

Du bør nå være i stand til å bruke datamaskinen for å løse noen enkle oppgaver som finnes til slutt i dette kapittelet. Nok en gang, dersom du ikke er vant med datamaskiner, forsøk i alle fall å løse noen av oppgavene.

Oppgaver

1. En planke er 168 cm lang. Hva er lengden målt i tommer?
   (1 tomme = 2.54 cm)
2. I en oppskrift går det med 0.75 kg mel. Hvor mange dl med mel går med?
   (1 kg = 16,5 dl)
3. En bil bruker 43,8 l bensin på en reise. Kilometerstanden ble lest av ved starten og slutten av reisen og var henholdsvis 22798 og 23411.
   Hvor mange liter bensin brukte bilen pr. mil?
4. Du setter inn 200 kroner på en sparekonto med 11% rente pr. år.
   Hvor mye har kontoen vokst til etter 5 år?
   ($K = B\ (1 + R/100)^N$, hvor B er beløpet som settes inn, R er rentefoten, N er antall år og K er kapitalen etter N år.)
5. Et sykkelhjul er 68 cm i diameter. Hvor mange omdreininger gjør hjulet i løpet av 1 km.
   (Omkretsen er 3.14159 x diameter)

Svar på oppgavene

1. 66.14 tommer
2. 12.375 dl
3. 0.715 liter/mil
4. Kroner 337.01
5. 468.1 omdreininger

2. Forklaringer av begreper

Konstanter

Alle eksemplene som vi har sett på hittil har kun inneholdt *konstanter*. En konstant er som navnet sier, noe som ikke forandrer seg. 3.145 er en konstant. Dersom vi forandrer det til 3.146 blir det en annen konstant. Konstanter er nyttige i datamaskinprogram, men de er ikke så nyttige som variabler.

Variabler

En variabel er noe som kan endre verdi. I ligningen:

X + 5 = Y

er X og Y variabler. Begge kan anta mange verdier, slik at ligningen fortsatt er gyldig. Variabler er steder i datamaskinens lager og kan best sammenlignes med en hylle i en stor reol. For å kunne identifisere hver enkelt av dem er det nødvendig å sette en "merkelapp" på dem (eller gi dem navn som X eller Y). Variablene i din datamaskin er av to typer: *Numerisk* eller *streng*. De finnes også i to "størrelser" *enkle* eller *tabell*. Vi vet allerede forskjellen mellom numeriske- og strengkonstanter. Selvfølgelig inneholder numeriske variable, numeriske konstanter (tall), og strengvariable har strengkonstanter (tegn). Foreløpig vil vi kun se på enkle variable. Tabellvariable vil vi behandle senere.

Variabelnavn

Navnet til en numerisk variabel kan være en hvilken som helst kombinasjon av en bokstav, og en bokstav eller tall.

N, AA, X, TI, Y, Z9, L5, BZ, PQ, K9

er alle eksempler på lovlige numeriske variabelnavn.

I virkeligheten vil Dragon tillate at variabelnavn har en vilkårlig lengde, men den vil kun gjenkjenne de første to tegnene i navnet. Altså, selv om den aksepterer navn som:

Numeriske variabelnavn

En numerisk variabel kan bare lagre tall.

Navnet til en numerisk variabel kan bestå av nesten hvilken som helst kombinasjon av bokstaver og tall, men må alltid starte med en bokstav.

Ettersom datamaskinen bare gjør bruk av de to første tegnene i et navn, vil navn som

STJERNE, STAT, STORTING

bli betraktet som den samme numeriske variabel (ST).

Variabelnavn som er lengre enn to tegn er nyttige fordi de gir en bedre beskrivelse av innholdet:

TALL, TELLER, SUM

vil alle bli akseptert som navn, men vil ta mer plass i lageret enn TA, TE, SU.

Reserverte ord som f.eks. IF er ikke tillatt brukt som variabelnavn.

Strengvariabelnavn

En strengvariabel kan bare inneholde strenger.

Et strengvariabelnavn kan bestå av en vilkårlig kombinasjon av bokstaver og tall, men må begynne med en bokstav og slutte med et $-tegn.

På samme måte som med numeriske variabelnavn, vil datamaskinen kun gjenkjenne de første to tegnene slik at:

SVAR$, SV1$, SV2$

alle vil bli betraktet som den samme variabelen, SV$.

BIL, BILDE, BILLETT

vil disse bli sett på som den samme variablen, BI. Den samme regelen gjelder for strengvariabler.

FRED$ FRISK$ FRIVOL$

blir alle oppfattet som FR$.

For å navngi en *streng*variabel, kan man gjøre bruk av de samme kombinasjonene, men navnet må ha et $-tegn til slutt.

A$, P7$, MN$, Z0$, FP$

er alle eksempler på gyldige navn på strengvariabler.
Merk at det ikke er tillatt å benytte variabelnavn som inneholder Basic-ord. F.eks. FORAN vil oppfattes som "FOR AN" og gi syntaksfeil.

Hvordan variabler får verdi

Hvordan gjør vi så bruk av disse variablene Tast inn følgende eksempel:

```
A = 5  (husk å trykke ENTER-tasten etter hver linje)
B = 2
C = A + B
D = D + 3
PRINT A,B,C,D  (pass på å taste kommaene)
```

Skjermen din skal nå vise:

```
5     2
7     3
```

Den første linjen betyr at variabelen A skal få verdien 5. Den neste linjen lagrer verdien 2 i variabelen B. Datamaskinen vet nøyaktig hvor disse stedene er i datamaskinens lager. Alt du har å gjøre er å angi navnet. Den tredje linjen sier finn verdiene lagret i variablene A og B, adder dem og plasser resultatet i en variabel som kalles C. Etter at datamaskinen har gjort dette, inneholder variablene A og B fortsatt de opprinnelige verdiene (5 og 2 i dette tilfellet) og C inneholder summen (7). Den fjerde linjen kan virke noe forvirrende for de som kan algebra. Dette kommer av at likhetstegnet (=) i Basic ikke betyr det samme som det gjør i matematikken. Likhetstegnet (=)i Basic betyr *å tilordne.* Det vil si å ta uttrykket på høyre side av likhetstegnet, eventuelt beregne det og plassere verdien i variabelen på venstre side av likhetstegnet.

Denne type programlinjer kalles for *tilordningssetninger.* På venstre side av en tilordningssetning, må det alltid finnes en variabel. Ligningsuttrykk som 2 = B + C kan gi mening i algebra, men det gir ingen mening innen Basic-språket.

For å vende tilbake til uttrykket D = D + 3, betyr det å ta den aktuelle verdien til variabelen D (D er null fordi vi ikke har gitt den noen annen verdi), og addere 3 til den og deretter legge verdien tilbake i variabelen D. (Eller sagt på en annen måte, øke verdien til variabelen D med 3). Dette kan virke en smule forvirrende, men det er en meget nyttig og meget vanlig type setning i datamaskinprogram. Det viser også et annet trekk ved variablene, de kan kun inneholde en verdi ad gangen. Hvis du tilordner en verdi til en variabel (dvs. den opptrer på venstre side av en tilordningssetning), vil verdien overskrive den gamle verdien og den gamle verdien går tapt. Du kan imidlertid kopiere

verdien i en variabel (enten inn i en annen variabel, eller ved å bruke den i uttrykk som C = A + B), så mange ganger som du måtte ønske uten at det forandrer den.Hvis du nå taster:

```
A = B  (husk å trykke på ENTER-tasten etter hver linje)
B = 17
D = D+2
PRINT A,B,C,D
```

vil du få følgende svar:

```
2      17
7       5
```

Variabelen A inneholder nå en kopi av B fra det forrige eksempelet. Variabelen B inneholder en ny verdi (17) og det tidligere innholdet i både A og B er tapt. Variabelen C er uforandret. Variabelen D er nå 5, fordi den inneholdt 3 fra det forrige eksempelet.

Strengvariabler oppfører seg på nøyaktig samme måte (bortsett fra at du må huske at navnet må slutte med et $-tegn). Forsøk dette eksempelet:

```
A$ = "DETTE ER EN"  (husk å trykke ENTER-tasten etter hver linje)
B$ = "MEGET"
C$ = "LANG STRENG"
D$ = A$ +B $+ B$ + B$ + B$ + B$ + B$
D$ = D$ + C$
PRINT D$
```

Skjermen vil nå vise:

```
DETTE ER EN MEGET MEGET MEGET ME
GET MEGET MEGET LANG STRENG
```

I linjene 1,2, og 3 har vi tilordnet verdier til strengvariablene A$, B$ og C$. I linje 4 legger vi seks kopier av B$ til A$. I forbindelse med strengvariablene betyr plusstegnet ikke det samme som i forbindelse med numeriske variable. Det betyr her sammenkjeding av tekststrengene. For dere som vil lære et vanskelig ord, så kalles dette konkatenering. I linje 5 adderte vi C$ til slutten av den sist konstruerte D$. På denne måten kan vi bruke datamaskinen til å konstruere setninger.

Strenger og tall kan ikke blandes

Pass på å holde numeriske og strengvariable fra hverandre. Kun tall kan lagres i numeriske variable og tilsvarende kan kun strenger lagres i strengvariable. Setninger av følgende type:

```
D = "STRENG"
A$ = 6
B = A$ * 2
```
vil gi feilmelding, ?TM ERROR (type mismatch error).

Plusstegnet (+) er den eneste aritmetiske operatoren som også kan brukes ved strenger og strengvariable. Alle andre aritmetiske operatorer (-, *, /, ↑), vil gi feilmelding.

Kommandobeskrivelser

I resten av boken vil vi presentere kommandoer i "rammer". Disse rammene vil gi detaljer angående hver kommando etter hvert som vi presenterer dem. På slutten av hver ramme er det et lite eksempel som demonstrerer bruken av kommandoen. Studer kommandobeskrivelsene nøye. Arbeid deg gjennom eksemplene og prøv å finne ut hva som vil skje før du lar datamaskinen utføre dem. Hensikten med disse programmene er å vise hvordan hver enkelt setning fungerer, men ofte inneholder de nyttige tips som du kan gjøre bruk av i dine egne programmer senere.

3. Lag et program

Hittil har datamaskinen din gjort lite, bortsett fra å skrive ut det du nettopp har tastet inn. Nå skal vi begynne å konstruere et datamaskinprogram.

Et program er en mengde instruksjoner som forteller datamaskinen hva den skal gjøre. Et Basic-program består av et antall linjer. En linje har to deler: Et linjenummer og en eller flere setninger. Dersom det er flere enn en setning på en linje, må hver setning skilles med et kolon (:).

```
PRINT A$: A = 47
```

Her har du et Basic-program:

```
10 CLS
20 PRINT"HVA HETER DU"
30 INPUT NAVN$
40 I = RND (255): J = RND (9) - 1
50 CLS J
60 PRINT@ 200+J,NAVN$;
70 SOUND I,2
80 GOTO 40
```

Som du kan se av dette eksempelet, inneholder programmet noen nye Basic-kommandoer. Bry deg ikke om disse foreløpig. De blir forklart senere. Legg merke til formen på Basic-programmet: En rekke linjer som hver inneholder et linjenummer og minst en setning (linje 40 har to).

Legg inn et program

Før vi kan legge inn et program i datamaskinens lager, må vi først fjerne det som måtte være der fra tidligere. Vi gjør dette ved å taste NEW og så trykke på ENTER-tasten. Skriv så hver linje nøyaktig som ovenfor og trykk ENTER-tasten til slutt på hver linje. Du vil legge merke til at etter at du trykker ENTER-tasten, skjer det ingen ting. En linje som begynner med et tall blir ikke utført umiddelbart, men blir lagret. Når et program utføres, starter det med det laveste linjenummeret og utfører så linje for linje etter verdien av linjenummerne. Fordi rekkefølgen i et program er avhengig av linjenummerne,

kan du skrive linjene i hvilken rekkefølge du ønsker. Datamaskinen sørger for å få dem i en en korrekt rekkefølge.

Forsøk å taste programmet inn i datamaskinen. Gjør du en feil før du har trykket ENTER-tasten, kan du gjøre bruk av rettetasten, (←). Oppdager du feilen først etter at du har trykket ENTER-tasten, så skriv hele linjen om igjen. Datamaskinen vil kun lagre den siste versjonen av linjen.

Når du har skrevet hele programmet, kan du taste LIST og trykke på ENTER-tasten, for å se hvordan datamaskinen har tatt imot og lagret programsetningene du har skrevet. Legg merke til at det ikke er noe tall foran LIST. En kommando uten et tall foran vil datamaskinen utføre umiddelbart. Undersøk programmet og forviss deg om at det er korrekt, og rett linjer som du oppdager feil i. Feil rettes ved å skrive hele linjen om igjen.

Nå er du endelig klar til å kjøre programmet. For å kunne kjøre det må du skrive RUN og trykke på ENTER-tasten.

Skjermen vil bli blanket og en melding vil komme fram på toppen av skjermen, som spør etter ditt navn. Tast navnet ditt og trykk ENTER-tasten.

Datamaskinen vil nå sette igang.

Skjermen vil lyse opp i forskjellige farger og det vil se ut som navnet ditt hopper omkring i midten av skjermen. Merkelige lyder vil også følge med i denne aktiviteten (hvis du har husket å skru opp volumkontrollen på TV-apparatet). Dette vil fortsette i det uendelige, dersom du ikke stopper det. En måte å stoppe programmet på, er å slå av strømmen. Dette er ikke noen hensiktsmessig måte å gjøre det på. Ved å gjøre det vil du nemlig miste programmet ditt. Den beste måten å stoppe programmet på er å trykke BREAK-tasten. En annen mulighet er å trykke RESET-knappen. Programmet vil i begge tilfeller fremdeles ligge i datamaskinens lager. Prøv å stoppe programmet på begge måter etter at det er startet med RUN (og ENTER-tasten).

Trinn for trinn

Nå har du altså sett hva programmet gjør, og vi skal nå fortelle hvordan du gjør det linje for linje.

10 CLS

Etter som denne programlinjen har lavest nummer, blir den utført først. Kommandoen CLS betyr blank skjermen og setter bakgrunnsfargen til den vanlige fargen (vanlig farge er grønn).

20 PRINT"HVA HETER DU"

Dette er PRINT-setningen som vi har sett tidligere. Denne linjen skriver meldingen øverst på skjermen.

LIST

LIST-kommandoen viser på skjermen det programmet som i øyeblikket befinner seg i lageret. LIST-kommandoen er ikke en del av programmet og har ikke noe linjenummer foran seg.

Er programmet for langt til å få plass på skjermen, kan listingen bli stoppet ved å trykke SHIFT-tasten og @-tasten samtidig, men da må du være rask. Listingen kan startes igjen ved å trykke på en hvilken som helst tast på tastaturet (unntatt SHIFT-, BREAK- og CLEAR-tastene).

For å liste kun en del av programmet kan du bruke:

LIST n_1-n_2

hvor n_1 og n_2 er to linjenumre og hvor n_2 må være større enn n_1.

LIST 40-100

vil vise alle programlinjene mellom linjenummer 40 og linjenummer 100.

LIST -80

vil vise alle programlinjene fra starten på programmet til og med linjenummer 80.

LIST 120-

vil vise alle programlinjene fra og med linjenummer 120 til slutten av programmet.

RUN

RUN-kommandoen brukes for å starte et program. Den har ikke noe linjenummer foran seg.

Dersom du ønsker å starte et program fra et annet sted enn begynnelsen, kan du også gjøre det ved å taste:

RUN *linjenummer*

hvor *linjenummer* er nummeret på den linjen hvor du vil starte.

RUN 250

30 INPUT NAVN$

Kommandoen INPUT forteller datamaskinen at den skal stoppe og vente til du har skrevet inn noe. Det du skriver inn vil den plassere i variabelen som følger etter kommandoen. Husk, datamaskinen vil bruke NA$ som navn i dette tilfellet og overse de andre bokstavene. Det er imidlertid ofte nyttig å bruke variabelnavn som er lengre for å gjøre programmet lettere å forstå. Et lengre navn fungerer altså som en bedre påminnelse om hva variabelen blir bruk til.

40 I = RND (255): J = RND (9)-1

Linje 40 viser hvordan flere setninger kan forekomme på en linje. Legg merke til at kolon (:) skiller dem. Denne linjen viser også en ny funksjon, RND. Denne kommandoen gir et slumptall. Et slumptall eller vilkårlig tall fungerer på samme måte som man trekker et lodd. Tallet i parantes etter RND, forteller datamaskinen hvilket tallområde den skal velge fra. I den første setningen I = RND (255), betyr det at den skal velge et tilfeldig tall i området 1 til 255 og så plassere dette tallet i variabelen I. I den andre setningen er tallområdet fra 1 til 9, men etter at tallet er blitt valgt ut, skal 1 trekkes fra før det legges inn i J. Dette betyr at J vil bli et tall mellom 0 og 8. På denne måten har en fått et tall i det området en ønsker.

50 CLS J

Denne linjen blanker skjermen. Denne gangen er imidlertid bakgrunnsfargen avhengig av verdien J. Det er ni farger tilgjengelig som er nummerert fra 0 til 8. Dette er den linjen som gjør at skjermen lyser opp med forskjellige farger.

60 PRINT@200+J,NAVN$;

Dette er en mer avansert versjon av den tidligere nevnte PRINT-setningen. Den instruerer datamaskinen til å skrive verdien av NAVN$ (i dette tilfellet ditt navn), og at det skal starte på et spesifisert sted på skjermen. Posisjonen her er 200+J. Det er et eller annet sted mellom 200 og 208. Posisjon 200 er på linje 7 og 8 plasser innover (les rammen som inneholder PRINT@ for å se hvordan skjermen er delt opp). Fordi verdien av J forandrer seg, vil navnet ditt se ut som det hopper fram og tilbake på linjen.

70 SOUND I,2

Dette er linjen som forårsaker de merkelige lydene. SOUND-setningen ber datamaskinen bruke tonegeneratoren for å lage lyd. Hvilken lyd og hvor lenge bestemmes av de to tallene som følger etter kommandoen.

80 GOTO 40

GOTO-setningen betyr ganske enkelt hopp til linjenummeret som følger etter kommandoen (40 i dette programmet).

NEW

Kommandoen NEW sletter lageret og setter alle variablene lik null. Legg merke til at kommandoen ikke skal ha noe linjenummer.

Det er en god regel å taste NEW før du skriver inn et program for å forsikre deg om at ikke noe av det gamle programmet finnes i lageret og kan ødelegge for det nye programmet.

Tilordningssetning

Tilordningssetningen brukes for å få en verdi inn i en variabel. Formatet til tilordningssetningen er:

LET *variabel* = *uttrykk*

Ordet LET i tilordningssetningen er en del av standard Basic, men det er ikke nødvendig å skrive det på Dragon. Derfor vil du ikke se ordet LET i noen av programlistene i denne boken.

Variabel kan være hvilket som helst variabelnavn.

Uttrykk kan være en konstant, en variabel eller en blanding av konstanter og variable som er knyttet sammen ved hjelp av operatorer (+ ,–,*,/,etc.). Ettersom streng og numeriske variabler ikke kan blandes, må både variabler og uttrykk være av samme type.

Legg spesielt merke til at likhetstegnet (=) ikke betyr det samme som likhetstegnet i algebraen. I Basic er det riktig å oppfatte det som "tilordnes" eller "settes lik". Det betyr at den samme variabelen kan forekomme på begge sider av likhetstegnet i en tilordningssetning som i:

```
40 X = X + 1
```

som forteller datamaskinen at den skal addere 1 til den aktuelle verdien til X og legge den nye verdien tilbake i X. Setningen betyr altså at verdien til X skal økes med 1.

```
10 S = 0: N = 0: CLS 5
20 PRINT @ 70,"TAST INN ET TALL";
30 INPUT X: CLS 5
40 S = S + X: N = N + 1
50 PRINT @ 198,"DU HAR SKREVET";N;"TALL";
60 PRINT @ 262,"MIDDELVERDIEN ER";S/N;
70 GOTO 20
```

Programmet vil nå vende tilbake til linje 40, hvor det velger et nytt tilfeldig tall for I og også for J. Ettersom bakgrunnsfargen bestemmes av J, vil den normalt forandres når programmet når linje 50, og ettersom I også er forskjellig, vil lyden forandres i linje 70. Når programmet når linje 80, vil det igjen gå tilbake til linje 40 hvor det på ny ..... (fortsetter i all evighet, eller til du trykker BREAK-tasten eller RESET-knappen).

Gjør endringer

Det var det første programmet. Det gjorde ikke noe særlig nyttig, men det var en start. Det fikk oss i gang og vi lærte noen nye setninger. En mer detaljert forklaring på de enkelte setningstypene, finner du i rammen for den enkelte setning. (Hver ramme inneholder også et lite program for å demonstrere setningen. Du bør forsøke å kjøre disse.)

Dersom du ikke var fornøyd med det første programmet, ønsker du kanskje å gjøre noen forandringer. Det å forandre et program som du allerede har tastet inn er enkelt. Fordi et Basic-program er ordnet på linjenummer, kan en linje i et program endres ved å skrive inn det samme linjenummeret på ny. Den nye linjen erstatter da den gamle. Forsøk å skrive inn en ny linje 70.

```
70 SOUND I,K  (Husk å trykke ENTER-tasten)
```

En ny linje kan legges inn i programmet ved å gi det et linjenummer som plasserer det på riktig sted. Forsøk å skrive:

```
45 K = RND (20)
```

Ettersom denne linjen er nummerert 45, vil den bli plassert mellom linje 40 og linje 50.

(Når du skriver dine egne program, kan du angi linjenummer mellom 0 og 63999. Det er vanligvis en god ide å nummerere linjene i trinn på 10. Altså, skriver du 10, 20, 30 vil du ha plass til å legge inn ekstra linjer om det blir nødvendig.)

Forsøk å kjøre det endrede programmet (tast RUN). Den nye linjen 45 velger et annet tilfeldig tall, denne gang mellom 1 og 20. Den endrede linjen 70, gjør bruk av dette tilfeldige tallet, (i variabelen K), for å endre varigheten av lyden.

PRINT

PRINT-setningen brukes for å vise resultatene. Den kan brukes for å skrive ut konstanter, verdien til variable, strenger og også til å beregne uttrykk.

Finnes det flere enn et dataelement i en PRINT-setning, må elementene være adskilt ved enten komma (,) eller semikolon (;).

Komma vil forårsake at utdata blir skrevet i to kolonner, hver 15 tegn bred. Dersom lengden til det første dataelementet er mer enn 15 tegn, blir det skrevet fullt ut og neste dataelement kommer på neste linje.

Semikolon forårsaker at utdata blir sammentrykket. Strenger blir skrevet helt inntil hverandre og numeriske elementer har en blank på hver side. Semikolon holder markøren i den siste posisjonen ferdig til å fortsette skrivingen når programmet når neste PRINT-setning.

En PRINT-setning uten dataelementer skriver en blank linje.

```
10 CLS
20 PRINT
30 PRINT"▽▽▽▽▽▽▽▽PRINT-SETNINGEN"
40 PRINT"▽▽▽▽▽▽▽▽EN  DEMONSTRASJON"
50 PRINT
60 PRINT"KOLONNE EN","KOLONNE TO"
70 PRINT 14.2,13.7
80 PRINT 1,2,7,11
90 PRINT
100 A$ = "SAMMENTRYKT": B = 3
110 PRINT,"ANDRE LINJE"
130 PRINT A$;"UTSKRIFT, LINJE"; B
140 PRINT
150 PRINT"DETTE VIL VISES";
160 PRINT"▽SOM EN LINJE"
170 PRINT"▽▽DEMONSTRASJONEN",
180 PRINT"▽▽AVSLUTTET"
```

INPUT

Når et program kommer til en INPUT-setning, stopper det og venter på at data blir tastet inn fra tastaturet. Etter ordet INPUT må det være en eller flere variabelnavn, adskilt med komma.

```
25 INPUT A,B,F$,H7
```

Setningen over krever at du taster inn 4 dataelementer. Det kan gjøres ved å trykke ENTER-tasten etter hvert av dataelementene eller at dataelementene tastes inn som en liste adskilt med komma som eksempelet viser:

146.2,78.1,STRENG,3 [ENTER] Det er en god regel alltid å skrive en melding før en INPUT-setning. Den bør angi hva som skal tastes inn. Du kan gjøre det ved hjelp av en PRINT-setning eller i selve INPUT-setningen på følgende måte:

```
35 INPUT"TAST TO TALL"; N1,N2
```

Legg merke til at semikolon brukes til å skille mellom strengen og listen med inndatavariabler.

Du må forsikre deg om at du taster inn riktig datatype (strenger til strengvariable og tall til tallvariable). Gjør du ikke det, vil programmet stoppe med feilmeldingen ?REDO og du må taste inn verdier på nytt.

Det er ikke nødvendig å bruke apostrofer når du taster inn verdier til en strengvariabel idet både "STRENG" og STRENG blir akseptert.

I det følgende demonstrasjonsprogrammet må du legge merke til følgende:

1. Linjene 100 og 170 gjør bruk av INPUT-kommandoen for å stoppe programmet til du er klar. A$ vil ikke inneholde noe.
2. Bruken av strengvariablene C$,P$ og T$ for å unngå å taste den samme meldingen flere ganger.

```
10 CLS : T$ = "DETTE ER EN INNDATADEMONSTRASJON"
20 P$ = "TRYKK ENTER FOR AA FORTSETTE "
30 C$ = "TAST INN 4 TALL,"
40 PRINT: PRINT T$: PRINT
50 PRINT C$;"TRYKK"
60 PRINT"ENTER-TASTEN ETTER HVERT TALL"
70 INPUT A,B,C,D
80 PRINT: PRINT"DU TASTET INN DISSE VERDIENE:"
90 PRINT A;B;C;D: PRINT
100 PRINT P$: INPUT A$: CLS
110 PRINT: PRINT T$: PRINT
120 PRINT"NAA▽";C$;"▽ADSKILT"
130 PRINT"AV KOMMA OG TRYKK ENTER"
```

```
140 INPUT A,B,C,D
150 PRINT: PRINT"DENNE GANGEN VAR TALLENE:"
160 PRINT A;B;C;D
170 PRINT: PRINT P$: INPUT A$: CLS
180 PRINT: PRINT T$: PRINT
190 INPUT"TAST INN EN STRENG OG ET TALL";B$,N
200 PRINT: PRINT"STRENGEN DU TASTET VAR:";PRINT
210 PRINT B$
220 PRINT: PRINT"OG TALLET VAR:";N
230 PRINT: PRINT"SLUTT PAA DEMONSTRASJONEN"
```

SOUND

SOUND-setningen genererer en tone med angitt tonehøyde og varighet. Den krever to argumenter:

SOUND *tone,varighet*

Tone er et tall mellom 1 og 255. Den laveste tonen er 1 og den høyeste 255. En tone som tilsvarer midlere C på piano får du med *tone* = 89. *Varighet* kan være et hvilket som helst tall mellom 1 og 255. *Varighet* = 16, gir en tone med varighet på ca. 1 sekund.

```
10 CLS
20 PRINT @ 7,"LYD-DEMONSTRASJON"
30 PRINT @ 64,"TAST ET TALL MELLOM 1 OG 255,";
40 PRINT @ 96,"SOM ANGIR TONEHOEYDEN";: INPUT H
50 PRINT @ 192,"TAST ET TALL FOR TONEVARIGHETEN";
60 INPUT V: CLS (RND (9)-1)
70 SOUND H,V
80 GOTO 10
```

Programkonstruksjon

Selv om det er lett å sitte ved tastaturet og skrive programlinjene, vil vanskelighetene vise seg etter en stund. Særlig når programmene begynner å bli lange. Det å taste inn programmet er den siste delen av det å lage et program.

Start med papir og blyant og skriv så ned hva du har til hensikt å gjøre. Del oppgaven i klart avgrensede deler. Mange oppgaver som løses ved hjelp av datamaskin, lar seg lett dele i minst tre deler:

1. forberedelse, overskrifter og instrukser for inndata
2. beregninger
3. presentasjon av resultater

Ta så for deg hver del for seg og del opp videre i mindre deler til du har kommet fram med enkle handlinger som "legg 1 til telleren". Legg disse handlingene i en fornuftig rekkefølge og skriv dem ned. Start så med den neste delen. Når du er ferdig med alle delene har du en rekke trinn (noen av dem kan være meget primitive), i hver seksjon (del). Alle som leser gjennom resultatet, bør nå være i stand til å løse oppgaven, hva den nå enn måtte være, ved kun å bruke enkel aritmetikk. Denne beskrivelsen av løsningen kalles en *algoritme* i datamaskinterminologi. Alt det som nå gjenstår er å omforme din plan til et språk som datamaskinen kan forstå. Ideelt sett kan hvert trinn i din plan oversettes med en linje i et Basic-program.

Etter oversettelse til Basic, kan du teste hver seksjon for seg for å forsikre deg om at den gjør det den skal gjøre, før du samler hele programmet. Når programmet kjører som det skal, må du ikke feire dette med å brenne opp alle papirene som ligger til grunn for programmet. Ta vare på den endelige planen, algoritmen. Deler av den kan vise seg å være nyttige for andre program og ikke minst kan det dukke opp feil lang tid senere når du har glemt hva du gjorde. Det er også nyttig å legge kommentarer inn i selve programmet for lettere å huske hva som skjer. For å kunne gjøre dette har Basic REM-setningen. Denne setningen gjør i virkeligheten ngen ting. Basic ignorerer alt etter REM-setningen. Hvis det finnes mer enn en setning på en linje, må REM-setningen være den siste på linjen. Det finnes også en kort versjon av REM-setningen som gjør bruk av en enkel apostrof (’). Du finner tegnet på tastaturet.

```
10 REM PROGRAM FOR BEREGNING AV MIDDELVERDI
140 A = B * C: GOTO 15: REM TILBAKE TIL START
48 K = K + 1: 'TRINNTELLER
```

Selv om REM-setningene tar noe plass i lageret er det uklokt å prøve å greie seg uten dem. Forsøker du å forstå ditt eget program et år etter at du har skrevet det, vil du oppleve at det er meget frustrerende dersom det ikke finnes noen kommentarer i det hele tatt.

Selv om denne delen av boken kan virke detaljert, er det et faktum at programerere bruker mer tid på å finne feil i program enn på å skrive selve programmet. Arbeider du på en systematisk måte som vi har foreslått her, er det en mindre sannsynlighet for feil i programmet og de vil også være lettere å finne.

PRINT @

PRINT @ -setningen brukes for å plassere utdata på et angitt sted på skjermen. For dette formålet er skjermen delt inn i et rutenett på 16 x 32, som gir 512 posisjoner. PRINT @ -rutenettet viser hvordan posisjonene er nummerert. Formatet til PRINT @ -setningen er:

PRINT @ *uttrykk, utskriftsliste*

Uttrykk kan enten være et tall, en variabel eller et aritmetisk uttrykk med verdi fra 0 til 511.

