Derek Ellershaw and Peter Schofield

VERY BASIC BASIC

The first 15 hours on your

COMMODORE 64

Derek Ellershaw and Peter Schofield

VERY BASIC BASIC

The first 15 hours on your COMMODORE 64



CENTURY PUBLISHING CO. LTD.
WITH
MELBOURNE HOUSE (PUBLISHERS) LTD.

Published in the United Kingdom by:
Century Publishing Co. Ltd
Portland House,
12-13 Greek Street,
London W1V 5LE
in conjunction with
Melbourne House (Publishers) Ltd.,
Church Yard,
Tring, Hertfordshire HP23 5LU
ISBN 0 7126 0412 X

Copyright © 1984 by D. Ellershaw and P. Schofield

All rights reserved. This book is copyright. No part of this book may be copied or stored by any means whatsoever whether mechanical or electronic, except for private or study use as defined in the Copyright Act. All enquiries should be addressed to the publisher.

Reproduced, printed and bound in Great Britain by
Hazell Watson & Viney Limited,
Member of the BPCC Group,
Aylesbury, Bucks

CONTENTS

SECTION 1 **PRE-PROGRAMMING** 1

CHAPTER 1 INTRODUCTION
Don't Panic 3
The First Step 3
How to use this Book 4
Before Programming 4
Basic Techniques 4
Useful Aids 5
More About Programming 5

CHAPTER 2 KEYBOARD BASICS
Before you Begin 7
How to use the Different Modes 8
To Summarise so far 9
Instruction Keys 9
More Pre-Programming Hints 11
To Clear the Memory 12
Take Care With Zeros 13
And With Spaces 13

CHAPTER 3 HOW TO CORRECT MISTAKES
Introduction 15
Correcting Typing Errors Before Pressing RETURN 15
Correcting Typing Errors After Pressing RETURN 16
Correcting Errors in a Program which you are trying to RUN 17
Exercises 19

CHAPTER 4 THE REM COMMAND
Its Purpose 21
Its Effect 21
Its Value 22

CHAPTER 5 USES OF THE PRINT COMMAND
The Print Instruction 23
Example 23
A Better Example 24
How Layout can be Affected 24
The Effect of Speech Marks 26
Screen Layout 26
Positional Printing 27
The TAB Function 27

CHAPTER 6 HOW TO USE END
Introduction 31
The END Command 31
Example 32
Why END Can Have No Effect 32
Another Example 32
When It Stops 33
Exercises 34

SECTION 2 **BASIC PROGRAMMING TECHNIQUES** 35

CHAPTER 7 HOW TO USE INPUT
Getting Information 37
Using INPUT 37
When You Enter the Wrong Thing 38
Let's Explore Deeper 38
Another Example 40
You Can Use These Examples 41
Using INPUT for Letters and Words 41
A Handy Program for Christmas 42
The Christmas Program Explained 42

CHAPTER 8 NUMBERS AND YOUR COMPUTER
Introduction 45
Addition + ... 45
Subtraction – 46
Explanation 46
Multiplication * 46
Division / 47
Order of Calculations 47
A Program to List Numbers 48
Explanation 48
N is a "Numeric Variable" 49
Naming and Giving a Value
to a Numeric Variable 49
What is Allowed as a Name 49

More Uses of Numeric Variables 50
Explanation 51
Two New Words — FOR and NEXT 51
To Repeat a Group of Instructions 52
Explanation 52
The Computer as a Calculator 53
A Program for the 10 Times Table 54
Random Numbers and Rounding 54
A List of Random Numbers Using RND and INT 56
A List of Calculations is Easily Printed 56
Some Useful Formulae 57
Varying the Level of Difficulty 58
Different Ways to Use the Computer 58
When the Computer Uses Decimal 60
Raising a Number to a Power 60
Dealing With Fractions 60
LET is Not Compulsory 61

