INTRODUCTION TO
COMPUTER PROGRAMMING

BASIC for beginners



Usborne Hayes Electronics

NO COMPUTER NEEDED

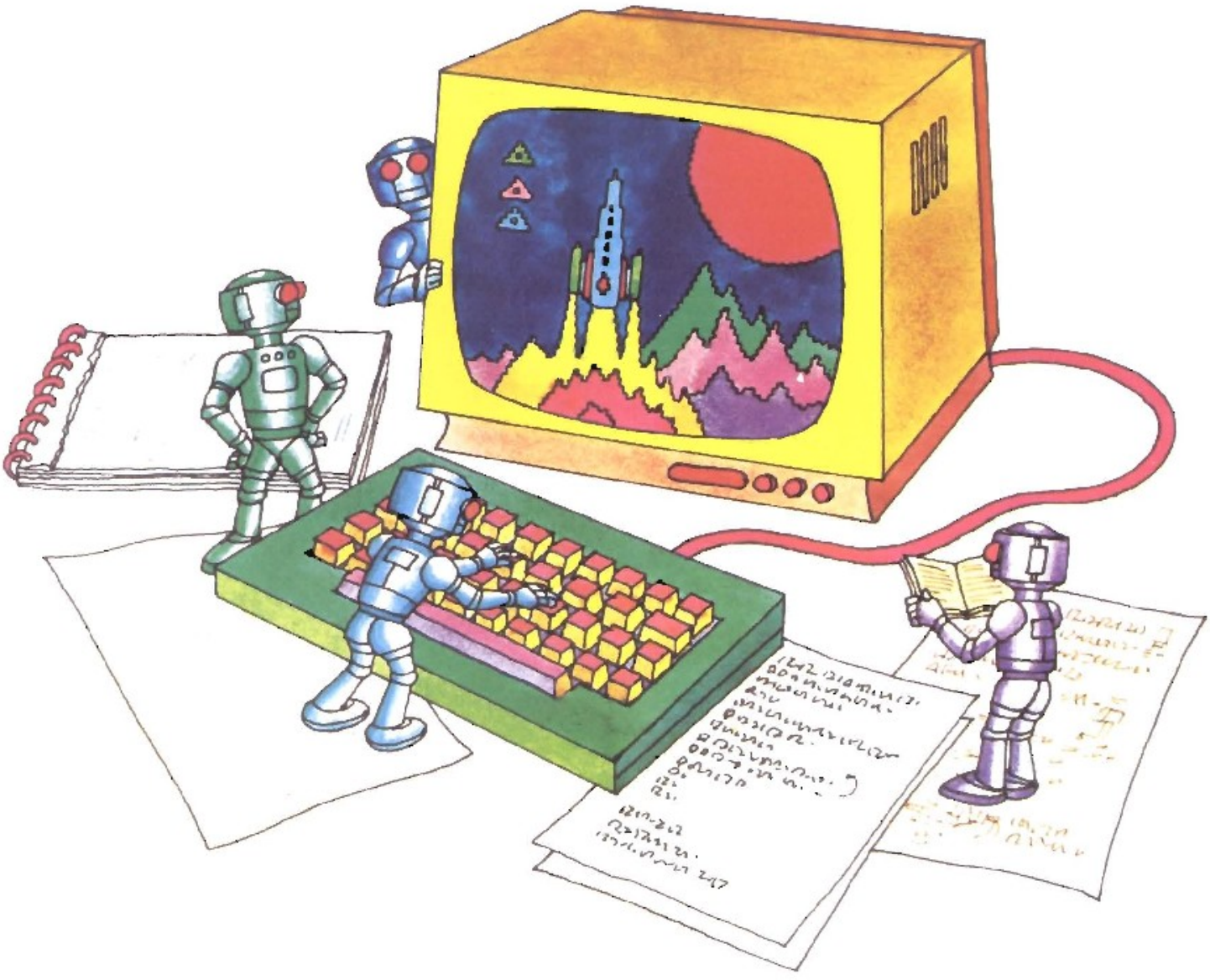
INTRODUCTION TO
COMPUTER PROGRAMMING

Brian Reffin Smith

Edited by Lisa Watts

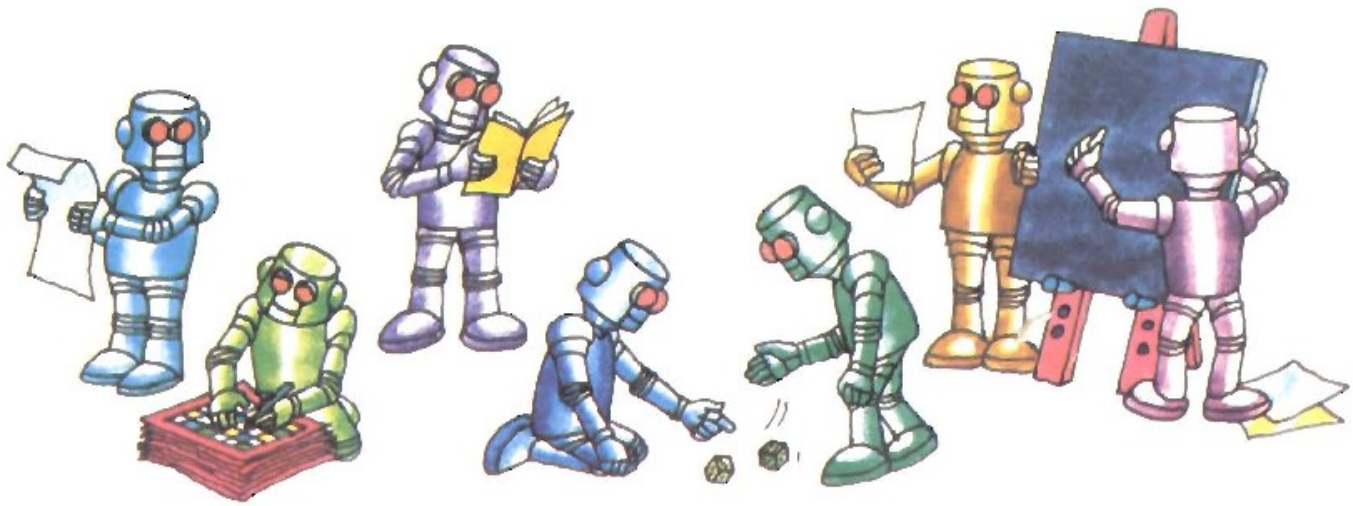
Designed by Kim Blundell

Illustrated by Graham Round
and Martin Newton



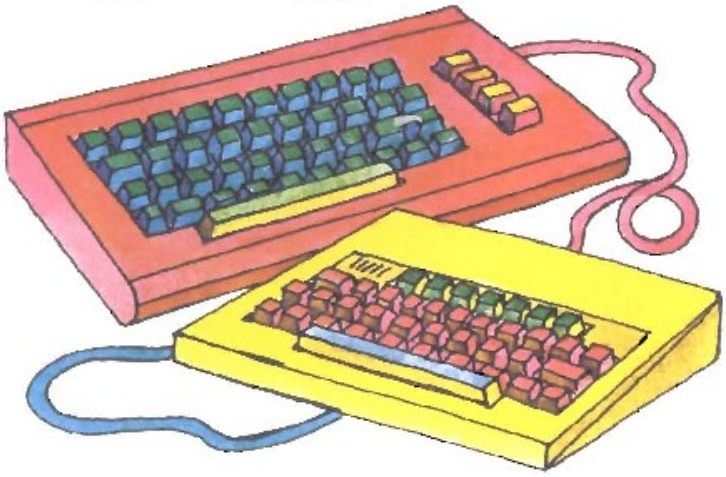
Contents

4 How a computer works
6 Giving the computer instructions
8 Writing programs
10 First words in BASIC
12 Giving the computer information
14 Using INPUT
16 Doing things with PRINT
18 How computers compare things
20 Programs with lots of BASIC
22 Drawing pictures
24 Playing games
26 Making loops
28 Tricks with loops
30 Subroutines
32 Doing things with words
34 Graphs and symbols
36 More graphics
38 Funny poems program
42 Programming tips
44 Puzzle answers
46 BASIC words
48 Going further and Index



About this book

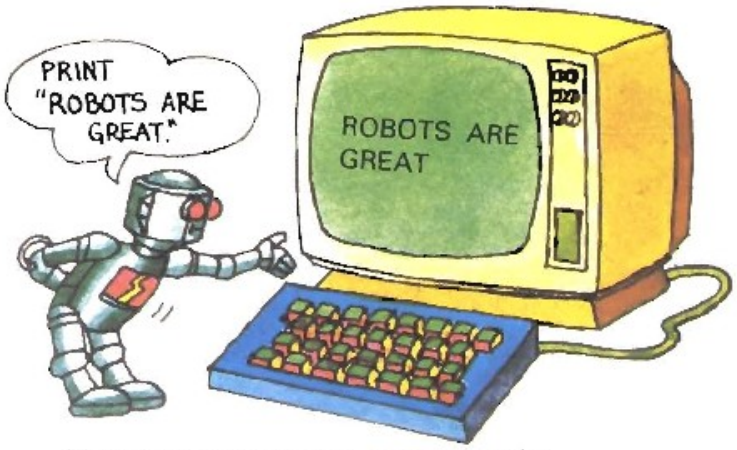
This is a guide to writing computer programs in BASIC for absolute beginners. BASIC is the language used on most home computers. It is a way of writing instructions for a computer in a form the computer can understand.



You do not need a computer to use this guide, though of course it helps you to understand the programs if you can try them out on a computer. Different makes of computer use slightly different versions of BASIC. Nearly all the terms in this book, though, will work on most microcomputers, and the few that are not standard are clearly marked.

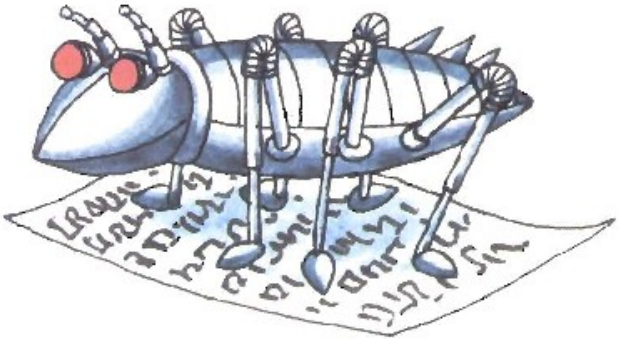


At the beginning there are some guidelines to programming a computer. Then, as you read through the book, the main BASIC words are introduced one by one, with short programs to show how they work.



To give you some practice in writing programs there are program puzzles to solve and suggestions for programs to write and for useful alterations you can make to the programs in this guide. The answers to the program puzzles are on pages 44-45.

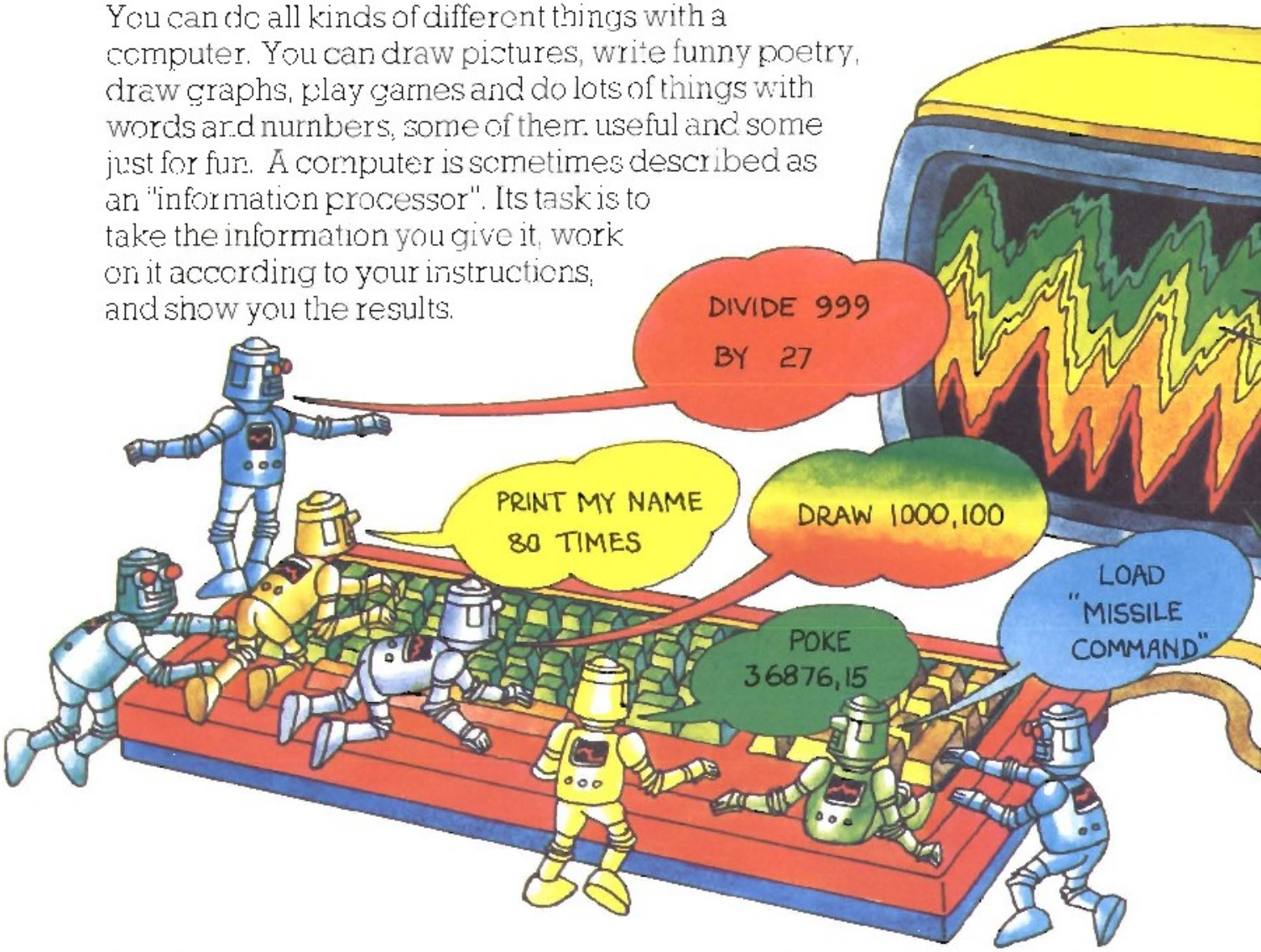
At the end of the guide there is a list of BASIC terms and other computer words with brief explanations. There are also some guidelines to help you write programs, and a list of "bugs" – the mistakes in programs which stop them working – with hints to help you recognize them.



If you have a micro, try out the programs in this guide, then, to find out more about how your micro works, look up the BASIC terms in your manual. You may find that some of the rules given here are not necessary on your micro. The best way to learn BASIC is to try out lots of programs from books and magazines, then alter them a little to see what happens. From there you will soon be writing your own programs.

How a computer works

You can do all kinds of different things with a computer. You can draw pictures, write funny poetry, draw graphs, play games and do lots of things with words and numbers, some of them useful and some just for fun. A computer is sometimes described as an "information processor". Its task is to take the information you give it, work on it according to your instructions, and show you the results.



To make a computer do what you want you have to give it very precise instructions. A list of instructions for a computer is called a program* and the information you give the computer to work on is called data. The program has to be written in a language, such as BASIC, that the computer can understand, and it must follow all the rules of the language too.

Microcomputers

Most micros consist of a keyboard which you plug into a TV set. You give the micro instructions and information by typing on the keyboard and everything you type, along with the computer's results, is displayed on the TV screen.

Some micros have a small, built-in display screen, like a pocket calculator. A few use a special screen called a monitor. A monitor is like a TV but it cannot pick up the signals from TV stations.

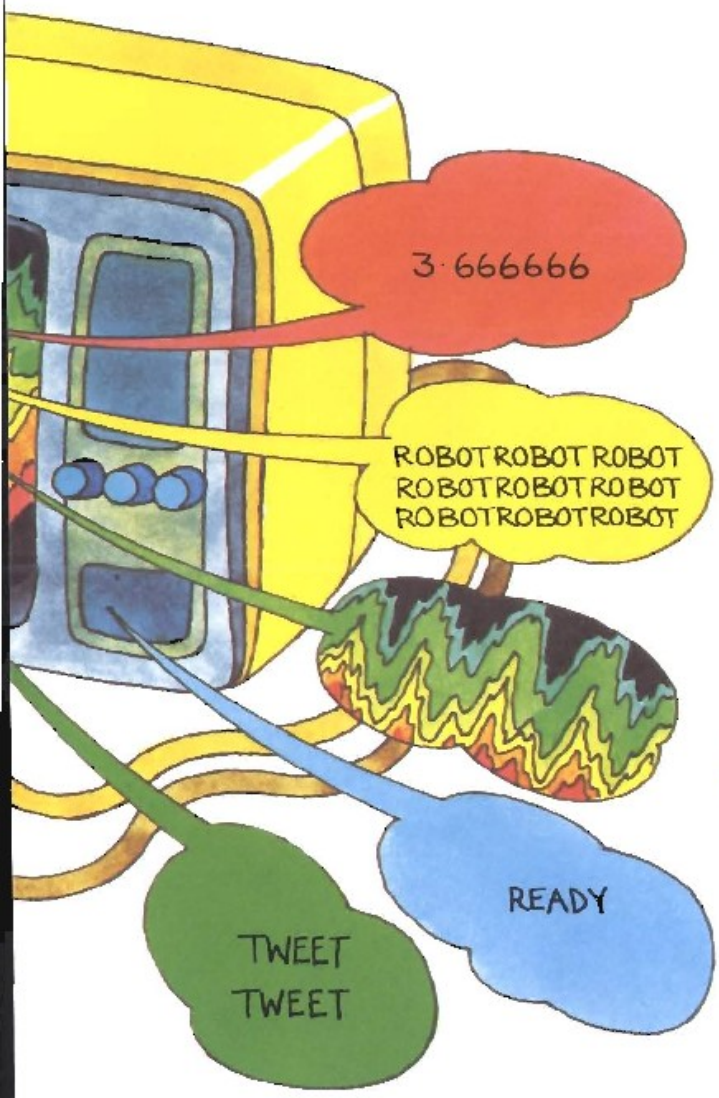


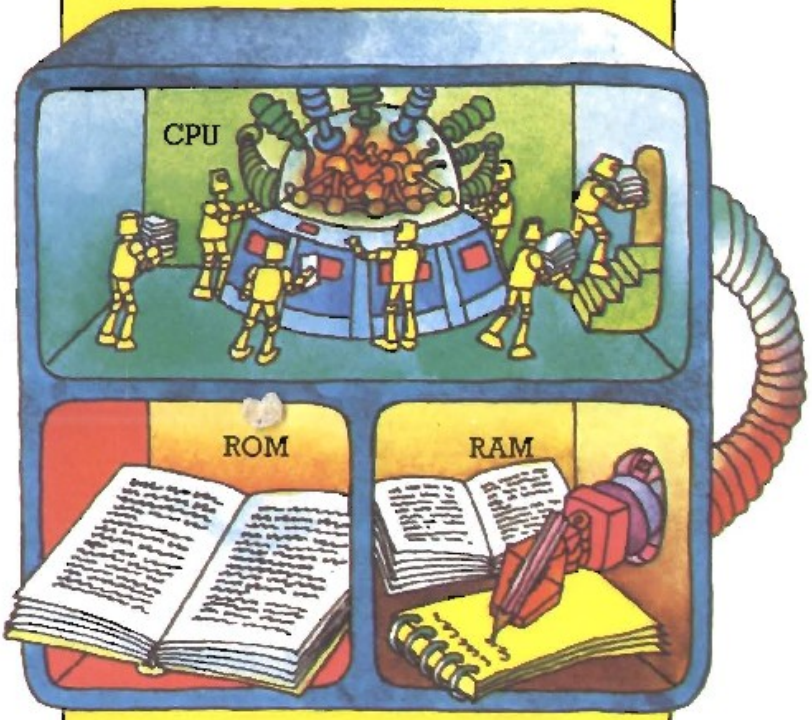
The keyboard of a micro looks like a typewriter keyboard with some extra keys. On some micros each key gives the computer a separate instruction in BASIC so you do not have to type the words in letter by letter.

*Spelt like this when used for a computer.

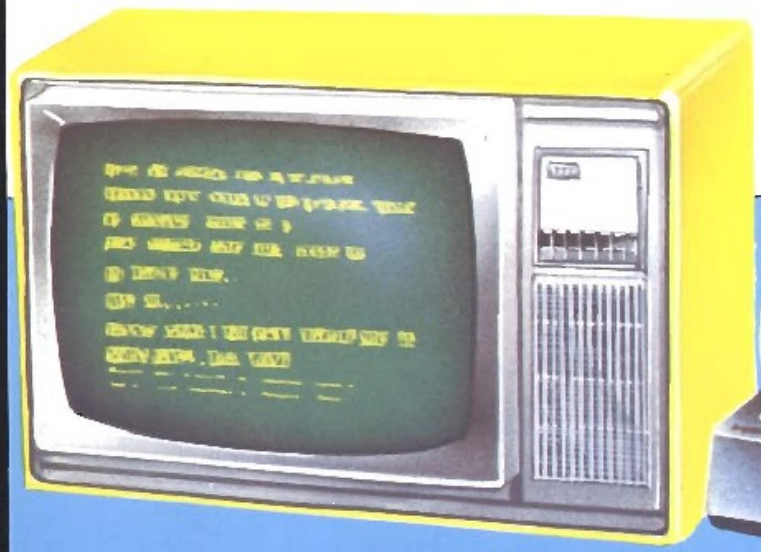
Inside a micro

A micro is made up of two main parts: the central processing unit (CPU) where all the work is done, and the memory where programs and data are stored.

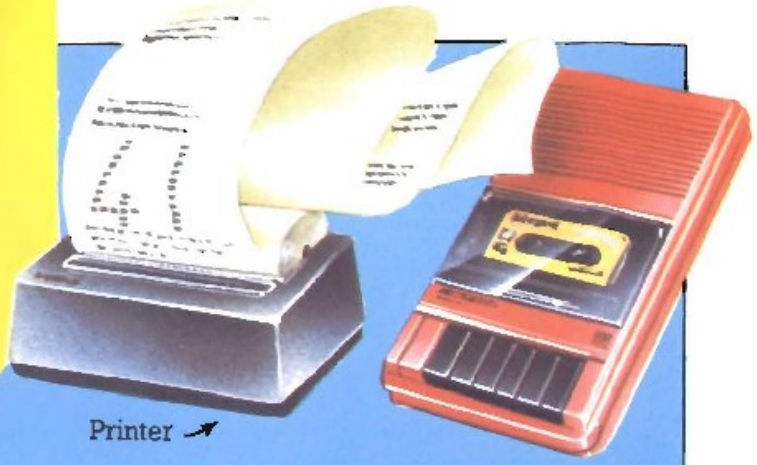




In fact, the computer has two memories. One, called ROM, contains a program which controls all the operations of the computer. The other, called RAM, is an empty memory where your programs and data are stored. When you switch off the micro all the information in RAM is lost, but the ROM program is permanent.



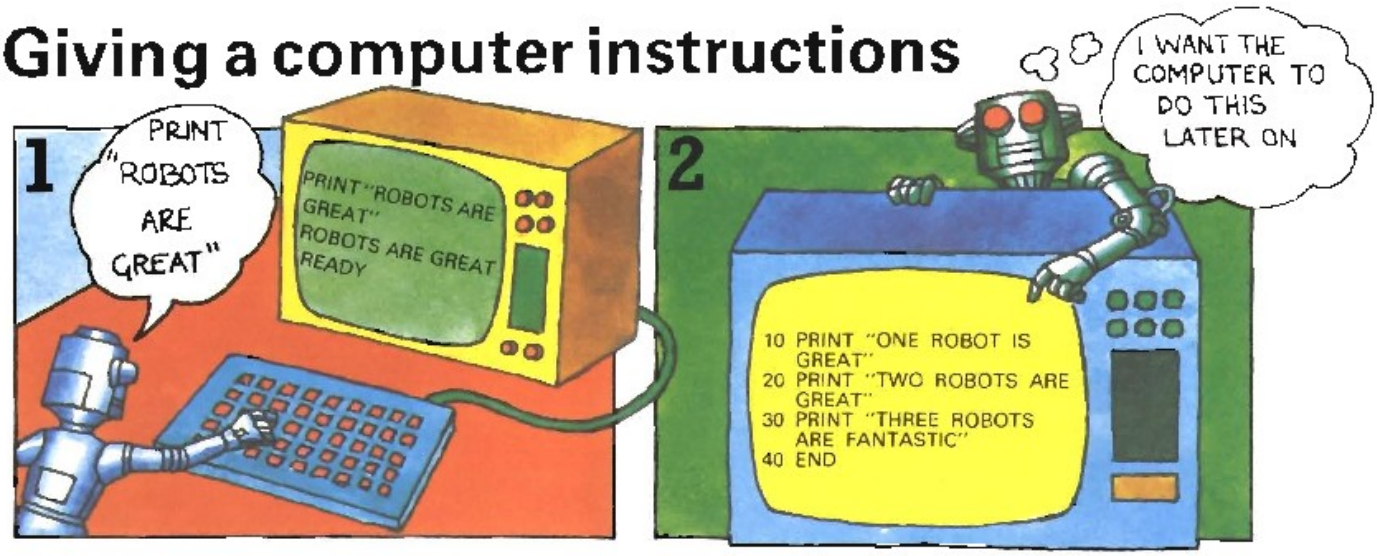
A TV screen is the most usual way to display the information from a micro. You can also print it out on paper, using a printer. This is useful as the information in the micro and on the TV screen is lost when you switch them off.



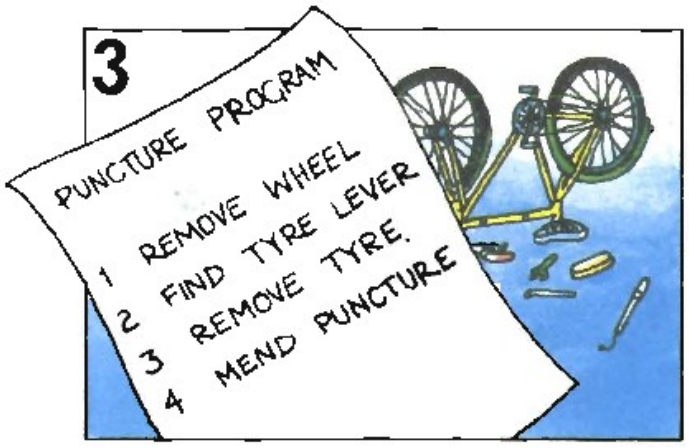
Printer

Another way to store information from a micro is with a cassette recorder. You can store programs and data on a cassette, then load them back into the micro when you want to use them.