Utskriftsliste er det samme som brukes i den vanlige PRINT-setningen. Det kan være tall, variabler, strenger eller uttrykk som er adskilt med komma eller semikolon.

Betrakter du skjermen som bestående av 16 linjer, så vil uttrykket:

PRINT @ 32 * (LINJE – 1),A

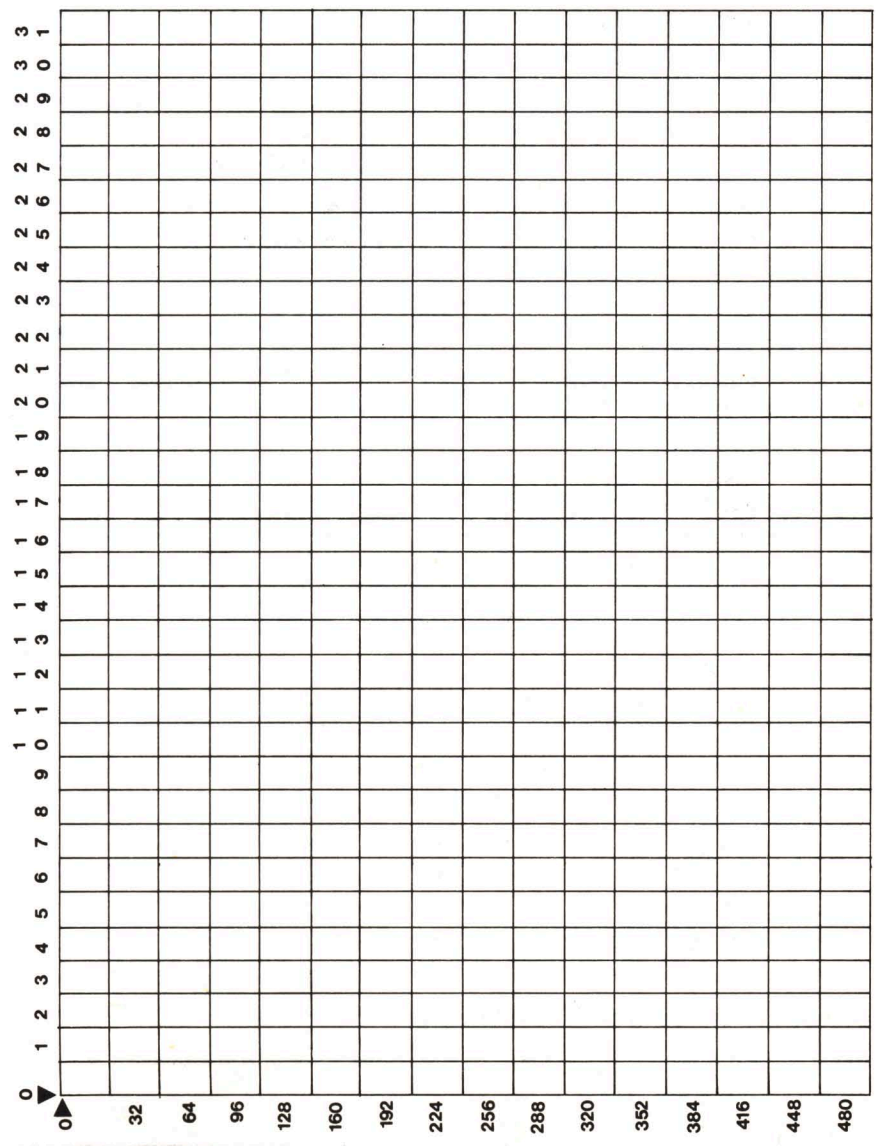
skrive verdien til A i begynnelsen av en linje på skjermen. Hvilken linje er avhengig av verdien til variabelen LINJE, som må være et tall mellom 1 og 16.

Det følgende eksempelet er en, med hensikt forvirrende, måte å produsere et ganske enkelt resultat. Forsøk å finne ut hva programmet gjør, kjør programmet og skriv så en enklere og mer rett fram versjon for å få fram samme resultat.

Legg merke til bruken av semikolon ved slutten av setninger for å forhindre at resten av linjen blir visket ut. (Dersom du ikke tror det, forsøk å taste inn linje 110 uten semikolon til slutt.)

```
10 CLS : P$ = "PRINT @ ": N = 1
20 RAD = 12: A$ = "PAA SKJERMEN"
30 PRINT @ 32 * (RAD – 1) + 15,"FOR AA PLASSERE";
40 PRINT @ 447 + 12,"OVERALT";
50 PRINT @ 259,"DETTE"; : PRINT @ 265,"VISER";
60 PRINT @ 32 * (RAD – 1) + 8,"BRUKES"; : PRINT @ 13,"SIDE";
70 PRINT @ 447 + 20, A$;
80 PRINT @ 271,"HVORDAN∇"; P$
90 PRINT @ 32 * (RAD – 1) + 4,"KAN";
100 PRINT @ 134,P$;"∇DEMONSTRASJON";
110 PRINT @ 450,"ELEMENTER";
120 PRINT @ 18,N
130 GOTO 130
```

Den siste linjen (130) setter datamaskinen i en "evig sløyfe", som ikke gjør noen ting. Sløyfen forårsaker at OK ikke kommer opp på skjermen. Trykk BREAK-tasten for å stoppe programmet.



CLS

CLS -setningen brukes for å slette skjermen og sette bakgrunnsfargen. Den normale bakgrunnsfargen er grønn. Bruker du bare CLS blir bakgrunnsfargen grønn.

For å forandre bakgrunnsfargen føyer du til et tall fra 0 til 8 etter CLS-setningen, (CLS 2).

Fargene er:

0-Svart	1-Grønn	2-Gul
3-Blå	4-Rød	5-Hvit
6-Turkis	7-Fiolett	8-Oransje

Den virkelige fargetonen vil avhenge av ditt fjernsynsapparat.

Du vil imidlertid legge merke til at hva du enn velger som bakgrunnsfarge, vil datamaskinen alltid skrive tegnene svarte eller grønne.

```
10 CLS
20 PRINT @ 0,"BAKGRUNNSFARGE-DEMONSTRASJON";
30 PRINT @ 192,"TAST ET TALL MELLOM 0 OG 8";
40 INPUT C
50 CLS C
60 PRINT @ 288,"DETTE ER BAKGRUNNSFARGE:";C;
70 GOTO 20
```

Et programmeringseksempel

Oppgave: Bruk datamaskinen til å simulere kast med 2 terninger

Seksjon 1: Vis overskrifter og instruksjoner.

1. blank skjermen
2. skriv overskrift
3. skriv instruksjoner
4. skriv overskrifter, 1. terning, 2. terning

Seksjon 2: Finn verdien til de to terningene

(Når du kaster en terning kan hvilken som helst av de 6 sidene vende opp).

1. første terning er et tilfeldig tall mellom 1 og 6
2. andre terming er et tilfeldig tall mellom 1 og 6

Seksjon 3: Skriv svaret

1.skriv verdien av 1. terning og 2. terning

Seksjon 4: Stopp programmet og gjenta det om ønskelig

1. stopp programmet
2. nytt kast?
3. gjenta seksjonene 1, 2 og 3 om ønskelig

Oversetter vi til et Basic-program (med kommentarer), får vi:

```
10 'TERNINGKASTSIMULERING
20 '
30 'SEKSJON 1
40 CLS : REM BLANKER SKJERMEN
50 PRINT"TERNINGKASTSIMULERING"; 'TITTEL
60 PRINT
70 PRINT"TRYKK BREAK FOR AA AVSLUTTE": 'INSTRUKS
80 PRINT
90 PRINT"1.TERNING","2.TERNING": 'OVERSKRIFTER
100 PRINT
110 REM SLUTT SEKSJON 1
120 REM
130 REM SEKSJON 2
140 T1 = RND (6): 'KAST TERNING 1
150 T2 = RND (6): 'KAST TERNING 2
160 'SLUTT SEKSJON 2
170 '
180 'SEKSJON 3
190 PRINT T1,T2: 'PRESENTERER RESULTATET
200 'SLUTT SEKSJON 3
210 '
220 'SEKSJON 4
230 INPUT"TRYKK ENTER FOR AA KASTE"; A$:
240 GOTO 40: 'RETUR TIL START
250 'SLUTT SEKSJON 4
```

Dette programmet trenger i virkeligheten ikke alle kommentarene. Skriv din egen versjon av programmet.

Det finnes ingen "riktig" versjon av et program. Den riktige versjonen er en som fungerer. Det kan finnes mer elegante eller effektive måter å løse den samme oppgaven på. Vanligvis vil et kortere program være raskere og bruke mindre lagerplass.

REM

REM-setningen brukes for å legge inn kommentarer i et program. Datamaskinen overser alt som følger etter REM på den linjen. REM har også en alternativ kort form: ' (apostrof).

```
10 REM DETTE ER EN KOMMENTARLINJE
35 O = 2 * PI * R : 'FINN OMKRETSEN
```

GOTO

GOTO-setningen har følgende format:

GOTO *linjenummer*

Linjenummeret må være et tall (ikke en variabel), og må eksistere et eller annet sted i programmet. Finnes ikke linjenummeret, vil programmet stoppe med feilmeldingen ?UL ERROR (udefinert linje).

GOTO-setningen blir utført umiddelbart og det har derfor ingen hensikt å plassere en annen setning på samme linje etter GOTO-setningen. Programmet vil aldri nå en slik setning.

```
20 CLS
30 GOTO 60
40 PRINT"I LINJE 40"
50 GOTO 80
60 PRINT"I LINJE 60"
70 GOTO 40
80 PRINT"SLUTT (LINJE 80)"
```

RND

Funksjonen RND genererer slumptall, tilfeldige tall.

RND er en *funksjon*. En funksjon i Basic er noe som på grunnlag av et eller flere tall gjør en beregning som resulterer i en enkelt verdi. Tallene som brukes av funksjonen kalles *argumenter* og finnes alltid i parenteser etter funksjonsnavnet. Resultatet av en funksjon sier vi blir *returnert* til programmet.

RND-funksjonen returnerer et slumptall som er avhengig av verdien til funksjonsargumentet.

Er verdien til argumentet 0 (RND (0)), vil funksjonen returnere en verdi mellom 0 og 1.

Er verdien til argumentet større enn 0 (RND (6)), vil funksjonen returnere en verdi som er et heltall mellom 1 og verdien til argumentet. RDN (6) vil returnere enten 1,2,3,4,5 eller 6. Hvilken av verdiene vet du ikke fordi det er tilfeldig!

```
10 CLS
20 PRINT @ 8,"TAST INN ET TALL";: INPUT N
30 CLS : PRINT @ 194,"3 SLUMPTALL FOR N = ";N
40 PRINT @ 270,RND (N)
50 PRINT @ 302,RND (N)
60 PRINT @ 334,RND (N)
70 GOTO 20
```

Vi avslutter dette kapittelet med en mer avansert versjon av det samme programmet. Her er seksjonene satt sammen i en annen rekkefølge, men resultatet blir det samme.

```
10 'PROGRAM FOR SIMULERING AV
20 ' TERNINGKAST
30 'SEKSJON 1
40 CLS 0: PRINT@ 4,"TERNINGKASTSIMULERING";
50 PRINT@ 166,"1.";: PRINT@ 177,"2.";
60 PRINT@ 198,"TERNING";: PRINT@ 209,"TERNING";
70 PRINT@ 450,"TRYKK BREAK FOR AA AVSLUTTE!";
80 'SEKSJON 4
90 PRINT@ 355,"TRYKK ENTER FOR AA KASTE";
100 INPUT A$
110 'SEKSJON 2
120 T1 = RND (6): T2 = RND (6)
130 'SEKSJON 3
140 PRINT@ 264,T1;: PRINT@ 275,T2;
150 GOTO 90
```

4. Hvordan holde god orden

Tilkopling av kassetspilleren

Noen av programmene har begynt å bli ganske lange, og det er mye arbeid å måtte taste inn et program på ny hver gang du ønsker å kjøre det. Du kan imidlertid på en enkel måte lagre programmene dine på kassettbånd og hente dem tilbake til datamaskinens lager når du trenger dem. For å kunne gjøre dette trenger du en kassettspiller og en tilkoplingskabel.

Enhver kassettspiller av rimelig kvalitet kan brukes, forutsatt at den har mulighet for å:

1. gjøre opptak fra en utvendig kilde (en kontakt, vanligvis merket MIC, AUX eller LINE IN),
2. gi utsignal til en høyttaler eller høretelefon, (en kontakt merket EAR, MONIT, L/S eller SPKR)
3. stoppe og starte ved fjernkontroll (en liten kontakt som vanligvis er merket REM og finnes ved siden av mikrofonkontakten),
4. tilknyttes lysnettet. Dette er ikke helt nødvendig, men svake batterier kan forårsake at du får problemer med å lagre og finne igjen program.

For å knytte båndspilleren til datamaskinen, tar du og plugger DIN-pluggen (det er den runde med 5 pinner) inn i kontakten merket TAPE på venstre side av datamaskinen.

De tre pluggene i den andre enden av kabelen, knyttes til kassettspilleren på følgende måte:

1. Den minste pluggen passer inn i kontakten merket REM (den fjernstyrte på/av bryteren).
2. Den store pluggen med grå eller rød ledning passer inn i kontakten merket MIC, AUX eller LINE IN. Bruk alltid AUX hvis du kan velge mellom AUX eller MIC.
3. Den andre store pluggen med svart eller hvit ledning går inn i kontakten merket EAR (øretelefon), MONIT,L/S eller SPKR.

Slå på kassettspilleren, sett inn et bånd og spol tilbake til begynnelsen av båndet. Sett volumkontrollen på 6 (eller litt over halvveis mellom ON og FULL). Du er nå klar til å lagre dine program.

Hvis du skal spole båndet frem eller tilbake i kassetspilleren må du først ta REM-pluggen ut. Husk å plugge den i igjen når du er ferdig med spolingen. Du kan også la REM-pluggen sitte på plass, men da må du taste MOTOR ON

og trykke ENTER-tasten før du spoler frem og tilbake. Etterpå må du taste MOTOR OFF og trykke ENTER-tasten, slik at kassettspilleren igjen brukes i forbindelse med datamaskinen. Hvis du vil bruke et kassettbånd på ny bør du slette det gamle innholdet først.

Lagring av program på kassett

Tast inn et program, kjør det og forviss deg om at det fungerer korrekt. Gjør så følgende:

1. forviss deg om at du har en ledig kassett i spilleren, og at den står ca. 10 sekund (eller mer) inne på kassetten,
2. trykk Play- og Record-knappene samtidig til de låses nede,
3. tast kommandoen:

CSAVE "PROGRAM1" (trykk ENTER-tasten)

Navnet "PROGRAM1" kan erstattes med hvilket som helst navn du måtte ønske. Det må begynne med en bokstav og ikke være lenger enn 8 tegn. Når du trykker ENTER-tasten, vil kassettmotoren starte og programmet bli registrert. Etter en liten stund vil OK vise seg på skjermen, og kassettspillermotoren vil stoppe.

Programmet vil fortsatt finnes i datamaskinens lager, det er bare en kopi som har blitt lagret på kassettbåndet. Du har nå tatt vare på et program som kalles "PROGRAM1" (eller hva du nå har kalt det) på kassettbåndet.

Du kan undersøke om programmet er tilfredsstillende lagret for senere innlesing. Spol tilbake til programstart, tast SKIPF og trykk ENTER-tasten. Hvis programmet ikke er lagret vil bokstaven S bli stående på skjermen (husk å trykke PLAY-knappen på kassettspilleren). Hvis det har oppstått en feil ved lagringen vil feilmeldingen I/O ERROR bli skrevet på skjermen. Etter SKIPF vil det gamle programmet fortsatt finnes i datamaskinen. Derfor kan det lagre på ny om det ikke gikk bra første gangen.

Les program inn i lageret

For å hente et lagret program inn til datamaskinen, taster du først NEW for å fjerne eventuelle programmer fra lageret. Deretter gjør du følgende:

1. spol magnetbåndet tilbake til begynnelsen,
2. trykk PLAY-tasten til den låser seg,
3. tast kommandoen:

CLOAD "PROGRAM1" (trykk ENTER-tasten)

Kassettmotoren vil starte og bokstaven S vil komme til syne i øvre venstre

CSAVE
CLOAD
SKIPF

Kommandoen CSAVE lagrer et program på kassettbånd. Programnavnet må være på åtte tegn eller mindre.

CSAVE"PROGRAM"

For å lagre data på kassettbånd brukes tilleggsparameteren A. Data vil da bli lagret i ASCII-format. Data kan da senere leses ved en INPUT #–1 kommando.

CSAVE"DATA",A

CLOAD-kommandoen henter en angitt programfil fra kassettbånd inn til datamaskinen.

CLOAD"PROGRAM"

SKIPF-kommandoen hopper til neste programfil etter programmet som er angitt, eller til slutten av det spesifiserte programmet.

SKIPF"PROGRAM"

hjørne av den ellers blanke skjermen. Den viser at datamaskinen søker etter programmet. Når programmet er funnet forandrer S seg til

F PROGRAM1

Når OK viser seg på skjermen og kassettspillermotoren stopper, er programmet lest inn i datamaskinens lager. For å kontrollere dette, kan du taste LIST.

Dersom programmet ikke finnes der eller I/O ERROR viser seg på skjermen, kan det hende at din kassettspiller krever en annen innstilling. Kontroller først at tilkoplingene er korrekte og gjenta så lagrings- og lesekvensene, men med en annen (høyere) innstilling av volumkontrollen til det hele virker.

Hvis du ikke får det til kan du høre på kassetten, etter at du har tatt ut pluggene til spilleren og skrudd ned volumkontrollen en del. Nå skal du først høre to pipetoner, og deretter tett "skurring". Dette er det lagrede programmet. Hvis du ikke hører noe, har du ikke fått lagret programmet. Prøv igjen etter å ha kontrollert at alle pluggene sitter i der de skal.

Lagre flere enn et program

For å lagre fler enn et program på kassetten, må du unngå å lagre programmene oppå de som allerede er lagret. Det betyr at kassettbåndet må posisjoneres forbi det siste programmet. Det gjøres på følgende måte:

1. spol først kassetten tilbake til begynnelsen,
2. trykk Play-knappen til den låser seg,
3. tast kommandoen:

SKIPF "PROGRAM1"

Motoren vil slå seg på og datamaskinen vil søke (S) etter programmet, finne (F) det, lese forbi det og så stoppe motoren og gi OK meldingen.

4. spol fram et par sekund for å få mellomrom mellom programmene,
5. trykk STOPP-knappen,
6. trykk PLAY og RECORD-knappene samtidig. Navngi det nye programmet og CSAVE det.

Etter å ha lagret dette programmet, befinner båndet seg forbi slutten av det, slik at du kan taste inn og lagre andre program hvis du ønsker det.

Tips for pålitelig registrering

1. Bruk båndspilleren tilknyttet lysnettet for å sikre konstant båndhastighet.
2. Bruk nye kvalitetskassetter. Det er best å bruke korte bånd (C30 eller C12). Lengre bånd er ikke hensiktsmessige.
3. Start alltid søkingen ved begynnelsen av båndet.
4. La ikke Play- eller Record-knappene være ned unødige. Trykk Stop etter at du er ferdig med å lagre eller lese.
5. Spol kassettene tilbake før du setter dem bort.
6. Sett merkelapp på kassettene umiddelbart etter at du har lagret noe. Sørg for slettingsbeskyttelse på baksiden av kassetten på kassetter som inneholder viktige program.
7. Oppbevar *ikke* kassetter på eller nær fjernsyn eller en høyttaler. Magnetfelt der i fra kan ødelegge programmene.

Selv om du er svært forsiktig, kan det hende uhell. Noen program kan representere en betydelig investering av både tid og krefter, så ta en ekstra kopi på en annen kassett.

Redigeringsprogrammet (Editor)

Etter som programlinjene blir lengre er det større muligheter for inntastingsfeil. Hittil har den eneste måten vært å skrive om hele linjen. Dette er ikke lenger nødvendig, idet vi skal introdusere editoren (redigeringsprogrammet). Editoren tillater deg å bevege markøren bakover og forover på linjen, forandre, fjerne eller sette inn tegn. For å kalle editoren taster du

EDIT *linjenummer* (ENTER-tasten)

hvor *linjenummer* er nummeret på den programlinjen du ønsker å arbeide på. Linjen vil vises på skjermen. Så vil linjenummeret bli skrevet med blinkende markør rett ved siden.

Markørbevegelse på linjen

Markøren er nå ved begynnelsen av linjen. Trykk på mellomslagstasten for å bevege markøren forover langs linjen. Markøren vil bevege seg og vise de tegnene den har passert. For å bevege markøren tilbake, kan vi bruke tilbaketasten (rettetasten, ←). Du kan få opp hastigheten i markørbevegelsen ved å taste et tall etterfulgt av tasten (dvs. 5 SPACE vil bevege markøren 5 posisjoner forover og 3← vil bevege den 3 bakover). Ved å bruke disse to tastene kan du bevege markøren til hvilket som helst tegn på linjen. Du vil imidlertid ikke se tegnet fordi markøren blinker over det. To andre kommandoer tillater deg å hoppe direkte til et spesielt sted på linjen. For å legge noe til slutten av en linje, tast X (extend). Markøren vil hoppe til slutten av linjen og du kan taste inn tegnene du ønsker å legge til. Ved å bruke søkemuligheten, kan du bevege deg direkte til det tegnet du søker. Tast S etterfulgt av tegnet du ønsker å bevege deg til. (SA vil bevege markøren til den første bokstaven A på linjen). Dersom det er flere enn en A på linjen og du ønsker å bevege deg til den tredje, kan du taste 3SA. Det betyr, søk den tredje forekomsten av A.

Endringer på linjen

Når markøren er kommet i posisjon ved hjelp av mellomslagstasten, tilbaketasten eller S-tasten, kan du:

1. *Fjerne* et tegn ved å taste D (delete). Dette vil fjerne tegnet som markøren står på. For å fjerne flere enn et tegn, tast et tall før D. (5D vil fjerne de neste fem tegnene fra markørposisjonen.)
2. *Endre* et tegn ved å taste C (change), etterfulgt av det nye tegnet. (CF vil endre tegnet under markøren til F. På tilsvarende måte som i punkt 1. vil 3C

endre de neste tre tegnene til de tre som tastes inn.)

3. *Innsette* tegn ved å taste I, etterfulgt av de tegnene du ønsker innskutt. Straks du taster I, vil editor gå over til innsettings-modus. I denne modus blir alt som tastes innskutt på linjen. For å forlate innsettingsmodus og gå tilbake til normal editormodus, må du trykke SHIFT samtidig med ↑.
4. *Liste* den aktuelle tilstanden til linjen ved å taste L.Linjen vil bli vist med de endringer som er gjort så langt. Etter at linjen er vist, vil markøren stille seg inn på begynnelsen.
5. *Forlat* editor ved å trykke ENTER-tasten. Dette vil plassere den endrede linjen tilbake i programmet i lageret og så kvittere med OK. Du kan forlate editor når som helst og i hvilken som helst modus ved å trykke på ENTER-tasten.
6. *Annulere* alle forandringer som er foretatt på linjen ved å taste A. Den opprinnelige linjen blir listet ut. Linjen kan redigeres på ny om nødvendig.

Her er et eksempel på redigering av en engelsk setning, så får vi med oss litt engelsk rettskrivning også. Til å begynne med kan det se litt komplisert ut, men du vil bli forbauset over hvor raskt du blir fortrolig med funksjonene. Tastene du trykker er angitt i hakeparanteser.

[E] [D] [I] [T] [1] [0] [ENTER]
10 PRINT "THEIR ARE MANY MISTOOK IN LINE
10 ■ ■betegner markørposisjon
[1] [0] [SPACE] flytter ti forover, kan også angis som:
[2] [S] [I] flytter markøren til andre I
[C] [R] [C] [E] endrer I til R og R til E
[S] [M] flytter markøren til starten på MANY
[2] [D] fjerner M og A
[SPACE] [C] [O] flytter markøren forbi N og endrer Y til O
[S] [O] flytter markøren til første O i MISTOOK
[D] [C] [A] fjerner første O og endrer andre O til A
[SPACE] [I] [E] [S] flytter markøren en posisjon og innsetter ES
[SHIFT] [↑] forlater innsettingsmodus
[S] [L] flytter markøren til L i LINE
[I] [T] [H] [I] [S] [SPACE] innsetter THIS og en blank før LINE
[SHIFT] [↑] forlater innsettingsmodus
[X] flytter markøren til slutten av linjen og går over i innsettingsmodus
[SPACE] [N] [O] [W] ["] innsetter en blank, NOW og apostrof som avslutter strengen
[SHIFT] [↑] forlater innsettningsmodus
[L] lister hvordan linjen ser ut i øyeblikket
10 PRINT"THERE ARE NO MISTAKES IN THIS LINE NOW"
10 ■
[ENTER] lagrer linjen og forlater redigeringsprogrammet

EDIT

EDIT-kommandoen brukes for å endre innholdet i en angitt linje.

EDIT *linjenummer*

Innenfor EDIT-modus kan alle de følgende kommandoene brukes:

	L	Lister hvordan linjen ser ut i øyeblikket.
	C *tegn*	Endrer (change) det aktuelle tegnet.
n	C *tegn*	Endrer de neste *n* tegn til nye tegn.
	I	Innsetter tegn.
	D	Fjerner (delete) aktuelt tegn.
n	D	Fjerner neste *n* tegn.
	H	Fjerner resten av linjen fra aktuell posisjon og går over i innsettingsmodus (hack).
	X	Utvider linjen. Flytter markøren til slutten av linjen og går over i innsettingsmodus (extend).
	S *tegn*	Søker etter første forekomst av tegnet og flytter markøren dit.
n	S *tegn*	Søker etter n'te forekomst av tegnet og flytter markøren dit.
	K	Fjerner resten av linjen fra aktuell posisjon (kill).
n	K *tegn*	Fjerner linjen til n'te forekomst av tegn.
n	[BLANK]	Flytter markøren *n* posisjoner forover. Dersom n er utelatt antas 1.
n	[←]	Flytter markøren *n* posisjoner tilbake. Dersom n er utelatt blir 1 antatt.
	[SHIFT] [↑]	Forlater innsettingsmodus og vender tilbake til EDIT-modus. (Hold SHIFT-tasten nede og trykk ↑).
	[ENTER]	Forlater redigeringsprogrammet (editor), lagrer linjen og returnerer kontrollen til tastaturet.

Det finnes også to andre redigeringskommandoer som bør brukes med spesiell forsiktighet:

1. Tegnet K (kill) fjerner resten av linjen fra markørens posisjon. K etterfulgt av tegn, vil fjerne alt fra markørens posisjon til første forekomst av tegnet. (3KA vil fjerne til og med tredje forekomst av A.)
2. Tegnet H (hack) fjerner resten av linjen fra markørposisjonen og går så i innsettingsmodus. Kommandoen er nyttig for omskriving av slutten av en linje.

Redigeringsprogrammet (editor) vil gjøre det enklere å rette og forandre

programmet. Tast inn et av eksemplene som du har sett tidligere, om du vil. Skriv det så om ved hjelp av redigeringsprogrammet.

Flere systemkommandoer

Kommandoer som LIST og RUN er systemkommandoer. De er ikke del av programmet, men en direkte kommando til datamaskinen om å gjøre noe nå. Her er flere kommandoer som gjør programmeringen enklere.

DEL - fjerne programlinjer

For å fjerne en linje fra et program har vi tidligere tastet linjenummeret etterfulgt av ENTER. Dette er en bra fremgangsmåte for en eller to linjer, men hva om vi skal fjerne 30-40 linjer

DEL *linjenummer-linjenummer*

vil fjerne en hel blokk med linjer som starter med det første linjenummeret og går til og med det andre linjenummeret.

DEL 100-250

vil fjerne alle linjene fra 100 til og med 250, fra programmet som i øyeblikket finnes i datamaskinens lager.

DEL - kommandoen kan brukes i andre former også:

DEL 20 vil fjerne kun linje 20
DEL 30- vil fjerne alle linjer fra linje 30 til slutten av programmet
DEL -200 vil fjerne alle linjer fra starten av programmet til og med linje 200
DEL - vil fjerne hele programmet

RENUM - renummerering av programlinjer

RENUM-kommandoen vil renummerere alle eller noen av linjenummerne i ditt program. Den vil også endre linjenummeret i GOTO, GOSUB, IF THEN, ON GOTO og ON GOSUB-setninger, og sørge for at disse fortsatt forgrener seg til samme sted. Vi skal møte disse setningene senere.

RENUM *nylinje, startlinje, tilvekst*

Nylinje er det nye linjenummeret til den første linjen som skal renummereres. *Startlinje* er hvor du ønsker renummereringen fra, og *tilvekst* er den nummeravstanden du ønsker skal brukes mellom hver renummerert linje. Alle eller noen av disse parameterne kan utelates. Utelater du *nylinje*, blir 10 antatt. Utelater du *startlinje*, forårsaker det at hele programmet blir

DEL

DEL-kommandoen brukes for å fjerne angitte linjer fra det programmet som i øyeblikket befinner seg i lageret.

DEL *linjenummer 1 – linjenummer 2*

Kommandoen vil fjerne alle programlinjer fra og med *linjenummer 1* opp til og med *linjenummer 2*. Begge linjenummerne kan utelates og kommandoen kan brukes på hvilken som helst av de følgende måtene:

DEL –	Fjerner hele programmet (som NEW).
DEL –100	Fjerner fra starten til og med linje 100.
DEL 300–	Fjerner fra og med linje 300 til slutten.
DEL 40	Fjerner kun linje 40.
DEL 100–200	Fjerner alle linjer f.o.m. 100 t.o.m. 200.

RENUM

RENUM-kommandoen gir mulighet for å endre enten alle eller noen av programmets linjenummere. RENUM-kommandoen forandrer altså også linjenummerne i forgreningssetninger (GOTO etc.) for å sikre at programflyten fortsetter på riktig måte.

RENUM *nylinje, startlinje, trinnverdi*

Kommandoen vil omnummerere alle linjene fra *startlinje* i det den begynner med *nylinje* og nummererer i trinn som angitt av trinnverdien. Alle parameterne er valgfrie og setningen kan brukes på hvilken som helst av følgende måter:

RENUM	omnummererer hele programmet. Linjene vil bli nummerert 10,20,30,...
RENUM 100	omnummererer hele programmet. Linjene vil bli nummerert 100,110,120,...
RENUM 100,50,5	omnummererer programmet fra linjenummer 50. Linjene vil bli nummerert 100,105,110,...
RENUM,,20	omnummererer hele programmet. Linjene vil bli nummerert 10,30,50,...

