Commodore 64

Whizz Kid

Geof Wheelwright

Longman

Commodore 64
Whizz Kid

Commodore 64 Whizz Kid

Geof Wheelwright

Longman

Remember

Typing in listings

It's important that you type in all the listings exactly as they are given in this book. Computers are fussy things and if you get even a comma in the wrong place it can make a nonsense of your program.

On the Commodore 64 you type in all the commands letter by letter. There are no built-in keywords. Most of the graphic symbols are shown on the front of the keys and you shouldn't have any problems with these. If you do, check back with your manual which will tell you exactly how to get each shape.

Have fun!

Longman Group Limited
Longman House, Burnt Mill, Harlow,
Essex CM20 2JE, England
and Associated Companies throughout the world.

First published 1984

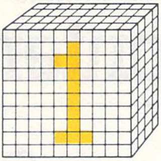
ISBN 0 582 91609 7

Printed in UK by Parkway Illustrated Press, Abingdon

Designed, illustrated and edited by Contract Books, London

The programs listed in this book have been carefully tested, but the publishers cannot be held responsible for problems that might occur in running them.

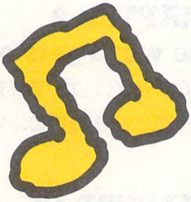
Contents

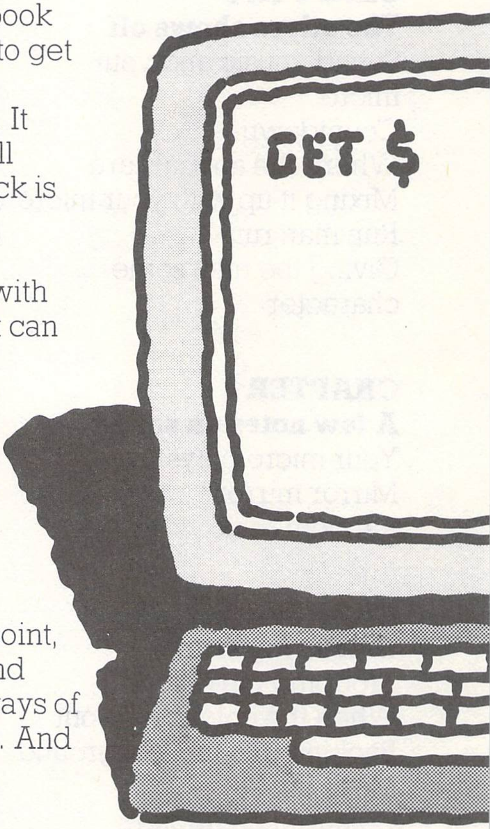


About this book

Playing with your computer should always be fun. If you learn something while you are enjoying yourself, all the better. This book has been written to allow newcomers to get the most from their computers.

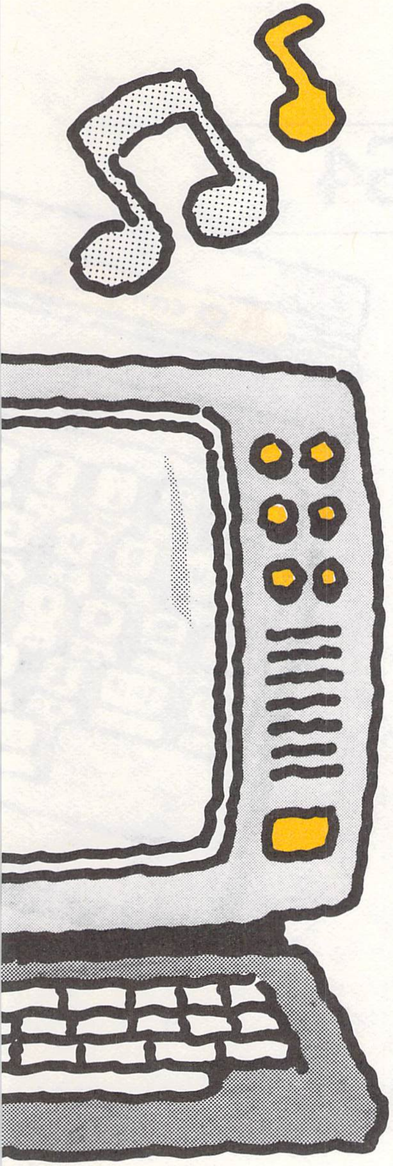
Your micro makes a good playmate. It will never get bored or tired, and it will always do exactly as you tell it. The trick is to make your instructions as clear as possible, so that your micro will know exactly what you expect of it. To start with however, you have to know just what it can do. That's where this book comes in.





Use each new project as a starting point, and go on to invent your own games and programs. You'll soon think of better ways of getting your micro to do various things. And then you'll have a real feeling of achievement.

All the projects on this book should help you to get exciting and colourful results from your micro.



Each project is divided up into different sections so you can find your way around the program easily and discover how it works. After a description of what the program is going to do there will generally be a listing of the program 'Try this'. There may also be a 'Remember' box to remind you of important things to do or to remember when typing in your project.

Try this

Remember

Once you have run the program you'll find a simple explanation of it in 'How it works'. There may also be a 'Did you see' section pointing out other things you ought to notice in the program.

How it works

Did you see?

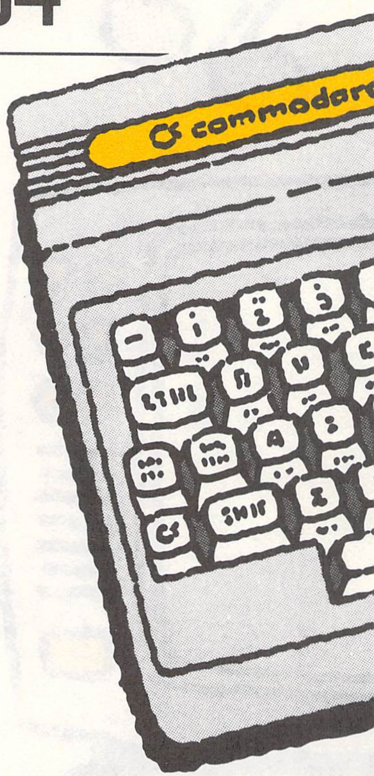
Finally it's 'Your turn', when you can experiment with your own improvements to the program and learn how to make your micro work for you.

Your turn

Now you are all set to begin. I hope that you are going to enjoy these projects as much as I have enjoyed writing them.

2

About your 64



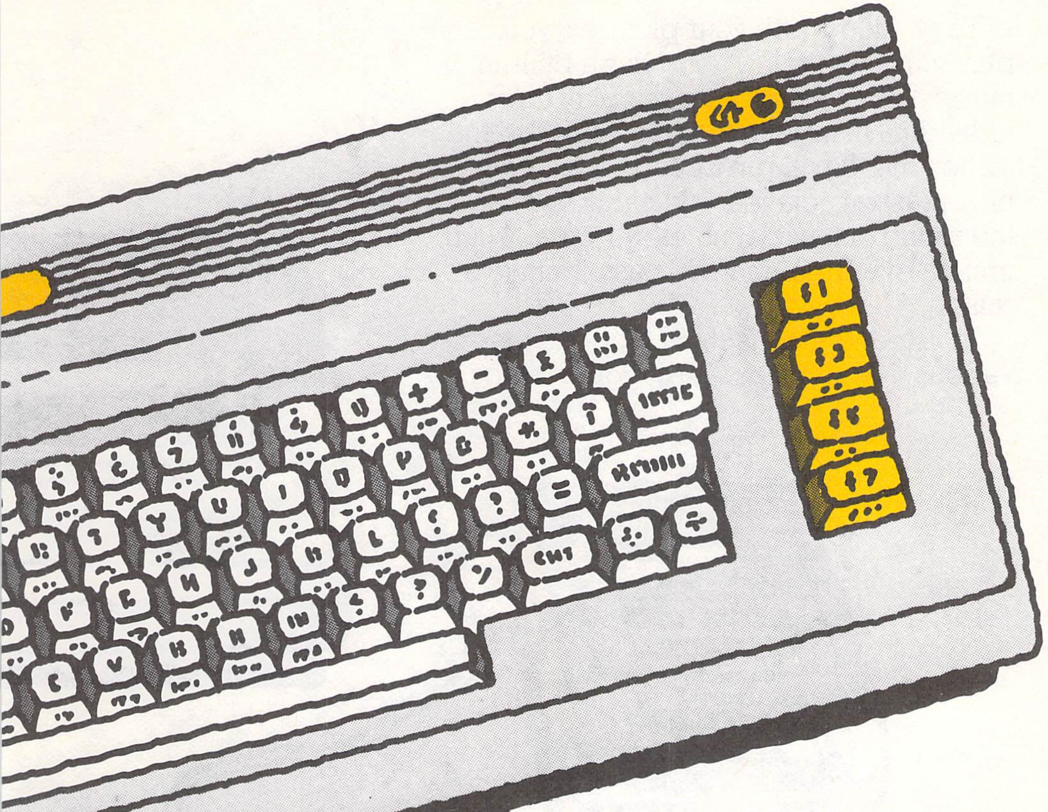
Your Commodore 64 is probably the most powerful computer available in its price range. It has excellent colour graphics and sound facilities, a good keyboard and lots of opportunities for expansion.

Commodore Business Machines, which began as a Canadian company supplying business equipment and furniture, is now one of the world's largest and most successful home computer companies.

The company's first popular computer was the Commodore PET – released in 1978. It had only 8 Kilobytes of memory (compared to 64K in the Commodore 64), and relied on cassette tape as its chief means of remembering programs. The PET had some success as a small business machine, but didn't have much impact on people like you and me – that is, people who want a computer for their homes.

Then came the introduction of the Vic 20 in the early part of this decade. It brought a cheap computer with a good keyboard, sound and graphics into many homes.

The Vic 20 had a very similar Basic programming language to the 64, used much the same kinds of add-on equipment and was about the same size. It's one great problem, however, was the size of its memory (or RAM). Expansion to even a



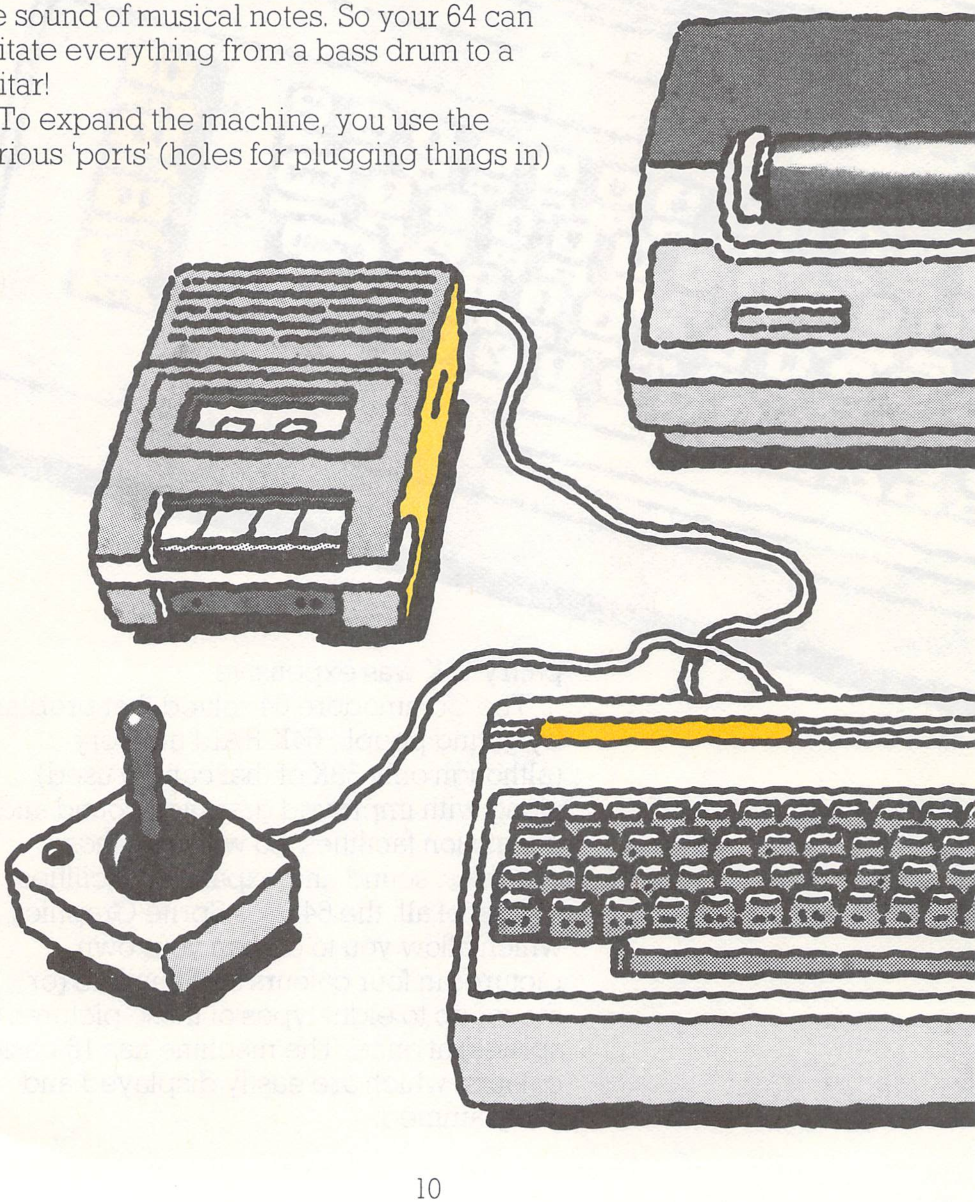
paltry 16K was expensive.

The Commodore 64 solved that problem by giving people 64K RAM memory (although only 39K of that can be used) along with improved graphics, sound and expansion facilities. So what *are* those graphics, sound and expansion facilities?

First of all, the 64 uses Sprite Graphics, which allow you to design your own pictures in four colours and animate (or move) up to eight types of those pictures (or sprites) at once. The machine has 16 basic colours, which are easily displayed and programmed.

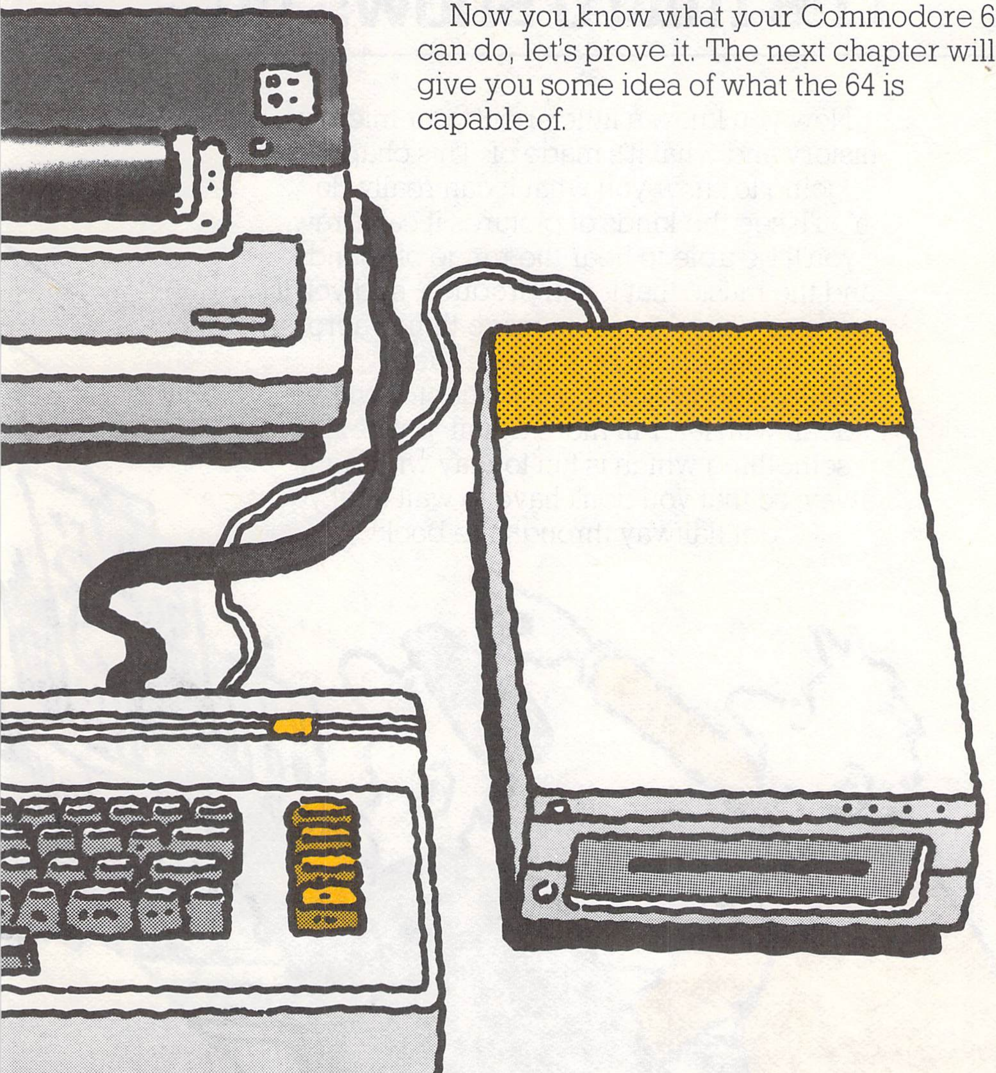
To go along with your pictures, you can play music as well. The 64 has a built-in music maker, which lets you play three individual instruments or 'voices' all at the same time. It will run through a full range of nine musical octaves, and can also change the sound of musical notes. So your 64 can imitate everything from a bass drum to a guitar!

To expand the machine, you use the various 'ports' (holes for plugging things in)



at the back of the 64. When it is connected up properly, you will be able to use disk drives, printers, cassette recorders and amplifiers. You can even take the music from your 64 and play it through the speakers on your home stereo!

Now you know what your Commodore 64 can do, let's prove it. The next chapter will give you some idea of what the 64 is capable of.

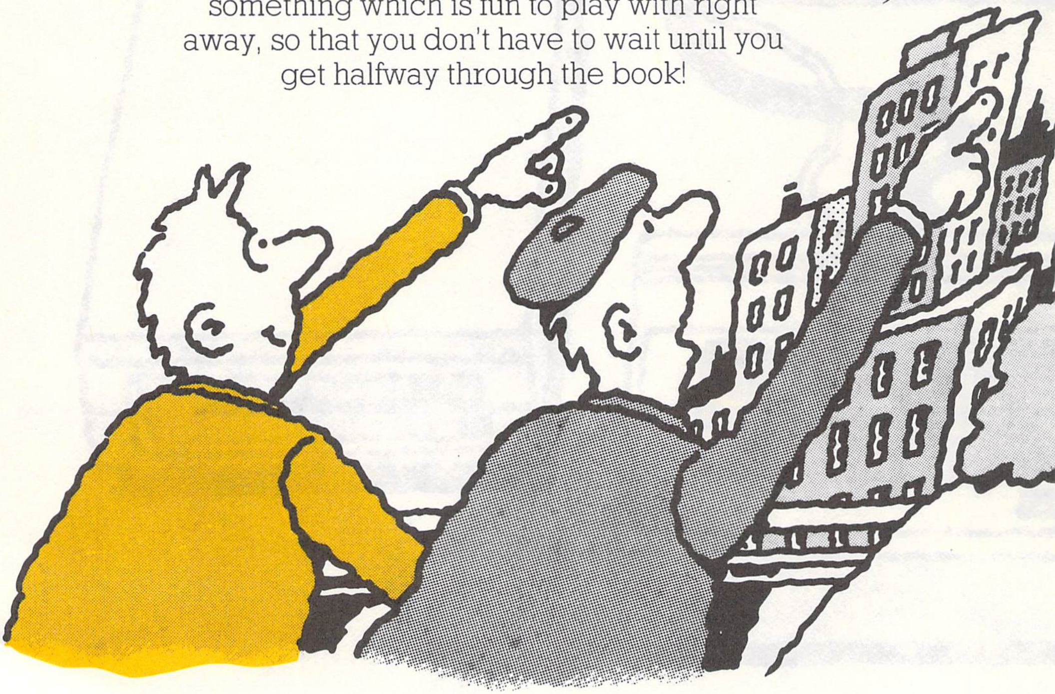


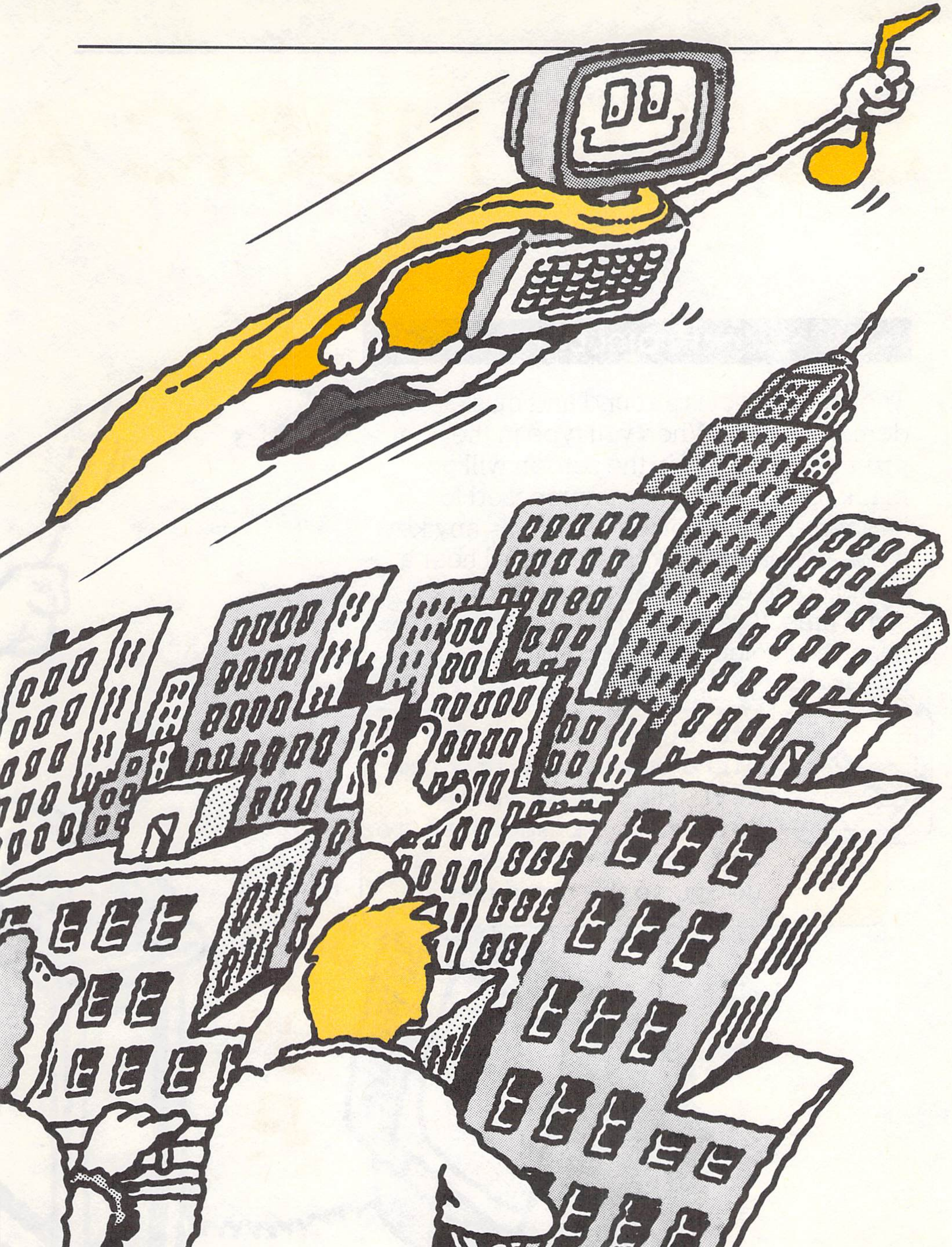
3

The micro shows off

Now you know a little about your micro's history and what it's made of. This chapter is going to show you what it can really do. You'll see the kinds of pictures it can draw, you'll be able to hear the range of sounds and the music that it can produce, and you'll get a preview of how to move things across the screen or 'animate' them.

Each of the projects in this chapter will be dealt with later in more detail, but next is something which is fun to play with right away, so that you don't have to wait until you get halfway through the book!





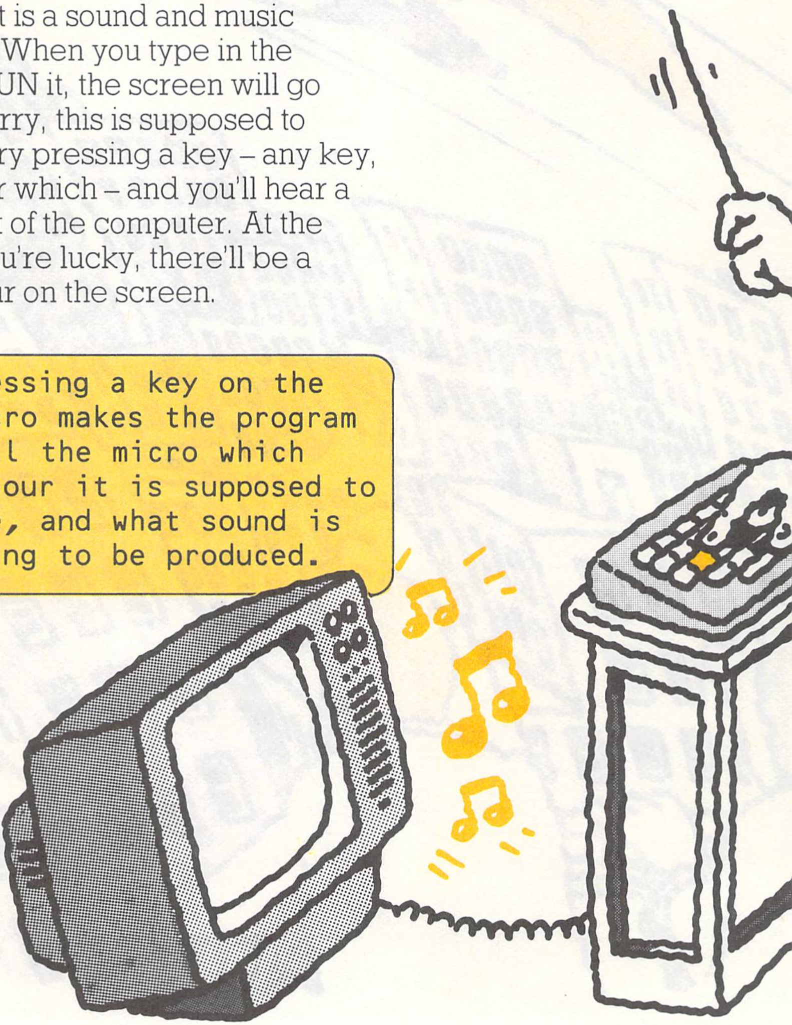
SOUND, MUSIC A

Project

The first project is a sound and music demonstration. When you type in the program and RUN it, the screen will go blank. Don't worry, this is supposed to happen. Then try pressing a key – any key, it doesn't matter which – and you'll hear a sound come out of the computer. At the same time, if you're lucky, there'll be a change of colour on the screen.

Remember

Pressing a key on the micro makes the program tell the micro which colour it is supposed to use, and what sound is going to be produced.



ND YOUR MICRO

```
 5  GET A$:IF A$="" THEN 5
 7  A=ASC(A$)
 8  POKE 5328Ø, (INT(A/12)):POKE
     53281,(INT(A/12))
1Ø  POKE 54296,15
2Ø  POKE 54277,15
3Ø  POKE 54278,27
4Ø  POKE 54273,A*.4:POKE
     54272,A*.5
5Ø  POKE 54276,17
6Ø  FOR T=1 TO 45Ø:NEXT T
7Ø  POKE 54276,Ø:POKE 54277,Ø:POKE
     54278,Ø
8Ø  GOTO 5
```

How it works

You get information from the keyboard using the GET statement. This then allows the program to give the micro different numbers as different keys are pressed.

Your turn

Change line 5 to read: 5 INPUT A$. This means that you must hit the RETURN or ENTER key *after* you have pressed a number or letter key. It will allow you to edit, or change, your number or letter *before* the computer uses it to produce a sound and colour.

Project

Micros, by their very nature, have to be precise and operate to a timetable, just like spaceships and planes. All spaceships, in fact, rely on computers to do many things for them – including the pre-launch countdown before blast-off.

Just like those big computers that control space shuttles, your computer has a clock in it and can count off chunks of time for a countdown. And it's not too difficult to write a program that will allow you to look at that clock and make it work for you. In this project, you type in a program that uses the micro's as a clock to countdown the last minute before take-off.

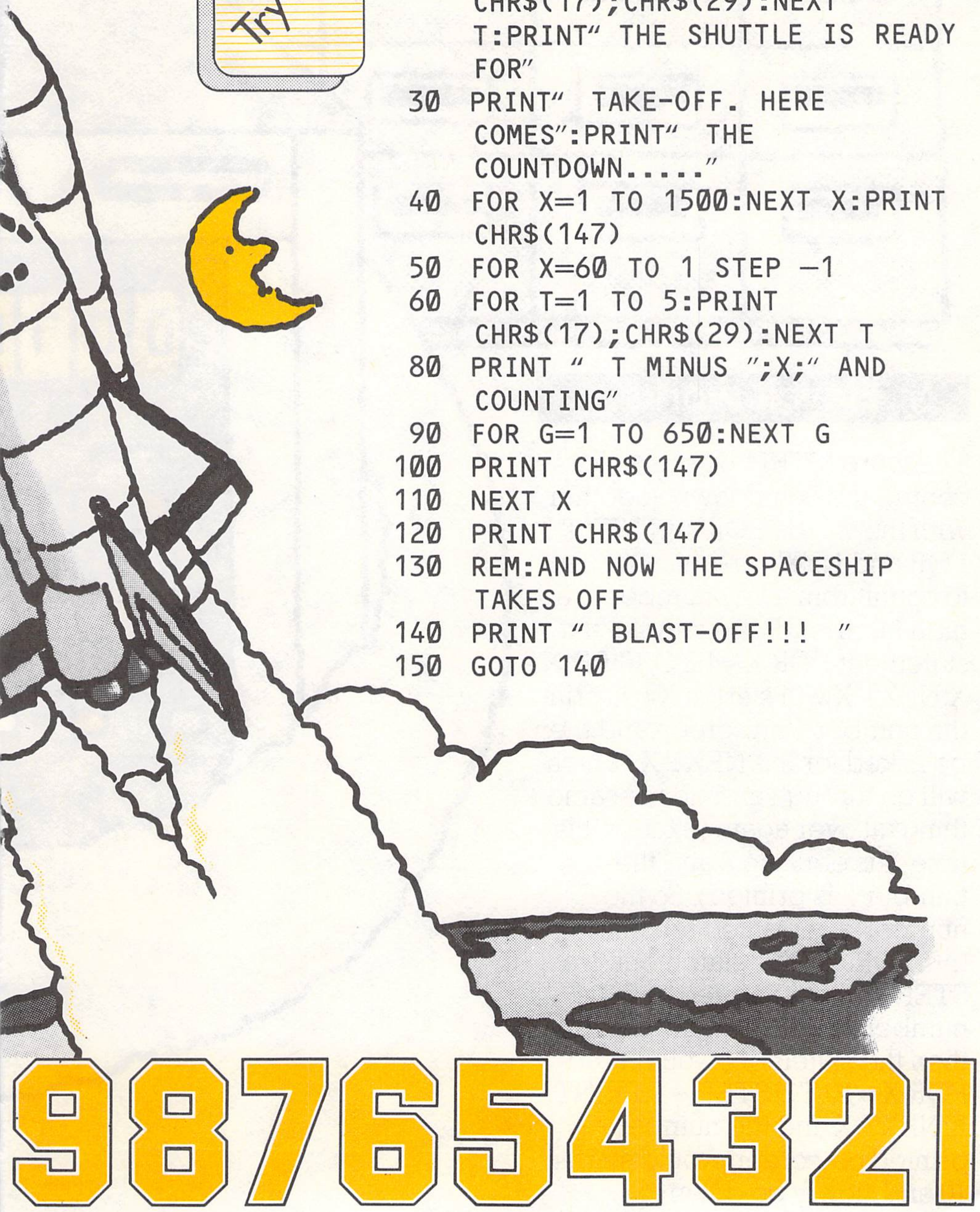
Remember

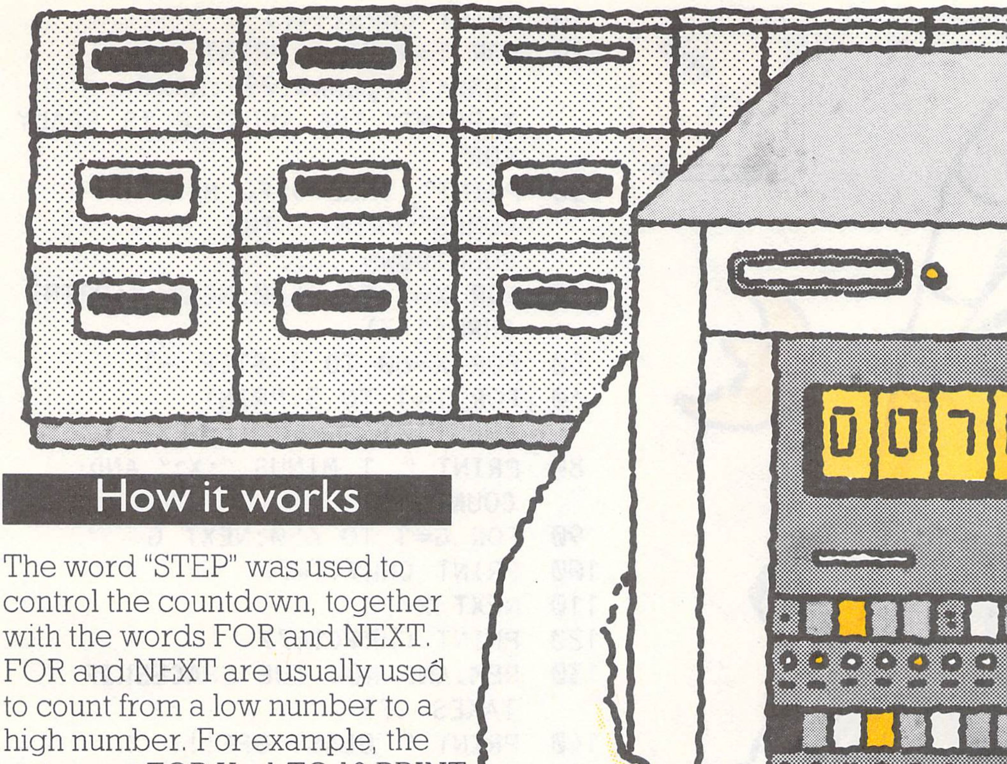
The clock inside a micro is far more accurate than any clock you're likely to use in everyday life. After one minute the computer will have counted out hundreds and hundreds of units of time while you've been counting sixty seconds! So if you were to use the computer clock, the total number of tiny fractions of time would have to be divided to give a number which is small enough to work with.