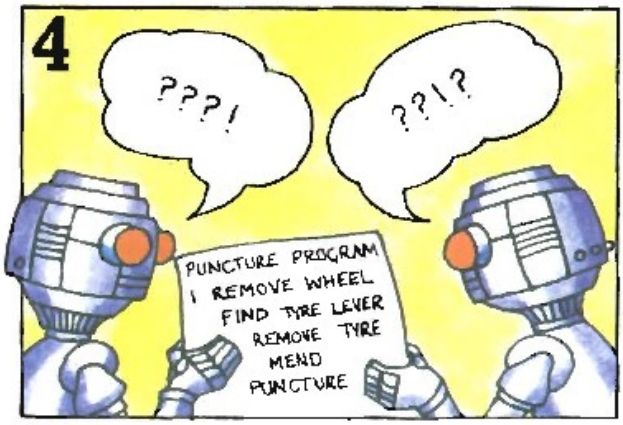
Giving a computer instructions



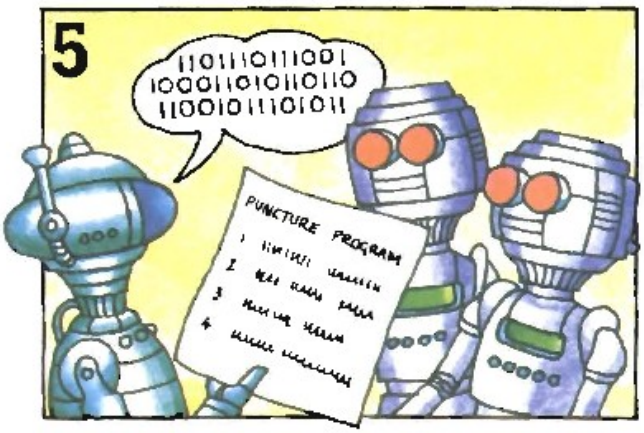
To make the computer do something, you have to type in an instruction it understands. This instruction can be a direct command which it carries out straight away, or it can be a program of instructions which it stores in its memory and does not carry out until you give it the go-ahead.



The instructions in a program have to be very carefully worked out. The computer will attempt to carry out your instructions precisely, even if they are wrong.



The computer cannot understand instructions written in our language, so you have to write them in one of the many computer languages. Some of these languages are described opposite.



All the work inside the computer is done with a code of tiny pulses of electricity. Your instructions are translated into computer code by a special program inside the computer called the interpreter.



Each piece of information in computer code is represented by patterns of pulses. Computer code can be written down using 1 to represent a pulse and 0 to show there is no pulse.

Computer languages

You could write programs in computer code but it would be very difficult. Instead, there are special computer languages, called high level languages, which the computer can translate into its own code.

There are hundreds of different high level languages, many of them specially designed to do one particular kind of work. BASIC is one of the most common languages. The letters stand for Beginner's All-purpose Symbolic Instruction Code. It is not just used by beginners though. Below there are examples of three different languages.

Names program

```
10 PRINT "WHAT IS
   YOUR NAME"
20 INPUT N$
30 IF N$="BRIAN"
   THEN PRINT "HELLO"
40 IF N$="GEORGE"
   THEN PRINT "GO AWAY"
50 END
```

Money program

```
PROCEDURE MONEY;
VAR MONTH:INTEGER
BEGIN (*CALCULATE*)
  FOR MONTH:=1 TO
  TOTALMONTHS DO
  AMOUNTDUE:=
  (AMOUNTDUE-REPAY)*
  (1+MONTHLYINTERESTRATE)
END;(*MONEY*)
```

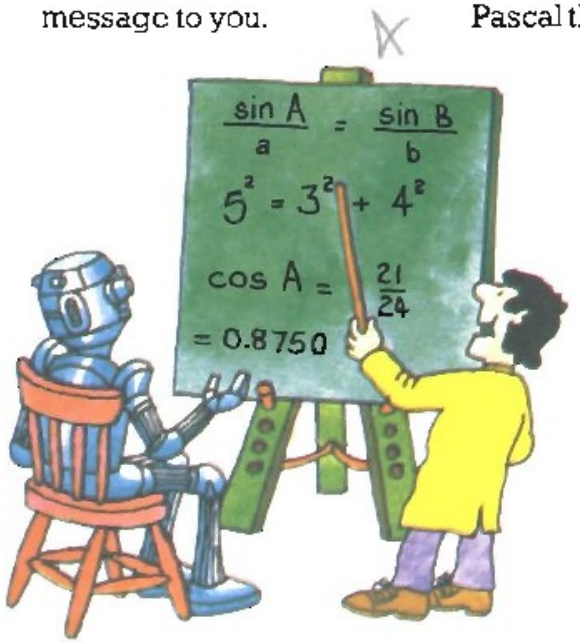
Geography program

```
T: WHAT IS THE CAPITAL OF FINLAND
*INPUT A: $A$
T(LEN(A$)=0):GO ON, HAVE A GUESS
JC: @A
M: COPE!STOCK!OSLO!GOTHEN!LAP
TY: CLOSE BUT NOT CLOSE ENOUGH - TRY
    AGAIN
JY: @A
M: HELSINKI! HELSING
TY: GOOD - IT IS HELSINKI.
```

This is a short program in BASIC. Line 10 tells the computer to print "What is your name" on the screen. Then the computer stores your answer in its memory and if your name is Brian or George, it prints out a message to you.

This program is written in Pascal, a language named after a famous French mathematician. It is part of a program to work out details about money. Many people think it is easier to write good, neat programs in Pascal than in BASIC.

This is a language called PILOT. It is used to write programs to help people learn new subjects. In this language, the computer can recognize answers even if they are not exactly right.





11. Bb3, Ne5
12. 0-0-0, Nc4
13. Bxc4, Rxc4
14. h5, Nxh5



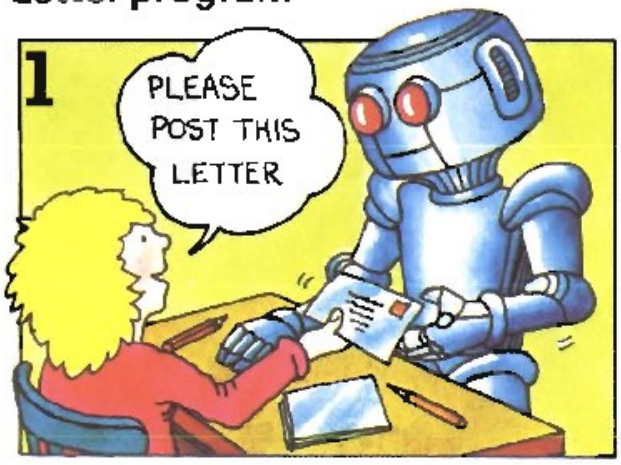
At first glance, computer languages seem very strange and difficult, but then, so do other languages such as the Finnish shown on the right, until you get to know them. There are lots of other subjects too, in which special languages are used. For instance, in mathematics a special notation is used to write down ideas and formulae which would need a lot of ordinary words to explain them and other kinds of notation are used to write down chess moves or music.

*Minus fifteen I guess.

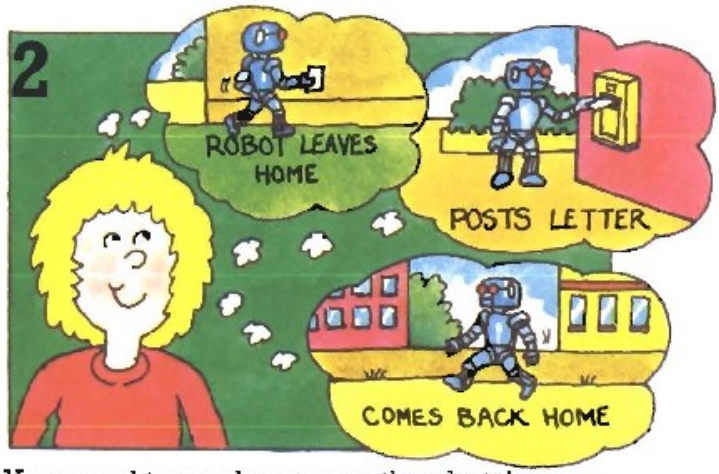
Writing programs

A program is like the rules for a game, or the recipe for a cake. If there is a mistake in the rules, or the recipe, you will not be able to play the game properly, or bake a good cake. In the same way, the results you get from a computer depend on the instructions you give it. To write a program for a computer you first need to study what you want to do very carefully and work out the main steps needed to achieve the result you want.

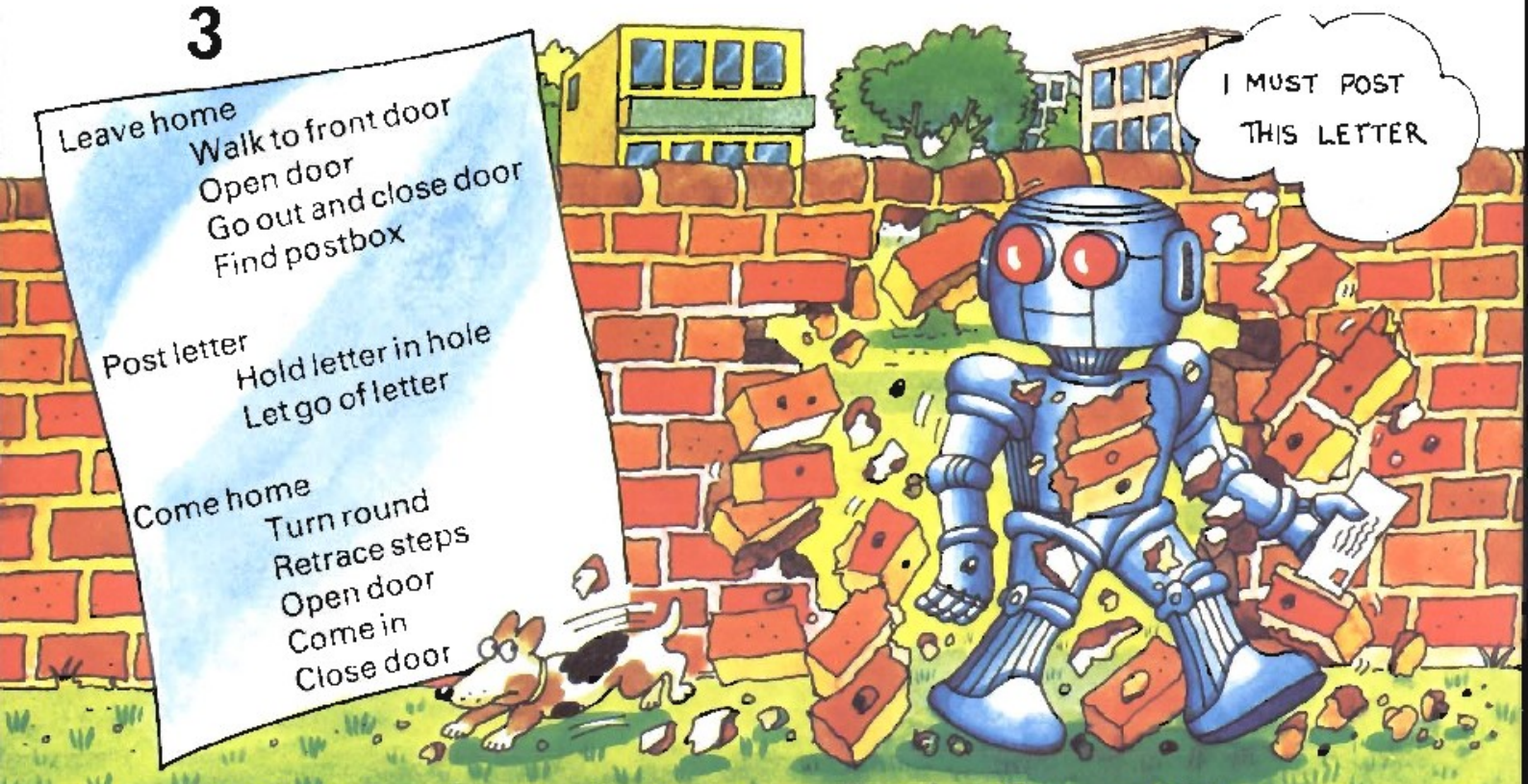
Letter program



Imagine trying to write a program to tell a robot to post a letter. A simple instruction as shown above would be too difficult for the robot's computer brain to understand.



You need to work out exactly what the robot needs to do to post the letter. Its computer needs instructions telling it what to do at every stage.

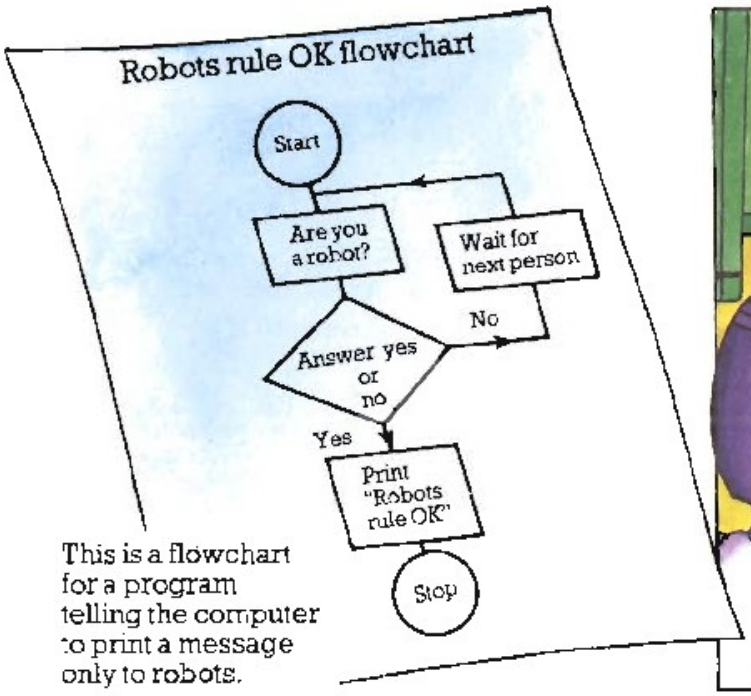


To write the program you need to break down the instructions for each stage into even smaller steps which can be translated into a language the robot can understand.

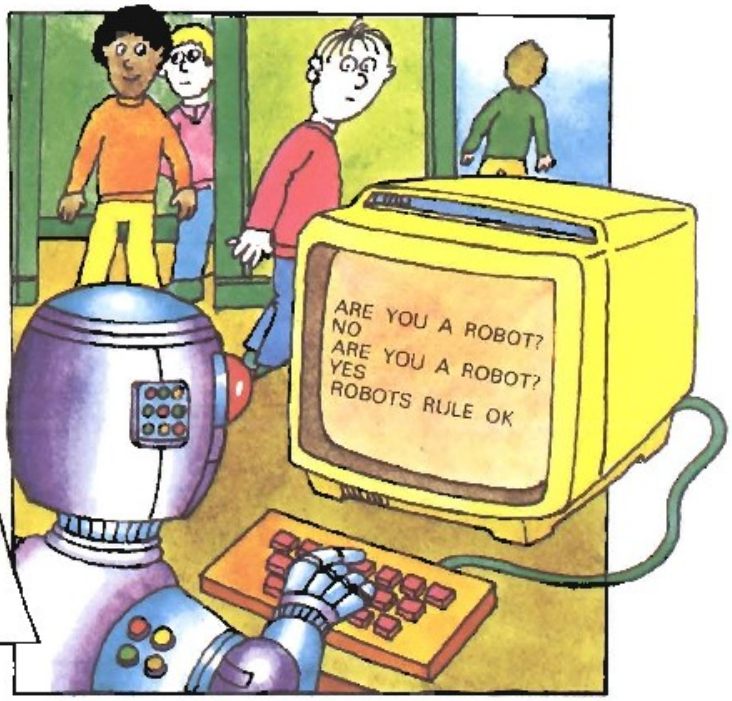
The robot will attempt to follow your instructions even if they are wrong, or incomplete. Mistakes in a program are called bugs and they can sometimes lead to unusual results from the computer.

Program diagrams

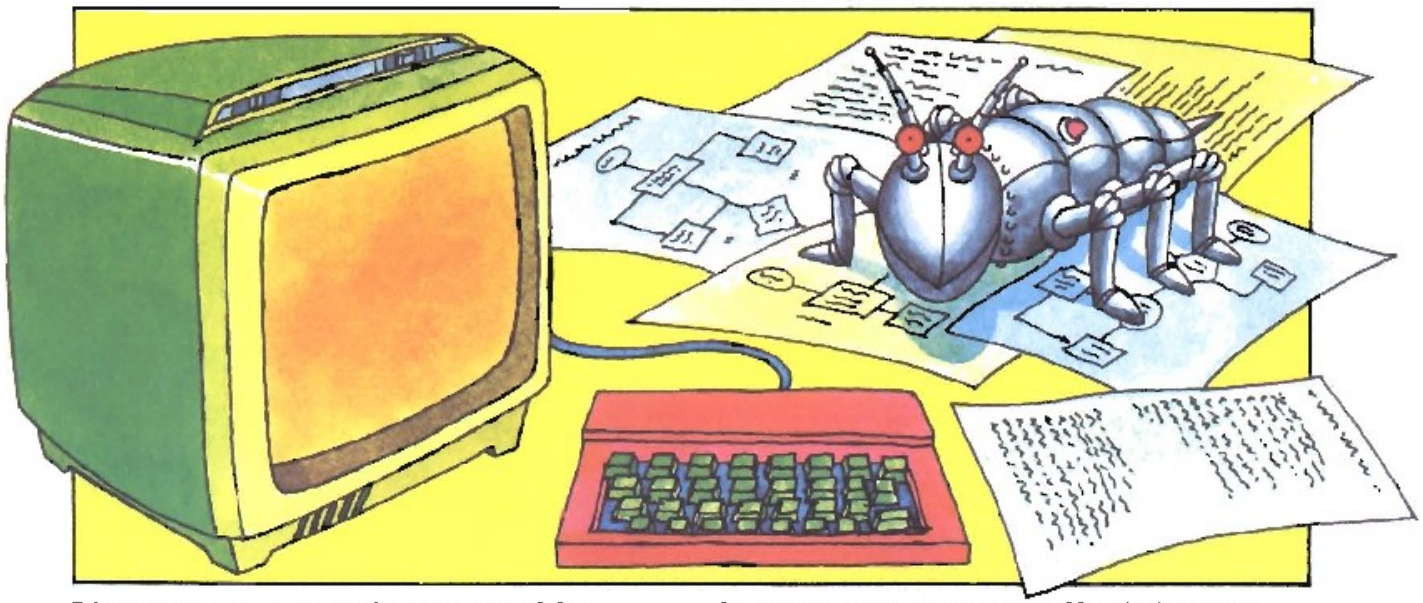
When you are writing a program it sometimes helps to draw a diagram like the one below, showing the main steps you need to solve the problem. A diagram like this is called a flowchart. It shows each of the steps the computer needs to carry out, and the order they should come in.



This is a flowchart for a program telling the computer to print a message only to robots.



A flowchart has different shaped boxes for different steps in the program. The beginning and end of the program have round boxes, instructions telling the computer to do something are in rectangular boxes and decision boxes, where the computer can do different things depending on the information it receives, are in diamond-shaped boxes. The lines show the possible routes the computer can follow.



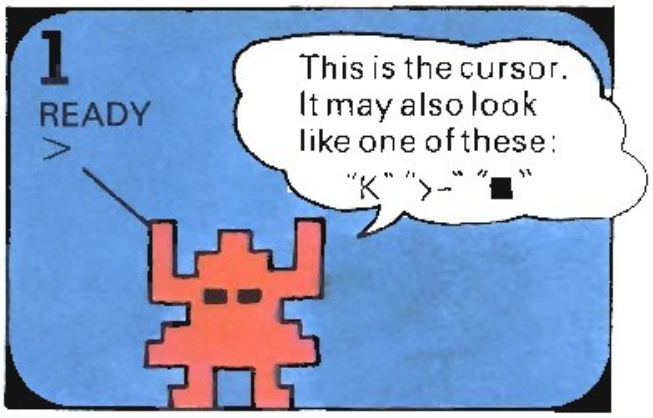
After working out all the details of the program you can translate it into BASIC and test it on the computer. The program will probably not work straight away though, as there will probably be some bugs in it. These may be typing mistakes made when you typed the program into the computer, or errors of logic in your program. Before you can get the program to work you have to find all the bugs and correct them.* Sometimes, a bug makes a program produce a slightly different result which you may prefer. Useful bugs like this are called "pugs".

*There are some tips to help you find bugs on pages 42-43.

First words in BASIC

Lots of the words in BASIC are based on English words and it is quite easy to guess what they mean. For instance, PRINT means "display on the screen". RUN means "carry out this program" and INPUT means "give the computer information". On these two pages you can find out how to use the word PRINT.

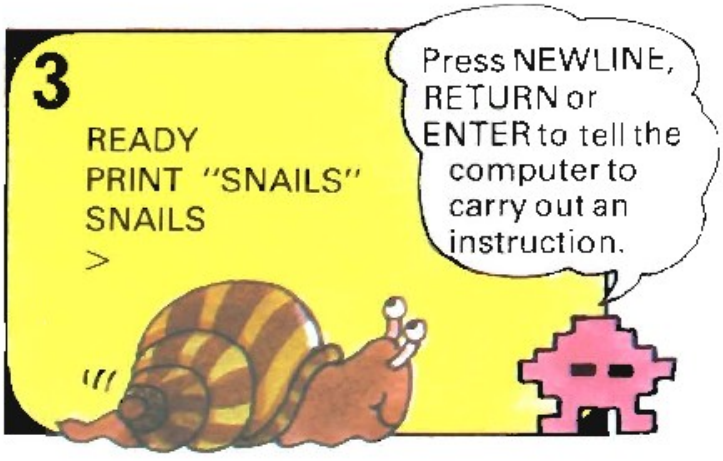
Most home computers have a BASIC language interpreter inside them already and when you switch them on they are ready to be programmed in BASIC.*



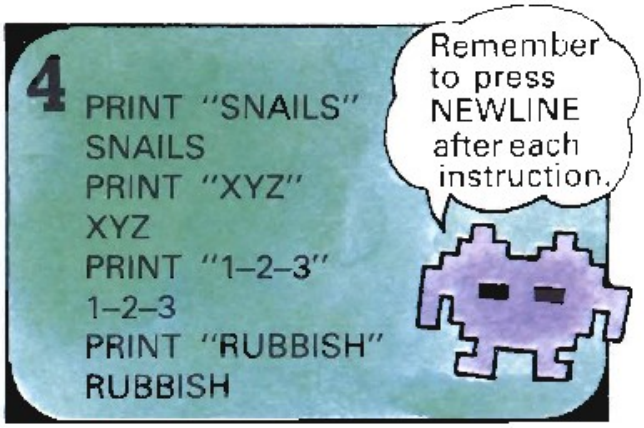
When you switch on a micro some words are usually displayed on the screen automatically, along with a small symbol called the cursor. The cursor shows where the next letter you type will appear.



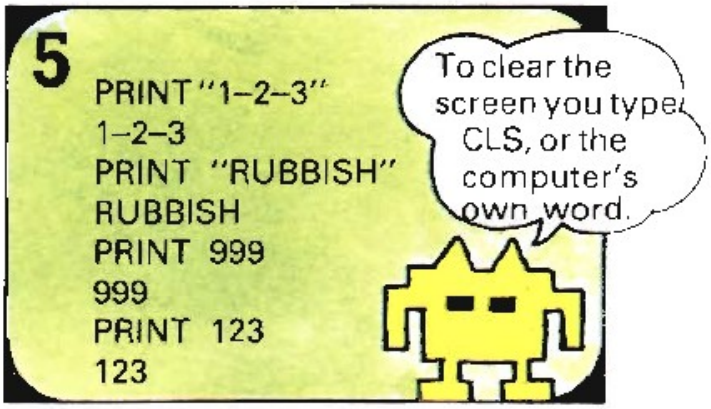
To tell the computer to display words on the screen you use PRINT with the words you want in quotation marks. For instance, PRINT "SNAILS" tells it to display the word SNAILS on the screen.



The computer will not carry out your instruction, though, until you press NEWLINE (or RETURN or ENTER – it varies on different computers) to tell it the instruction is complete.



The computer will display on the screen whatever you type between the quotation marks. It can be letters, numbers, words or symbols. Note that it does not display the quotation marks themselves.



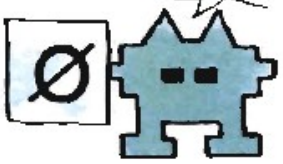
To display numbers by themselves, you do not need to use quotation marks. Now, to clear the screen you type CLS on most micros. (Check this in your manual if you have a computer.)

*Some computers have to have a special program loaded from cassette tape before they understand BASIC.

A program in BASIC

In a program, each line of instructions starts with a number. This tells the computer to store the instructions in its memory and not to carry them out until you give the go-ahead. On the opposite page, the instructions to the computer did not have numbers, so the computer carried them out straight away. Here is a short program which makes the computer display symbols in the shape of a face on the screen.

On some computers the figure 0 has a line through it, like this.



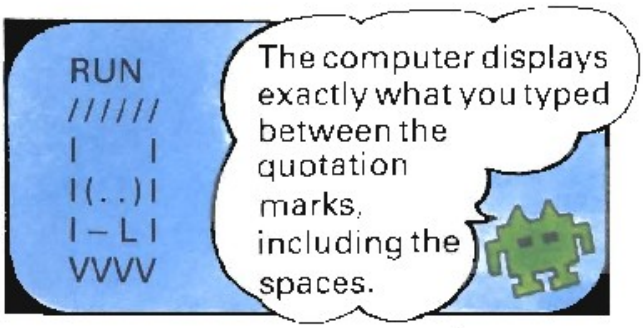

```
10 PRINT "//////"
20 PRINT "I    I"
30 PRINT "I(..)I"
40 PRINT "I -L I"
50 PRINT "VVVV"
60 END
```

Line numbers usually go up in tens so you can add extra instructions without renumbering the whole program.

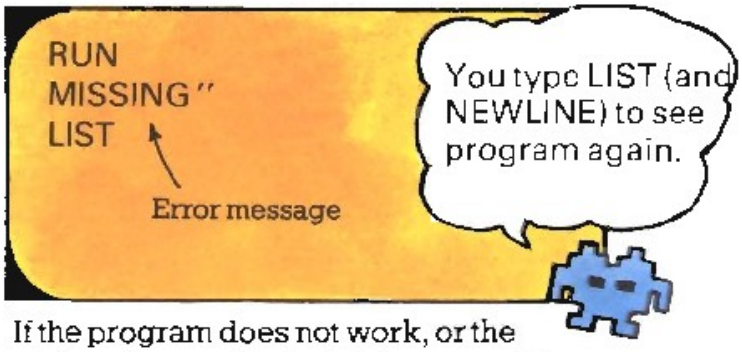


Many computers do not need this line.

When you type in a program you have to press NEWLINE (or the computer's word) at the end of each line. The lines are displayed on the screen but the computer does not carry out the instructions until you tell it to by typing RUN. Be careful not to mix up the letter O and the figure 0 as this will cause a bug. Most computers have a RUBOUT or DELETE key for correcting typing mistakes.



When you have typed in all the lines, check them carefuly to make sure there are no mistakes. Then, to tell the computer to carry out the program, you type RUN, followed by NEWLINE.



If the program does not work, or the picture does not look right, you need to display the program again to find the bug. To do this you type LIST. The computer may give you an error message telling you what the bug is.