Legg merke til at dersom en parameter er utelatt og den etterfølgende parameteren skal brukes, må komma være angitt. RENUM kan ikke brukes til å endre rekkefølgen på linjene.

renummerert. Utelater du *tilvekst*, vil trinnet mellom linjenummerne bli på 10.

RENUM vil nummerere hele programmet som 10, 20, 30, ...
RENUM 100,50,5 vil nummerere linjenummerne fra 50 som 100, 105, 110, ... (alle linjer før 50 vil bli uforandret)
RENUM 110,,2 vil renummerere hele programmetsom110, 112, 114, ...
RENUM ,,5 vil renummerere hele programmet som10, 15, 20, ...

Legg merke til at dersom du utelater en parameter, men ønsker å bruke en som kommer senere i rekkefølgen, må du ha med komma.

TR - følge programflyten

Noen ganger når du har problemer med et program, er det nyttig å kunne følge flyten i programmet. Sporingshjelpemiddelet (trace) gir deg denne muligheten. Ved å taste TRON før du kjører programmet, setter du sporingen på. Linjenummerne vil da bli skrevet på skjermen etter hvert som programmet kommer til dem. Dette setter deg i stand til å se om programmet hopper til rett sted. For å slå av sporingsmekanismen taster du TROFF. TRON og TROFF kan også benyttes i programmet, med et linjenummer foran. På den måten kan en følge utførelsen av en mindre del av programmet.

STOP - og start

Et program kan stanses under utførelsen ved å legge inn en programlinje som inneholder STOP-kommandoen.

185 STOP

Denne linjen vil forårsake at programmet stopper når det når linje 185. En melding skrives på skjermen, som forteller at programmet har stoppet og ved hvilken linje det har stoppet. Du kan nå se på innholdet av en hvilken som helst variabel ved å bruke PRINT eller ? fulgt av variabelnavn. For å starte opp programmet igjen, taster du CONT (continue = fortsett), og programmet vil fortsette fra linjen etter STOP.

Nå kan du renummerere program, fjerne uønskede linjer, endre innholdet i en hvilken som helst linje og lagre programmet på en kassett. Etter som du nå er i stand til å vedlikeholde programmene, vil vi nå begynne å konstruere program som gjør noe mer interessant enn å lyse opp skjermen og lage merkelige lyder.

TR

Programflyten kan følges ved bruk av sporing (trace). Etterhvert som programlinjene nås, vil linjenummerne bli skrevet på skjermen. Sporingen må slås på før programmet kjøres.

TRON slår på sporingen
TROFF slår av sporingen

Begge er direkte kommandoer og skal ikke ha linjenummer.

Hvis en vil benytte sporing for å undersøke om når en liten del av programmet blir utført, kan en gjøre dette ved å bruke TRON og TROFF inne i programmet.

END
STOP
CONT

END-setningen avslutter programutførelsen og sender kontrollen tilbake til tastaturet.

STOP-setningen avbryter utførelsen av et program på linjen som inneholder STOP. En melding BREAK AT n viser seg på skjermen for å angi at programmet har stoppet ved linjenummer n. For å starte programmet igjen taster du kommandoen CONT uten linjenummer. Programmet vil fortsette utførelsen på linjen etter STOP. Et program vil ikke kunne fortsette etter en END-setning, da må det kjøres på ny.

5. Programkontroll

I slutten av kapittel 3 så vi hvordan vi kunne konstruere et program ved å betrakte det som et antall separate deler. Nå skal vi se hvordan vi kan styre programmet for å knytte disse delene sammen. Dette kalles forgrening. Vi har allerede møtt forgreningssetningen GOTO. Dette er en forgrening uten vilkår, fordi straks programmet når GOTO-setningen, hopper det umiddelbart til det spesifiserte linjenummeret og fortsetter der. Programmet har ikke noe valg når det gjelder dette. GOTO betyr gå til setningsnummeret straks. Så langt har vi hovedsakelig brukt GOTO for å returnere til begynnelsen av programmet. Evnen til å gjenta en del av et program gang etter gang er svært nyttig, men det er lite sannsynlig at vi vil ønske å gjøre dette bestandig. Det er heller ingen god regel å være avhengig av BREAK-tasten for å stoppe programmet. Heldigvis inneholder Basic-språket en rekke setninger som gir mulighet til å ha kontroll med flyten i programmet.

Valg av forgrening

Den første av disse setningene er en utvidelse av GOTO-setningen. Det er ON...GOTO-setningen og den har følgende form:

ON *numerisk uttrykk* GOTO *linjenummerliste*

Det *numeriske uttrykket* blir beregnet og om nødvendig trunkert til et helt tall (avrundet ved at tallene etter desimalpunktumet fjernes). Kontrollen blir så overført til et av linjenummerne i listen. Hvis det beregnede uttrykket er 1, går den til det første linjenummeret, hvis det er 2, til det andre osv. Er verdien av uttrykket mindre enn 1 eller større enn antall linjenummer i listen, vil datamaskinen overse setningen fullstendig og fortsette på neste linje. Et negativt tall som resultat av uttrykket vil imidlertid forårsake at programmet stopper med en feilmelding. Det er vanligvis fornuftig å kontrollere at verdien er innenfor det planlagte området før man når ON...GOTO-setningen.

ON...GOTO

ON GOTO-setningen gir mulighet for mange forgreninger til angitte linjenummere.

ON *uttrykk* GOTO *linjenummerliste*

Uttrykket blir beregnet og avkuttet etter komma (trunkert), dersom resultatet ikke er et heltall. Programmet går så til det linjenummeret i listen som har en posisjon som svarer til verdien av uttrykket. Er uttrykkets verdi 4, vil ON GOTO-kommandoen hoppe til det fjerde linjenummeret i *linjenummerlisten*. Er verdien til uttrykket negativt, vil det komme en feilmelding. Er uttrykkets verdi null eller større enn antall linjenummer i listen, vil setningen bli ignorert og programmet vil fortsette på neste linje.

```
10 CLS : PRINT"BEREGNING AV ANNENGRADSLIGNING"
20 INPUT"A,B,C"; A,B,C: IF A = 0 THEN 20
30 R = -B/ (2*A): D = B*B-4*A*C: S = SGN (D)
40 P = SQR (D*S)/ (2*A)
50 ON S + 2 GOTO 80,60,70
60 PRINT"LIKE SVAR": PRINT R,R: END
70 PRINT"REELLE SVAR": PRINT R + P,R - P: END
80 PRINT"KOMPLEKSE SVAR": PRINT R,R: PRINT P,-P: END
```

Her er noen eksempler på setninger:

```
140 ON P GOTO 200,300,400
210 ON X-4 GOTO 20,40,700,10,690
185 ON B*C/D-E GOTO 115,285,900,40
```

ON...GOTO-setningen er en nyttig måte å velge ut fra et antall muligheter. Den kan betraktes som en betinget forgreningssetning. Vi vil gjøre bruk av den på denne måten i et eksempel senere i kapittelet.

Betinget beslutning

En mer nyttig form for betinget forgrening er tilgjengelig ved å bruke IF...THEN-setningen. Dette er sannsynligvis den nyttigste setningen i Basic. Den kan være svært enkel, eller omfattende og sammensatt. I sin enkleste form:

IF *vilkår* THEN *linjenummer*

Setningen har følgende mening: Dersom vilkåret er sant, gå da til linjenummeret, ellers fortsett på neste linje.

```
120 IF D>9 THEN 250
180 IF A$ = "JA" THEN 600
```

I linje 120 ovenfor vil programmet overføre kontrollen til linje 250 hvis, og bare hvis, verdien til D er større enn 9. Er verdien mindre enn eller lik 9, vil programmet fortsette på neste linje. I linje 180 vil forgreningen til linje 600 kun skje dersom strengvariabelen A inneholder tegnene JA. Overenstemmelsen må i dette tilfellet være eksakt. J, JO, JADA, OK, (eller til og med "ja") vil forårsake at programmet ignorerer forgreningen til linjenummer 600. I sin fullstendige form ser IF...THEN-setningen ut som følger:

IF *vilkår* THEN *handling 1* ELSE *handling 2*

som har betydningen: Dersom *vilkåret* er sant så utfør *handling 1*, dersom vilkåret ikke er sant så utfør *handling 2*.

```
210 IF P = 3 THEN PRINT "SANT" ELSE PRINT "USANT"
```

I dette eksempelet vil SANT bli skrevet ut bare dersom P er lik 3. Har P en annen verdi, vil USANT bli skrevet. I begge tilfeller vil programmet fortsette på neste linje. Både *handling 1* og *handling 2* kan bestå av mer enn en programsetning.

```
210 IF P = 3 THEN PRINT "SANT": R = R + 1: GOTO 560
    ELSE PRINT "USANT": L = L + 1
```

Dersom P er lik 3, vil datamaskinen skrive SANT, legge 1 til variabelen R og fortsette programmet på linje 560. For hvilken som helst annen verdi vil den skrive USANT, legge 1 til L og fortsette på neste linje.

Handling 1 og *handling 2* kan være hvilken som helst lovlig Basic-setning, også inkludert andre IF...THEN-setninger. Den eneste begrensningen er at den fullstendige IF...THEN-setningen må få plass på en linje. (En linje kan inneholde maksimalt 256 tegn inkludert linjenummeret.)

Fram til nå har vi bare sett på hva som hender etter at et vilkår har blitt beregnet. Selve valget ligger i å teste vilkåret. Et vilkår kan være sant eller usant, ikke noe annet. (Datamaskinen gir sant verdien 1 og usant verdien 0. Et enkelt vilkår har formen:

Uttrykk 1 relasjon uttrykk 2

Uttrykk 1 og *uttrykk 2* er de vanlige Basic-uttrykkene slik som de forekommer i tilordningssetninger. Begge uttrykkene må være av samme type (begge numeriske eller begge strenger).

IF...THEN...ELSE

Det fullstendige formatet til IF-setningen er:

IF *vilkår* THEN *handling 1* ELSE *handling 2*

Setningen undersøker *vilkåret* som enten er sant eller usant. Dersom sant, blir *handling 1* utført og dersom usant, *handling 2*.

Et *vilkår* består av et uttrykk, en relasjon og et uttrykk.

Uttrykkene kan være hvilke som helst Basic-uttrykk av samme type (altså begge numeriske eller begge strenger). En relasjon er en av de følgende "operatorene":

=	Lik	<>	Ikke lik
>	Større enn	<	Mindre enn
>=	Større enn eller lik	<=	Mindre enn eller lik

Flere vilkår kan kombineres ved hjelp av de logiske operatorene AND, OR, NOT:

Vilkår AND *vilkår* er sant dersom begge *vilkårene* er sanne
Vilkår OR *vilkår* er sant dersom et av *vilkårene* er sant
NOT *vilkår* er sant dersom *vilkår* er usant

Handling 1 og *handling 2* kan være en vilkårlig Basic-setning og også en annen IF-setning. De kan også være flere setninger, adskilt med kolon (:).

ELSE-delen av kommandoen kan utelates. Programmet vil da fortsette på neste linje, dersom vilkåret er usant.

```
10 CLS : PRINT @ 11,"GJETTELEK": N = RND (100): T = 0
20 INPUT"GJETT PAA ET TALL "; G: T = T + 1
30 IF G = N THEN PRINT"RIKTIG ETTER"; T;"GANGER": END
40 IF G>N THEN PRINT"NEI, DET VAR FOR STORT" ELSE PRINT
   "NEI, DET VAR FOR LITE"
50 GOTO 20
```

En *relasjon* kan være en av de følgende typer:

Betydning	*Symbol*	*Eksempel*
Lik	=	60 IF X = Y+2 THEN 100
Mindre enn	<	110 IF A*B+2<C/2 THEN 20
Større enn	>	185 IF A$>B$ THEN PRINT A$
Mindre enn eller lik	<=	220 IF 4*W9<=B/Z9 THEN A = A-1
Større enn eller lik	>=	415 IF B7 >= 0 THEN X = 0
Ikke lik	<>	80 IF A$<>"JA"THEN 999

Legg merke til at relasjonen kan brukes på strenger like godt som på numeriske uttrykk. Når strenger blir sammenlignet, blir hvert tegn undersøkt i rekkefølge. Vi kan altså bruke IF...THEN-setninger for å sammenligne alfabetisk rekkefølge f.eks.: Vilkåret "A"<"B" er sant, fordi bokstaven A kommer før bokstaven B i alfabetet. Vilkåret "AAA"<="AA" er usant, fordi AAA kommer etter AA i f.eks. en ordbok (blank kommer før A).

Vilkår kan utvides ved å kombinere to eller flere vilkår ved bruk av operatorene AND og OR.

```
270 IF A = 4 AND B = 7 THEN 500
```

AND operatoren betyr at begge vilkårene må være sanne samtidig for at hele vilkårsuttrykket skal være sant. I eksempelet betyr det at dersom A er lik 4 og B har en verdi forskjellig fra 7, vil hele uttrykket bli usant og programmet vil fortsette på neste linje.

OR operatoren betyr at hvis ett (eller flere) av vilkårene er sant, vil hele uttrykket være sant.

```
320 IF D<3OR F+G>25 THEN 100
```

Programmet vil gå til linje 100 hvis D er mindre enn 3, uten hensyn til hva verdien av F+G er. Tilsvarende vil det gå til linje 100 dersom F+G er større enn 25, uten hensyn til verdien til D.

Programmet som følger er et forenklet undervisningsprogram. Kommentarene i programmet forteller hva hver enkelt del utfører. Legg merke til bruken av ON...GOTO for å velge opsjon (valgmulighet), og IF...THEN-setninger for å undersøke svarene. I linje 800 introduserer vi en ny kommando, INKEY$. Denne kommandoen avsøker tastaturet for å se om en tast er blitt trykket ned. Tegnet vil i såfall bli lagret i A$ (se linje 800). Kommandoen krever ikke at ENTER tasten trykkes etterpå. Se rammen merket INKEY$.

Programmet ser lengre ut enn det i virkeligheten er, etter som over halvparten av det er kommentarlinjer. For å spare plass, vil det i framtiden ikke være fullt så omfattende kommentarer. Mellomslag er angitt med trekant (∇).

INKEY$

INKEY$ er en *funksjon*. Den undersøker tastaturet for å se om en tast er trykket ned, og er det tilfellet tar den tegnet fra tasten.

Den kan brukes for å mate et enkelt tegn inn i en strengvariabel og krever ikke at ENTER-tasten trykkes ned.

```
10 CLS 0: PRINT @ 5,"TRYKK EN TAST OG JEG VIL";
20 PRINT @ 38,"FORTELLE DEG HVILKEN:▽";
30 B$ = "DU TRYKKER IKKE NOEN TAST▽▽"
40 C$ = "TASTEN DU TRYKKET VAR:▽"
50 A$ = INKEY$
60 IF A$ = "'' THEN PRINT @ 193,B$; ELSE PRINT @ 193,C$;A$;
70 FOR D = 1 TO 600: NEXT D: GOTO 50
```

```
10  REM ARITMETISK TRENINGSPROGRAM
20  REM
30  REM NULLSTILLER TELLERE OG FINNER VANSKELIGHETSGRAD
40  REM
50  R = 0: W = 0: CLS
60  T$ = "ARITMETISK TRENING"
70  PRINT @ 2,T$
80  PRINT @ 64,"▽▽ ANGI VANSKELIGHETSGRAD"
90  PRINT @ 96,"▽▽ ET ANTALL MELLOM 1 OG 10":
    INPUT L : L1=10*L-1
100 REM
110 REM VISER VALGMULIGHETENE
120 REM
130 CLS : PRINT@ 6,T$
140 PRINT @ 71,"1. ▽ ADDISJON"
150 PRINT @ 103,"2. ▽ SUBTRAKSJON"
160 PRINT @ 135,"3. ▽ MULTIPLIKASJON"
170 PRINT @ 167,"4. ▽ DIVISJON"
180 REM
190 REM SETTER SKRIVEPOSISJONER
200 REM
210 P = 224: Q = 352
220 REM
230 REM VELGER REGNEART
240 REM
250 PRINT @ P,"▽▽ HVILKEN VIL DU REGNE"; : INPUT A
```

```
260 REM
270 REM VELGER TO TILFELDIGE TALL FOR OPPGAVEN
280 REM
290 N1 = RND (L1): N2 = RND (L1)
300 REM
310 REM HOPPER TIL REGNEARTEN
320 REM
330 ON A GOTO 390,460,530,600
340 REM
350 REM ULOVLIG VALG - NYTT VALG
360 REM
370 CLS : SOUND 160,3: GOTO 130
380 REM
390 REM ADDISJONSDELEN
400 REM
410 PRINT @ P,"▽▽▽▽▽▽▽▽▽▽▽▽▽ ADDISJON"
420 PRINT @ Q,"▽▽ HVA ER"; N1;"PLUSS ▽"; N2; : INPUT N4
430 N3 = N1 + N2
440 IF N4 = N3 THEN 690 ELSE 730
450 REM
460 REM SUBTRAKSJONSDELEN
470 REM
480 PRINT @ P,"▽▽▽▽▽▽▽▽ SUBTRAKSJON"
490 PRINT @ Q,"HVA ER"; N1;"MINUS"; N2; : INPUT N4
500 N3 = N1 - N2
510 IF N4 = N3 THEN 690 ELSE 730
520 REM
530 REM MULITPLIKASJONSDELEN
540 REM
550 PRINT @ P,"▽▽▽▽▽MULTIPLIKASJON"
560 PRINT @ Q,"HVA ER"; N1; "GANGER"; N2; : INPUT N4
570 N3 = N1*N2
580 IF N4 = N3 THEN 690 ELSE 730
590 REM
600 REM DIVISJONSDELEN
610 REM
620 PRINT @ P,"▽▽▽▽▽▽▽▽ DIVISJON"
630 PRINT @ Q,"HVA ER"; N1;"DIVIDERT MED"; N2; : INPUT N4
640 N3 = N1/N2
650 IF N3 = N4 THEN 690 ELSE 730
660 REM
670 REM RIKTIG SVAR
680 REM
```

```
690 R = R+1: PRINT @ P,"∇∇∇∇∇∇∇∇∇ RIKTIG": GOTO 770
700 REM
710 REM GALT SVAR
720 REM
730 G = G+1: PRINT @ P,"∇∇∇∇∇∇∇∇∇∇∇∇ GALT":
    PRINT @ P + 32, "∇∇∇∇∇SVARET ER ∇"; N3: GOTO 770
740 REM
750 REM KONTROLL FOR GJENTAKELSE OG BLANKING AV LINJER
760 REM
770 FOR D = 1 TO 1600: NEXT D
780 PRINT @ P,"∇": PRINT @ P+32,"∇": PRINT @ Q,"∇ "
790 PRINT @ P,"NY OPPGAVE? (J/N)"
800 A$ = INKEY$: IF A$ = ""THEN 800
810 IF A$ = "J" THEN 130
820 REM
830 REM PRESENTERER RESULTAT OG AVSLUTTER
840 REM
850 CLS : PRINT @ 128,"DU FIKK"; R;"RETTE OG"; G;"GALE"
860 END
```

Gjentakelser

Ofte vil man i et program trenge å gjenta en rekke linjer et antall ganger. Den neste typen av forgreningssetninger legger forholdene tilrette for å gjøre dette. Tast inn følgende lille program og kjør det.

```
10 CLS
20 FOR I = 1 TO 50
30 PRINT @ 198,"TELLER I = "
40 PRINT @ 214,I
50 NEXT I
60 PRINT @ 396,"SLOYFE SLUTT"
```

Programmet er for raskt, slik at vi ikke kan se hva som skjer. Føy derfor til følgende linje:

```
45 FOR J = 1 TO 100: NEXT J
```

Som du kan se, teller programmet fra 1 til 50. Linjene som gjentas er linjene 30, 40 og 45. En slik setningsrekkefølge som gjentas, kalles en *sløyfe*. Programmet går i dette tilfellet fra setning 50 tilbake til setning 20. Setningene som kontrollerer sløyfen er setningsnummer 20 (starten på sløyfen), og setningsnummer 50 (slutten av sløyfen). Linje 20 tolker vi som "for alle verdier av I fra 1 til 50 i trinn på 1, utfør de etterfølgende setningene til NEXT-

setningen blir nådd". Variabelen I fungerer her som teller og har verdiene 1,2,3,...,50. Vi føyet til setningsnummer 45, fordi programmet telte for fort. Denne sløyfen har ingen setning som den utfører, slik at telleren (variabelen J denne gangen) bare teller fra 1 til 100. Denne pausen (tellingen) er tilstrekkelig til å forsinke den andre sløyfen, slik at vi blir i stand til å lese tallene på skjermen. Forandrer vi nå setningsnummer 20 til :

```
20 FOR I = 1 TO 50 STEP 2
```

vil telleren nå telle fra 1 i trinn på to (1,3,5,7,...,49). Tillegget av STEP tillater oss å velge tellerskritt. Utelates ordet STEP, antas det at det skal telles i trinn på en. Gjør endring i eksempelet ved å taste følgende linjer:

```
15 INPUT"START,SLUTT,TRINN"; A,B,C
20 FOR I = A TO B STEP C
70 PRINT @ 428,"OG I ="; I
80 GOTO 10
```

Kjør programmet med forskjellige verdier for start, slutt og trinn. Forsøk noen av de følgende verdiene: (tast komma mellom tallene)

START	SLUTT	TRINN
50	1	1
-50	50	5
1	10	0.5
2.6	3.9	0.1
1	-2	1

Du vil legge merke til at du kan telle fremover eller bakover i hva slags trinn du vil. Den eneste regelen er at om trinnene er positive, må startverdien være mindre enn sluttverdien. Tilsvarende dersom trinnene er negative, må startverdien være større enn sluttverdien. Av det siste eksempelet ovenfor (1,-2,1), vil du se at selv om du bryter denne regelen, (det er ikke mulig å telle fra 1 til -2 i trinn på +1), vil sløyfen fortsatt bli utført en gang.

Sløyfer kan nestes inn, omsluttes fullstendig, i andre sløyfer. (Sløyfen på linje 45 er nestet inn i en annen sløyfe som starter på linje 20). Du må forsikre deg om at sløyfene blir avsluttet i korrekt rekkefølge.

```
20 FOR I = 1 TO 10
30 FOR J = -2 TO 4 STEP 0.6
40 FOR K = 1 TO 0 STEP -0.1
.
.
100 NEXT K
110 NEXT J
120 NEXT I
```

FOR...NEXT

FOR...NEXT er to setninger:

FOR *numerisk variabel* = *uttrykk* TO *uttrykk* STEP *uttrykk*

og

NEXT *numerisk variabel*

FOR...NEXT-setningene arbeider sammen og kontrollerer antall ganger at en del av et program blir utført. Dette kalles en *sløyfe*.

Uttrykkene blir beregnet og sløyfen teller fra verdien av det første uttrykket til verdien av det andre uttrykket, med en trinnverdi (step) gitt av det tredje uttrykket. Den numeriske varianten inneholder den aktuelle verdien til telleren. Sløyfen vil alltid bli utført minst en gang selv om verdiene ikke tilsier det. Dersom STEP-delen av setningen er utelatt, blir +1 brukt.

Sløyfer kan være flere inne i hverandre (nestet), men må avsluttes i riktig rekkefølge.

Du kan hoppe ut av en FOR...NEXT-konstruksjon ved hjelp av GOTO,IF...THEN eller lignende setninger. Du kan imidlertid aldri hoppe inn i midten av en sløyfe.

```
10 CLS : CLEAR: DIM L (1000)
20 INPUT "TAST ET TALL"; N: IF N>1000 OR N<2 THEN 20
30 CLS : PRINT"PRIMTALL MELLOM 2 OG"; N
40 FOR I = 2 TO N: IF L (I) < 0 THEN 80
50 PRINT I;
60 IF I>SQR (N) THEN 80
70 FOR J = I TO N STEP I: L (J) = -1 : NEXT J
80 NEXT I
```

Alle FOR-setninger må ha sin tilhørende NEXT-setning. Variabelen etter NEXT viser hvilken FOR-setning den tilhører. Sløyfene må lukkes i motsatt rekkefølge av den de ble åpnet i. Dersom de tre siste linjene ovenfor var:

```
100 NEXT J
110 NEXT K
120 NEXT I
```

ville programmet stoppe og gi en feilmelding, fordi sløyfe J og sløyfe K overlapper hverandre.Slutter alle sløyfene på samme sted, kan NEXT-setningene slås sammen til

```
100 NEXT K,J,I
```

men rekkefølgen av variablene må fortsatt være korrekt som tidligere.

Variablene som brukes for å etablere sløyfene (i eksempelet I,A,B og C), kan brukes inne i sløyfen. Du har imidlertid ikke lov til å endre start, slutt eller trinnstørrelsen. Selve telleren kan endres ved at den opptrer på venstre side av et tilordningsuttrykk, men det er ikke å anbefale.

Andre forgreningssetninger som GOTO eller IF...THEN, kan brukes inne i en sløyfe for å forlate den før den er ferdig. Du kan imidlertid aldri hoppe inn i midten av en sløyfe, men må alltid starte med FOR-setningen.

Følgende program viser bruken av nestede sløyfer. Det simulerer en digital klokke. For å gjøre den virkelig nøyaktig, kan det være at du må justere forsinkelsessløyfen i linje 220. Legg merke til bruken av INKEY$ for å stoppe og starte klokken på linjene 90, 150 og 170.

```
10 CLS 0: PRINT @ 10,"DIGITAL KLOKKE";
20 'SKRIVEPOSISJONER
30 P = 261: Q = 196
40 'INSTRUKSJONSTEKST
50 PRINT @ 386,"TRYKK BLANKTAST FOR STOPP OG";
60 PRINT @ 421,"START OG 'R' FOR OMSTART";
70 'SKRIVER TEKST OG VENTER PAA START
80 PRINT @ Q,"TIMER"; : PRINT @ Q + 10,"MIN "; : PRINT @ Q +
   22,"SEK";
90 A$ = INKEY$: IF A$ <>"∇"THEN90
100 'START SLOYFER FOR TIMER, MIN OG SEK
110 FOR H = 0 TO 23: FOR M = 0 TO 59: FOR S = 0 TO 59
120 SOUND 220,1
130 'SLOYFE FOR TIDELS SEK
140 FOR T = 0 TO 9
150 A$ = INKEYS$: IF A$ <>"∇"THEN200
160 'KONTROLL FOR TASTETRYKK
170 A$ = INKEY$: IF A$ = "R"THEN 110
180 IF A$ <>"∇"THEN170
190 'SKRIVER TIDEN
200 PRINT @ P,H; : PRINT @ P + 10,M;: PRINT @ P + 19,S;".";T;
210 'KLOKKENS JUSTERINGSSLOYFE
220 FOR D = 1 TO 13: NEXT D
230 NEXT T
240 'SLUTT TIDELS SEK SLOYFEN
250 NEXT S,M,H
260 'SLUTT SLOYFER FOR SEK, MIN OG TIMER
270 PRINT @ 448,"STOPPET"
280 END
```

Program i program

Et program har en hovedstruktur som normalt består av en sammenstilling av en rekke mindre delprogram. Noen av disse delene kan det være behov for mange ganger og forskjellige steder i hovedprogrammet. Programstrukturen kan ofte forenkles ved å behandle slike situasjoner som delprogram (subrutiner). Et delprogram er som navnet antyder en del av et større program eller om du vil et program i et program. Det viktigste ved et delprogram er at når det kalles, så utføres programsetningene i delprogrammet, og det returnerer så tilbake til det sted det ble kalt fra. For å kalle et delprogram, bruker du setningen:

GOSUB *linjenummer*

hvor *linjenummer* er det linjenummeret i programmet hvor delprogrammet starter. Etter som et delprogram alltid må returnere, må ethvert delprogram slutte med en spesiell setning

RETURN

GOSUB-setningen oppfører seg på en liknende måte som GOTO-setningen. Forskjellen ligger i at med GOTO, forgrenes programmet og fortsetter der. Det kommer ikke tilbake uten at det møter på en annen GOTO-setning som sender det tilbake. Tast inn følgende eksempel og kjør det:

```
10 CLS : PRINT "I HOVEDPROGRAMMET"
20 GOSUB 50
30 PRINT "TILBAKE I HOVEDPROGRAMMET"
40 END
50 PRINT "I 1. DELPROGRAM"
60 GOSUB 90
70 PRINT "TILBAKE I 1. DELPROGRAM"
80 RETURN
90 PRINT"I 2. DELPROGRAM"
100 RETURN
```

Når programmet kjøres, utføres programmet i følgende linjenummerrekkefølge:

10,20,50,60,90,100,70,80,30,40

Legg merke til hvordan det første delprogrammet (linjene 50-80), kaller det andre delprogrammet (linjene 90-100). END-setningen på linje 40, betyr nøyaktig hva den sier. Den brukes for å angi hvor programmet avsluttes. Forsøk å ta ut linje 40 og kjør eksempelet på ny. Da bør du få ?RG ERROR IN 80. Dette er fordi programmet støtte på en RETURN-setning uten at det var klar over at det var kommet inn i et delprogram. Programmet falt gjennom

bunnen av hovedprogrammet inn i det første delprogrammet. Du må alltid beskytte delprogram og sørge for at den eneste måten de kan nås på er ved en GOSUB-setning. Og den eneste måten å komme ut av dem på er ved en RETURN-setning. Du kan bruke GOTO, IF...THEN og liknende forgreningssetninger inne i et delprogram, men de må ikke forårsake en forgrening til en linje som ligger utenfor delprogrammet.

Tilsvarende ON...GOTO-setningen, finnes en flerveis forgrening også tilgjengelig for delprogram og den har en liknende form:

ON *uttrykk* GOSUB *linjenummerliste*

De fleste erfarne programmerere har et bibliotek av delprogram, slik at mange nye program kan lages på grunnlag av disse. Det er vanligvis fornuftig å skrive delprogrammene med høye linjenummer, 10000-, slik at de kan plasseres direkte inn i et program, uten at man behøver å nummerere om.

I de senere kapitlene vil vi få mange eksempler på program som inneholder delprogram. Vi vil derfor ikke gi noen flere eksempler nå.

GOSUB
RETURN
ON...GOSUB

GOSUB-setningen overfører programkontrollen til begynnelsen av et delprogram (subprogram). RETURN-setningen overfører kontrollen tilbake til linjen som følger etter GOSUB-setningen.

GOSUB er etterfulgt av et linjenummer som er første linje i delprogrammet.