COUNTDOWN

Try this

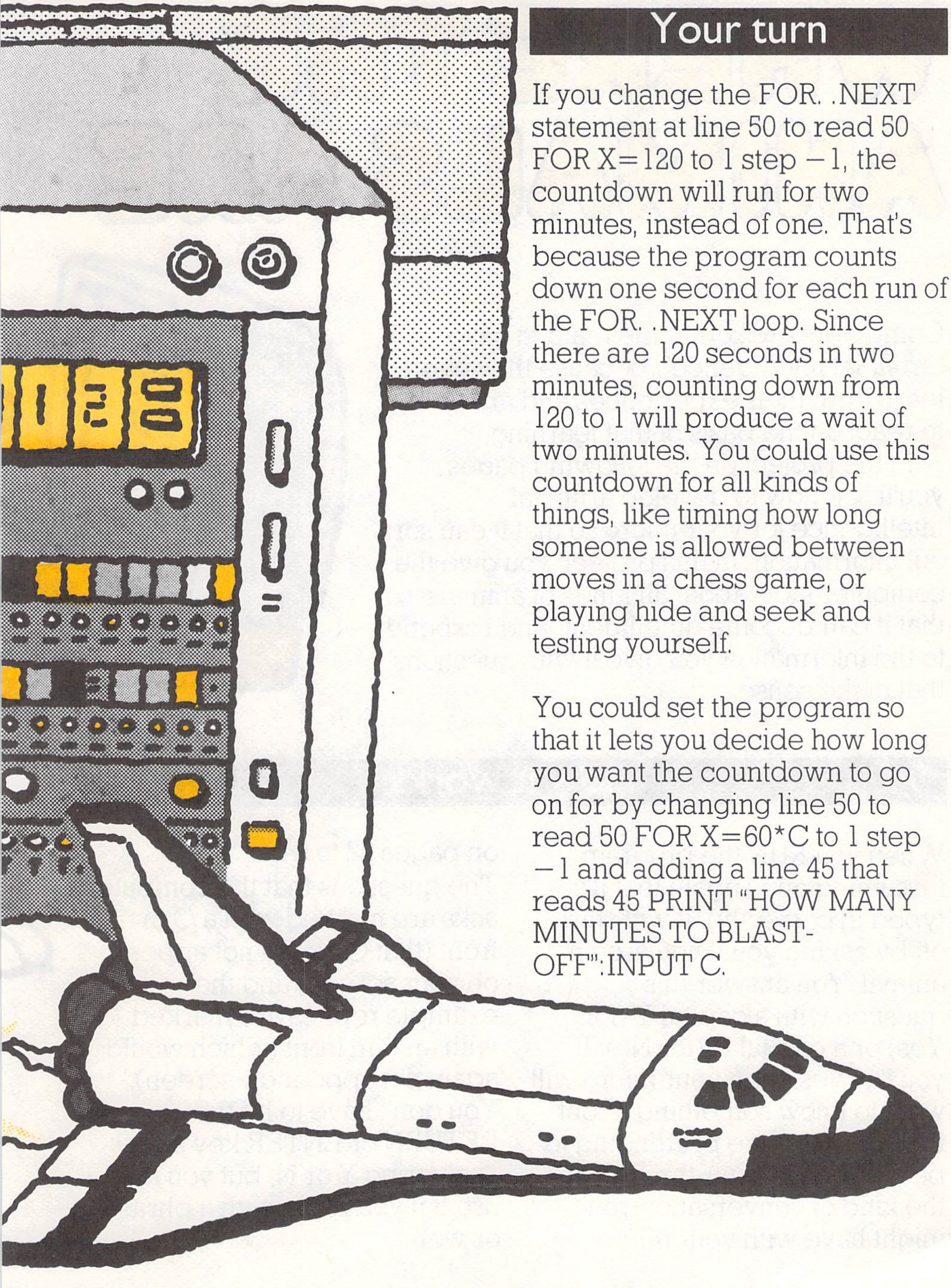
```
 10  PRINT CHR$(147)
 20  FOR T=1 TO 5:PRINT
     CHR$(17);CHR$(29):NEXT
     T:PRINT" THE SHUTTLE IS READY
     FOR"
 30  PRINT"  TAKE-OFF. HERE
     COMES":PRINT"  THE
     COUNTDOWN....."
 40  FOR X=1 TO 1500:NEXT X:PRINT
     CHR$(147)
 50  FOR X=60 TO 1 STEP -1
 60  FOR T=1 TO 5:PRINT
     CHR$(17);CHR$(29):NEXT T
 80  PRINT "  T MINUS ";X;" AND
     COUNTING"
 90  FOR G=1 TO 650:NEXT G
100  PRINT CHR$(147)
110  NEXT X
120  PRINT CHR$(147)
130  REM:AND NOW THE SPACESHIP
     TAKES OFF
140  PRINT "  BLAST-OFF!!!  "
150  GOTO 140
```





How it works

The word "STEP" was used to control the countdown, together with the words FOR and NEXT. FOR and NEXT are usually used to count from a low number to a high number. For example, the statement FOR X=1 TO 10:PRINT X:NEXT X will start at X=1, print the number 1 on screen and then be asked for the NEXT X, which will go through and do the same thing all over again (except this time X is equal to 2 and the number 2 is printed). So the number X is printed ten times in this FOR. .NEXT statement. If a STEP −1 were added and the numbers for X were swapped, then the statement would read FOR X=10 TO STEP −1:PRINT X:NEXT X and the number printed on-screen would start at 10 and slowly go down to 1.



Your turn

If you change the FOR. .NEXT statement at line 50 to read 50 FOR X=120 to 1 step −1, the countdown will run for two minutes, instead of one. That's because the program counts down one second for each run of the FOR. .NEXT loop. Since there are 120 seconds in two minutes, counting down from 120 to 1 will produce a wait of two minutes. You could use this countdown for all kinds of things, like timing how long someone is allowed between moves in a chess game, or playing hide and seek and testing yourself.

You could set the program so that it lets you decide how long you want the countdown to go on for by changing line 50 to read 50 FOR X=60*C to 1 step −1 and adding a line 45 that reads 45 PRINT "HOW MANY MINUTES TO BLAST-OFF":INPUT C.

WHERE THE ANIMALS ARE

Computers are sometimes said to have "artificial intelligence", because they can learn from their experiences and are able to react on the basis of that learning.

In the project on the following pages, you'll see how to develop artificial intelligence for your micro so that it can sort out information. In this project, you give the computer facts about all kinds of animals, so that it can become "intelligent" and respond to the information you give it with questions that make sense.

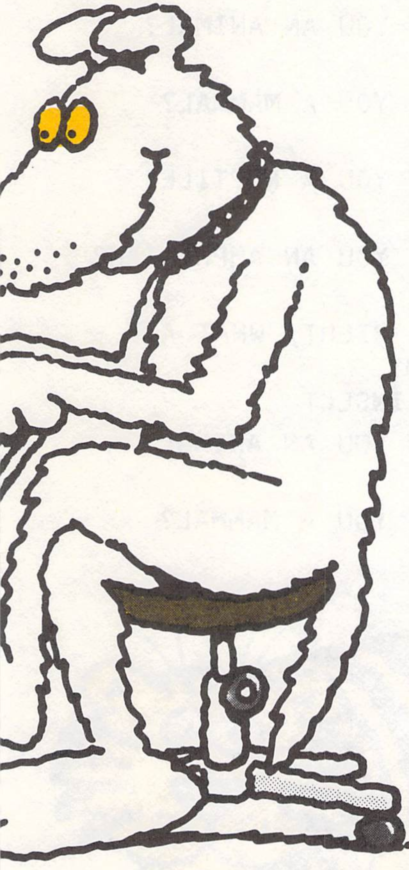
How it works

When you RUN the program (assuming once again that it's typed in correctly), it will start off by asking you if you are an animal. You answer this question with a capital Y (for Yes) or a capital N (for No). If you say yes, then your micro will want to know something about the animal you're pretending to be. Look at the run-through of the kind of conversation you might have with your micro on pages 22 to 24.

The questions that the computer asks are marked with a Q in front (that Q would not appear on your screen) and the example replies are marked with an R in front (which would again not appear on-screen). You don't have to hit the RETURN or ENTER key after answering Y or N, but you must use it if you reply with a phrase or word.

Try this

```
 10  PRINT CHR$(147)
 15  DIM QA$(256)
 20  QA$(1)="AN ANIMAL"
 30  R=-1:A=0
 40  R=R+1:PRINT "ARE YOU ";
      QA$(A+(2↑R));"?"
 50  GET Q$:IF Q$="" THEN 50
 60  IF Q$="Y" THEN 120
 70  IF Q$="N" THEN 100
 80  R=R-1
 90  GOTO 40
100  IF QA$(A+(2↑R)+(2↑R))=""
      THEN 180
110  GOTO 40
120  IF QA$(A+(2↑R)+(2↑(R+1)))=""
      THEN 150
130  A=A+(2↑R)
140  GOTO 40
150  PRINT "WHAT KIND OF ";
      QA$(A+(2↑R));" ARE YOU ";
160  INPUT QA$(A+(2↑R)+(2↑(R+1)))
170  GOTO 30
180  PRINT "ALRIGHT, WHAT ARE YOU";
190  INPUT QA$(A+(2↑R)+(2↑R))
200  GOTO 30
```

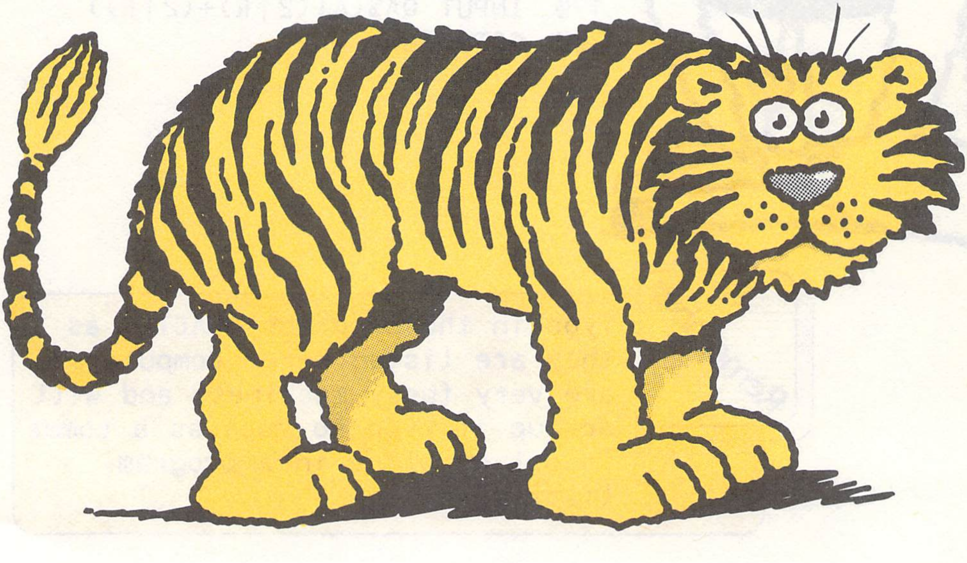


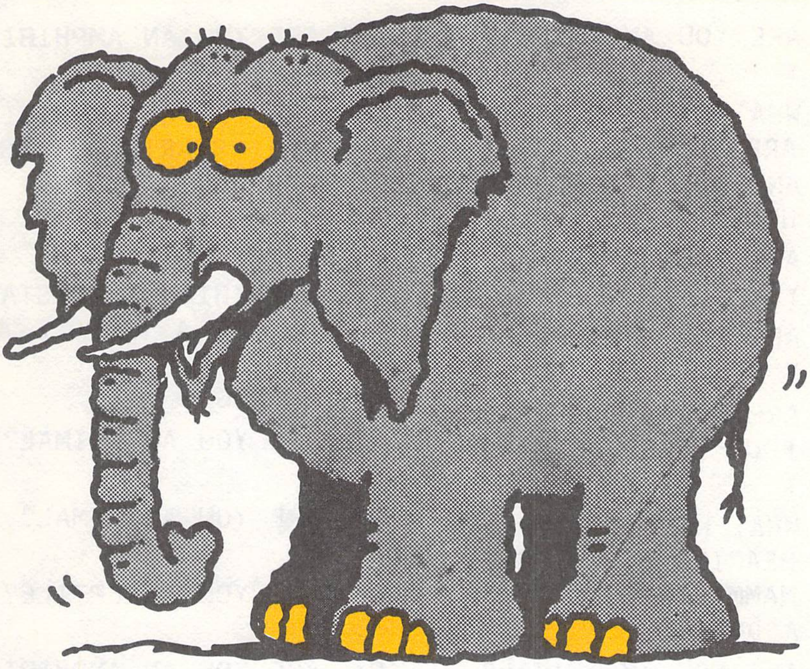
Remember

Type in the programs EXACTLY as they are listed here. Computers are very fussy machines, and will act up if even so much as a comma is out of place in a program listing.

Your conversation

```
Q:  ARE YOU AN ANIMAL?
A:  Y
Q:  WHAT KIND OF AN
     ANIMAL ARE YOU?
A:  A MAMMAL
Q:  ARE YOU AN ANIMAL?
A:  Y
Q:  ARE YOU A MAMMAL?
A:  N
Q:  ALL RIGHT, WHAT ARE
     YOU?
A:  A REPTILE
Q:  ARE YOU AN ANIMAL?
A:  Y
Q:  ARE YOU A MAMMAL?
A:  N
Q:  ARE YOU A REPTILE?
A:  N
Q:  ALL RIGHT, WHAT ARE
     YOU?
A:  AN AMPHIBIAN
Q:  ARE YOU AN ANIMAL?
A:  Y
Q:  ARE YOU A MAMMAL?
A:  N
Q:  ARE YOU A REPTILE?
A:  N
Q:  ARE YOU AN AMPHIBIAN?
A:  N
Q:  ALL RIGHT, WHAT ARE
     YOU?
A:  AN INSECT
Q:  ARE YOU AN ANIMAL?
A:  Y
Q:  ARE YOU A MAMMAL?
A:  Y
```





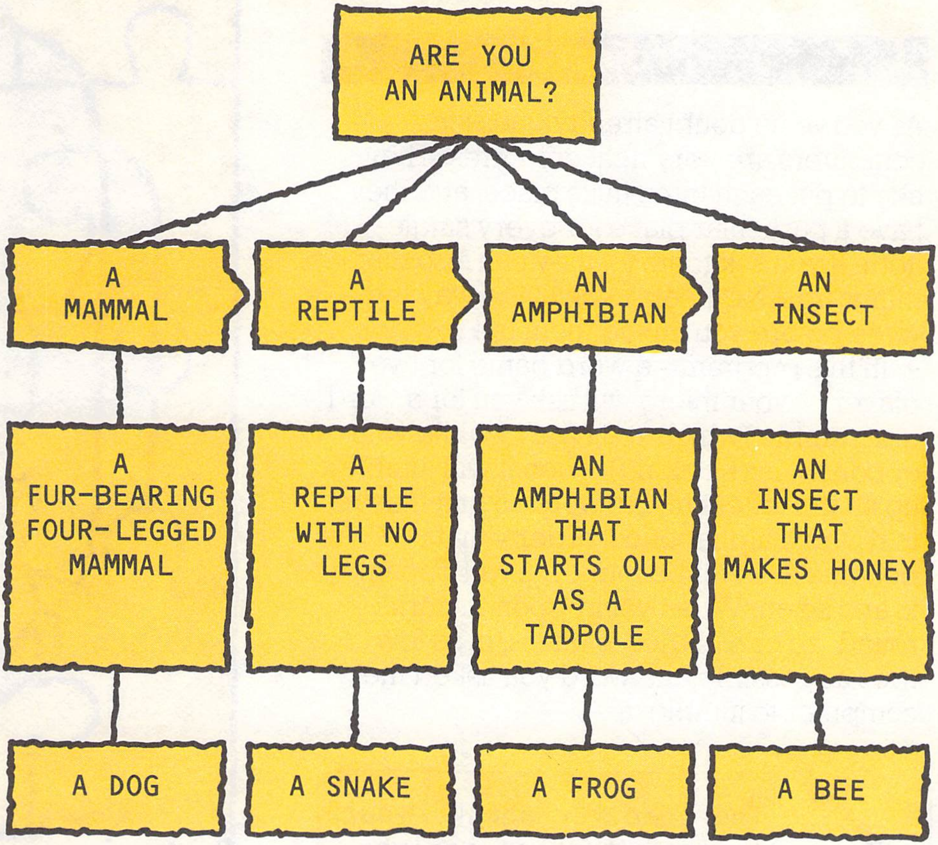
```
Q:  WHAT KIND OF A MAMMAL
     ARE YOU?
A:  A FUR-BEARING FOUR-
     LEGGED MAMMAL
Q:  ARE YOU AN ANIMAL?
A:  Y
Q:  ARE YOU A MAMMAL?
A:  N
Q:  ARE YOU A REPTILE?
A:  Y
Q:  WHAT KIND OF A
     REPTILE ARE YOU?
A:  A REPTILE WITH NO
     LEGS
Q:  ARE YOU AN ANIMAL?
A:  Y
Q:  ARE YOU A MAMMAL?
A:  N
```

```
Q:  ARE YOU A REPTILE?
A:  N
Q:  ARE YOU AN AMPHIBIAN?
A:  Y
Q:  WHAT KIND OF AN
     AMPHIBIAN ARE YOU?
A:  AN AMPHIBIAN THAT
     STARTS OUT AS A
     TADPOLE
Q:  ARE YOU AN ANIMAL?
A:  Y
Q:  ARE YOU A MAMMAL?
A:  N
Q:  ARE YOU A REPTILE?
A:  N
Q:  ARE YOU AN AMPHIBIAN?
A:  N
```

```
Q:  ARE YOU AN INSECT?
A:  Y
Q:  WHAT KIND OF INSECT
    ARE YOU?
A:  AN INSECT THAT MAKES
    HONEY
Q:  ARE YOU AN ANIMAL?
A:  Y
Q:  ARE YOU A MAMMAL?
A:  Y
Q:  ARE YOU A FUR-BEARING
    FOUR-LEGGED MAMMAL?
A:  Y
Q:  WHAT KIND OF FUR-
    BEARING FOUR-LEGGED
    MAMMAL ARE YOU?
A:  A DOG
Q:  ARE YOU AN ANIMAL?
A:  Y
Q:  ARE YOU A MAMMAL?
A:  N
Q:  ARE YOU A REPTILE?
A:  Y
Q:  ARE YOU A REPTILE
    WITH NO LEGS?
A:  Y
Q:  WHAT KIND OF REPTILE
    WITH NO LEGS ARE YOU?
A:  A SNAKE
Q:  ARE YOU AN ANIMAL?
A:  Y
Q:  ARE YOU A MAMMAL?
A:  N
Q:  ARE YOU A REPTILE?
A:  N
Q:  ARE YOU AN AMPHIBIAN?
A:  Y
Q:  ARE YOU AN AMPHIBIAN
    THAT STARTS OUT AS A
    TADPOLE?
A:  Y
Q:  WHAT KIND OF
    AMPHIBIAN THAT STARTS
    OUT AS A TADPOLE ARE
    YOU?
A:  A FROG
Q:  ARE YOU AN ANIMAL?
A:  Y
Q:  ARE YOU A MAMMAL?
A:  N
Q:  ARE YOU A REPTILE?
A:  N
Q:  ARE YOU AN AMPHIBIAN?
A:  N
Q:  ARE YOU AN INSECT?
A:  Y
Q:  ARE YOU AN INSECT
    THAT MAKES HONEY?
A:  Y
Q:  WHAT KIND OF AN
    INSECT THAT MAKES
    HONEY ARE YOU?
A:  A BEE
```

How it works

Each question you answer about an animal or type of animal adds to the computer's information about that particular one, in the way shown below.



Your turn

If you say NO to the original question of 'Are you an animal?', you can turn the program into an animal, vegetable, mineral game by giving your computer information about animals, vegetables and minerals.

Mixing it up wi your micro

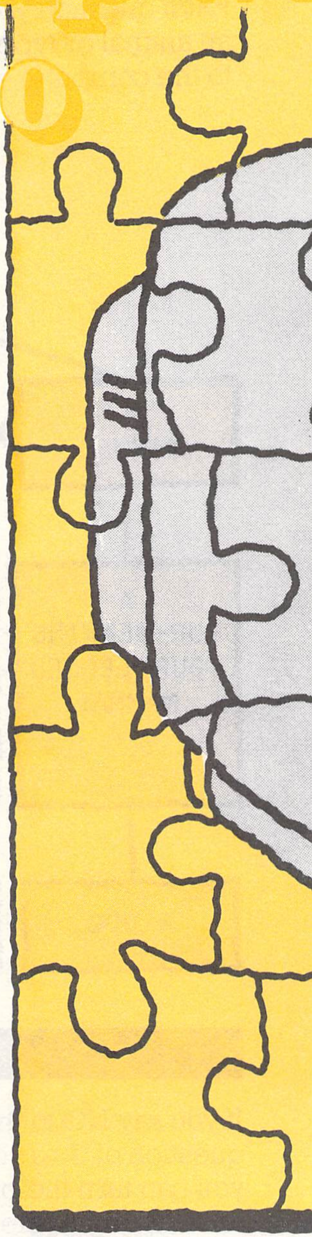
Project

As you've no doubt already guessed, computers are very neat creatures. They like to put each thing in its place, and they have a particular place for every single item. If you ask them to, they can also mix things up AND keep track of the way they were before you mixed them up!

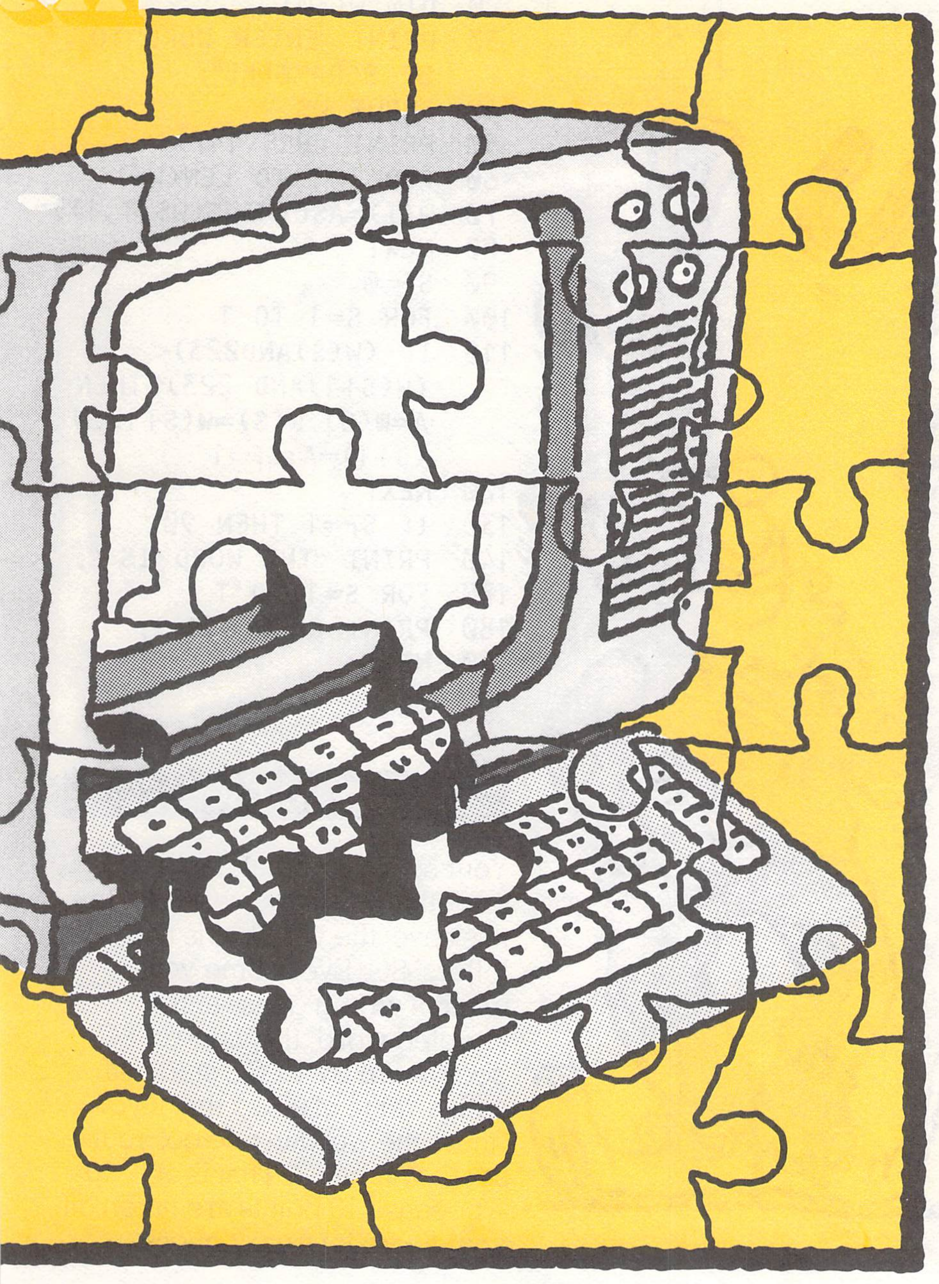
In this program – a word game for two players – your micro will ask you for a word to scramble (which you type in while your opponent isn't looking). Then it will jumble up all the letters in that word so that they're in reverse alphabetical order – that is, a will become z, b will become y, c will become x, and so on. When your opponent turns round, he or she tries to guess, in as few tries as possible, the word you asked the computer to jumble up.

Remember

The computer checks your opponent's guess against your original unjumbled word by 'string-matching' — which lets you compare the value of one word with the value of another until a match is found.



th



Try this

```
 10  PRINT CHR$(147)
 15  SC=90
 20  DIM W(30)
 30  PRINT “ENTER WORD TO
     BE SCRAMBLED”;
 40  INPUT W$
 50  PRINT CHR$(147)
 60  FOR T=1 TO LEN(W$)
 70  W(T)=ASC(MID$(W$,T,1))
 80  NEXT
 90  SF=0
100  FOR S=1 TO T
110  IF (W(S)AND223)<
     (W(S+1)AND 223) THEN
     A=W(S):W(S)=W(S+1):W
     (S+1)=A:SF=1
120  NEXT
130  IF SF=1 THEN 90
140  PRINT “THE WORD IS ”;
170  FOR S=1 TO T
180  PRINT CHR$(W(S));
190  NEXT
```

Did you see?.

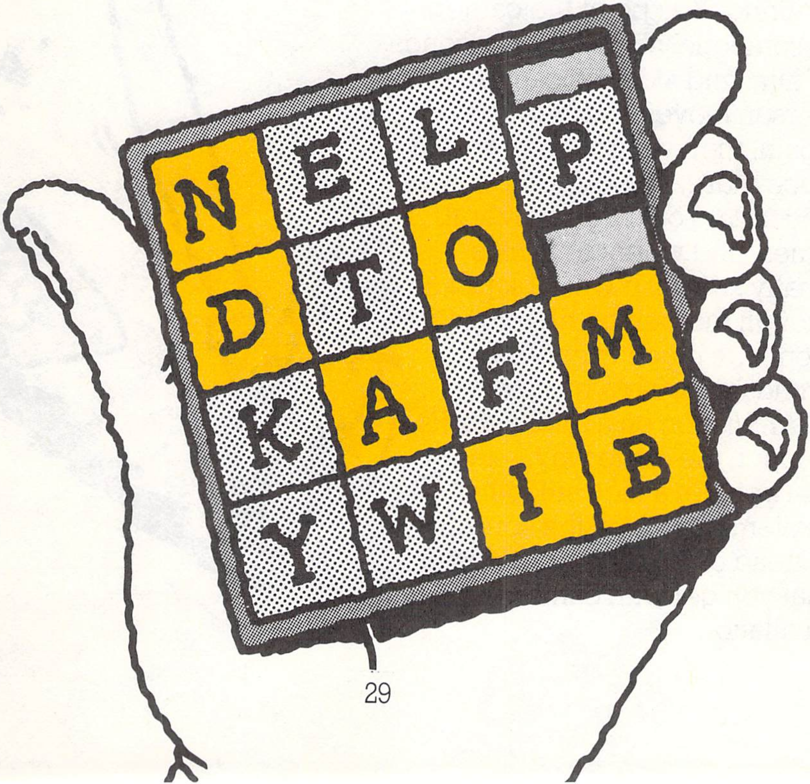
Your score is calculated using something called a variable. In this case, the variable is two letters: SC. Every time you make a wrong guess at the jumbled word, the value of SC goes down by 10. SC starts off with a value of 90, so that if you guess the word in one go, your score will be 80. That is, 90 – 10 – because 10 points are taken off your score for every guess.

```
200  PRINT
220  SC=SC-10
230  INPUT "THE REAL WORD
     IS ";R$
240  PRINT "SCORE:";SC
250  IF SGN(SC)=-1 THEN
     310
260  IF W$<>R$ THEN 220
270  PRINT "NICE ONE. ";
280  PRINT "ANOTHER
     GAME?";
290  INPUT A$:IF A$="Y"
     THEN RUN
300  END
310  PRINT "YOU DON'T SEEM
     TO HAVE GOT IT YET
     SO I'LL TELL YOU. IT
     IS: ";W$
320  GOTO 280
```

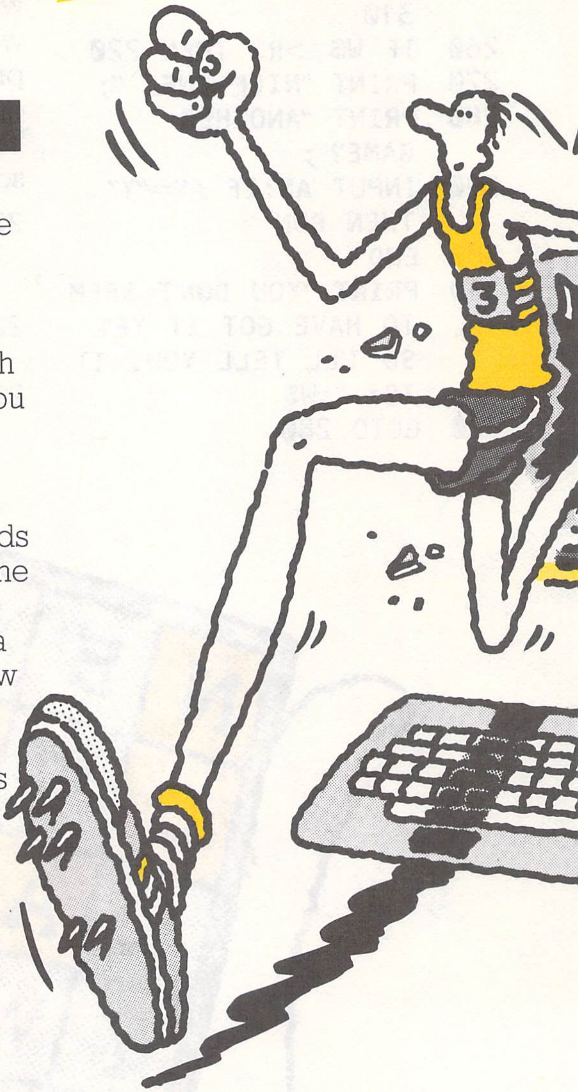
Your turn

You could build a "hint" into the program by letting people have a quick peek at the correct word. The best way to do this would be an extra line or two of program just before they have a guess at the word in line 230. The extra lines would go something like this:

```
225  INPUT "DO YOU WANT A
     QUICK LOOK AT THE
     WORD";Y$
226  IF Y$<>"Y" THEN GOTO
     230
227  PRINT W$:PRINT
     CHR$(147)
```



Run man



Project

One of the things a computer perhaps does best is to animate or move things on screen. This is done by what is called pixillation.

Pixillation is what you do with a home movie camera when you want to make it look as if someone is moving without walking. You point the camera at someone, shoot a few seconds of film, and stop the camera. The person moves and stands still again, then you roll the camera once more, stopping after a few seconds. You do this several times, and when the film is finally shown, it will look just as though the person has moved from one place to another without moving a muscle.

Animation on a micro works in much the same way, except that you get a nice smooth movement across the screen instead of the jerky movement that you get with camera pixillation.

run

How it works

In this program, you'll get a little 'man' to run across the screen by printing two images, one of a man with legs together, and the other with them apart. The image will be printed progressively in different places on the screen with a statement to clear the screen between each appearance of the image. Here is how it's done.

There are two images. The first shows our man standing still, and it is made up of an = sign, a # mark, a – sign, a [symbol, a] symbol and the letter I.

The man standing still looks like this:

```
  ==
   []
  ##
  ##--
  I I
  I I
  I I
```

When the man is in full flight while running, he looks like this:

```
  ==
   []
  ##
  ##--
  I I
 I   I
I     I
```

Remember that when you type this program in, you will have to press either the Commodore or Shift key to get some of the symbols.

Try this

```
   10  PRINT " ==   "
   20  PRINT " []   "
   30  PRINT " ##   "
   40  PRINT " ##--"
   50  PRINT " II   "
   60  PRINT " II   "
   70  PRINT " II   "
   75  GOSUB 80:GOTO 90
   80  FOR N=1 TO 200:NEXT
       N:RETURN
   90  PRINT CHR$(147)
  100  PRINT "        ==   "
  200  PRINT "        []   "
  300  PRINT "        ##   "
  400  PRINT "        ##--"
  500  PRINT "        II   "
  600  PRINT "       I  I  "
  700  PRINT "      I    I "
  800  PRINT "             "
  900  GOSUB 80
  950  PRINT CHR$(147)
 1000  PRINT "              ==   "
 2000  PRINT "              []   "
 3000  PRINT "              ##   "
 4000  PRINT "              ##--"
 5000  PRINT "              II   "
 6000  PRINT "              II   "
 7000  PRINT "              II   "
 8000  GOSUB 80
 9000  PRINT CHR$(147)
10010  PRINT "                  ==   "
20010  PRINT "                  []   "
30010  PRINT "                  ##   "
40010  PRINT "                  ##--"
50010  PRINT "                  II   "
60010  PRINT "                 I  I  "
60011  PRINT "                I    I "
60012  PRINT CHR$(147):GOTO 10
```

Did you see?

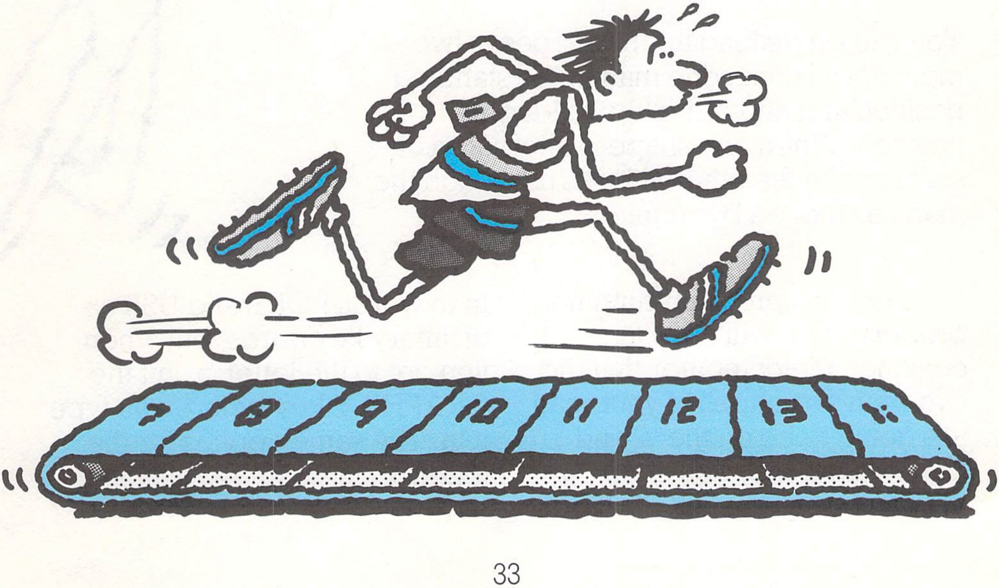
The animated man was made entirely of ordinary letters. He could however have been made just as easily from special block characters which you can type from your micro's keyboard. In chapter 6, you'll see how to use those block characters.

Your turn

You probably noticed that the animated man moved quite slowly – that's because of the FOR. .NEXT statement at line 80, which controls how long each picture of the man stays on the screen. Line 80 has a FOR. .NEXT statement which says FOR N=1 to 200:NEXT N:RETURN. This means that the computer counts up to 200 before it RETURNs to the next bit of the program and displays the man in a new position. So to change the speed of the man's movement, change the Line 80 to something like:

```
FOR N=1 to 100:NEXT
N:RETURN
```

As you can see, the only change is in the number to which the computer counts up. It has been changed from 100 to 200. To make the man move even faster, you could lower that number even further.



Giving the man some character

Project

Although the man in the last project may have moved across the screen, he was pretty jerky, and only made up of letters.

In this project, you'll see how to make up a much better-built man, and move him across a screen smoothly. This man will be made up of a user-defined or programmable character. All that means is that you fool the computer into thinking a certain letter looks like a man, a spaceship or whatever you want it to look like.

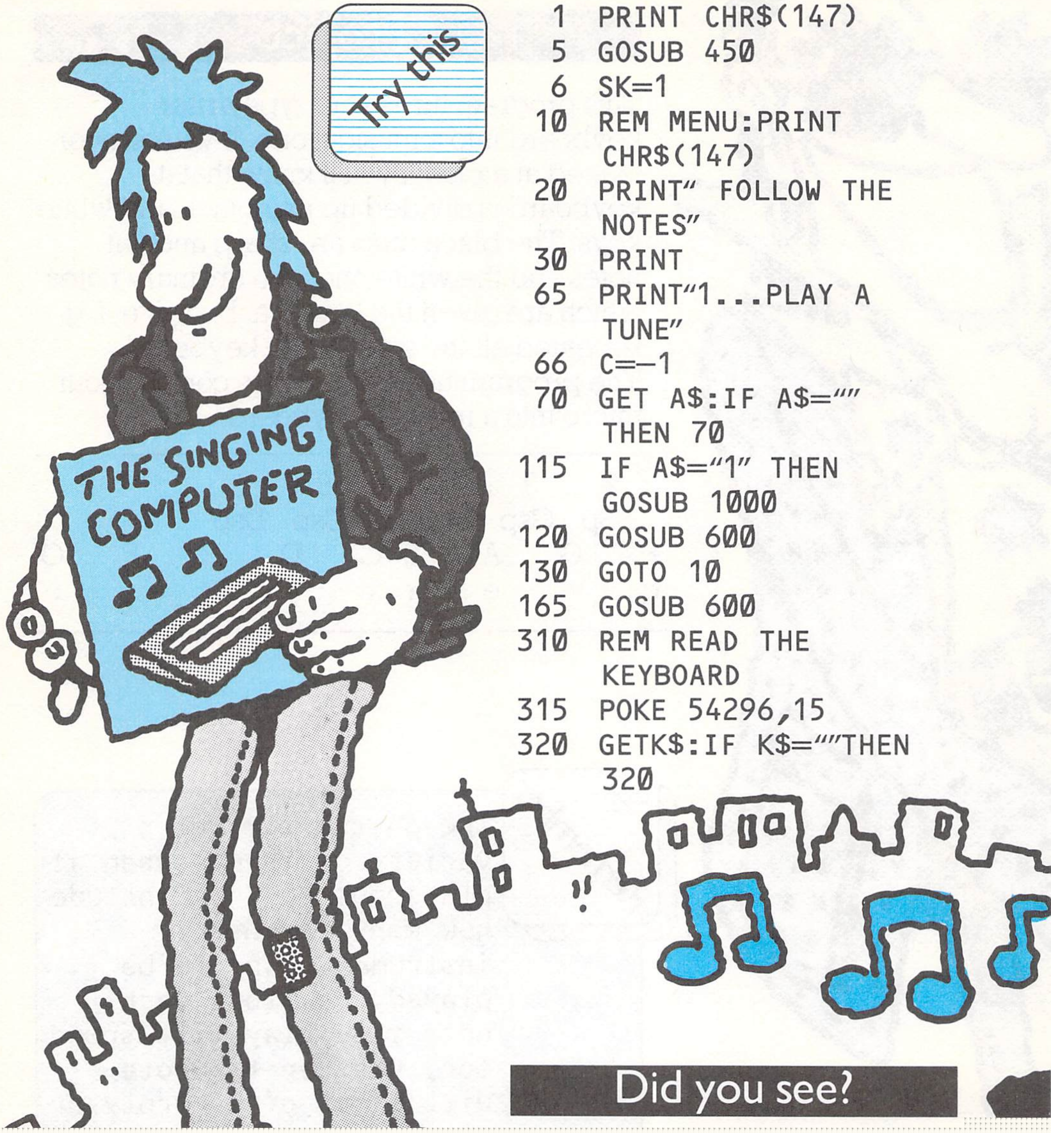
How it works

You will be making up the shapes of two men, not just one: one man in the standing position and another in the full-flight position. When one shape is alternated on-screen with the other, it looks as though the man has moved by running.