1 Debugging programs

```
RUN
MISSING"
LIST
10 PRINT "//////"
20 PRINT "I    I"
30 PRINT "I(..)I"
40 PRINT "I -L I"
50 PRINT " VVVV
60 END
```

No quotes

The computer will give you an error message for most bugs. The error messages are explained in the computer's manual. The easiest way to correct a mistake is to type the whole line again. The computer will replace the old line with the new one. To get rid of a line from the program altogether, just type the line number, followed by NEWLINE. Each computer also has its own way for correcting or altering parts of lines, using words such as EDIT or COPY. This is explained in the computer's manual.

2

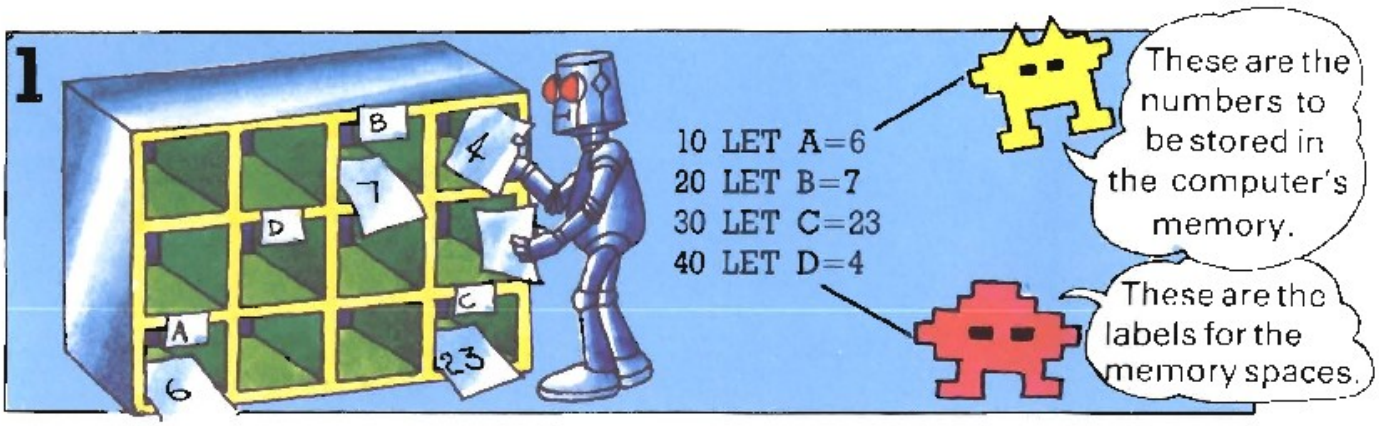
```
RUN
MISSING"
LIST
10 PRINT "//////"
20 PRINT "I    I"
30 PRINT "I(..)I"
40 PRINT "I -L I"
50 PRINT " VVVV
60 END
50 PRINT " VVVV "
```

Line typed in again correctly.

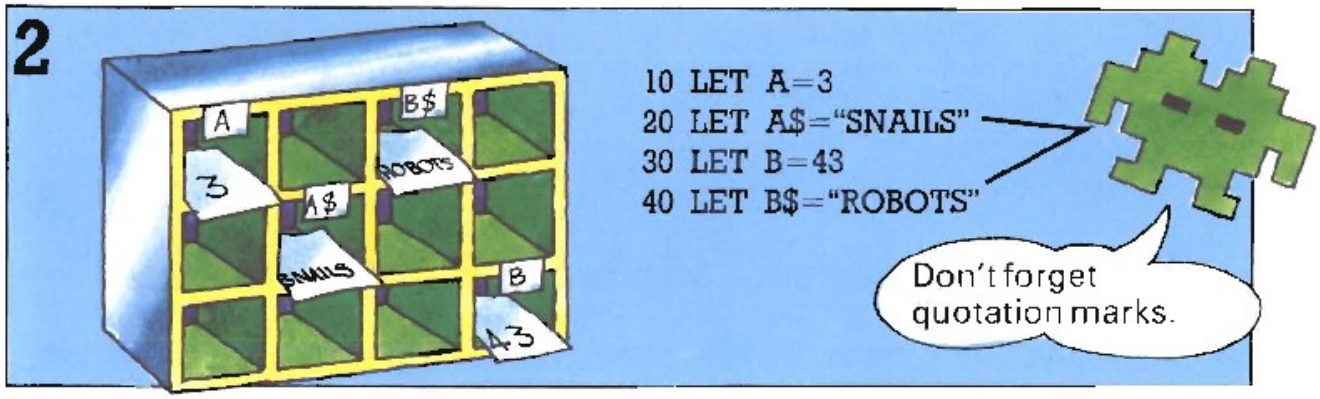
Program puzzle – Try changing the program to give the face different features.

Giving the computer information

To make the computer do something more useful than just displaying things on the screen you have to give it information or "data" to work on. The computer stores this information in its memory until you tell it to use it.

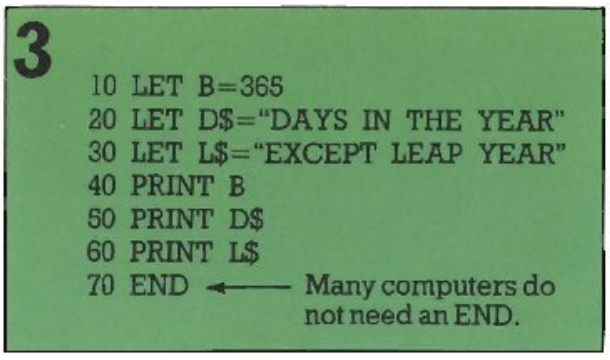


When you put a piece of data into the computer's memory you have to give it a label so you can find it again. You can use letters of the alphabet as labels. To label a memory space and put a number in it you can use the word LET, as shown above. A labelled memory space is called a variable because it can hold different data at different times in the program.

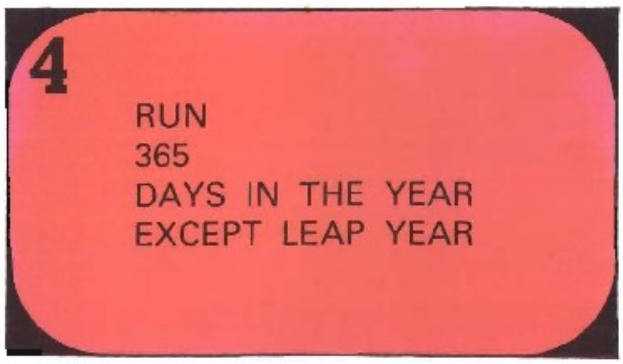


You use a different kind of label to store letters and symbols in memory spaces. Letters and symbols are called "strings" and you use letters of the alphabet with dollar signs to label them, e.g. C$.*

You put a string in a memory space using LET in the same way as for a number variable, but the letters and symbols must be enclosed in quotation marks, as shown above.



To display the information on the screen you use the word PRINT with the name of the variable, e.g. PRINT A$. This short program prints out the information from variables B, D$ and L$.



You can run the program as many times as you want. Each time the computer will print out the same information. The data in the variables stays the same until you change it.

*This is pronounced "C dollar" or "C string".

Another way

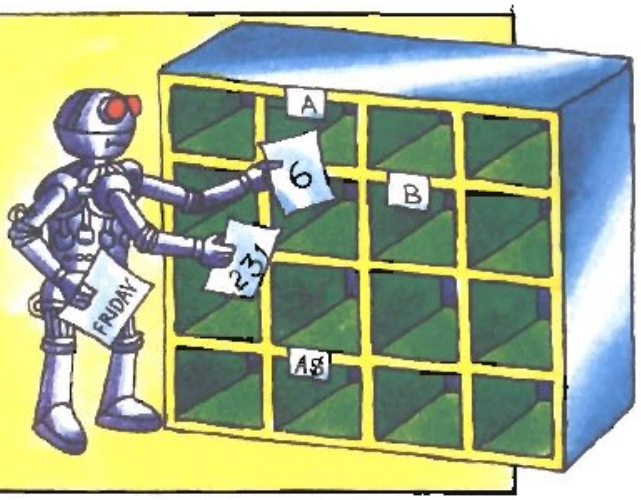
```
10 READ A
20 READ B
30 READ A$
40 DATA 6, 231, FRIDAY
```

You must have the correct kind of label for numbers and letters.

Commas



Some computers need their data words in quotes.



Another way to store information is with the words READ and DATA, as shown above. The READ lines tell the computer to label memory spaces and the DATA line contains the information.

When you run the program the computer puts each piece of data in a memory space, taking them in order. The items of data must have commas in between so the computer knows how long each one is.*

Some programs

1

```
10 READ Q
20 READ X$
30 DATA 24,CHEESE BURGERS
40 PRINT Q
50 PRINT X$
60 END
RUN
24
CHEESE BURGERS
```

Comma

This is one item of data, including the space.

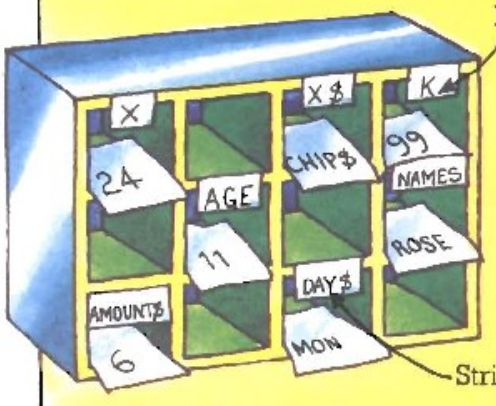
2

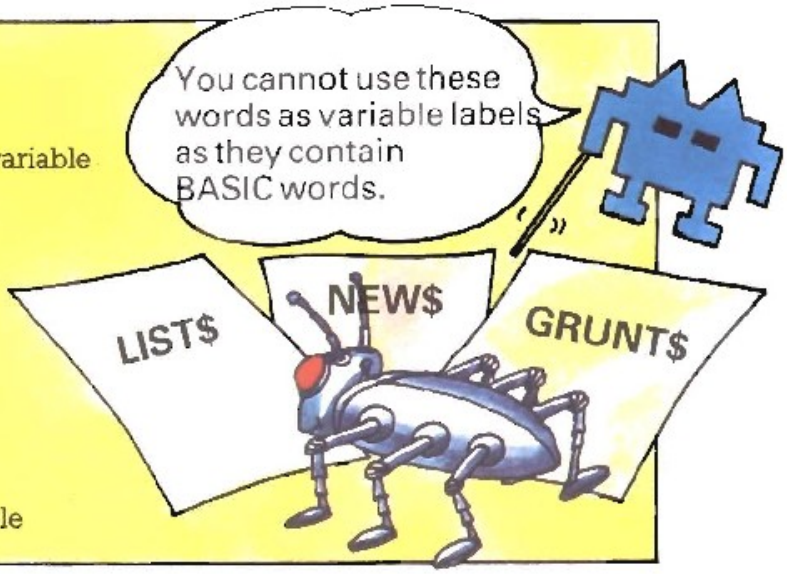
```
10 LET A$="ROBOTS ARE GREAT"
20 LET B$="IF YOU LIKE"
30 LET C$="GREAT METAL IDIOTS"
40 PRINT A$
50 PRINT B$
60 PRINT C$
70 END
RUN
ROBOTS ARE GREAT
IF YOU LIKE
GREAT METAL IDIOTS
```

Quotes

Here are two programs, one using READ and DATA and the other using LET to store information in the computer's memory.

More about variables



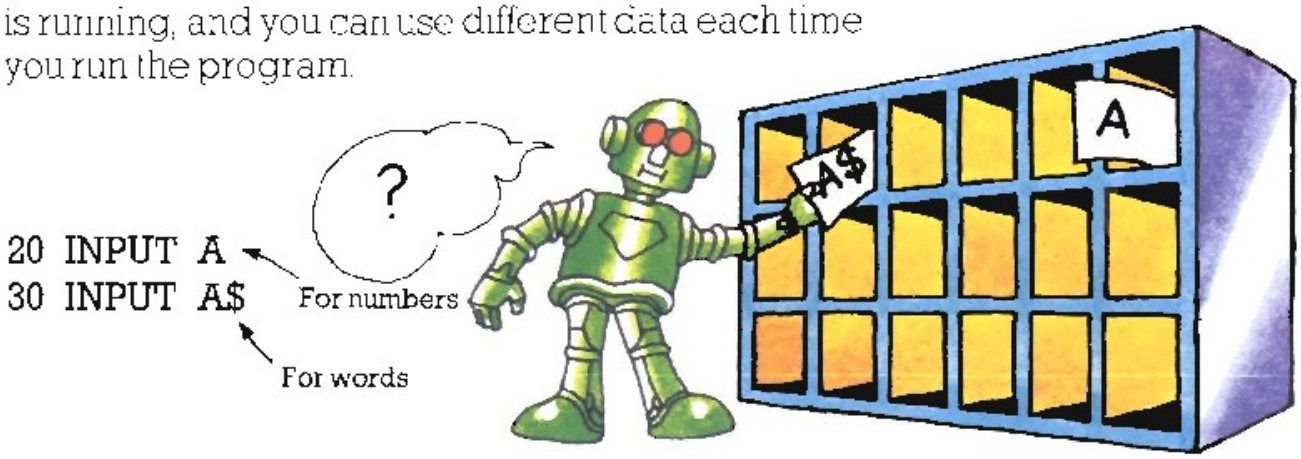


Variables are labelled spaces in the computer's memory where information is stored. A variable containing numbers is called a number variable and one which contains letters and symbols is called a string variable. The contents of variables can change during the program. Some computers can use words as labels for variables, but not words which contain BASIC words as this would confuse the computer.

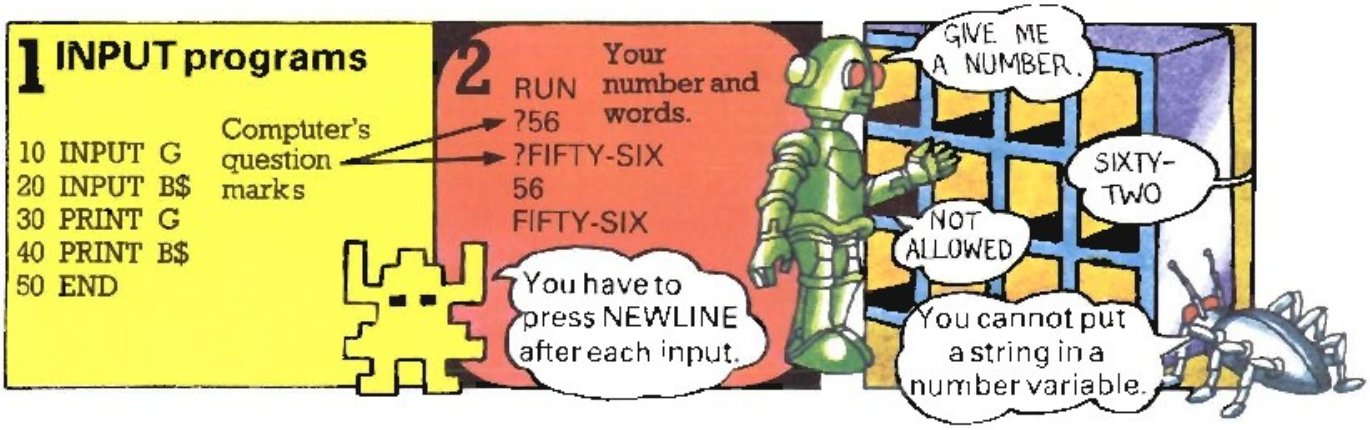
*You cannot use this method on the ZX81 computer.

Using INPUT

Another way to give the computer data is with the word INPUT. This lets you put in information while the program is running, and you can use different data each time you run the program.



You use INPUT with a label such as A for a number and A$ for a string. When the computer meets the word INPUT in a program it puts the label on a memory space and asks you for the data, usually by printing a question mark, or other symbol, on the screen. Then you type in the data and the computer stores it in the memory space and goes on with the rest of the program.



Picture 2 shows what happens when you run this program. When the computer meets the word INPUT in line 10 it prints a question mark on the screen and waits for you to type in a number for G. Then it prints another question mark for the INPUT instruction in line 20. This time you have to type in words or symbols as the label B$ told the computer to expect a string.



If you have a computer, try typing in this program, then press RUN to start it off. When the computer asks you for information, type in your name and age, or a silly name and crazy number, as shown in the sample run above. Try it lots of times with different data, pressing RUN to start the program again each time. The computer always prints exactly what you put in N$ and A.

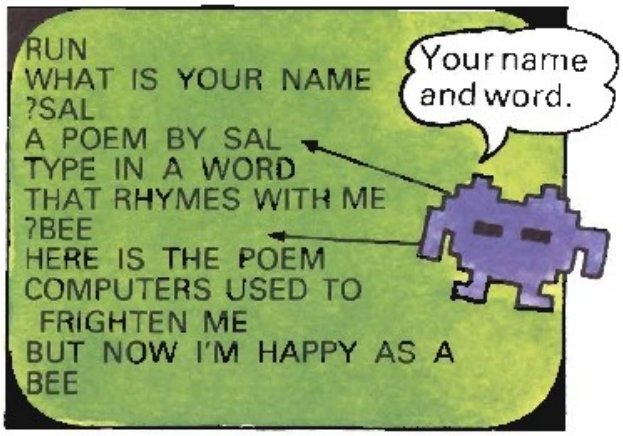
Poetry writing program

Now you know enough BASIC to write a poem on a computer. Here is a poetry writing program which uses PRINT and INPUT.

```
10 PRINT "WHAT IS YOUR NAME"
20 INPUT N$
30 PRINT "A POEM BY"
40 PRINT N$
50 PRINT "TYPE IN A WORD"
60 PRINT "THAT RHYMES WITH ME"
70 INPUT A$
80 PRINT "HERE IS THE POEM"
90 PRINT "COMPUTERS USED TO
   FRIGHTEN ME"
100 PRINT "BUT NOW I'M HAPPY AS A"
110 PRINT A$
120 END
```

This line prints out your name.

This line prints out your word.





You type run to try it again with another word.

The program makes the computer ask you your name, then store your reply in N$ and print it out at line 40. It stores the word you choose in A$, then prints it out as part of the poem at line 110. If you have a computer try running the program lots of times, inputting different words at line 70.

Program puzzle

Can you write a program to get the computer to ask you your name and then print hello, followed by your name and a message to you?

Checklist for typing in programs

1. Before typing in a new program type NEW. This clears any old programs and variables out of the computer's memory.
2. When you are typing in the program, remember to press NEWLINE, or your computer's word, at the end of each line.
3. After typing in the program, check all the lines on the screen to see if there are typing mistakes. Make sure none of the lines are missing, too.
4. Next you can type CLS (or your computer's word) to clear the program off the screen. Then type RUN to start the program.
5. To get the program listing back again to check it or alter a line, type LIST. To display one particular line you can usually type LIST with the line number, but check this command as it varies slightly on different computers.
6. To stop the program while it is running type BREAK or ESCAPE. Check this command in your manual, though, as it varies on different computers. On some computers ESCAPE wipes the whole program out of the computer's memory. To start the program again type RUN.

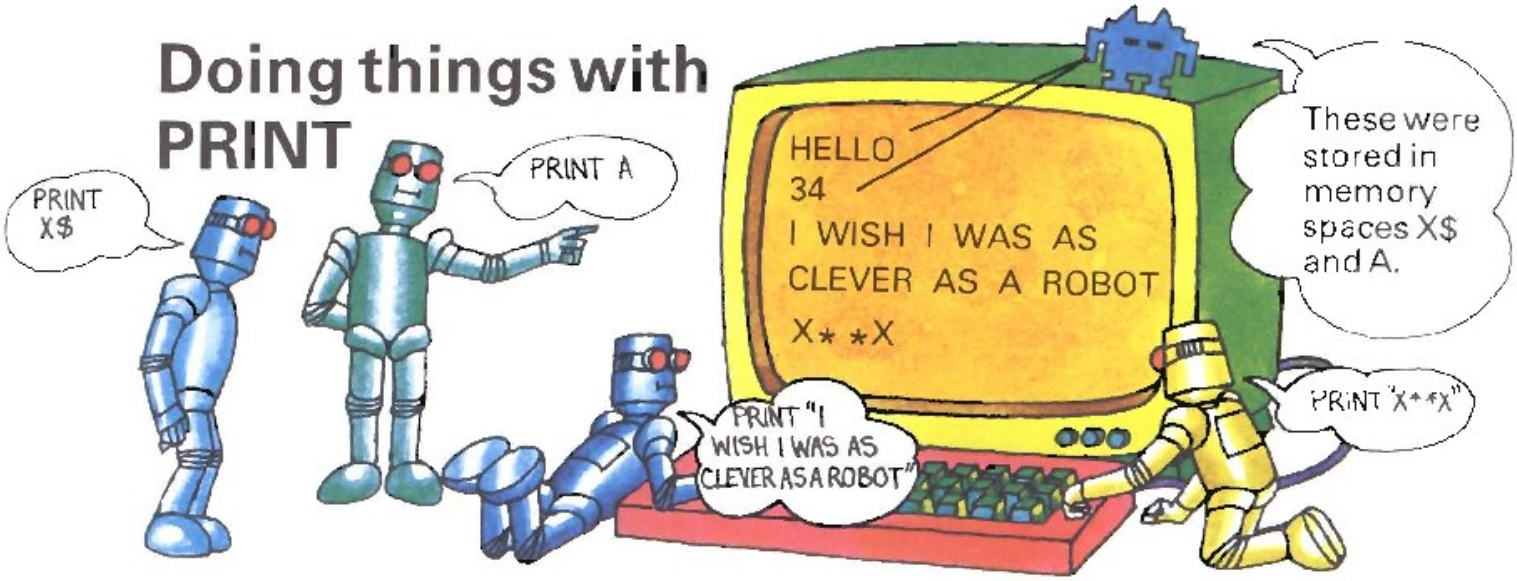






There are some hints to help you find bugs on pages 42-43

Doing things with PRINT



So far you have seen how to use PRINT to display words and numbers on the screen, and to print out the contents of variables. Below you can find out how to use commas and semi-colons to space things out on the screen. You can also use PRINT to do calculations on a computer. You can find out how at the bottom of the page. On the opposite page you can find out more about doing things with variables.

Commas and semi-colons

```
10 PRINT "I AM SPACED",
20 PRINT "OUT"
```

Comma

```
I AM SPACED      OUT
```

```
10 PRINT "I AM SQUASHED";
20 PRINT "UP"
```

Semi-colon

```
I AM SQUASHEDUP
```

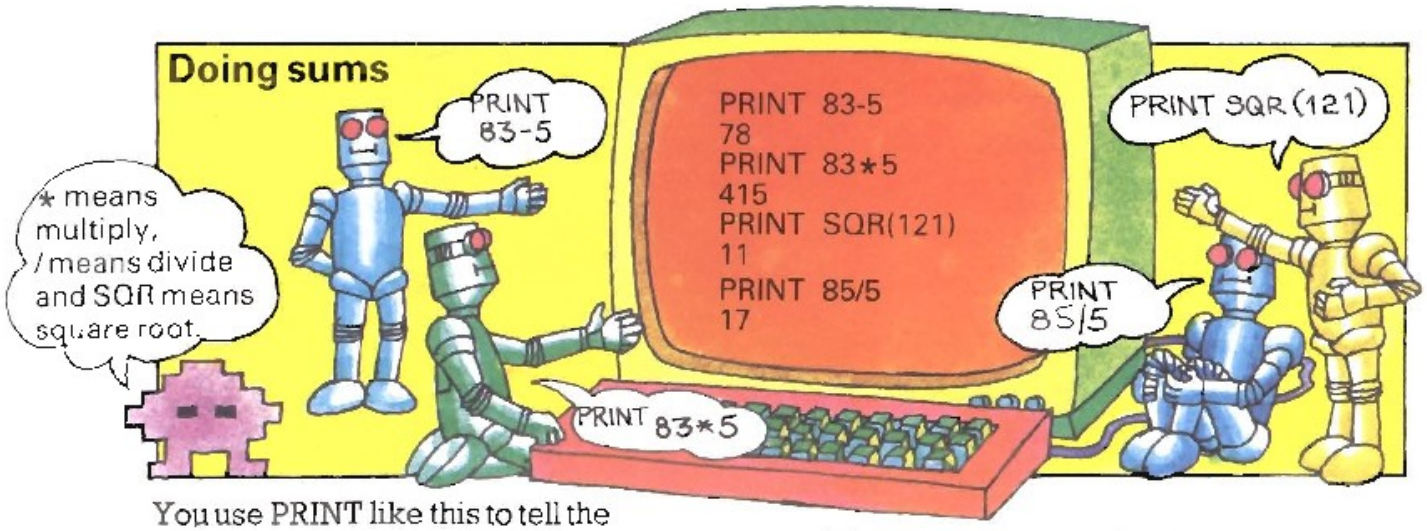
```
10 PRINT "I AM REALLY"
20 PRINT
30 PRINT "SPACED OUT"
```

```
I AM REALLY

SPACED OUT
```

The word PRINT by itself made this empty line.

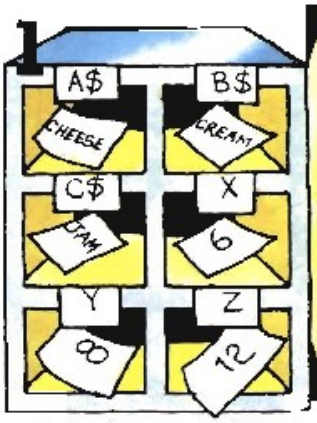
These lines show how you can use commas and semi-colons to tell the computer where to print the next letter. A comma tells it to move along the screen a bit and a semi-colon tells it to stay where it is. The picture above shows how the lines would be printed on the screen. The word PRINT on a line by itself tells the computer to leave an empty line.

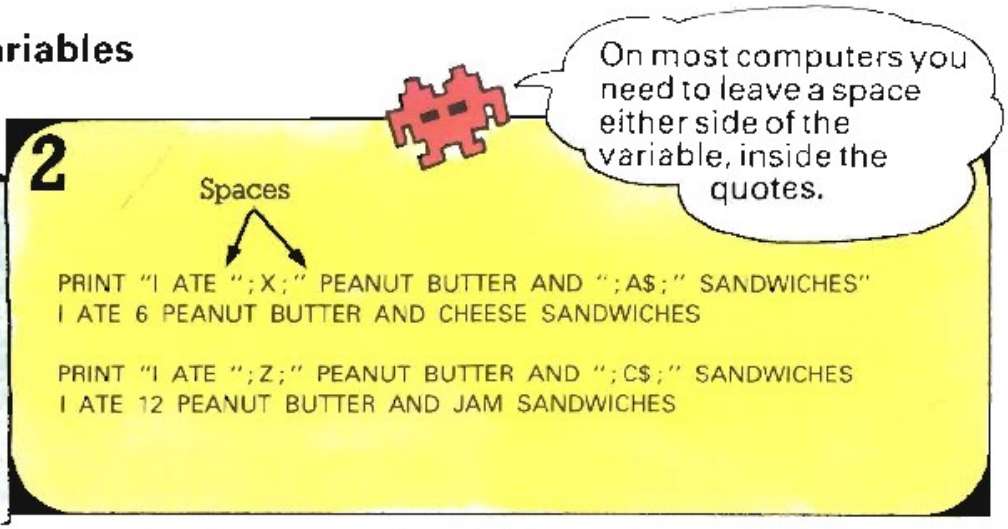


You use PRINT like this to tell the computer to do sums. You use the normal signs for addition and subtraction and * for multiplication and / for division.