GOSUB 1600

Et delprogram må inneholde minst en RETURN-setning.

ON...GOSUB-setningen tillater en flerveis forgrening til delprogram på en måte som er lik den til ON...GOTO-setningen.

ON *uttrykk* GOSUB *linjenummerliste*

Er uttrykket negativt, vil programmet stoppe med en feilmelding. Er uttrykket null eller større enn antall linjenummer i listen, blir setningen ignorert og programmet vil fortsette på neste linje.

```
10 CLS : INPUT"TAST TO TALL"; A,B
20 INPUT"TAST ET TALL MELLOM 1 OG 4"; C
30 ON C GOSUB 100,200,300,400
40 IF C >= 1 AND C <= 4 THEN END
50 PRINT C;"ER IKKE MELLOM 1 OG 4": GOTO 20
100 PRINT"ADDISJON"; A;"PLUSS"; B;"ER"; A + B
110 RETURN
200 PRINT"SUBTRAKSJON"; A;"MINUS"; B;"ER"; A - B
210 RETURN
300 PRINT"MULTIPLIKASJON"; A;"GANGER"; B;"ER"; A * B
310 RETURN
400 PRINT"DIVISJON"; A;"DIVIDERT MED"; B;"ER"; A/B
410 RETURN
```

6. Nye dimensjoner

I kapittel 2 diskuterte vi variabeltyper. Vi sa at variablene kom i to "størrelser": enkle og tabell. Hittil har vi kun sett på enkle variabler.

Dersom du ønsker å bruke datamaskinen for å få en oversikt over bøkene eller platene dine, vil det ikke ta lang tid før du ikke lenger har flere variabler å lagre dem i. Det vil også bli svært vanskelige program å skrive, hvor du er nødt til å holde styr på alle de forskjellige navnene. Datamaskinen kommer her til hjelp med *tabellvariable*.

Lister og tabeller

Tabellvariable er spesielt nyttige når man har med lister av ting å gjøre. F.eks. kan sette opp en bokliste på følgende måte:

1. Tittel 1
2. Tittel 2
3. Tittel 3

.
.
.

Nå kan du referere til en av bøkene dine ved å f.eks. spørre etter nummer 8 i listen. I programmet vårt gir du et navn på denne rekkefølgen av variabler (titlene), og refererer til en enkelt verdi i listen ved å angi en *indeks* (nummeret på elementet i listen). Først må du altså gi listen et navn.

Navnet på en datatabell følger de samme regler som enkle variabler. Datamaskinen kjenner igjen forskjellen mellom de to typene fordi datatabellvariablene alltid har paranteser etter seg som inneholder indeksen.

A (5)refererer til det 5. elementet i en numerisk datatabell (liste kalt A).
D7$ (28)refererer til det 28. elementet i en strengtabell som er kalt D7$.

En tabell må defineres ved å fortelle datamaskinen hva tabellen skal hete og størrelsen på den. Dette gjøres ved hjelp av DIM-setningen:

DIM *datatabellnavn* (tall), *datatabellnavn* (tall, tall)

DIM

DIM-setningen brukes for å dimensjonere tabeller. Tabeller kan være en- eller todimensjonale og kan enten inneholde tall eller strenger (tekst). Tabellnavn følger samme regler som de for enkle variable.

DIM *tabellnavn* (n), *tabellnavn* (n,n)

Dersom den maksimale størrelsen til tabellen ikke overskrider 10, kan DIM-setningen utelates.

For å referere til et tabellelement i f.eks. et uttrykk, brukes navnet etterfulgt av indeks, satt i parenteser.

A (14,K), N9 (B), L$ (14,4)

DIM står for dimensjon, slik at setningen ikke bare navngir datatabellen, men også dimensjonerer den (angir den maksimale størrelsen på den). Tall kan være et vilkårlig positivt tall, eller en enkel variabel, forutsatt at den enkle variabelen allerede har fått verdi. Definer ikke datatabellene større enn du trenger dem, da store tabeller opptar mye lagerplass.

```
10 DIM A (22),NA$ (40)
```

Linjen over ber datamaskinen etablere en datatabell som heter A og består av 22 elementer og en strengtabell kalt NA$ med 40 elementer. (Virkelig antall elementer er 23 og 41, fordi indeksen starter med 0). For å kunne referere til et tabellelement i et uttrykk, må man ha med indeksen (etter navnet).

```
25 A (4) = 7.0
32 A (M) = B*C/A
```

Linje 25 vil gi element 4 i datatabell A verdien 7. Linje 32 vil beregne uttrykket og plassere resultatet i element nummer M i tabellen A. Legg spesielt merke til A på høyre siden av uttrykket i linje 32 er en enkel variabel kalt A og at den har ingenting å gjøre med tabellvariabelen A på venstre side av uttrykket. Tabeller kan også ha to dimensjoner, slik at programlinjen:

```
10 DIM T (10,5),TB$ (12,4)
```

vil danne en numerisk tabell T med 10 rader og fem kolonner. Tilsvarende vil strengtabellen TB$ ha 12 rader og 4 kolonner. En lærer kan f.eks. ønske å registrere eksamensresultatene til 25 elever i seks forskjellige fag. En datatabell EKSAMEN (25,6), vil være hensiktsmessig til dette formålet. For å kunne referere til et element i tabellen, trenger du to indekser.

```
25 PRINT EKSAMEN (10,3)
```

presenterer resultatet for den 10. eleven i det 3. faget. Selvfølgelig må indeksene vise til dataelementer som virkelig eksisterer. Forsøk på å bruke EKSAMEN (30,7) vil forårsake feil, fordi du ikke definerte tabellen så stor.

Nå følger et eksempel hvor vi bruker datatabeller. Programmet er kort, men krever litt tankearbeid for å forstå hvordan det fungerer. Programmet brukes for å blande en vanlig kortstokk. Hvert kort har fått et nummer fra 1 til 52 i en tabell X. Enkeltelementer velges så tilfeldig fra X og plasseres i en tabell Y. Når programmet er ferdig inneholder Y tallene 1 til 52, men i en tilfeldig rekkefølge (stokket). Du kan ikke bruke generatoren som gir tilfeldige tall direkte, for å produsere en stokket kortstokk, fordi du bare kan ha et av hvert kort. RND-funksjonen kan kommme opp med det samme tallet mer enn en gang. Linje 50 skriver den stokkede tabellen. Legg spesielt merke til hvordan den bruker et uttrykk som en tabellindeks.

```
10 DIM X (52),Y (52): CLS
20 FOR I = 1 TO 52: X (I) = I: NEXT I
30 FOR I = 52 TO 1 STEP -1
40 J = RND (I): Y (I) = X (J): X (J) = X (I): NEXT I
50 FOR I = 1 TO 13: FOR J = 1 TO 4: PRINT Y (4 * (I-1)+J); : NEXT J,I:
   END
```

Elementene i en strengtabell kan flyttes på samme måte. En av de mest vanlige anvendelsene av strenger er å sortere en liste i alfabetisk rekkefølge. I det følgende eksempelet blir en liste som inneholder ord sortert av et delprogram som starter på linje 200. Det er skrevet mange bøker om sortering ved hjelp av datamaskin og den metoden som her brukes er den enkleste. Kjenner du ikke metoden, bør du arbeide deg gjennom programmet for hånd ved kun å bruke dataene D,B,A,E,C.

```
10 CLS : DIM W$ (50)
20 INPUT"HVOR MANGE ORD"; N
30 CLS : PRINT"OPPRINNELIG"
40 FOR I = 1 TO N: PRINT I;".▽▽";
50 INPUT W$ (I): NEXT I
60 GOSUB 200
70 PRINT @ 18,"SORTERT"
80 FOR I = 1 TO N
90 PRINT @ 18 + 32*I,I;".▽▽"; W$ (I)
100 NEXT I: END
200 M = N
210 F = 0: FOR I = 1 TO M-1
220 IF W$ (I)<=W$ (I+1) THEN 240
230 T$ = W$ (I): W$ (I) = W$ (I+1): W$ (I+1) = T$: F = 1
240 NEXT I: IF F = 1 THEN M = M-1: GOTO 210 ELSE RETURN
```

Hva er en funksjon

Husker du RND og hvordan vi stadig kalte den en funksjon. Den er ikke den eneste. En funksjon er i datamaskinsammenheng et spesielt delprogram, som når det får noen argumenter, returnerer en enkelt verdi. En funksjon i Basic har formen:

Funksjonsnavn (argumenter)

og det kan brukes i uttrykk på samme måte som andre aritmetiske operatorer (↑,*,/,+,-). Funksjoner vil imidlertid ha høyere prioritet enn alle andre operatorer unntatt parenteser.

Argumentene til en funksjon er de verdiene som gis til funksjonen. Funksjonen returnerer så resultatet. Argumentene kan være konstanter, variabler eller uttrykk.

RND (10),RND (X) eller RND (A*2+F)

er alle lovlige argumenter. Legg merke til at argumentene alltid er omsluttet av paranteser.

Datamaskinen har en rekke standardfunksjoner, innebygde funksjoner som er en del av Basic-språket. De innebygde funksjonene kan vi dele opp i fem klasser. Vi skal ta for oss hver klasse og presentere funksjonene med korte forklaringer og eksempler. Funksjonene vil forekommme i programmer fra nå av slik at vi ikke vil gi programeksempler for hver enkelt. Til det er de alt for mange.

Klasse 1 funksjoner

Dette er numeriske funksjoner som i hovedsak dreier seg om matematikk. De har numeriske argumenter og returnerer numeriske verdier. Klasse 1 funksjonene kan bare brukes i numeriske uttrykk. De som ikke er kjent med trigonometriske funksjoner bør lese Tillegg D, bak i boken.

Funksjons-navn	*Operasjon*	*Eksempel*
ABS (X)	Absoluttverdien til X	100 A = ABS (D*2-C)
ATN (X)	Arctangens til X, i *radianer*. Den inverse til TAN (X).	110 PRINT"VINKEL-"; ATN (R3)
COS (X)	Cosinus til X, hvor X er en vinkel målt i *radianer*.	510 F7 = COS (X+4)

Funksjons-navn	*Operasjon*	*Eksempel*
EXP (X)	Opphøy grunntallet e, (naturlig logaritme) med eksponenten x, (e^x). Den inverse til LOG (X).	215 Q = EXP (-A*A)
FIX (X)	Returnerer heltallsdelen av X, (dvs. kutter av alle siffer etter desimalpunktum) altså "trunkering".	172 N = FIX (Z*.05)
INT (X)	Kutter av som for FIX dersom X er positiv. Dersom X er negativ, rundes av nedover (dvs. INT (-12.001) er -13).	280 P = INT (100*X)
JOYSTK (X)	Returnerer den aktuelle horisontale eller vertikale posisjonen til venstre eller høyre styrespak som følger: X = 0, horisontal venstre styrespak X = 1, vertikal venstre styrespak X = 2, horisontal høyre styrespak X = 3, vertikal høyre styrespak	1040 A = JOYSTK (0): B = JOYSTK (1)
LOG (X)	Naturlig logaritme til X. Verdien av argumentet må være større enn null. Den inverse til EXP (X).	617 L1 = 5.2*LOG (W4)
PEEK (X)	Returnerer innholdet i lageradressen X.	55 P = PEEK (65280)
POINT (X,Y)	Undersøker om en lavoppløsnings grafisk celle er på eller av. X må være i området 0 - 63 (horisontalt) og Y i området 0 - 31 (vertikalt). Returnert verdi -1 i tekstmodus eller 0 til 8 som angir fargekoden dersom på.	50 IF POINTS (5,A) = C THEN 210

Funksjons-navn	*Operasjon*	*Eksempel*
POS (X)	Returnerer den aktuelle skriveposisjon. De eneste argumentene er: 0 for skjermen og 1 for skriver.	168 IF POS (0)<30 THEN PRINT A$
PPOINT (X,Y)	Undersøker høyoppløsnings-grafikk celle. Returnerer 0 dersom cellen er av, ellers returneres farge-koden fra (1 - 8). X må være i området 0 - 255 og Y i området 0 - 191.	115 C = PPOINT (A1,A2)
RND (X)	Returnerer slumptall (heltallsverdier) mellom 1 og X. Dersom X = 0 retur-neres slumptall mellom 0 og 1.	220 PRINT@ RND (510);
SGN (X)	Returnerer fortegnet til argumentet: X negativ returnerer -1 X null returnerer 0 X positiv returnerer +1	412 Y = RND (ABS (N))* SGN (N)
SIN (X)	Sinus til X, hvor X er en vinkel målt i *radianer.*	205 S = SIN (K*PI/180)
SQR (X)	Kvadratroten til argumen-tet, hvor X ikke må være negativ. Dersom X er nega-tiv, returneres kvadrat-roten av ABS (X) med nega-tivt fortegn.	330 C = SQR (A*A+B*B)
TAB (X)	Tabuleringtilposisjon på skjermen.	520 PRINT TAB (7); "OVERSKRIFT"
TAN (X)	Tangens til X, hvor X er en vinkel målt i *radianer.* Den inverse til ATN (X):	840 R5 = B/TAN (EQ-5)

Klasse 2 funksjoner

Klasse 2 funksjonene har et numerisk argument, men returnerer en strengverdi. De kan bare brukes i strengoperasjoner.

Funksjons-navn	*Operasjon*	*Eksempel*
CHR$ (X)	Returnerer tegnet som har koden med desimalverdi X. X må være i området 0 - 255. Se kodelisten i Tillegg A.	20 M$ = CHR$ (143) +CHR$ (128)
HEX$ (X)	Beregner den heksadesimale verdien til et desimaltall X.	42 PRINT HEX$ (30)
STR$ (X)	Omformer et numerisk utrykk til en strengverdi. Motsatt av VAL (X$).	175 A$ = STR$ (12.49)

Klasse 3 funksjoner

Klasse 3 funksjonene er strengfunksjoner. Argumentene (det er vanligvis minst to), er en streng og et tall. De returnerer alle en strengverdi og må derfor kun inngå i strenguttrykk.

Funksjons-navn	*Operasjon*	*Eksempel*
LEFT$ (X$,N)	Returnerer de første N tegnene i strengen X$.	114 A$ = LEFT$ (B$,7)
MID$ (X$,M,N)	Returnerer N tegn fra strengen X$ med start i posisjon M. Dersom N utelates vil hele strengen til høyre for M bli returnert. M må være større enn 0.	760 K$ = MID$ (W$,I,4)
RIGHT$ (X$,N)	Returnerer de siste N tegnene i strengen X$.	340 T$ = RIGHT$ (Q$,B+7)
STRING$ (N,C)	Returnerer en streng med lengde N som består av tegnet definert av C.	400 A$ = STRING$ (5,67)
	Argumentet C kan enten være et tall, (ASCII-koden til tegnet) eller tegnet selv omsluttet av apostrofer.	410 PRINT STRING$ (32,"*")

Klasse 4 funksjoner

Disse funksjonene er blandede funksjoner som ligner på klasse 2. De har et strengargument og returnerer en numerisk verdi, og de kan følgelig kun forekomme i numeriske uttrykk.

Funksjons-navn	*Operasjon*	*Eksempel*
ASC (X$)	Returnerer desimalverdien til ASCII-koden til det første tegnet i streng-argumentet. Se kodene i Tillegg A.	715 P = ASC (F$)-64
INSTR (P,S$,T$)	Søker i strengen S$ etter strengen T$ med start i posisjon P i S$-strengen. Returnerer 0 om den ikke finnes, ellers returneres posisjonen til T$-strengen.	212 F = INSTR (N,X$,"AB")
LEN (X$)	Returnerer lengden av strengen X$. Alle tegn inklusiv blanke telles. Dersom strengen er tom returneres 0.	845 N = LEN (N$)
VAL (X$)	Omformer strengpresenta-sjonen av et tall til tall-format. Dersom strengen starter med et alfabetisk tegn blir 0 returnert. Motsatt av STR$ (X).	92 Z = VAL (AB$)

Klasse 5 funksjoner

Klasse 5 er systemfunksjoner. De har ingen argumenter.

Funksjons-navn	*Operasjon*	*Eksempel*
INKEY$	Undersøker tastaturet og returnerer tegnet til tas-ten som er nedtrykket (om det er noen). Returnerer en strengverdi slik at INKEY$ må brukes i et streng-uttrykk.	146 P3$ = INKEY$

Funksjons-navn	*Operasjon*	*Eksempel*
MEM	Finner omfanget av ledig lagerplass.	PRINT MEM
TIMER	Returnerer verdien til "klokken" som er i området 0 - 65535. Nullstilles ved å angi TIMER = 0. Timer intervallet er ca. 22 millisekund.	62 T1 = TIMER-T 63 TIMER = 0

Dine egne funksjoner

I tillegg til funksjonene som systemet gir deg, er det mulig å lage 26 egne funksjoner. De brukerdefinerte funksjonene defineres med følgende setning:

DEF FN *bokstav (formell parameter) = formel*

Bokstav er en av de 26 bokstavene A til Z. Den *formelle parameteren* er en bokstav som blir erstattet av funksjonsargumentet når funksjonen kalles. *Formelen* er et Basic-uttrykk som består av den formelle parameteren og eller andre variable. Andre funksjoner, både innebygde og brukerdefinerte, kan forekomme i uttrykket, men *en funksjon kan ikke kalle seg selv.*

Ligningen $y = ((x-3)^2 + (x-4)^4)/x^3$ kan oversettes direkte til en definert funksjon som:

```
25 DEF FNY (X) = ( (X-3)↑2+ (X-4)↑4)/X↑3
```

X er her *den formelle parameteren* og ikke et variabelnavn. Når funksjonen kalles senere i programmet, vil den bli erstattet av argumentet. Funksjonen brukes ved rett og slett å inkludere den i et uttrykk på samme måten som en innebygd funksjon.

```
150 Y (I) = FNY (X)+FNY (W)
```

Funksjoner kan altså brukes som hjelpeprogram (vanlig brukte operasjoner). Du har kanskje lagt merke til at tallene som skrives ut på skjermen som resultater har en tendens til å se litt uryddige ut. Datamaskinen prøver å være hjelpsom ved å skrive tallene med maksimal nøyaktighet. Noen ganger er denne nøyaktigheten ikke nødvendig og til tider kan den være en plage. Et eksempel på det er utskrift av kronebeløp. Følgende funksjon kan brukes for å skrive ut det ønskede antall desimalsifre, (D).

```
10 DEF FND (X) = INT (X*10↑D + 0.5)/10↑D
```

DEF FN

DEF FN-setningen brukes for å definere en numerisk funksjon.

DEF FN *navn (formell parameter) = formel*

Navnet kan være en hvilken som helst bokstav fra A til Z.

Den *formelle* parameteren kan være en bokstav og den blir erstattet av argumentet når funksjonen brukes. Det kan kun brukes en formell parameter.

Formelen beskriver operasjonene ved hjelp av den formelle variabelen og/eller andre variabler.

Brukerdefinerte funksjoner må få plass på en programlinje. En brukerdefinert funksjon kan benytte andre funksjoner, enten brukerdefinerte eller innebygde, i formelen. Bemerk at de ikke kan kalle (bruke) seg selv.

En funksjon må være definert før den brukes. Derfor bør funksjonene defineres i begynnelsen av programmet.

Andre matematiske funksjoner kan defineres som brukerfunksjoner på følgende måte:

```
10 DEF FNS (X) = 1/COS (X):'SEKANT
20 DEF FNI (X) = -ATN (X/SQR (-X*X+1))+1.5708
30 DEF FNH (X) = -EXP (X)/ (EXP (X)+EXP (-X))*2+1
40 DEF FNM (A) = INT ( (A/B-INT (A/B))*B+0.5)*SGN (A/B)
50 B = 8: PRINT FNM (13)
```

Legg merke til at X er en formell parameter, mens D ikke er noen formell parameter. Verdien til D må angis utenfor funksjonen, f.eks. ved hjelp av en INPUT-setning. Et eksempel på bruk av funksjonen:

```
205 PRINT FND (A)
```

Etter som alle trigonometriske funksjoner krever at argumentene erangitt i radianer, vil en funksjon som omformer fra grader til radianer være nyttig:

```
10 DEF FNR (X)= X/57.295779
```

Vi kan få et mer nøyaktig resultat ved å bruke:

```
10 DEF FNR (X) = X*ATN (1.0)/45
```

Konstanten PI kan dannes ved:

```
20 DEF FNP (X) = 4.0*ATN (1.0)
```

Legg merke til at i dette tilfellet har den formelle variabelen ingen effekt i det hele tatt. Den er kun tilstede fordi det kreves et argument.

Ettersom man ikke kan kalle en funksjon som ikke allerede er definert, er det hensiktsmessig å plassere alle funksjonsdefinisjonene i begynnelsen av programmet.

Andre inndata setninger

Den eneste måten vi hittil har vært i stand til å få verdier inn i variable på, har vært ved hjelp av INPUT-setningen. Det er ikke særlig hensiktsmessig. Du har kanskje lagt merke til at INPUT-setningen ikke aksepterer visse tegn. Starter du en streng med blanke, forsvinner de, og taster du et komma, mister du alt etter det. Det finnes et alternativ, LINE INPUT-setningen:

LINE INPUT "*melding*"; *strengvariabel*

LINE INPUT oppfører seg på samme måte som INPUT bortsett fra at den aksepterer alle tegn inkludert blanke og komma. Meldingen er den samme som for INPUT-setningen og strengvariabel kan være en hvilken som helst strengvariabel. Du kan bare ha en variabel i hver LINE INPUT-setning.

```
25 LINE INPUT"TAST INN EN TEKSTLINJE";L$
```

Ofte vil det i et program være nødvendig å initialisere et antall konstanter før programmet virkelig begynner å arbeide. Du kan selvfølgelig taste disse inn hver gang programmet skal kjøres ved hjelp av INPUT-setninger. Det finnes imidlertid en mer hensiktsmessig måte, hvor man gjør bruk av READ- og DATA-setningene. De benyttes alltid sammen og de har følgende form:

READ *liste av variable*
DATA *liste av verdier*

READ-setningen oppfører seg på samme måte som INPUT-setningen, men istedenfor å stoppe programmet og vente på at verdier skal tastes inn, ser den etter verdier i DATA-setningene som er en del av programmet. DATA-setningene kan plasseres hvor som helst i programmet. Dersom det er flere enn en DATA-setning, vil READ-setningen starte med det laveste linjenummeret og arbeide seg oppover.

```
10 DATA 1,2,3,4,5
20 FOR I = 1 TO 3
30 READ A: PRINT A: NEXT I
40 READ D,G: PRINT D,G
```

Eksempelet over vil lese første element i DATA-listen (1) og skrive det ut, så lese det andre (2), osv. Etter som hvert dataelement er lest, flytter

LINE INPUT

LINE INPUT-setningen mater en hel linje inn i en strengvariabel, inkludert kommaer og ledende blanke. Disse blir ikke akseptert (lest inn) av INPUT-setningen.

LINE INPUT "*melding*"; *strengvariabel*

Meldingen er en hvilken som helst melding i apostrofer. Den kan utelates, men dersom den finnes med, må den være adskilt fra *strengvariabelen* med semikolon. Strengvariabelen kan være en vilkårlig strengvariabel. Det kan kun forekomme en variabel i LINE INPUT-setningen. Den maksimale lengden til en linje, lagret av LINE INPUT-kommandoen er 255 tegn.

```
10 CLEAR 500: CLS
20 LINE INPUT"TAST DITT FULLE NAVN";N$
30 LINE INPUT"OG ADRESSE";A$
```

READ
DATA
RESTORE

READ-setningen leser neste element i en DATA-setning og tilordner verdien av elementet til den angitte variablen i variabellisten.

READ *liste av variabelnavn*

DATA-setningen lagrer data inne i programmet og kan befinne seg hvor som helst i programmet.

DATA *liste av verdier*

Både strengvariabler og numeriske variabler kan brukes i READ- og DATA-setninger under forutsetningen at rekkefølgen er korrekt. En streng må tilordnes en strengvariabel, osv. Elementer i listen skilles med komma (,).

RESTORE-setningen setter datapekeren tilbake til det første elementet i den DATA-setningen som har lavest linjenummer. Det vil si at neste READ-setning som blir utført leser inn verdier fra den første DATA-setningen i programmet.

RESTORE

Et lite programeksempel belyser bruken:

```
10 CLS : PRINT: PRINT: PRINT
20 READ A,B: IF A = -9999 THEN RESTORE: GOTO 20
30 PRINT A;"∇ + ∇5∇ER∇";: INPUT C
40 IF C = B THEN PRINT"RIKTIG"ELSE PRINT"GALT"
50 FOR D = 1 TO 600: NEXT D: GOTO 10
60 DATA 8,13,12,17,5,10,27,32,14,19,3,8
70 DATA 7,12,6,11,1,6,-9999,-9999
```

datapekeren seg til neste dataelement. Linje 40 leser de siste to dataelementene. Dersom du nå legger til linjen:

```
50 READ X: PRINT X
```

og utfører programmet igjen, vil du få feilmeldingen ?OD ERROR IN 50. Dette betyr at READ ikke har flere dataelementer i DATA-listen. OD står for Out of Data. Føy til nok en linje:

```
45 RESTORE
```

Når du nå kjører programmet, fungerer det OK og X får verdien 1. RESTORE-setningen flytter pekeren tilbake til begynnelsen av den første DATA-setningen. Strenger kan også legges inn i DATA-setninger. Du må passe på at en blanding av variabler i READ-setningen stemmer overens med rekkefølgen av verdiene i DATA-listen. Har du skrevet et program som bruker mange strenger, kan det være at du allerede har fått meldingen ?OS ERROR. Dette betyr at du ikke har nok lagerplass reservert for strengene. For å reservere mer lagerplass for strenger, kan du bruke CLEAR-setningen.

```
10 CLEAR 1000
```

Denne setningen reserverer plass for 1000 tegn (bytes) i lageret til lagring av strenger. Etter som CLEAR også setter alle variable til null, må du sørge for å bruke den kun i begynnelsen av et program.

En liten repetisjon

Innholdet av dette kapittelet sammen med kapitlene 1,2,3 og 5 utgjør kjernen av Basic-språket. Selv om det kommer flere setninger senere, er de på mange måter å betrakte som mere utfyllende stoff.

Alt stoffet vi har gått gjennom hittil, vil vi ofte gjøre bruk av i de følgende kapitlene. De utgjør en vesentlig del av de fleste programmene. Det kan være

CLEAR

CLEAR-setningen nullstiller alle variablene og reserverer plass for strenglagring.

CLEAR 500

reserverer 500 tegn (bytes) med lagerplass for strengvariabler. CLEAR-setningen kan også brukes for å angi den høyeste Basic-adressen i lageret, for å reservere plass for maskinspråkprogram.

CLEAR 200,14000

reserverer 200 tegn (bytes) for lagring av strenger og setter den høyeste adressen for Basic til 14000. Maskinspråkprogram kan nå bli lagret fra 14001 og oppover.

Blir CLEAR ikke brukt, vil 200 tegn med strengplass bli automatisk reservert.

at du nå er ivrig etter å komme igang med å tegne figurer ved hjelp av datamaskinen. Likevel vil det være lurt om du nå repeterer litt og forsikrer deg om at du forstår det vi hittil har gått gjennom. Det vil gjøre bruken av grafikk enklere.

Gå tilbake, repeter eksemplene og gjør endringer i dem slik at de passer med dine egne ideer. Vi avslutter dette kapittelet med to eksempler. Det første er en utvidelse av kortstokkingseksempelet i forrige kapittel. Programmet deler ut en kortstokk. Legg merke til følgende :

1. Selve utdelingen er nå et delprogram på linje 190.
2. Bruken av READ og DATA for å etablere starttabellene.
3. Linjene 130 og 140 som finner farge og hvilket kort innen fargen.

```
10 DIM X (52),STOKK (52),KORT$ (13),FARGE$ (3)
20 FOR I = 0 TO 3: READ FARGE$ (I): NEXT I
30 DATA SPAR,RUTER,KLOEVER,HJERTER
40 FOR I = 1 TO 13: READ KORT$ (I): NEXT I
50 DATA ESS,TOER,TREER,FIRER,FEMMER,SEKSER,SJUER
60 DATA AATTER,NIER,TIER,KNEKT,DRONNING,KONGE
70 CLS : INPUT "HVOR MANGE KORT SKAL DELES"; N
80 GOSUB 190
90 START = 1
100 SLUTT = START + N-1: IF SLUTT>52 THEN GOTO 80
110 CLS : PRINT@ 10,"DIN HAND":PRINT:PRINT
120 FOR I = START TO SLUTT
```

```
130 F = INT ( (STOKK (I)-1)/13)
140 K = STOKK (I)-F*13
150 PRINT TAB (8); FARGE$ (F);"-"; KORT$ (K)
160 NEXT I: START = START+N
170 PRINT@ 448,"NY HAND. JA ELLER NEI"; : INPUT A$
180 IF A$ = "JA"THEN 100 ELSE END
190 FOR I9 = 1 TO 52: X (I9) = I9: NEXT I9
200 FOR I9 = 52 TO 1 STEP -1
210 J9 = RND (I9): STOKK (I9) = X (J9): X (J9) = X (I9)
220 NEXT I9: RETURN
```

Det andre eksempelet bruker nesten alle tilgjengelige strengfunksjoner. Programmet kontrollerer en inntastet tekst og angir antall forekomster av hver bokstav. Denne type program brukes ofte for å dechiffrere (tyde) kodede meldinger. Med litt ekstra arbeide kan du utvide det til å søke etter ord eller sekvenser av tegn.

```
10 CLEAR 1000: CLS : READ A$
20 DATA ABCDEFGHIJKLMNOPQRSTUVWYZ
30 PRINT"TAST INN EN TEKSTLINJE": PRINT
40 LINE INPUT L$
50 FOR I = 1 TO LEN (A$): CLS
60 T$ = MID$ (A$,I,1): C = 0: P = 1: P$ = L$
70 F = INSTR (P,L$,T$)
80 IF F>0 THEN C = C+1 ELSE 140
90 P$ = LEFT$ (P$,F-1)+STRING$ (LEN (T$),CHR$ (128))
100 IF F>LEN (L$) THEN 120
110 P$ = P$+RIGHT$ (L$,LEN (L$)-F)
120 P = F+LEN (T$)
130 IF P< = LEN (L$)-LEN (T$)+1 THEN 70
140 PRINT P$
150 PRINT@ 354,"FUNNET"; C;"FOREKOMSTER AV"; T$
160 PRINT@ 416,"TRYKK BLANK FOR AA FORTSETTE, N FOR AA
    STOPPE"
170 Z$ = INKEY$: IF Z$ = ""THEN 170
180 IF Z$ = "N"THEN 200
190 NEXT I
200 CLS : END
```

7. Bilde og bevegelse

Byggesteinen er et punkt

Når datamaskinen viser noe på TV-skjermen, lager den punkter for å bygge opp bildet. Hvis punktet er tent, ser det ut som en farget flekk, ellers er det svart. Alle de bokstaver vi har skrevet er bygget opp av disse små lyspunktene. Størrelsen på punktet bestemmer du ved *oppløsningen* du arbeider med. Er punktet stort har vi *lav oppløsning*. Er punktet lite betyr det en *høy oppløsning* (fordi jo mindre punktet er, dess større antall punkter får plass på skjermen).