CHAPTER 9 STRINGS AND THINGS!
Introduction 63
How to Recognise a String Variable 63
An Example 65
A Use for the Example 66
Adapting the Example 66
Using the Adaptation 67
Another Use of String Variables 67
Another Example — A Quiz 67
String Variables Do Vary 68
A True String Variable Explained 68
Well Done! 68
Using the Quiz 68
A Primitive Word Processor 69

CHAPTER 10 LOOPING AND BRANCHING
Using GOTO to Repeat 71
Going Round in a Loop 71
An Example of Other Uses of GOTO 71
Over To You 72
Explanation 72
How an IF-THEN Statement Works 73
Things to Notice 74
String Variables 74
Exercises 75

CHAPTER 11 SCREEN CLEARING
Introduction 77
To Recap 77
Consider This Program 77
Explanation 78
The Untidy Result 78
A Neater Approach 78
A Slower, More Readable Result 79

CHAPTER 12 SUBROUTINES
Introduction 81
When to Use GOSUB 81
For Example 81
Explanation 82
An Improved Example 82
Explanation 83
Conclusion 83

CHAPTER 13 SUMMARY 85

SECTION 3 **USEFUL AIDS** 87

CHAPTER 14 USING THE PRINTER
The Printer is Useful 89
Warning! 89
Commands for the Printer 90
Printing a Program Listing 90
A PRINT Statement for the Printer 90

CHAPTER 15 USING THE TAPE RECORDER
Get the Right Tape Recorder 91
How to Use Pre-recorded Programs 91
How to Save a Program on Tape 91

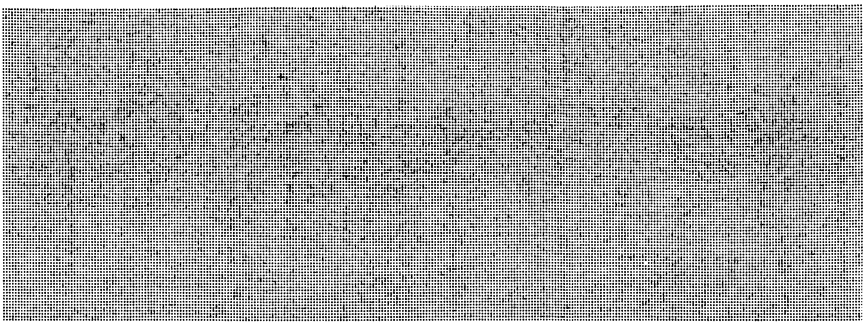
SECTION 4 **MORE ABOUT PROGRAMMING** 93

CHAPTER 16 CONDITIONAL STATEMENTS
IF is a Conditional Statement 95
Example 95
Explanation 95
= and < > Can Also Be Used
With Words 96
More Signs 96
Example 96
Explanation 97
Two Important Words Used With IF 98

The Meaning of "AND" 98
An Example of Its Use 98
Logical Operators Can Be Used
With Words 99
An Example of the Use of "OR" 100
An Example of the Use of "NOT" 101

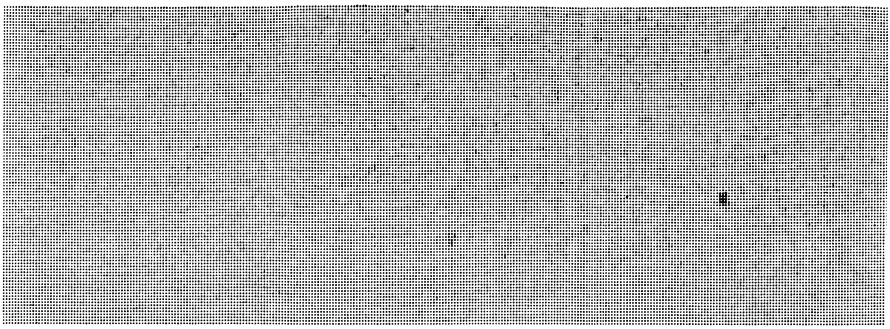
CHAPTER 17 HOW TO HANDLE DATA
An Example of "READ"
and "DATA" Explained 103
Now You Go On 104
Another Example Explained 104
And It's Over To You 106
An Example of Arrays 106
A Different Use of DIM 107
Sorting Letters 109
Now You Can Experiment 110