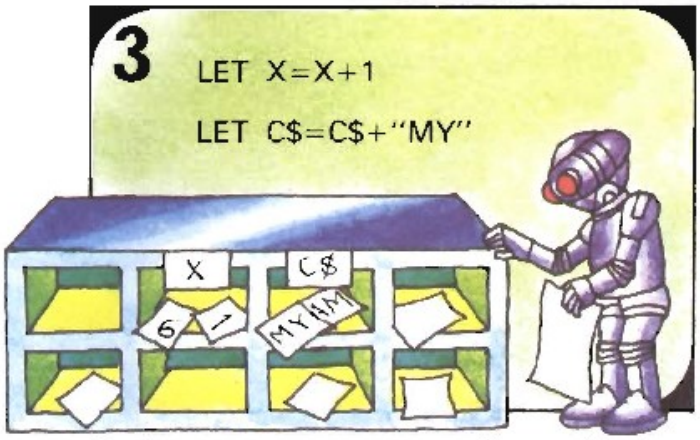
The computer can also do more complex mathematical calculations such as sines, cosines, square roots, etc.

More about variables

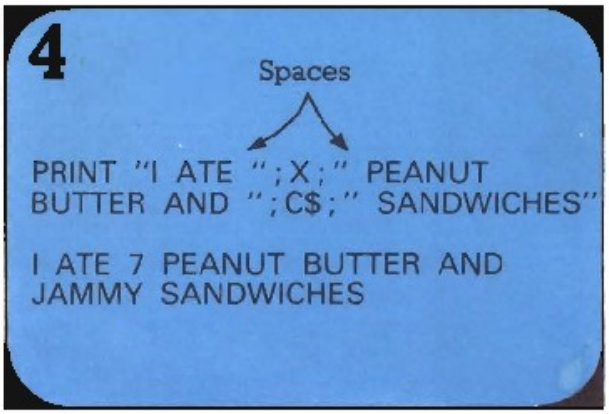




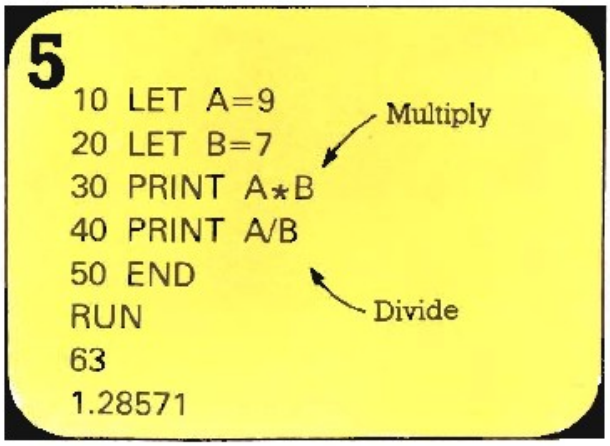
Printing variables by themselves is not very useful. You usually need some words with them to say what they are. To print words and a variable together the words must be in quotation marks as usual, and the variable must have a semi-colon either side of it, as shown above. If you want to space out the information you can use commas instead of semi-colons.



During a program you can change the contents of memory spaces like this. To the computer these statements mean add one to the figure in memory space X and add "MY" to the letters in C$.



Next time you ask the computer to print the variables it will display the new words and numbers stored in the memory spaces.



You can do sums with variables too, as shown in the program above. The computer finds the numbers in the memory spaces, then works out the sums.

Program puzzles

1. Write a program to add numbers to the variables in the program on the left so that it would print out the answers 100 and 1 on one line with a space between.

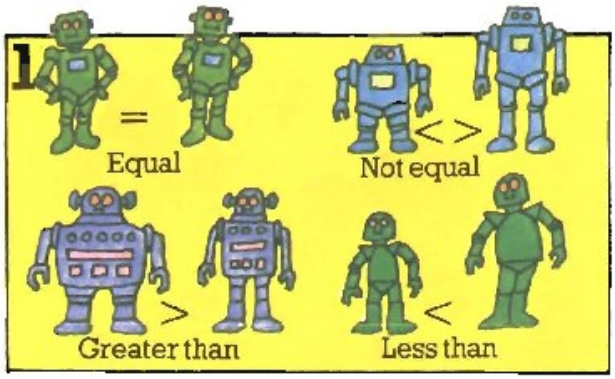
2. Change lines 30 and 40 so that they print out the numbers, what you are doing to them and the answer, e.g. "7 times 9 is 63".

3. Change your answer to the program puzzle on page 15 so it prints your name and the message on one line.

How computers compare things



One of the most useful things a computer can do is to compare pieces of information and then do different things according to the results. To do this you use the words IF . . . THEN.



The computer can do several different tests on information to compare it. The symbols for the tests are shown above. It can test to see if two pieces of data are equal, different, or if one is greater or less than the other.

2

```
IF A=B THEN PRINT "THEY ARE EQUAL"
IF A>B THEN PRINT "A IS BIGGER"
IF A<B THEN PRINT "A IS SMALLER"
IF A<>B THEN PRINT "THEY ARE NOT
                              EQUAL"
```

These lines show how you use the symbols with IF and THEN to make the computer compare two pieces of data. You can compare any kind of data – words, numbers and variables, i.e. the contents of memory spaces, too.

3 Weather program

```
10 PRINT "WHAT'S THE WEATHER LIKE TODAY"
20 INPUT W$
30 IF W$="RAIN" THEN PRINT "UMBRELLA TIME"
40 IF W$="SUNNY" THEN PRINT "GOOD"
50 END
```



Here is a program using IF and THEN. At line 20 the computer stores the word you input in variable W$. Then, at lines 30 and 40 it checks to see if the word in W$ is the same as "rain" or "sunny". If it is, it prints out one of the responses. If you put in a different word at line 20 nothing will happen. You could change the words in lines 30 and 40, though, then try inputting one of the new words.

4

```
RUN
WHAT'S THE WEATHER
LIKE TODAY
?SUNNY
GOOD
RUN
WHAT'S THE WEATHER
LIKE TODAY
?RAIN
UMBRELLA TIME
```

5 Age program

```
10 PRINT "HOW OLD ARE YOU"
20 INPUT A
30 IF A>16 THEN PRINT "OLD"
40 IF A<16 THEN PRINT "YOUNG"
50 IF A=16 THEN PRINT "JUST RIGHT"
RUN
HOW OLD ARE YOU
?16
JUST RIGHT
```

6 French lesson

```
10 PRINT "HOW DO YOU SAY RED IN
   FRENCH"
20 INPUT A$
30 IF A$="ROUGE" THEN PRINT "CORRECT"
40 IF A$<>"ROUGE" THEN PRINT "NO,
   ROUGE"
RUN
HOW DO YOU SAY RED IN FRENCH
?BLEU
NO, ROUGE
```

In the age program, the computer compares input A with the figure 16. If it is bigger than 16 it prints "old". If it is smaller it prints "young" and if it is 16 it prints "just right". In the other program the computer prints out one of two different responses depending on whether A$ equals "rouge" or not.

Program puzzle – Can you write a program to get the computer to ask you sums, then either print "correct" or give you the right answer?

Branching programs

1

```
IF A=6 THEN LET A$="SIX"
IF X=Y-2 THEN LET Z=0
IF S=T THEN STOP
IF R<10 THEN GOTO 30
```

This tells the computer to go to line 30.

2

```
10 INPUT K$
20 IF K$="YES" THEN GOTO 100
30 IF K$="NO" THEN GOTO 200
:
100 PRINT "YOU TYPED YES"
110 STOP
:
200 PRINT "YOU TYPED NO"
210 END
```

These two lines make the computer branch to other parts of the program.

You can give the computer almost any instruction after the word THEN, as shown above. A useful instruction is to make it go to another line. (On most computers, but not the ZX81, you can leave out the word GOTO.) You usually need a STOP instruction in programs with GOTO, or the computer will go on repeating the program endlessly.

Maths program

```
10 PRINT "TYPE IN A NUMBER"
20 INPUT A
30 PRINT "TYPE IN ANOTHER NUMBER"
40 INPUT B
50 PRINT "DO YOU WANT TO"
60 PRINT "ADD, SUBTRACT, MULTIPLY"
65 PRINT "DIVIDE OR STOP"
70 INPUT C$
80 IF C$="ADD" THEN PRINT A+B
90 IF C$="SUBTRACT" THEN PRINT A-B
100 IF C$="MULTIPLY" THEN PRINT A*B
110 IF C$="DIVIDE" THEN PRINT A/B
120 IF C$="STOP" THEN STOP
130 GOTO 10
```

```
RUN
TYPE IN A NUMBER
?17
TYPE IN ANOTHER NUMBER
?184
DO YOU WANT TO
ADD, SUBTRACT, MULTIPLY
DIVIDE OR STOP
?ADD
201   <-- Computer's answer
TYPE IN A NUMBER
?
```

THE PROGRAM WILL ONLY STOP WHEN YOU INPUT THE WORD STOP

In this program the numbers you type in are stored in A and B and your instructions are stored in C$. At lines 80 to 120 the computer compares C$ with five different words, and when it finds the right word, it carries out the instruction. It passes over all the lines which are not true.

Age guessing program

1

```
10 PRINT "GUESS MY AGE"
20 INPUT G
30 IF G<>14 THEN PRINT
   "TRY AGAIN"
40 IF G<>14 THEN GOTO
   20
50 PRINT "CORRECT"
60 END
```

2

```
RUN
GUESS MY AGE
?15
TRY AGAIN
?14
CORRECT
```

3

```
GUESS MY AGE
?15
YOUNGER THAN THAT
?13
OLDER THAN THAT
?14
CORRECT
```

CAN YOU WRITE THE PROGRAM FOR THIS ?

This program will go on repeating itself until G=14. When G=14 the computer will pass over lines 30 and 40 and print "correct". Can you alter the program so that it gives you some clues, as shown in the picture on the right?

Programs with lots of BASIC



The programs on these two pages use most of the BASIC covered so far. The first program is a space game for two people to play with the computer. If you do not have a computer, study the programs and try and follow how they work.

Space commando

```
10 PRINT "ALIEN'S SQUARE ALONG"
20 INPUT A
30 PRINT "ALIEN'S SQUARE UP"
40 INPUT B
50 CLS
60 PRINT "COMMANDO'S SQUARE ALONG"
70 INPUT C
80 PRINT "COMMANDO'S SQUARE UP"
90 INPUT D
100 CLS
110 LET X=SQR((A-C)*(A-C)+(B-D)*(B-D))
120 PRINT "YOU ARE NOW"
130 PRINT X;"SPACE UNITS APART"
140 IF X<1.5 THEN PRINT "ALIEN FOUND"
150 IF X<1.5 THEN STOP
155 PRINT "WHAT ARE YOUR NEW POSITIONS"
160 GOTO 10
170 END
```

Lines 10 to 40 store alien's co-ordinates in A and B.

Line 50 clears alien's numbers off screen.

Lines 60 to 90 store commando's co-ordinates in C and D.

This line works out how far apart they are and stores the answer in X.

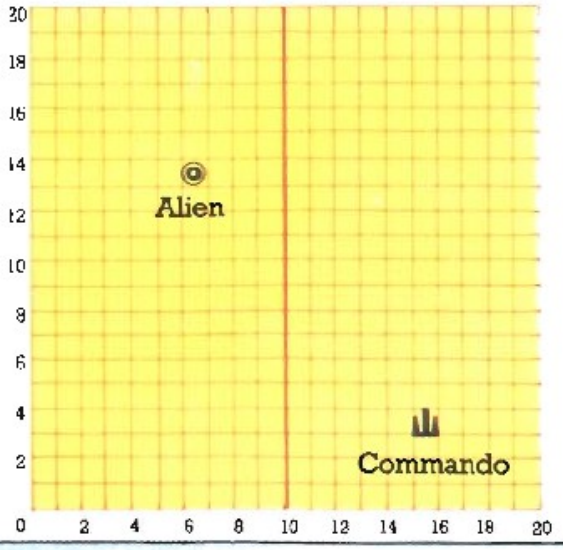
If X is less than 1.5 the program stops. If it is more than 1.5 the program repeats.



In this game, one person is a hostile alien and the other is a space commando trying to catch the alien. Each player draws a secret map on which they plot their positions (you can find out how to do this below). They give the computer the grid co-ordinates of their positions and the computer then works out how far apart they are. The players use the computer's figures to help them work out their next moves.

How to play

For their secret map, each player draws a grid of 20 × 20 squares and numbers them as shown on the right. The alien starts in the left side of the grid and the commando starts in the right. Each turn, they can move two squares up, down, sideways or diagonally and then give the computer their new positions. When they are less than 1.5 space units (i.e. squares) apart, the commando has caught the alien.



How to make the computer look clever

In this program the computer appears to respond to your answers to its questions. You can see how the program works in the pictures at the bottom of the page. The program uses INPUT in a slightly different way which makes the program shorter and easier to read.

1

```
10 INPUT "GIVE ME A NUMBER";N
20 INPUT "AND ANOTHER";M
30 PRINT N;" TIMES ";M;
" IS ";N*M
```

The BBC micro does not need a semi-colon.

On most computers (not the ZX81) you can make the INPUT line clearer by putting words in quotes before the variable name.

2

On the ZX81 you have to type
10 PRINT "GIVE ME A NUMBER"
15 INPUT N

```
RUN
GIVE ME A NUMBER?10
AND ANOTHER?8
10 TIMES 8 IS 80
```

When you run the program the input question mark appears after the words.

The program

```
5 LET C=0
10 PRINT "I WOULD LIKE TO TALK TO YOU"
20 INPUT "TELL ME ANYTHING SILLY THAT
   HAPPENED TO YOU THIS WEEK";A$
30 READ B$
40 PRINT B$;
50 INPUT C$
60 LET C=C+1
70 IF C=6 THEN GOTO 100
80 GOTO 30
90 DATA WHY, WHY IS THAT
95 DATA WHY, CAN YOU EXPLAIN
98 DATA CAN YOU SAY WHY, WHAT WAS THE REASON
100 PRINT "SO THE REASON YOU TYPED"
110 PRINT "     ";A$
120 PRINT "WAS REALLY GIVEN BY YOUR ANSWER"
130 PRINT "     ";C$
140 PRINT "HOW ODD!"
150 PRINT "RUN ME AGAIN FOR FURTHER
    ENLIGHTENMENT"
160 END
```

This is the new input way. Your reply is stored in **A$**.

At line 30 the computer looks for the first line with DATA and takes the first item and puts it in **B$**.

This makes the computer stay on the same line.

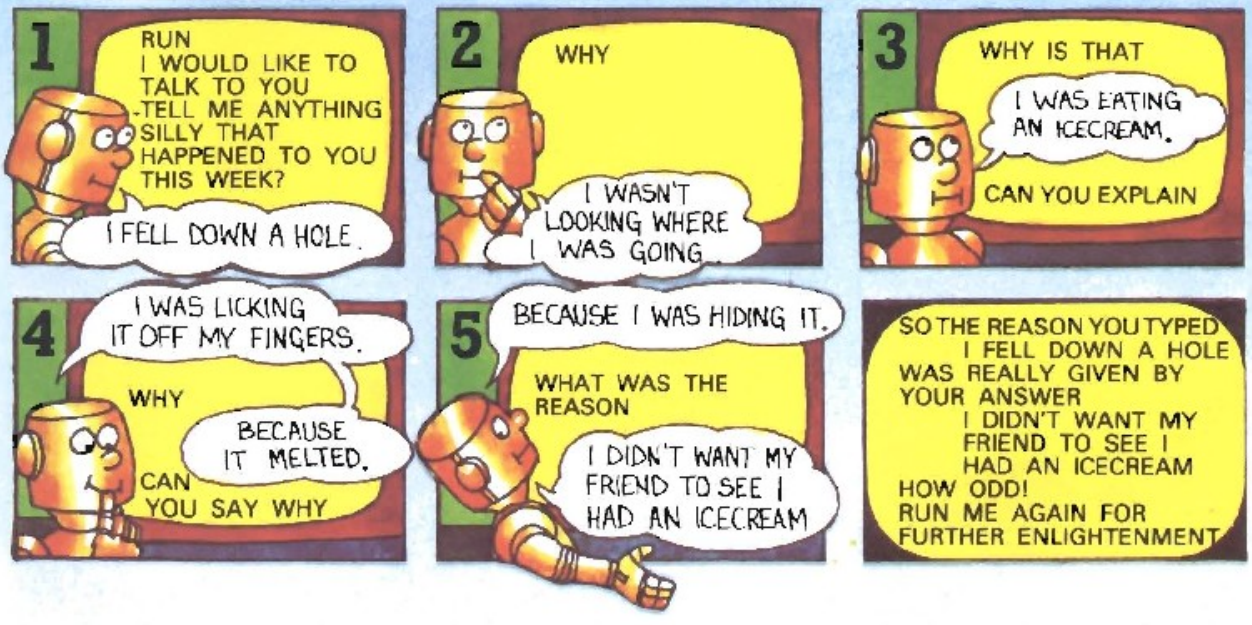
Your reply is stored in **C$**.

Variable C in lines 60 and 70 is a counter. It keeps count of the number of times the program is repeated. When C = 6 all the data items have been used and the computer moves on to line 100.

Line 80 makes the computer go back to line 30 and replace the data in **B$** with the next item in the data list.

The spaces in lines 110 and 130 leave spaces on the screen before your replies. It does not matter how many spaces you leave in the program.

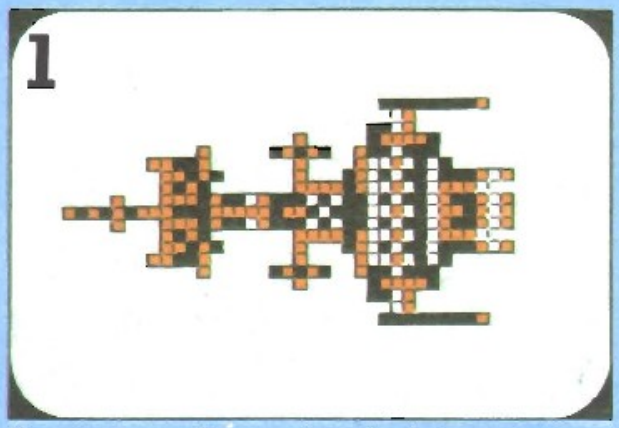
How it works



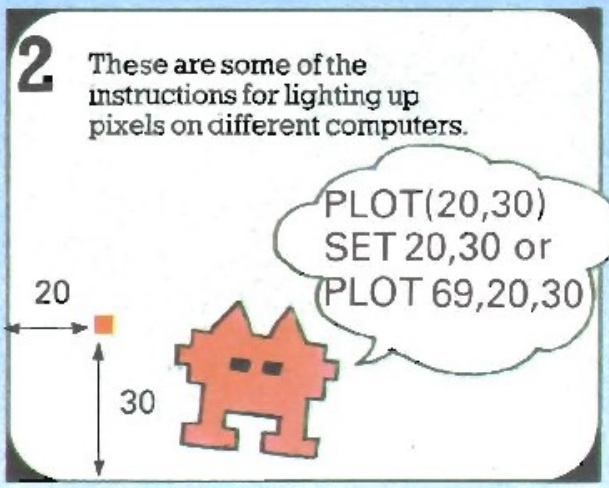
Drawing pictures

A computer makes pictures by lighting up little rectangles on the screen. Each rectangle is called a pixel and each pixel needs a separate instruction from the computer to switch it on. Most computers can also make the pixels different colours.

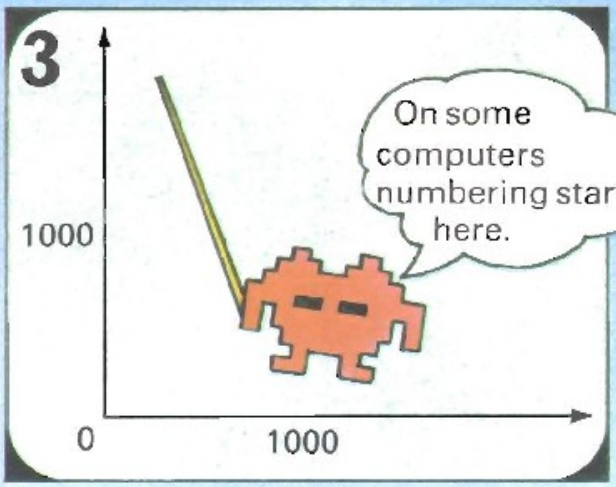
On these two pages you can find out how to use BASIC to make simple pictures on the screen. The instructions given here are for single colour pictures only.



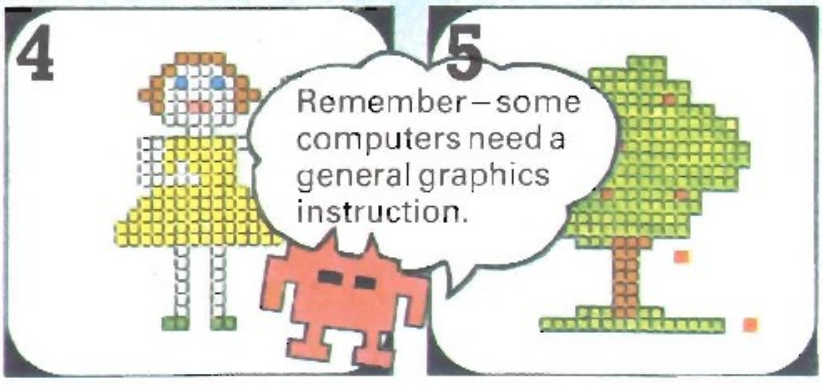
You can usually see the pixels in a computer picture. A computer with a large memory, though, can make pictures with thousands of very small pixels. These pictures are called high resolution graphics.



The instruction for lighting up a pixel varies on different computers, but it is usually something like PLOT (X, Y). X and Y are the pixel's co-ordinates and X is the number of pixels along and Y is the number of pixels up.



On a computer with high resolution graphics you may be able to plot 1000 points along the screen and 1000 up. A less powerful computer has about 60×40. (If you have a computer, check the size of your screen as you may get a bug if you plot outside its range.)





Pictures made by a computer are usually called graphics. Some computers need a special command before you do graphics. For instance, on the BBC micro you need the word MODE with a number.*

You can also switch a pixel off with a command such as UNPLOT (X, Y). In the programs in this book we use PLOT and UNPLOT. If you have a computer check these commands in your manual.

*For the programs in this book use MODE 5 on the BBC micro with the plot command PLOT 69, X, Y. For unplotting use PLOT 71, X, Y.

Plotting program

PLOT command varies on different computers.

1

```
10 PRINT "TYPE IN
   TWO NUMBERS"
20 INPUT X
30 INPUT Y
40 PLOT (X,Y)
50 GOTO 10
```

You have to press NEWLINE or RETURN after inputting each number

2

```
RUN
TYPE IN TWO NUMBERS
?24
?24
TYPE IN TWO NUMBERS
?30
?15
```

1st pixel

2nd pixel

This short program asks you for two numbers, then plots the pixel with those numbers as co-ordinates. If you try this program make sure the numbers you type in are within the range of your computer.

Line 50 makes the program repeat itself endlessly and the only way to stop it is with BREAK (or the computer's own word). Can you insert a counter (see page 21) to make it run, say, six times.

Plotting a picture



```
10 LET X=10
20 LET Y=10
30 PLOT (X,Y)
40 LET X=X+1
50 LET Y=Y-1
60 IF X<14 THEN GOTO 30

100 LET Y=Y+1
110 LET X=X+1
120 PLOT (X,Y)
130 IF X<20 THEN GOTO 100
```

This plots a diagonal line going down.

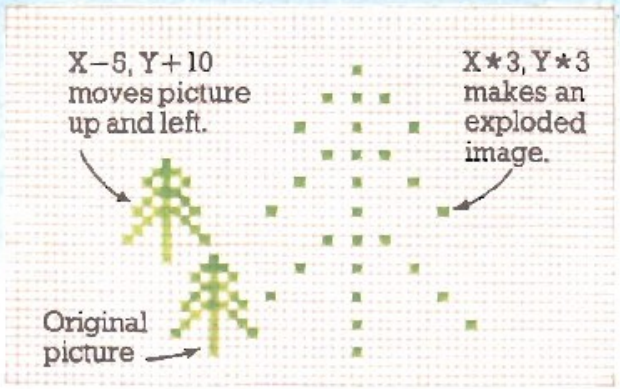
This plots a diagonal line going up.

Adding 1 to X and not to Y plots a horizontal line

Adding 1 to Y and not to X plots a vertical line.

First you need to draw the picture on squared paper and work out the co-ordinates of the squares.

Then you can work out the program to plot all the squares. By giving X and Y starting values, then adding to them or subtracting from them, and repeating parts of the program, you can make the computer plot sequences of pixels as shown above.



After writing the program it is easy to change the picture by altering the numbers. You can move it to a different place on the screen by changing the starting values, or multiply all the numbers by three to make an "exploded image".

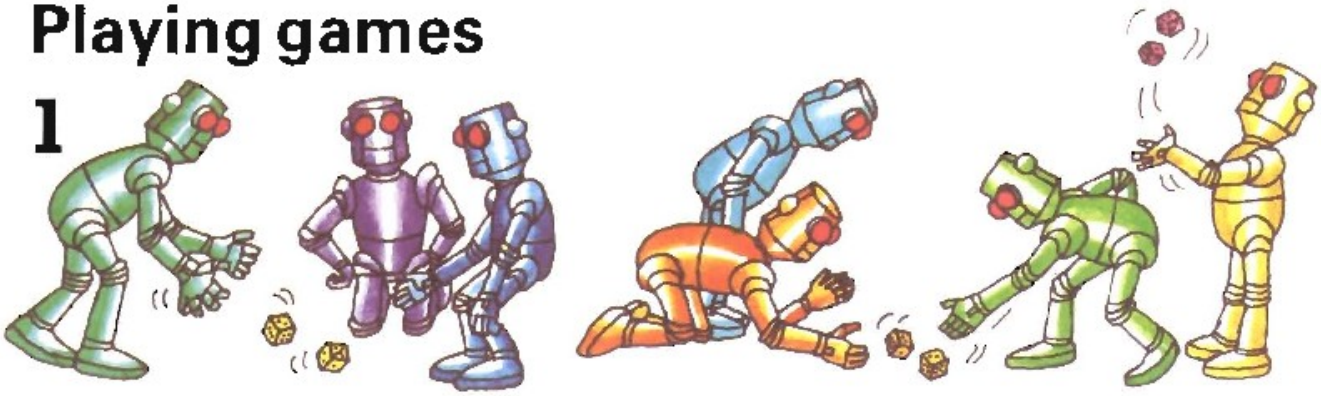
Another way



You can really only make very simple pictures with PLOT. To make more complicated ones you need special equipment such as a graphics tablet. You place a drawing on the tablet and trace over it with a special device called a "puck". This automatically reads the co-ordinates into the computer.