Datamaskinen kan arbeide i fire forskjellige oppløsninger som spenner fra 512 til 49 152 punkter på skjermen. Dette gir deg en stor valgfrihet når det gjelder den detaljeringsgrad du ønsker å legge inn i bildene dine. Vi vil begynne med å lage bilder i den laveste oppløsningen og så arbeide oss over i de høyere oppløsningene etter hvert. Fremgangsmåten for å lage tegninger og bevegelse på skjermen er i hovedsak den samme, uavhengig av den oppløsningen som du arbeider med.

Lage figurer

Fra kapittel 3 husker du sikkert at vi innørte PRINT@-setningen, hvor vi beskrev hvordan skjermen var delt inn i et nettverk på 16 x 32 ruter. Dette gjorde at vi kunne skrive et tegn hvor som helst på skjermen ved å angi den aktuelle posisjonen. Ved å bruke CHR$-funksjonen (se kapittel 6, funksjoner), kan vi nå danne spesielle grafiske tegn. Følgende program vil vise alle de tegn som er tilgjengelige ved hjelp av CHR$.

```
10 FOR I = 1 TO 255: CLS 0
20 PRINT @ 100,"CHR$ ("; I;")";
30 PRINT @ 120,CHR$ (I);
40 FOR D = 1 TO 600: NEXT D,I: CLS
```

Tallene fra 1 til 31 er brukt til kontrolltegn, slik at de ikke vises på skjermen. Fra 32 til 127 har vi tegnene på tastaturet. Kodene fra 128 til 255 er de spesielle grafiske tegnene. Du finner en fullstendig liste over de grafiske tegnene i Tillegg A. Disse grafiske tegnene er enkle fargemønstre som kan samles til enkle former. Det enkleste mønsteret er et farget rektangel. F.eks.

CHR$ (143) gir et grønt rektangel (grønn er farge 1). Legg til 16, og CHR$ (159) gir et gult rektangel (farge 2). Slik kan du fortsette. Det finnes 16 mønstre fra CHR$ (128) til CHR$ (143) som er bygget opp av grønt og svart. Ønsker du det samme mønsteret, men i en annen farge, kan du bare legge det nødvendige multiplum av 16 til koden.

+16 gul	+32 blå	+48 rød
+64 hvit	+80 turkis	+96 fiolett
+112 oransje		

Programmet nedenfor viser hva som skjer dersom du øker koden med 16. Programmet kan også brukes til å justere fargebalansen på TV-apparatet. Bruk da F = 143.

```
10 CLS 0: INPUT"TAST EN KODE FRA 128 TIL 143"; F
20 FOR I = 1 TO 14 : FOR J = F TO 255 STEP 16
30 FOR K = 1 TO 4: PRINT CHR$ (J); : NEXT K,J,I
40 GOTO 40
```

Som vist ovenfor kan CHR$-tegnene skrives direkte på skjermen. De er imidlertid et tegn slik at de også kan plasseres i streng-variable. Det er mer hensiktsmessig ettersom de da kan behandles på en enklere måte.

La oss nå starte med å konstruere en figur. Vi skal tegne en borg, først og fremst fordi den har en enkel form og for det andre fordi den viser oss hvordan vi kan starte med et enkelt utgangspunkt og så legge på flere og flere detaljer. Tast inn og kjør hver enkelt del etter som vi presenterer den, slik at du kan se trinnene som figuren går gjennom.

Først vil vi bygge en mur tvers over skjermen. Den vil altså være 32 blokker bred og vi gjør den 6 blokker høy.

```
10 CLEAR 500: CLS 0
20 FOR I = 1 TO 6: FOR J = 1 TO 32
30 MUR$ = MUR$+CHR$ (207): NEXT J,I
40 PRINT @ 256,MUR$;
200 GOTO 200
```

Den første linjen reserverer plass for strengen som vi vil gjøre bruk av. Linje 20 og 30 bygger muren ved hjelp av hvite blokker, CHR$ (207), og lagrer den i en streng som kalles MUR$. Linje 40 skriver MUR$ som kommer fram på skjermen som en masiv fargeblokk. Den siste linjen (200) er kun til for å beholde bildet på skjermen.

Det neste vi gjør er å føye til brystvern. Dette gjør vi med blokker av hvitt og svart. Vi trenger kun en rad med dette.

```
50 FOR I = 1 TO 16: B$ = B$+CHR$ (128)+CHR$ (207): NEXT I
60 PRINT @ 224,B$;
```

Linje 50 konstruerer brystvernet og linje 60 tegner det på toppen av muren.

Nå trenger vi et tårn. Tårnet bygger vi av samme materiale som muren, så la oss ta noen blokker fra MUR$.

```
70 P = 11
80 FOR I = 1 TO 3: PRINT @ 128+32*I+P,LEFT$ (MUR$,10); : NEXT I
```

Linje 80 tar 10 blokker fra MUR$ og bygger 3 rader på midten av muren. Verdien til P sørger for å plassere tårnet på midten. Ønsker du å plassere tårnet på et annet sted, kan du forandre P. Tårnet er 10 blokker bredt, slik at P kan være hvilket som helst tall fra 0 til 22. Vi kan nå også bygge brystvern på tårnet.

```
90 PRINT @ 128+P,LEFT$ (B$,10);
```

Dette fullfører hovedkonstruksjonen av borgen. Vi kan føye til skyteskår ved å bruke et annet tegn og skrive over.

```
100 FOR I = 2 TO 32 STEP 4: PRINT @ 288+I,CHR$ (206); : NEXT I
```

Vi trenger også en port, så derfor skriver vi svarte blokker på det aktuelle stedet.

```
110 G$ = CHR$ (128)+CHR$ (128)+CHR$ (128)
120 FOR I = 353 TO 417 STEP 32: PRINT @ I+P+5,G$; : NEXT I
```

Porten blir lagret i G$ og skrives av linje 120, som gjør bruk av P i PRINT@-uttrykket for å plassere porten under tårnet.

En borg med en port som alltid er åpen er ikke til mye nytte, derfor trenger vi et fallgitter. Ved å bruke et annet tegn (142+112), kan vi konstruere noe som ligner på et fallgitter.

```
130 P$ = CHR$ (254)+CHR$ (254)+CHR$ (254)
140 FOR I = 353 TO 417 STEP 32: PRINT@I+P+5,P$;
150 FOR K = 1 TO 300: NEXT K,I
```

Linje 130 konstruerer et oransje fallgitter som skrives fra toppen og nedover. Forsinkelsessløyfen K forårsaker at det blir senket sakte.
Vi vil forlate borgen her, men det kan være at du vil legge til andre detaljer som en blå vanngrav, et flagg osv.

Bevegelige figurer

I borgeksempelet viste linjene 140 og 150 oss hvordan vi kunne få litt bevegelse inn i bildet ved å skrive delene i rekkefølge. Denne typen bevegelse er begrenset til i hovedsak å åpne og lukke dører. En langt bedre metode er å skrive hele bildet og så stryke det ut og skrive det igjen i en litt annen posisjon.

Ettersom du da stadig tegner om igjen figuren, bør tegnedelen være et delprogram. La oss først se på figuren som vi vil konstruere i en 3x4 blokk. Den øverste linjen skal være hodet, den neste kroppen og den siste benene.

```
10 CLEAR 500: CLS 0
20 M1$ = CHR$ (128)+CHR$ (193)+CHR$ (194)+CHR$ (128)
30 M2$ = CHR$ (196)+CHR$ (207)+CHR$ (207)+CHR$ (200)
40 M3$ = CHR$ (128)+CHR$ (202)+CHR$ (197)+CHR$ (128)
```

Figuren er nå laget i tre strengvariable M1$, M2 og M3$. Her er delprogrammet som skriver strengene i riktig rekkefølge:

```
500 P = 32*Y+X
510 PRINT @ P,M1$; : PRINT @ P+32,M2$;
520 PRINT @ P+64,M3$; : RETURN
```

Denne programbiten vil presentere figuren som tre linjer direkte under hverandre og starter i et punkt bestemt av X og Y. Husk X og Y er koordinater i rutenettet for PRINT@. X er 0 til 31 horisontalt og Y er 0 til 15 vertikalt. Du må taste inn alle linjene til og med 160 før du kan kjøre programmet.

For å kunne flytte figuren trenger vi å kunne endre skriveposisjonen. Dvs. å endre X og Y. Det kan vi gjøre fra tastaturet. Vi vil gjøre bruk av INKEY$ for å avlese tastaturet, og de naturlige tegnene å gjøre bruk av er piltastene. Disse tastene har også koder på samme måte som bokstavene:

```
[←] CHR$ (8)
[→] CHR$ (9)
[↓] CHR$ (10)
[↑] CHR$ (94)
```

Vi vil bruke X til å inneholde den horisontale posisjonen til figuren og Y for å inneholde den vertikale posisjonen. Er ←tasten trykket ned, ønsker vi å bevege figuren til venstre, slik at vi reduserer den aktuelle X verdien med en. Vi må imidlertid sørge for at vi ikke går over kanten på skjermen.

```
90 GOSUB 500
100 A$ = INKEY$: IF A$ = ""THEN 100
120 IF A$ = CHR$ (8) THEN X = X-1: IFX < 0 THEN X = 0
```

Linje 100 avleser tastaturet til en tast er trykket ned. Er det en ←tast vil linje 120 redusere verdien til X med en, undersøke om vi er utenfor skjermer og dersom vi er det, stoppe bevegelsen mot venstre. For å bevege mot høyre oppover og nedover er mønsteret helt tilsvarende.

```
130 IF A$ = CHR$ (9) THEN X = X+1: IF X > 28 THEN X = 28
140 IF A$ = CHR$ (94) THEN Y = Y-1: IF Y < 0 THEN Y = 0
150 IF A$ = CHR$ (10) THEN Y = Y+1: IF Y > 13 THEN Y = 13
```

```
160 GOSUB 500: GOTO 100
```

I linje 130 satte vi den maksimalt tillatte verdien av X til 28. Den kan ikke bli større enn 31 på noen måte. Husk at figuren vår er 4 blokker bred. Det samme gjelder Y, vi må sette av plass for å skrive de 3 linjene. Linje 160 kaller delprogrammet som skriver og kommer så tilbake igjen for å undersøke om det er nye tastetrykk. Kjører du programmet *nå*, vil du være i stand til å bevege figuren rundt på skjermen, men det blir bare rot fordi vi ikke fjerner figuren i den gamle posisjonen. For å kunne gjøre det, trenger vi en blank streng og et delprogram for å skrive den.

```
50 BL$ = CHR$ (128)+CHR$ (128)+CHR$ (128)+CHR$ (128)
600 P = 32*Y+X: PRINT @ P,BL$;
610 PRINT @ P+32,BL$; : PRINT @ P+64,BL$;
620 RETURN
```

Det nye delprogrammet (600) gjør nøyaktig det samme som det første, bortsett fra at det denne gang skriver en blokk med svarte firkanter. Alt vi nå trenger å gjøre, er å stryke ut figuren idet vi skal til å bevege den.

```
110 GOSUB 600
```

Du bør nå være i stand til å bevege figuren overalt på skjermen. I sin nåværende form er dette programmet bare et eksempel, men bevegelige figurer som denne kan bygges inn i spill og enkle undervisningsprogram.

En bedre oppløsning

Vi går nå over til en finere oppløsning. Denne oppløsningen har et 32x64 rutenett som gir 2048 punkter på skjermen. Denne oppløsningen og den forrige på 16x32 kalles for *lavoppløsnings*-skjermene og kan brukes sammen hvis vi ønsker det. Til å tenne og slukke punkter på denne skjermen finnes det to kommandoer:

```
SET (X,Y,F) og RESET (X,Y)
```

SET-kommandoen *tenner punktet* X,Y i fargen F. X har verdi fra 0 til 63 og Y fra 0 til 31 og er henholdsvis den horisontale og vertikale posisjonen på samme måte som tidligere. F er en kode for den fargen du ønsker på flekken, og er et tall mellom 0 og 8.

RESET-kommandoen *slukker punktet* X,Y. Ved å bruke disse kommandoene kan man få til bevegelse ved å tenne og slukke punkter i rekkefølge.

Forsøk programmet nedenfor:

```
10 CLS 0: X1 = 0: Y1 = 0: XI = 2: YI = 2
20 X2 = X1+XI: IF X2>63 OR X2<0 THEN XI = -XI: SOUND 180,1:
   GOTO 20
30 Y2 = Y1+YI: IF Y2>31 OR Y2<0 THEN YI = -YI: SOUND 180,1:
   GOTO 30
40 SET (X2,Y2,8): RESET (X1,Y1): X1 = X2: Y1 = Y2: GOTO 20
```

Du kan se hva programmet gjør, men hvordan skjer det Det starter med et punkt X1,Y1. Programmet øker verdien til X1 med et lite tillegg XI og verdien til Y1 med YI og danner på denne måten et nytt punkt X2,Y2. Det nye punktet blir tent og det gamle (X1,Y1) blir slukket i linje 40. X2,Y2 punktet blir så til det gamle punktet (X1,Y1), og programmet går tilbake til linje 20 for å danne et nytt X2,Y2. Dette beveger ballen over skjermen. Når ballen når kanten på skjermen blir tilleggets fortegn endret. Det betyr at X f.eks. nå begynner å gå ned i verdi, noe som forårsaker at retningen forandrer seg. Dette forårsaker sprettingen fra kantene. Forandrer du størrelsen på tillegget (XI og YI i linje 10),kan du få ballen til å bevege seg med forskjellige hastigheter. Denne typen program er grunnlaget for de fleste ballspill på datamaskiner, men disse er vanligvis skrevet i maskinspråk og ikke i Basic.

En annen anvendelse av bevegelige punkter finnner vi i "skytespill". Slike spill krever bevegelse over skjermen og muligheten til å avfyre et våpen. Vi kunne fortsatt med å bruke piltastene som tidligere, men det er mye bedre å bruke styrespaker. Styrespakene plugges inn i kontakter på siden av datamaskinen og gir mulighet for en bedre kontroll over bevegelser enn det piltastene gir. Styrespakposisjonen avleses av funksjonen JOYSTK. JOYSTK (0) returnerer den horisontale posisjonen til den høyre styrespaken og JOYSTK (1) den vertikale posisjonen. JOYSTK (2) og JOYSTK (3) gjør tilsvarende for den venstre styrespaken. Verdien som returneres av funksjonen er i hvert av tilfellene mellom 0 og 63 og det er derfor nødvendig å skalere verdien slik at den passer skjermoppløsningen som du arbeider med.

```
10 CLS 0: FOR I = 0 TO 3
20 PRINT @ 74+32*I,"STYRESPAK (";I;")▽▽";JOYSTK (I);
30 NEXT I: FOR D = 1 TO 400: NEXT D: GOTO 10
```

Kjør programmet ovenfor og beveg styrespakene. Du vil da se at verdiene forandrer seg med posisjonen til styrespakene. Du kan også bruke knappen på styrespaken. Føy til linjen:

```
25 P = PEEK (65280): PRINT @ 202,"KNAPP-VERDI ▽▽";P;
```

PEEK-funksjonen forteller datamaskinen at den skal se etter på et spesielt sted i lageret. Lageradresse 65280 inneholder resultatet av kontrollen med styrespak-knappen. Det vil enten være 127 eller 255. Trykker du ned den

SET

SET-setningen brukes for å slå på et spesifisert punkt med en spesifisert farge på lavoppløsningsskjermen.

SET *(x,y,f)*

x,y er koordinatene til skjermpunktet. x må være i området 0 til 63 og y i området 0 til 31.

f er fargekoden til den ønskede fargen. Den må være et tall mellom 0 og 8.

```
10 CLS 0: SET (5,27,8): SET (6,27,8)
20 FOR X = 0 TO 6: FOR Y = 28 TO 30
30 SET (X,Y,8): NEXT Y,X
40 FOR X = 7 TO 63: FOR D = 1 TO 200: NEXT D
50 FOR Y = 27 TO 30: IF Y = 27 THEN RESET (X-2,Y)
60 SET (X,Y,8): RESET (X-7,Y): NEXT Y,X
70 GOTO 70
```

RESET

RESET-setningen brukes for å slå av et punkt som er slått på av SET-setningen. Den brukes med lavoppløsningsgrafikk.

RESET *(x,y)*

x,y er koordinatene til punktet som skal slås av. x må være mellom 0 og 63 og y mellom 0 og 31.

Punktet blir satt til bakgrunnsfargen, som forårsaker at det blir "visket ut". Se eksemplet under SET-setningen.

venstre knappen, vil det forandre seg til 125 eller 253. Trykker duned den høyre, vil det forandre seg til 126 eller 254. Er begge knappene trykket ned samtidig, vil tallene være 124 eller 252.

La oss så starte arbeidet med et spillprogram Et slag mellom to romskip. Vi kan bruke styrespakene til å bevege romskipene og knappene til å avfyre våpen.

Først må vi konstruere romskipene og vi gjør da bruk av en lignende metode som den vi har brukt tidligere. Hvert romskip er en blokk på 2x3, en gul og den andre blå og de lagres i strengtabellene S$ og S2$.

```
10 CLEAR 500: FOR I = 0 TO 5: READ S (I): NEXT I
20 DATA 128,131,128,134,140,137
30 FOR Y = 0 TO 1: F = (Y+1)*16
40 S$ (Y) = CHR$ (S (0)+F)+CHR$ (S (1)+F)+CHR$ (S (2)+F)
50 S2$ (Y) = CHR$ (S (3)+F)+CHR$ (S (4)+F)+CHR$ (S (5)+F)
60 NEXT Y
```

Liker du ikke formen på romskipene, kan du konstruere dine egne og forandre data i linje 20.

Det neste vi må gjøre er å avlese styrespakene, kontrollere at vi fortsatt er på skjermen og finne ut hvor vi skal plassere romskipene.

```
70 FOR Y = 0 TO 1: A (Y) = JOYSTK (Y*2)
80 B (Y) = INT (JOYSTK (1+Y*2)/2)
85 IF A (Y)>58 THEN A (Y) = 58
90 IF A (Y)<2 THEN A (Y) = 2
100 IF B (Y)>27 THEN B (Y) = 27
110 L (Y) = INT (B (Y)/2)*32+INT (A (Y)/2): NEXT Y
```

Styrespakposisjonene leses inn i rekkefølge av linje 70 og 80. Grensene blir satt i linjene 85 til 100 (husk romskipsstørrelsen). Det endelige resultatet blir så omformet til en verdi til PRINT@-kommandoen. Dette er et eksempel på blanding av de to lavoppløsningskjermene. Styrespakene arbeider i den ene, mens PRINT@-kommandoen arbeider i den andre.

Nå må vi plassere romskipene og gå tilbake for å se om styrespakene er beveget.

```
120 CLS 0: FOR Y = 0 TO 1: PRINT @ 0,Z (0); : PRINT @ 26,Z (1);
130 PRINT @ L (Y),S$ (Y); : PRINT @ L (Y)+32,S2$ (Y); : NEXT Y
170 A$ = INKEY$: IF A$ = ""THEN 70
180 CLS : END
```

Linje 120 skriver også resultatet, men den delen har vi ikke gjort ferdig ennå. For å avslutte spillet, kan vi trykke hvilken som helst tast ellers går programmet til linje 70 og avleser styrespakene igjen. Kjør programmet så langt og kontroller at romskipene kan bevege seg overalt på skjermen.

Neste trinn er å avfyre kanonene og vise lynet. Dette er den vanskelige delen, fordi vi må avlese styrespak-knappene og avgjøre hvem det er som skyter. Og fordi skipene er i stand til å bevege seg overalt på skjermen, må vi vite retningen på lynet. For å gjøre det enkelt, vil vi kun tillate at lynet beveger seg fra skipet som avfyrer det langs en horisontal linje mot målskipet. Det vil altså bevege seg i den vertikale høyden til skipet som skyter.

```
140 P = PEEK (65280)
150 IF P = 125 OR P = 253 THEN F = 1: T = 0: GOSUB 200
160 IF P = 126 OR P = 254 THEN F = 0: T = 1: GOSUB 200
```

Disse linjene avleser knappene og avgjør hvilket av skipene som skyter. Lynet blir laget av delprogram 200.

```
200 V1 = B (F): H1 = A (F): H2 = A (T): ST = 1
210 IF H1>H2 THEN ST = -1
220 FOR H = H1+ST*5 TO H2+2 STEP ST
240 SET (H,V1,4): SOUND 200,1: RESET (H-2*ST,V1)
250 NEXT H: RETURN
```

Legg merke til ombyttingen av trinnene i linje 210, dersom venstre- og høyreposisjonene blir byttet om. Selve bevegelsen av lynet gjøres i linje 240.

Alt som nå gjenstår er å undersøke om et romskip er truffet og i såfall lage passende støy og registrere treffet.

Funksjonen POINT brukes for å registrere et treff. Formen er POINT (X,Y), hvor X,Y er punktet som vi ønsker å undersøke. Funksjonen returnerer 0 hvis punktet er av og tallet for fargekoden dersom det er på.

Ettersom skjermen er svart og vi skyter i riktig retning, så trenger vi bare å vite om det finnes noe farget punkt i skuddlinjen. Føyer vi til følgende linje i vårt delprogram

```
230 IF POINT (H,V1)>0 THEN GOSUB 300: RETURN
```

vil det når et treff finner sted, skje et kall av delprogram 300, hvor vi holder styr på poeng osv. Når kontrollen kommer tilbake har det ikke noen hensikt å skyte mer, slik at vi forlater også dette delprogrammet og starter på nytt igjen.

```
300 Z (F) = Z (F)+1
310 FOR K = 1 TO 15: I = RND (5)-2: J = RND (4)-2
320 SET (H+I*ST,V1+J,8): SOUND (RND (95)),1
330 NEXT K: RETURN
```

Dette delprogrammet holder orden på poengene, tegner og lager lyd for eksplosjon.

Selv om programmet kun består av 28 linjer, er dette programmet tilstrekkelig for å produsere et spill som inneholder mye bevegelse. Vi overlater til deg å utvikle spillet videre med forbedringer som bringer det opp på et mer profesjonelt nivå.

Dette har vært et langt og innviklet kapittel, men det inneholder de fleste elementer som er nødvendige for å utnytte grafikken på din datamaskin, samme hvilken oppløsning du bruker.

8. Over til høyoppløsning

Vi går nå over til høyoppløsningsskjermene, som er fullstendig adskilt fra lavoppløsningsskjermene. De to lavoppløsningsskjermene kan brukes sammen og blir presentert på hva vi kaller tekstskjermen.

Høyoppløsningsskjermene kan ikke blandes med tekstskjermen. Du kan veksle fra den ene til den andre, men du kan ikke skrive tekst på høyoppløsningsskjermen eller tegne høyoppløsningsgrafikk på tekstskjermen.

Når noe tegnes i høyoppløsning, vil datamaskinen gi instrukser til en spesiell del av sitt lager som kalles video RAM (bilde-hukommelse) om hvordan informasjonen skal presenteres. Video RAM blir så lest til fjernsynet og omformet til bilder. Det blir reservert et antall sider (pages)i video RAM for dette formålet. Det normale antallet er fire. Når detaljeringsgraden du bruker øker, vil antallet instruksjoner som trengs for å vise resultatet også øke. Flere instruksjoner trenger mer plass og dermed må du reservere flere sider. Dette gjøres med PCLEAR, etterfulgt av antall sider som du vil reservere. Maksimalt antall sider er 8.

PCLEAR 8

Da hver side belegger 1536 (1,5k) lagerplasser, bør du ikke reservere mer enn du virkelig trenger. Tilgjengelig lagerplass er begrenset, slik at jo mer du benytter for grafiske sider, dess mindre blir tilgjengelig for program. PCLEAR oppfører seg på en tilsvarende måte som CLEAR og bør derfor kun brukes i begynnelsen av et program.

Valg av oppløsning

Den plassen du trenger å reservere er avhengig av oppløsningen du vil bruke. En ulempe ved økende oppløsning er at det ikke er mulig å benytte hele fargeskalaen som er tilgjengelig i lavoppløsning. De tilgjengelige fargene og oppløsningen bestemmes av den modus du arbeider i. Modus blir satt med PMODE-setningen.

PCLEAR

PCLEAR-setningen brukes for å reservere grafiske sider i høyoppløsningsmodus.

PCLEAR *n*

n må være et tall mellom 1 og 8. Dersom PCLEAR-setningen er utelatt i programmet, er PCLEAR 4 standardverdien.

Etter som hver grafisk side krever 1536 tegn (1,5 Kbytes) med lagerplass, så sørg for å reservere bare det du trenger.

PMODE *modus, startside*

Modus er et tall fra 0 til 4 og *startside* er det sidenummeret i video RAM som du vil skrive til først. Som tidligere er skjermene delt i rutenett. Denne gangen er det imidlertid bare nødvendig å huske en størrelse (256 x 192). Selv om oppløsningen endrer seg med forskjellige modus, vil du fortsatt referere til punkter på skjermen ved å bruke rutenett på 256 x 192. Forskjellen er i størrelsen på punktene som blir tegnet. Modusen du velger avgjør også hvilke farger du kan bruke. Hver modus har to fargepaletter. Fargepalettene velges med SCREEN-kommandoen, hvor du også velger skjermtype (lav-/høyoppløsning).

SCREEN *type, fargepalett*

Type er 0 for tekstskjerm og 1 for høyoppløsningsskjerm. *Fargepaletten* er også enten 0 eller 1. Normalverdien som vi har brukt inntil nå er SCREEN 0,0. Det gir tekstskjerm med svart på grønn bakgrunn. (Det er mulig å bruke SCREEN 0,1 som gir svart tekst på orange bakgrunn, men hver gang datamaskinen skriver vil den gå tilbake til svart på grønn). For å presentere høyoppløsning, må du sette *type* til 1. Tabellen i rammen for SCREEN-kommandoen viser alle muligheter og fargepaletter.

Som du ser av tabellen, henger oppløsningen og fargepalettene nøye sammen. Du vil også se at ettersom modus øker fra 0 til 4 vil antall sider som er nødvendige også øke.

For å vise en grafikkskjerm, vil PMODE 0 bare trenge en side med lagerplass, mens PMODE 3 og PMODE 4 krever fire sider. Når fargepaletten er valgt, vil datamaskinen velge den farge som har lavest tall fra paletten som bakgrunnsfarge. Den fargen med høyest nummer i paletten brukes som forgrunnsfarge. F.eks. vil datamaskinen med PMODE 3 og SCREEN 1,0, tegne med rødt på en grønn bakgrunn. Du kan endre forgrunns- og bakgrunnsfargene med COLOR-setningen.

COLOR *forgrunn, bakgrunn*

hvor *forgrunn og bakgrunn* er blant de fargene som kan brukes i den aktuelle modus.

Punkter

Du husker sikkert setningene CLS , SET, RESET og POINT fra lavoppløsningsskjermene. Det finnes også tilsvarende setninger for høyoppløsning. De kalles PCLS , PSET, PRESET og PPOINT for å skille dem fra lavoppløsning. De gjør det samme som tidligere:

PCLS klargjør høyoppløsningsskjermen og om den etterfølges av en fargekode vil den sette bakgrunnen til den fargen. PSET tenner et punkt og PRESET slukker et punkt. PPOINT avgjør om et punkt er tent eller slukket. Det følgende eksempelet arbeider seg gjennom alle modus og fargepaletter etter tur. Det setter punkter av vilkårlig farge på skjermen. Punktene skal være i et rektangulært rutenett. De blanke plassene i rutenettet kommer av at den vilkårlige fargen er den samme som bakgrunnsfargen, eller ikke tilgjengelig i fargepaletten.

```
10 FOR P = 0 TO 4: PMODE P,1
20 FOR S = 0 TO 1: SCREEN 1,S
30 PCLS : FOR I = 50 TO 150 STEP 20
40 FOR J = 50 TO 150 STEP 20
50 F = RND (8): PSET (I,J,F): NEXT J,I
60 FOR D = 1 TO 1000: NEXT D,S,P
```

Se nøye på størrelsen til punktene. De viser oppløsningen som er tilgjengelig i den modusen.

Tegn en linje

Nå kan vi altså plassere punkter på skjermen. Ved hjelp av LINE-setningen kan vi trekke en linje mellom to punkter. Fjern linjene 40 og 50 fra det siste eksempelet og endre linje 30 til:

```
30 PCLS : LINE (10,180)- (245,10),PSET
```

Kjør programmet. Det blir tegnet en linje tvers over skjermen fra nede til venstre til oppe til høyre. Setningen betyr tegn en linje fra startpunktet (10,180) til sluttpunktet (245,10) i forgrunnsfargen (PSET). Forandrer du PSET til PRESET, blir linjen tegnet med bakgrunnsfargen. Å tegne med bakgrunnsfargen betyr at den ikke kan ses. Det kan også brukes for å stryke

PMODE
SCREEN

SCREEN-setningen brukes for å skifte skjermen mellom grafisk modus og tekstmodus.

SCREEN *type, fargeskala*

Type er enten 0 for tekstskjerm og lavoppløsningsgrafikk, eller 1 for høyoppløsningsgrafikk.

Fargeskala er enten 0 eller 1. Fargeskalaen for tekstskjermen er 0, svart på grønt eller 1, svart på oransje. For grafikk er fargeskalaen avhengig av arbeidsmodus på følgende måte:

PMODE *tall*	*Rutenett størrelse*	*Punkt størrelse*	*Sider brukt*	*Tilgjengelig fargeskala* SCREEN 1,0	SCREEN 1,1
0	128 x 96	■	1	Svart (0), Grønn (1)	Svart (0), Hvit (5)
1	128 x 96	■	2	Grønn (1) Gul (2) Blå (3) Rød (4)	Hvit (5) Turkis (6) Fiolett (7) Oransje (8)
2	192 x 128	▬	2	Svart (0), Grønn (1)	Svart (0) Hvit (5)
3	192 x 128	▬	4	Grønn (1) Gul (2) Blå (3) Rød (4)	Hvit (5) Turkis (6) Fiolett (7) Oransje (8)
4	256 x 192	▬	4	Svart (0) Grønn (1)	Svart (0) Hvit (5)

ut en linje du har tegnet tidligere. PSET og PRESET er vesentlig deler av LINE-setningen og har ingen ting å gjøre med instruksjonene som tenner og slukker punkter.