To get this program running properly, you will have to convince the computer that the two running shapes of your man are going to be on the 'a' and 'b' keys. This is how you do it. When you type in line 30, type in th words POKE and USR as ordinary keywords, but when you get to the letter 'a', hit the GRAPHICS key before you type it in. The same applies for the letters in lines 70, 130 and 140.

Try this

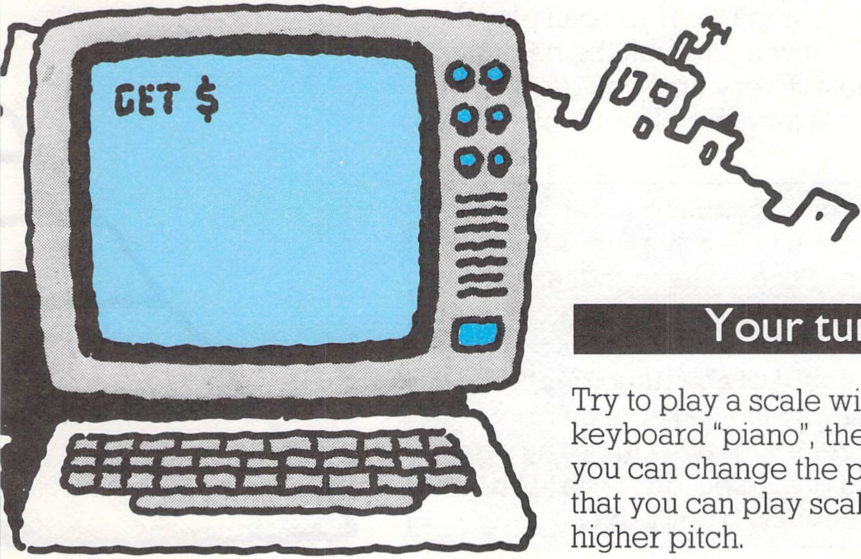
```
  1  PRINT CHR$(147)
  5  GOSUB 450
  6  SK=1
 10  REM MENU:PRINT
      CHR$(147)
 20  PRINT” FOLLOW THE
      NOTES”
 30  PRINT
 65  PRINT“1...PLAY A
      TUNE”
 66  C=-1
 70  GET A$:IF A$=“”
      THEN 70
115  IF A$=“1” THEN
      GOSUB 1000
120  GOSUB 600
130  GOTO 10
165  GOSUB 600
310  REM READ THE
      KEYBOARD
315  POKE 54296,15
320  GETK$:IF K$=“”THEN
      320
```



Did you see?

```
330  F1=(K$="Q")*N(0)+
     (K$="2")*N(1)+
     (K$="W")*N(2)+
     (K$="3")*N(3)+
     (K$="E")*N(4)
340  F1=(K$="R")*N(5)+
     (K$="5")*N(6)+
     (K$="T")*N(7)+
     (K$="6")*N(8)+
     (K$="Y")*N(9)+F1
345  F1=(K$="7")*N(10)+
     (K$="U")*N(11)+
     (K$="I")*N(12)+F1
350  IF F1=0THEN320
360  F1=-F1
361  GOSUB600
365  POKEI,F1-INT
     (F1/256)*256
366  POKEH,INT(F1/256)
370  RETURN
380  REM PLAY THAT TUNE
     AGAIN
```

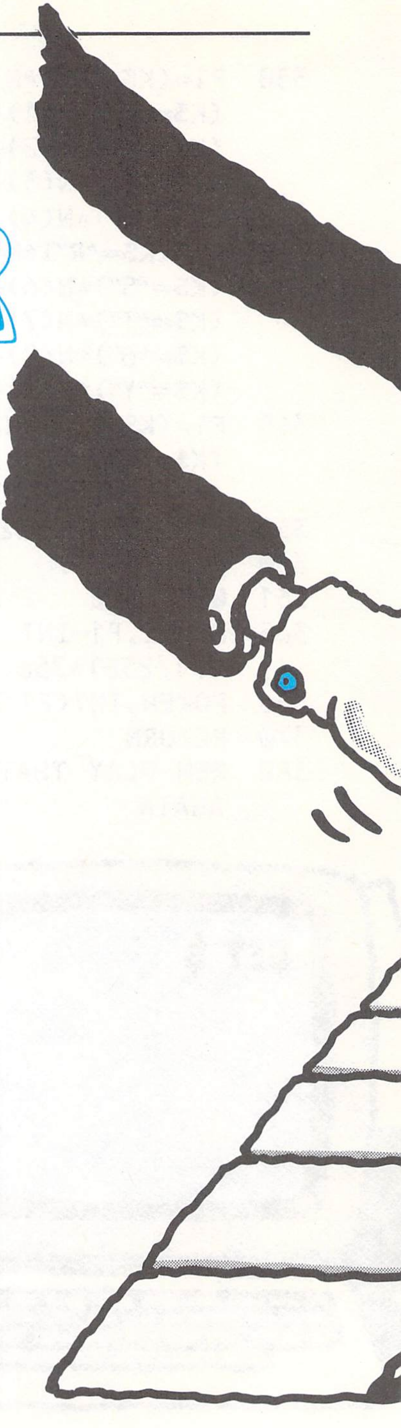
```
450  REM INITIALISE
     SOUND CHIP
460  V=54296:AD=54277:
     SR=54278:W=54276
470  GOSUB 600
480  H=54273:I=54272
490  DIM N(12),S(30)
500  FOR T=0 TO 12:READ
     A:N(T)=A:NEXT
510  DATA 2195,2325,
     2463,2630,2795,2930,
     3104,3288,3484,3691,
     3910,4142,4389
520  DEF FNR(X)=INT
     (RND(1)*X)
530  RETURN
600  FOR B=0TO9:POKEB+
     54272,0:NEXTB
605  POKEV,15:POKEAD,10:
     POKESR,0:POKEW,17
610  RETURN
1000 GOSUB 320
1010 GOTO1000
```



Your turn

Try to play a scale with your keyboard "piano", then see if you can change the program so that you can play scales at a higher pitch.

MIRROR MIRROR ?



Project

Now that you've got the micro to be a musical instrument, you can get it to do the finger work as well.

You can make it compose a tune, by getting it to mix up a bunch of notes and play them back. It will play you a bar (that's four notes) of music and then let you try to play back the same bar. If you can play the bar back to it, it will play you the bar again and add a note. Every time you get it right the bar will get longer.

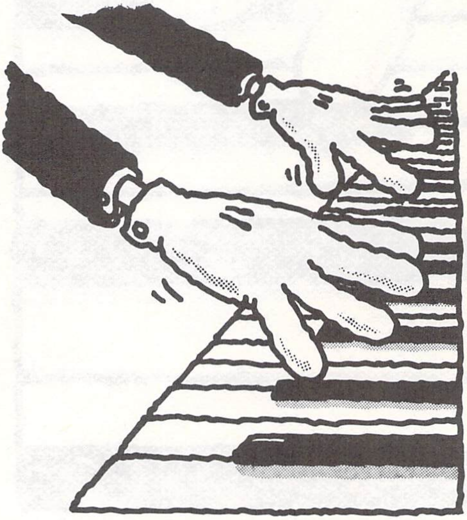
Remember

Your computer can come up with a value which neither the programmer or the player can determine before it happens: it's completely unexpected. It's the electronic version of throwing some dice, and it's called the random function.

Try this

```
  5  GOSUB 450
  6  SK=1
 10  REM MENU
 20  PRINT “ FOLLOW THE
     NOTES”
 30  PRINT
 40  PRINT“1...NEW RANDOM
     TUNE”
 50  PRINT“2...OLD RANDOM
     TUNE”
 60  PRINT“3...END”
 65  PRINT“4...PLAY A
     TUNE”
 66  C=-1
 70  GET A$:IF A$=“”
     THEN 70
 90  IF A$=“1”THEN GOSUB
     140:GOSUB 230
100  IF A$=“2” THEN
     GOSUB 390
110  IF A$=“3”THEN END
115  IF A$=“4” THEN
     GOSUB 1000
120  GOSUB 600
130  GOTO 10
140  REM PLAY THE RANDOM
     TUNE
150  FOR T=0 TO SK
160  F=N(FNR(12)):S(T)=F
161  IF F=N(1)ORF=N(3)
     ORF=N(6)ORF=N(8)
     ORF=N(10)THEN160
165  GOSUB 600
170  POKE I,F-INT(F/
     256)*256
180  POKE H,INT(F/256)
190  FOR D=0 TO150:NEXT
200  NEXT
210  RETURN
```

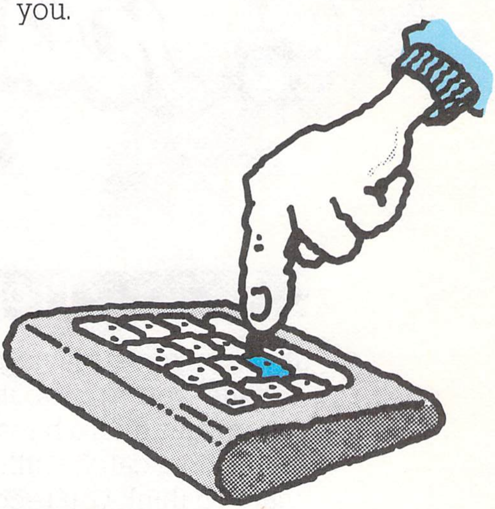
```
220  REM GET TUNE FROM
     KEYBOARD
230  FOR T=0 TO SK
240  GOSUB 320
250  IF S(T)<>F1 THEN
     PRINT CHR$(119);
255  IF S(T)=F1 THEN
     PRINT CHR$(113);:
     C=C+1
260  NEXT
265  PRINT
266  IF C<>SKTHEN290
267  SK=SK+1
270  PRINT"
     CONGRATULATIONS
     MAESTRO"
275  FORT=1TO8:GOSUB 600
     :POKEL,0:POKEH,
     10*T:FORD=1TO10:
     NEXT:NEXT
276  GOSUB 540
280  RETURN
290  PRINT"BAD LUCK!"
295  FORT=1TO8:GOSUB 600
     :POKEL,0:POKEH,
     9-T:FORD=1TO10:
     NEXT:NEXT
297  GOSUB 540
300  RETURN
310  REM READ THE
     KEYBOARD
320  GETK$:IF K$=""THEN
     320
330  F1=(K$="Q")*N(0)+
     (K$="2")*N(1)+
     (K$="W")*N(2)+
     (K$="3")*N(3)+
     (K$="E")*N(4)
340  F1=(K$="R")*N(5)+
     (K$="5")*N(6)+
     (K$="T")*N(7)+
     (K$="6")*N(8)+
     (K$="Y")*N(9)+F1
345  F1=(K$="7")*N(10)+
     (K$="U")*N(11)+
     (K$="I")*N(12)+F1
350  IF F1=0THEN320
360  F1=-F1
361  GOSUB600
365  POKEL,F1-INT(F1/256)
     *256
366  POKEH,INT(F1/256)
370  RETURN
```



```
380  REM PLAY THAT TUNE
      AGAIN
390  FOR T=0 TO SK
395  GOSUB600
400  POKEL,S(T)-INT(S(T)/
      256)*256
410  POKEH,INT(S(T)/256)
420  FOR D=0 TO150:NEXT
430  NEXT
440  RETURN
450  REM INITIALISE
      SOUND CHIP
460  V=54296:AD=54277:
      SR=54278:W=54276
470  GOSUB 600
480  H=54273:I=54272
490  DIM N(12),S(30)
500  FOR T=0 TO 12:READ
      A:N(T)=A:NEXT
510  DATA 2195,2325,2463,
      2620,2795,2930,3104,
      3288,3484,3691,
      3910,4143,4389
520  DEF FNR(X)=INT
      (RND(1)*X)
530  RETURN
540  PRINT"PRESS THE
      SPACE BAR TO
      CONTINUE"
550  GET A$:IF A$<>" "
      THEN 550
560  RETURN
600  FOR B=0TO9:POKEB+
      54272,0:NEXTB
605  POKEV,15:POKEAD,
      10:POKESR,0:POKEW,17
610  RETURN
1000 GOSUB 320
1010 GOTO1000
```

Type this in and see how good you are at recognising notes yourself. Who knows, with a bit of practice you might decide you're good enough to become a musician as well as a computer programmer!

Just press the space bar to keep playing. Your mission is to play back the same sequence of notes as the computer plays to you.



Your turn

There are two types of messages in this program. One says "Congratulations, maestro" if you win a game, and the other says "Bad luck" if you don't. Try changing what's inside the quotes on both those messages (a hint: look for the PRINT statements) to your own message.

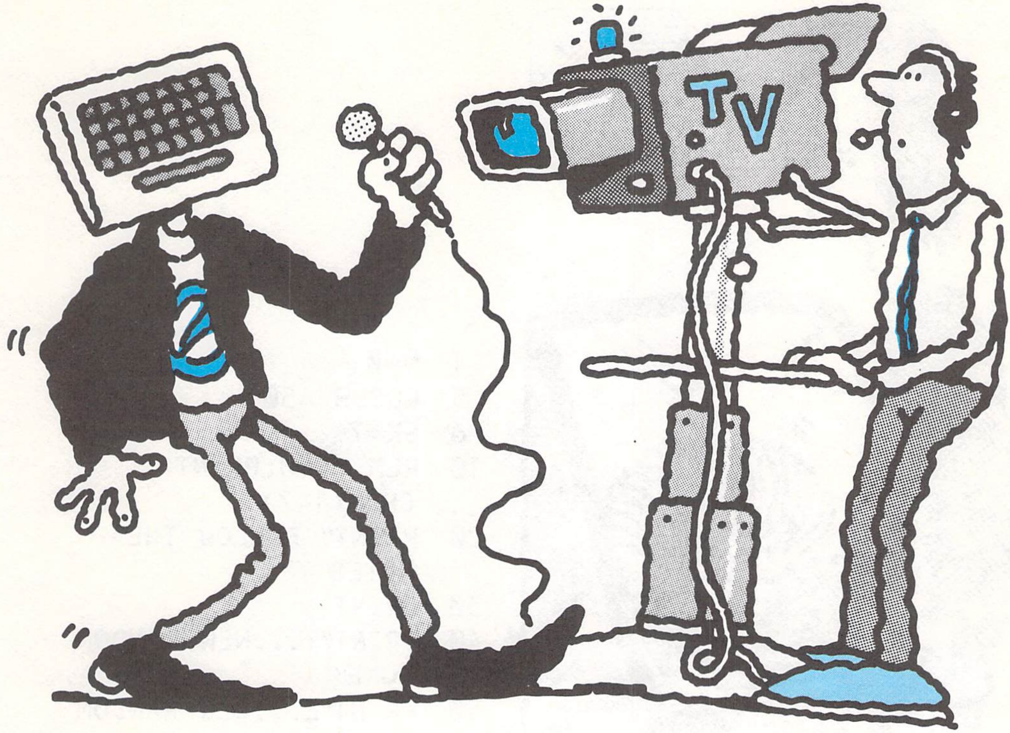
Random rock

Project

If you've been playing with the last program a while, you've probably been surprised when your micro has come up with an occasional catchy little melody, or even one that you think you recognise.

This is not surprising when you consider there are only twelve notes (although they can all be played in several pitches) and thousands of songs. The law of probability says that you will eventually discover a hit tune if you tinker around randomly with a muscial instrument for long enough.

As you've seen, computers are very good at being random, so try to put them to work composing. Most songs have four beats to the bar. A bar is a sort of musical sentence – put enough of these sentences together and you can end up with a song.



With this program you have five choices:

1 New random tune – which will start the computer playing the first few notes of a tune that it composes randomly.

2 Old random tune – will play whatever you've recorded onto the computer's memory using option 5: Attach a tune.

3 End – stops the program.

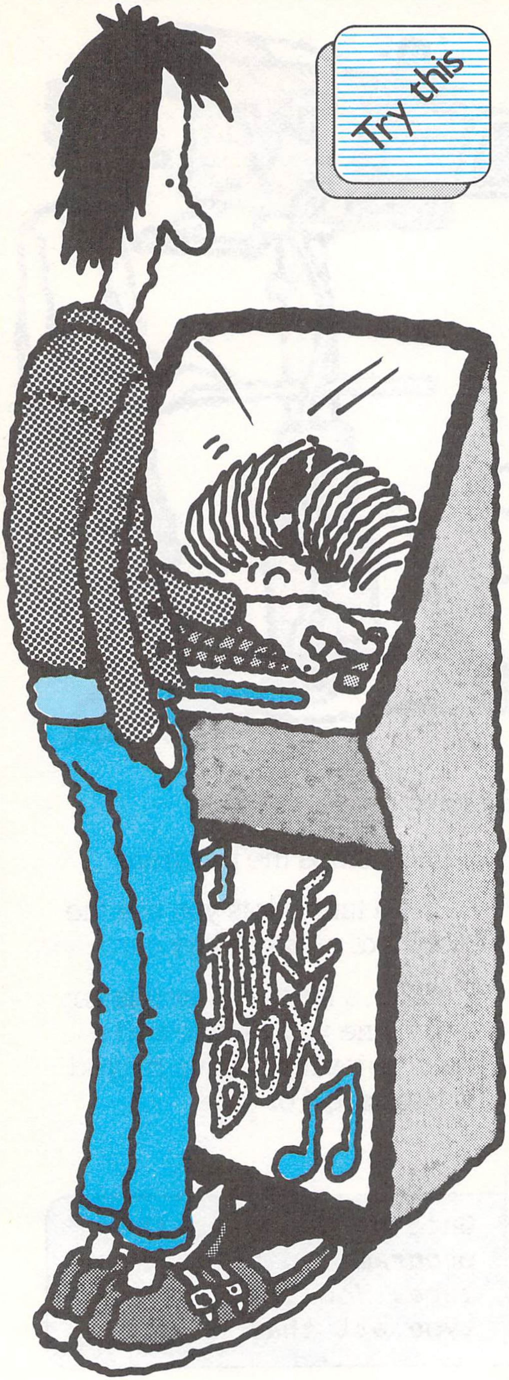
4 Play a tune – lets you use the micro like a piano keyboard.

5 Attach a tune – press this key every time you want to add a bar of notes that you've played by pressing 4 or 1.

Remember

Once you've typed the program in, SAVE it to tape. You don't want to type all that in again.

Try this



```
  1  N=0
  5  GOSUB 450
  6  SK=7
 10  REM MENU:PRINT
      CHR$(147)
 20  PRINT" FOLLOW THE
      NOTES"
 30  PRINT
 40  PRINT"1...NEW RANDOM
      TUNE"
 50  PRINT"2...OLD RANDOM
      TUNE"
 60  PRINT"3...END"
 65  PRINT"4...PLAY A
      TUNE"
 66  PRINT"5...ATTACH
      TUNE"
 69  C=-1
 70  GET A$:IF A$=""
      THEN 70
 90  IF A$="1"THEN GOSUB
      140
100  IF A$="2" THEN
      GOSUB 390
110  IF A$="3"THEN END
115  IF A$="4" THEN
      GOSUB 1000
116  IFA$="5"THENGOSUB
      2000
```

```
120  GOSUB 6ØØ
130  GOTO 69
140  REM PLAY THE RANDOM
     TUNE
150  FOR T=Ø TO SK
160  F=N(FNR(12)):S(T)=F
161  IFF=N(1)ORF=N(3)
     ORF=N(6)ORF=N(8)ORF=
     N(1Ø)THEN16Ø
165  GOSUB 6ØØ
170  POKE L,F-INT(F/
     256)*256
180  POKE H,INT(F/256)
190  FOR D=Ø TO15Ø:NEXT
200  NEXT
210  RETURN
220  REM GET TUNE FROM
     KEYBOARD
230  FOR T=Ø TO SK
240  GOSUB 32Ø
250  IF S(T)<>F1 THEN
     PRINT CHR$(119);
255  IF S(T)=F1 THEN
     PRINT CHR$(113);:
     C=C+1
260  NEXT
265  PRINT
266  IF C<>SKTHEN 29Ø
267  SK=SK+1
270  PRINT"
     CONGRATULATIONS
     MAESTRO"
275  FORT=1TO8:GOSUB 6ØØ:
     POKEL,Ø:POKEH,1Ø*T:
     FORD=1TO1Ø:NEXT:
     NEXT
276  GOSUB 54Ø
280  RETURN
290  PRINT"BAD LUCK!"
```

```
295  FORT=1TO8:GOSUB 6ØØ:
     POKEL,ØPOKEH,9-T:
     FORD=1TO1Ø:NEXT:
     NEXT
297  GOSUB 54Ø
300  RETURN
310  REM READ THE
     KEYBOARD
320  GETK$:IF K$=""THEN
     32Ø
330  F1=(K$="Q")*N(Ø)+
     (K$="2")*N(1)+
     (K$="W")*N(2)+
     (K$="3")*N(3)+
     (K$="E")*N(4)
340  F1=(K$="R")*N(5)+
     (K$="5")*N(6)+
     (K$="T")*N(7)+
     (K$="6")*N(8)+
     (K$="Y")*N(9)+F1
345  F1=(K$="7")*N(1Ø)+
     (K$="U")*N(11)+
     (K$="I")*N(12)+F1
```

```
350  IF F1=0THEN320
360  F1=-F1
361  GOSUB600
365  POKEL,F1-INT
     (F1/256)*256
366  POKEH,INT(F1/256)
370  RETURN
380  REM PLAY THAT TUNE
     AGAIN
390  FOR T=0 TO N
395  GOSUB600
400  POKEL,T(T)-INT(T(T)/
     256)*256
410  POKEH,INT(T(T)/256)
420  FOR D=0 TO150:NEXT
430  NEXT
440  RETURN
450  REM INITIALISE
     SOUND CHIP
460  V=54296:AD=54277:
     SR=54278:W54276
470  GOSUB 600
480  H=54273:I=54272
490  DIM N(12),S(8),
     T(100),N$(13)
500  FOR T=0 TO 12:READ
     A,A$:N$(T)=A$:N(T)
     =A:NEXT
510  DATA 2195,C,2325,
     C#,2463,D,2620,D#,
     2795,E,2930,F
515  DATA 3104,F#,3288,
     G,3484,G#,3691,A,
     3910,A#,4143,B,
     4389,C
520  DEF FNR(X)=INT(RND
     (1)*X)
530  RETURN
```

```
 540  PRINT"PRESS THE
       SPACE BAR TO
       CONTINUE"
 550  GET A$:IF A$<>" "
       THEN 55Ø
 56Ø  RETURN
 6ØØ  FOR B=ØTO9:POKEB+
       54272,Ø:NEXTB
 6Ø5  POKEV,15:
       POKEAD,1Ø:POKESR,
       Ø:POKEW,17
 61Ø  RETURN
1ØØØ  FORT=ØTO7
1Ø1Ø  GOSUB32Ø
1Ø2Ø  S(T)=F1
1Ø3Ø  NEXT
1Ø4Ø  RETURN
2ØØØ  FORT=ØTOSK
2Ø1Ø  T(T+N)=S(T)
2Ø15  FORS=ØTO12
2Ø16  IFN(S)=S(T)THEN
       PRINTN$(S);
2Ø17  NEXT
2Ø3Ø  NEXT
2Ø4Ø  N=N+T
2Ø5Ø  RETURN
```

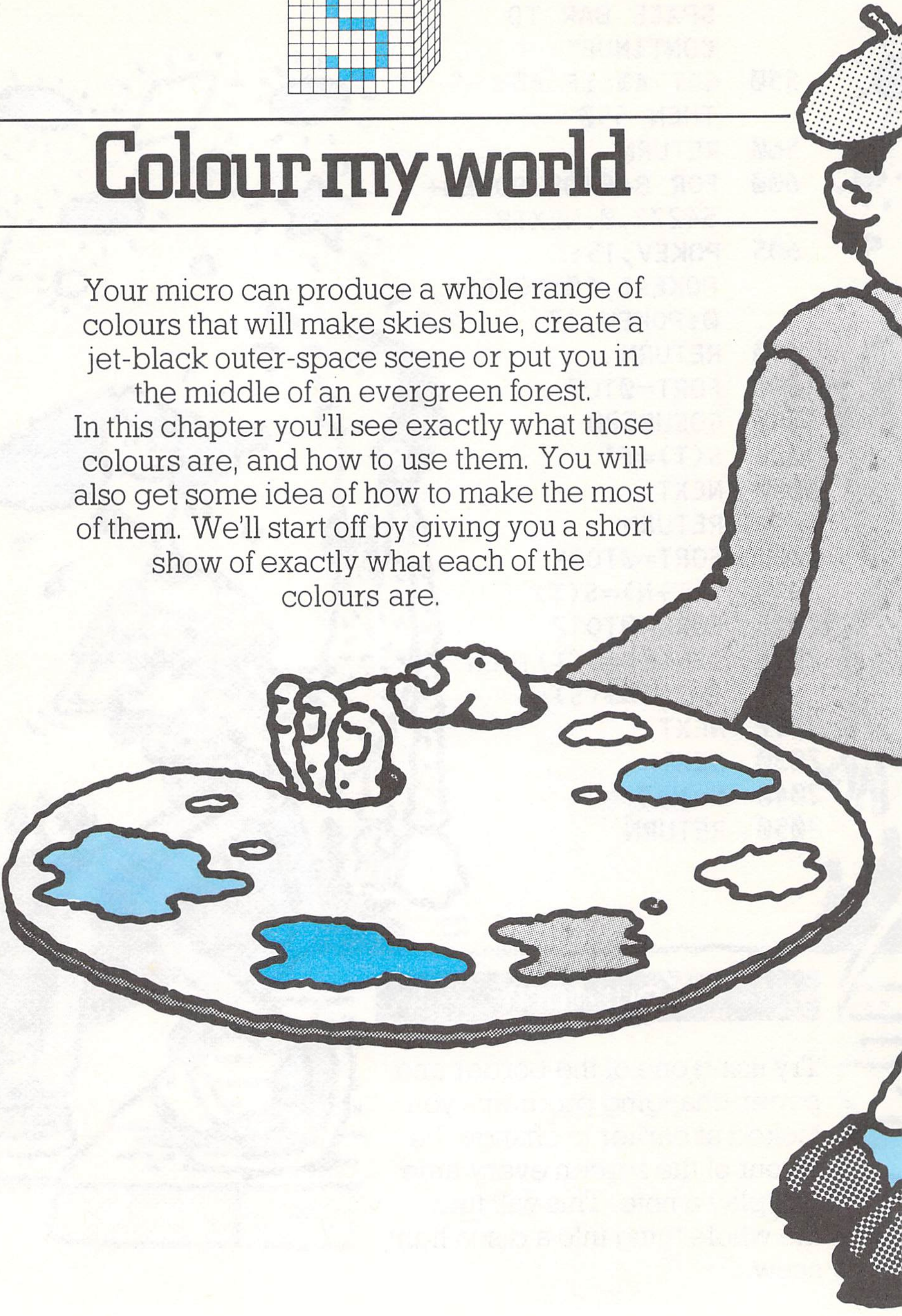
Your turn

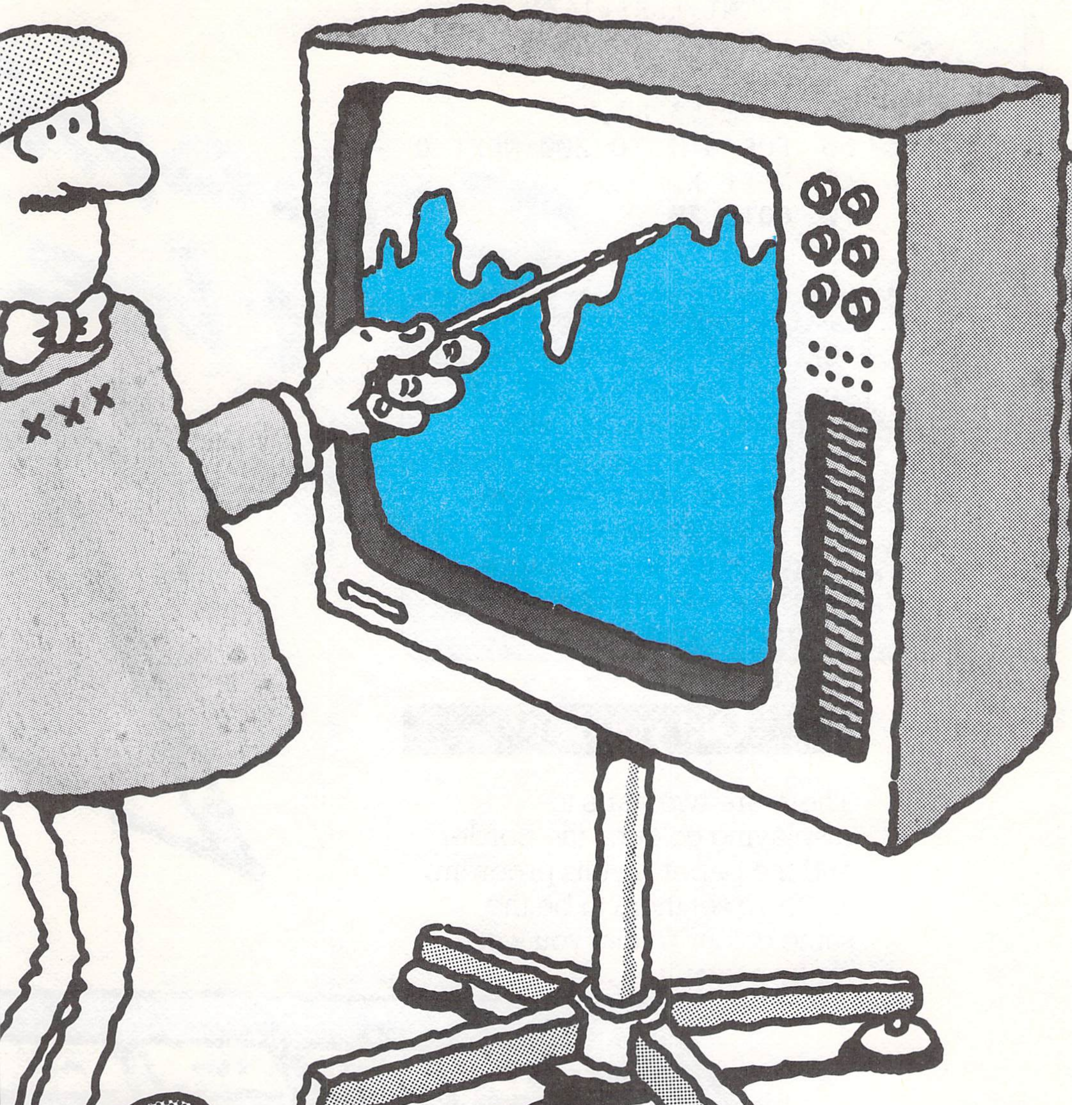
Try using one of the border and paper-changing programs you looked at earlier to change the colour of the screen every time you play a note. This will turn the whole thing into a disco light show.

5

Colour my world

Your micro can produce a whole range of colours that will make skies blue, create a jet-black outer-space scene or put you in the middle of an evergreen forest. In this chapter you'll see exactly what those colours are, and how to use them. You will also get some idea of how to make the most of them. We'll start off by giving you a short show of exactly what each of the colours are.





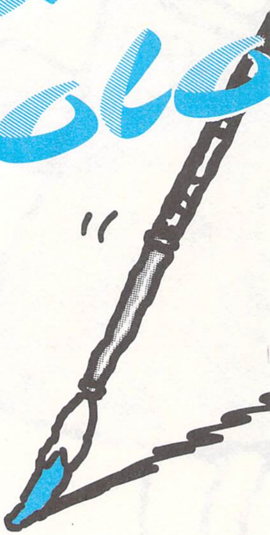
Try this

```
10  PRINT CHR$(147)
20  FOR X=1 TO 15
30  POKE 53281,X
40  POKE 53280,X
50  FOR Q=1 TO 200:NEXT Q
60  NEXT X
70  GOTO 10
```

Troopi the colo

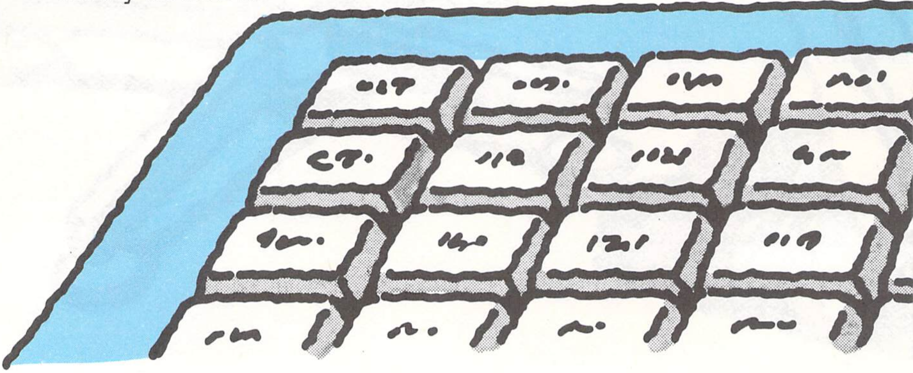
Remember

If the numbers in the FOR..NEXT loop are typed in incorrectly, and it tries to display the number for a colour that the micro doesn't have, the program will stop. So be careful when you're typing, otherwise the program won't work.



Did you see?

There are two parts to displaying colours: the border and the paper. In this program, we have set them to be the same colour so that you won't really notice it.



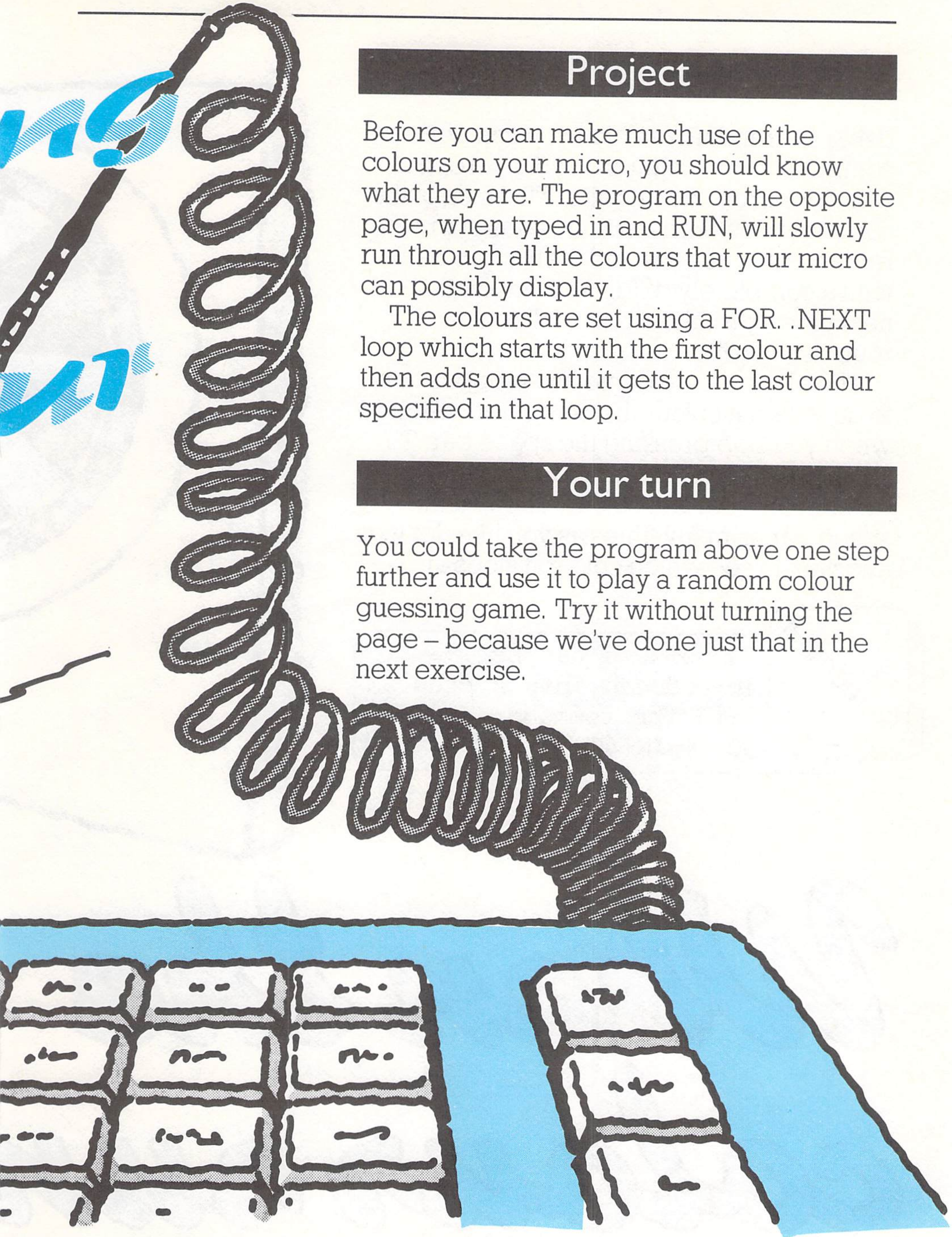
Project

Before you can make much use of the colours on your micro, you should know what they are. The program on the opposite page, when typed in and RUN, will slowly run through all the colours that your micro can possibly display.