CHAPTER 18 HOW TO ADAPT PROGRAMS
Introduction 111
A More Complex Program 113
Explanation 114
The Program Teaches As Well As Tests ... 114
How to Alter the Program 114
Another Example 115
How to Adapt It 116
Variations on a Theme 117
A Test of Logic 118



SECTION 1

PRE-PROGRAMMING



CHAPTER 1
Introduction

DON'T PANIC

This introduction tells you how to use the book and explains how the book is laid out.

By now you may have unpacked your computer, plugged everything in and perhaps have run a few games on it. You may even have tried some programs. If you are finding the manual difficult to follow, if you are finding that you understood the first few lines and very little after that, then THIS IS THE BOOK FOR YOU. Don't feel ashamed! Don't burst into tears of frustration! Thousands before you have found computing difficult to understand. Only a very few people understand computers and programming from the very start. Many people feel that computing books go too far, too fast, too soon. This book is written for adults, teenagers, or any beginner. It avoids the use of jargon for its own sake and explains computing in English. New and technical terms are always fully explained.

THE FIRST STEP

This book does NOT aim to tell you everything about your computer. It is very limited in its scope. It tries to guide you through the first hours and weeks of understanding, when confusion can often creep in. However, if you follow it carefully, it should provide a basis which will help you to understand the more complex books on the subject of programming.

HOW TO USE THIS BOOK

(1) Look carefully at the Sections and Chapters at the front of the book. They are set out very clearly because you will have to use them when you start to program. You will often be referred back or forward to another chapter if you want a more detailed explanation, but only do so if you really want an immediate explanation.

(2) You should ideally start at the first Chapter and work through the book unless the instructions indicate otherwise. The Chapters are arranged in a sequence. There are exercises in some chapters. Make a real effort to attempt them to get the maximum benefit.

(3) The book is split into four main sections.

BEFORE PROGRAMMING

SECTION 1:

This section outlines a number of things you need to know before you can begin programming. Yes, there are some simple programs, but the main weight of the section is on helping you to be clear about using easy commands and knowing how to set out your programs.

BASIC TECHNIQUES

SECTION 2:

These chapters concentrate on the bare bones of programming techniques (for more information see Section 4). You are introduced to variables, looping, branching, counting, together with the associated commands. We have aimed to make all the programs have a practical application, rather than make them mere theoretical exercises. We aim to help fathers and

mothers, daughters and sons to discover how computers can help with education and leisure.

USEFUL AIDS

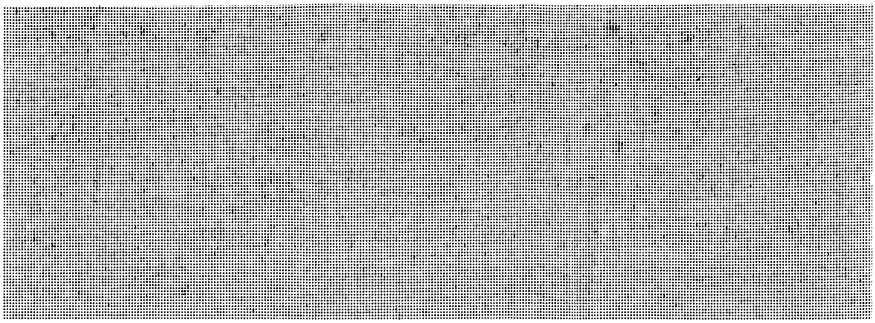
SECTION 3:

You may find it odd to have chapters on using the tape recorder and printer in this section but this is quite deliberate. You do NOT need a tape recorder to program, you do NOT need a printer. We also take a very narrow view of the use of sound, graphics and colour, for the purposes of this book. Its prime purpose is not to teach how to use graphics, nor how to use sound and colour. Therefore, we do not include material on these features.