Det er ikke alltid nødvendig å angi startpunktet på LINE. Uten startpunkt, vil linjen starte ved det siste punktet. (Dersom LINE-setningen ikke er brukt tidligere i programmet, vil det siste punktet bli valgt til å være 128,96, dvs. sentrum av skjermen). Føy nok en linje til eksemplet.

```
40 LINE - (130,180),PSET
```

En linje blir nå trukket fra det siste endepunktet (245,10) til et punkt langt nede på skjermen (130,180). For å tegne en firkant eller rektangel kan du bruke fire linjer, men det finnes en utvidelse av LINE-kommandoen som tar hånd om dette. Bruk EDIT for å legge til B i linje 30, som nå skal se slik ut:

```
30 PCLS : LINE (10,180)- (245,10),PSET,B
```

I stedenfor en diagonal linje, får du nå et rektangel. For å tegne et rektangel må du spesifisere posisjonen til to motsatte hjørner og føye til B i LINE-setningen. Gå tilbake til EDITOR og føy til F på slutten av linje 30.

```
30 PCLS : LINE (10,180)- (245,10),PSET,BF
```

F'en som er lagt til betyr fyll rektangelet med forgrunnsfargen. En slik fleksibel kommando vil vi nå bruke til å tegne et hus.

Kjør programmet bit for bit, slik at du kan se hvordan det hele bygger seg opp. Først vil vi angi oppløsning og tegne veggen.

```
10 PMODE 3,1: SCREEN 1,0: PCLS
20 LINE (60,48)- (200,144),PSET,B
260 GOTO 260
```

Så føyer vi til taket.

```
40 LINE (60,48)- (130,20),PSET
50 LINE- (200,48),PSET
```

og en garasje med en dør.

```
70 LINE (200,144)- (255,94),PSET,B
90 LINE (210,144)- (245,104),PSET,BF
```

Vi kan gjøre bruk av den samme teknikken for å plassere en dør i huset.

```
100 LINE (160,144)- (188,105),PSET,BF
```

For å tegne vinduer trenger vi et rektangel med to linjer i kryss.

```
110 LINE (85,132)- (135,108),PSET,B
120 LINE (110,108)- (110,132),PSET
130 LINE (85,120)- (135,120),PSET
```

På vinduene i andre etasje bruker vi samme fremgangsmåte.

```
140 LINE (90,84)- (125,64),PSET,B
150 LINE (90,74)- (125,74),PSET
160 LINE (110,84)- (110,64),PSET
170 LINE (155,64)- (175,84),PSET,B
180 LINE (165,84)- (165,64),PSET
190 LINE (155,74)- (175,74),PSET
```

COLOR

COLOR-setningen brukes til å endre standardsettingen av forgrunns- og bakgrunnsfarger i høyoppløsningsgrafikk-modus.

COLOR *forgrunn, bakgrunn*

Både *forgrunn* og *bakgrunn* er tall fra 0 til 8 som angir fargekoden. Begge farger må finnes i fargeskalaen for den aktuelle modus.

PCLS

PCLS -setningen er høyoppløsningsversjonen av CLS , og den brukes for å klargjøre skjermen med en viss bakgrunnsfarge i høyoppløsningsmodus.

PCLS *f*

f er fargekoden til bakgrunnsfargen som ønskes. Den må være en av fargene i den aktuelle arbeidsmodus. Er fargen ikke tilgjengelig eller *f* er utelatt, vil standard bakgrunnsfarge bli brukt.

Se rammen for CLS når det gjelder fargekodene.

PSET

Høyoppløsningsversjonen av SET-setningen.

PSET *(x,y,f)*

slår på punktet *(x,y)* med fargen *f*. *x* må være i området 0 til 255 og *y* i området 0 til 191. *f* er en fargekode mellom 0 og 8 og må være en av fargene i fargeskalaen.

PRESET

Høyoppløsningsversjonen til RESET-setningen.

PRESET *(x,y)*

slår av punktet *(x,y)*. Det betyr at punktet settes tilbake til bakgrunnsfargen. *x* må være mellom 0 og 255 og *y* mellom 0 og 191.

For å gjøre ferdig huset, lager vi en pipe.

```
200 LINE (150,40)- (160,15),PSET,BF
```

Dette lille programmet viser hvor raskt vi kan tegne et bilde kun ved å gjøre bruk av en setningstype. Det forutsetter selvfølgelig at du vet hvor du skal tegne linjene. Den enkleste måten å finne disse punktene er å kopiere det grafiske rutenettet i Tillegg B, og så lage en skisse av figuren på det og lese av punktene.

Fargelegging

PAINT-setningen gir deg muligheten til å fargelegge en hvilken som helst form med farge. Du må fortelle hvor fargeleggingen skal begynne, og hvilken farge du vil bruke. Du må også angi fargen på kanten hvor fargeleggingen skal stoppe.

PAINT *(x,y), a, b*

hvor *x,y* er koordinatene til startpunktet og *a* og *b* er fargekoder for fargeleggingen og kanten. Føy følgende programlinje til huseksempelet:

```
30 PAINT (90,90),2,4
```

Dette betyr start med punkt (90,90) og mal med gul farge (farge 2) til du møter en rød (farge 4) kant. Kjør programmet for å se hva det gjør. Fjern så linje 30 og skriv den inn igjen som linje 195 og kjør programmet på nytt.

```
195 PAINT (90,90),2,4
```

Legg nå merke til hvordan den stopper ved vinduskantene som ikke var der tidligere. Fargelegg garasjen på samme måte:

```
80 PAINT (210,140),2,4
```

Så fargelegger vi taket. Dersom du utelater å angi en farge eller en kant i PAINT-setningen vil den aktuelle forgrunnsfargen bli valgt for begge.

```
60 PAINT (130,25)
```

Vi avslutter så med å legge farger på himmelen.

```
210 LINE (0,64)- (60,64),PSET
220 LINE (200,64)- (255,64),PSET
230 PAINT (0,54),3,4
```

Stedet ser litt lysere og vennligere ut nå. Kanskje vil du fortsette med forbedringer, så føy gjerne til en sti og et gjerde. Du kan også ta deg litt tid til å trene på å tegne dine egne former, male dem og se hva som skjer.

LINE

LINE-setningen brukes for å tegne linjer og rektangler i høyoppløsningsgrafikk-modusene.

LINE *(x1,y1) – (x2,y2),a,b*

x1,y1 er koordinatene til linjens startpunkt. *x2,y2* er koordinatene til linjens endepunkt.

a er enten PSET eller PRESET. Dersom PSET brukes, vil linjen bli tegnet i den aktuelle forgrunnsfargen. Dersom PRESET brukes, vil linjen bli tegnet med bakgrunnsfargen.

b er en frivillig parameter. Dersom den brukes er den enten B eller BF. Er B angitt, vil et rektangel bli tegnet i stedenfor en linje, og det øvre hjørnet til rektangelet vil være x1,y1 og det nedre høyre hjørnet x2,y2. Er BF angitt, vil rektangelet bli tegnet og fylt med den aktuelle forgrunnsfargen.

```
10 PMODE 4,1: SCREEN 1,1: PCLS 5: COLOR 0,5
20 FOR I = 1 TO 1000
30 X = X + L*SIN (R): Y = Y + L*COS (R)
40 IF X<-128 OR X>128 THEN 90
50 IF Y<-96 OR Y>95 THEN 90
60 LINE – (X + 128,Y + 96),PSET
70 R1 = R1+60: R = R1/57.29578: L = L+0.5
80 NEXT I
90 GOTO 90
```

PAINT

PAINT-setningen brukes i høyoppløsningsgrafikk-modus for å fylle en form med en spesifisert farge.

PAINT *(x,y),f,g*

x,y er koordinatene til punktet, hvor fargeleggingen skal starte.

f er fargekoden til fargen som skal brukes til fargeleggingen. Fargekoden må være mellom 0 og 8 og være en av de tilgjengelige for arbeidsmodusen. Er den utelatt, vil den aktuelle forgrunnsfargen bli brukt.

g er fargekoden til grensen, hvor fargeleggingen skal stoppe. Den må også være en fargekode mellom 0 og 8.Vær oppmerksom på at

fargeleggingen vil fortsette over en grense som er av en annen farge enn *g*. Er *g* utelatt, vil den aktuelle forgrunnsfargen bli brukt.
Se CIRCLE hvor det står et eksempel på bruken.

Sirkler

Vi har linjer, firkanter og rektangler, og nå vil vi lage sirkler. CIRCLE-setningen kan tegne sirkler, ellipser og buer.

CIRCLE *(x,y), radius, farge, hb-forhold, start, slutt*

x,y angir sentrum av sirkelen. *Radius* er sirkelens radius målt i skjermpunkter. *Fargen* er en av de tilgjengelige fargene i den modus du arbeider i (dersom utelatt blir forgrunnsfargen valgt). De andre parameterne er for å tegne ellipser og buer og dem skal vi behandle senere. La oss først lage sirkler.

```
10 FOR P = 0 TO 4: PMODE P,1
20 SCREEN 1,1: PCLS
30 FOR R = 120 TO 10 STEP - 10
40 CIRCLE (128,96),R: NEXT R
50 FOR D = 1 TO 500: NEXT D,P
```

Dette programmet vil tegne sirkler med sentrum midt på skjermen. Sirkler er vanskelige å tegne og for å få en nøyaktig sirkel, må du sannsynligvis arbeide med PMODE 4.

Legg merke til at dersom en sirkel går utenfor skjermen er ikke det noe problem. Førsøker du å tegne en linje til et punkt som ikke er på skjermen, vil den ikke bli tegnet og spesielt ikke i de høyere oppløsningsmodusene. Forsøk å legge inn

```
42 LINE (128,96)- (300,40),PSET
```

og se på resultatet. PAINT-kommandoen kan også brukes for å fylle sirkler:

```
45 PAINT (127,96)
```

fyller inn blinken. Ved å bruke *hb-forhold*-parameteren, kan du endre sirkelen til en ellipse. *Hb-forholdet* betyr høyde-breddeforhold. Bredden i sirkelkommandoen er alltid den samme, to ganger radius. Høyden kan varieres ved hb-forholdet. Hvis det er større enn 1 vil sirkelen bli høyere enn den er bred. En verdi mindre enn 1 vil klemme sammen sirkelen i den andre retningen og gjøre den bredere enn den er høy. Bredden langs X-aksen (horisontal) er alltid den samme, og det er bare høyden på Y-aksen

CIRCLE

CIRCLE-setningen tegner sirkler, ellipser og buer. Den kan bare brukes i høyoppløsningsgrafikk-modus.

CIRCLE *(x,y), radius, farge, hb-forhold, start, slutt*

x	er x-koordinaten til sentrum i sirkelen (mellom 0 og 255).
y	er y-koordinaten til sentrum i sirkelen (mellom 0 og 191).
radius	er sirkelens radius målt i skjermpunkter.
farge	er en fargekode (mellom 0 og 8).Den må finnes i den tilgjengelige fargeskalaen.Dersom fargekoden er utelatt, blir forgrunnsfargen brukt.
hb-forhold	er høyde-bredde forholdet (mellom 0 og 255). Brukes til å tegne ellipser. Dersom hb-forhold er utelatt, vil 1 bli brukt.
start	er starten på en sirkelbue (fra 0 til 1). 0-posisjonen representerer klokken 3. Dersom parameteren er utelatt, blir 0 brukt.
slutt	er slutten på buen (fra 0 til 1).Tegningen følger klokken fra start. 0.5-posisjonen representerer klokken 9. Dersom parameteren er utelatt, vil 1 bli brukt.

```
10 PMODE 3,1: SCREEN 1,0: PCLS
20 CIRCLE (180,156),28,3: PAINT (180,156),3,3
30 CIRCLE (110,156),28,3: PAINT (110,156),3,3
40 CIRCLE (144,80),68,4,1,0,0.5
50 LINE (212,80) - (76,80),PSET: LINE - (48,32),PSET
60 PAINT (144,82): CIRCLE (144,80),70,4,.8,.79,1
70 LINE (160,80) - (160,28),PSET: PAINT (210,75)
80 GOTO 80
```

(vertikal) som endrer seg. Når hb-forholdet er 0, er sirkelen en horisontal linje. Når hb-forhold blir stort, vil den nærme seg til en vertikal linje (i virkeligheten et langt, tynt rektangel). Den høyeste verdien som er tillatt er 255. Gjør endringer i linjene 30 og 40 i vårt aktuelle eksempel, slik:

```
30 FOR H = 0.5 TO 3 STEP 0.5
40 CIRCLE (128,96),40,,H: NEXT H
```

Legg merke til de ekstra kommaene i linje 40, de er der fordi vi har utelatt *farge*-parameteren.

Den siste utvidelsen på CIRCLE-kommandoen er muligheten for å tegne buer (deler av en sirkel). For å bruke denne muligheten, må du spesifisere *start* og *slutt* på buen. Både start- og sluttverdiene må være et tall mellom 0 og 1. Startpunktet på sirkelen er tilsvarende klokken 3 posisjonen på en klokke. Tegningen skjer så med klokken fra start. F.eks. en start ved 0.25 og en slutt ved 0.75 ville tegne fra klokken 6 til klokken 12, altså den venstre halvdelen av en sirkel. Start ved 0.5 og slutt ved 1.0 for å tegne den øvre halvdelen av en sirkel. Programmet nedenfor bruker buer for å tegne et mønster.

```
10 PMODE 4,1: SCREEN 1,1: COLOR 0,5: PCLS
20 FOR R = 15 TO 60 STEP 5
30 CIRCLE (128,96+R),R,,1,.5,1
40 CIRCLE (128,96-R),R,,1,0,.5
50 CIRCLE (128-R,96),R,,1,.75,.25
60 CIRCLE (128+R,96),R,,1,.25,.75
70 FOR D = 1 TO 500: NEXT D
80 NEXT R
90 GOTO 90
```

Bla i sidene

En måte å føre inn bevegelse i tegningene er å plassere litt ulike bilder på hver side og så "bla" raskt gjennom sidene. Husk at du angir nummeret på siden ved hjelp av PCLEAR-kommandoen og den andre parameteren til PMODE bestemmer siden du skriver på. Selvfølgelig må du huske oppløsningen som du arbeider i. I PMODE 3 og PMODE 4 krever hvert skjermbilde 4 sider, slik at det bare gir mening å bla mellom side 1 og 5. I PMODE 1 og PMODE 2 hvor hvert skjermbilde trenger 2 sider, vil du kunne bla mellom 1,3,5 og 7. Det følgende eksempelet viser hvordan dette gjøres. Forsøk programmet med alle PMODE-verdiene.

```
10 PCLEAR 8: PCLS
20 INPUT"MODE"; M: ON M GOTO 40,40,50,50
30 S = 1: GOTO 60
40 S = 2: GOTO 60
50 S = 4
60 FOR P = 1 TO 8 STEP S: PMODE M,P: PCLS
70 LINE (128,0)-(128, (P-1)*15),PSET
80 SCREEN 1,1: FOR I = 1 TO 1000: NEXT I, P
90 FOR P = 1 TO 8 STEP S: GOSUB 150: NEXT P
100 IF M>2 THEN D = 4: S1 = 3 ELSE D = 7: S1 = S
```

PCOPY

PCOPY er en høyoppløsningsgrafikk-setning, som brukes for å kopiere innholdet av en grafisk side (page) til en annen grafisk side.

PCOPY *kilde* TO *bestemmelsessted*

Kilde og *bestemmelsessted* må være tall fra 1 til 8 og må henvise til sidene som tidligere ble reservert med PCLEAR-kommandoen. Lagerplassen som trengs for å lagre et skjermbilde, er forskjellig for de ulike modusene. Dette må man ta hensyn til når man bruker PCOPY.

PCOPY 3 TO 5

```
110 FOR P = D TO 1 STEP -S1: GOSUB 150: NEXT P
120 GOTO 90
150 PMODE M,P: SCREEN 1,1
160 FOR T = 1 TO 20: NEXT T: RETURN
```

Linjene 60 og 70 tegner figuren som endrer seg på forskjellige sider. All denne tegningen skjer uten å bli vist, ettersom ingen SCREEN-kommandoer er blitt angitt foreløpig. Resten av programmet viser sidene etter hverandre idet det blar først framover og så bakover for å gi inntrykk av bevegelse. Du vil se at jo flere sider du bruker, jo mykere bevegelse får du.

En annen måte å konstruere bilder på, er å gjøre bruk av PCOPY-kommandoen.

PCOPY *kildeside* TO *bestemmelsesside*

Du kan kopiere innholdet av en side til en annen side, under forutsetning av at siden tidligere er blitt reservert med PCLEAR. PCOPY kan også brukes for å legge duplikater på en PMODE 3 eller PMODE 4 side. Det følgende programmet viser hvordan PCOPY brukes på denne måten. Legg merke til hvor forsiktig du må være med plasseringen av figuren.

```
10 PCLEAR 8: PMODE 3,4: PCLS
20 LINE (100,20)-(140,40),PSET,BF
30 CIRCLE (50,25),20
40 CIRCLE (200,50),20
50 FOR D = 3 TO 1 STEP -1
60 PCOPY 4 TO D: NEXT D
70 FOR P = 4 TO 1 STEP -1: PMODE 3,P
80 SCREEN 1,1: FOR I = 1 TO 1000: NEXT I,P
90 GOTO 90
```

I PMODE 3 og 4 består skjermen av fire sider. Side 1 er den øvre fjerdedelen av skjermen, side 2 den neste fjerdedelen osv. Ved å kopiere innholdet av side 4 over på side 1 har du laget en duplikat i den øvre fjerdedelen av skjermen fra den nedre fjerdedelen. For PMODE 1 og PMODE 2 kan den samme effekten oppnås, men denne gangen vil skjermen være todelt og ikke firedelt.

Grafikk behandles videre i kapittel 10.

9. Lyd

Legg på lyd

Datamaskinprogram med og uten grafikk, kan ofte gjøres enda mer interessante ved å legge på lyd. Vi har allerede gjort bruk av SOUND-kommandoen, spesielt i eksemplet i kapittel 7. Du kan selv utstyre programmet med lyd på en enklere måte. Datamaskinen bruker en kassettspiller for å lagre program og den kan også kjøre kassettbånd på kommando. Kommandoene MOTOR ON og MOTOR OFF styrer motoren. AUDIO ON og AUDIO OFF-kommandoene kopler til og fra kassettutgangen på fjernsynets høyttaler. Dette betyr at ved å legge disse setningene inn i programmet, kan du f.eks ha bakgrunnsmusikk til grafikken. Mer seriøst kan du gjøre bruk av ett kassettbånd som inneholder et undervisningsprogram med instruksjoner og spørsmål. Det neste eksemplet viser hvor enkelt dette er. Hvis du ikke har en passende kassett, så bruk en av programkassettene dine. De merkelige lydene som du hører forteller deg hvordan datamaskiner "snakker" med hverandre!

```
10 CLS : PRINT @ 135,"TRYKK BLANKTASTEN"
20 PRINT @ 197,"FOR START OG STOPP"
25 PRINT @ 229,"AV KASSETTSPILLEREN"
30 A$ = INKEY$: IF A$ <> "∇" THEN 30
40 IF F = 0 THEN MOTOR ON: AUDIO ON: F = 1 ELSE MOTOR OFF:
   AUDIO OFF: F = 0
50 GOTO 30
```

Spol kassettbåndene til begynnelsen, trykk på PLAY-knappen og kjør så programmet. Når du trykker ned mellomslagstasten, stopper du avspillingen av kassetten.

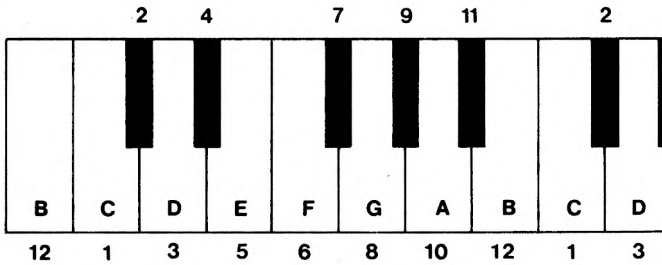
Du kan bruke denne metoden til å lage et undervisningsprogrammer med spørsmål og svar. Dine "tegnefilmer" kan du likeledes utstyre med musikk og lydeffekter.

Spill i vei!

Du kan også få datamaskinen til å spille melodier. PLAY-kommandoen omformer innholdet av en streng til lyd.

PLAY *musikkstreng*

En *musikkstreng* kan være en strengkonstant, strenguttrykk eller en strengvariabel. Det er her ikke snakk om en hvilken som helst gammel streng, men en *musikkstreng* som består av *note*, *oktav*, *notelengde*, *tempo og pause*. *Note* er selvfølgelig den musikknoten som du vil spille. Den enkleste måten å gjøre dette på er å taste inn en bokstav som representerer en av de vanlige engelske musikknotene A,B,C,D,E,F,G. Den norske noten H heter altså B på engelsk. Du må sørge for å bruke B, da Basic ikke vil oppfatte H som note. For å angi en halvtone høyere bruker du # eller +. En halvtone lavere angis med -.Legg merke til at datamaskinen ikke vil gjenkjenne B# eller C- etter som de ikke finnes i 12-tone skalaen. En annen måte å legge notene inn på er å bruke tallet som angir notens posisjon i 12-tone skalaen.



Notene og deres tilsvarende tall er merket på klaviaturet ovenfor.
PLAY-kommandoen kan brukes som en direkte kommando, noe som er nyttig for å kontrollere musikkstrengen før den legges inn i et program. Vi vil starte med å trene på skalaen.

PLAY"CDEFGABCCBAGFEDC" (Skalaen i C)
PLAY"GABCDEF#GGF#EDCBAG" (Skalaen i G)

Skalaen i C ovenfor er nesten riktig, men den som går i G er et eneste kaos. Dette skyldes at skalaene beveger seg inn i en ny *oktav* og vi må fortelle datamaskinen dette. For å velge oktav bruker vi O etterfulgt av et tall mellom 1 og 5. O3 som inneholder midlere C blir automatisk satt når datamaskinen slås på. Denne verdien for oktaven brukes inntil den gis en ny verdi. Derfor er det alltid bedre å spesifisere oktaven du ønsker å bruke allerede i begynnelsen. La oss forsøke skalaen igjen.

PLAY"O3CDEFGABO4CCO3BAGFEDC"
PLAY"O3GABO4CDEF#GG#EDCO3BAG"

Du kan også spille C-skalaen ved å bruke tall istedet for bokstaver:

PLAY"O3;1;3;5;6;8;10;12;O4;1;1;O3;12;10;8;6;5;3;1"

AUDIO

AUDIO-setning knytter forbindelsen mellom lydsignalene fra kassettspilleren til fjernsynsapparatets høyttaler. AUDIO ON styrer kassettspillerens signaler til fjernsynsapparatet. AUDIO OFF bryter forbindelsen.

MOTOR

MOTOR-setningen kontrollerer kassettspillerens motor. MOTOR ON starter motoren og MOTOR OFF stopper motoren.

Play-knappen på kassettspilleren må være trykket ned for at kommandoen skal fungere.

Legg merke til skilletegnet (;) som brukes i strengen. Du kan bruke semikolon over alt, men sammen med tall er det nødvendig for å unngå forvirring.

Ettersom musikkstrengen fortsatt er en streng, kan den behandles med alle de vanlige strengoperasjonene. Det følgende eksempelet spiller skalaen i C over hele området til PLAY-kommandoen.

```
10 A$ = "CDEFGAB": FOR I = 1 TO 5
20 B$ = "O"+STR$ (I)+A$
30 PRINT B$: PLAY B$: NEXT I
```

Ved å bruke den samme teknikken og tall i stedenfor bokstaver, kan vi spille hele den kromatiske skalaen til PLAY-kommandoen.

```
10 FOR I = 1 TO 5: A$ = "O"+STR$ (I)+";"
20 FOR J = 1 TO 12: PLAY A$+STR$ (J): NEXT J,I
```

I de fleste melodier er notene sjelden av lik lengde, slik at vi trenger å angi varigheten til hver note. Dette gjøres ved notelengde-parameteren (L) i musikkstrengen. Bokstaven L blir fulgt av et tall mellom 1 og 255. Etter som størrelsen på tallet øker, minker lengden på tonen. Således er L1 en helnote, L2 en halvnote, L4 en kvartnote osv. Det er mulig å ha 1/255 note, men det er ikke mange komponister som gjør bruk av den. Punktum angir at du skal øke lengden på noten med halvparten av dens normale verdi. For å oppnå den effekten i PLAY-kommandoen, legger du til et punktum (eller så mange som du vil) etter tallet i L-parameteren.

L4. = 1/4+1/8 = en 3/8 note

Vi kan nå nok til å spille en enkel melodi:

```
5 CLEAR 500
10 A$ = "O2L4GG; L2GDL4BB; L2BGL4GB;
   O3L2DDL4CO2B; L1AL4AB;
   O3L2CCO2L4BA; L2BGL4GB;
   L2ADL4F#A; L1G;"
20 B$ = A$+A$: PLAY B$
```

Skilletegnene (;) brukes her for å angi taktstrekene og de er strengt tatt ikke nødvendige. Du skulle kunne kjenne igjen melodien *Clementine*, men den spilles altfor sakte. *Tempo*-parameteren tar hånd om dette ved at man angir bokstaven T etterfulgt av et tall mellom 1 og 255. Jo høyere tall, dess raskere blir melodien spilt. Forsøk med å endre linje 20 slik at den blir:

```
20 B$ = A$+A$: PLAY"T6"+B$
```

Gjør eksperimenter med å endre T6 og finn et tempo som du synes passer til melodien.

I de fleste melodier må vi kunne lage pauser mellom fraser og også å kunne variere lydstyrken i visse passasjer. *Pause*-parameteren er bokstaven P etterfulgt av et tall. Den følger samme mønsteret som notelengdeparameteren (L), bortsett fra at du ikke kan bruke punktum etter tallet. For å legge inn en pause med lengde tilsvarende en L4. note, må vi bruke P4P8. *Volum*-parameteren tillater oss å variere lydstyrken ved å angi bokstaven V etterfulgt av et tall mellom 0 og 31. Etterhvert som vi øker tallet, spilles stykket høyere. Eksemplet nedenfor bruker volumparameteren for å få fram et crescendo.

```
10 A$ = "V10O2L4GG; L1GP4V14L4GGG;
   L1GP4V18L4GGG; L2BL4BBBV22L2BL4BBB;
   V26O3L2DL4DDDL2DL4DDD;
   V30L1GL2.F#L4C#; L2EDCO2A;
   L1GL2AL4.DL8A; L2B"
20 PLAY"T5"+A$
```

Ofte inneholder musikkstykker visse passasjer som blir gjentatt forskjellige steder i stykket. Istedetfor å taste inn passasjen flere ganger, er det vanligvis bedre å legge den inn i en egen strengvariabel. Bokstaven X etterfulgt av en strengvariabel tillater at slike delstrenger kan opptre som en del av en normal sekvens av musikk-kommandoer. X må etterfølges av navnet på strengvariabelen og et semikolon som i:

```
10 X$ = "O3L2GBO4C; DL4CO3BAG;"
20 Y$ = "L2ADD; L1.A"
30 Z$ = "L2ADD; L1.G"
```

PLAY

PLAY-setningen brukes for å lage en musikk-sekvens. Argumentene er enten et strenguttrykk, en strengkonstant eller en strengvariabel. Setningens format er:

PLAY *musikkstreng*

hvor *musikkstreng* er en parameterliste sammensatt av elementene:

note	en bokstav fra 'A' til 'G' eller et siffer mellom 1 og 12. Norsk 'H' må skrives som 'B', da dette er vanlig i engelsktalende land.
oktav	'O' etterfulgt av et tall mellom 1 og 5. Standardverdien er O3. Standardverdien blir satt av datamaskinen når den slås på.
notelengde	'L' etterfulgt av et tall mellom 1 og 255. Standardverdien er L4.
tempo	'T' etterfulgt av et tall mellom 1 og 255. Standardverdien er T2.
volum	'V' etterfulgt av et tall mellom 1 og 31. Standardverdien er V15.
pauselengde	'P' etterfulgt av et tall mellom 1 og 255.
utførelse av delstrenger	'X' etterfulgt av strengvariabel og et semikolon.

En note med kryss foran, angis ved '+' eller '#'. En note med b foran, angis ved '–'. Begge deler må være bak tonen.

Notelengde-parameteren kan modifiseres ved tillegg av et punktum (.) etter tallet. (Eks. L2.) angir på denne måten en punktert note.

Oktav, volum, tempo og notelengde kan alle modifiseres ved å gjøre bruk av et av de følgende suffiksene:

+	adder en til den aktuelle verdien
-	subtraher en fra den aktuelle verdien
>	multipliser den aktuelle verdien med to
<	divider den aktuelle verdien med to

```
10 X$ = "O3L4EF#L4.EL8AAG#ABL4O+C#O-B"
20 A$ = "XX$;O4C#O-AF#O+DC#O-BL2AXX$;
   O+C#DEL8DO-BL<AG#L<AL4.BL8O+C#L4
   DO-BL4.O+C#L8DL<EC#L4.EL8EEEEEL1
   EL4.EL8DC#EDO-BL<AG#L<A"
30 PLAY"T2V20"+A$
```

```
40 A$ = "XX$; XY$; XX$; XZ$; L2BGG; O4CO3L4
   BAGF#; XY$; XX$; XZ$;"
50 PLAY"T8"+A$
```

Vi kunne ha brukt en delstreng i vårt *Clementine* eksempel, ved å forandre linje 20 til:

```
20 PLAY"T6XA$; XA$;"
```

Semikolon (;) *må* etterfølge dollartegnet ($). Delstrenger og bruken av tall som noter (i stedenfor bokstaver) er de eneste tilfeller hvor semikolon er nødvendig.

Det finnes nok en mulighet som kan brukes sammen med volum (V), oktav (O), tempo (T) og notelengde (L)-parameterne. I stedenfor et tall som følger etter bokstaven, kan du bruke et av de følgende suffiksene (tegn som følger etter):

+ Adderer en til den aktuelle verdien
- Subtraherer en fra den aktuelle verdien
> Multipliserer den aktuelle verdien med to
< Dividerer den aktuelle verdien med to

Skalaeksemplet kan nå skrives om igjen og det vi gjør bruk av denne nye, ekstra muligheten:

```
10 PLAY"O1C": FOR I = 1 TO 4: PLAY"DEFGABO+C": NEXT I
```

Melodier finner du ved å bruke noter skrevet for enkle instrumenter som fløyte og trompet.