The colours are set using a FOR. .NEXT loop which starts with the first colour and then adds one until it gets to the last colour specified in that loop.

Your turn

You could take the program above one step further and use it to play a random colour guessing game. Try it without turning the page – because we've done just that in the next exercise.



Project

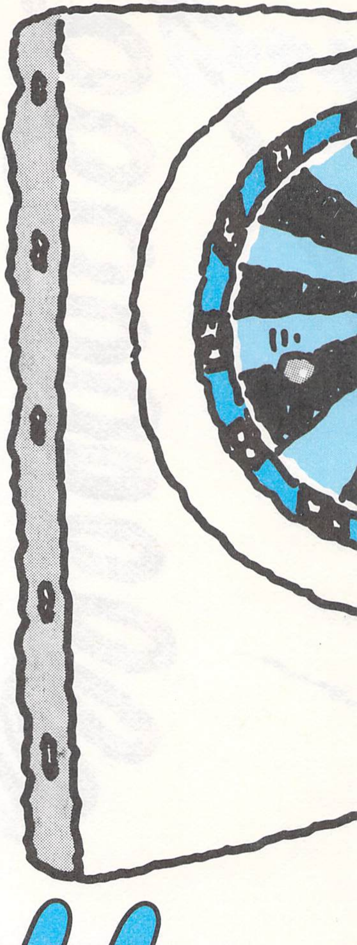
Using the random function you can get the computer to put colours on your screen in whatever order it feels like. This means the computer could first pick a series of green, red and blue colours and then come back with a completely different combination next time – just like a pair of dice or a roulette wheel.

The object of the game listed opposite is to guess what colour the computer will show when you stop pressing the space bar. To start with, have all the players make a guess or a bet on a certain colour. Then hit the space bar as many times as you like, let it go and wait to see which colour it stops at.

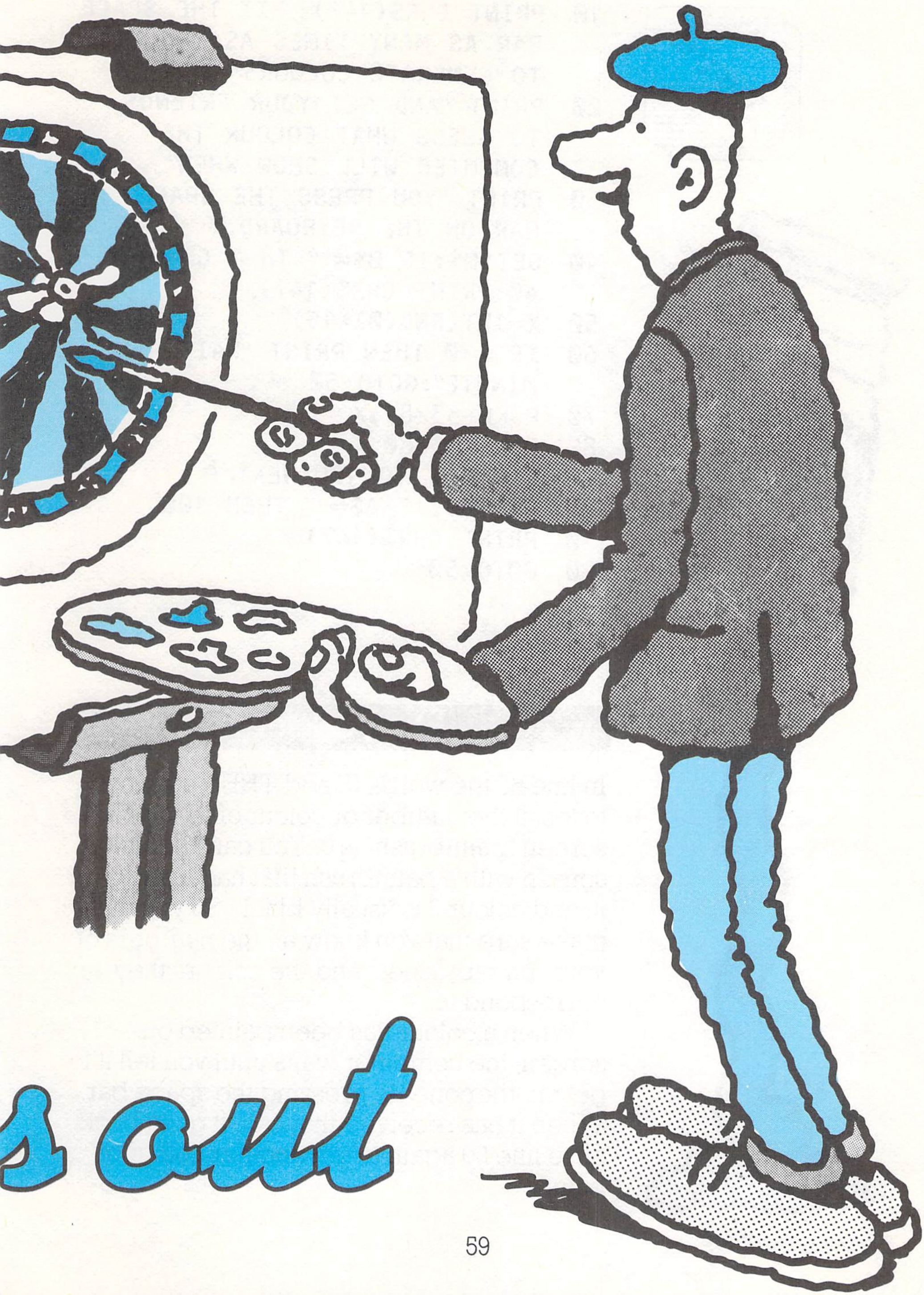
Remember

Randomness can be introduced into a program with the command:

```
50 X=INT(RND(0)*15)
```

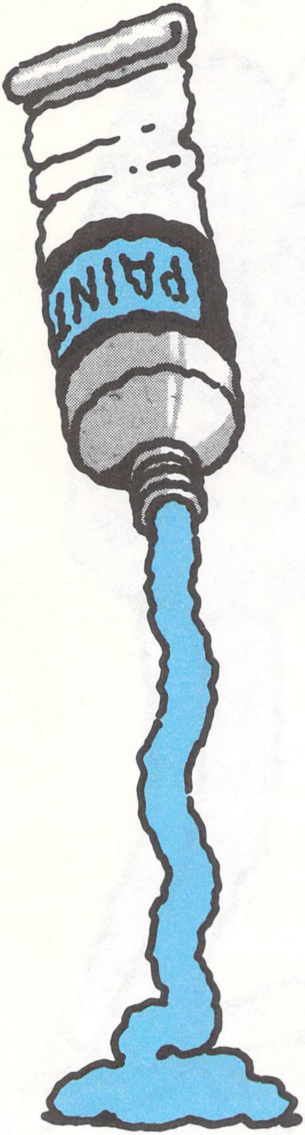


When the colour run


s out

Try this

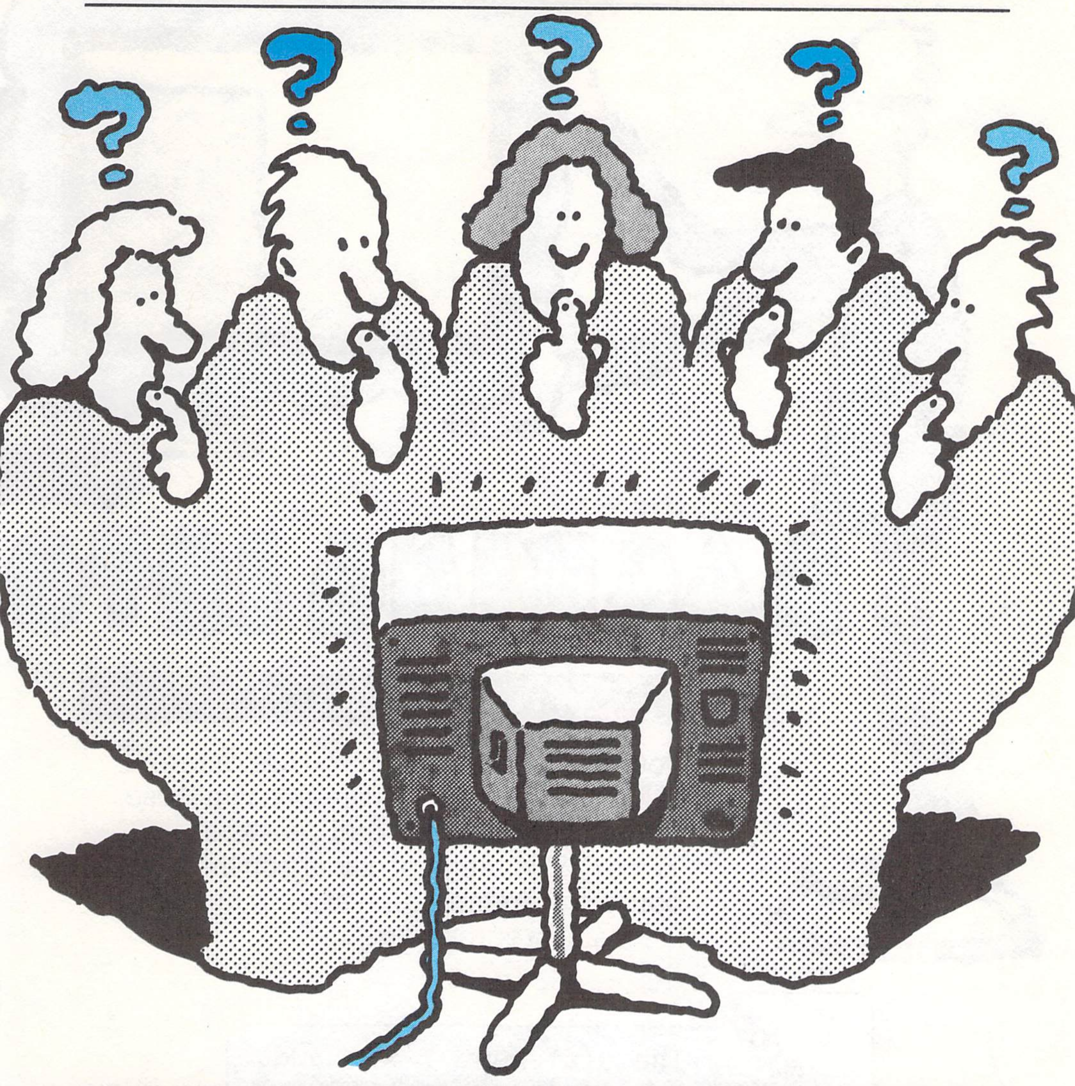
```
 10  PRINT CHR$(147);“HIT THE SPACE
     BAR AS MANY TIMES AS YOU LIKE
     TO GENERATE COLOURS”
 20  PRINT “AND GET YOUR FRIENDS
     TO GUESS WHAT COLOUR THE
     COMPUTER WILL SHOW WHEN”
 30  PRINT “YOU PRESS THE SPACE
     BAR ON THE KEYBOARD.”
 40  GET B$:IF B$=“” THEN GOTO
     40:PRINT CHR$(147)
 50  X=INT(RND(Ø)*15)
 60  IF X=Ø THEN PRINT “WAIT A
     MINUTE“:GOTO 5Ø
 70  POKE 53281,X
 80  POKE 5328Ø,X
 90  FOR Q=1 TO 1ØØ:NEXT Q
100  GET A$:IF A$=“” THEN 1ØØ
110  PRINT CHR$(147)
120  GOTO 5Ø
```



Did you see?

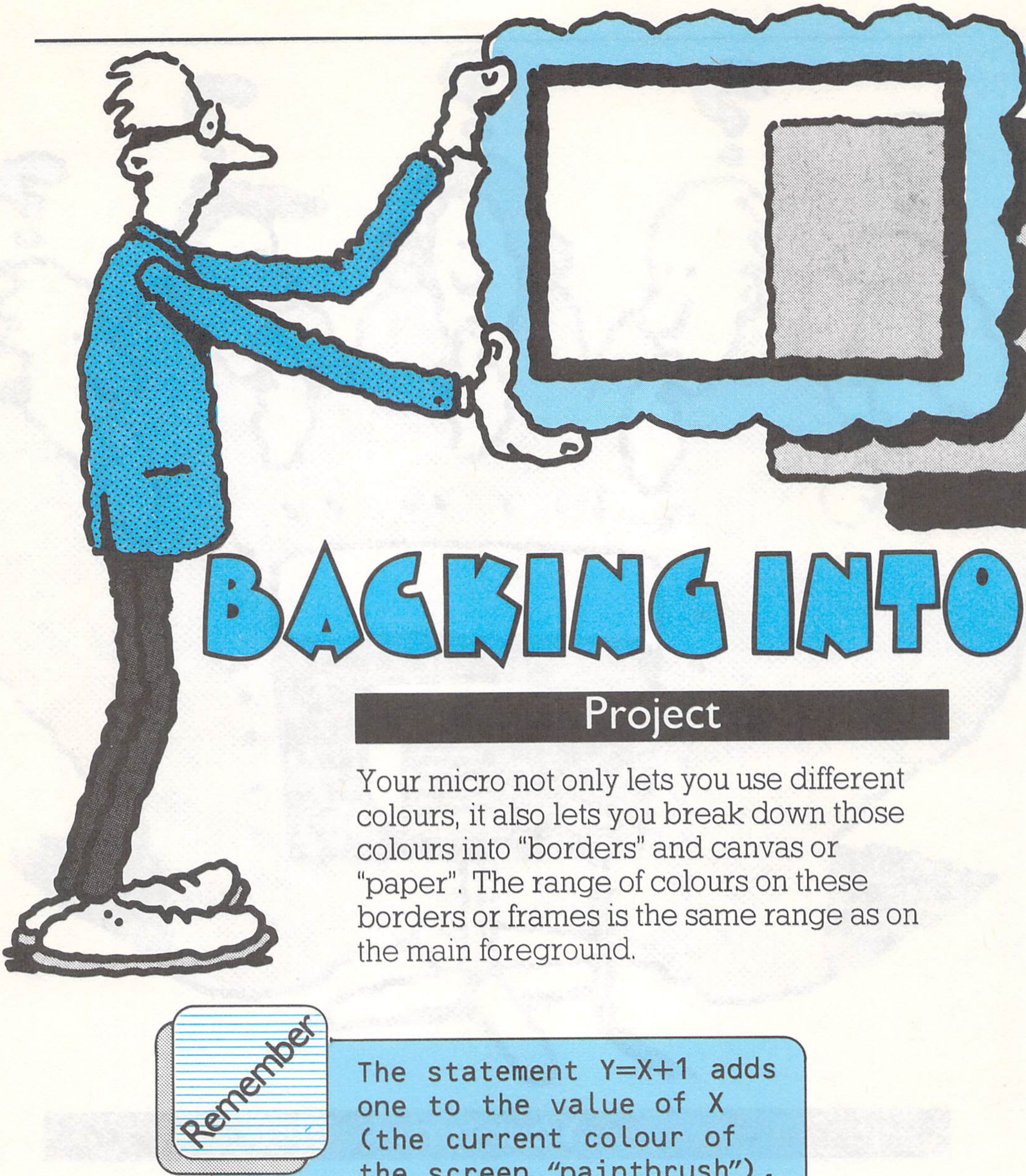
In line 60 the words IF and THEN are used to test if the number or colour of the on-screen "paintbrush" is 0. You can't paint a screen with a paintbrush that has nothing on it, and colour 0 is usually black. So you must make sure that you know all the numbers of your "paintbrushes" and the colours they correspond to.

When a colour has been painted on-screen, the computer waits until you tell it to get another one by pressing the space bar. When it has received that call, it goes back off to line 50 again to get another colour of paint.



Your turn

This program lets you play a little colour-guessing game with your family and friends. To run the program, just press the space key (or space bar) as many times as you like – and get your friends and family to guess what colour will be on-screen when you stop. To make the game more interesting, you could make the border and the paper two different colours.



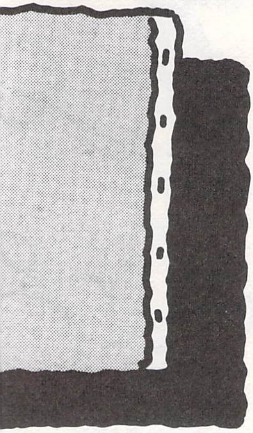
BACKING INTO

Project

Your micro not only lets you use different colours, it also lets you break down those colours into "borders" and canvas or "paper". The range of colours on these borders or frames is the same range as on the main foreground.

Remember

```
The statement Y=X+1 adds
one to the value of X
(the current colour of
the screen "paintbrush"),
so that background and
foreground colours will
be different.
```



Try this

```
10  PRINT CHR$(147)
20  PRINT "BACKGROUND AND
     FOREGROUND COLOURS
     CAN EITHER BE THE
     SAME OR DIFFERENT"
30  FOR U=1 TO 1500:
    NEXT:FOR X=1 TO 15
40  Y=X+1
50  POKE 53281,Y
60  POKE 53280,X
70  PRINT CHR$(147)
80  FOR Q=1 TO 100:NEXT Q
90  NEXT X
```

THE FOREGROUND

Did you see?

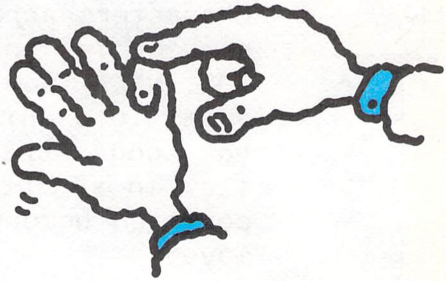
To keep the message displayed in line 20 on screen long enough for you to read it, there is a 'delay loop'. This makes the computer count all the way up to 1500 before it can go on to the rest of the program and get on with the painting.

Delay loops are formed using the words FOR and NEXT, with the number you want to count up to coming after the word TO. So to get the computer to count from 1 to 1000, you would write:

```
FOR U=1 TO 1000:NEXT
```

Your turn

By putting a few PRINT statements in the program containing the names of the colours, you could help to distinguish between them even if they were on a black and white TV.



MOVE IT

Project

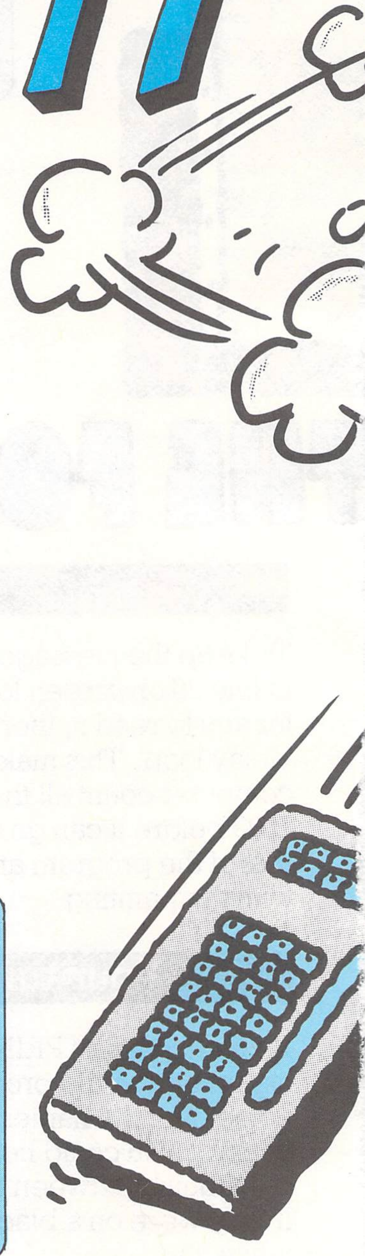
You know now how to set background and foreground colours (border and paper) and you have had a try at drawing. Now you are going to see how you can move things around on the backgrounds and foregrounds.

The project opposite doesn't move any special characters around (that will follow shortly). It takes some standard keyboard characters, and shows you how two of them can be made to move at once. The characters will move across and up the screen together, until they meet and pass one another.

Here is how it's done. You place a character on-screen somewhere, then place it one step further across the screen and print a blank space where it used to be.

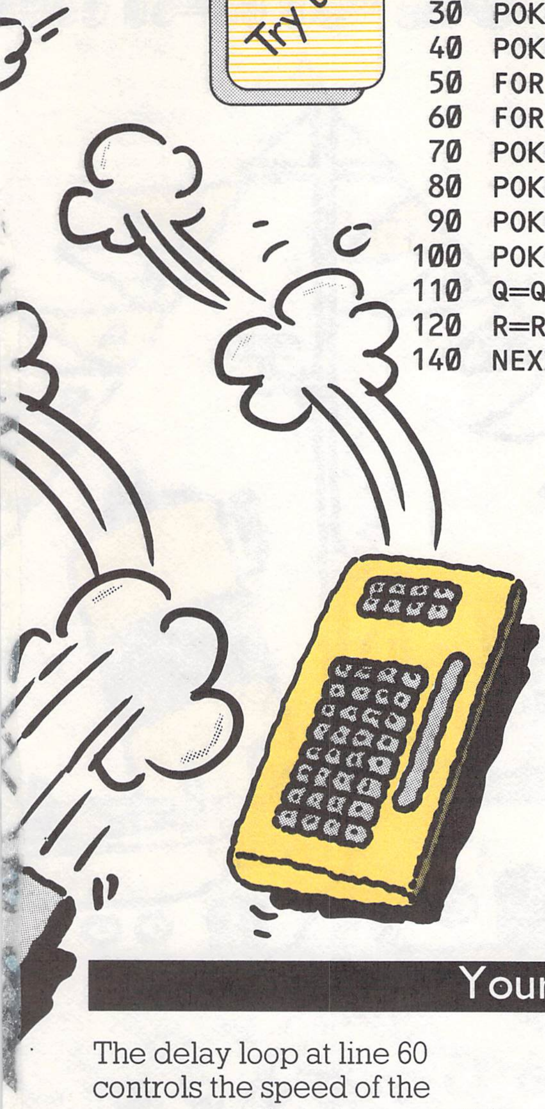
Remember

A FOR..NEXT loop controls the movement of the characters across the screen. Yet another FOR..NEXT loop provides a delay loop which times how long a character will stay in a particular position before it's moved.



Try this

```
 10  PRINT CHR$(147)
 20  Q=1224:R=2004
 30  POKE 53280,0
 40  POKE 53281,0
 50  FOR T=1 TO 27
 60  FOR D=1 TO 20:NEXT D
 70  POKE Q,32
 80  POKE Q+1,28
 90  POKE R,32
100  POKE R-40,30
110  Q=Q+1
120  R=R-40
140  NEXT T
```



Did you see?

At the start of the program, the initial screen positions of the two characters are set at line 20. If these numbers are changed then the characters will start moving from different places. It's rather like figuring out where you are starting from on a map, and where you want to go to.

Your turn

The delay loop at line 60 controls the speed of the movement. If you increase this loop to, say, 100 then the movement would be a good deal slower. The FOR. .NEXT loop here acts like the foot pedal on a car – the lower the number in the delay loop, the harder the pedal is pushed. Try changing this loop upwards and downwards to see what happens.

COLOURFUL CHARA

CTERS

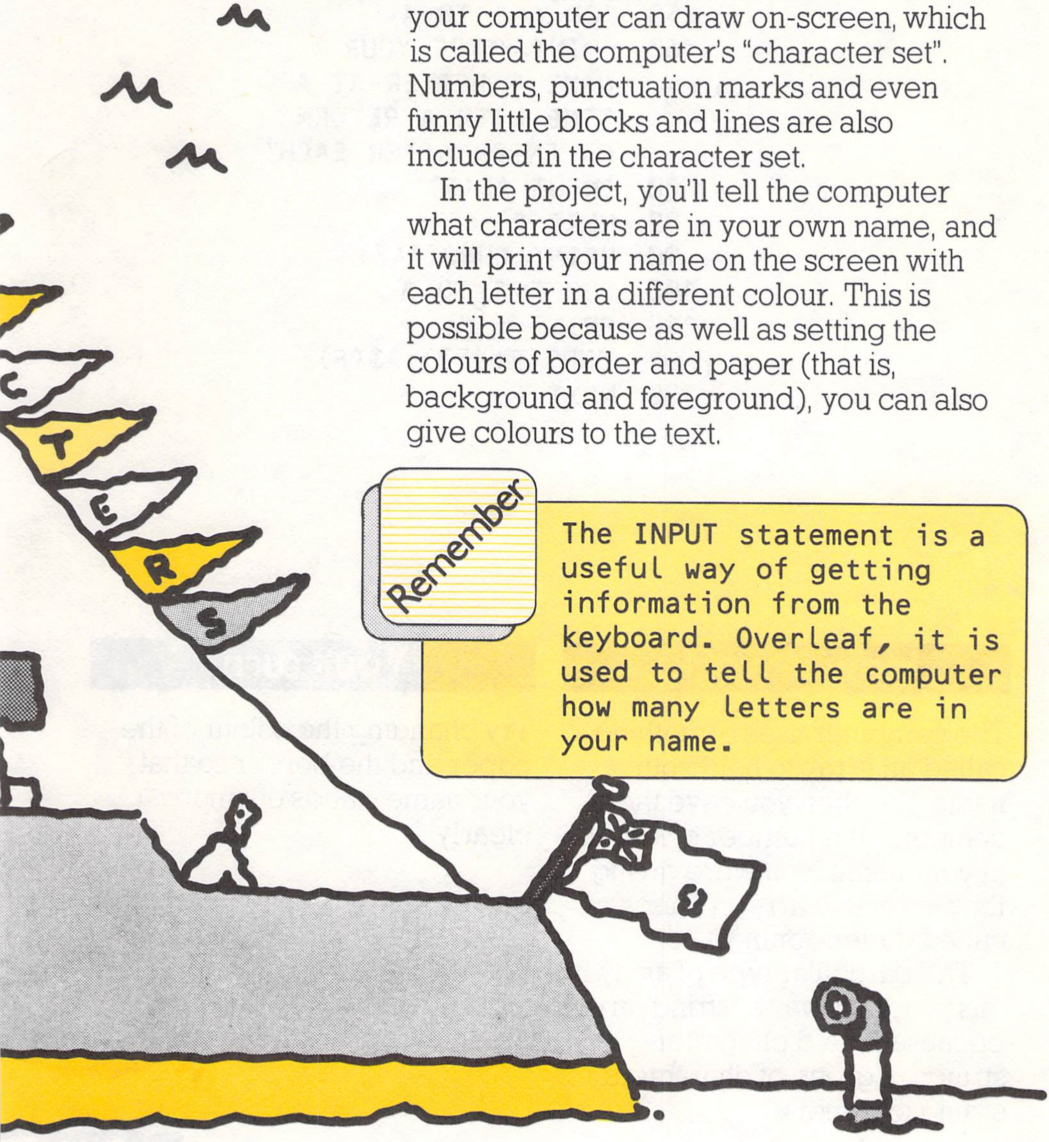
Project

Did you know that letters of the alphabet are sometimes called "characters" or "character strings"? Those alphabetical letters are just part of a series of things that your computer can draw on-screen, which is called the computer's "character set". Numbers, punctuation marks and even funny little blocks and lines are also included in the character set.

In the project, you'll tell the computer what characters are in your own name, and it will print your name on the screen with each letter in a different colour. This is possible because as well as setting the colours of border and paper (that is, background and foreground), you can also give colours to the text.

Remember

The INPUT statement is a useful way of getting information from the keyboard. Overleaf, it is used to tell the computer how many letters are in your name.



Try this

```
 10  PRINT CHR$(147)
 20  PRINT "HOW MANY
      LETTERS ARE IN YOUR
      FIRST NAME?"
 30  INPUT A
 40  DIM A$(A)
 50  FOR F=1 TO A
 60  PRINT "TYPE YOUR
      NAME 1 LETTER AT A
      TIME WITH A RETURN
      OR ENTER AFTER EACH"
 70  INPUT A$(F)
 80  NEXT F
 90  PRINT CHR$(147)
100  FOR F=1 TO A
110  PRINT " ";
      CHR$(F+152);A$(F);
120  NEXT F
```

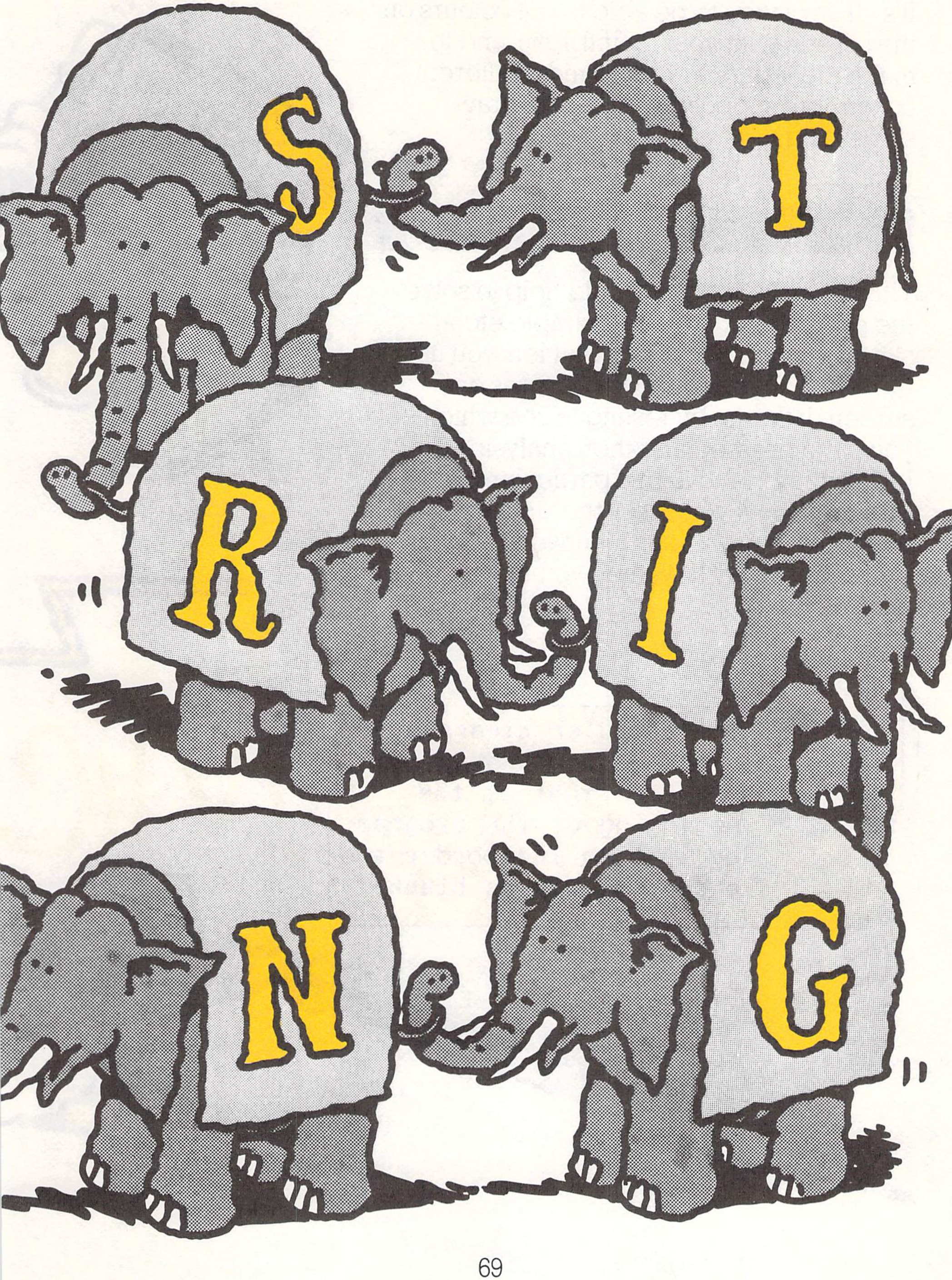
How it works

The computer uses something called an array to hold your name. So when you gave the computer the number of letters in your name, you were giving the size of that array. (This is called dimensioning.)

The particular type of array in this program was a "string" array because it held character strings (a group of characters strung together).

Your turn

Try changing the colour of the paper and the border so that your name stands out more clearly.



S
T
R
I
N
G

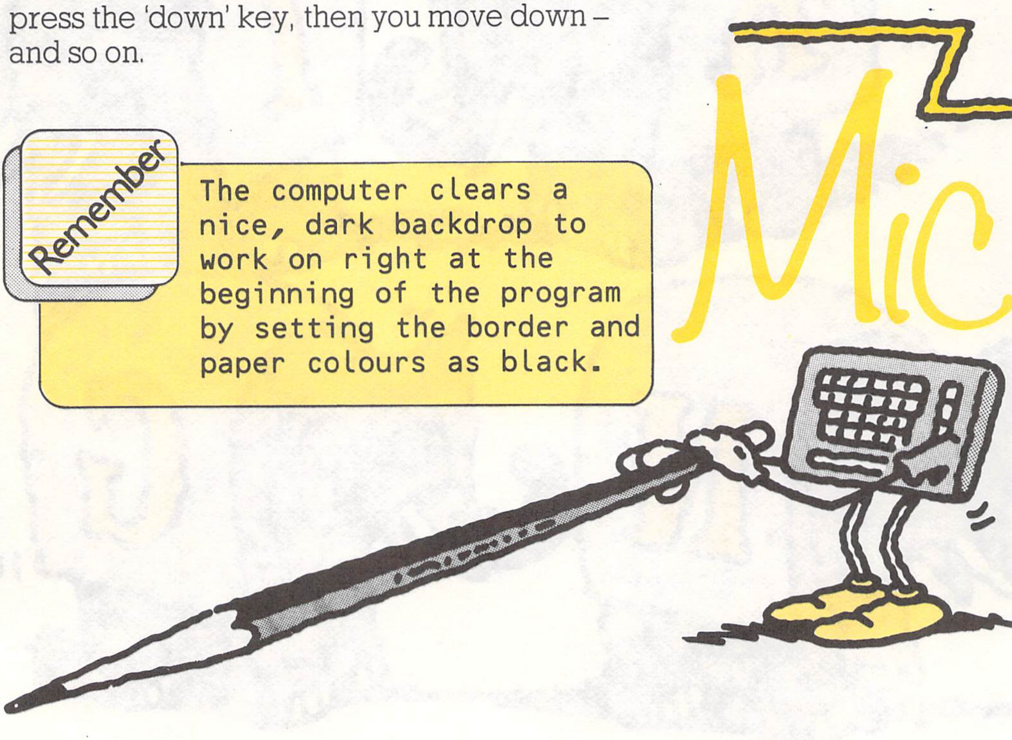
It's all very well to be able to put colours on the screen with your 'paintbrush' and to make the letters in your name a different colour, but suppose you want to draw something?

Project

The project opposite should help to solve the problem for you. It's a simple etch-a-sketch drawing program that lets you use the micro to put your own pictures on screen. It works by testing to see which key you have pressed and then analysing that information to move in a particular direction. So if you press the 'up' key, the computer knows it has to move up. If you press the 'down' key, then you move down – and so on.

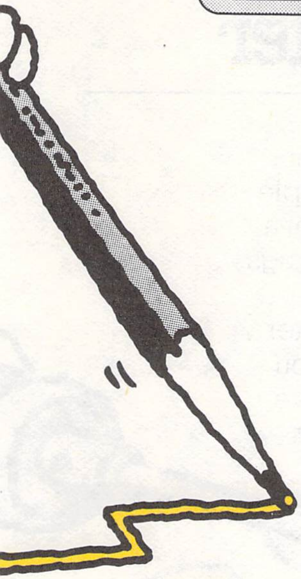
Remember

The computer clears a nice, dark backdrop to work on right at the beginning of the program by setting the border and paper colours as black.