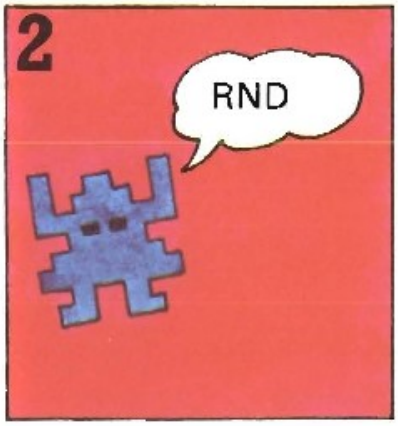
Program puzzle – Can you write a program to plot your initial on the screen? There is a sample program on page 44.

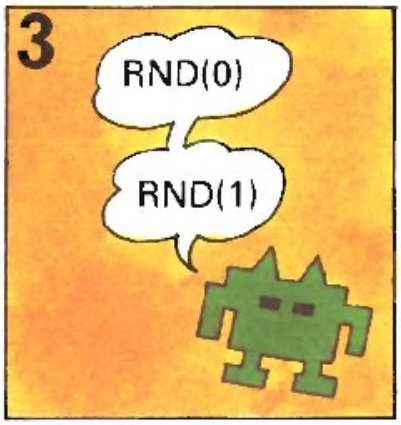
Playing games

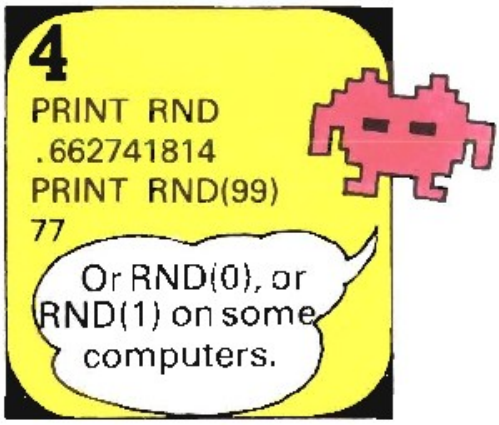


When you throw a pair of dice you cannot predict what the numbers will be. The chances are equal that the numbers will be anything from one to six. You can produce unpredictable numbers on a computer. They are called random numbers.

The computer contains a special program for producing random numbers. Sometimes it repeats the same number several times, but in sequences of lots of random numbers, the number of times each number is picked is about even.

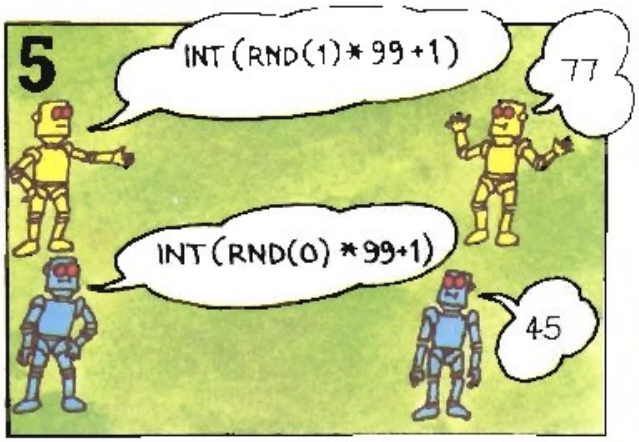


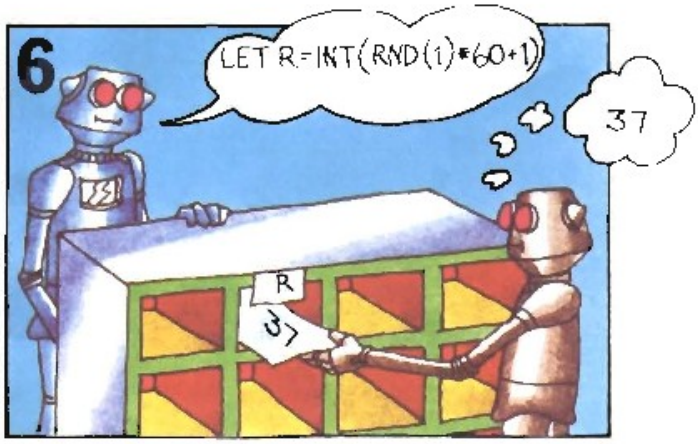




To make the computer produce a random number you use the word RND. Some computers need a 1 or 0 in brackets after the word. If you have a computer, check your manual for the correct command.

The RND instruction always produces a number below one. On some computers you can put a number in brackets after RND, e.g. RND(99). This makes it produce a whole number between 1 and the number in brackets.





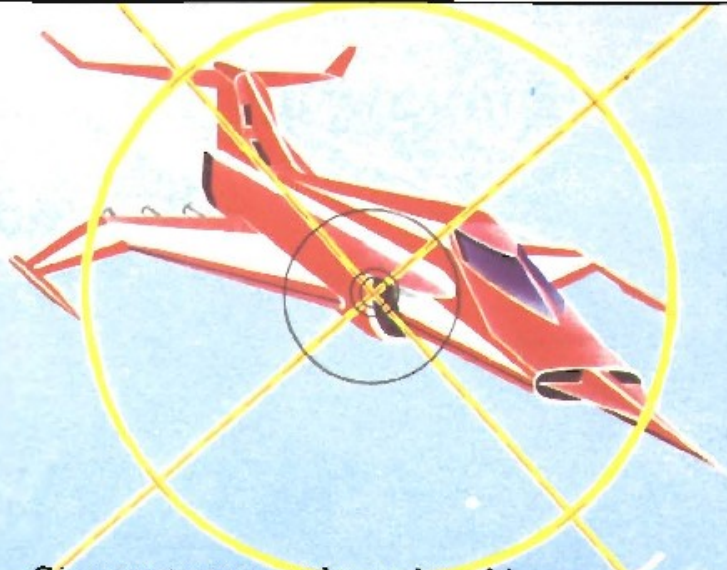
On other computers you need the word INT (short for integer, meaning whole number) followed by the RND instruction (either RND(1) or RND(0) on different computers). Then you multiply by the highest number you want and add one so the number is above one.

This instruction means pick a random number and store it in variable R. In the programs in this book we use INT(RND(1)*60+1) to mean pick a random number between 1 and 60. You may need to convert this instruction for some computers.

Program puzzle – Can you work out how to make the computer pick a random number between 10 and 20?

Space attack

This is a program for a game using random numbers. In the game you are on a star ship being attacked by a wave of alien fighters. Your ship's computer locates the aliens and gives you their coded positions. To hit each alien you have to work out the firing range by multiplying the codes and typing in the answer.



```
10 LET C=0
20 LET A=INT(RND(1)*20+1)
30 LET B=INT(RND(1)*20+1)
40 PRINT "ALIEN SHIP'S CODES"
45 PRINT "ARE ";A,B;" FIRE"
50 INPUT X
60 LET C=C+1
70 IF X=A*B THEN PRINT "ALIEN SHIP
   DESTROYED"
80 IF X<>A*B THEN PRINT "MISSED"
90 IF C<6 THEN GOTO 20
100 END
```

C is a counter to count the number of times the program is repeated. Each time, line 60 adds 1 to C.

These two lines produce random numbers for the alien ship's codes and store them in A and B.

Your number is stored in X.

In lines 70 and 80 the computer checks to see if you got the right answer.

This line repeats the program if C is less than 6.

Running the program

The picture on the right shows what happens when you run the program. If you type in the correct answer for the two numbers multiplied together the computer will print "alien ship destroyed". If your answer does not equal A×B the computer prints "missed".

```
RUN
ALIEN SHIP'S CODES
ARE 17      3 FIRE
?41
MISSED
ALIEN SHIP'S CODES
ARE 11      5 FIRE
?55
ALIEN SHIP DESTROYED
ALIEN SHIP'S CODES
ARE 13      6 FIRE
```

The comma in line 45 spaced out the numbers like this.

Program puzzle

Can you add another counter to the program to count your number of hits and print out your score at the end of the game? You need to set up a variable called S and give it a value of 0 to start with, then add 1 for each hit.

Random pattern program

```
5 CLS
10 LET X=INT(RND(1)*30+1)
20 LET Y=INT(RND(1)*30+1)
30 PLOT (X,Y)
40 GOTO 10
```

This clears the program off the screen before the pixels are plotted.

The random numbers must fit on the computer's screen.

This line makes the program repeat endlessly.

Computers' commands for CLS, RND and PLOT may vary and some will need a graphics line.



This program uses random numbers to plot spots of light on the screen. Lines 10 and 20 produce random numbers between 1 and 30 and store them in X and Y. Line 30 then plots the pixel with co-ordinates X,Y. As the screen fills up you see less pixels appearing as many of them are already plotted. To stop the program you have to type BREAK or ESCAPE, or another word on different computers.

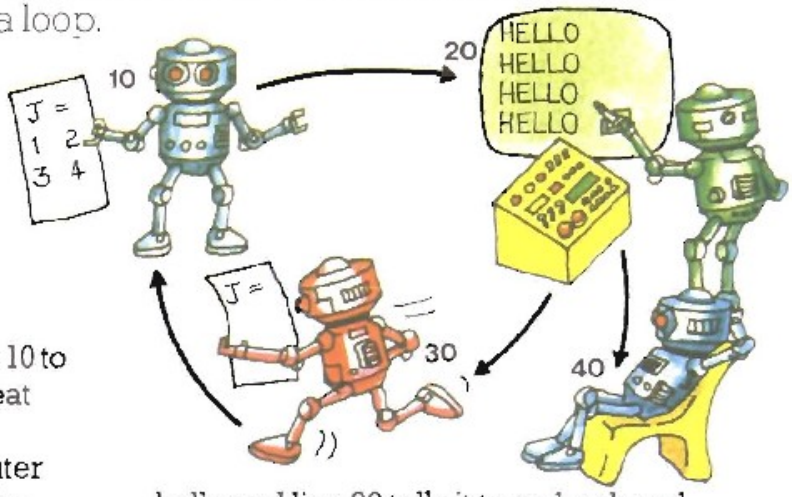
Making loops

You often need the computer to do the same thing several times in a program. On page 21 you can see how to make it repeat part of a program using GOTO and a variable which acts as a counter. Another way is to repeat the same lines several times using the words FOR . . . TO and NEXT. This is called making a loop.

1 Hello loop

```
10 FOR J=1 TO 6
20 PRINT "HELLO"
30 NEXT J
40 END
```

Loop



This program has a loop from lines 10 to 30 which makes the computer repeat line 20 six times. The letter J is a variable and line 10 tells the computer to set J at 1 on the first run through the program, 2 the next time, then 3, etc., up to 6. Line 20 tells it to print the word hello and line 30 tells it to go back and find the next value for J. When J=6 the computer goes on to line 40.

2 Silly sums program

```
10 FOR J=1 TO 8
20 PRINT "2 PLUS 2 IS 5"
30 NEXT J
40 PRINT
50 PRINT "ONLY JOKING!"
60 END
```

Loop



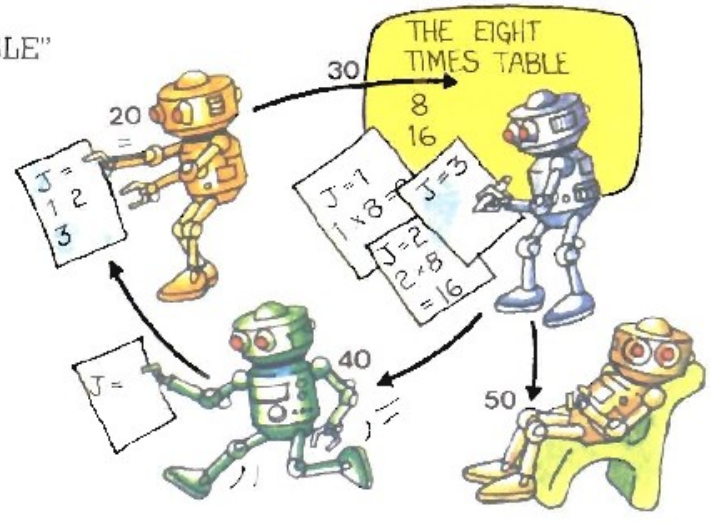
2 PLUS 2 IS 5
2 PLUS 2 IS 5
2 PLUS 2 IS 5
2 PLUS 2 IS 5
2 PLUS 2 IS 5
2 PLUS 2 IS 5
2 PLUS 2 IS 5
2 PLUS 2 IS 5
ONLY JOKING!

In this program, the loop from lines 10 to 30 makes the computer repeat line 20 eight times. Each time it passes through line 20 it prints out the same silly sum. After doing it eight times the computer carries on with the rest of the program. Line 40 just makes it leave an empty line.

3 Eight times table program

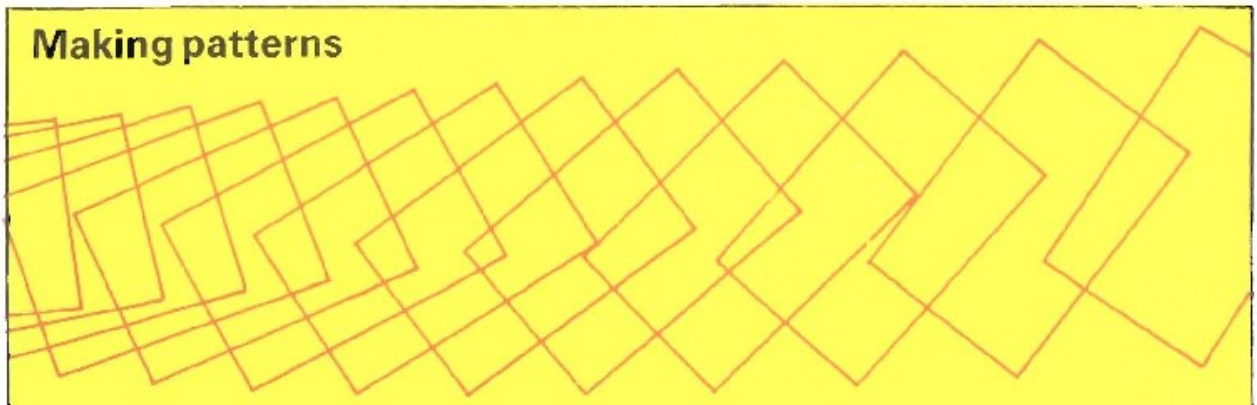
```
10 PRINT "THE EIGHT TIMES TABLE"
20 FOR J=1 TO 12
30 PRINT J*8
40 NEXT J
50 END
```

Loop



This time J is used to count the number of loops and also as part of the sum J*8. Line 20 tells the computer to set J at 1, then 2, 3, etc, up to 12. Line 30 takes the current value of J, multiplies it by 8 and prints out the answer. Then line 40 sends the computer back to line 20 to find the next value of J.

Making patterns



FOR . . . NEXT loops are useful for making patterns, like this, of a simple shape repeated lots of times. The program for this pattern is too long to write out here in full, but it would look something like this:

```
10 FOR I=1 TO 45
20 Draw a rectangle and change its
   position a little each time.
30 NEXT I
40 END
```

Steps

Sometimes it is useful to change the value of J by amounts other than 1. For instance, you may want to go up in 3s or down in 7s. To do this you use the word STEP. In the following program STEP −1 makes J go down by 1 each time the computer passes through the loop in lines 10 to 40.

Greedy computer program

```
5 CLS
10 FOR J=7 TO 2 STEP -1
20 PRINT "THERE ARE ";J;" PIES LEFT"
30 NEXT J
40 PRINT
50 PRINT "I SHALL EXPLODE"
60 FOR K=1 TO 1000
70 REM: DO NOTHING
80 NEXT K
90 PRINT
100 PRINT "B A N G S P L A T T"
```

The figure 2 stops the loop after J−2 (i.e. when there is one pie left).

Loop

Loop

```
THERE ARE 7 PIES LEFT
THERE ARE 6 PIES LEFT
THERE ARE 5 PIES LEFT
THERE ARE 4 PIES LEFT
THERE ARE 3 PIES LEFT
THERE ARE 2 PIES LEFT
I SHALL EXPLODE
B A N G S P L A T T
```

Some computers are slower than others and they need a lower figure such as 500 or 250 in line 60.



There are two loops in this program. The one from lines 10 to 30 makes the computer print line 20 six times. Each time, the value of J is reduced by one and the figure for J is printed in line 20. In the loop from lines 60 to 80 the computer does not have to do anything. It just runs through all the values for K from 1 to 1000 and this makes it pause for a moment. Lines which start with REM (short for remark) are ignored by the computer and are useful to remind you what the program is doing.

Program puzzles

1. Can you alter the eight times table program on the left to make it display "1×8=" as well as the answer?

2. Can you write a program for the "N" times table, that is, a program which works out the tables for any number you type into the computer? First you need to get the computer to ask you for a number, N. Then use a loop to work out and display the tables. If you want, include some lines at the end of the program so it asks you if you want the tables for another number and the program repeats itself.

Tricks with loops

Here are some more programs using loops. Below you can find out how you can use loops within loops to repeat several things at the same time. These are called nested loops.

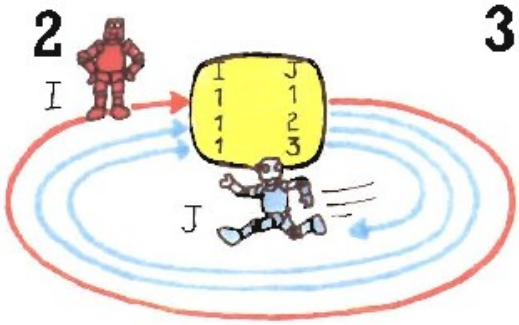
1 Nested loops

```
5 PRINT "I", "J"
10 FOR I=1 TO 3
20 FOR J=1 TO 3
30 PRINT I, J
40 NEXT J
50 NEXT I
60 END
```

J loop

I loop

2



3

I	J
1	1
1	2
1	3
2	1
2	2
2	3
3	1
3	2
3	3

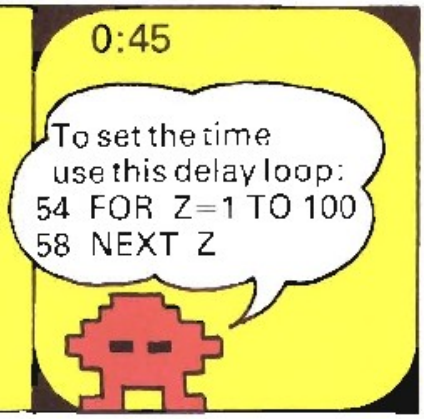
This program has an I loop and a J loop. The J loop is nested inside the I loop and for each time that the I loop is carried out, the J loop is repeated three times, printing out the value for J each time. The picture above shows the result of this program. The commas spaced the figures out like this.

Computer clock

```
5 CLS
10 LET M=0
20 LET S=0
30 FOR M=0 TO 59
40 FOR S=0 TO 59
50 PRINT M;":";S
60 CLS
70 NEXT S
80 NEXT M
90 END
```

Seconds loop

Minutes loop



Inside a computer there is an electronic "clock" which sets the rhythm for all the computer's work. The clock pulses at between one and four million pulses a second. This program makes the computer behave like a digital clock. It has nested loops, one to count the seconds and one to count the minutes.

Bugs in loops

```
10 FOR I=1 TO 4
20 FOR J=1 TO 4
30 PRINT I
40 PRINT J
50 NEXT I
60 NEXT J
```



The seconds loop is carried out 59 times for each minute loop. If you try this program on a computer it might run very fast at first. You need to put in an extra "delay loop", then set it by changing the figure in the loop so your computer clock "ticks" at the same rate as a real one.

Random number tester

```
10 FOR I=1 TO 1000
20 LET R=INT(RND(1)*6+1)
30 IF R=1 THEN LET A=A+1
40 IF R=2 THEN LET B=B+1
50 IF R=3 THEN LET C=C+1
60 IF R=4 THEN LET D=D+1
70 IF R=5 THEN LET E=E+1
80 IF R=6 THEN LET F=F+1
90 NEXT I
100 PRINT "FINISHED"
110 PRINT A, B, C
120 PRINT D, E, F
130 END
```



This's program takes a long, long time. You can make it shorter by changing the number in line 10 to 500 or even 250.

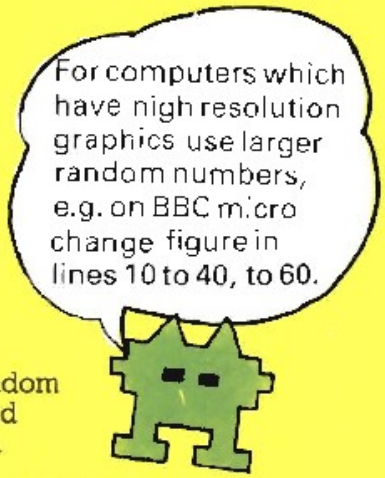
```
RUN
FINISHED
162        168        167
160        187        156
```

This program shows if RND really works. The loop from lines 10 to 90 makes the computer pick a random number between 1 and 6 a thousand times. It keeps count of how often each number is picked in the variables A to F, then prints out the results.*

*On some computers, e.g. ZX81 and BBC micro, you need some extra lines at the beginning of the program to set each variable to 0.

Pattern repeat program

This program uses nested loops to repeat a small pattern all over the screen. The program looks quite complicated but if you read it through carefully and work out what each line does, you will soon see how it works. The shape of the pattern is decided by random numbers and will be different each time you run the program.




```
5 CLS
10 LET A=INT(RND(1)*6+1)
20 LET B=INT(RND(1)*7+1)
30 LET C=INT(RND(1)*6+1)
40 LET D=INT(RND(1)*4+1)
50 INPUT "HOW MANY POINTS
   ACROSS THE SCREEN";W
60 INPUT "HOW MANY UP";V
65 CLS
70 FOR I=0 TO V STEP V/6
80 FOR J=0 TO W STEP W/6
90 PLOT (J+A,I+B)
100 PLOT (J+A,I+C)
110 PLOT (J+C,I+D)
120 PLOT (J+B,I+D)
130 NEXT J
140 NEXT I
150 END
```

J loop

I loop

These lines choose the random numbers for the pattern and store them in A, B, C and D.

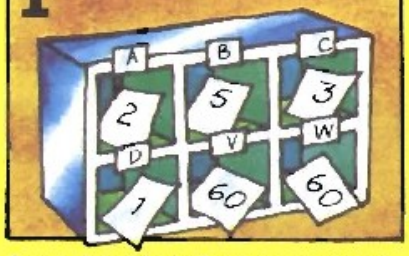
Lines 50 and 60 ask for the width (W) and height (V) of your screen.

The I loop counts the number of times the pattern is repeated up the screen. Each time, I is increased by the height of the screen (V) divided by 6, so the pattern is repeated six times up the screen.

Each time the loops are repeated, lines 90 to 120 tell the computer to plot four pixels using the current values for I and J plus the random numbers.

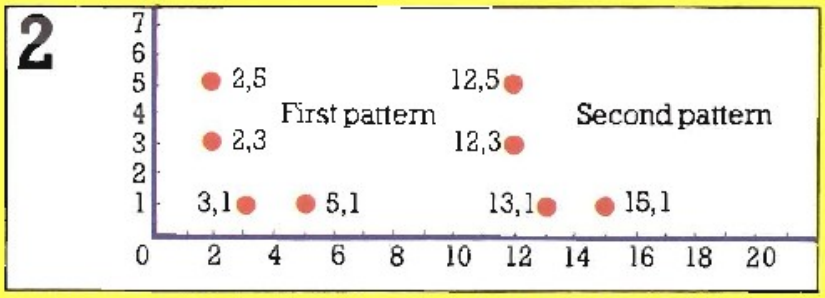
The J loop counts the number of times the pattern is repeated across the screen. It works in the same way as the I loop.

1 How it works



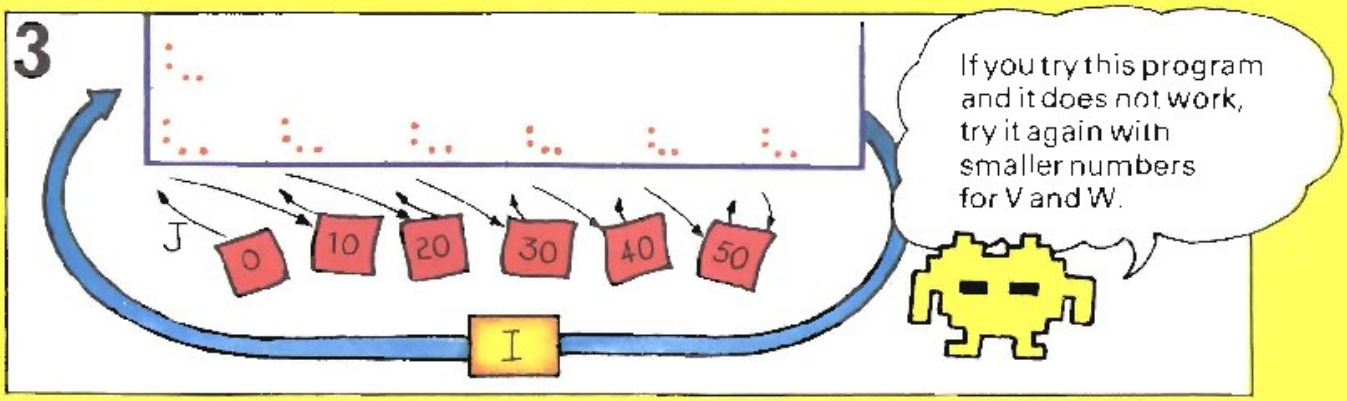
Imagine that the computer has chosen the random numbers 2, 5, 3 and 1 and that the width and height of the screen are both 60.

2



On the first run through the program I and J are 0 so the computer plots the first pattern of dots using only the random numbers. Line 130 sends it back to find the next value for J which is J+60/6, i.e. 10. Then it plots the second pattern using the random numbers plus 10 for J. This repeats the pattern along the screen.

3

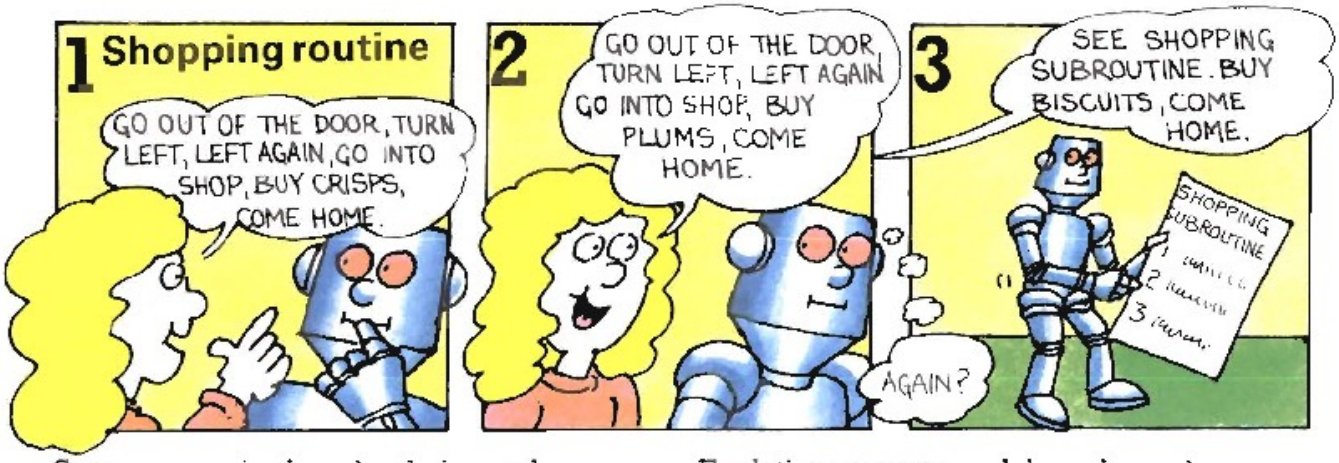


The computer repeats the J loop six times, each time adding 10 to J and so plotting the pattern further along the screen. It then goes back to find the next value for I which is 10. J is set to 0 again and the computer plots the next line of patterns using 10 for I and increasing J by 10 each time as before.