Selvom du ikke vil spille melodier på datamaskinen, bør du ikke overse PLAY-kommandoen. Spill-entusiaster kan bruke den for å lage effekter. Forsøk et av eksemplene i dette kapittelet med tempo-parameteren satt til T255.

Det siste eksempelet gjør bruk av all den moderne teknologi som PLAY-kommandoen gir oss på en 400 år gammel sang.

```
10 A$ = "O3L2E; L1GL2AL2.BL4O+C#L2.O-B;
   L1AL2F#L2.DL4EL2F#;L1GL2EL2.EL4DL2E;
   L1F#V10L2DV8L1O-BV6L2O+E; L1GL2AL2.B
   L4O+C#L2O-B; L1AL2F#L2.DL4EL2F#;
   L2.GL4F#L2EV8L2.D#V10L4C#V15L2D#;
   L1.EL1EP1;"
20 B$ = "O4L1.DL2DL4C#O-L2B;L1AL2F#L2.D
   L4EL2F#; L1GL2EL2.EL4DL2E; L1F#L2D
   O-L1BO+L2B; O+L1DL2DL2.DL4C#O-L2B;
   L1AL2F#L2.DL4EL2F#;L2.GV10L4F#L2
   EV6L2.D#L4C#V4L2D#;V15L1.EL2EP1;"
30 PLAY"T10XA$; XB$; XA$; XB$;"
```

10. Mer grafikk

I kapittel 8 viste vi hvordan LINE og CIRCLE-setningene kunne brukes for å produsere regulære former som rektangler, sirkler, ellipser og buer. Selv om disse setningene er svært nyttige, vil det kreve omfattende oppfinnsomhet å konstruere mer detaljerte eller uregulære former ved hjelp av disse setningene. Den enkleste måten å behandle uregulære former på er å tegne dem.

Tegning

Når du tegner på et stykke papir, starter du alltid på et bestemt sted og beveger blyanten et visst stykke. Med DRAW-setningen kan du gjenta denne framgangsmåten på skjermen. Formen er:

DRAW *streng*

hvor *streng* er enten en strengkonstant eller en strengvariabel som inneholder en rekke tegneinstrukser. Framgangsmåten er svært lik den i PLAY-setningen i forrige kapittel.

Når du skal begynne å tegne, beveger du deg til startpunktet.

Mx,y betyr å bevege til koordinatene gitt av x,y. F.eks. vil tegneinstruksjonen M128,96 gi en bevegelse til sentrum på skjermen. Når du flytter til et punkt er det hensiktsmessig å gjøre den bevegelsen som en blank bevegelse. Det betyr å bevege uten å tegne, altså med blyanten løftet fra papiret. Gjør du ikke det, kan det være at du får uønskede linjer på tegningen din. En blank bevegelse angis ved hjelp av bokstaven B. Enhver tegneinstruksjon som følger etter bokstaven B vil være en blank linje. BM128,96 betyr å bevege til midten av skjermen uten å tegne.

Etter å ha bestemt et startpunkt, kan du nå bevege oppover (U = up), nedover (D = down), høyre (R = right) eller venstre (L = left) så mange punkter du vil. Rekkefølgen U20R20D20L20 tegner en linje 20 punkter oppover til å begynne med, så 20 punkter til høyre, 20 punkter nedover og 20 punkter til venstre. Sekvensen tegner altså en firkant. Her er et eksempel:

```
10 PMODE 3,1: PCLS : SCREEN 1,1
20 DRAW"BM120,96; U26; R13; D26; L13"
```

80 GOTO 80

Semikolon i en streng brukes som et skilletegn. Det er ikke absolutt nødvendig, men vi bruker det for å gjøre det enklere å lese strengene. Eksemplet tegner et rektangel nær midten av skjermen.

Bortsett fra vertikale og horisontale linjer kan du også tegne diagonale linjer. Disse gjør bruk av delinstruksjonene E,F,G og H, slik at f.eks. E12 tegner en diagonal linje som er 12 punkter lang og i 45 graders vinkel fra vertikalen. Alle vinkler måles fra vertikalen som følger:

E 45 grader F 135 grader
G 225 grader H 315 grader

Dette gjør det mulig å tegne diagonale linjer i hvilken som helst av de fire retningene. Føy til linjen:

40 DRAW"L6; U6; E6; BR13; F6; D6; L6; BU26; H6; G6"

og vårt aktuelle program og rektangelet blir til en rakett. Datamaskinen husker sin siste posisjon, slik at linje 40 vil fortsette tegningen fra det punktet. Arbeid deg gjennom strengen i linje 40 for å se hvordan det gjøres. Den siste posisjonen som er tegnet er det nedre venstre hjørnet av rektangelet og BR13 betyr å bevege 13 punkter til høyre uten å tegne.

Raketten er blitt tegnet i standard forgrunnsfarge, men vi kan forandre den hvis vi ønsker det ved å bruke bokstaven C etterfulgt av en fargekode fra 0 til 8. Gjør bruk av editor for å forandre linje 40 så den blir slik:

40 DRAW"C7; L6; U6; E6; BR13; F6; D6; L6; BU26; H6; G6"

Vi har dermed fått en tofarget rakett. Tegningen kan fargelegges på samme måte som andre former, men C-parameteren endrer standard forgrunnsfarge slik at du må være forsiktig for å unngå å fargelegge overalt.

Tegningen er litt liten, og vi vil skalere den opp med S-parameteren. Den muliggjør at en tegning eller deler av en tegning kan skaleres opp eller ned i enheter på 1/4. Derved vil S1 redusere tegningen til 1/4 av skalaen, S2 til 2/4 (halvparten av skalaen), S8 til 8/4 (dobbelt skala) osv. Standardverdien for skala er 4/4, altså den originale størrelsen. S kan etterfølges av et tall mellom 1 og 62. Føy til linjen:

15 DRAW"S12"

og raketten er nå tre ganger sin opprinnelige størrelse.

En annen mulighet er vinkel-parameteren A. Den gjør det mulig for oss å rotere hele eller deler av en tegning i det alle linjene etter A vil bli tegnet med forskyvningen angitt av An, hvor n er et tall mellom 0 og 3 som følger:

0 0 grader 1 90 grader
2 180 grader 3 270 grader

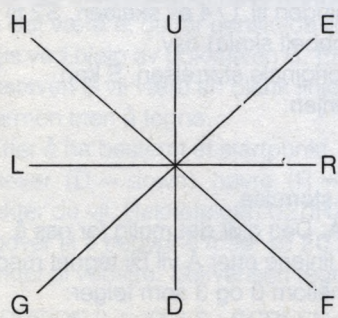
DRAW

DRAW-setningen tegner en linje eller en rekke linjer ifølge instruksjonene i en streng. Den fungerer bare i høyoppløsningsgrafikkmodus.

DRAW *streng*

Streng kan være en strengkonstant eller en strengvariabel og kan inneholde de følgende delinstruksjonene:

M *x,y*	flytt til tegnposisjonen *x,y* på skjermen
U *n*	opp *n* punkter
D *n*	ned *n* punkter
L *n*	venstre *n* punkter
R *n*	høyre *n* punkter
E *n*	*n* punkter langs 45 grader
F *n*	*n* punkter langs 135 grader
G *n*	*n* punkter langs 225 grader
H *n*	*n* punkter langs 315 grader
X	utfør delstreng og returner
C *m*	angir fargekoden *m* til linjen
A *k*	roter neste linje med vinkelen (angle)
	k = 0 0 grader *k* = 1 90 grader
	k = 2 180 grader *k* = 3 270 grader
S *k*	skaler tegningen i enheter på 1/4.
	k varierer fra 1 til 62.
	k = 1 gir kvart skala *k* = 8 gir dobbelt skala
	k = 4 er standardverdi
N	ingen oppdatering av tegneposisjonen (no update)
B	blank (tegn ikke, bare beveg)



Relativ forskyvning kan spesifiseres ved M parameteren på formen:

M *x-forskyvning, y-forskyvning*

Hvor *x-forskyvning* og *y-forskyvning* er tall som angir avstanden fra den aktuelle posisjonen. Begge tallene *må* ha et fortegn, enten pluss (+) eller minus (−).

Et eksempel på bruken av DRAW finnes i under PUT.

Utvid programmet med følgende linjer:

```
18 FOR I = 0 TO 3: DRAW"A"+STR$ (I): PCLS
50 FOR D = 1 TO 100: NEXT D,I
```

Raketten snur seg, og den forandrer også farge. Dette skyldes at datamaskinen ikke bare husker den siste posisjonen, men også den siste verdien satt for C og A. Dette problemet kan løses ved å angi C8 i begynnelsen av strengen i linje 20.

Linje 20 viser at på samme måte som med PLAY-setningen, kan strengene som brukes av DRAW-setningen bli brukt med strengfunksjoner. På en tilsvarende måte som i PLAY-setningen kan du utføre delstrenger med X-instruksjonen etterfulgt av en strengvariabel (XA$) og et semikolon (;).

```
10 PMODE 3,1: SCREEN 1,1: PCLS
20 S$ = "L8E4F4"
30 D$ = "A0; XS$; A1; XS$; A2; XS$; A3; XS$;"
40 DRAW"S24"+D$
50 GOTO 50
```

En trekant er lagret som en delstreng i S$, som så brukes i linje 30 for å bygge opp et mønster. Legg merke til at det eneste stedet semikolon er nødvendig, er etter dollartegnet· ($).

Den siste parameteren er N, som betyr ingen oppdatering av tegneposisjonen. Det medfører å tegne en linje som spesifisert, men ikke bruke slutten av linjen som ny posisjon. NU10L5 vil tegne en linje 10 punkter oppover, så returnere til start på linjen og tegne 5 punkter til høyre (altså en L-form).

```
10 PMODE 3,1: SCREEN 1,1: PCLS
20 DRAW"BM128,96; NU25ND25NR25NL25; NE17NF17NG17NH17"
30 GOTO 30
```

Eksempelet ovenfor vil tegne linjer ut fra sentrum og alltid returnere til sentrum for å starte den neste linjen.

Ofte vil du føye til en tegning i nærheten av den du nettopp har tegnet. Du vet hvor den nye tegningen skal være i relasjon til den gamle, men du ønsker ikke å beregne koordinatene. Dette kan du løse ved hjelp av *relativ forskyvning*, f.eks. 5 punkter til høyre og 10 punkter opp. Bevegelsesparameteren (M) gjør dette enkelt. Alt du trenger å gjøre er å spesifisere avstanden som pluss eller minus i forhold til det nåværende punktet, f.eks. M+5,-10. Husk å gjøre bruk av B for å unngå uønskede linjer.

```
25 DRAW"BM-25,-25, U10R25D10L25"
```

Føy linjen ovenfor til det siste eksempelet og vi får tegnet et rektangel ovenfor den siste tegningen. Den siste posisjonen var 128,96 på grunn av N-parameteren. Vi har nå beveget oss 25 punkter til venstre og 25 punkter opp (husk y = 0 er øverst på skjermen), og tegningen av rektangelet vil starte ved punkt 103,71.

Resultatene fra DRAW-setningen kan kombineres med former fra LINE- og CIRCLE-setningene, men husk at enhver etterfølgende skalering (S), fargelegging (C) eller vinkelforandring (A) vil virke inn bare på DRAW-delen av figuren.

PSET- og PRESET-instruksjonene kan brukes sammen med PAINT-setningen for å få inn ekstra detaljer og farge. Vær imidlertid forsiktig, spesielt med PAINT ettersom endringer i DRAW-delen av figuren kan forårsake farge på gale steder.

Flytt bildet

Etter at du har laget en tegning ved hjelp av LINE, CIRCLE og DRAW, ønsker du kanskje å flytte den rundt på skjermen. Du kunne selvfølgelig gjøre dette ved å blanke den ut og tegne den om igjen hver gang på samme måte som tidligere. Dette ville imidlertid ta ganske lang tid dersom tegningen er komplisert. Derfor har vi to nye setninger som løser dette. Alt du trenger å gjøre er å ta (GET) en kopi av bildet ditt og plassere (PUT) det et annet sted. GET-setningen gir deg muligheten til å kopiere et rektangulært området på skjermen inn i en tabell som senere kan plasseres tilbake på skjermen.

GET *(x1,y1)-(x2,y2), tabellnavn*, G

x1,y1 og *x2,y2* er koordinatene for øvre venstre hjørne og nedre høyre hjørne i det rektangulære området som inneholder bildet som du ønsker å lagre. *Tabellnavnet* er navnet på en tidligere dimensjonert tabell, hvor vi nå skal lagre bildet. Dersom du har glemt hvordan du skal lage tabeller, så ta en repetisjon av stoffet i begynnelsen av kapittel 6. Størrelsen av tabellen må passe til størrelsen av rektangelet på skjermen. Den første tabelldimensjonen er bredden til rektangelet (x2-x1) og den andre er lengden (y2-y1). Den siste

parameteren G er valgfri og avgjør detaljeringsgraden. Det er nødvendig å ta med G-parameteren i PMODE 0,1 og 3 ellers kan horisontalbevegelse ved hjelp av PUT bli unøyaktig.

Vi vil nå bruke tegningen av raketten vår for å vise hvordan dette gjøres. Først må vi beregne størrelsen på tabellen som vi trenger. Tegningen starter ved (120,96), den venstre finnen er 6 punkter tvers over raketten og høyre finne er 13+6, slik at tegningen er 25 punkter bred fra (114,96) til (139,96). Høyden er 26 punkter oppover pluss nesekjeglen som er 5 punkter, slik at høyden totalt er 31. Legg til noen punkter på alle sider og kall det en 30 x 40 rektangel med venstre øvre hjørne i 112,60 og høyre nedre hjørne i 142,100.

```
10 PMODE 3,1: SCREEN 1,1: PCLS : DIM R (29,39)
20 R$ = "C8BM120,96; U26R13D26L13; C7L6U6E6BR13
   F6D6L6BU26H6G6"
30 DRAW R$
40 GET (112,60)- (142,100),R,G
100 GOTO 100
```

Eksempelet ovenfor tegner raketten som tidligere (alt er i en streng nå), og lagrer den i tabellen R. Legg merke til at vi bare trengte en 29 x 39 tabell fordi vi bruker nullelementene i tabellen.

Etter å ha lagret tegningen, må vi nå kunne plassere den tilbake på skjermen. PUT-setningen gjør dette og har en form som ligner GET.

PUT *(x1,y1)- (x2,y2), tabellnavn, handling*

(x1,y1) og (x2,y2) er som tidligere koordinatene til rektangelet, men denne gangen refererer de til området hvor du vil plassere (PUT) tegningen og ikke hvor den kommer fra. *Tabellnavn* er tabellvariabelen som inneholder den lagrede tegningen. *Handlings*-parameteren er kun nødvendig når G-parameteren er blitt brukt i GET-kommandoen. Handlingen må være et av de følgende ordene og avgjør hvordan resultatet blir vist i den nye posisjonen.

PSET	Tenner hvert punkt som er tent i tabellen. Med andre ord viser bildet slik som det er lagret.
PRESET	Slukker hvert punkt som er satt i kildetabellen. Dette vil enten blanke ut bildet eller bytte om fargene, avhengig av de forgrunns- og bakgrunnsfarger som er brukt.
AND	Sammenligner punktene i den originale tegningen med de som allerede finnes på det aktuelle stedet på skjermen. Dersom begge er tent, vil punktet være tent. Hvis den ene eller den andre ikke er tent, vil punktet slukkes. Dette betyr at dersom et bilde blir plassert oppå et annet bilde, vil kun de punktene som faller sammen bli vist.

GET

GET-setningen kan kun brukes i høyoppløsningsgrafikk-modus. GET vil kopiere det grafiske innholdet i et spesifisert rektangulært område på skjermen og lagre det i en tabell. Tabellen må være definert på forhånd og ha korrekt størrelse.

GET *(x1,y1) – (x2,y2), tabellnavn, G*

x1,y1 og *x2,y2* er henholdsvis øvre venstre og nedre høyre koordinat i rektanglet på skjermen.

Tabellnavn er navnet på den på forhånd definerte tabellen som lagrer innholdet av rektanglet.

G er en valgfri parameter. Den angir at full grafisk detaljeringsgrad skal lagres.

Se PUT-rammen for eksempel på hvordan GET brukes.

PUT

PUT-setningen brukes for å vise innholdet i en grafisk tabell som er lagret av GET-setningen.

PUT må brukes i samme modus som ble brukt for å danne tabellen, ellers vil resultatene være umulige å forutsi.

PUT *(x1,y1)- (x2,y2), tabellnavn, handling*

x1,y1 er koordinatene til det venstre øvre hjørnet av skjermområdet, og *x2,y2* er det høyre nedre hjørnet. *Tabellnavn* viser til den tidligere definerte tabellen som inneholder de grafiske detaljene. *Handlings-parameteren* er valgfri, men må angis dersom G parameteren ble brukt i GET-setningen:

PSET	slår på punktene på skjermen ifølge kildetabellen
PRESET	slår av alle punkter som er satt i kildetabellen
AND	sammenligner kildetabellen og bestemmelsesstedet. Dersom begge punkter er satt, forblir punktet påslått, ellers blir det slått av.
OR	sammenligner punktene som ovenfor og dersom et av punktene er påslått, vil skjermpunktet være påslått
NOT	snur tilstanden til hvert punkt i bestemmelsesområdet, uten hensyn til kildetabellen.

Det valgte visningsområdet må være av samme størrelse som tabellen ellers blir det rot på skjermen.

```
10 PCLEAR 4: PMODE 3,1: PCLS : SCREEN 1,1: DIM W (30,30)
20 DRAW"BM10,12;S8;R1U3R1D2R2U2R1D3R1D2L1
   D2R1D1L2U3L4D3L2U1R1U2L1U2R1U2BR1BD1D2R2
   U2NL2R2D2L2U2"
30 PAINT (11,13),6,5: GET (0,0)-(30,30),W
40 A$ = INKEY$: IF A$ = ""THEN 40
50 PCLS : FOR C = 0 TO 100 STEP 20
60 FOR A = 0 TO 200 STEP 20
70 PUT (A,C) - (30+A,30+C),W
80 PUT (A,C+30) - (30+A,60+C),W
90 PUT (A,C+60)-(30+A,90+C),W
100 PLAY"T255;ABFGBA": PCLS : NEXT A,C
```

OR — Sammenligner punkter som ovenfor. Dersom enten kilden eller bestemmelsespunktet er satt, vil punktet på skjermen bli tent. Dette har som resultat at en tegning blir overlagret en annen.

NOT — Dette reverserer hvert punkt i visningsområdet og dermed vises bildet mot forgrunnsfargen.

Du må alltid bruke PUT i samme modus som GET for ellers vil det kunne føre til merkelige resultater. Vi kan nå gå tilbake til eksempelet vårt og plassere (PUT) raketten på et annet sted. Føy til disse ekstra linjene:

```
50 Y = 150: FOR X = 10 TO 210 STEP 40
60 PUT (X,Y)-(X+30,Y+40),R,PSET
80 NEXT X
```

Det skal nå finnes en rekke raketter langs bunnen av skjermen. For å få raketten til å bevege seg langs bunnen, føyer du bare til linjen:

```
70 FOR D = 1 TO 200: NEXT D: PCLS
```

Ved å bruke styrespakene i forbindelse med GET- og PUT- setningene, kan du bevege tegningen din som du ønsker.

```
10 PMODE 3,1: SCREEN 1,1: PCLS : DIM S (48,48)
20 DRAW"BM24,12,S8; C4; E2H2D4D8R8H8G8R2NR6F3R6E3"
30 GET (0,0)-(48,48),S
40 A$ = INKEY$: IF A$ = ""THEN 40
50 PCLS : A = JOYSTK (0)*3.25: B = JOYSTK (1)*2.25
60 PUT (A,B)-(A+48,B+48),S: GOTO 50
```

Eksempelet ovenfor tegner en figur i det øvre venstre hjørnet på skjermen. Når du trykker ned hvilken som helst tast, blir skjerm-bildet visket ut og figuren kan beveges omkring ved hjelp av venstre styrespak.

Hvis du ønsker å spare plass kan du dimensjonere tabellen mindre for bruk i GET og PUT. Størrelsen av tabellen beregnes av rektangelets areal i høyoppløsningspunkter. Arealet ((x2 - x1) * (y2 - y1))divideres med D, hvor D=160 i PMODE0, D=80 i PMODE 1 og 2, D=40 i PMODE 3 og 4. Rund resultatet opp til nærmeste heltall og pluss 1 til.

Med dette er vi ferdig med å behandle de grafiske mulighetene som finnes. Eksemplene som vi har gått gjennom, er nødvendigvis begrensede og gir ikke på noen måte et fullstendig bilde av hva som er mulig om man har tålmodighet og evner. Planlegging gjør det mye enklere å tegne figurer og former enn om man direkte starter arbeidet på skjermen. Bruk derfor de grafiske arbeidsarkene før du setter igang på skjermen.

11. Siste hånd på verket

Ekstra om utskrift

Selv om din styring med hvordan resultatene vises på skjermen er ganske omfattende ved hjelp av PRINT og PRINT@-setningene, finnes det enda en mulighet. Med PRINT USING-setningen kan du spesifisere nøyaktig hvordan hver linje skal skrives ut. Setningen er spesielt nyttig for å produsere tabeller, skjemaer og regnskapsmessige oppstillinger.

PRINT USING *format; utdataliste*

Format er en strengkonstant eller en strengvariabel som inneholder instruksjoner om hvordan *utdatalisten* skal skrives. *Utdatalisten* er den vanlige listen som består av konstanter og variable på samme måte som i PRINT-setninger. *Format* inneholder instruksjonene om det enkelte felt. Disse består av tegn som forteller datamaskinen hvor mange skriveposisjoner som skal brukes for å skrive et tall eller en streng.

 # spesifikasjonen

Dette tegnet brukes for å angi posisjonen til hvert siffer i et tall.

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

Setningen ovenfor vil skrive ut innholdet av A som 3 siffer før desimalpunktumet og 2 etter. Er det mer enn 2 siffer etter desimalpunktumet, vil tallet bli avrundet. Ubrukte posisjoner til venstre for desimalpunktet vil bli vist som blanke. Er tallet for stort til å få plass på den avsatte plassen, vil datamaskinen gjøre sitt beste for å skrive tallet og indikere overskridelsen med %-tegn foran tallet.

```
PRINT USING"###.##";13.4695
▽13.47
PRINT USING"###.##";1492.878
%1492.88
PRINT USING"###.##";146
146.00
PRINT USING"###";18.76
▽19
```

* spesifikasjonen

De fleste som steller med penger liker ikke tanken på å skrive tall med innledende blanke, og spesielt ikke på sjekker. Dette kan man løse ved å bruke * spesifikasjonen. Setter du to stjerner i begynnelsen av det numeriske feltet, vil ubrukte posisjoner bli fylt med stjerner.

```
PRINT USING"**###.##";1.492
****1.49
```

+ spesifikasjonen

Når +tegnet plasseres i begynnelsen av et numerisk felt, medfører det at fortegnet til tallet blir skrevet.

```
PRINT USING"+###.##";14.7
▽+14.70
PRINT USING"+**###.##";-7.4
****-7.40
```

Dersom plusstegnet plasseres etter det numeriske feltet, blir tegnet skrevet etter tallet.

```
PRINT USING"###.##+";27.86
▽27.86+
PRINT USING"###.##+";-1.6
▽▽1.60-
```

Dersom et minustegn plasseres etter et tall, blir alle negative tall skrevet med et minustegn etter seg, og alle positive tall blir etterfulgt av en blank.

```
PRINT USING"**###.##-";-12.418
***12.42-
PRINT USING"###.##-";47.25
▽47.25▽
```

↑↑↑↑ spesifikasjonen

Denne spesifikasjonen gjør at tall skrives ut på eksponensiell form. De fire pilene oppover må følge etter tallfeltet.

```
PRINT USING"##.###↑↑↑↑"; 123456
1.2346=+05
```

! spesifikasjonen

Denne spesifikasjonen brukes i forbindelse med strenger. Den vil sørge for at bare det første strengtegnet som forekommer skrives.

```
PRINT USING"!";"KREDITT"
K
```

% spesifikasjonen

For å kunne skrive ut strenger er det nødvendig å spesifisere lengden på feltet som de skal forekomme i. Dette gjøres ved to % tegn som er adskilt ved et antall blanke. Lengden av feltet vil da være antall blanke pluss to. Dersom strengen er lengre enn det tilgjengelige feltet, vil bare de første *n* tegnene bli skrevet, hvor *n* er lengden av feltet.

```
PRINT USING" %▽▽▽▽▽%";"DEBET"
DEBET ▽▽
PRINT USING" %▽ %";"BALANSE"
BAL
```

$ spesifikasjonen

Dollartegnet ($) er et pengesymbol. Er det plassert foran en numerisk spesifikasjon, vil det medføre at dollartegnet kommer med i utskriften.

```
PRINT USING"$###.##";2.87
$▽▽2.87
```

Brukes to dollartegn, blir dollartegnet skrevet rett foran tallet.

```
PRINT USING"$$###:##";2.87
▽▽▽$2.87
```

Brukt i forbindelse med de to stjernene, vil dollartegnet sørge for følgende resultat:

```
PRINT USING"**$#.##";14.9
*$14.90
```

Blanke og andre tegn som forekommer i formatstrengen, vil også komme med i utskriften:

```
PRINT USING"GJENNOMSNITT▽▽##.##▽▽▽TOTAL▽▽###:##";
3.4,40.8
GJENNOMSNITT▽▽▽3.40▽▽▽TOTAL▽▽▽40.80
```

PRINT USING

PRINT USING-setningen gir bedre styring med utskrift av resultater på skjerm og skriver.

PRINT USING *format; utdataliste*

Format er en strengkonstant eller en strengvariabel som inneholder feltspesifikasjoner som angir hvordan utdatalisten skal skrives. Utdatalisten er en liste av strengvariable eller numeriske variable som er adskilt med komma. Det finnes følgende feltspesifikasjoner:

Tegn	*Handling*	*Eksempel*	*Resultat*
#	Formaterer tall, høyrejustert	"####";147.2	147
.	Desimalpunktum i fast posisjon	"##.##";34.678	34.68
,	Viser komma til venstre for hvert tredje siffer.	"######.#"; 123456	123,456
**	Fyller ledende blanke med stjerner	"**###.###"; 1.47	****1.470
$	Setter inn dollar-tegn foran tall	"$####.##"; 12.689	$▽▽12.69
**$	Flytende dollartegn	"**$####.##"; 12.689	****$ 12.69
+	I første posisjon forårsaker at tegnet blir skrevet foran, i siste siste posisjon at det blir skrevet etter tallet	"##.##+"; -12.689	12.69-
↑↑↑↑	Skriv på eksponen-siell form	"##.##↑↑↑↑"; 12.689	1.27E+01
!	Skriv bare det første strengtegnet	"!";"KREDIT'	K
%blanke%	Strengefelt. Lengden av strengfeltet er antall blanke pluss 2	"%▽▽▽▽▽%"; "BALANSE"	BALANSE

Hver feltspesifikasjon kan skilles med et vilkårlig antall blanktegn som vil komme fram som blanke på utskriftslinjen.

```
10 CLS : INPUT"TAST SISTE BALANSE";B:C=0:D=0
20 CLS :TS="▽▽▽▽%▽▽▽%▽▽▽%▽▽▽▽%▽▽▽%▽▽▽▽▽%"
30 L$="▽▽▽######▽▽####.##▽▽####.##+"
40 PRINT USING T$;"DEBET","KREDIT","BALANSE"
50 PRINT USING L$;D,C,B
60 OPEN"I",#-1,"SJEKK"
70 IF EOF (-1)THEN 110
80 INPUT#-1,A:D=0:C=0
90 IF A<0THEN D=ABS (A)ELSE C=A
100 B=B+C-D:PRINT USING L$;D,C,B:GOTO 70
110 CLOSE#-1:END
```

Dersom utdatalisten inneholder flere elementer enn antall felt i formatet, vil formatet bli brukt om igjen fra starten.

```
PRINT USING"###.##▽▽▽";7.84,142.5,.234
▽▽7.84▽▽▽142.50▽▽▽▽▽0.23
```

Så lenge vi gjør bruk av skjermen er selvfølgelig lengden på linjen som produseres ved hjelp av PRINT USING-setningen fortsatt begrenset til 32. Er den lengre enn dette, vil den fortsette på neste linje. På en skriver kan linjelengden være mye lenger (minst 80 tegn på de fleste skrivere). For skrivere gjelder følgende format:

PRINT #-2, USING *format; utdataliste*

Format og *utdataliste* er som tidligere og #-2 betyr at dataene sendes ut på skriverkanalen og ikke på skjermen. Ønsker du å få det skrevet ut både på skjermen og skriveren, må du altså bruke to PRINT USING-setninger. Du kan også bruke PRINT #V, hvor V er en variabel med verdien -2 for skriver og 0 (null) for skjerm.

Data på kassett

Så langt har alle programmene våre forutsatt at vi taster inn de data som vi trenger (eller leser dem fra en data-setning), og at utdata har blitt vist på skjermen. Du kan imidlertid bruke kassettspilleren din til å lagre både data og program. De lagrede dataene kan så leses senere. Kassettspilleren er tilkoplet på samme måte som når vi lagret program. Det du må gjøre er å fortelle

Utskrift til skriver

For de av dere som har en skriver knyttet til parallell-I/O-porten finnes det varianter av noen av setningene som kan å styre utskriften til skriveren og ikke til skjermen.

PRINT #-2, *utdataliste*
PRINT #-2, *USING format; utdataliste*

Format og *utdataliste* er de samme som for skriving på skjerm.
POS (-2) vil returnere den aktuelle posisjonen til skrivehodet.
LLIST vil liste et program direkte på skriveren. Bruken er tilsvarende LIST-setningen. Bruk av SHIFT-tasten og 0-tasten i kombinasjon gjør det mulig å skrive små bokstaver på skriveren. Muligheten for små bokstaver kan bare brukes i strenger og REM-setninger. Alle programsetninger og kommandoer må være med store bokstaver. Hvis skriveren skriver uten linjeskift, kan det løses på to måter: Endre bryter (e) på skriveren eller POKE 330,2.

For å få norske bokstaver på skriveren bruker du følgende taster:

Engelsk	*Norsk*	*Tastetrykk*
[	Æ	[SHIFT] - [↓] (SHIFT først)
\	Ø	[SHIFT] - [CLEAR] (samtidig!)
]	Å	[SHIFT] [→] (SHIFT først)

datamaskinen at den arbeider med datafiler. Det gjøres med OPEN-setningen.