Try this

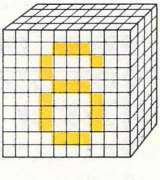
```
10  PRINT CHR$(147)
12  POKE 53280,0:POKE
    53281,0
15  Q=1484:C=55756
40  S=1
50  GET B$:IF B$="" THEN
    50
60  IF B$=CHR$(17) THEN
    Q=Q+40:C=C+40
70  IF B$=CHR$(145) THEN
    Q=Q-40:C=C-40
80  IF B$=CHR$(157) THEN
    Q=Q-1:C=C-1
90  IF B$=CHR$(29) THEN
    Q=Q+1:C=C+1
100 POKE Q,27
110 POKE C,S
120 S=INT(RND(1)*15)
130 GOTO 50
```



ro on the draw

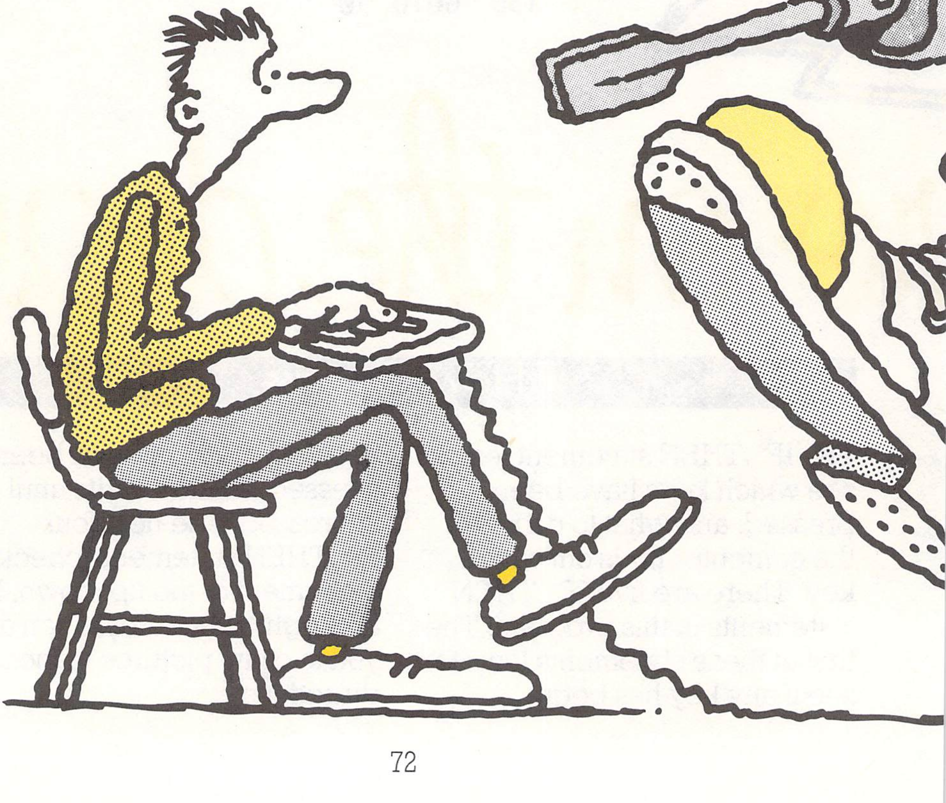
How it works

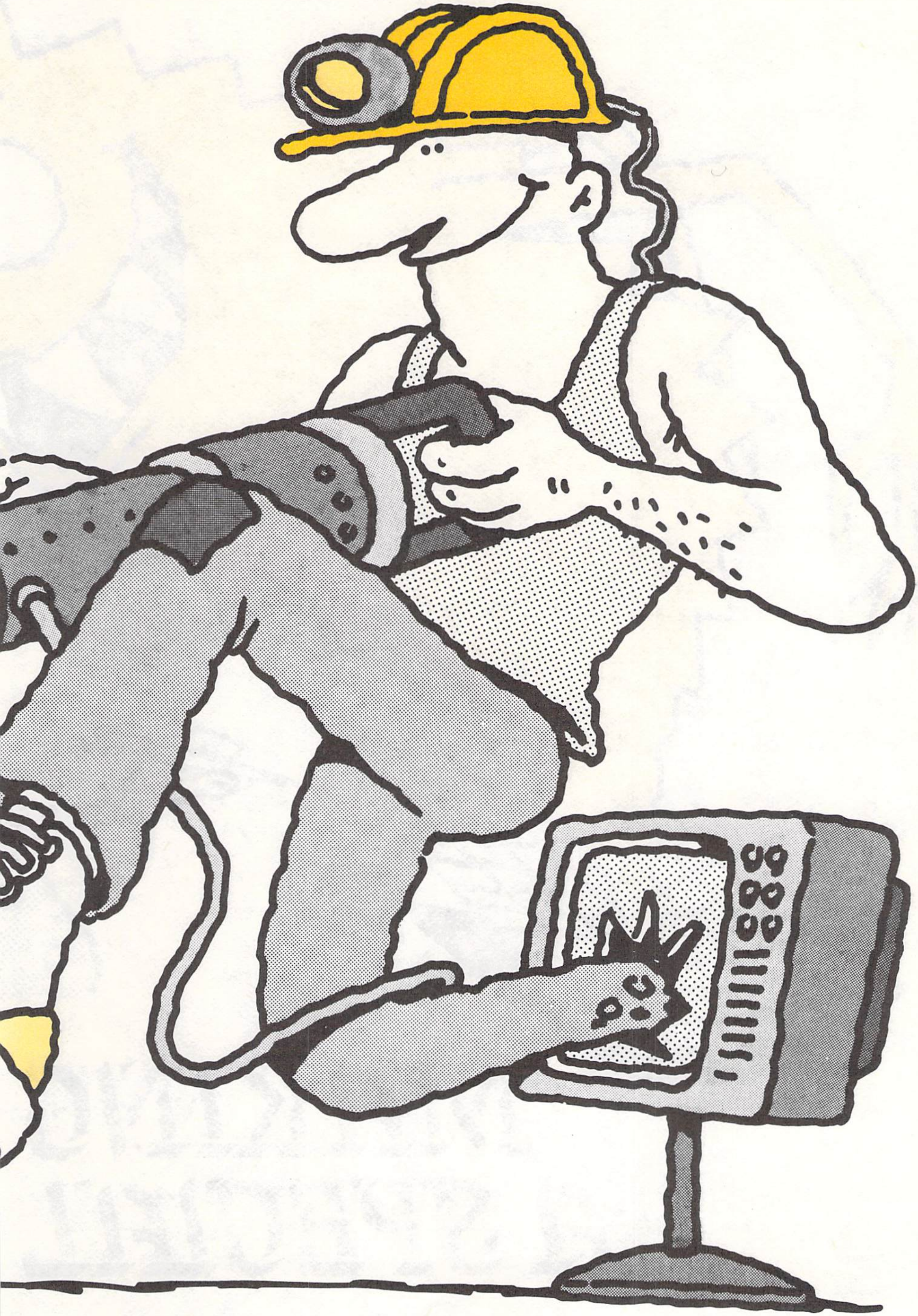
The IF. .THEN statement tests to see which keys have been pressed, and what to do when the computer finds out which key. There are five IF. .THEN statements in this program. The first of these statements tests to see if any key has been pressed. IF no key has been pressed THEN it waits until one is pressed. The next four IF. .THEN statements check for movement of the up, down, left and right cursor keys then allow you to draw pictures in those directions.

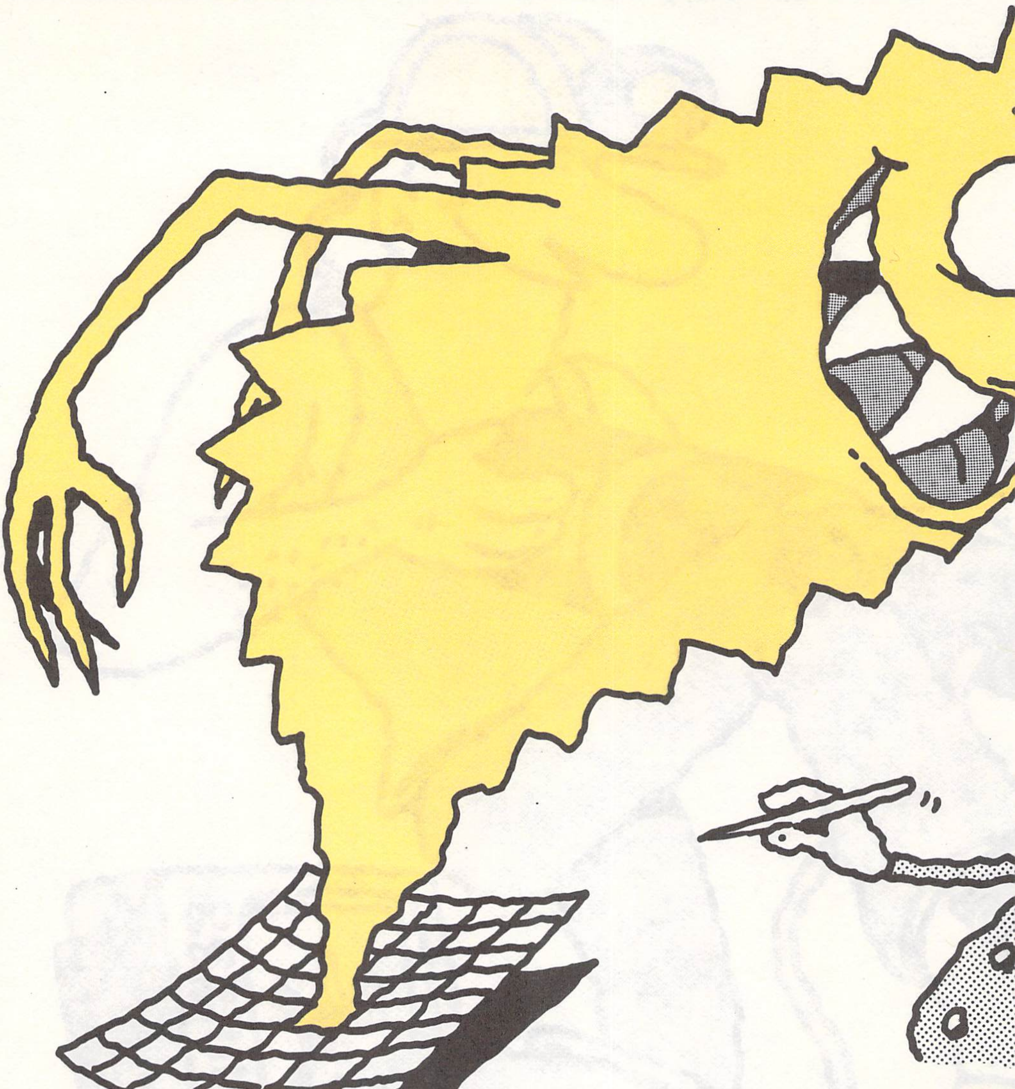


Getting in character

You know a little now about how to get colours on-screen and how to move simple objects around, and you are going to get a chance to do a few more complicated things with your computer's 'characters'. When you come to the end of this chapter there's a game – Morris Miner – that you can type in, using the skills you will have learned about developing characters.





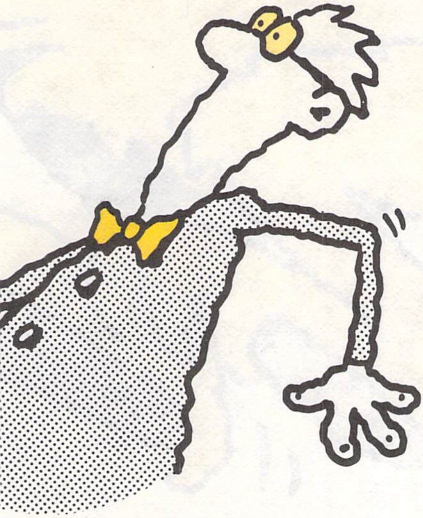


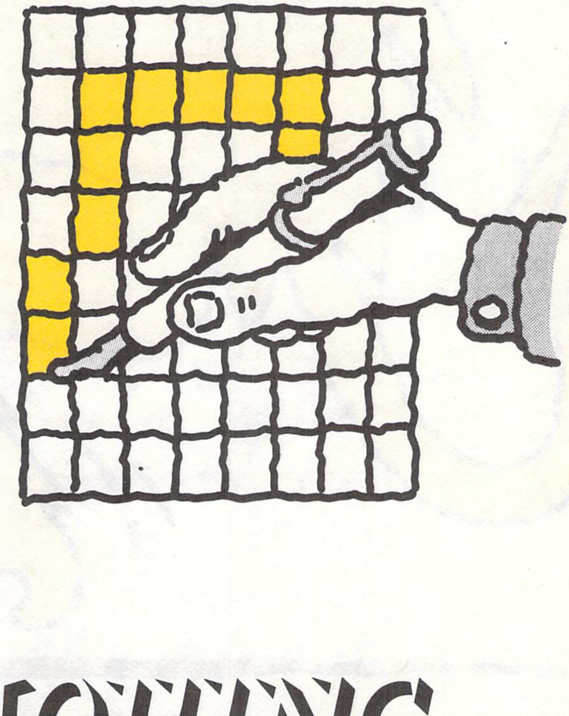
MAKING A SPECIAL

Project

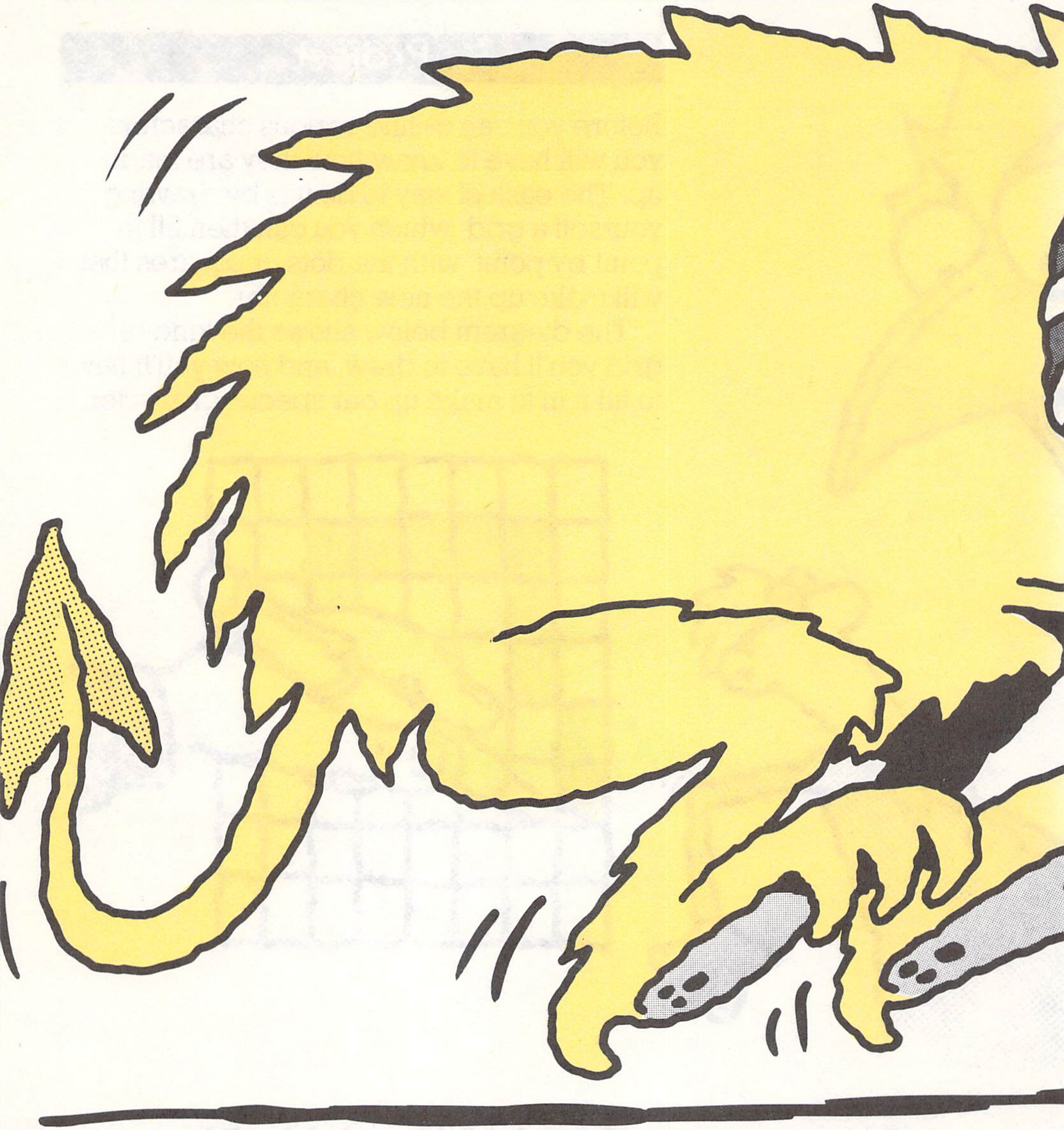
Before you can define various characters, you will have to know how they are made up. The easiest way to do it is by drawing yourself a grid, which you can then fill in, point by point, with the dots or squares that will make up the new character.

The diagram below shows the kind of grid you'll have to draw, and how you'll have to fill it in to make up our special character.



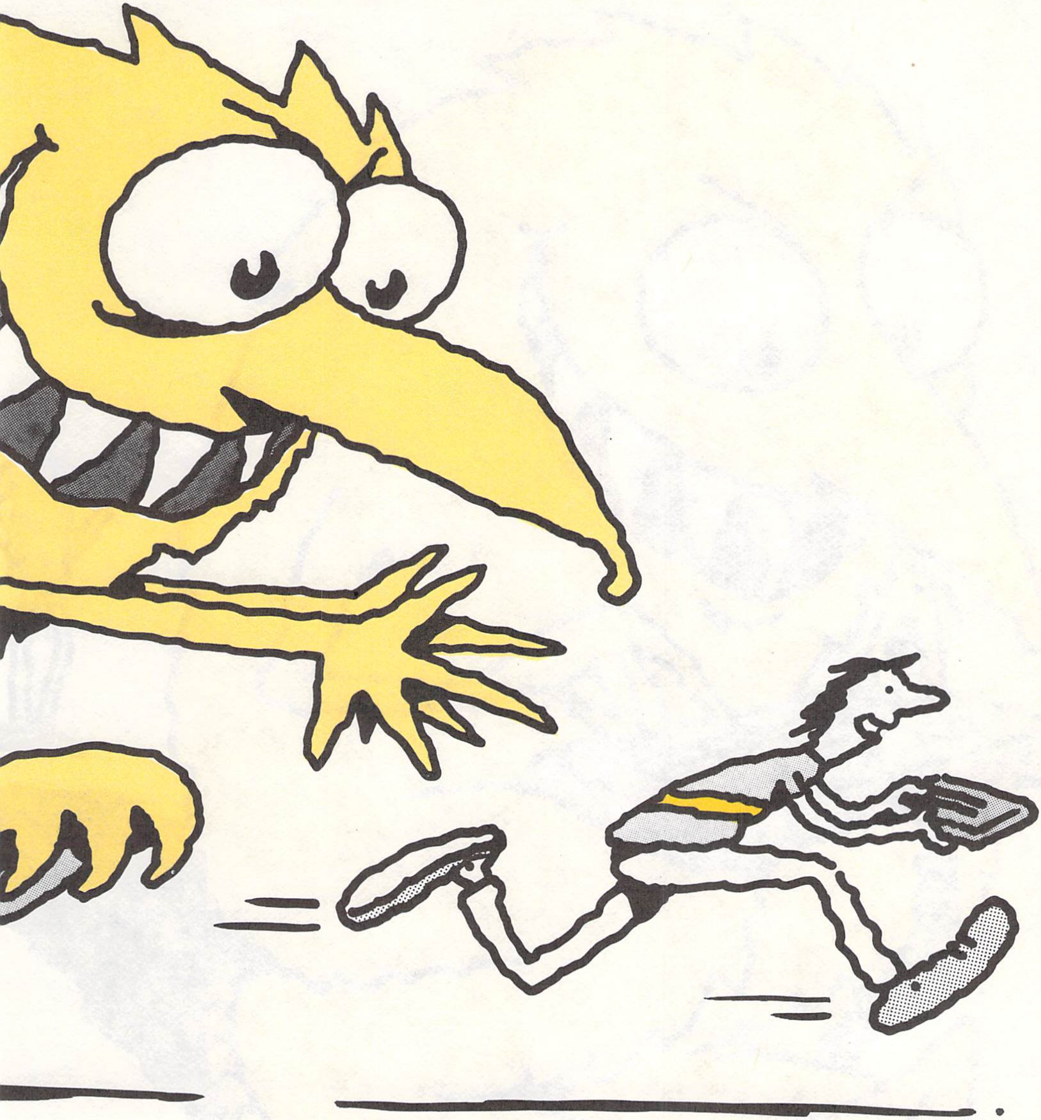


AND MOVING CHARACTER



The creature you're making is also known as a programmable character, and it is made up from an eight by eight matrix. That is, a group of eight 'bytes' arranged one on top of the other forms the matrix. Each byte in its turn is made up of eight bits – when a bit is one, there is a dot in it; when it is zero, there is a space in it.

First of all you are going to



enter a special character, and then you are going to move that character across the screen. In the next chapter, you'll move it across the screen in the same way as you moved your running man earlier. You will print a character at a certain position, then print a space where the character used to be, and then move the character forward a spot.

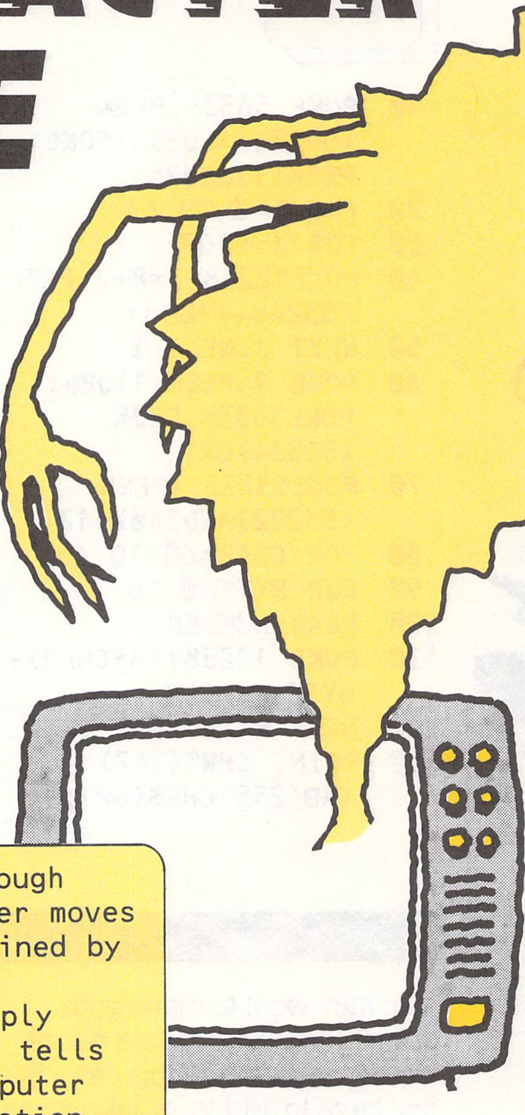
Try this

```
10  POKE 56334,PEEK
     (56334)AND254:POKE1,
     PEEK(1)and251
20  FOR I=0 TO 63
30  FOR J=0TO7
40  POKE12288+I*8+J,PEEK
     (53248+I*8+J)
50  NEXT J:NEXT I
60  POKE 1,PEEK(1)OR4:
     POKE56334,PEEK
     (56334)OR1
70  POKE53272,(PEEK
     (53272)AND240)+12
80  FOR CHAR=60 TO 63
90  FOR BYTE=0 TO 7
100 READ NUMBER
120 POKE 12288+(8*CHAR)+
     BYTE,NUMBER
140 NEXT BYTE:NEXT CHAR
150 PRINT CHR$(147)
     TAB(255)CHR$(60);
160 PRINT CHR$(61)
     TAB(55)CHR$(62)
     CHR$(63)
170 GET A$
180 IF A$="" THEN GOTO
     170
190 POKE 53272,21
200 DATA 7,7,7,1,1,31,
     3,3
210 DATA 224,224,224,
     128,128,248,
     192,192
220 DATA 1,2,4,8,16,32,
     64,128
230 DATA 128,64,32,16,8,
     4,2,1
240 END
```

Your turn

You may want to make your special character move either slower or faster. In this case, you just have to add to, or take away from, the number that the computer counts up to in the FOR. .NEXT loop that keeps the character on the screen for a period of time. You can also change the character itself by changing the data statements that make up each special character. That's what you're going to do in the next project.

A CHARACTER ON THE MOVE



Project

You'll now take the character we created in the last exercise, and start to move him across the screen in the way we suggested earlier. The character will start moving from the top left-hand corner to the top right-hand corner and stop when he gets to the edge of the screen.

Remember

The positions through which the character moves on screen are defined by what are known as "POKES", which simply means that a POKE tells you where the computer stores its information about where to put things on the screen.

Try this

```
 10 POKE 56334,PEEK
    (56334)AND254:POKE1,
    PEEK(1)AND251
 20 FOR I=0 TO 63
 30 FOR J=0TO7
 40 POKE12288+I*8+J,PEEK
    (53248+I*8+J)
 50 NEXT J:NEXT I
 60 POKE1,PEEK(1)OR4:
    POKE56334,PEEK(56334)
    OR1
 70 POKE53272,(PEEK
    (53272)AND240)+12
 80 FOR CHAR=60 TO 63
 90 FOR BYTE=0 TO 7
100 READ NUMBER
120 POKE 12288+(8*CHAR)
    +BYTE,NUMBER
140 NEXT BYTE:NEXT CHAR
145 FOR L=40 TO 78
150 PRINT CHR$(147)TAB
    (L)CHR$(60);
160 PRINT CHR$(61)TAB(L)
    CHR$(62)CHR$(63)
165 FOR H=1 TO 10:NEXT
    H:NEXT L
200 DATA 7,7,7,1,1,31,
    3,3
210 DATA 224,224,224,
    128,128,248,
    192,192
220 DATA 1,2,4,8,16,32,
    64,128
230 DATA 128,64,32,16,8,
    4,2,1
240 END
```

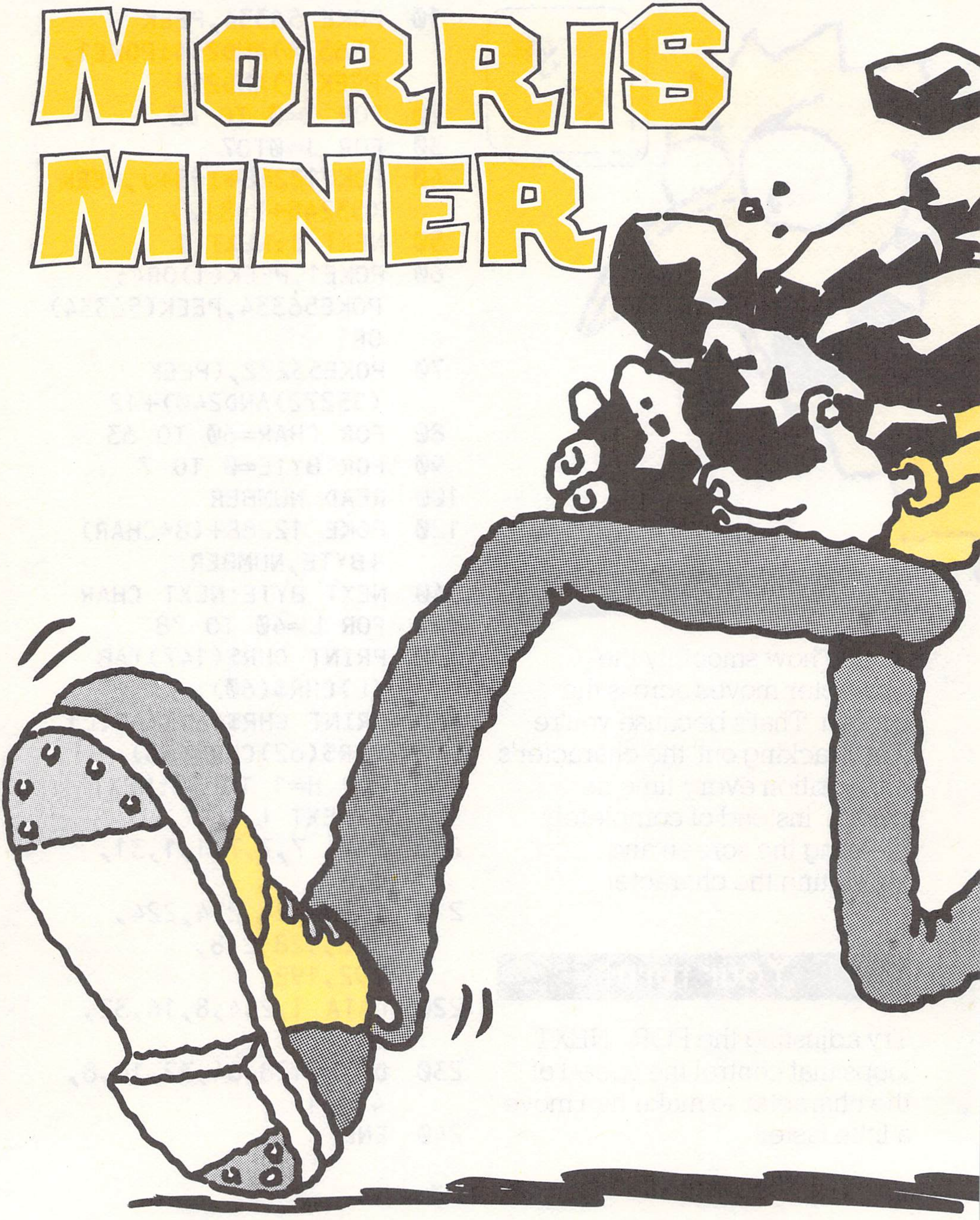
Did you see?

Notice how smoothly the character moves across the screen. That's because you're just 'blacking out' the character's old position every time he moves, instead of completely clearing the screen and reprinting the character.

Your turn

Try adjusting the FOR. .NEXT loops that control the speed of the character to make him move a little faster.

MORRIS MINER



Project

Don't take the power plug out of the computer just yet – you can get your character to play a game for you.

Did you know that your character's real name is Morris and that he works in a coal mine? His job is to rush around the mine shaft, picking up pieces of coal.

Because of union rules, he is only allowed to take 125 steps for every ten pieces of coal he picks up. The computer will count the number of steps he takes and print the total on the screen (don't worry about steering him over the printed total, the numbers won't be erased). When the 125 steps are reached the shift is over.

If you and Morris haven't picked up all the coal, then you lose. You also lose if you run him into the wall of the mine in the rush! The program should 'break' out if this happens.

Remember

You have to use the cursor keys to control the movement of Morris. He will appear on screen as soon as you RUN the program and then press a cursor key.

Holding down the cursor key will make him move faster. But be careful – if he moves *too* fast you'll miss the bits of coal and lose the game.

Try this

```
 10  POKE 56334,PEEK
     (56334)AND254:
     POKE1,PEEK(1)AND251
 20  FOR I=0 TO 63
 30  FOR J=0TO7
 40  POKE12288+I*8+J,
     PEEK(53248+I*8+J)
 50  NEXT J:NEXT I
 60  POKE1,PEEK(1)OR4:
     POKE56334,PEEK
     (56334)OR1
 70  POKE53272,(PEEK
     (53272)AND240)+12
 80  FOR CHAR=60TO63
 90  FOR BYTE=0TO7
100  READ NUMBER
110  POKE 12288+(8*CHAR)
     +BYTE,NUMBER
120  NEXT BYTE:NEXT
     CHAR
130  FOR L=40 TO 78
140  PRINT CHR$(147)TAB
     (L)CHR$(60);
150  PRINT CHR$(61)TAB
     (L)CHR$(62)CHR$(63)
160  FOR H=1 TO 10:NEXT
     H:NEXT L
200  DATA 7,7,7,1,1,31,
     3,3
210  DATA 224,224,224,
     128,128,248,
     192,192
 220  DATA 1,2,4,8,16,
      32,64,128
 230  DATA 128,64,32,16,
      8,4,2,1
1000  PRINT CHR$(147)
1500  PRINT CHR$(5)
2000  Q=1484
2500  S=1
2600  FOR U=1 TO 12
      :X=INT(RND(1)*1000)
      +1024:POKE
      X,35:NEXT U
3000  FOR T=1 TO 125
3100  GET B$:IF B$=""
      THEN 3100
4000  IF B$=CHR$(17)
      THEN Q=Q+40:POKE
      Q-40,32:POKE
      Q-39,32:POKE
      Q,32:POKE Q+1,32
5000  IF B$=CHR$(145)
      THEN POKE Q,32:
      POKE Q+1,32:
      POKE Q+40,32:
      POKEQ+41,32:Q=Q-40
6000  IF B$=CHR$(157)
      THEN POKE Q,32:
      POKE Q+1,32:
      POKE Q+40,32:
      POKEQ+41,32:Q=Q-1
7000  IF B$=CHR$(29)
      THEN POKE Q,32:
      POKE Q+1,32:
      POKE Q+40,32:
      POKEQ+41,32:Q=Q+1
8000  POKE Q,60:POKE
      Q+1,61
8200  POKE Q+40,62:POKE
      Q=41,63
```

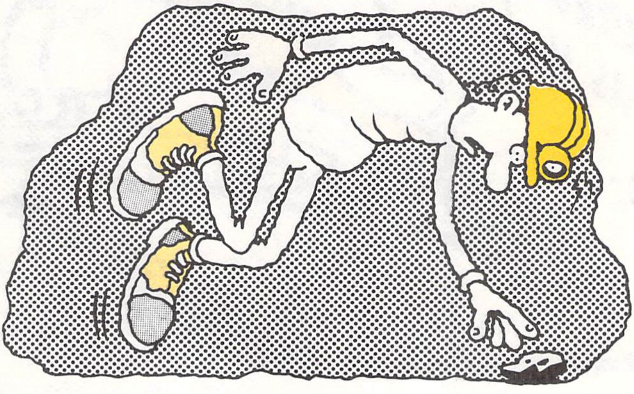
```
8800  S= INT(RND(1)*15)
9000  PRINT T:PRINT
       CHR$(19):PRINT
       CHR$(32):NEXT T
9500  PRINT CHR$(147)
9700  POKE 53272,21
10000 FOR G=1 TO 23
11000 PRINT
      "................
      TIME'S UP
      .............."
12000 NEXT G:GOTO 10000
```

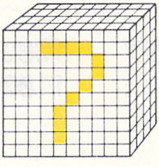
Did you see?

The computer knows that Morris has gone 'out of bounds'. If you try to go above or below certain areas on the screen, the program will stop working or 'bombout'. You could try to change this by putting an 'error trap' in the program which prevents you from going off the edge of the screen and bouncing back if you do.

Your turn

First an easy one: try decreasing the number of steps Morris can take. Change the value of T at line 3000. This will make the game more difficult (probably impossible). Or you can increase the value of 'U' at line 2600. This will produce more coal for Morris to pick up. Instead of Morris ending the game when he hits the wall of the mine, have him bounce back, and increase the value of 'T' by a set number as a penalty.

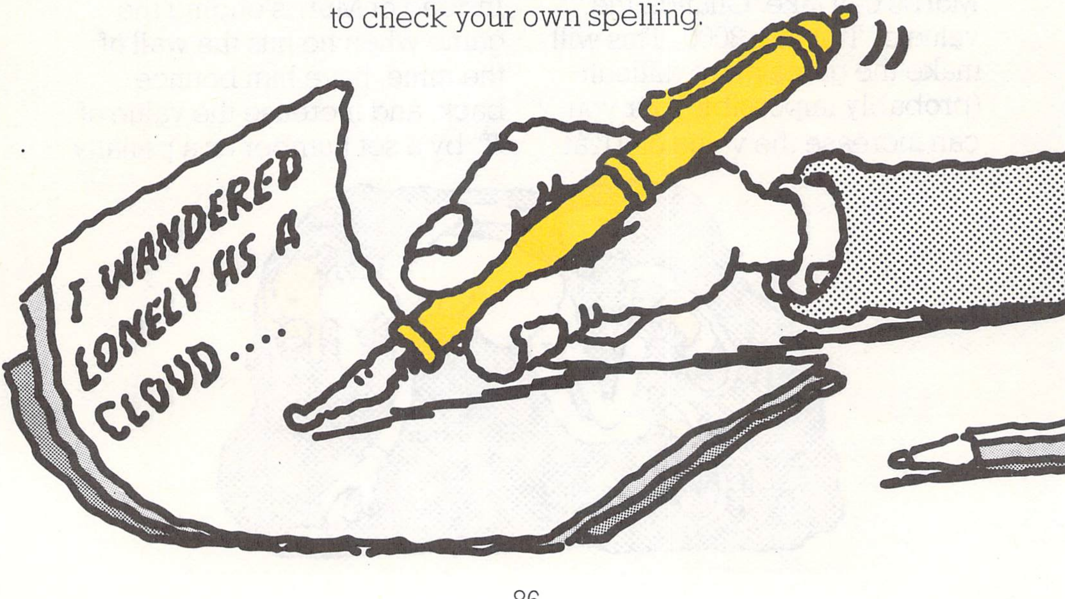


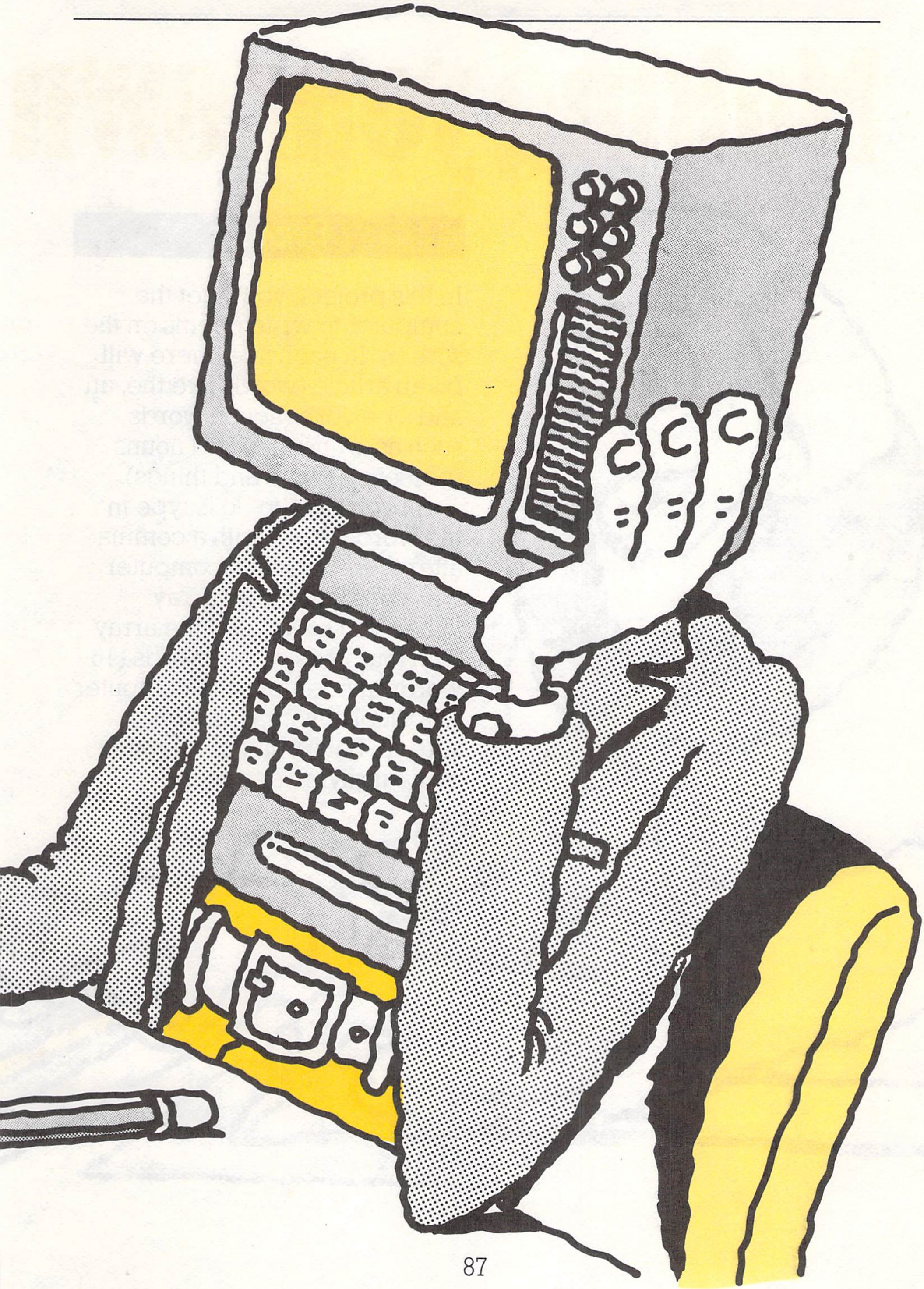


Stringing it together

Letters of the alphabet are accepted by your micro as 'strings'. This chapter will show you how you can use the computer to do things with those strings.