Program puzzle – Can you write a pattern repeat program which repeats a space invader shape over the screen? There are some hints to help you on page 45.

Subroutines

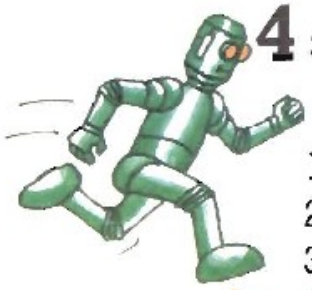
A subroutine is a sort of mini-program within a program. It carries out a particular task, such as adding numbers or keeping a score, and you can send the computer to it whenever you want this task carried out. This saves writing out the program lines each time and makes the program shorter and easier to read and type into the computer.



Suppose you had a robot helper whom you could program to run errands for you. If you wanted something from the shop you would have to give it precise instructions telling it how to get there.

Each time you wanted the robot to buy something you would have to give it the same instructions. It would be much simpler to give the robot a shopping subroutine and tell it to refer to it each time.

4 Shopping program



```
10 PRINT "WHAT DO YOU WANT FROM THE SHOP"
20 INPUT X$
30 GOSUB 100
40 PRINT "ANYTHING ELSE"
50 INPUT M$
60 IF M$="YES" THEN GOTO 10
70 STOP

100 REM: SHOP SUBROUTINE
110 PRINT "GO OUT, TURN LEFT"
120 PRINT "LEFT AGAIN, ENTER SHOP"
130 PRINT "BUY ";X$;" COME HOME"
140 RETURN
```



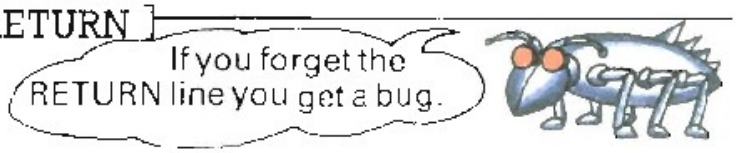


Line 30 sends the computer to the first line of the subroutine.

You need the word STOP at the end of the main program to stop the computer carrying on into the subroutine.

It is useful to label a subroutine with a REM line so you know what it is for.

This sends the computer back to line 40 – the line after GOSUB.



In BASIC, to tell the computer to go to a subroutine you use the word GOSUB with the word RETURN at the end of the subroutine. GOSUB should be followed by the number of the first line of the subroutine. RETURN does not need a line number. The computer automatically goes back to the instruction after the one where it left the main part of the program. You can send the computer to a subroutine anywhere in the program as many times as you like.

Gosub programs

A subroutine is useful for carrying out any task which you want to repeat several times at different stages in the program. Here are some more programs with subroutines.

Numbers program

```
50 INPUT A
60 INPUT B
70 GOSUB 250
80 PRINT "A DIVIDED BY B=";A/B
90 GOTO 50
250 REM: SUBROUTINE TO STOP
260 IF A=0 AND B=0 THEN STOP
270 RETURN
```

This subroutine provides an escape from the program. If you want to stop dividing you input 0 at lines 50 and 60. This program does not need STOP before the subroutine as line 90 sends it back.

Conversion program

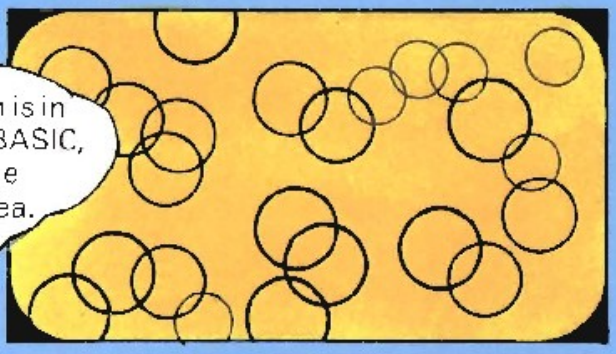
```
100 INPUT "DISTANCE";M
110 INPUT "TIME";T
120 GOSUB 200
130 PRINT "AVERAGE SPEED WAS"
140 PRINT M/T;"MPH AND";K/T;"KPH"
150 STOP
200 REM: SUBROUTINE TO CONVERT MILES
210 LET K=M*1.609
220 RETURN
```

This is a subroutine to convert miles to kilometres. You can often use the same subroutine in lots of different programs. Check that you use the same variable names, though.

Circles program

```
1 Centre of circle = X,Y
2 Radius of circle = R
3 Colour = X
4 Gosub 10
5 Goto 1
10 Rem: Subroutine to draw circles
11 Draw a circle with centre X,Y;
   radius R and colour X.
12 Return
```

This program is in English, not BASIC, to give you the general idea.



Subroutines are useful in graphics programs like this to draw diagrams with numbers worked out in the main part of the program. With this program you could draw lots of different circles by giving the computer different information in lines 1 to 5.

Quiz program

```
5 LET C=0
10 PRINT "WHEN WERE THESE THINGS INVENTED?"
20 READ C$, F
30 PRINT C$
40 INPUT A
50 LET C=C+1
60 IF C=3 THEN STOP
70 GOSUB 100
80 GOTO 10

100 REM: ANSWERS SUBROUTINE
110 IF ABS(A-F)<10 THEN PRINT "OK"
120 IF ABS(A-F)>10 THEN PRINT "NO"
130 PRINT "TRY ANOTHER ONE"
140 RETURN

200 DATA TELEPHONE, 1876, PRINTING PRESS, 1450, BICYCLE, 1791
```

At line 20 the computer looks for the data line and puts the first word item in C$ and the first number item in F.

This prints out the word in C$.

The counter C stops the program after it has repeated three times as there are only three data items for C$ and F.

This is the subroutine.

Each time the program is repeated the words and numbers in C$ and F are replaced by the next pair of data items.

This program uses a subroutine to check the answers to questions. The correct answers are stored in F and the person's answers go in A. In lines 100 and 110 of the subroutine the computer compares A with F. The word ABS stands for "absolute" and it makes the computer check the difference between the numbers in A and F (it ignores any minus signs). If the difference is less than 10 it prints "OK". If it is more than 10 it prints "NO".

Doing things with words

Most computers can examine the words stored in variables and do various things with them. They can check the contents of a variable and see if it contains a particular word or letter. This is useful for checking the words input by someone using the program. Computers can also rearrange the letters or words in a different order and add them to letters in other variables. Below you can find out how you do these things in BASIC.

1

```
10 A$="I AM STUPID"
20 B$="ONLY FOOLS THINK"
30 PRINT B$+" "+A$
RUN
ONLY FOOLS THINK I AM STUPID
```

On most computers, but not the ZX81, you can leave out the word LET.

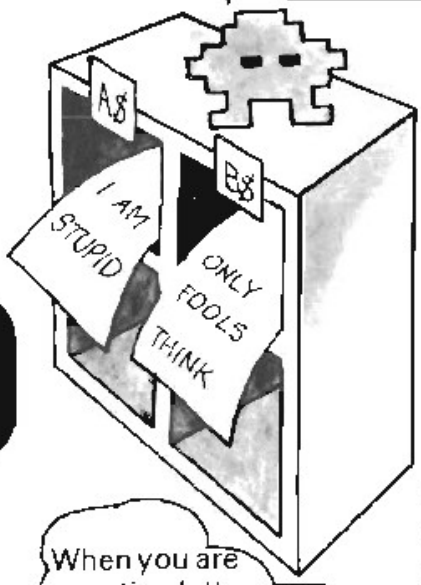
You can add the contents of two variables like this. You need the space between quotation marks to leave a space between the words.

2

```
PRINT LEFT$(B$,4)
ONLY
PRINT LEFT$(B$,4)+" "+A$

ONLY I AM STUPID
```

You can also add parts of variables, like this. LEFT$(B$,4) means take the first four letters from the left of B$.



3

```
PRINT RIGHT$(A$,6)
STUPID
```

To tell the computer to use letters from the right you use RIGHT$ with the name of the string and the numbers of letters you want.

4

```
PRINT MID$(B$,6,5)
FOOLS
```

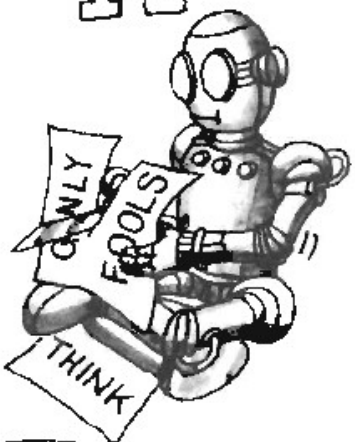
This tells the computer to take the middle letters. The first number tells it where to start and the second tells it how many letters to take.

When you are counting letters, count spaces and punctuation too.

5

```
10 K$="DING DONG!"
20 PRINT LEN(K$)
RUN
10
```

You can also find out the length of a string – the number of letters, spaces and symbols it contains. To do this you use LEN, short for length.



IF A$="COMPUTER BOOK" what is LEFT$(A$,8)? RIGHT$(A$,10)? MID$(A$,5,8)?

Note for Sinclair users

```
PRINT A$(6 TO 11)
STUPID
PRINT B$(14 TO 16)
INK
```

This means take letters numbers 6 to 11.

The Sinclair computers do not use LEFT$, RIGHT$ and MID$, but you can tell the computer to take any letters you want as shown above.

Codemaker program

This program automatically puts words into code. Similar, but much more complex programs are used by intelligence services to write and crack codes.

The easiest way to understand this program is to write a secret message on a piece of paper, then work through the lines of the program carrying out the computer's tasks on your message and writing them down.

```
5 LET C$=""
7 LET D$=""
10 PRINT "TYPE IN A SHORT MESSAGE"
20 INPUT M$
30 PRINT "NOW TYPE IN A SECRET
   NUMBER BETWEEN 2 AND";LEN(M$)-1
40 INPUT N
50 LET A$=RIGHT$(M$,N)
60 LET B$=LEFT$(M$,LEN(M$)-N)
70 LET M$=A$+B$
80 FOR I=1 TO LEN(M$) STEP 2
90 LET C$=C$+MID$(M$,I,1)
100 NEXT I
110 FOR J=2 TO LEN(M$) STEP 2
120 LET D$=D$+MID$(M$,J,1)
130 NEXT J
140 LET M$=C$+D$
150 PRINT "CODED MESSAGE IS"
160 PRINT M$
170 END
```

Sets up empty string variables.

This means the length of your message minus 1.

N (your secret number) letters from the right of M$.

The length of M$ minus N number of letters from the left of M$ (i.e. the rest of the letters).

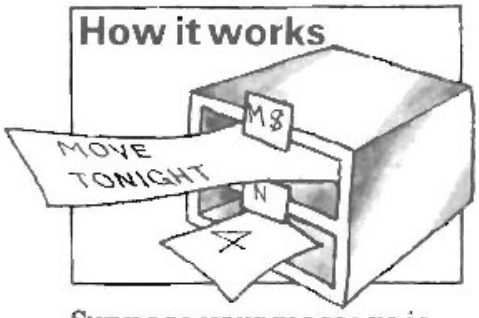
Replaces the letters in M$ with A$+B$.

From 1 to the number of letters in your message going up in twos, i.e. 1, 3, 5, etc. Each time the I loop is repeated line 90 takes one letter from position I of M$ and puts it in C$.

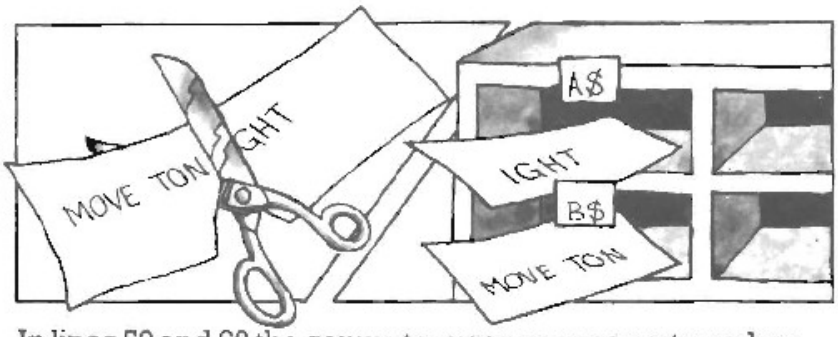
From 2 to the number of letters in your message, going up in twos, i.e. 2, 4, 6, etc. Works in the same way as the I loop.

Replaces the letters in M$ again.

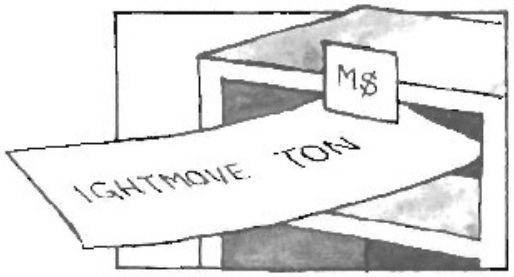
How it works



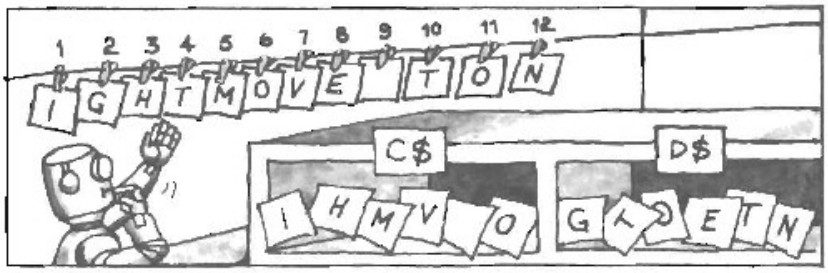
Suppose your message is "Move tonight" and your secret number is 4. These are stored in M$ and N.



In lines 50 and 60 the computer uses your secret number to divide the message. At line 50 it takes four letters from the right of the message and puts them in A$. At line 60 it puts the rest of the letters in B$.



At line 70 it adds A$ and B$. This puts the letters from the end of the message at the front.

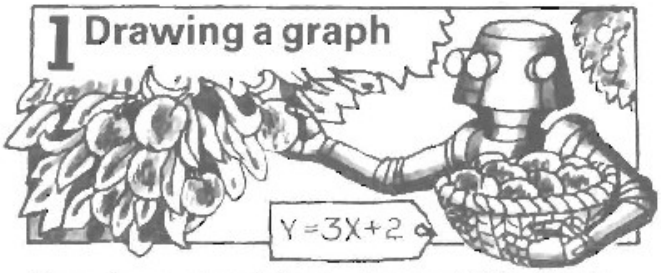


Each time the I loop repeats it puts an odd-numbered letter in C$ (e.g. I, H, M, etc.). Each time the J loop repeats it puts an even numbered letter in D$ (e.g. G, T, O, etc.). Then it adds C$ and D$ to make the coded message.

Graphs and symbols

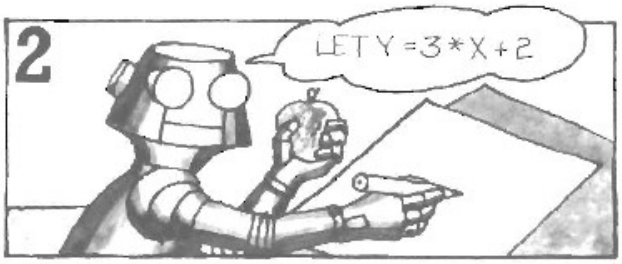
You can program a computer to present information in all kinds of different ways, for instance, as words, numbers, pictures or graphs. Complicated information can be made much easier to understand if you illustrate it with graphs, pictures and symbols.

1 Drawing a graph



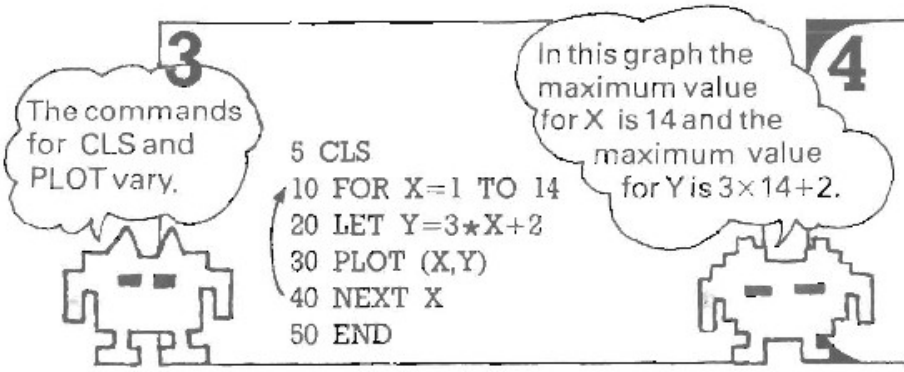
Imagine a peach tree whose yield of fruit increases each year in relation to its age. This can be expressed as an equation, say Y=3X+2 (Y is the yield and X is the age). It is hard to grasp what this means, though, and drawing a graph would help.

2



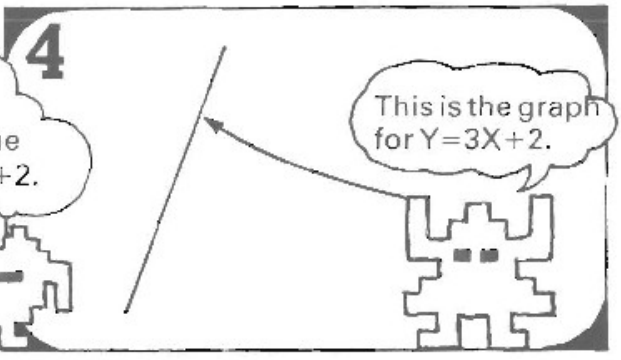
With a computer it is very easy to draw a graph of the way Y changes in relation to X. To plot the graph you need to find the value of Y for each value of X. You can do this very easily in a program using the statement LET Y=3*X+2.

3




```
5 CLS
10 FOR X=1 TO 14
20 LET Y=3*X+2
30 PLOT (X,Y)
40 NEXT X
50 END
```

4



This is the program for drawing this graph. The loop sets X at all the values from 1 to 14. Each time the loop is repeated, line 20 uses the value of X to calculate Y and line 30 plots X and Y on the screen. In graphs programs, you must make sure the maximum values for X and Y will fit on the screen or you will get a bug.

Computers and maths

In calculations which have several parts, such as 3×X+2, the computer always does the multiplications or divisions before it adds or subtracts. This means that the computer would give the same answer for these two sums:

```
PRINT 4*6+8        PRINT 8+4*6
32                 32
```

If you want the computer to do the sum in a different order you use brackets, like this:

```
PRINT (8+4)*6
72
```

This time the computer adds 8 and 4, then multiplies by 6.

Program puzzle

THINK OF A NUMBER
DOUBLE IT, ADD 4
DIVIDE BY 2, ADD 7
MULTIPLY BY 8, SUBTRACT 12
DIVIDE BY 4 AND TAKE AWAY 11
TELL ME THE RESULT.
THE NUMBER YOU FIRST THOUGHT OF IS ....

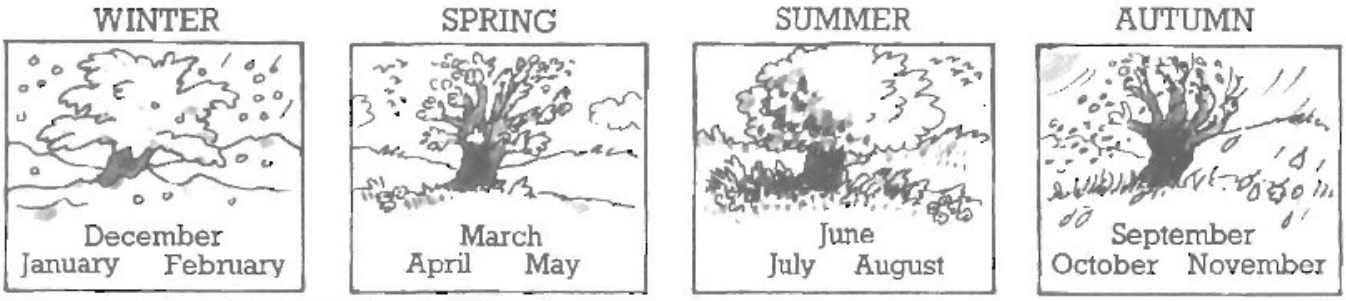
Can you write a program to get the computer to carry out this well known number trick? (To find the number you first thought of you subtract 4 from the result, then divide by 2.)

Birthdays program

This program uses another way to display information on the screen. It uses symbols to compare the number of people who were born in different seasons of the year. You could use a program like this to compare, say, sightings of a certain bird in different seasons, or the number of wins of different football teams. Before writing a long program like this it is a good idea to write a program plan.

Program plan

Aim: To compare the number of people with birthdays in winter, spring, summer and autumn.



1. Give the computer the data (i.e. the seasons when the people were born) for a survey of 20 people.
2. Store the data in the computer.
3. Present the data on the screen.

Sample run

```
RUN
WINTER  TOTAL*******
SPRING  TOTAL***
SUMMER  TOTAL*****
AUTUMN  TOTAL*****
```

The program

```
5 LET A=0
6 LET B=0
7 LET C=0
8 LET D=0
10 FOR I=1 TO 20
20 PRINT "PERSON ";I;" WAS BORN IN"
30 PRINT "WINTER, SPRING, SUMMER OR AUTUMN"
40 PRINT "TYPE W, SP, SU OR A"
50 INPUT B$
60 IF B$="W" THEN LET A=A+1
70 IF B$="SP" THEN LET B=B+1
80 IF B$="SU" THEN LET C=C+1
90 IF B$="A" THEN LET D=D+1
100 NEXT I
110 PRINT "WINTER TOTAL";
115 LET N=A
120 GOSUB 200
130 PRINT "SPRING TOTAL";
135 LET N=B
140 GOSUB 200
150 PRINT "SUMMER TOTAL";
155 LET N=C
160 GOSUB 200
170 PRINT "AUTUMN TOTAL";
175 LET N=D
180 GOSUB 200
190 STOP
200 REM: SUBROUTINE TO PRINT STARS
210 IF N=0 THEN GOTO 250
220 FOR I=1 TO N
230 PRINT "*";
240 NEXT I
250 PRINT
260 RETURN
```

Empty variables ready to use for running totals for each season.

Loop to make computer ask question once for each person in survey.

Lines 60 to 100 check the answer in B$ and add one to the variable for that season.

Sends computer back to repeat question.

The subroutine makes the computer print a number of stars equal to the number in each variable.

By putting the total into N each time, the computer can use the same routine for each season.

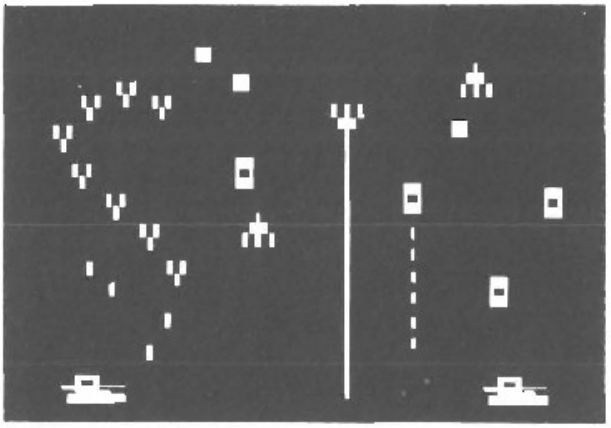
Makes the computer stay on the same line to print the stars.

Line 210 checks in case no-one was born in a particular season.

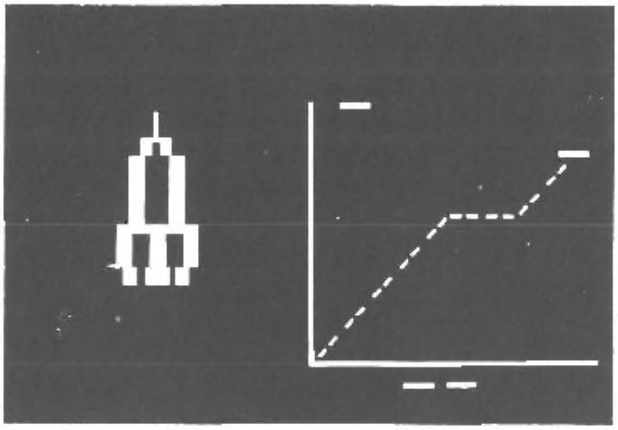
The main program sets N to the total for A, B, C or D. The loop makes the computer carry out line 230 "N" times.

More graphics

Those two pages show how you can use PLOT and UNPLOT to make moving pictures on the screen. Moving pictures are called animated graphics and they are useful for games programs, or to illustrate programs which explain, say, the principles of gravity or ballistics and flightpaths.



The pictures for video and arcade games are controlled by a small computer. The computer is programmed to play only the games and the programs are in the computer's own code, not in BASIC.

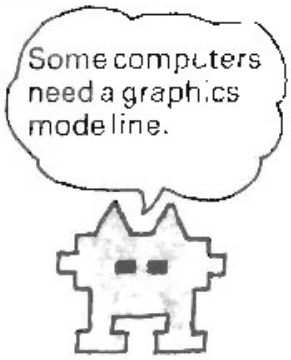


A general purpose microcomputer programmed in BASIC makes slower, simpler pictures. It cannot handle all the instructions for the screen quickly enough to make really fast moving graphics.