MORE ABOUT PROGRAMMING

SECTION 4:

The purpose of this section is to take you a little further into programming.



CHAPTER 2
Keyboard Basics

This chapter is concerned with clearing a number of obvious things out of the way so that you can begin programming.

BEFORE YOU BEGIN

(a) Plug the aerial lead into the T.V. socket at the back of the computer and the other end into the aerial socket of your television.

(b) Plug the mains lead into the MAINS socket at the right hand side of the computer.

You are now ready to start. BUT if you want to connect your tape recorder and printer either now or in the future — proceed as follows:

(c) If you have a tape recorder, connect it to the computer via the plug at the back of the computer. (Turn to Chapter 15 if you want to know about loading tapes.)

(d) If you have a printer, connect this carefully via its plug at the back of the computer.

IMPORTANT NOTE: Do not plug in or unplug the printer whilst the computer is switched ON. You will damage your computer.

(e) Plug into the mains. Select a spare channel on your T.V. set and tune the television until the screen displays the word READY and a flashing rectangle known as the CURSOR.

(f) You are now ready to begin computing.

(g) If in doubt at any stage, switch off at the mains and start again.

HOW TO USE THE DIFFERENT MODES

The computer is now READY to accept instructions from you — the flashing CURSOR acts as a prompt — in other words it is encouraging you to type some instruction. Whenever you see the word READY and the flashing cursor you know that the computer is waiting for you.

Press the number keys on the top line of the keyboard and then try some letters of the alphabet. Now press some of the keys with symbols (e.g. + , − =). This shows that the computer is automatically locked into capitals. Now find one of the SHIFT keys, there is one on the bottom row at each side, either will do, and hold it down. Whilst holding it down press the number keys, excluding Ø, again. This time you will see that the symbols above the numbers have been displayed. Now press all the letters of the alphabet, again whilst holding down SHIFT. This time the computer has displayed the right hand graphics on the front of each key. To obtain left hand graphics on the front find the  key which is on the left of the bottom line and hold it down. Now press the letters again. Fascinating isn't it! The SHIFT LOCK key can be used if you need to type a lot of SHIFTed keys. Try it. But don't forget to press it again to release it.

Now find the CTRL key — it is on the second row, left hand side. This is the ConTRoL key and with it you can change the colours of your display. Whilst holding it down press the number keys from 1 to 7 — one at a time

of course. Now look at key number 8 — you will see the letters RVS ON. Whilst holding down CTRL press 8 and then type some letters — this key is the Reverse key — in other words it reverses the letters. To get out of this mode hold down CTRL and press 9 RVS OFF.

TO SUMMARISE SO FAR

(a) We refer to the flashing rectangle as a CURSOR (it means that the computer is ready).
(b) The computer is set initially in capitals (upper case) mode.
(c) With the SHIFT, SHIFT LOCK keys we can obtain the symbols written above the numbers or the right hand graphics on the front of the keys.
(d) With the ◀ key we can obtain the left hand graphics set and the small (lower case) letters.
(e) With the CTRL key we can change colour and reverse the display.

INSTRUCTION KEYS

You will have noticed other keys with instructions written on them. Let us have a look at these.

Your screen should still be displaying numbers, letters, graphics etc. and you should also have the flashing cursor. Press the key marked RETURN. You will see the following:

? SYNTAX ERROR
followed by READY and the flashing CURSOR.

The words SYNTAX ERROR mean that the computer does not understand the message you have typed in. This is to be expected because we are only 'playing' with the keyboard and not giving instructions. We can ignore the error message for now (it is

explained more fully later) and continue playing because we have the word READY and the flashing cursor tells us that the computer is awaiting further instructions.