OPEN *a*, #-1, *filnavn*

hvor *a* må være enten "O" eller "I". "O" betyr utdata (output), altså at data går fra datamaskinen til kassettbåndet. "I" betyr inndata (input), altså at data kommer fra kassetten og går til datamaskinen.

#-1 forteller datamaskinen at det er kassettspilleren du bruker. *Filnavn* er det navnet som du setter på datafilen (et hvilket som helst navn er lovlig, bare det starter med en bokstav og er på 8 tegn eller mindre).

Neste trinn er å skrive data til kassettbåndet. Det gjøres ved en PRINT-setning på følgende måte:

PRINT #-1, *utdataliste*

Den eneste forskjellen fra de PRINT-setningene vi har brukt fram til nå er #-1. #-1 forteller datamaskinen at den skal skrive *utdatalisten* på kassettbåndet og ikke på skjermen. Når du er ferdig med å skrive data, må du lukke filen med CLOSE-setningen.

CLOSE #-1

For å lese data tilbake fra filen gjør du bruk av de samme trinnene, bortsett fra at denne gangen åpnes filen for INPUT ("I") og i stedenfor PRINT gjør du bruk av:

INPUT #-1, *inndataliste*

CLOSE-setningen er lik for begge tilfeller.

Eksemplet nedenfor viser hvordan dette gjøres. Gjør først klar kassettspilleren og spol magnetbåndet til det stedet på båndet du ønsker at filen skal lagres. Bruk SKIPF. Trykk så PLAY og RECORD-knappene samtidig.

```
10 CLS : PRINT"LAG TELEFONLISTE"
20 OPEN"O", #-1,"TELEFON": PRINT"TAST XXX,XXX FOR AA AVSLUTTE"
30 PRINT @ 128,""; : INPUT"NAVN▽",N$;
40 INPUT"TLF.NR."; T$: IF N$ = "XXX"OR T$ = "XXX"THEN 60
50 PRINT #-1,N$,T$: PRINT @ 128,"": GOTO 30
60 CLOSE #-1: END
```

Når du kjører programmet vil båndspilleren bli slått på og åpne filen. Hver gang du taster inn et navn og et tall, blir det skrevet på filen. PRINT @ 128 setningen i linje 30 sørger for å blanke skjermen. Dette fortsetter til du taster inn XXX,XXX som sørger for at filen blir lukket og programmet avslutter.
Alt vi nå har å gjøre er å lese dataene tilbake igjen. Den viktigste forskjellen mellom utdata og inndata, er at med inndata må du sørge for ikke å lese forbi slutten av filen. Dette gjøres med en ekstra setning som du trenger for inndata. EOF-instruksjonen kontrollerer for å se om du har kommet til slutten av filen.

Spol kassettbåndet tilbake til begynnelsen og trykk bare ned PLAY-knappen.

```
10 CLS : PRINT"LES TELEFONLISTE"
20 OPEN"I", #-1,"TELEFON"
30 PRINT"NAVN","NUMMER"
40 IF EOF (-1)THEN 60
50 INPUT #-1,A$,B$: PRINT A$,B$: GOTO 40
60 CLOSE #-1: END
```

Når du kjører programmet denne gangen, vil magnetbåndet starte og søke etter filen "TELEFON". (Det kan være at du må vente en liten stund dersom filen er på slutten av kassettbåndet). Programmet vil så lese inn navn og telefonnummer og vise dem på skjermen. Legg merke til at du ikke behøver å bruke de samme variabelnavnene som du brukte for å skrive dataene. Du må imidlertid bruke samme typer variable. Når slutten av filen er nådd, blir den lukket og programmet avslutter.

EOF-instruksjonen må være før INPUT #-1-setningen, ellers vil du få en IE feil (forsøk på å lese forbi slutten av filen). Glem aldri å lukke (CLOSE) en fil etter som dette kan forårsake problemer spesielt når du skriver til en fil.

Maskinspråk

Neste trinn etter Basic-programmering er maskinspråk. Det er datamaskinens eget språk. Fram til nå har du snakket med datamaskinen ved hjelp av et tolkeprogram (interpreter) som forstår Basic.

Fordelen med maskinspråk er at instruksjonene i maskinspråk utføres mye raskere, bruker mindre lagerplass og kan gjøre ting som du ikke får til ved hjelp av Basic.

En god måte å gå fram på er å anskaffe en håndbok om maskinspråk, som spesielt behandler 6800-serien av mikroprosessorer. Se litteraturlisten bak i boken

Datamaskinen har en rekke rutiner som gir deg mulighet til å gjøre bruk av program som er skrevet i maskinspråk. Her følger noen korte beskrivelser.

Ved hjelp av USRn er du i stand til å kalle inntil ti (0-9) program, skrevet i maskinspråk. Formatet er:

USR *n (argument)*

Argument er en streng eller et numerisk uttrykk. Når et USR-kall påtreffes i et program, blir kontrollen overført til adressen som er angitt i DEF USR *n*-setningen. Adressen angir startpunktet for maskinspråkrutinen. DEF USR *n* brukes for å definere adressen til en USR *n*-funksjon. Formen er:

DEF USR *n* = *adresse*

hvor *n* er et tall mellom 0 og 9 og samsvarer med *n* i USR. Adressen må være mellom 0 og 65535 og inneholde startadressen for USR *n*.

CLEAR *s,h*

CLEAR-setningen brukes for å avsette lagerplass til USR-funksjonene. Lagerplass for strengene reserveres ved hjelp av *s* som tidligere. Den høyeste lageradressen som Basic kan bruke angis ved hjelp av *h*. Maskinspråkrutinene har hermed fått reservert plassen fra *h*+1 og oppover.

POKE-kommandoen brukes for å skrive en verdi inn i en bestemt del av lageret.

POKE *adresse, verdi*

Adressen er som ovenfor og *verdien* må være mellom 0 og 255.

VARPTR står for variabelpeker. En peker til en Basic-variabel kan brukes som et argument i en USR-funksjon. Dette gir USR-funksjonen tilgang til innholdet i en tabell.

VARPTR *(variabelnavn)*

Her er *variabelnavn* den Basic-variabelen du ønsker tilgang til. VARPTR brukes som en del av et USR-argument som i:

USR0 (VARPTR (X))

Maskinspråkrutiner kan lagres på kassett og hentes tilbake igjen ved hjelp av CSAVEM og CLOADM.

CSAVEM *navn, start, slutt, startpunkt*
CLOADM *navn, forskyvning*

Navn er betegnelsen på filen på kassettbåndet. *Start* er den laveste adressen til rutinen i det indre lageret. *Slutt* er den høyeste adressen som opptas av rutinen. *Startpunkt* er programmets startpunkt. *Forskyvning* i CLOADM-kommandoen tillater deg å lese inn igjen rutinen i lageret på en adresse gitt av *start + forskyvning*.

Straks den er lest inn kan rutinen utføres ved hjelp av EXEC-kommandoen:

EXEC *adresse*

Adresse er starten på rutinen. Dersom adresse er utelatt, vil datamaskinen bruke *start* fra den siste CLOAD-kommandoen.

Programeksempelet under viser hvordan en kan overføre verdiene til variabler mellom Basic- og maskinkodeprogrammet. Maskinkodeprogrammet leses inn fra DATA-setningene og utfører multiplikasjon av to tall med maksimalverdier på henholdsvis 127 og 255.

```
10 REM MASKINKODE-PROGRAMMERING
20CLEAR 20,3199: START = 3200 : ANTALL =8
40 DEF USR5=START
50 CLS
60 FOR ADRESSE=START TO START+ANTALL-1
70 READ VERDI
80 POKE ADRESSE,VERDI
90 NEXTADRESSE
100 ' MASKINKODEN INNE I LAGERET
110 INPUT "TAST INN TO TALL";T1,T2
120 T3=256*T1+T2
130 T4=USR05 (T3)
140 PRINT T1;"*"T2;"=";T4
150 END
160 'HENT VERDI JSR $8B2D
170 DATA &HBD, &H8B, &H2D
180 'MULTIPLISER MUL
190 DATA &H3D
200 'LEVER VERDI JSR $8C37
210 DATA &HBD, &H8C, &H37
220 'TIL BACIC RTS
230 DATA &H39
```

Tillegg A

ASCII-koder

Tast	*Desimalverdi uten shift-tast*	*Desimalverdi med shift-tast*
[BREAK]	3	3
[CLEAR]	12	92
[ENTER]	13	13
[BLANK]	32	32
!	33	-
"	34	-
#	35	-
$	36	-
%	37	-
&	38	-
'	39	-
(	40	-
)	41	-
*	42	-
+	43	-
,	44	-
-	45	-
.	46	-
/	47	-
0	48	18
1	49	-
2	50	-
3	51	-
4	52	-
5	53	-
6	54	-
7	55	-
8	56	-
9	57	-
:	58	-
;	59	-
<	60	-

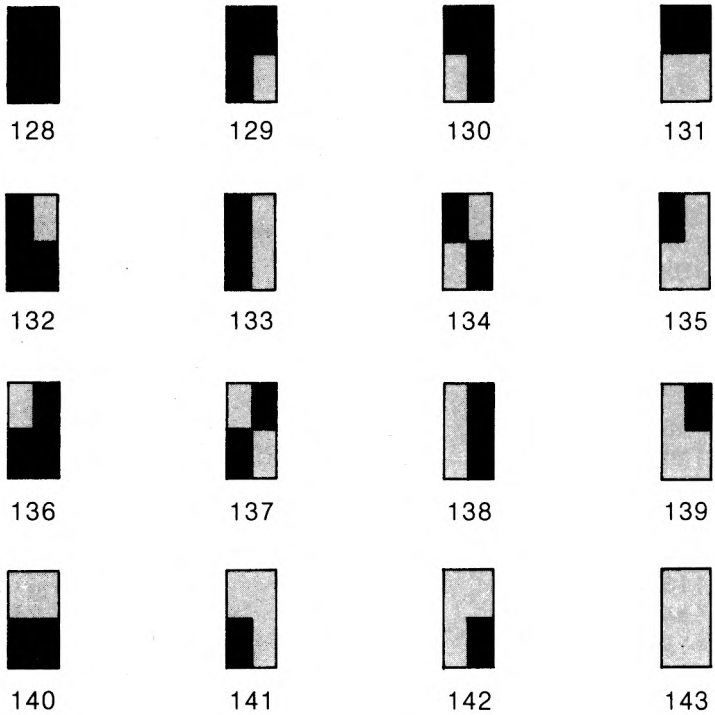
Tast	*Desimalverdi uten shift-tast*	*Desimalverdi med shift-tast*
=	61	-
>	62	-
?	63	-
@	64	19
A	97	65
B	98	66
C	99	67
D	100	68
E	101	69
F	102	70
G	103	71
H	104	72
I	105	73
J	106	74
K	107	75
L	108	76
M	109	77
N	110	78
O	111	79
P	112	80
Q	113	81
R	114	82
S	115	83
T	116	84
U	117	85
V	118	86
W	119	87
X	120	88
Y	121	89
Z	122	90
↑	94	95
↓	10	91
←	8	21
→	9	93

SHIFT- og 0-tasten i kombinasjon gjør at bokstaver som følger etter blir skrevet med små bokstaver på en skriver. SHIFT- og 0-tasten igjen gir normal modus. Følgende "små bokstav"-tegn er tilgjengelig ved hjelp av CHR$-funksjonen:

[CHR$ (123)
/ CHR$ (124)
] CHR$ (125)
↑ CHR$ (126)
← CHR$ (127)

Grafiske tegn

Tegnene fra 128 til 255 er grafiske tegn som ser slik ut:

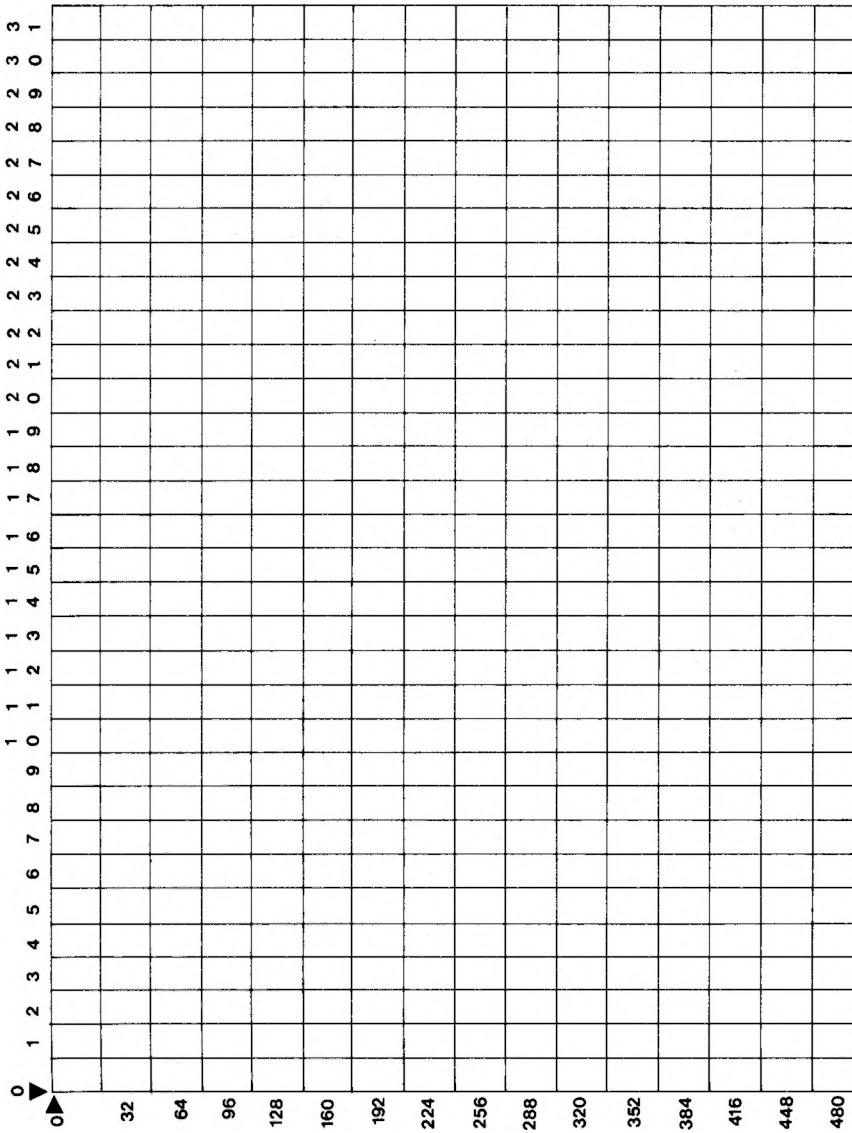


For å få fram tegnene som vist ovenfor bruker du CHR$ med den passende tallkoden. Det grå feltet er grønt på fargefjernsyn.For å få fram andre farger legger du til tallet som svarer til fargen. F.eks. vil PRINT CHR$ (142+112) gi tegn 142 bortsett fra at det grønne området er oransje.

+16 gul	+32 blå	+48 rød
+64 hvit	+80 turkis	+96 fiolett
+112 oransje		

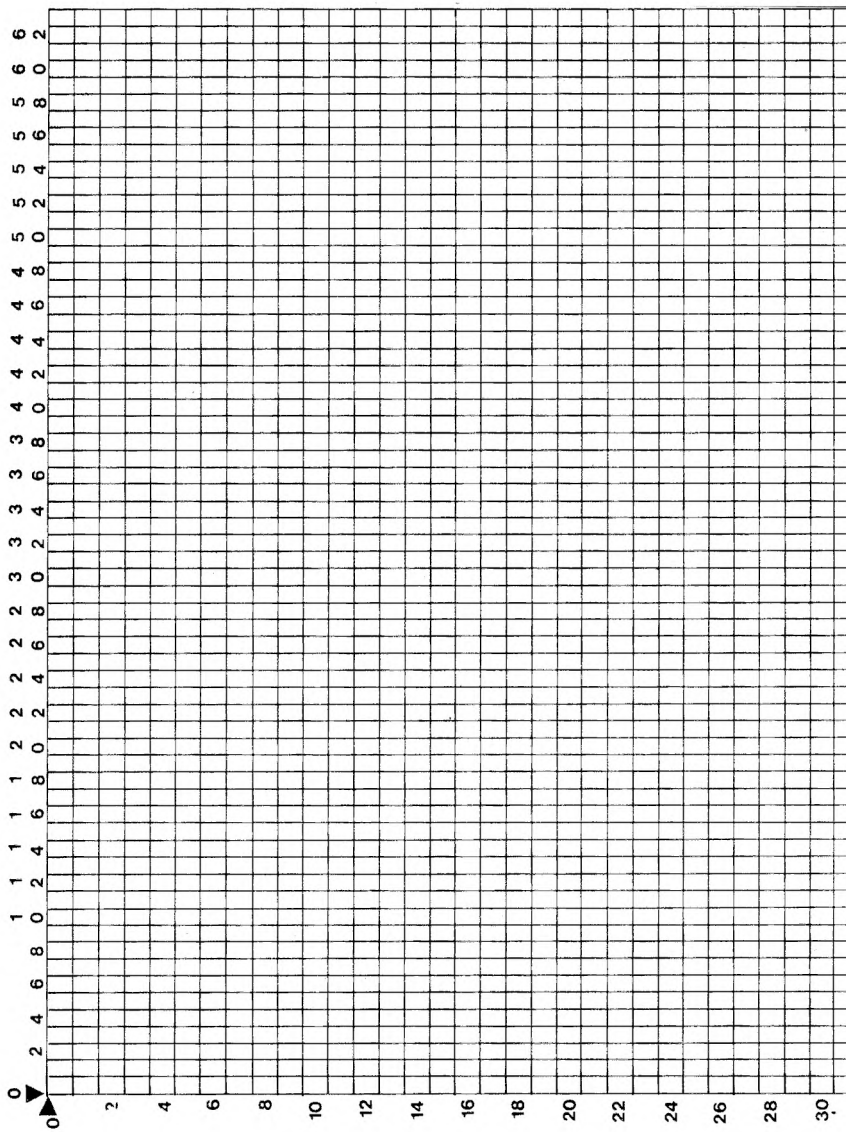
Tillegg B

Rutenett for PRINT@-setningen



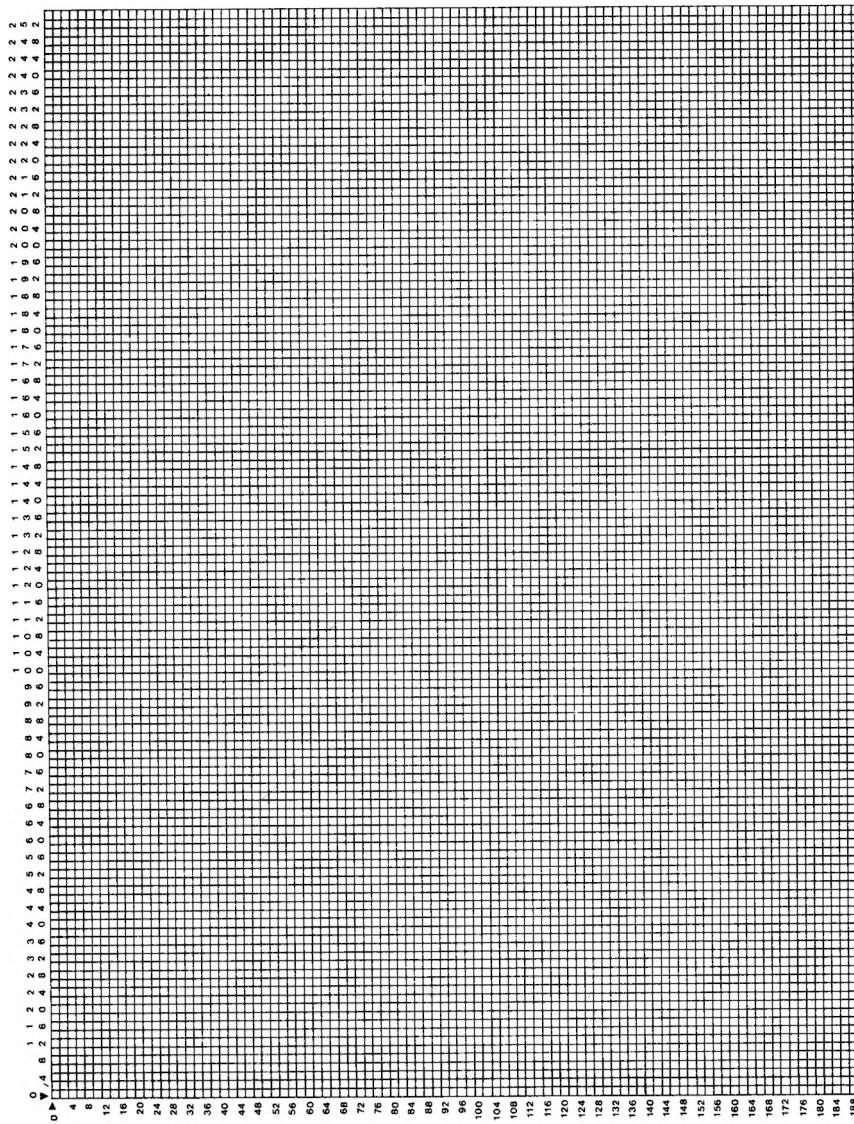
Rutenett for lavoppløsningsgrafikk

Det andre rutenettet er for lavoppløsningsgrafikk på tekstskjermen, som gjør bruk av SET- og RESET-setningene.



Rutenett for høyoppløsningsgrafikk

Det tredje rutenettet viser høyoppløsningsskjermen og kan brukes i forbindelse med alle instrukser som gjelder høyoppløsningsgrafikk.



Tillegg C

Feilkoder

Kode	*Forklaring*
/0	Divisjon med null, ikke mulig.
AO	Forsøk på å åpne en fil som allerede er åpen. Inntreffer vanligvis etter at RESET er blitt trykket for å stoppe et program som bruker filer. Slå maskinen av og på igjen.
BS	Ulovlig indeks. Inntreffer vanligvis fordi verdien til indeksen er større enn tabellen er dimensjonert til.
CN	Kan ikke fortsette. Forsøk på å bruke CONT når man er i slutten av programmet.
DD	Forsøk på redimensjonering av tabell. Tabeller kan bare dimensjoneres en gang i et program.
DN	Feil enhetsnummer (device number). F.eks. ved PRINT # A når A>O eller A<-2.
DS	Kommandofeil. Inntreffer vanligvis dersom du forsøker å ta CLOAD på en datafil.
FC	Ulovlig funksjonskall. Vanligvis en ulovlig parameterverdi eller en feil variabeltype.
FD	Dårlige data på filen. Forårsaket av forsøk på å lese datastreng inn i en strengvariabel hvor man gjør bruk av filer på kassettbånd.
FM	Gal filmodus. Forsøk på INPUT fra en fil som er OPEN for utskrift (O) eller PRINT av data til en fil som er OPEN for inndata (I).
ID	Ulovlig kommando. Forsøk på å bruke en setning som bare kan brukes i program. Eks. INPUT, DEF FN.
IE	Forsøk på å lese forbi filslutt. Bruk IF EOF (-1) for å kontrollere at dette ikke skjer.
IO	Inndata/utdata feil. Kassettspilleren ikke justert skikkelig eller dårlig kassettbånd.
LS	Strengen for lang. Maksimal strenglengde er 255 tegn.
NF	NEXT uten FOR. Inntreffer vanligvis når NEXT-setninger er byttet om i nestede sløyfer.
NO	Filen ikke åpen. Lesing og skriving på en datafil kan kun foregå etter at filen er OPEN.
OD	Slutt på data. En READ-setning har lest alle DATA-setningene.
OM	Lageret fullt. Alt tilgjengelig lagerplass brukes eller er reservert.

OS Slutt på lagerplass for strenger. Bruk CLEAR for å gjøre mer plass tilgjengelig.

OV Overflyt. Tallet er for stort til å kunne bearbeides av datamaskinen (ABS (X)>1E38).

RG RETURN uten GOSUB. Hovedprogrammet har sannsynligvis "falt" gjennom seg selv og inn i et delprogram. Bruk END i hoved-programmet så unngår du dette. Alternativt kan det ha skjedd en forgrening direkte inn i delprogrammet.

SN Syntaksfeil. Forårsakes vanligvis av skrivefeil eller feil skilletegn.

ST Strenguttrykk for komplekst. Del opp uttrykket i mindre deler.

TM Type passer ikke. Forsøk på å tilordne datastreng til numerisk variabel eller motsatt.

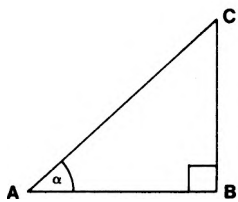
UF Udefinert funksjon. FNA (X) uten DEF FNA (X).

UL Udefinert linje. En forgreningssetning har et setningsnummer som ikke eksisterer.

Tillegg D

Trigonometriske funksjoner

I en rettvinklet trekant ABC blir:
AB kalt for den *tilstøtende* siden til vinkelen *a*.
BC kalt for den *motstående* siden til vinkelen *a*.
AC kalt for *hypotenusen*.



Sinus, cosinus og tangens til vinkelen *a* er definert slik:

$$\text{Sin } \alpha = \frac{\text{motstående side}}{\text{hypotenusen}}$$

$$\text{Cos } \alpha = \frac{\text{tilstøtende side}}{\text{hypotenusen}}$$

$$\text{TAN } \alpha = \frac{\text{motstående side}}{\text{tilstøtende side}}$$

Alle de trigonometriske funksjonene i Basic forutsetter at vinkler angis i *radianer*. En radian er et mål på en vinkel angitt i hva vi kan kalle sirkelenheter. Det er 360 grader eller 2 * π radianer i en sirkel. (π er en gresk bokstav som uttales pi og står for konstanten 3.1415926). Det er enkelt å regne om fra den ene til den andre:

Grader/ (180/π) = radianer
Radianer * (180/π) = grader

Den inverse av en funksjon er det motsatte av å bruke funksjonen. F.eks. er tangens til en vinkel på 1.5 radian:

TAN (1.5) = 14.101419

Den inverse til funksjonen TAN er ATN. Den brukes for å finne vinkelen dersom du allerede kjenner tangens:

```
ATN (14.101419) = 1.5
```

Den inverse til sinus og cosinus-funksjonene er ikke tilgjengelig standardfunksjon i Basic, men kan finnes ved hjelp av ATN-funksjonen og følgende formler:

```
Invers sinus = ATN (X/SQR (-X*X+1))
Invers cosinus =  (ATN (X)SQR (-X*X+1))+1.5708
```

Tillegg E

Bruk av skrivere

Dragon 32 fungerer godt med mange av skriverne som finnes på markedet.Tilkoplingen skjer over kontakten merket P.I/O. på Dragon. Dette er en 20-veis kontakt som gir signaler for standard "centronics"-type skrivere. Pinne-tilkoplingen er som følger:

Øvre rekke		*Nedre rekke*		
Pinne 1	Print strobe	Pinne 2	+5 Volt	Til
Pinne 3	Data bit 0	Pinne 4	+5 Volt	høyre
Pinne 5	Data bit 1	Pinne 6	0 Volt	
Pinne 7	Data bit 2	Pinne 8	0 Volt	
Pinne 9	Data bit 3	Pinne 10	0 Volt	
Pinne 11	Data bit 4	Pinne 12	0 Volt	
Pinne 13	Data bit 5	Pinne 14	0 Volt	
Pinne 15	Data bit 6	Pinne 16	0 Volt	
Pinne 17	Data bit 7	Pinne 18	0 Volt	Til
Pinne 19	ACK (negativ)	Pinne 20	BUSY (negativ)	venstre

Følgende skrivere har fungert sammen med Dragon:
Amber 2400, Centronics 351, Epson MX80, OKI microline 80 og 93, Seikosha GP100A og GP250A.

Tillegg F

Lageradressering (memory map)

Desimal adresse	*Innhold*	*Heksadesimal adresse*
0-1023	System bruk	0000-03-FF
255	"Direct page" RAM	00FF
1023	"Extended page" RAM	003F
1024-1535	Tekstskjerm-lager	0400-05FF
	Grafikkskjerm-lager	
1536-3071	Side 1	0600-0BFF
3072-4607	Side 2	0C00-11FF
4608-6143	Side 3	1200-17FF
6144-7679	Side 4	1800-1DFF
7680-9215	Side 5	1E00-23FF
9216-10747	Side 6	2400-29FF
10748-12287	Side 7	2A00-2FFF
12228-13824	Side 8	3000-35FF
13824-32767	Program- og variabellager	3600-7FFF
32768-49151	Basic fortolker	8000-BFFF
49152-65279	Program-modul lager	C000-FEFF
65280-65375	Inn/ut-område	FF00-FF5F
65376-65503	SAM kontroll-bits	FF60-FFDF
65504-65535	CPU-vektorer	FFE0-FFFF

Litteraturliste

Bøker om Dragon 32

George Knight: *Learning to use the DRAGON 32 computer* Gower. ISBN 0-566-03494-8
Don Monro: *Know your DRAGON.* Tiny. ISBN 0-907909-02-7
Ian Sinclair: *The DRAGON 32 and how to make the most of it.* Granada Publishing. ISBN 0-246-12114-9
S. M. Gee: *The DRAGON programmer.* Granada Publishing. ISBN 0-246-12133-5
James, Gee and Ewbank: *The DRAGON 32 book of games.* Granada Publishing. ISBN 0-246-12114-9
Tim Longdell: *35 programs for DRAGON 32.* Century Publishing. ISBN 0-7126-0173-2
Phipps and Toms: *Load and go with your DRAGON.* Phipps Associates. ISBN 0-950-7302-9-7
David Lawrence: *The working DRAGON.* Sunshine Books. ISBN 0-946408-01-7
Colin Carter: *Enter the DRAGON.* Melbourne House. ISBN 0-86161-114-4
Steward and Jones: *Easy Programming for the DRAGON 32* Shiva Publishing. ISBN 0-906812-38-0
Steward and Jones: *Further Programming for the DRAGON 32.* Shiva Publishing. ISBN 0-906812-39-0

Bøker om Assemblerprogrammering

Zaks and Labiak: *Programming the 6809.* Sybex. ISBN 0-89588-078-4
Lance A. Leventhal: *6809 Assembler Language Programming.* Osborne/McGraw Hill. ISBN 0-931988-35-7
Ron Bishop: *Basic Microprocessors and the 6800.* Hayden Books.

Tidsskrifter

Norske

Hobbydata
HiFi & Elektronikk

Svenske

Dator Hobby
Mikrodatorn

Engelskspråklige

Dragon User
The Rainbow
The Color Computer Magazine
Color Computer News
Popular Computing
Your Computer
Practical Computing
Which Micro & Software Review
Personal Computer World

Stikkordregister

ISBN 82-991032-0-7