Plot/unplot program

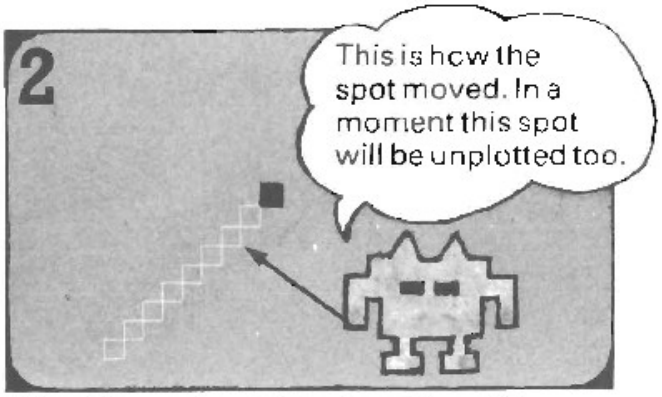
1

```
10 LET X=1
20 LET Y=1
30 PLOT (X,Y)
40 UNPLOT (X,Y)
50 LET X=X+1
60 LET Y=Y+1
70 GOTO 30
```



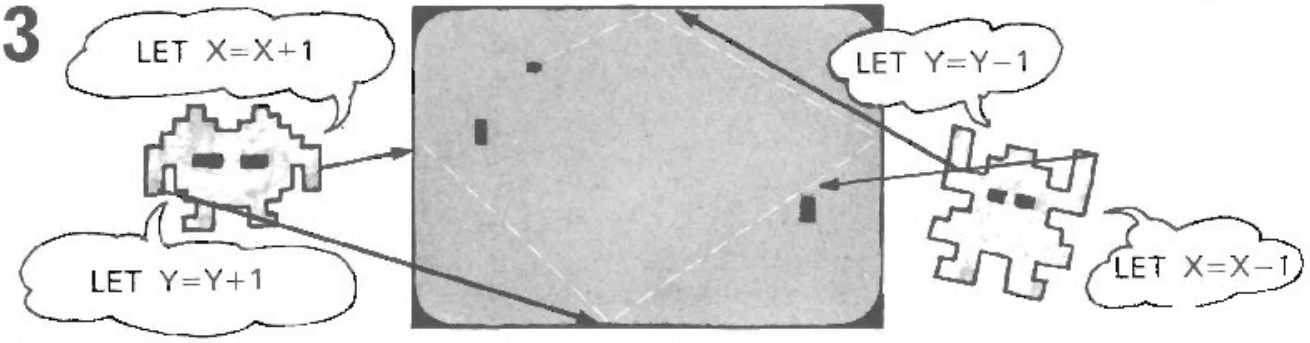
This short program makes a spot of light move across the screen. Remember, the commands for PLOT and UNPLOT vary on different computers.

2



When the spot reaches the edge of the screen the program may stop with an error message as the values for X and Y are outside the screen range of the computer.

3

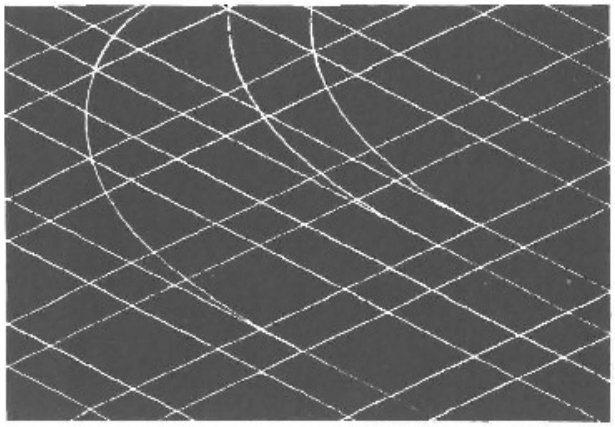


Bat and ball video games use programs like the one above to move the ball on the screen. There are simple program rules to keep the ball moving when it reaches the edge of the screen.

When the ball reaches the top of the screen the amount to be added to Y is subtracted instead. In the same way, when it reaches the right edge, the amount is subtracted from X.

Line pattern program

This program plots a line across the screen and when it reaches the sides, sends it back again in another direction. It does not use UNPLOT so the lines leave a pattern on the screen. The picture on the right shows what happens when you run the program. The program is set by line 100 to plot 10,000 pixels. You can change this figure to make it shorter, or BREAK the program at a pattern you like.



```
10 REM: SET UP GRAPHICS MODE HERE IF NECESSARY
20 PRINT "HOW MANY PIXELS ACROSS?";
30 INPUT H
40 PRINT "AND UP?";
50 INPUT V
55 CLS
60 LET X=H/2
70 LET Y=V/2
80 LET S=1
90 LET T=1
100 FOR I=1 TO 10000
110 LET S=S+(INT(RND(1)*10+1)-5)/50
120 LET X=X+S
130 LET Y=Y+T
140 IF X<5 THEN LET S=-S
150 IF X>H-5 THEN LET S=-S
160 IF Y<5 THEN LET T=-T
170 IF Y>V-5 THEN LET T=-T
180 GOSUB 300
190 NEXT I
200 STOP
300 REM: PLOT LINE
310 PLOT (X,Y)
320 RETURN
```

Lines 20 to 50 ask for the height and width of the screen. The semi-colon puts your reply on the same line as the question.

This makes X and Y start at the centre of the screen.

S and T are the amounts that will be added to X and Y to make the line move.

The loop from lines 100 to 190 is repeated 10,000 times Each time, X and Y are changed by a small amount.

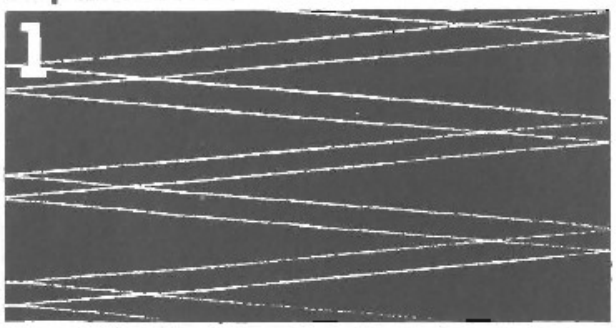
This gives a very small number to add to X. The number varies each time the loop is repeated.

These lines test for the edges and reverse S and T when X and Y come within five pixels of an edge.

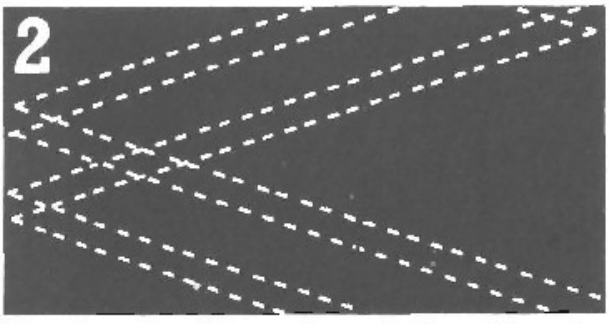
Sends the computer to the subroutine to plot the line

Plots the pixel with the current value for X and Y.

Experiments



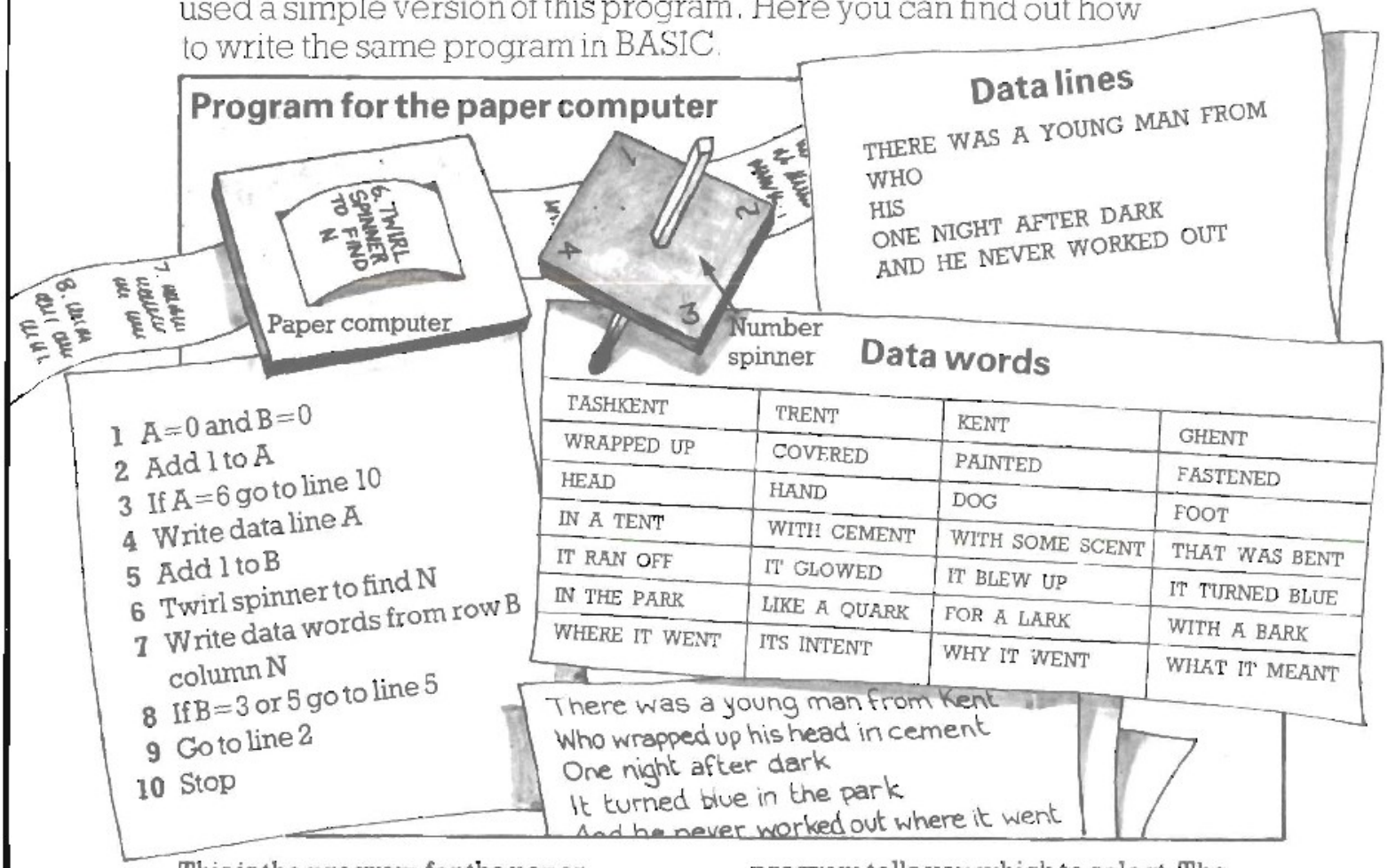
Line 110 adds a very small random amount to X each time and this makes the line wiggle across the screen. If you have a computer, try deleting this line. The lines on the screen should become parallel.



Try changing the numbers in lines 80 and 90 to, say, 5 or 10 (or larger on a computer with high resolution graphics). This makes the computer plot the pixels at intervals.

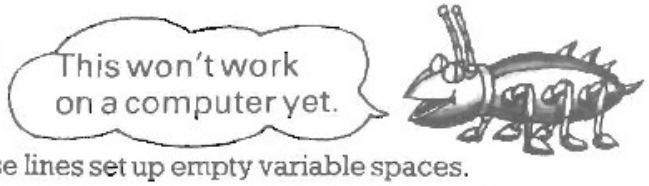
Funny poems program

The next few pages show you how to write a program which can compose lots of poems. A version of this program first appeared in the *Usborne Guide to Computers*. That book showed how to make a "paper computer" which used a simple version of this program. Here you can find out how to write the same program in BASIC.



This is the program for the paper computer. It looks a little like BASIC, but it would not work on a real computer. The words and phrases for the poem are "stored" on pieces of paper and the program tells you which to select. The number spinner is a random number generator to give random numbers between one and four.

1 Translating the program into BASIC




```
10 LET A=0
20 LET B=0
30 LET A=A+1
40 IF A=6 THEN STOP
50 Write data line A
60 LET B=B+1
70 LET N=INT(RND(1)*4+1)
80 Write data words from row B
   column N
90 IF B=3 THEN GOTO 60
100 IF B=5 THEN GOTO 60
110 GOTO 30
120 END
```

These lines set up empty variable spaces.

Lines 30 and 40 keep count of the number of data lines the computer has selected.

Lines 50 and 80 are not in BASIC yet.

Line 60 keeps count of the number of data words.

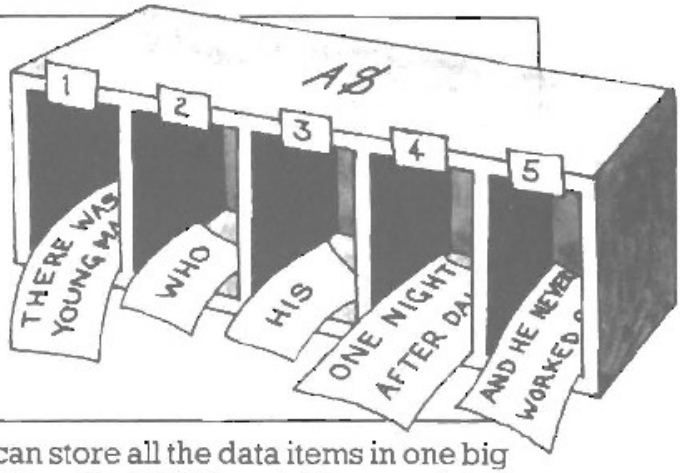
Gives a random number between 1 and 4

Lines 90 and 100 send it back to select another data line.

Most of the program is easy to translate into BASIC, but lines 50 and 80 are more difficult. The computer needs a way of storing and picking out the data lines and words which are needed for each line of the poem.

2 Giving the computer data

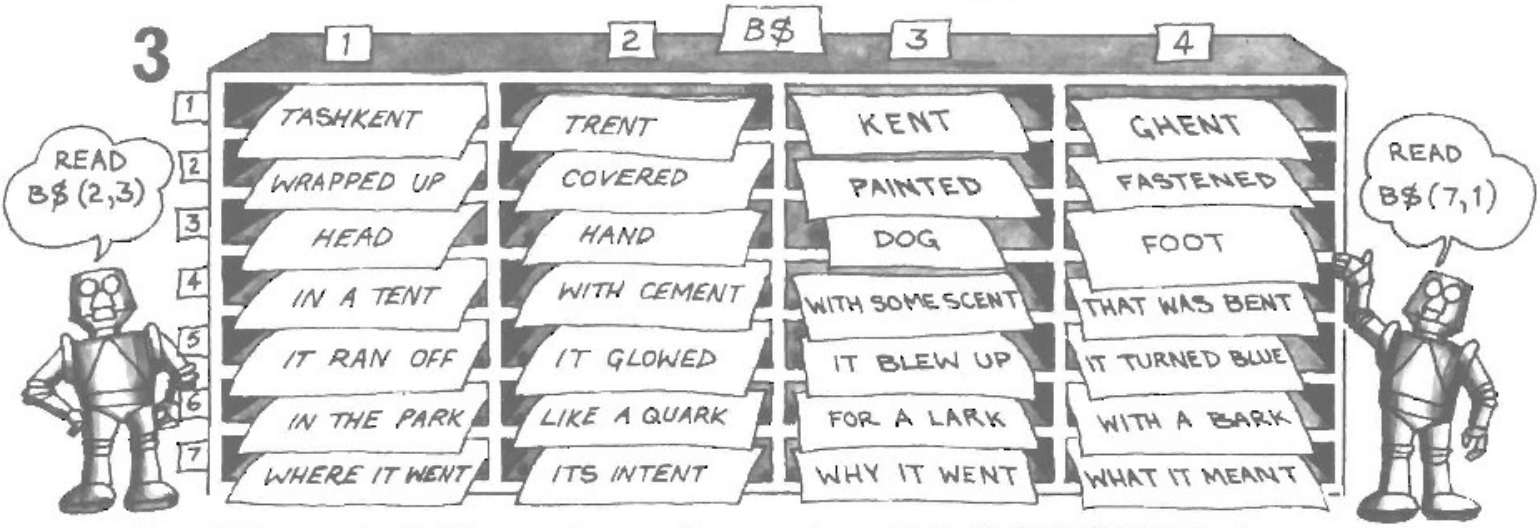
```
50 READ A$
:
180 DATA THERE WAS A YOUNG MAN FROM,
    WHO, HIS
190 DATA ONE NIGHT AFTER DARK, AND HE
    NEVER WORKED OUT
```



To give the computer the data lines and words you can use READ . . . DATA. Each time the computer carries out the READ instruction it takes another item from the DATA line and stores it in the variable.

You can store all the data items in one big variable called A$. A variable containing more than one data item is called an array and each item is referred to by a number, e.g. READ A$(3) gives HIS.*

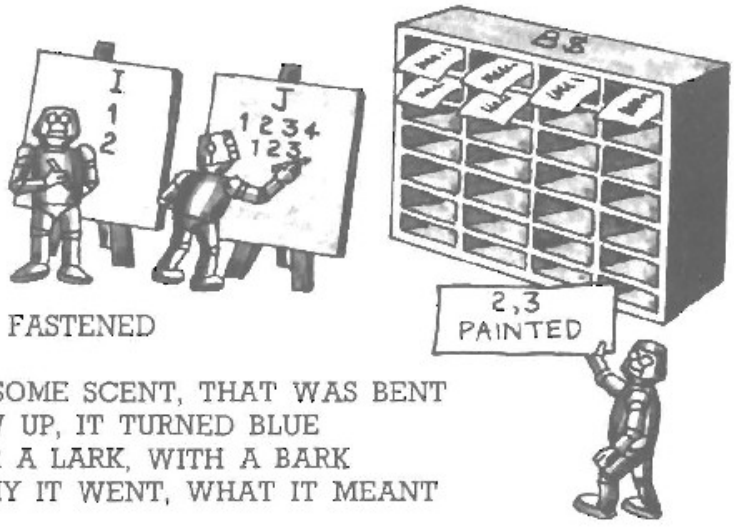
3



A variable can also hold several rows of data and you can store all the data words in a variable like this. It is called a two-dimensional array. Here, each data item is referred to by the number of the row and column it is in. So READ B$(4,2) gives WITH CEMENT and READ B$(6,3) gives FOR A LARK. You can store numbers in arrays, too, using a number variable, e.g. N(5,7).

4 Putting the data in the variables

```
10 FOR I=1 TO 7 }—— I is the row number
20 FOR J=1 TO 4 }—— J is the column
30 READ B$(I, J)        number
40 NEXT J
50 NEXT I
60 DATA TASHKENT, TRENT, KENT, GHENT
70 DATA WRAPPED UP, COVERED, PAINTED, FASTENED
80 DATA HEAD, HAND, DOG, FOOT
90 DATA IN A TENT, WITH CEMENT, WITH SOME SCENT, THAT WAS BENT
100 DATA IT RAN OFF, IT GLOWED, IT BLEW UP, IT TURNED BLUE
110 DATA IN THE PARK, LIKE A QUARK, FOR A LARK, WITH A BARK
120 DATA WHERE IT WENT, ITS INTENT, WHY IT WENT, WHAT IT MEANT
```



To read each data item into the variable you need to be able to alter the numbers in brackets after READ. You can do this with loops. B$ needs nested loops as shown above with an I loop for the row number and a J loop for the column number. Each time the I loop is carried out the J loop is repeated four times – once for each of the columns in a row.

*Sinclair computers deal with variables in a different way and this program will not run on a Sinclair. You can find out more about this over the page.

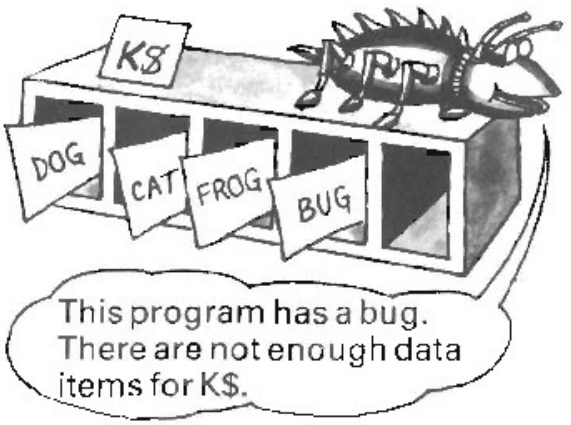
5 Making space for variables

```
5 DIM K$(5)
10 FOR I=1 TO 5
20 READ K$(I)
30 NEXT I,
40 STOP

60 DATA DOG, CAT, FROG, BUG
```

This is the size of the variable, i.e. 5 items in a row.

This line puts the data in K$ each time the loop is repeated.



At the beginning of the program you have to tell the computer how big you want the variable to be. You do this with the word DIM followed by the variable name and the number of data items, e.g. DIM K$(5).

For a two dimensional array you give the computer the number of rows and columns in the variable, e.g. DIM C$(5,3). You must always have the right number of data items for the variable or you get a bug.

6 Printing out the data

```
200 LET A=0
210 LET B=0
220 LET A=A+1
230 IF A=6 THEN STOP
240 PRINT A$(A)
250 LET B=B+1
260 LET N=INT(RND(1)*4+1)
270 PRINT B$(B,N)
280 IF B=3 THEN GOTO 250
290 IF B=5 THEN GOTO 250
300 GOTO 220
310 END
```

A keeps count of the number of times this section of the program is repeated.

B keeps count of the data word rows and makes sure that the correct row is used with each data line.

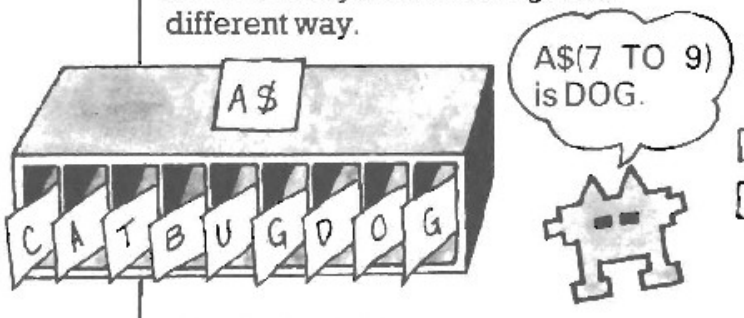
Lines 280 and 290 make the computer print out words from another data word row before printing the next data line.

This sends the computer back to print the next data line.

The computer needs these lines to print out the data lines and words in the right order. This section of the program is repeated five times. Each time, the computer prints out data line number A and some data words from row number B. The actual data words which are chosen are decided by random number N.

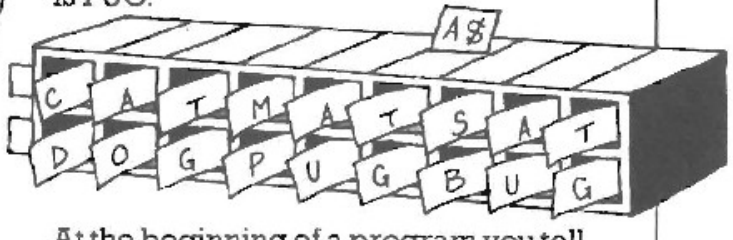
Sinclair computers and variables

This program does not work in its present form on Sinclair computers because they handle strings in a different way.



To tell a Sinclair computer to pick out a particular data item from a variable you have to give it the numbers of the first and last characters of the item you want. This is the same system as the Sinclairs use for LEFT$, RIGHT$, etc. (See page 32.)

For two-dimensional arrays you have to tell the computer the number of the row as well as the numbers of the characters. For instance, A$(2, 4 TO 6) is PUG.



At the beginning of a program you tell the computer how many rows the array has, and how many characters there are in each row, e.g. DIM A$(2,9) means two rows, each with nine characters. All the rows in the array must have the same number of characters.

The complete funny poems program

Now you can put the parts of the program together and write the complete poetry program. The first part of the program (lines 10 to 190) give the computer the data and the second part (lines 200 to 310) prints out the poem. Each time you run the program you get a different version of the poem because the random number N makes the computer pick different words.

```
 10 DIM A$(5)
 20 DIM B$(7,4)
 30 FOR I=1 TO 7
 40 FOR J=1 TO 4
 50 READ B$(I, J)
 60 NEXT J
 70 NEXT I
 80 DATA TASHKENT, TRENT, KENT, GHENT
 90 DATA WRAPPED UP, COVERED, PAINTED, FASTENED
100 DATA HEAD, HAND, DOG, FOOT
110 DATA IN A TENT, WITH CEMENT, WITH SOME SCENT, THAT WAS BENT
120 DATA IT RAN OFF, IT GLOWED, IT BLEW UP, IT TURNED BLUE
130 DATA IN THE PARK. LIKE A QUARK, FOR A LARK, WITH A BARK
140 DATA WHERE IT WENT, ITS INTENT, WHY IT WENT, WHAT IT MEANT
150 FOR I=1 TO 5
160 READ A$(I)
170 NEXT I
180 DATA THERE WAS A YOUNG MAN FROM, WHO, HIS
190 DATA ONE NIGHT AFTER DARK, AND HE NEVER WORKED OUT
200 LET A=0
210 LET B=0
220 LET A=A+1
230 IF A=6 THEN STOP
240 PRINT A$(A)
250 LET B=B+1
260 LET N=(RND(1)*4+1)
270 PRINT B$(B,N)
280 IF B=3 THEN GOTO 250
290 IF B=5 THEN GOTO 250
300 GOTO 220
310 END
```

Lines 10 and 20 tell the computer how much space to leave for the variables – a row of 5 for A$ and 7 rows of 4 for B$.

These are the nested loops for putting the data in B$.

Lines 80 to 140 contain all the data words to be stored in B$.

This is a loop to put the data into A$.

Lines 180 to 190 contain all the data lines to be stored in A$.

This prints the data line stored in A$ compartment number A.

This prints the data words stored in B$ row B, column N.

The program stops at line 230 when A=6, so it never reaches line 310, but some computers need an END anyway.

Sample runs

```
THERE WAS A YOUNG MAN FROM
KENT
WHO
WRAPPED UP
HIS
HEAD
IN A TENT
ONE NIGHT AFTER DARK
IT GLOWED
LIKE A QUARK
AND HE NEVER WORKED OUT
WHY IT WENT
```

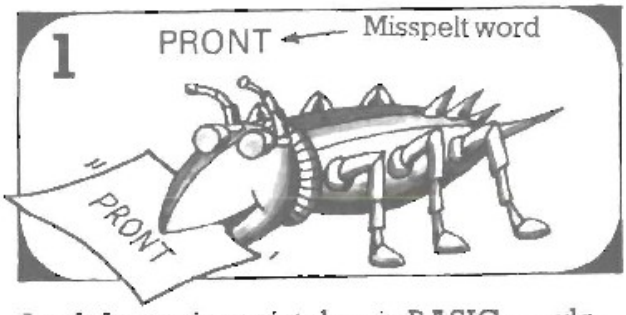
```
THERE WAS A YOUNG MAN FROM
GHENT
WHO
PAINTED
HIS
FOOT
WITH CEMENT
ONE NIGHT AFTER DARK
IT TURNED BLUE
WITH A BARK
AND HE NEVER WORKED OUT
ITS INTENT
```

Here are two of the 16,384 possible different versions of the poem. If you try this program and always get the same poems, look in your manual for how to make the computer produce different random numbers. Some computers produce the same sequence of random numbers each time they are switched on.

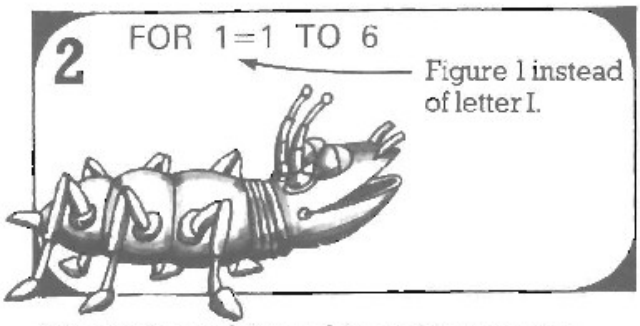
Programming tips

On these two pages there are some tips to help you write your own programs, and a list of the most common bugs you might get, and what causes them. The most likely bugs are listed first, so if you have a program which will not work, check through this list until you find the reason.