The RETURN key (sometimes known as ENTER on other computers) is the most important key of all because it is this key that is used to enter the information typed (∴ displayed) into the computer. After every instruction line of a program typed, the RETURN key must be pressed. You will soon become used to this.

Now find the key marked CLR HOME and press it. You will see that the flashing cursor is now occupying the position at the top left hand corner of the screen. This is known as the HOME position — hence the word HOME on the key. Now hold down one of the SHIFT keys and again press CLR HOME. This time the screen has been cleared and the cursor has remained in its 'Home' position.

Let us now look at the CRSR keys. There are two of these on the bottom line at the right hand side. These are the CURSOR control keys and with them you can move the cursor to any position on the screen.

First press the CRSR ↑ ↓ several times and you will see the cursor move down the screen. To move it back up again use the same key but this time hold down SHIFT while doing so. Try holding down the CRSR key as well. Try the other CRSR key. The CURSOR moves in the direction of the arrow.

Experiment with this function and type your name at different positions on the screen.

We will explain the functions of the other Command keys as we come across them in later Chapters.

Practice using the keyboard to familiarise yourself with it.

MORE PRE-PROGRAMMING HINTS

The following is useful as a reference list. On their own the points made may not seem to mean much at this stage but you will need to refer back to this list as you work through the next few pages until you become familiar with the workings of the machine.

Type this. (Make sure your screen is clear. Press SHIFT and CLR HOME, together.)

```
10 PRINT 3
```

You will notice that the computer doesn't leave a space unless you do. Therefore, to leave a space, press the long space bar below the keys.

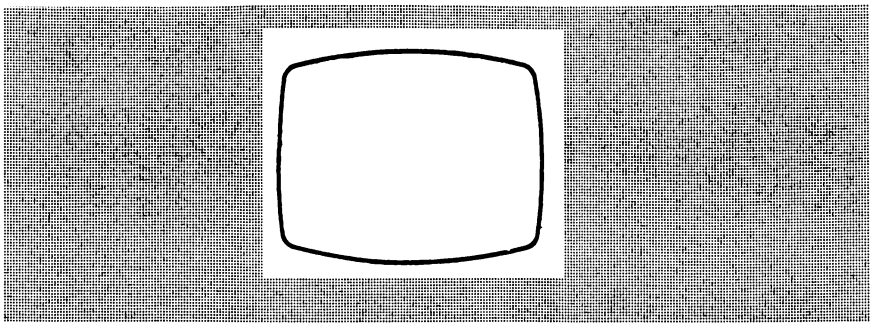
(a) When you first type something and it appears on the screen the information is NOT in the computer's memory. To get it into its memory, you must press RETURN. You must press RETURN to get each line into the computer's memory.

Now press RETURN. You will notice that the flashing cursor has moved down to the next line expecting the further instruction. The computer is storing the previous instruction.

(b) You cannot get the computer to do anything with your information unless you type the word RUN. Until then, it simply stores your instructions. Type RUN and then press RETURN.

3 appears beneath the word RUN because your instruction was — PRINT 3.

Congratulations. You have just RUN your first program! Note that we put a 10 at the start of the line. All programs have their lines numbered e.g. 10, 20, 30 etc. They are important for the computer to know the order in which it should carry out instructions.



We number lines 10, 20, 30 so that if a line has been missed out of a program, you can insert it as line 11, say, without retyping the rest of the program line numbers e.g. 10, 11, 20, 30.

(c) TO SUMMARISE
When writing a program, to get a line of information into the computer you must give it a line number, type in the information, and then press RETURN.

To get the computer to carry out your instructions you must type RUN and then press RETURN.