You can use the idea of 'string-matching' – that is, comparing one string to another to see if they're the same – in many computer programs. In the next chapter, we'll use that string-matching idea to help you to create some poetry, as well as to check your own spelling.



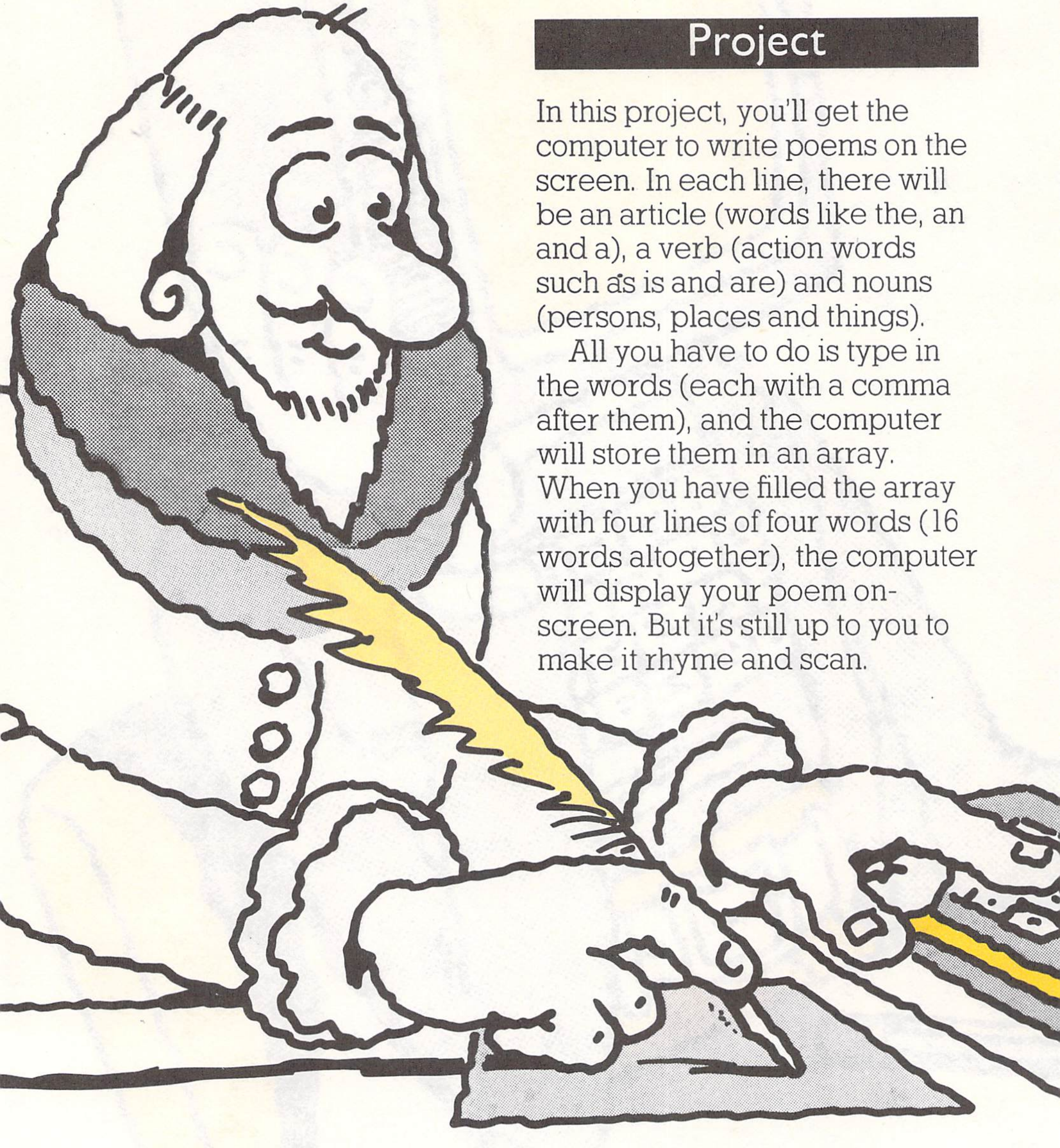


Making your own

Project

In this project, you'll get the computer to write poems on the screen. In each line, there will be an article (words like the, an and a), a verb (action words such as is and are) and nouns (persons, places and things).

All you have to do is type in the words (each with a comma after them), and the computer will store them in an array. When you have filled the array with four lines of four words (16 words altogether), the computer will display your poem on-screen. But it's still up to you to make it rhyme and scan.



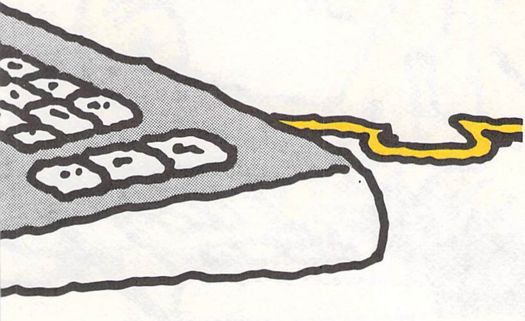
poem

Remember

There are four arrays in this program, each with a maximum of sixteen elements. The program, as it stands, only makes use of the first four elements in each of the arrays. You would only need to change the FOR..NEXT loops that get information from the keyboard however to extend the poem to sixteen lines.

Try this

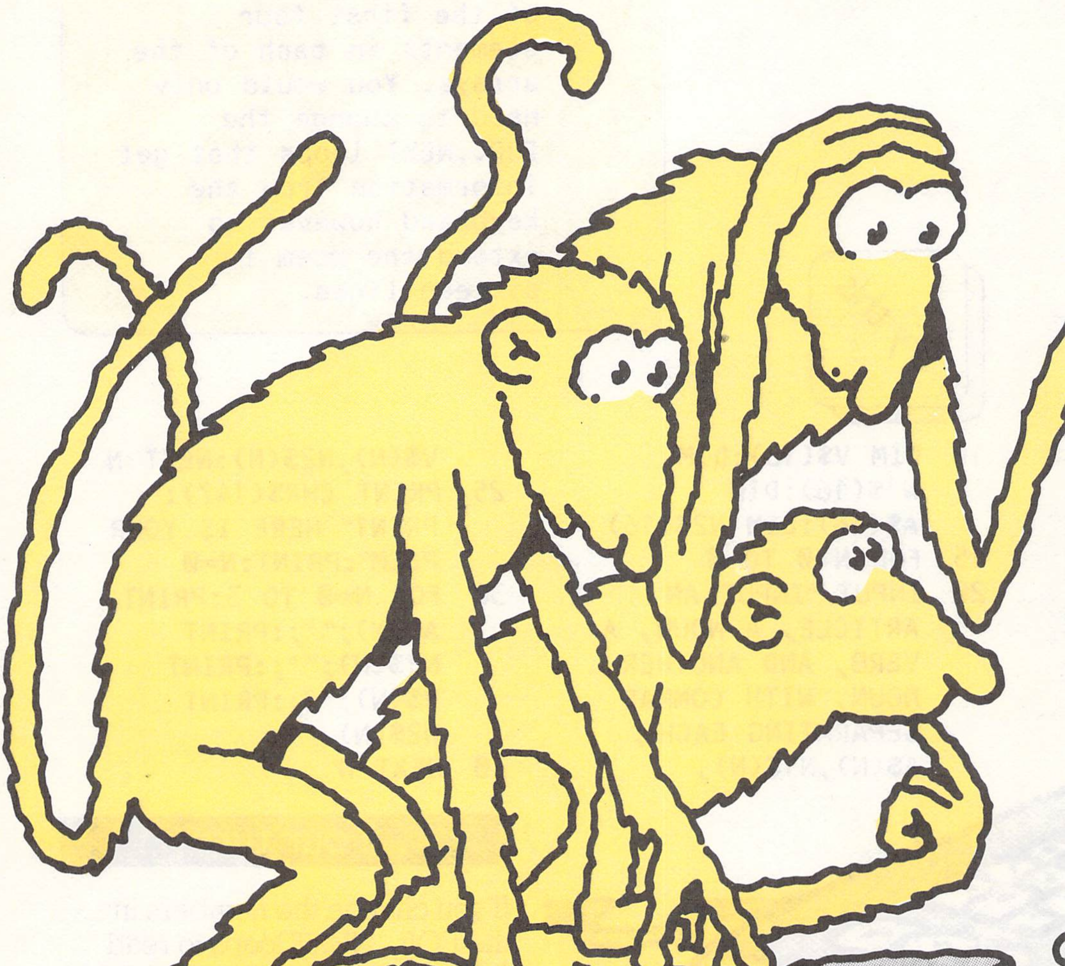
```
10  DIM V$(16):DIM
    N1$(16):DIM
    A$(16):DIM N2$(16)
15  FOR N=Ø TO 3
2Ø  INPUT "INPUT AN
    ARTICLE, A NOUN, A
    VERB, AND ANOTHER
    NOUN, WITH COMMAS
    SEPARATING EACH";
    A$(N),N1$(N),
    V$(N),N2$(N):NEXT N
25  PRINT CHR$(147):
    PRINT" HERE IS YOUR
    POEM":PRINT:N=Ø
3Ø  FOR N=Ø TO 3:PRINT
    A$(N);" ";:PRINT
    N1$(N);" ";:PRINT
    V$(N);" ";:PRINT
    N2$(N)
4Ø  NEXT N
```



Your turn

If you change the numbers in the FOR. .NEXT loops to read FOR N=0 TO 15 instead of FOR N=0 TO 3, then you can create a poem of up to 16 lines (because there are sixteen elements between 0 and 15).

Automatic wr

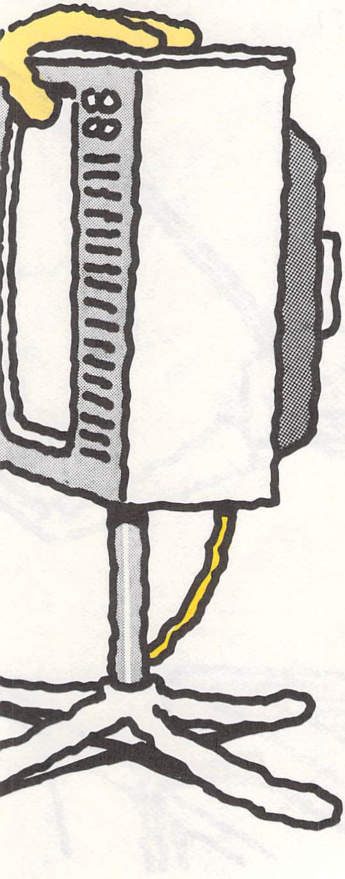


iting

Project

It has often been said that literature and writing is simply a matter of getting words in the right order. It has also been said that if a bunch of monkeys were locked up together with the appropriate writing materials for long enough, they might produce Shakespeare.

Well, in this project we try to prove that. First of all, you are going to give the computer a bunch of words that it can use to make up a poem. The computer will then shuffle those words around and put them in some sort of order. For the moment, the computer still tries to put the words in an order that takes account of whether a word is a verb, noun or an article. But if you wanted to, you could structure the poem far more loosely and let the computer put the words in any order it liked.

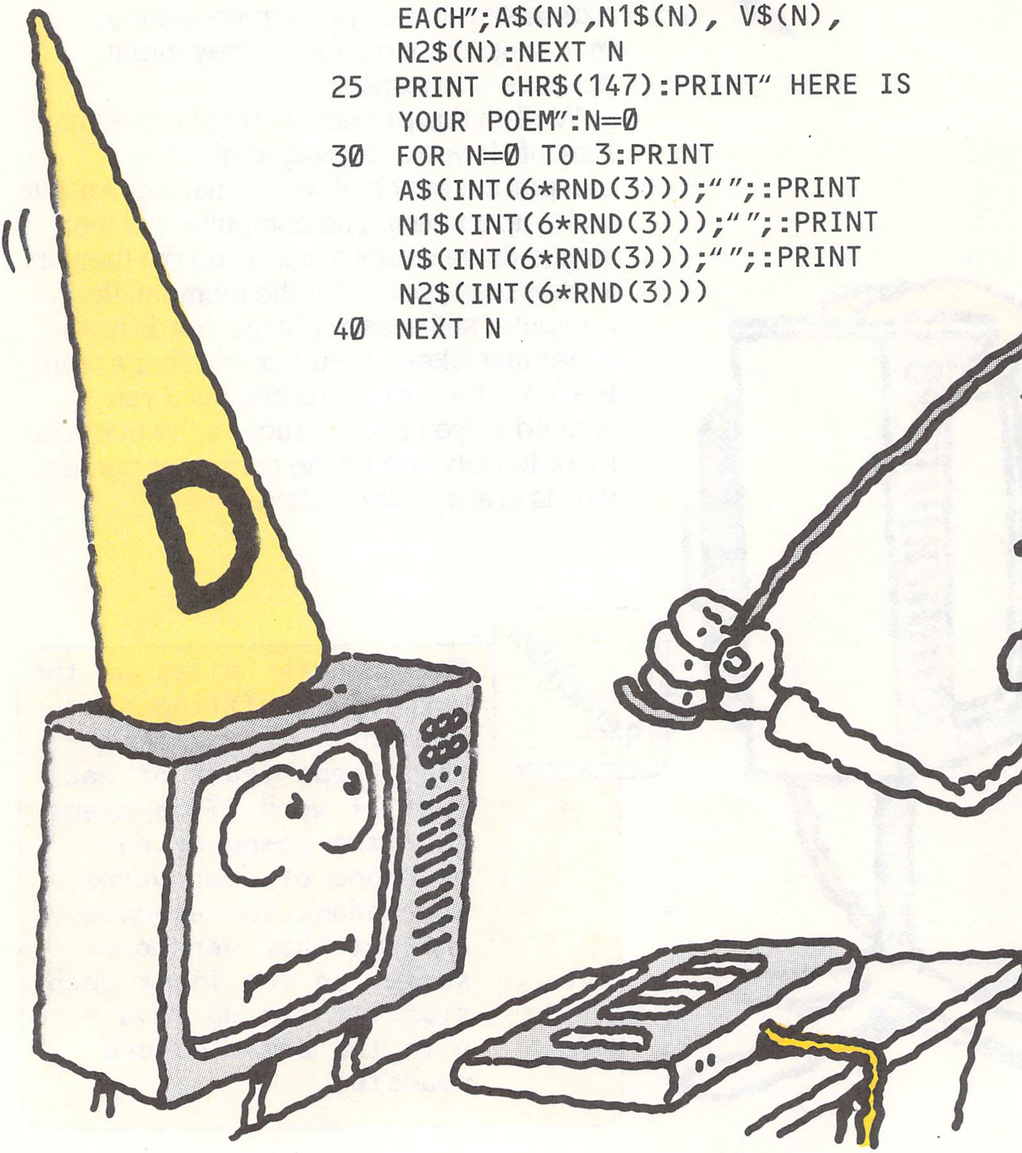


Remember

The computer mixes up the words by shuffling the 'elements' in the 'array' that keeps track of each type of word. This means that the computer can pick one of four verbs at random, for example, and put that verb on-screen in the first verb slot. It can do this with the articles and nouns too.

Try this

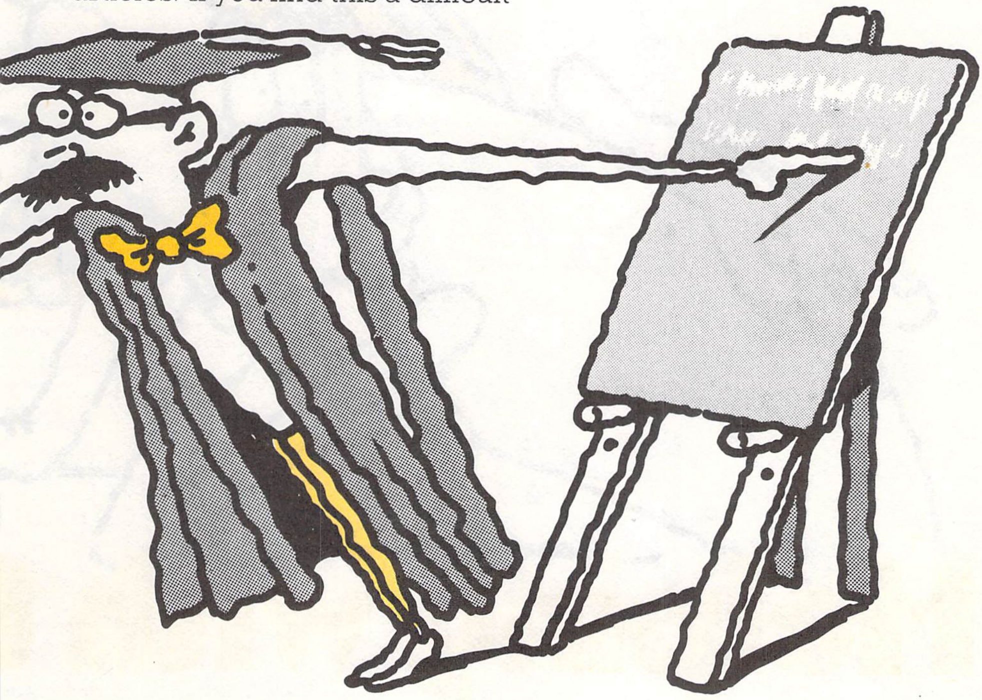
```
10  DIM V$(16):DIM N1$(16):DIM
     A$(16):DIM N2$(16)
15  FOR N=0 TO 3
20  INPUT “INPUT AN ARTICLE, A
     NOUN, A VERB, AND ANOTHER
     NOUN, WITH COMMAS SEPARATING
     EACH”;A$(N),N1$(N), V$(N),
     N2$(N):NEXT N
25  PRINT CHR$(147):PRINT“ HERE IS
     YOUR POEM”:N=0
30  FOR N=0 TO 3:PRINT
     A$(INT(6*RND(3)));“ ”;:PRINT
     N1$(INT(6*RND(3)));“ ”;:PRINT
     V$(INT(6*RND(3)));“ ”;:PRINT
     N2$(INT(6*RND(3)))
40  NEXT N
```



Your turn

You could change this program in one of a number of ways. The first change – and probably the simplest – would be to 'dimension' or set aside one big array to store all kinds of words, and then get the computer to print words in random sentences on-screen. You could however start to apply certain English language rules to the 'elements' or words in the array, so that the computer would know what to do when it encountered certain adverbs, adjectives, nouns, verbs and articles. If you find this a difficult task, don't get too discouraged. Many expert computer programmers before now have tried to program whole roomfuls of computers to understand all the ins and outs of the English and other languages, and no one has succeeded particularly well yet.

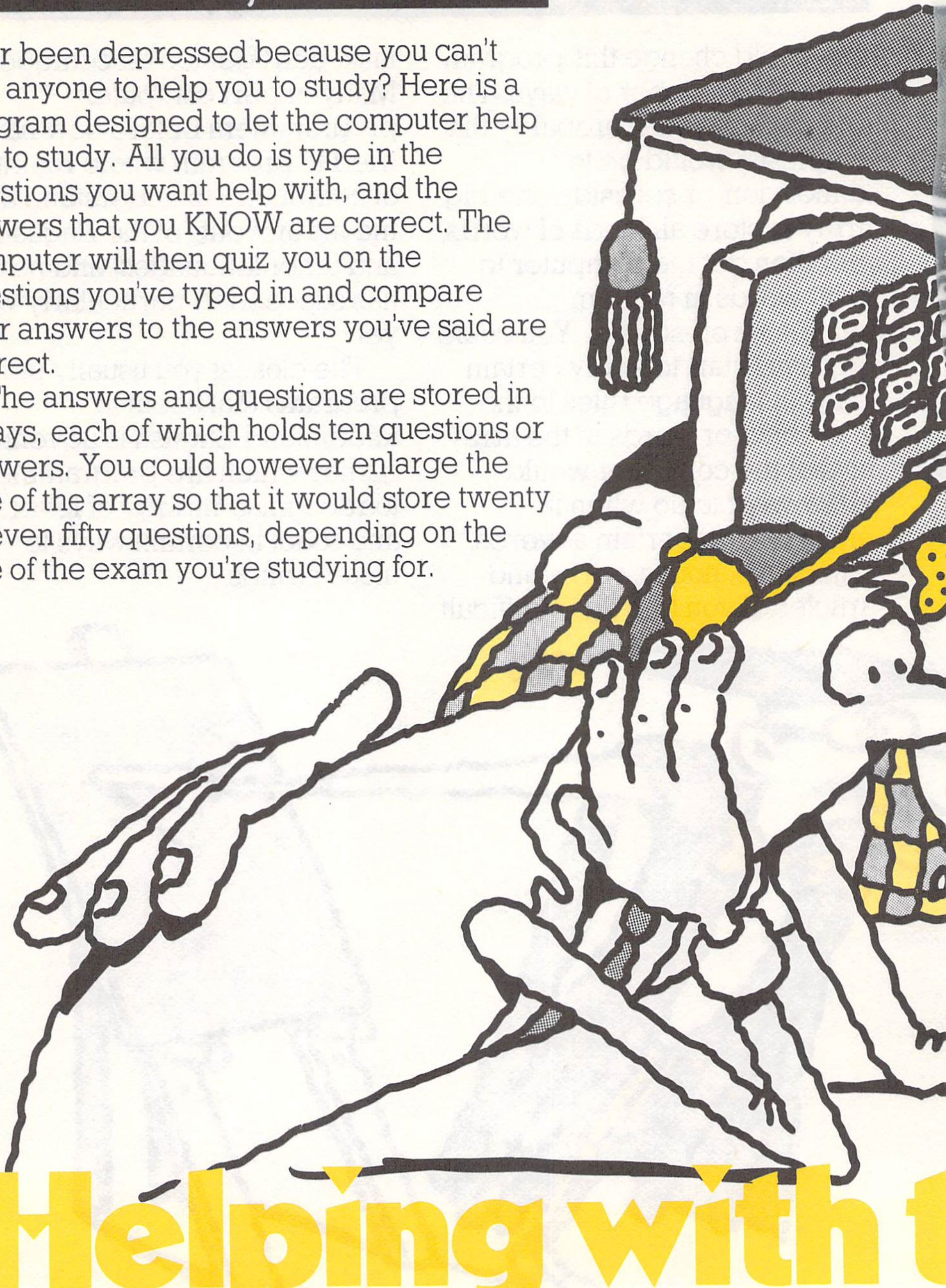
The closest you usually get to programs that seem to understand English is adventure games which are programmed to recognise 'strings' of letters and react in certain ways to those strings.



Project

Ever been depressed because you can't find anyone to help you to study? Here is a program designed to let the computer help you to study. All you do is type in the questions you want help with, and the answers that you KNOW are correct. The computer will then quiz you on the questions you've typed in and compare your answers to the answers you've said are correct.

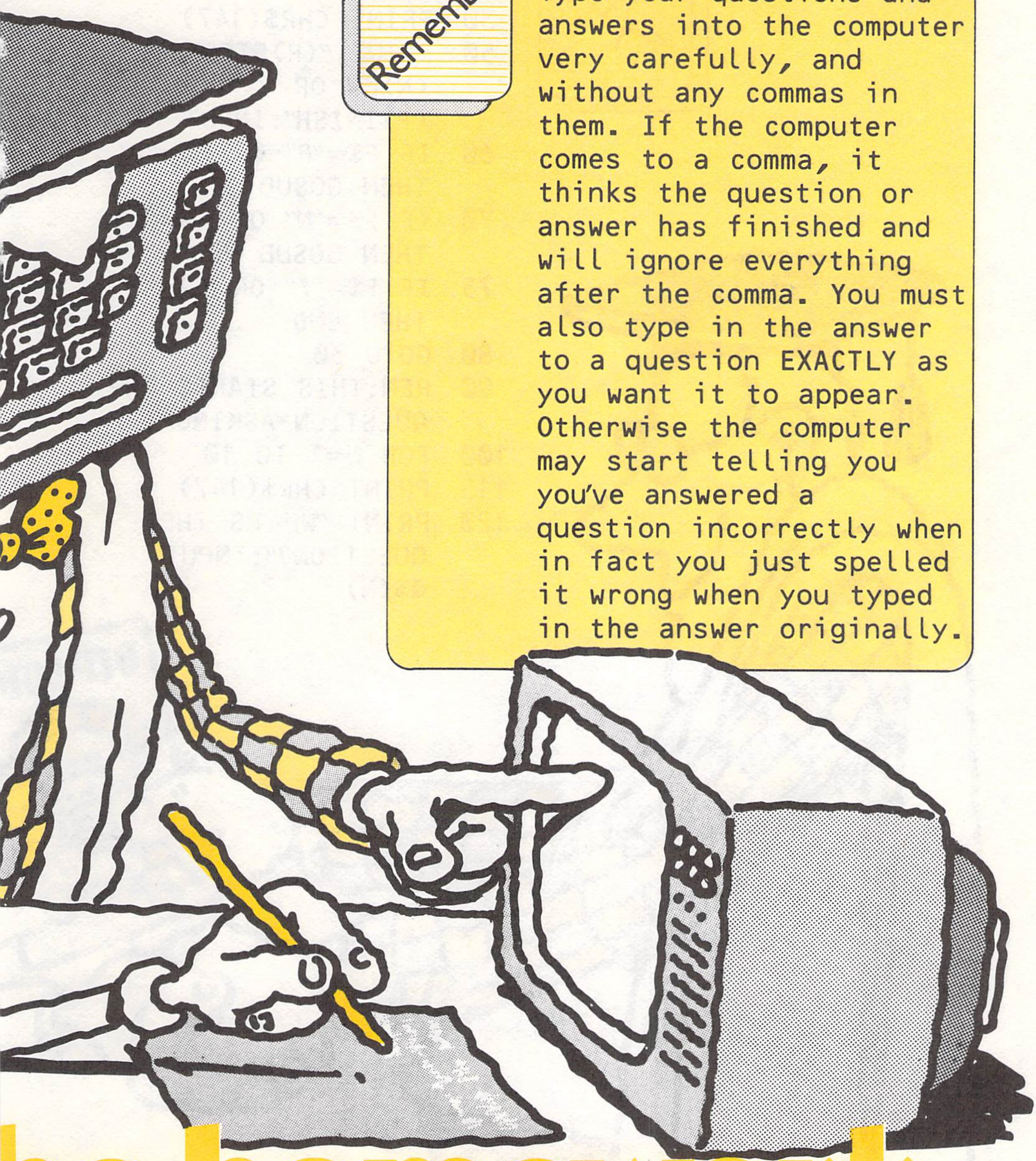
The answers and questions are stored in arrays, each of which holds ten questions or answers. You could however enlarge the size of the array so that it would store twenty or even fifty questions, depending on the size of the exam you're studying for.



Helping with t

Remember

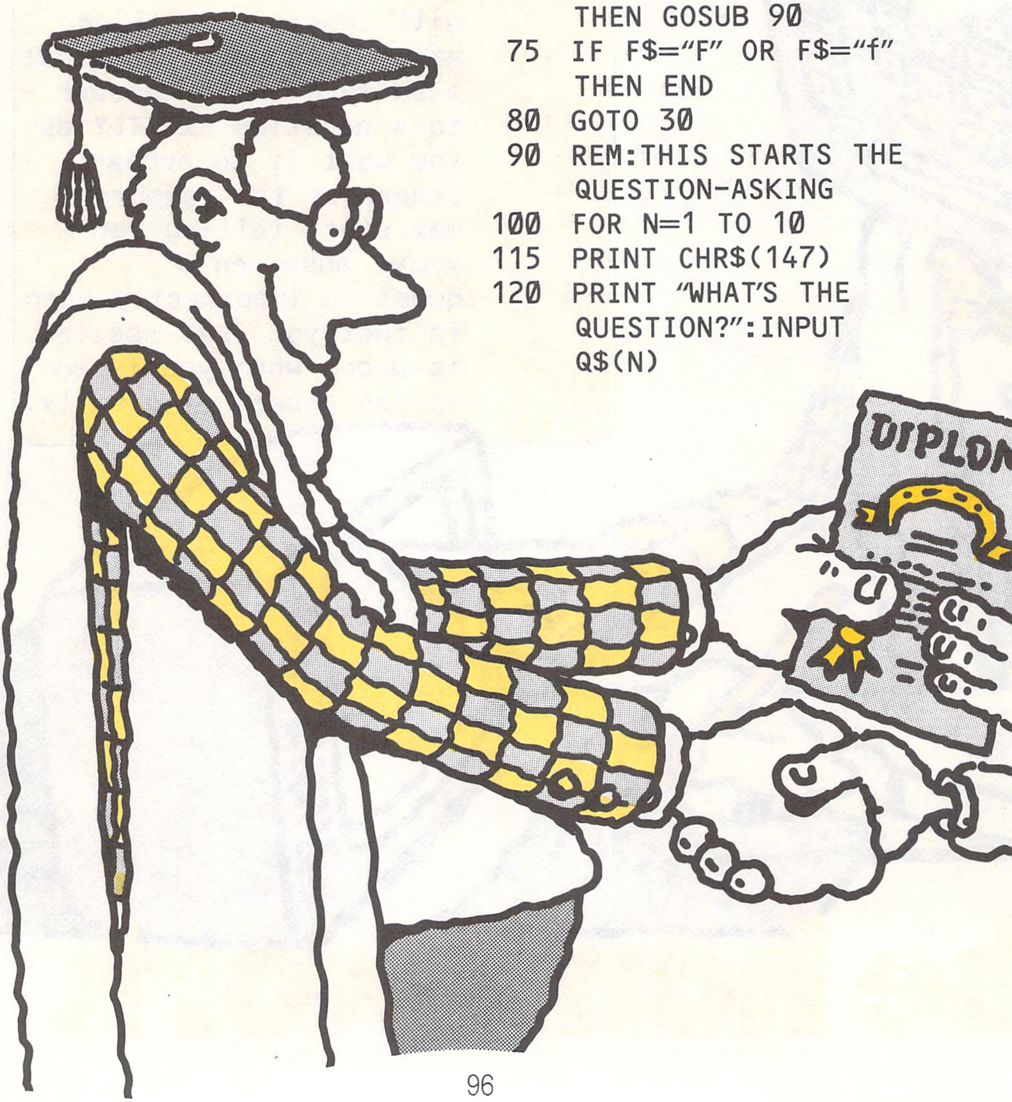
Type your questions and answers into the computer very carefully, and without any commas in them. If the computer comes to a comma, it thinks the question or answer has finished and will ignore everything after the comma. You must also type in the answer to a question EXACTLY as you want it to appear. Otherwise the computer may start telling you you've answered a question incorrectly when in fact you just spelled it wrong when you typed in the answer originally.



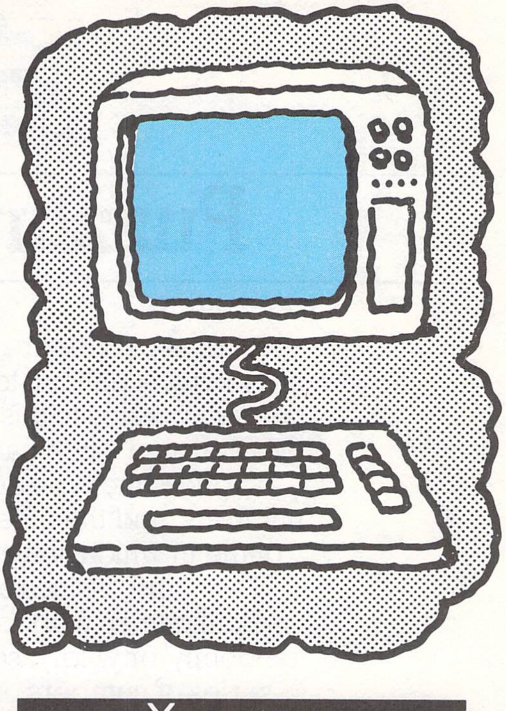
he homework

Try this

```
10  DIM Q$(10)
20  DIM A$(10)
30  PRINT CHR$(147)
50  PRINT "(R)ETEST,
     (N)EW OR
     (F)INISH":INPUT F$
60  IF F$="R" OR F$="r"
     THEN GOSUB 160
70  IF F$="N" OR F$="n"
     THEN GOSUB 90
75  IF F$="F" OR F$="f"
     THEN END
80  GOTO 30
90  REM:THIS STARTS THE
     QUESTION-ASKING
100 FOR N=1 TO 10
115 PRINT CHR$(147)
120 PRINT "WHAT'S THE
     QUESTION?":INPUT
     Q$(N)
```



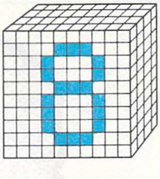
```
125  PRINT CHR$(147)
130  PRINT "WHAT'S THE
     ANSWER TO THAT
     QUESTION?":INPUT
     A$(N)
140  NEXT N
150  RETURN
160  REM:THIS SETS UP THE
     QUIZ
165  PRINT CHR$(147)
170  FOR N=1 TO 10
180  PRINT Q$(N)
190  INPUT R$
200  IF R$<>A$(N) THEN
     PRINT "YOU GOOFED,
     TRY AGAIN":GOTO 180
210  PRINT "YES, YOU'RE
     RIGHT"
220  NEXT N
230  RETURN
```



Your turn

You could add more questions to the program by changing the size of the array and the FOR. .NEXT loops that determine how many questions are to be asked.

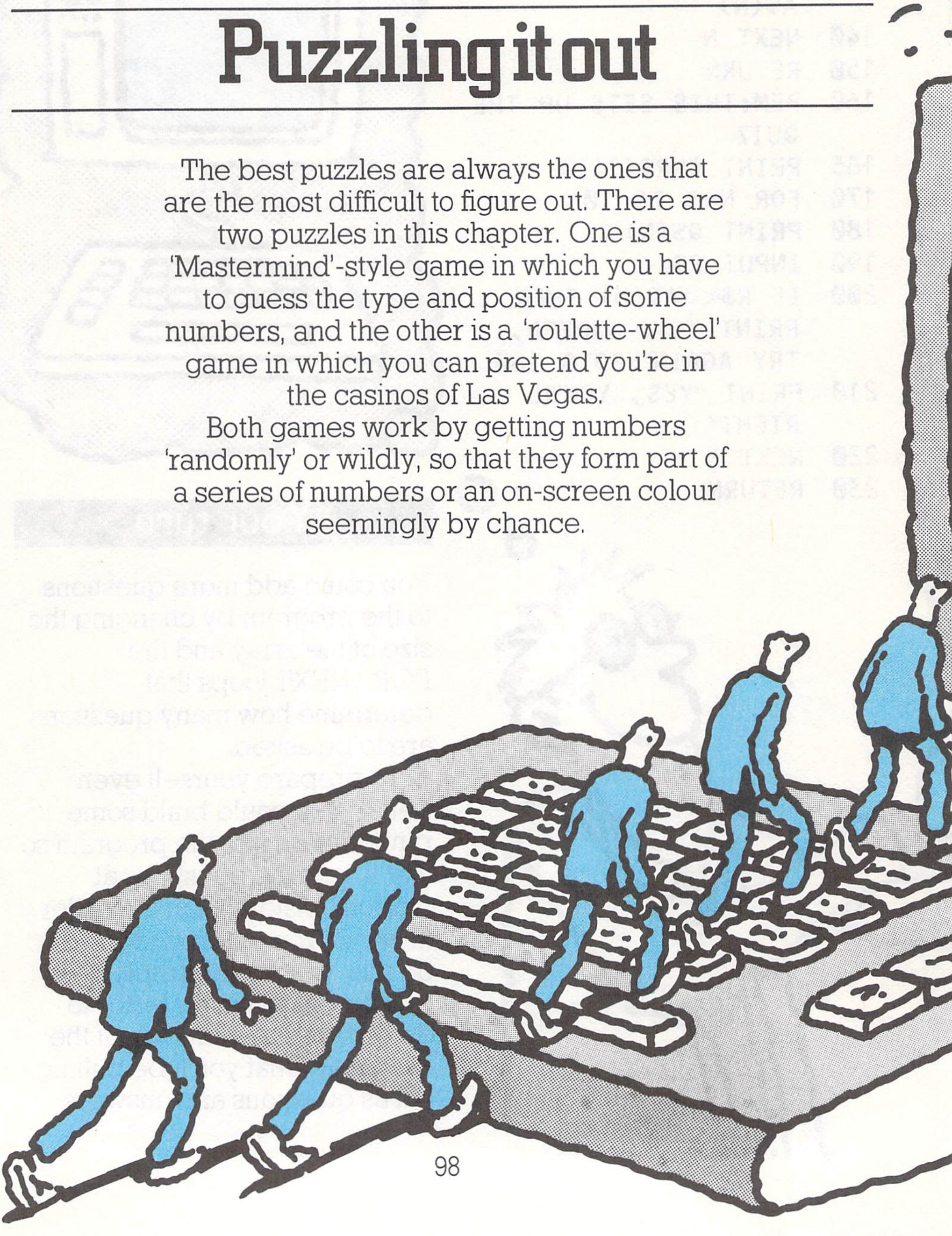
To prepare yourself even better, you could build some randomness into the program so that it asks the questions at random, instead of in the order in which you entered them. To do this, you would simply have to set a randomising factor to determine the elements of the two arrays that you'll be pulling out as questions and answers.