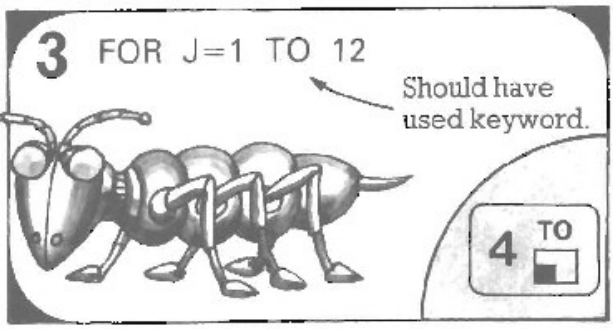
Finding bugs



Look for typing mistakes in BASIC words. If you misspell one of these words the computer will not recognize it.



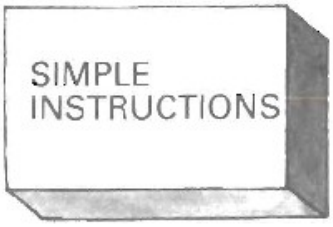
Check Os and 0s and 1s and Is to make sure you have typed the right ones in the right places.



If you have a Sinclair computer, make sure you have not typed a word in letter by letter instead of pressing the key for that word.

Writing programs

When you are writing programs it helps to remember that the computer can carry out three main activities: simple instructions, repeating things and making decisions. These are the building blocks of all programs.




```
LET A=3
LET N=N+1
PRINT A/T
PLOT (X,Y)
```



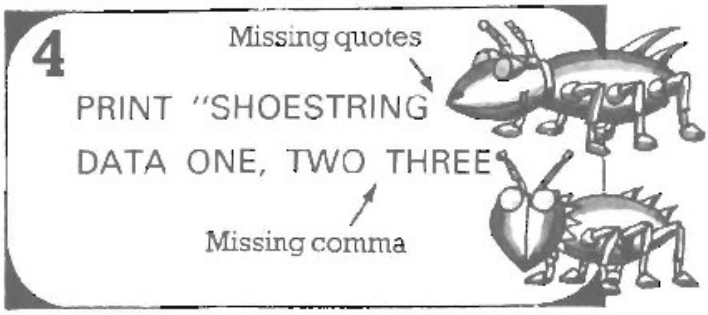

```
FOR J=1 TO 6

20 LET A=1
30 IFA< 10 THEN
   GOTO 100
```



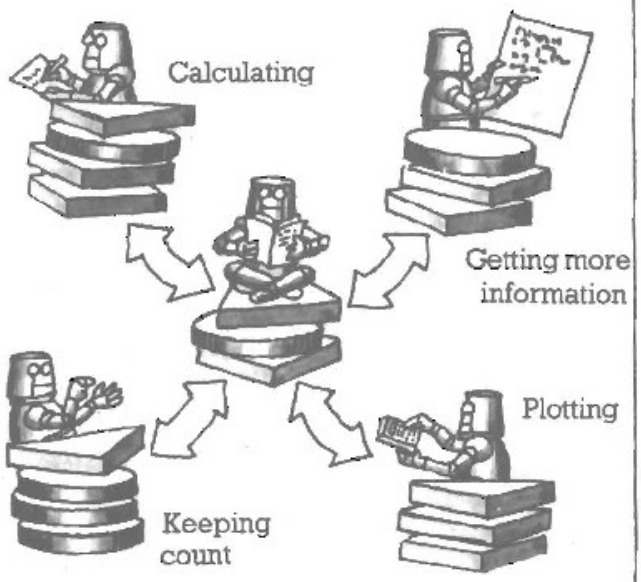

```
IF X=Y THEN STOP
IF K$="HELLO"
THEN PRINT A
```

This book has covered all the main instructions you need in BASIC to tell the computer to carry out these activities. When you are writing a program, work out what the computer needs to do at each stage, then decide which instructions you want to use.

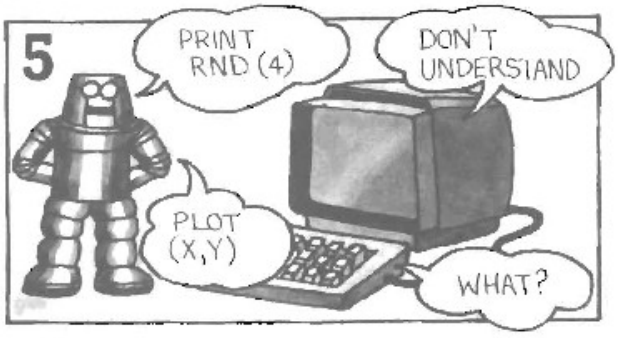


Make sure you have not left out any quotation marks, or the commas between data items. Check complicated lines which have lots of symbols especially carefully.

There are usually several different ways to write a program and some of them may be neater and shorter than others. When you are writing a long program it is a good idea to divide it up into lots of sections with subroutines to carry out each activity. The central core of the program may be a simple set of instructions, decisions and repeats which controls when and how often the computer carries out the subroutines.



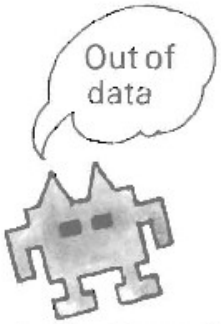
Breaking up programs into sections like this makes it much easier to find any mistakes. Each section can usually be tested by itself without running through all the program. Remember to label each section with a REM line so you know what it is for.



Make sure you use the correct RND, PLOT and CLS commands for the computer. Check, too, that you have given the computer a general graphics line if it needs one.

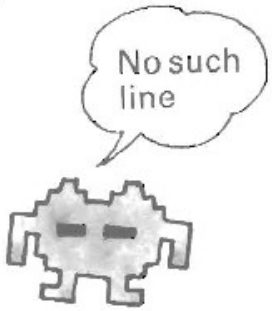
Error messages

All computers print out some sort of message when there is a bug in the program and the messages are explained in the computer's manual. Here are some of the most common messages you may get.



◀ This means there are not enough data items for the computer to read in the DATA lines. It may be because you have missed out a comma between two items, so the computer has read them as one.

▶ The line with the number given in a GOTO or GOSUB statement does not exist. You may have accidentally erased the line by typing in another line with the same number, or you may have just mistyped the number.





◀ You may get this report on a BBC or Sinclair computer. It usually means you have not set up a variable with a line such as LET C=0 or LET C="" before using it.

▶ This means the NEXT line of a loop is missing. It may be because you typed the wrong variable name, or even put a 1 instead of an I so the computer did not recognize it.



Last word

Some bugs are very hard to find, but if the computer will not run the program there must be a bug in it somewhere. If you really cannot find the bug, try typing in suspect or complicated lines again, you might get them right the second time without even noticing what the bug was.

Puzzle answers

Page 15
Name and message program

```
10 PRINT "WHAT IS YOUR NAME"
20 INPUT N$
30 PRINT "HELLO"
40 PRINT N$
50 PRINT "HOW ARE YOU"
```

Page 17
1. Sums program

```
10 LET A=9
20 LET B=7
30 PRINT A*B
40 PRINT A/B
50 LET A=A+1
60 LET B=B+3
70 PRINT A*B,A/B
80 END
```

Comma to leave space

2. Tables program

Spaces

```
30 PRINT A;" TIMES ";B;" IS ";A*B
40 PRINT A;" DIVIDED BY ";B;" IS ";A/B
```

3. Name and message alterations

```
10 PRINT "WHAT IS YOUR NAME"
20 INPUT N$
30 PRINT " HELLO ";N$;" HOW
   ARE YOU
```

Page 18
Sums program

```
10 PRINT "WHAT IS 7 TIMES 7"
20 INPUT A
30 IF A=49 THEN PRINT "CORRECT"
40 IF A<>49 THEN PRINT "NO";7*7
```

You need a semi-colon after the quotes, like this.

Page 19
Age guessing game

Replace line 30 and add a new line 35:

```
30 IF G<14 THEN PRINT "OLDER
   THAN THAT"
35 IF G>14 THEN PRINT "YOUNGER
   THAN THAT"
```

Page 23
Plotting counter

```
 5 LET C=0
45 LET C=C+1
50 IF C<6 THEN GOTO 10
```

Plotting your initial

Here is an example of a program to plot the letter L.

```
10 LET X=15
20 LET Y=30
30 PLOT (X,Y)
40 LET Y=Y-1
50 IF Y>5 THEN GOTO 30
60 LET X=X+1
70 PLOT (X,Y)
80 IF X<45 THEN GOTO 60
90 END
```

Page 24
Random numbers

The formula for a random number between 10 and 20 would be INT(RND(1)*11+9). On computers which need only a number in brackets after RND, it would be RND(11)+9. There are eleven possible numbers between 10 and 20 so you need to pick random numbers between 1 and 11, then add 9.

Page 25
Space attack

These are the lines you need to add to count the number of hits:

```
15 LET S=0
75 IF X=A*B THEN LET S=S+1
95 PRINT "YOU HIT "
   ;S;" OUT OF 6 ALIENS"
```

Page 27
1. Eight times table

```
10 PRINT "THE EIGHT TIMES TABLE"
20 FOR J=1 TO 12
30 PRINT J;"x8 = ";J*8
40 NEXT J
```

Page 27
2. N times table

```
10 INPUT "TYPE IN A NUMBER";N
20 PRINT "HERE IS THE "
   ;N;" TIMES TABLE"
30 FOR I=1 TO 12
40 PRINT I;" TIMES ";N;" IS ";I*N
50 NEXT I
60 INPUT "ANOTHER NUMBER (Y or N)"
   ;M$
70 IF M$="Y" THEN GOTO 10
```

For the ZX81 you need separate PRINT and INPUT lines.

Page 32
Computer book string puzzle

LEFT$(A$,8) is "COMPUTER"
RIGHT$(A$,10) is "PUTER BOOK"
MID$(A$,5,8) is "UTER BOO"

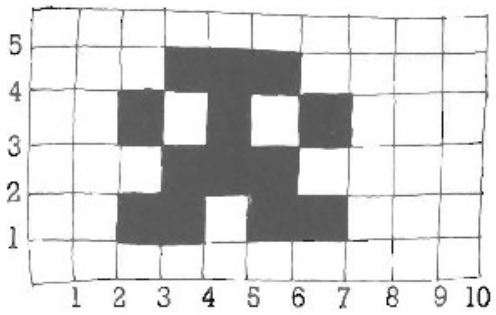
Page 34
Number trick program

```
10 PRINT "THINK OF A NUMBER"
20 PRINT "DOUBLE IT, ADD 4"
30 PRINT "DIVIDE BY 2, ADD 7"
40 PRINT "MULTIPLY BY 8,
   SUBTRACT 12"
50 PRINT "DIVIDE BY 4 AND TAKE
   AWAY 11"
60 PRINT "TELL ME THE RESULT"
70 INPUT N
80 PRINT "THE NUMBER YOU FIRST
   THOUGHT OF IS";(N- 4)/2
```

You need the brackets to make the computer do the sum in the order you want.

Space invaders repeat program

1



Draw a simple space invaders shape on squared paper.

2

4,5; 5,5; 6,5

3,4; 5,4; 7,4

4,3; 5,3; 6,3

3,2; 4,2; 6,2; 7,2

Then work out the co-ordinates of all the squares which make up the space invader.

3

```
5 CLS
50 INPUT "HOW MANY POINTS ACROSS THE SCREEN";W
60 INPUT "HOW MANY UP";V
65 CLS
70 FOR I=0 TO V STEP V/6
80 FOR J=0 TO W STEP W/6
130 NEXT J
140 NEXT I
150 END
```

Change the 6 to a higher figure to increase the number of times the invader shape repeats on the screen. (If you get a bug you have made the number too big.)

Put your plot lines here, e.g.
90 PLOT (J+3, I+2)
92 PLOT (J+4, I+2)
for the two bottom left-hand squares of the space invader shown above. You need a program line for each square.

Copy out the pattern repeat program, excluding lines 10 to 40 and 90 to 140, as shown above. (These lines produce the random pattern for the program so you do not need them.)
Now you need to put your own plot lines into the program between lines 80 and 140 (you can renumber the lines in the program). For each pair of co-ordinates you need to add J to the first figure and I to the second figure, to make the space invader repeat.

BASIC words

Here is a list of the BASIC words used in this book, with short explanations of what they mean. Some of the words, such as CLS, are not standard on all computers and these words have a small star beside them. If you have a micro you should check these commands in your manual.

* **BREAK** On some computers this stops the program running. Be careful, though, on others it erases the whole program from the computer's memory and you should use ESCAPE, or some other word instead.

* **CLS** Clears the screen.

* **DATA** A list of items, e.g. words or numbers, to be stored by the computer in variables. See READ.

DIM Tells the computer how many memory spaces it should set aside for a variable. E.g. DIM A$(5,4) means the variable needs five rows of four columns.

* **EDIT** Allows you to alter a line in a program without typing in the whole line again.

* **END** Tells the computer it is the end of a program. Some computers must always have an END statement, others, such as the BBC micro and Sinclair computers, do not need one.

FOR . . . NEXT Makes the computer loop back through the program and repeat any instructions inside the loop a fixed number of times.

GOSUB Makes the computer leave the main part of the program and go to a part called a subroutine to carry out a special task.

GOTO Tells the computer to go to another line in the program.

IF . . . THEN Compares pieces of data (e.g. numbers or words or the contents of variables) and does different things depending on the results.

INPUT A way of getting the computer to ask you for data while the program is running.

INT Converts a number with a decimal point to a whole number by ignoring all the figures to the right of the decimal point. E.g. INT(3.40) = 3.

* **LEFT$** Tells the computer to do something with a number of characters from the left-hand side of a string. E.g. LEFT$(A$,4) means take four characters from the left of A$.

LEN Gives the length of a string, i.e. the number of characters in a variable.

Computer words

Array A set of variables containing several pieces of data.
Bug A mistake in a program.
CPU The central processing unit of the computer which controls all the operations and does all the work, e.g. comparing variables, adding, etc.
Cursor A small, sometimes flashing light, square or other shape on the screen which shows where the next character will be printed.
Flow chart A chart showing the main operations needed in a program. Often used as an aid to writing programs.
Graphics Ways of producing information visually on the screen.
Kilobytes (K) A unit of measurement for the memory of a computer. One kilobyte is 1024 bytes and in most micros each character takes up one byte.
Prompt A question mark or other symbol which appears on the screen when the computer asks for information after an INPUT statement.

LET Puts a variable label on a memory space and puts some information in it. E.g. LET N=4 or LET B$="CATS".

* LIST Displays the program listing on the screen.

* MID$ Tells the computer to do something with characters from the middle of a string. E.g. MID$(A$,4,3) means take three letters starting from the fourth letter of A$.

NEW Wipes the program from the computer's memory to clear it for the next program.

* NEWLINE KEY Tells the computer that you have finished typing in a program line or piece of input. Some computers have keys marked RETURN or ENTER.

NEXT See FOR.

* PLOT Tells the computer to light up a pixel. E.g. PLOT (X,Y) means light up the pixel with co-ordinates X along and Y up.

PRINT Tells the computer to display something on the screen.

* READ Tells the computer to read the information in a DATA line and store it in a variable. See DATA.

* READY Some computers say this when they are ready to be given another instruction.

REM The computer ignores lines starting with REM but displays them in the program listing. They are useful to remind you what different parts of the program do.

RETURN At the end of a subroutine, tells the computer to go back to the instruction after the one where it left. See GOSUB.

* RIGHT$ Tells the computer to do something with the right-hand characters in a string. E.g. RIGHT$(A$,4) means take the four characters from the right of A$.

* RND Picks a random number.

RUN Tells the computer to carry out a program.

SQR Tells the computer to find the square root of a number.

STEP Used with FOR . . . NEXT loops. Tells the computer when to repeat the loop.

STOP Used within a program to tell the computer to stop running the program.

THEN See IF.

* UNPLOT Tells the computer to switch off a pixel.

Pixels Short for picture elements. The small squares which the computer can light up on the screen to make pictures.
Program A numbered list of instructions telling the computer how to carry out a particular task.
RAM Random Access Memory. The memory inside the computer where the program and data are stored. All the information in RAM is automatically erased when the computer is switched off.
ROM Read Only Memory. Permanent memory where information telling the computer how to operate is stored by the manufacturers.
String A series of characters for storing in a variable, e.g. "SAUSAGES" or "ABC123".
Subroutine A section of the program for carrying out a particular task which is usually repeated several times during the running of the program.
Syntax error A mistake in the BASIC in the program.
Variable A labelled memory space which contains a piece of information.

Going further

The best way to learn to write programs is to try out your programs on a computer. If you do not have a computer, there may be somewhere where you can go and use one. Try asking at your local school, or at the library, or go along to a computer user's group and see if you can borrow some time on someone's computer.

You can learn a lot about writing programs from reading and typing in programs written by other people. Here is a list of books about computers and programming that you may find useful.

Usborne Guide to Computers by Brian Reffin Smith, Usborne 1981
Understanding the Micro by Judy Tatchell and Bill Bennett, Usborne, 1982
The Computer Book by Robin Bradbeer, Peter DeBono and Peter Laurie, BBC 1982
Illustrating Computers by Colin Day and Donald Alcock, Pan 1982
Illustrating BASIC by Donald Alcock, Cambridge University Press 1977
Computer Spacegames by Daniel Isaaman and Jenny Tyler, Usborne 1982
Computer Battlegames by Daniel Isaaman and Jenny Tyler, Usborne 1982

Index

The numbers in **bold** type show the pages on which each term is first explained.

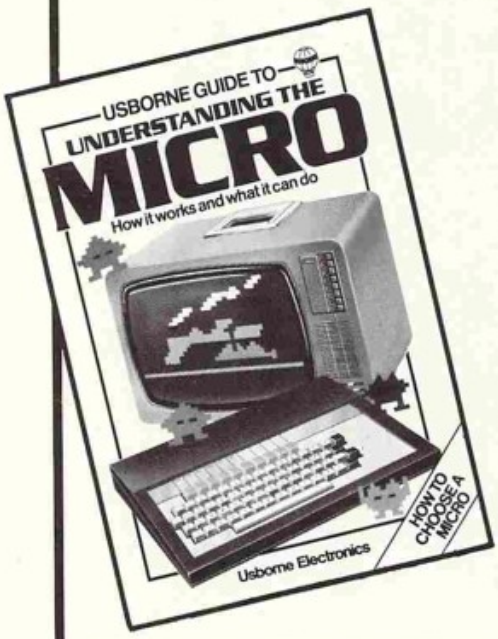
ABS, **31**
addition, 16, 34
Age guessing program, 19
Age program, 18
animated graphics, 36
arrays, 39, 40, 41
BASIC, 7, 9, 10, 36, 38, 42
BBC micro, 21, 22, 43
Birthdays program, 35
brackets, 34
BREAK, **15,** 23, 25, 37, 47
bugs, 8, 9, 11, 28, 40, 42-43
cassette recorder, 5, 10
Circles program, 31
Clever computer program, 21
CLS, **10,** 15, 20 25, 27, 28, 29, 34, 37, 43, 47
Codemaker program, 33
commas, **13, 16,** 17, 28, 42, 43
Computer clock program, 28
computer,
  code, 6, 7, 36
  languages, 4, 6
Conversion program, 31
COPY, **11**
counters, variables used as, **21,** 23, 25, 26, 31, 35, 38, 40
CPU (central processing unit), 5
cursor, 10
data, 4, 12-15, 18, 38-39
DATA, **13,** 21, 31, 39, 40, 41, 43, 47
debugging programs, 11, 42-43
delay loops, 27, 28
DELETE key, **11**
DIM, **40,** 41, 42, 47
division, 16, 34
EDIT, **11,** 47
Eight times table program, 26
END, **11, 47**
ENTER key, **10**
error messages, 11, 36, 43
ESCAPE, **15,** 25, 47
Face program, 11
flowchart, 9
FOR . . . NEXT, **26-29,** 33, 34, 35, 37, 39, 41, 43, 47
French lesson program, 18
Funny poems program, 38-41
games programs, 20, 24-25, 36
GOSUB, **30-31,** 35, 37, 43, 47
GOTO, **19,** 20, 21, 23, 25, 31, 38, 40, 41, 43, 47
graphics, 22-23, 25, 29, 31, 36-37, 43
graphs, 34
greater than, **18,** 31, 37
Greedy computer program, 27
Hello loop program, 26
high level languages, 7
high resolution graphics, 22, 29, 37
IF . . . THEN, **18-19,** 20, 21, 23, 25, 28, 31, 35, 37, 38, 40, 41, 47
INPUT, 10, **14-15,** 21, 47
INT, **24,** 47
interpreter, 6, 10
keyboard, 4
LEFT$, **32-33,** 40, 47
LEN, **32,** 33, 47
less than, **18,** 20, 23, 25, 31, 37
LET, **12,** 13, 17, 32, 47
Line pattern program, 37
LIST, **11,** 15, 47
loops, **26-29,** 33, 34, 35, 37, 39, 41, 43
maths and sums, 16, 17, 34
Maths program, 19
memory, 5, 6, 12, 14, 15, 22
MID$, **32,** 33, 47
MODE, **22**
multiplication, 16, 34
nested loops, **28-29,** 39, 41
NEW, **15,** 47
NEWLINE key, **10,** 15, 47
NEXT, see FOR
Numbers program, 31
number variables, 12, 13, 14
Pascal, 7
Pattern repeat program, 29
Pilot, 7
pixels, 22-23, 25, 29, 37
PLOT, **22-23,** 25, 29, 34, 36, 37, 43, 47
poetry writing program, 15
PRINT, **10-11,** 12, 15, **16-17,** 47
printer, 5
program, 4, 6, 7, 8-9
  line numbers, 11, 30, 43
  plans, 9, 35, 42-43
puck, 23
pugs, 9
Quiz program, 31
quotation marks, 10, 12, 16, 17, 42
RAM (random access memory), 5
random,
  numbers, **24-25,** 28, 29, 37, 38, 40
  number tester program, 28
  pattern program, 25
READ, **13,** 21, 31, 39, 40, 41, 43, 47
REM, **27,** 30, 31, 35, 37, 43, 47
RETURN, see GOSUB
  key, 10, 47
RIGHT$, **32,** 33, 40, 47
RND, **24-25,** 28, 29, 37, 38, 40, 41, 43, 47
ROM (read only memory), 5
RUBOUT key, **11**
RUN, **10-11,** 15, 47
semi-colons, **16,** 17, 21, 35, 37
Silly sums program, 26
Sinclair computers, 39, 40, 42, 43
Space attack program 25
Space commando program, 20
spacing words on the screen, 16, 21
square roots, 16
STEP, **27,** 29, 33, 47
STOP, **19,** 30, 31, 35, 38, 40, 41, 47
strings, 12-13, 14, 32-33
subroutines, 30-31, 35, 37, 43
subtraction, 16, 34
THEN, see IF
UNPLOT, **22,** 36, 47
variables, 12-15, 17, 18, 21, 27, 32, 35, 38, 39, 40
  as counters, 21, 23, 25, 26 31, 38, 39, 40
Weather program, 18
ZX81 computer, 13, 19, 21, 32

First published in 1982 by Usborne Publishing Ltd, 20 Garrick Street, London WC2E 9BJ, England.

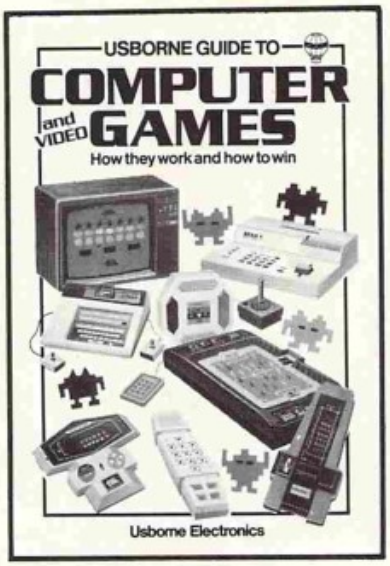
© 1982 Usborne Publishing
The name Usborne and device are Trade Marks of Usborne Publishing Ltd. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the publisher.

Usborne Computer Books

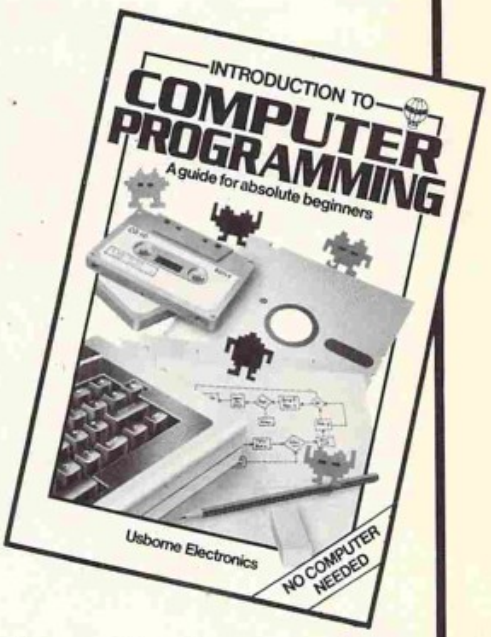
Computers are fun. You can play games with them, ask them questions, write poetry with them and play music on them too. This colourful new series of books shows you some of the exciting things computers can do and explains how they work and how to use them. Written in clear and simple language with lots of pictures, these books provide a fun introduction to computers and computing for absolute beginners.



Understanding the Micro
A colourful guide to microcomputers, how they work and what they can do, with lots of ideas for things you can do with a micro.



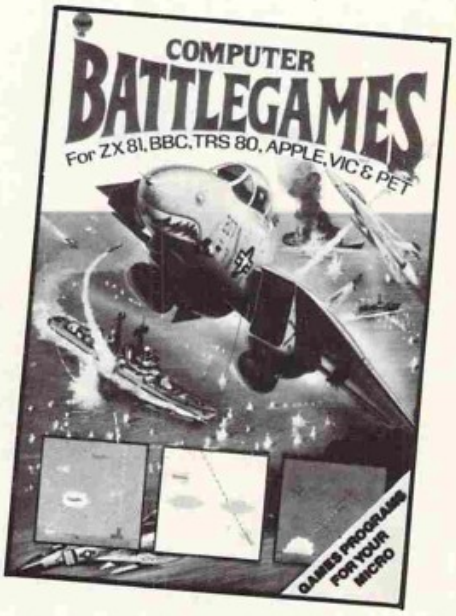
Computer Games
A colourful look at how computers play Space Invaders, chess and other games, with lots of tips on how to beat the computer.



Computer Programming
A step-by-step guide to programming in BASIC for absolute beginners. With lots of programs to run on any microcomputer.

Usborne Computer Fun
These two superbly illustrated books are packed with games programs to play on a microcomputer. Each game is suitable for use on the most common micros, and there are lots of tips and hints for writing your own programs.





Published in the USA by
EDC PUBLISHING, 8141 E. 44th Street,
Tulsa, Oklahoma 74145, USA.

Published in Canada by Hayes Publishing
Ltd, 3312 Mainway, Burlington, Ontario,
Canada, L7M 1A7.

ISBN 0 86020 674 2