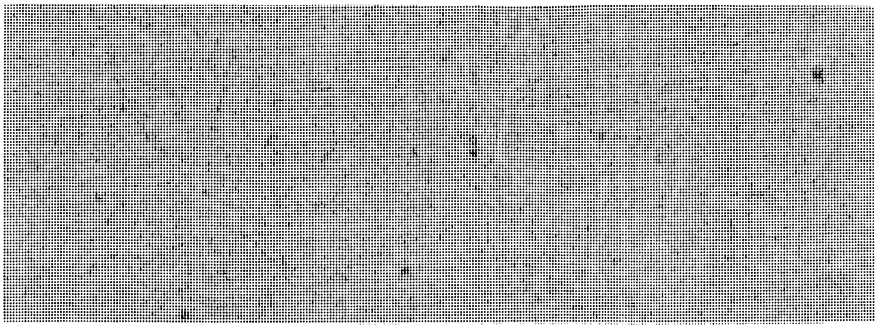
TO CLEAR THE MEMORY

To clear information out of the computer's memory completely in order to start again you can:

(a) Whilst holding down the SHIFT key, press the CLR HOME key, type NEW and then press the RETURN key.

(b) Or switch the computer off.

Whenever you do either of the above — everything stored in the computer's memory is lost — for ever. (More detail on screen clearing, is given in Chapter 11.)



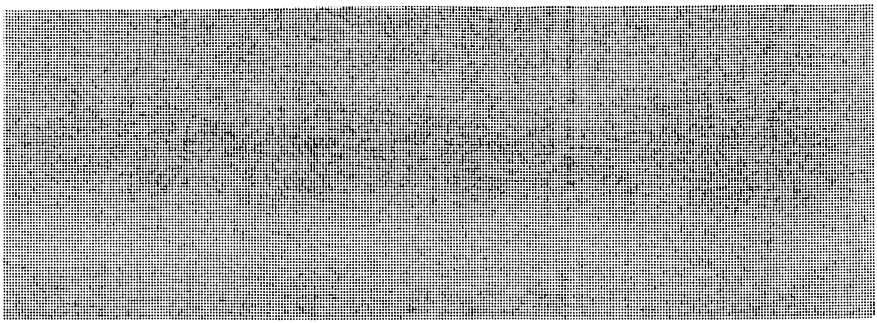
To LOAD tapes see Chapter 15.
To SAVE programs see Chapter 15.
To PRINT programs using a Printer see Chapter 14.

Be careful not to press Ø for O. They are not interchangeable. Ø is always a number.

TAKE CARE WITH ZEROS

AND WITH SPACES

In this book, ▲ means make sure that you leave a space, i.e. Press the Space Bar once, or twice if ▲▲ appears.



CHAPTER 3

How to correct mistakes

INTRODUCTION

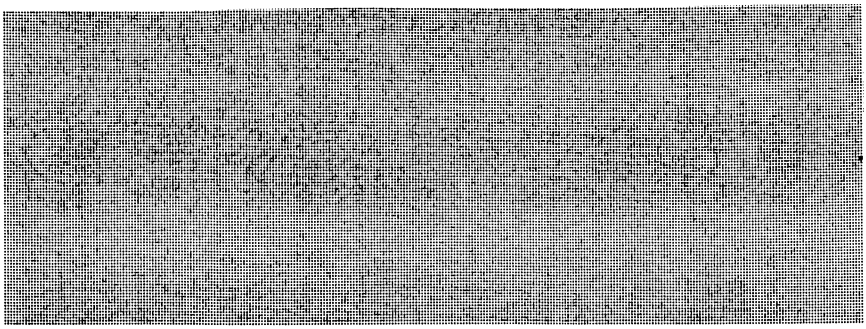
When you begin to write programs you will make mistakes, you may not spot these until you try to RUN the programs. Also, you will make simple typing errors. This short chapter shows how to correct these and other mistakes.

CORRECTING TYPING ERRORS BEFORE PRESSING RETURN

(a) Consider the following first line in a program. You are ready to press RETURN and you find a typing error. Type this line with the error.

10 REM THIS IS TZ SHOW YOU HOW TO CORRECT MISTAKES.
Do not press RETURN.

(b) Correct the mistake by pressing the INST DEL key (right hand side of top line). This will rub out or DELETE your typing one letter at a time. When you have deleted the mistake type in the corrected line. The INST DEL key is particularly useful if you realise that you have just made a typing error and want to correct it immediately. It is less useful if you have typed in a long line or if you have pressed RETURN.