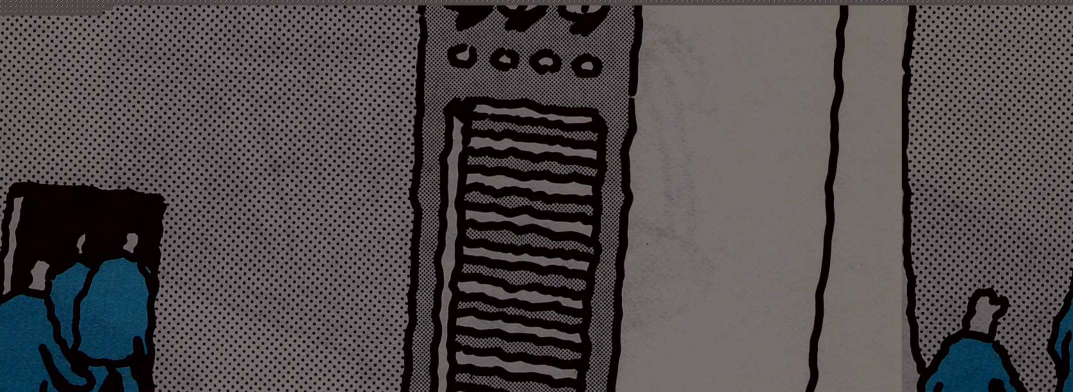
Puzzling it out

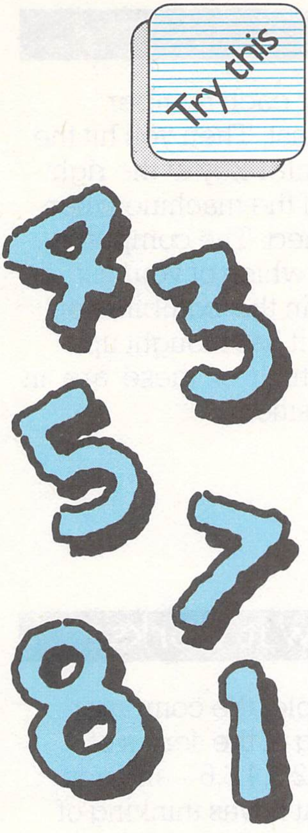
The best puzzles are always the ones that are the most difficult to figure out. There are two puzzles in this chapter. One is a 'Mastermind'-style game in which you have to guess the type and position of some numbers, and the other is a 'roulette-wheel' game in which you can pretend you're in the casinos of Las Vegas.

Both games work by getting numbers 'randomly' or wildly, so that they form part of a series of numbers or an on-screen colour seemingly by chance.



[redacted]



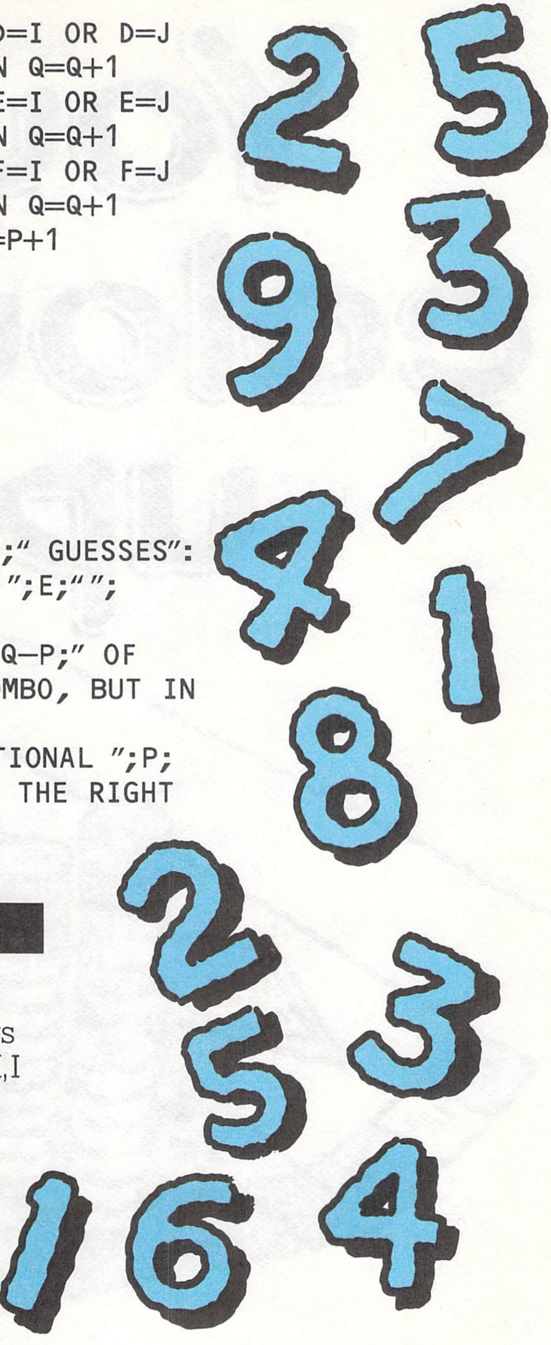


```
10  PRINT CHR$(147):T=Ø:N=6
20  A=RND(N):B=RND(N):C=RND(N):
    D=RND(N):E=RND(N):F=RND(N)
30  A=INT(1Ø*A):B=INT(1Ø*B):
    C=INT(1Ø*C):D=INT(1Ø*D):
    E=INT(1Ø*E):F=INT(1Ø*F)
40  IF A=B OR A=C OR A=D OR A=E
    OR A=F OR B=C OR B=D OR B=E
    OR B=F THEN 2Ø
50  IF C=D OR C=E OR C=F OR D=E
    OR E=F THEN 2Ø
55  IF A*B*C*D*E*F=Ø THEN 2Ø
60  T=T+1:P=Ø:Q=Ø:PRINT “GUESS 6
    NUMBERS BETWEEN 1 AND 1Ø,
    WITH COMMAS AFTER EACH”
65  PRINT “NUMBER YOU GUESS”
70  INPUT G,H,I,J,K,L:
    IF G*H*I*J*K*L=Ø THEN 7Ø
80  Q=Ø:IF A=G OR A=H OR A=I OR
    A=J OR A=K OR A=L THEN Q=Q+1
90  IF B=G OR B=H OR B=I OR B=J
    OR B=K OR B=L THEN Q=Q+1
100 IF C=G OR C=H OR C=I OR C=J
    OR C=K OR C=L THEN Q=Q+1
```

Did you see?

A number of different comparisons are made between numbers in this program to make it work properly. First, all the numbers in your guess are compared to all the numbers in the combination the computer is 'thinking of' to see if any of the first bunch match any of the second. The second set of comparisons sets only the first number of your guess against the first number of the combination, then the second number against the second number of the combination and so on until the sixth number in your guess has been compared with the sixth number in the computer's combination.

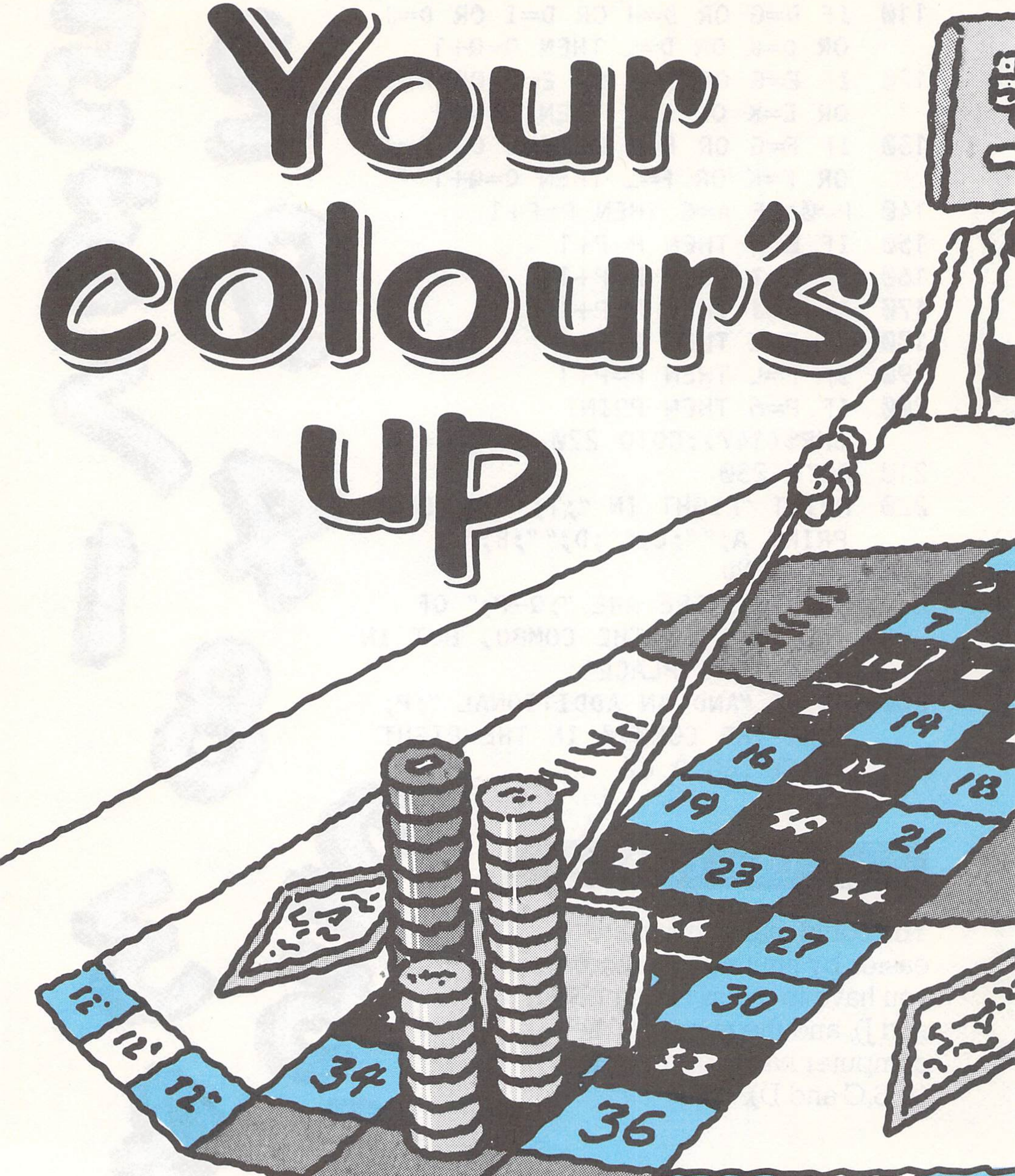
```
110  IF D=G OR D=H OR D=I OR D=J
     OR D=K OR D=L THEN Q=Q+1
120  IF E=G OR E=H OR E=I OR E=J
     OR E=K OR E=L THEN Q=Q+1
130  IF F=G OR F=H OR F=I OR F=J
     OR F=K OR F=L THEN Q=Q+1
140  P=0:IF A=G THEN P=P+1
150  IF B=H THEN P=P+1
160  IF C=I THEN P=P+1
170  IF D=J THEN P=P+1
180  IF E=K THEN P=P+1
190  IF F=L THEN P=P+1
200  IF P=6 THEN PRINT
     CHR$(147):GOTO 220
210  GOTO 230
220  PRINT “RIGHT IN ”;T;“ GUESSES”:
     PRINT A;“ ”;C;“ ”;D;“ ”;E;“ ”;
     F;“ ”:END
230  PRINT “THERE ARE ”;Q–P;” OF
     YOUR NOS IN THE COMBO, BUT IN
     THE WRONG PLACE”
240  PRINT “AND AN ADDITIONAL ”;P;
     “ IN THE COMB & IN THE RIGHT
     PLACE”:GOTO 60
```

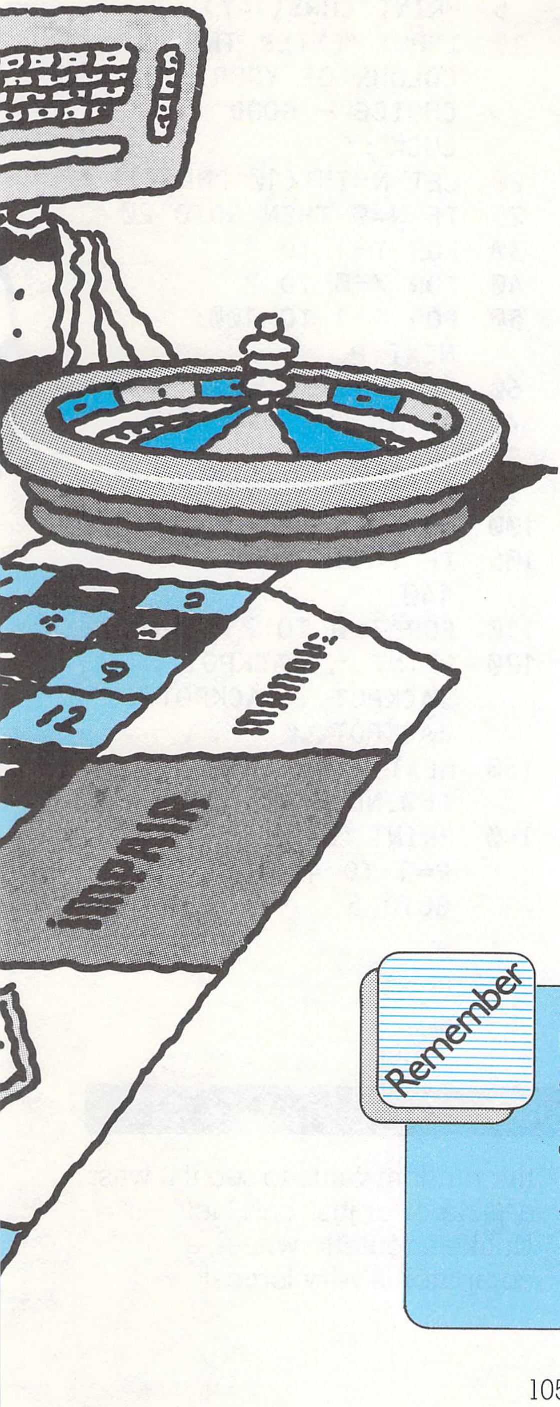


Your turn

You could make this game easier by limiting the numbers you have to guess to four (G,H,I and J), and the numbers the computer has to guess to four (A,B,C and D).

Your colour's up





Project

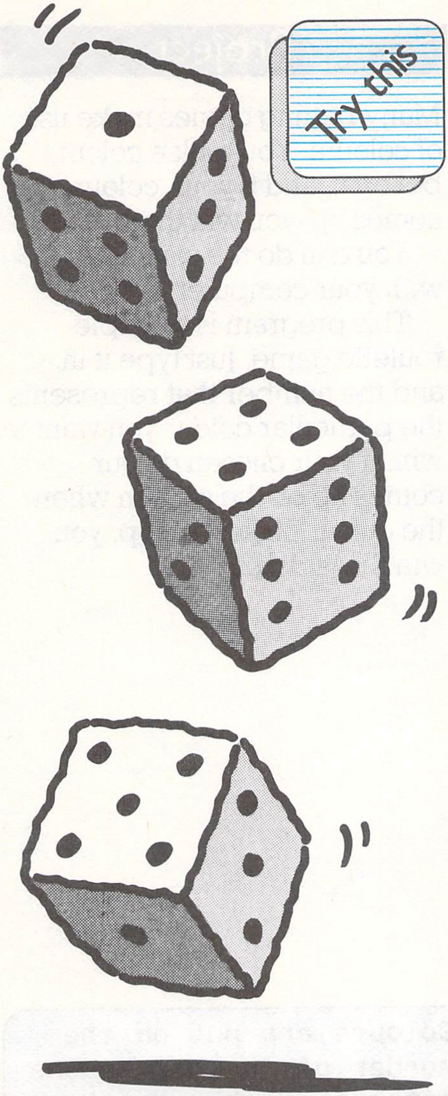
Many betting games make use of colours. You pick a colour, bet on it, and if your 'colour comes up' you win the bet.

You can do the same thing with your computer.

This program is a simple roulette game. Just type it in, and the number that represents the particular colour you want to win. If your chosen colour comes up on the screen when the colour changes stop, you win the jackpot!

Remember

Colours are put on the border of the screen one after the other by the inner FOR..NEXT statement or 'loop'. Then the colours are displayed a further five times by the FOR..NEXT loop started at line 30.

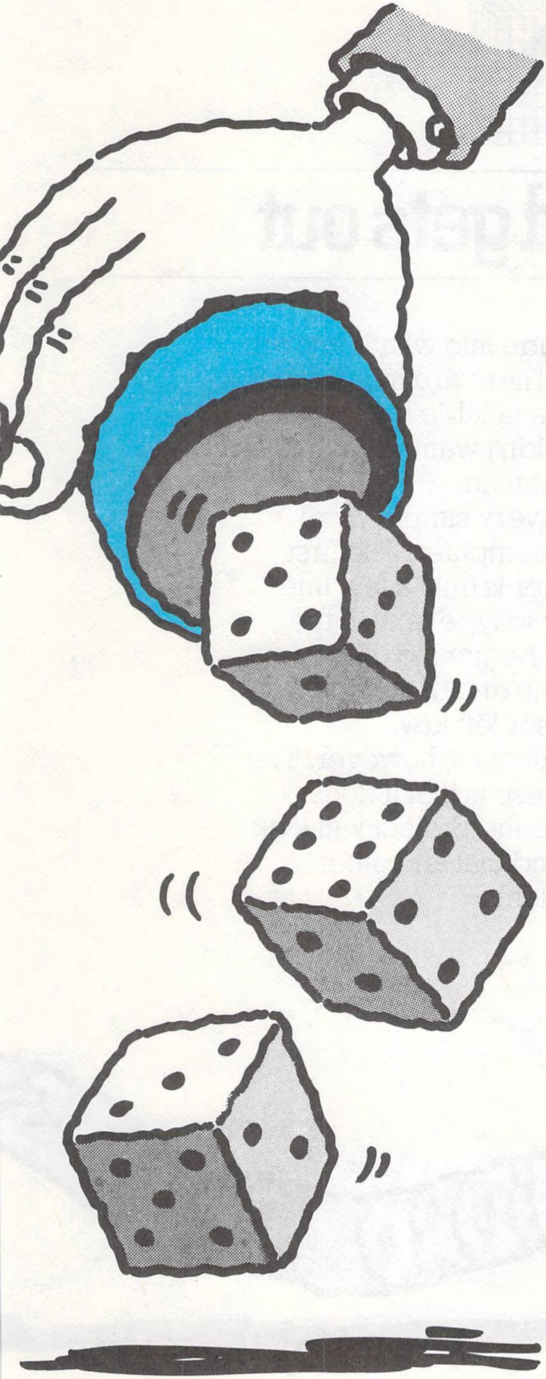


Try this

```
   5  PRINT CHR$(147)
  10  INPUT "ENTER THE
       COLOUR OF YOUR
       CHOICE - GOOD
       LUCK";Y
  20  LET N=INT(10*RND(7))
  25  IF N=0 THEN GOTO 20
  30  FOR T=1 TO 5
  40  FOR X=0 TO 7
  50  FOR P=1 TO 100:
       NEXT P
  60  POKE 53280,X
  70  PRINT CHR$(147)
  80  NEXT X
  90  NEXT T
 100  POKE 53280,N
 105  IF Y<>N THEN GOTO
       140
 110  FOR Z=0 TO 21
 120  PRINT "..JACKPOT..
       JACKPOT...JACKPOT..
       JACKPOT.."
 130  NEXT Z:FOR P=1 TO
       400:NEXT P:GOTO 5
 140  PRINT "BAD LUCK":FOR
       P=1 TO 400:NEXT P:
       GOTO 5
```

Did you see?

This program 'cheated'. It generated a random number between 0 and 7, put its colour on the screen and then compared your guess against the random value to see if it was a 'jackpot' or just 'bad luck'. Unlike a roulette wheel, a computer is very logical.



Your turn

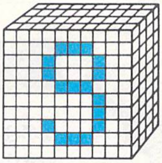
You are trying to mimic a roulette wheel, so a couple of lines will improve the program further. Since a roulette wheel is spun by hand, it begins by revolving quickly then slows down gradually, increasing the suspense.

Add these lines so that the colours change more slowly as the FOR. .NEXT statements are executed. As long as you've entered the numbers shown above, line 50 will replace the line 50 already there.

```
45  LET P=3*T+5
5Ø  FOR V=1 TO P:NEXT V
```

You will have noticed that as the wheel slows down, the last and winning colour can't be predicted at all, because it is random. It would make the game more exciting if you knew that the sequence of colour changes would stay till the very last moment.

A few more lines in the program will allow you to finish on the random colour. Just put in another FOR. .NEXT loop at the end, going from 0 to n (n is the number the computer has chosen).

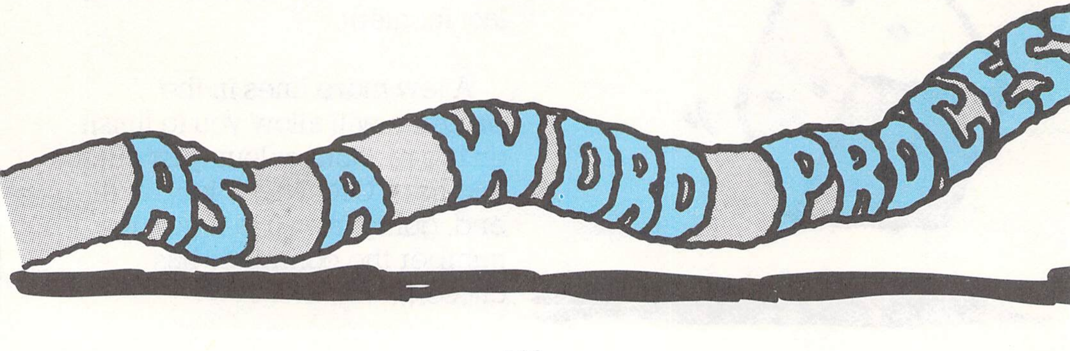


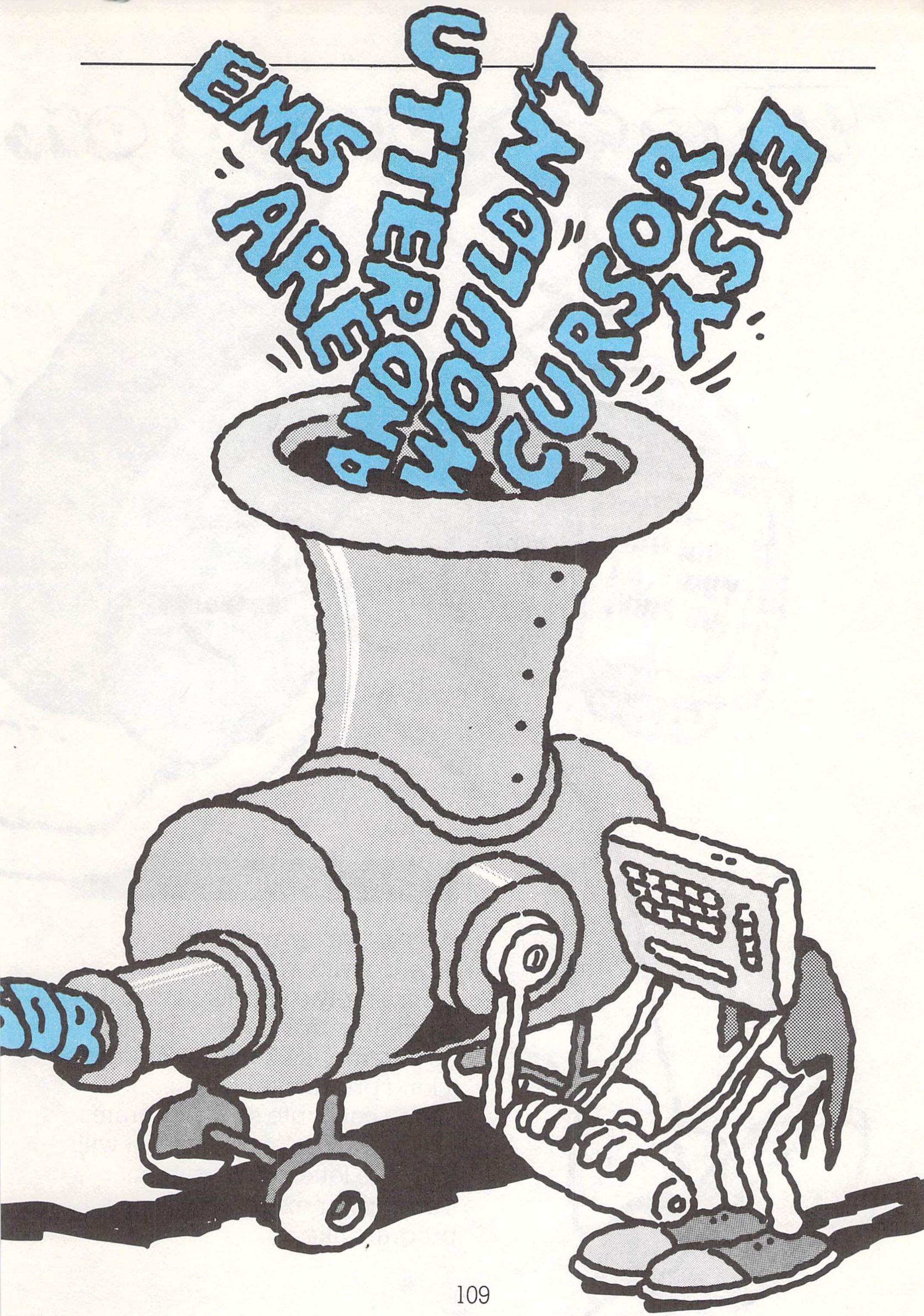
The word gets out

Your micro can be made into what's known as a word processor. There are many word processing programs available but they are long ones – you wouldn't want to have to type them in.

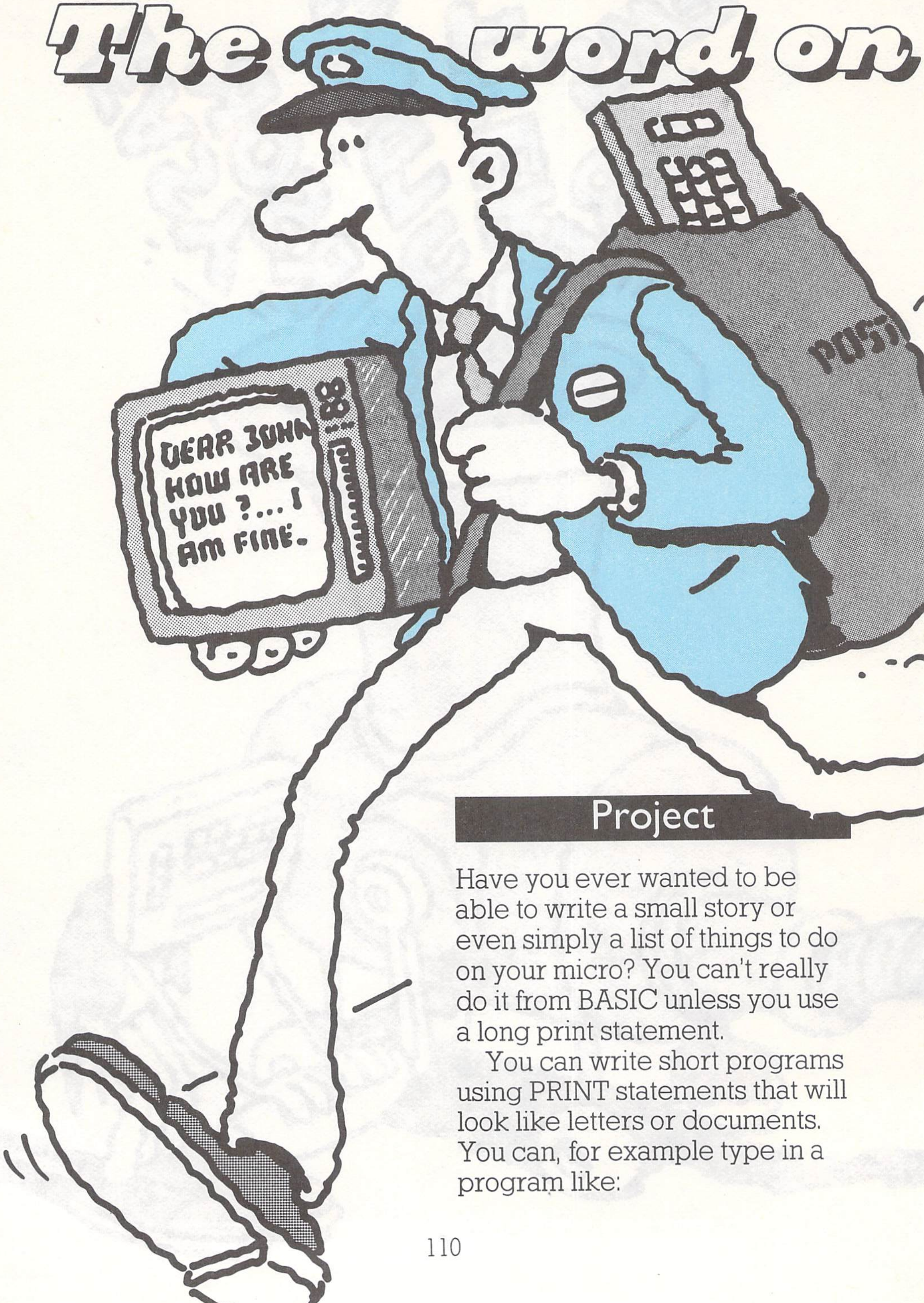
It's easy to make a very simple word processor for your computer. The first program in this chapter is only a few lines, but it will allow you to type text on the screen. To print at the beginning of the next line when you get to the end of the first, you use the cursor left key.

It does have some limitations, however. The most important of these is that it doesn't have a cursor. The second project will look at how to get round that and other problems.




EMS
ARE
UTTER
AND
YOU
WOULDN'T
CURSOR
EASY
DOR

The word on



Project

Have you ever wanted to be able to write a small story or even simply a list of things to do on your micro? You can't really do it from BASIC unless you use a long print statement.

You can write short programs using PRINT statements that will look like letters or documents. You can, for example type in a program like:

words

```
10  PRINT "DEAR YOU,"
20  PRINT
30  PRINT "THANKS FOR
     BUYING THIS BOOK.
     I HOPE "
40  PRINT "THAT YOU
     ENJOY IT AND HAVE
     FUN WITH THE"
50  PRINT "PROJECTS."
60  PRINT
70  PRINT
80  PRINT "YOURS
     SINCERELY,"
90  PRINT
100 PRINT
110 PRINT "THE AUTHOR"
```

That little program would produce something that looks like the beginnings of a letter, but the only way to change the PRINT statements to read as you want them to, is to use the computer's built-in 'editor'.

What you'll need is something that takes information that you've typed in from the keyboard and allows you to edit it on the screen.

Remember

One of the best ways to get information from a computer keyboard in the Basic programming language is to use the GET, GET$, INKEY$ and INPUT statements.

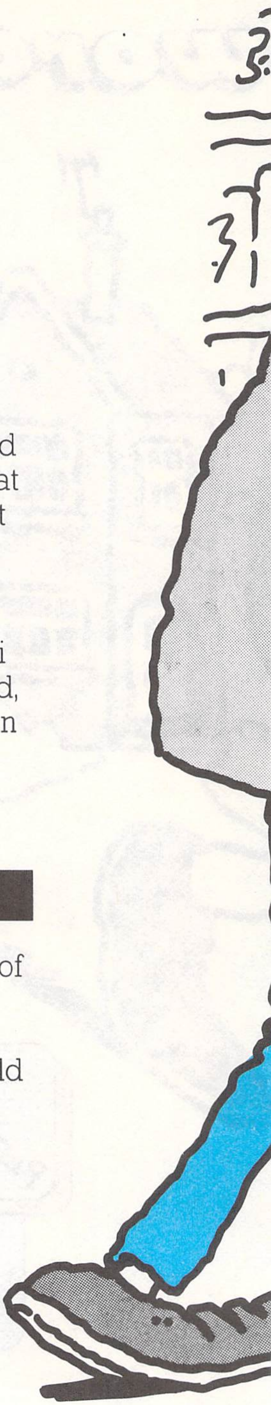
Try this

```
10  PRINT CHR$(14)
20  PRINT CHR$(147)
30  POKE 204,0
40  GET A$:IF A$=""
     THEN 40
50  PRINT A$;
60  GOTO 40
```

There is a semi colon at the end of the PRINT A$ statement. That tells the computer that the next letter to be printed on the screen will follow on the same line as A$ and be printed in front of it. If you left off the semi colon and put in a colon instead, each letter would be printed on a new line.

Your turn

You could change the colours of the 'paper' and 'ink' you're typing on by using what you learned in the Colour My World section. (Hint: Remember that border and paper colours are often different.)