CORRECTING TYPING ERRORS AFTER PRESSING RETURN

If you have pressed RETURN and then spot the typing error, this is what to do.

(a) Either retype a new line completely using the same line number. The computer will replace the incorrect line with the corrected line.

(b) Or by moving the cursor back to the incorrect line — using the CRSR ↑↓ key — and then using the CRSR ←→ key to move the cursor to the end of the line and DELETE with the INST DEL key to enable you to correct the line as in paragraph 3(b).

However:

(c) It is possible simply to DELETE your error as follows.

Type this line:
10 PRINT 2 4 7 8
press RETURN.

Now by using the CRSR ↑↓ key move up the screen to line 10. Now using the CRSR ←→ key position the cursor after the number 7. It will in fact flash on the number 8. Now press the INST DEL key. The number 7 has disappeared, and its place has been taken by the number 8 and the flashing cursor. Hold

```
1Ø PRINT 4
2Ø PRINT 16
3Ø PRINT "JOHN"
4Ø PRINT  MARY"
5Ø PRINT "ELSIE"
```

down SHIFT and press the INST DEL key again. This time the SHIFT key has allowed the INST function of the computer to work. It has in fact made room to INSerT the correction by moving the number 8 along the line! Type 6 and then press RETURN. That looks better!

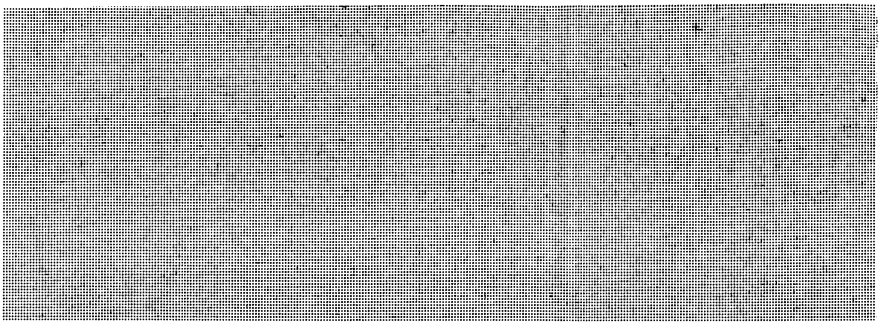
CORRECTING ERRORS IN A PROGRAM WHICH YOU ARE TRYING TO RUN

Imagine that you have typed in the above program lines and went to RUN the program.

Type in this program exactly as it appears and then type RUN. (Don't forget the RETURN key after each line and after you have typed RUN.)

The computer will carry out the instructions at lines 10, 20, 30 and 50 by printing 4, 16, JOHN, ELSIE but it will print Ø on the next line after JOHN because we have forgotten to put the " (speech marks) after the instruction PRINT. (See Chapter 5 for an explanation on the use of speech marks with letters and numbers.) This mistake can be corrected by:

(a) Type in a new line 40 as follows:
40 PRINT "MARY"
press RETURN and then type RUN followed by the RETURN key.



(b) Using the CRSR ↑↓ CRSR ←→ keys and INST DEL as explained in para 3c.

(c) Or by using another command LIST. This is an important command which you will need. It tells the computer to display the whole program. You have corrected line 40 and RUN it — to prove that the corrected line has been put in place of the incorrect line type LIST and press RETURN. The updated program is displayed.

LIST can be used in a variety of ways.

You should still have the listing of your program on the screen. Add these two further lines:
60 PRINT 2468
70 PRINT "HELLO"
don't forget RETURN. Now type LIST again.

You have already seen that LIST on its own displays the full program listing.

Now type LIST 40 followed by RETURN. Line 40 has appeared for you to examine. Type LIST to obtain the program again.