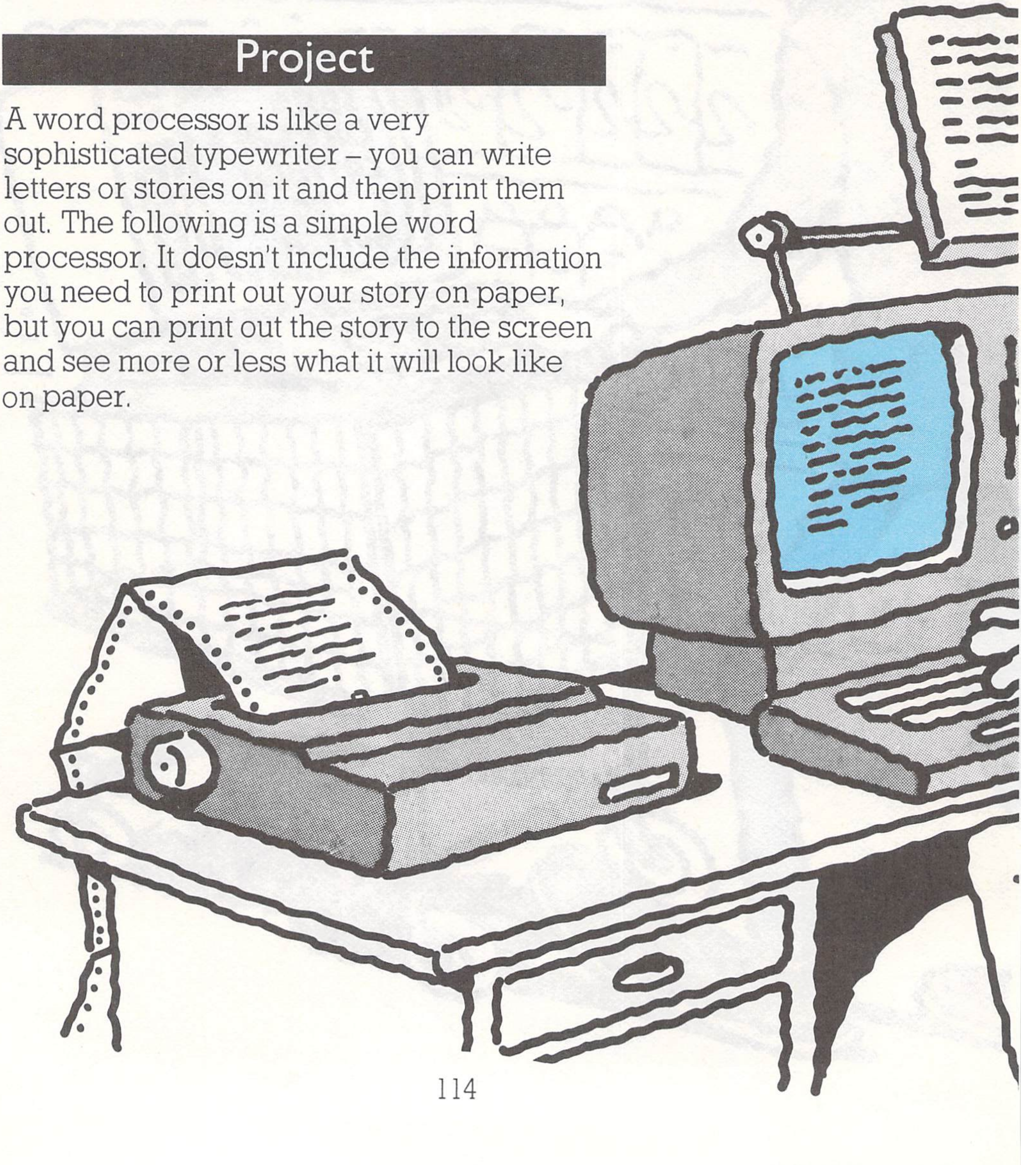


SOAP
CUCUMBER
CATFOOD
BREAD
BUTTER

Getting to with words

Project

A word processor is like a very sophisticated typewriter – you can write letters or stories on it and then print them out. The following is a simple word processor. It doesn't include the information you need to print out your story on paper, but you can print out the story to the screen and see more or less what it will look like on paper.



grips

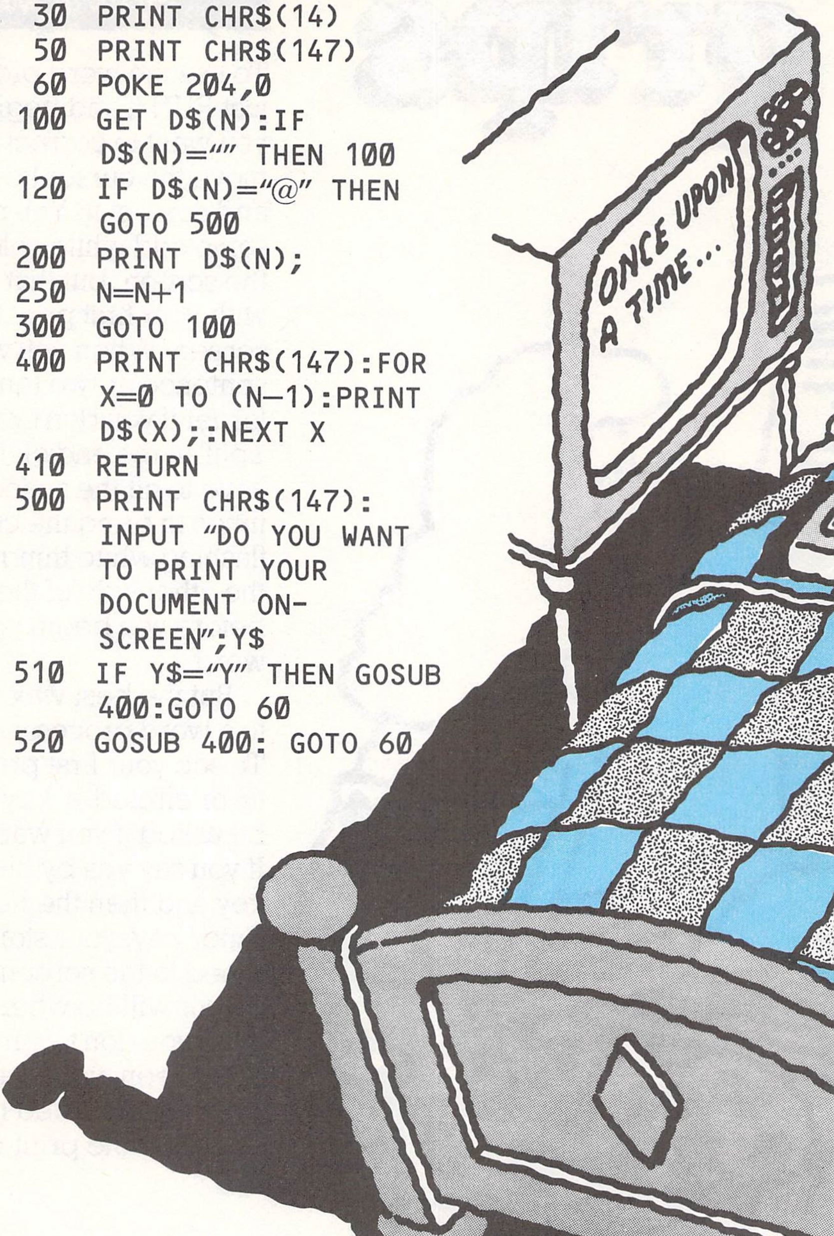
How it works

To use the word processor, you just RUN it and begin typing. If you want to correct a word, just move the cursor key backwards and correct it. You may get some odd white colouring on the screen, but that will go away with your first print to the screen. When you've typed a sentence or two (and don't forget if you don't want words to 'split' at the end of the line) you'll have to hit the space bar a few times to bring the cursor (the flashing white thing) round to the other side of the screen before you begin your next word.

But the best way to get used to a word processor is to try it. To see your first print-out, hit the @ or circled 'a' key, and you'll be asked if you want to print out. If you say yes by hitting the Y key and then the Return or Enter key, your story will be typed to the screen and your cursor will be where you last left off. If you don't want to type it to the screen, you'll be put back to where you started before you asked for the print-out.

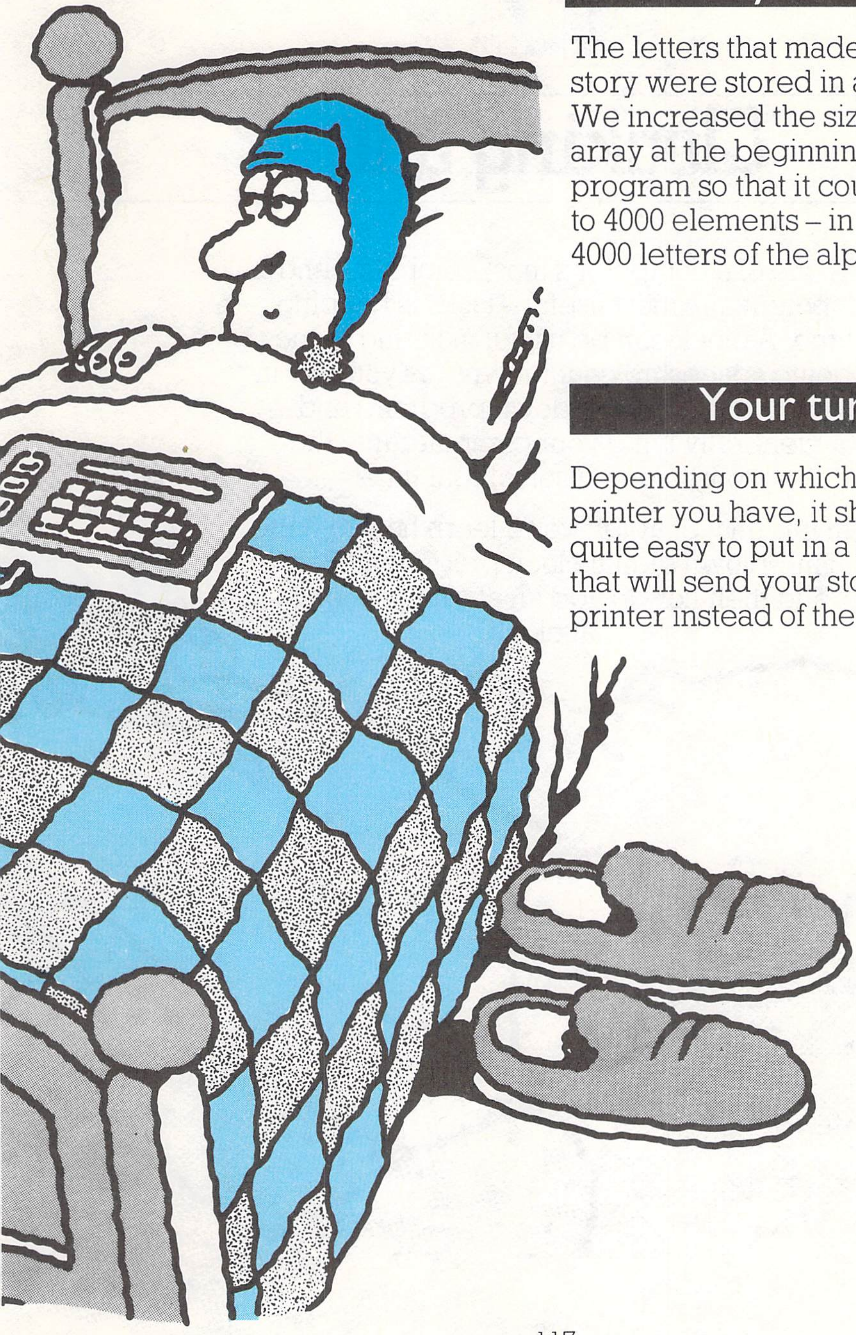
Try this

```
 20  DIM D$(4000):N=0
 30  PRINT CHR$(14)
 50  PRINT CHR$(147)
 60  POKE 204,0
100  GET D$(N):IF
     D$(N)=“” THEN 100
120  IF D$(N)=“@” THEN
     GOTO 500
200  PRINT D$(N);
250  N=N+1
300  GOTO 100
400  PRINT CHR$(147):FOR
     X=0 TO (N–1):PRINT
     D$(X);:NEXT X
410  RETURN
500  PRINT CHR$(147):
     INPUT “DO YOU WANT
     TO PRINT YOUR
     DOCUMENT ON-
     SCREEN”;Y$
510  IF Y$=“Y” THEN GOSUB
     400:GOTO 60
520 GOSUB 400: GOTO 60
```



Did you see?

The letters that made up your story were stored in an array. We increased the size of that array at the beginning of the program so that it could hold up to 4000 elements – in this case, 4000 letters of the alphabet.



Your turn

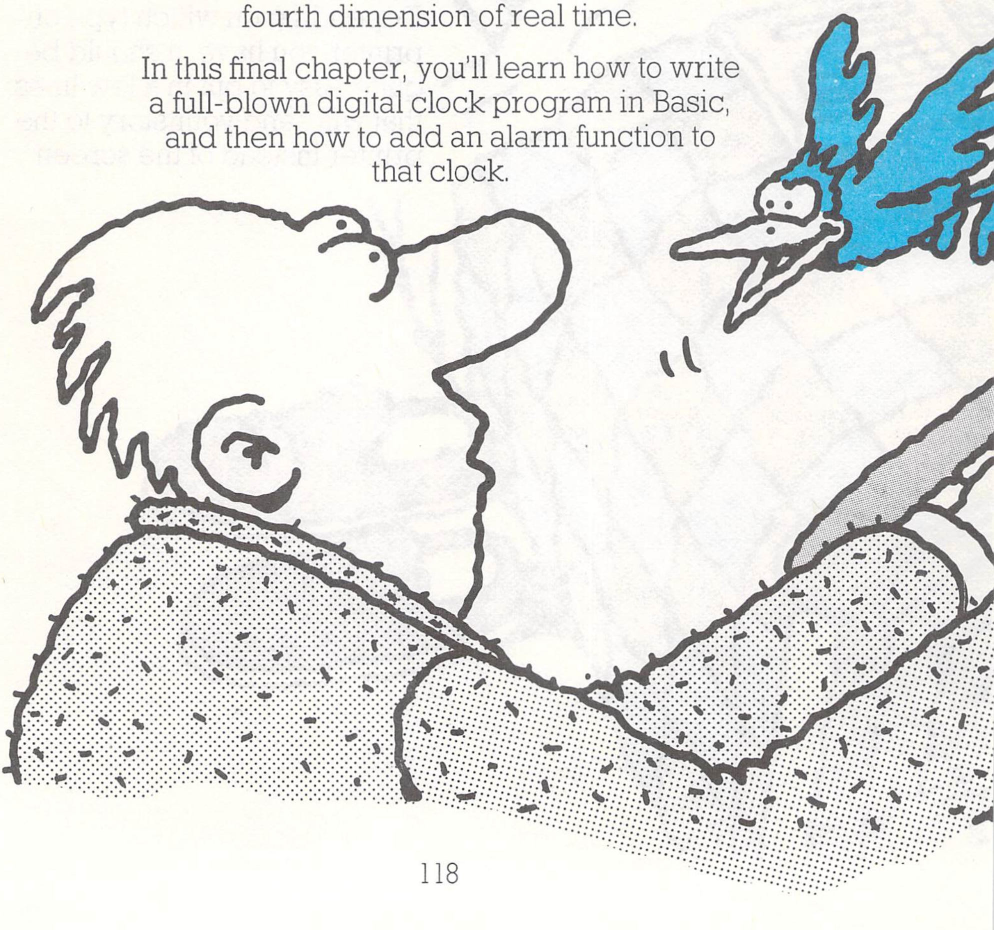
Depending on which type of printer you have, it should be quite easy to put in a few lines that will send your story to the printer instead of the screen.

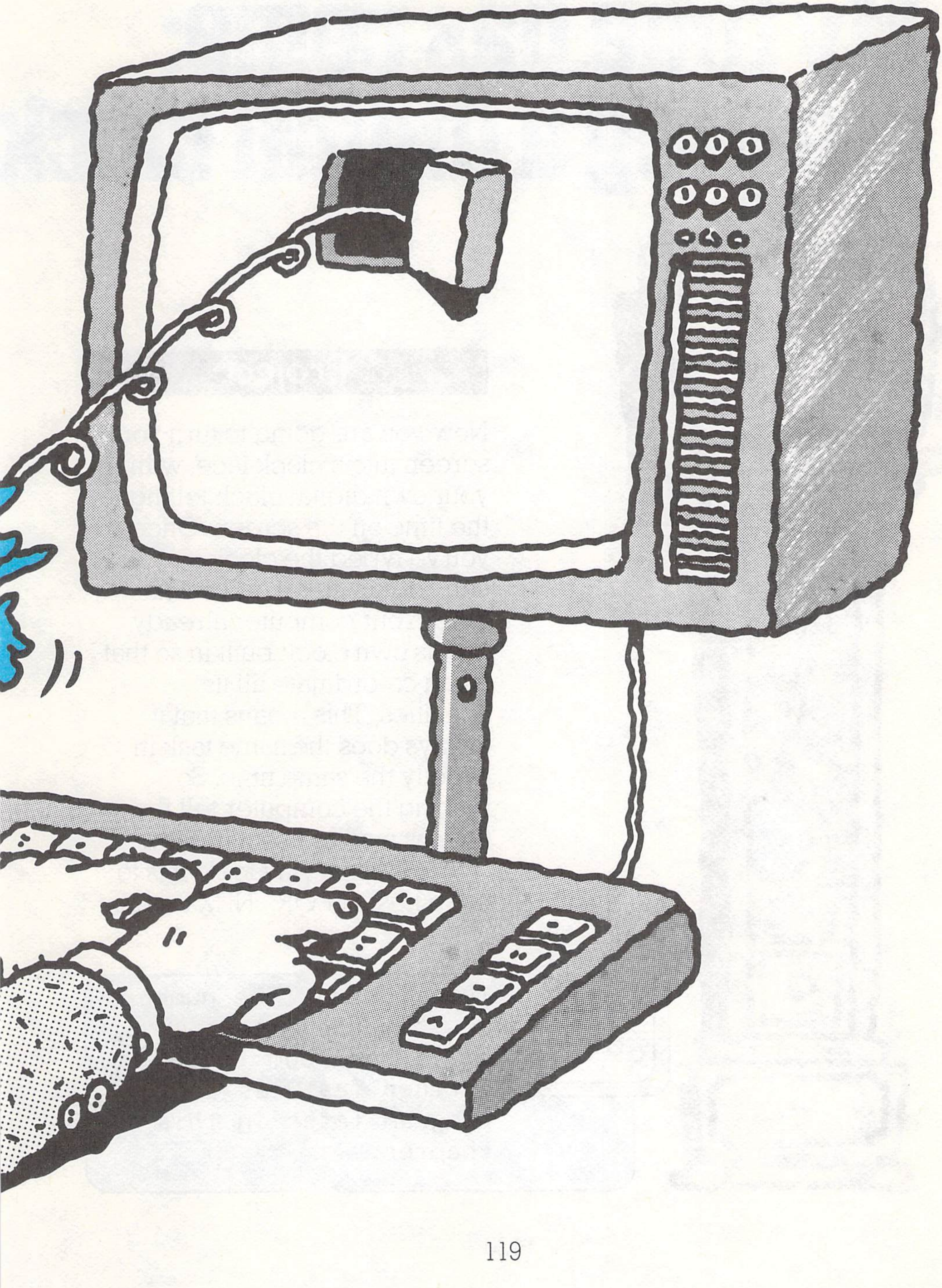
10

Clocking out

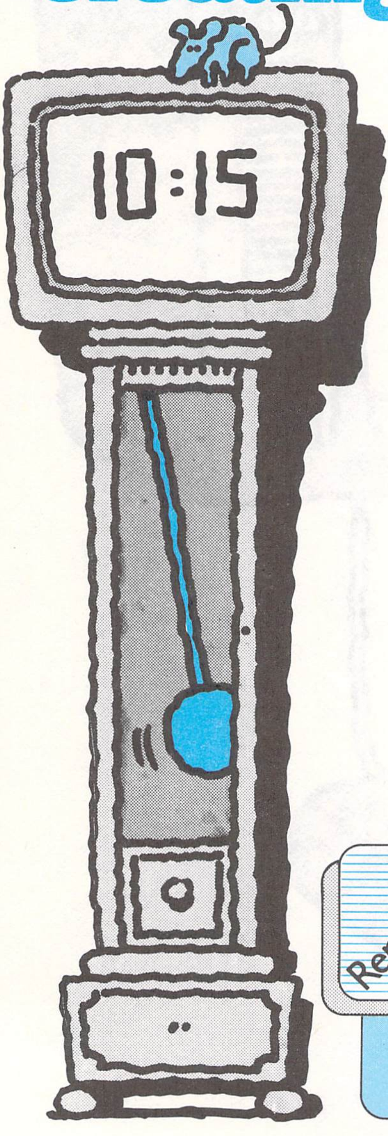
One of a computer's most common – and potentially most useful – tasks is keeping time. A clock can be useful in timing games, doing spaceship countdowns (as you saw in chapter 3's Countdown program) and generally tying your computer into the fourth dimension of real time.

In this final chapter, you'll learn how to write a full-blown digital clock program in Basic, and then how to add an alarm function to that clock.





My time has come: creating a digital clock



Project

Now you are going to turn your screen into a clock face, with your own digital clock telling the time on the screen. Once you've typed the clock in, you can add an alarm and set its time. Your computer already has its own clock built in so that it can co-ordinate all its activities. This means that it always does the same task in exactly the same time. So making the computer tell the time as well as use it is easy.

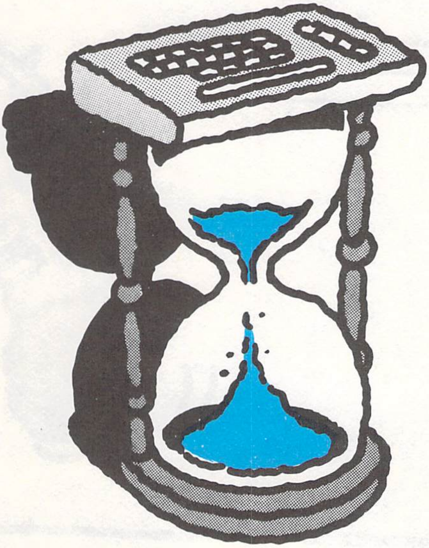
The easiest way to do it is to use a nest of FOR. .NEXT loops.

Remember

Type in the line numbers as they are shown. This gives you room to add further features to the program later in this chapter.

Try this

```
 5Ø  PRINT CHR$(147)
10Ø  FOR H=Ø TO 12
11Ø  FOR M=Ø TO 59
12Ø  REM SPACE HERE FOR
     FURTHER CLOCK
     FEATURES
5ØØ  FOR S=Ø TO 59
51Ø  FOR T=Ø TO 9
52Ø  FOR A=1 TO 28:NEXT A
53Ø  PRINT “     ”;H;“ ”;
     M;“ ”;S;“ ”;T;
532  PRINT CHR$(19)
535  FOR Z=1 TO 40:PRINT
     “ ”;:NEXT Z
54Ø  NEXT T
71Ø  NEXT S
72Ø  PRINT CHR$(147)
73Ø  NEXT M
74Ø  NEXT H
```



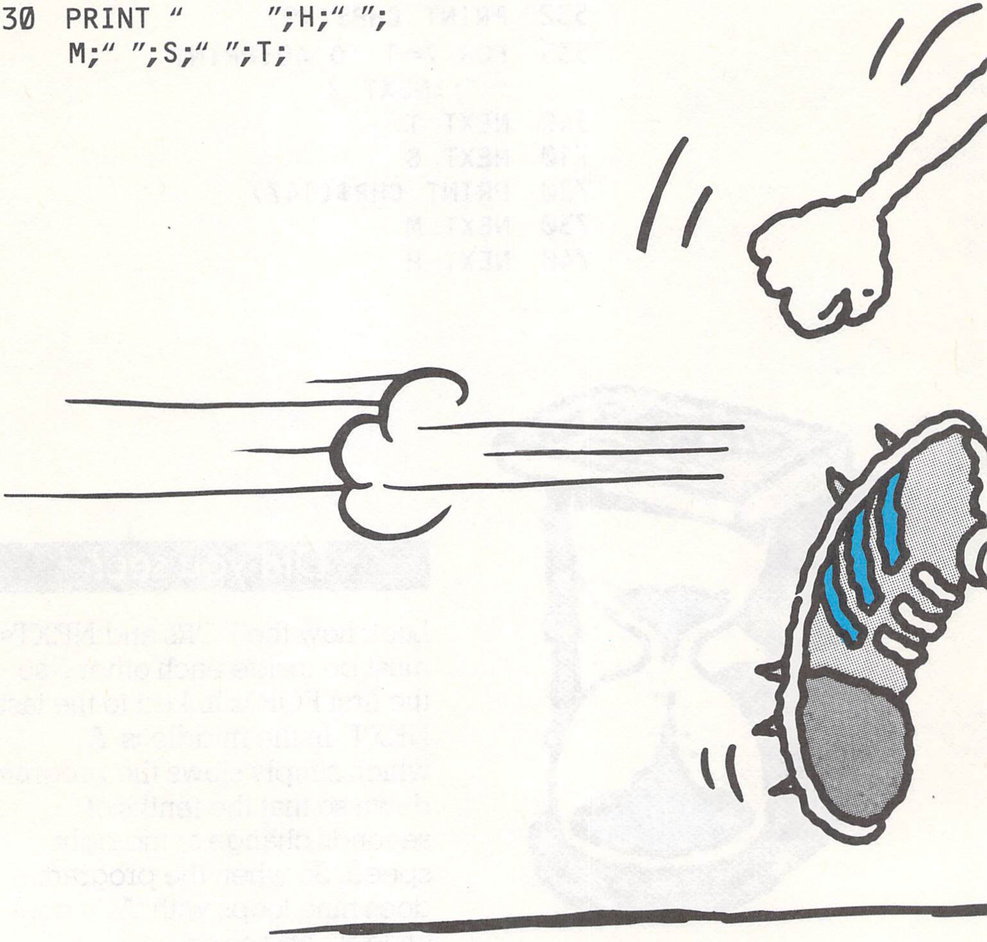
Did you see?

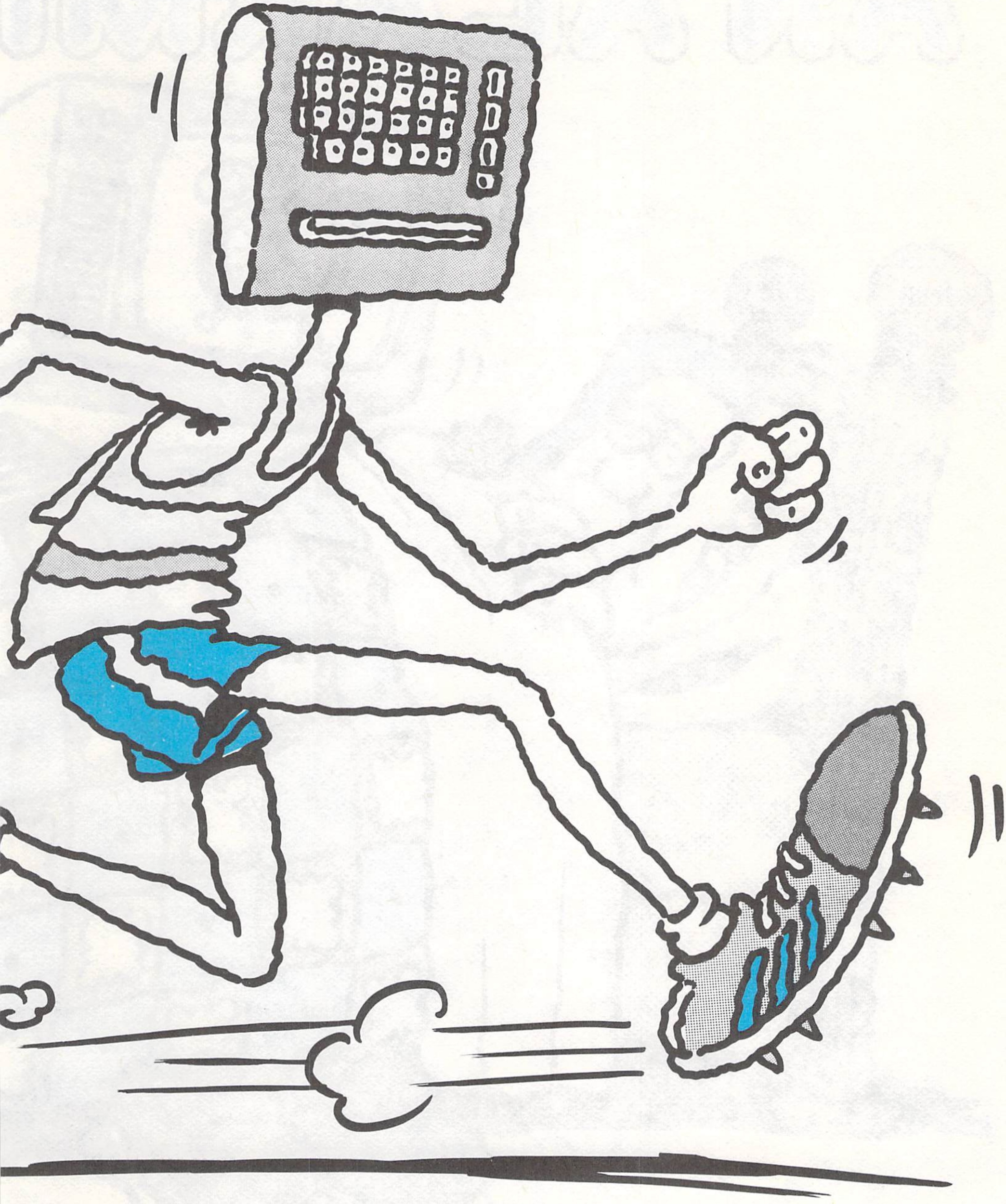
Look how the FORs and NEXTs must be inside each other – so the first FOR is linked to the last NEXT. In the middle is ‘A’, which simply slows the program down so that the tenths of seconds change at the right speed. So when the program does nine loops with ‘A’, it goes on to ‘T’, and so on.

Your turn

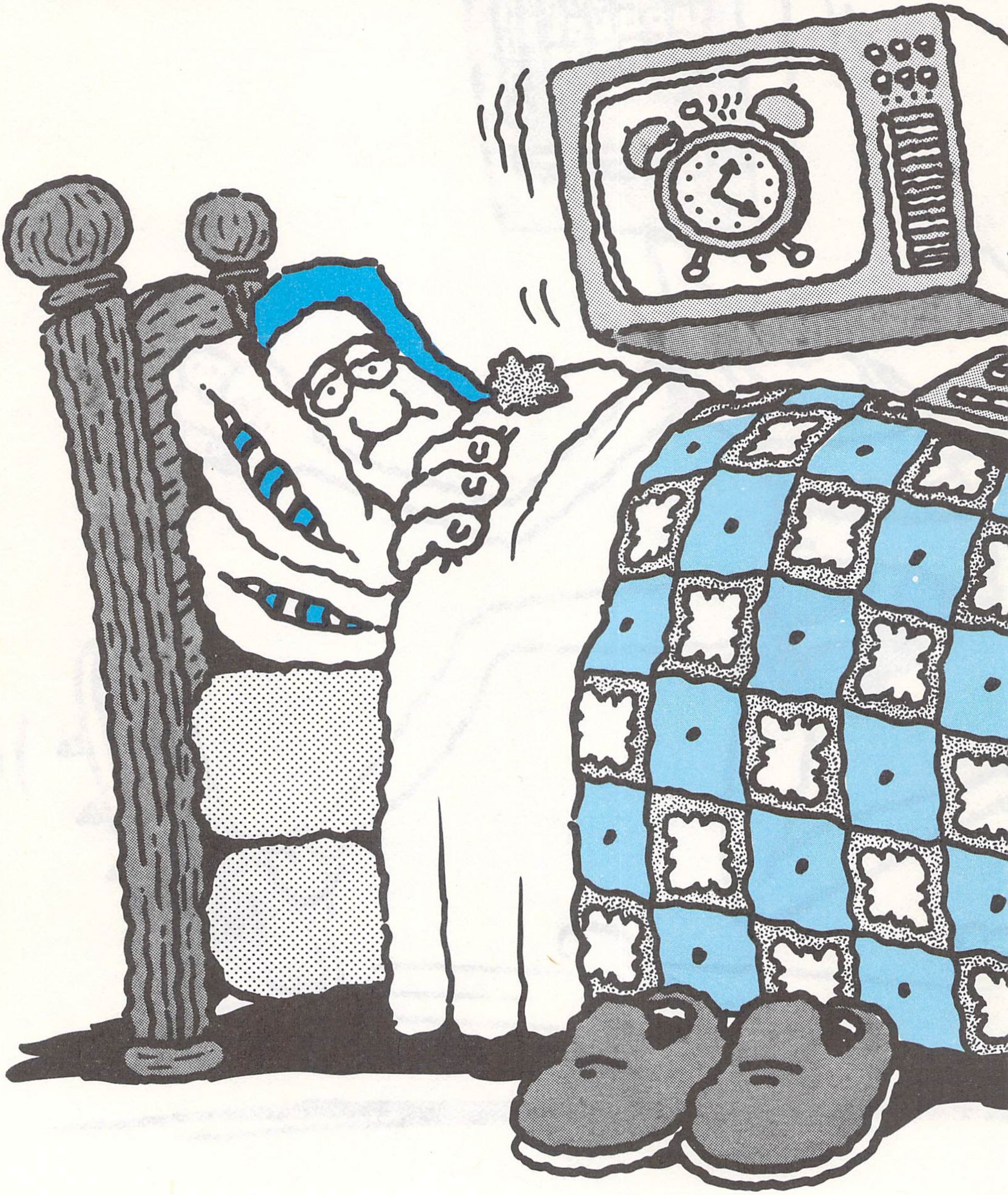
Numbers changing on the screen don't look particularly impressive so now we can add a little colour to the program to dress the clock up. To make your clock a little more attractive you can now enter these lines into the program.

```
300  PRINT CHR$(158)
525  PRINT “     HR MIN SEC F”
530  PRINT “     ”;H;“ ”;
     M;“ ”;S;“ ”;T;
```





AN ALARMI



NG CHANGE

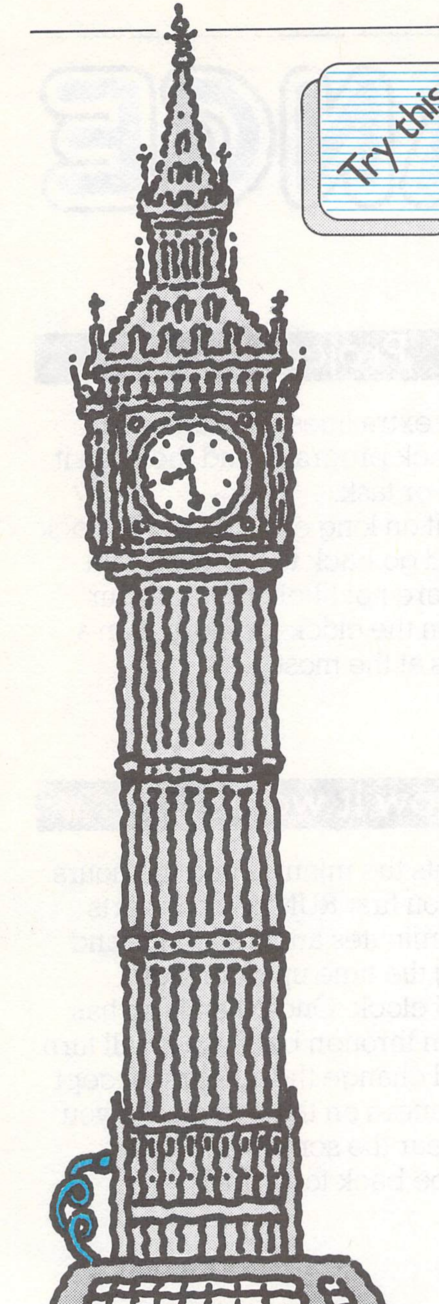
Project

With just a few extra lines you can put an alarm in the clock program, and then use it to time a game or task.

If you leave it on long enough, it will clock up 12 hours and go back to 8 hours again. However, you are not likely to want your computer to run the clock for more than a couple of hours at the most.

How it works

The clock counts the minutes and the hours from the time you first RUN it – so it starts from 0 hours, 0 minutes and 0 seconds and keeps clocking the time up from there, unlike a normal clock. Once the alarm has started, let it run through because it will turn off the flash and change the colours, except for the alarm pattern on the screen. All you have to do is clear the screen and your computer will be back to normal.

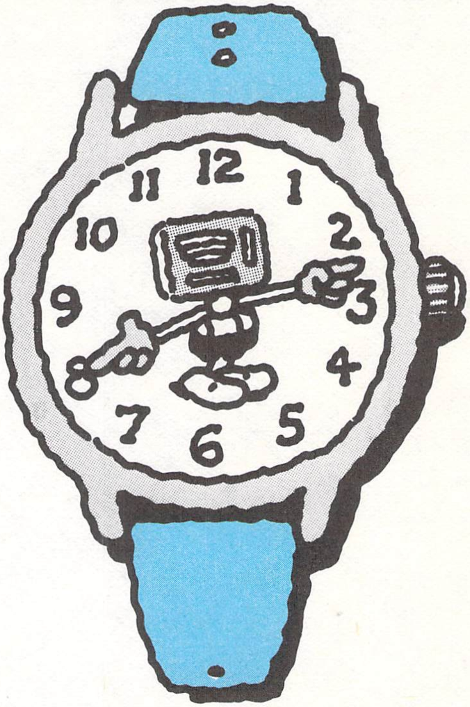


Try this

Here's the revised version of the clock program – with the built-in alarm.

```
  5  PRINT CHR$(147)
 10  INPUT "HOW MANY
     HOURS FROM NOW DO
     YOU WANT TO SET THE
     ALARM FOR";Y
 20  INPUT "AND HOW MANY
     MINUTES AFTER THAT
     HOUR";X :PRINT
     CHR$(147)
 30  PRINT "ALARM SET
     FOR ";Y;" HOURS AND
     ";X;" MINUTES FROM
     NOW"
 50  FOR B=1 TO 800:NEXT
     B:PRINT CHR$(147)
100  FOR H=0 TO 12
110  FOR M=0 TO 59
120  REM
200  IF M=X AND H=Y
     THEN GOTO 900
300  PRINT CHR$(158)
500  FOR S=0TO59
510  FOR T=0TO9
520  FOR A=1 TO 28:
     NEXT A
525  PRINT "      HR MIN
     SEC F"
530  PRINT "      ";H;
     " ";M;" ";S;" ";T;
532  PRINT CHR$(19)
535  FOR Z=1 TO 40:PRINT
     " ";:NEXT Z
540  NEXT T
700  REM
710  NEXT S
```

```
 720  PRINT CHR$(147)
 730  NEXT M
 740  NEXT H
 900  PRINT CHR$(147)
 910  FOR Z=1 TO 100
 920  POKE 53281,2
 930  PRINT
      "ALARM.......ALARM
      .....ALARM.......
      ALARM"
 950  POKE54276,0:POKE
      54277,0:POKE54272,0
 960  POKE54296,15:POKE
      54276,33:POKE
      54277,64:POKE
      54278,64
 970  POKE 54273,36:POKE
      54272,85
 980  FOR W=1 TO 30:
      NEXT W
1000  NEXT Z
1010  POKE 54296,0:END
```



Your turn

The alarm goes for about 40 seconds and then the program stops. You know how long the alarm has been going for the first 40 seconds because of the number of 'Alarm' lines on the screen. What happens if you miss the alarm completely?

It might be useful if you made the clock start up again so that you knew how much time had passed since the alarm went off. Get the clock to start again, using a RETURN statement.

You can also use this feature to give you the time every quarter of an hour. You'll find that you'll have to do some thinking to get the clock to start up again. If you just put in a RETURN, you won't get the alarm after the first time. Try a GOTO instead. You might find that a statement to clear the screen in the right place would get rid of the flashing between alarms.

Break into the amazing world of computers

A great book of projects for the new computer generation

Commodore 64 Whizz Kid

MAKE IT FUN

The Commodore 64 Whizz Kid is for kids from six upwards and their parents. It's a book of programming projects that are great to look at and easy to create. There's computer music, drawing, animation, games of chance and guess work and much, much more.

MAKE IT EASY

You can learn a lot from these projects. Between them they cover the basics of programming. But that doesn't mean heavy reading. The Commodore 64 Whizz Kid makes light work of it all. There are colour pictures and diagrams on every page, plenty of handy hints - and good listings.

MAKE IT WORK

The listings are the starting point for on-screen action - and plenty of it. They're the starting point for your own ideas as well. Follow the Whizz Kid projects and you'll find your programming muscles start expanding - fast.

THE BOOK TOMORROW'S PROGRAMMER NEEDS TODAY.

COMMODORE 64 WHIZZ KID

Longman
Computer
Books

ISBN 0-582-91609-7

9 780582 916098