Now type LIST 40 – (minus sign on top row).

The computer has now LISTed the program from line 40 to the end.

Get the program listing back and type LIST 40–60. This displays lines 40 and 60.

Try LIST 20–60

You will appreciate how useful this function can be when you start to write programs of your own. You will be able to

```
1Ø REM THIS IS AN INKORRECT LIST.
2Ø PRINT "I HAV A COMPUTER"
3Ø PRINT "IT IS GUD FUN"
4Ø PRINT COMPUTER
```

LIST any part of the program to correct errors or simply to examine them.

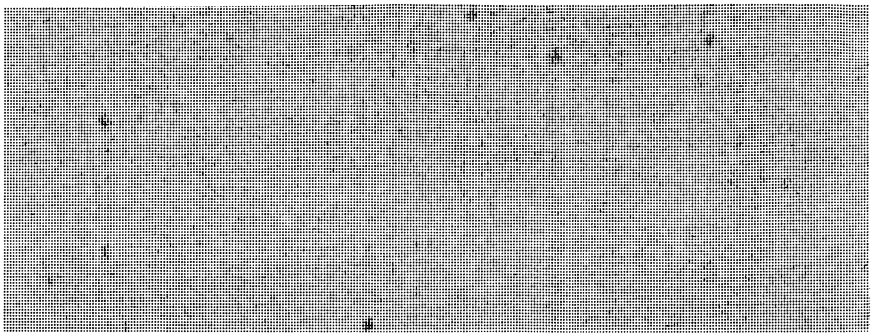
Experiment with the commands you have come across in this Chapter.

EXERCISES

(1) Type in the above lines as they appear and correct each one BEFORE pressing RETURN. Remember to clear the computer's memory, as discussed previously, every time you start on a new program.

(2) Type the following program pressing RETURN after each new line. Then using the CRSR keys and INST DEL correct each line and display the full program listing before RUNning it.

```
1Ø REM THERE IS A MISTAKE ON EVERY LINE.
2Ø CORRECT EACH AFTER EACH LINE HAS BEEN RETURNED.
3Ø PRINT HELLO AGAIN
4Ø PRINT "3 + 3 = 7"
5Ø PRINT THIS IS THE END.
```



CHAPTER 4

The REM Command

ITS PURPOSE

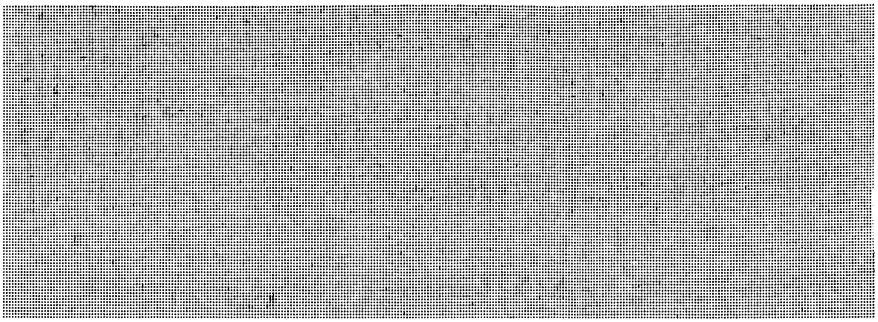
It is important to remember that the REM statement does not have a computer function nor does it affect the running of a program. Its purpose is to act as a reminder or remark for your benefit when writing your own programs. You can use it to remind yourself of the title of a program, or to remind you what each section of a program is going to do next.

ITS EFFECT

Type this simple program into your computer remembering to press the RETURN key after each line!

```
1Ø REM BANK STATEMENT
2Ø LET CAR = 15Ø
3Ø PRINT CAR
4Ø LET GAS = 7Ø
5Ø PRINT GAS
```

Now type RUN and press the RETURN key. Your screen will show 150 followed by 70. You will notice that nothing has happened as a result of line 10. Delete line 10 — to do this simply type 10 and press RETURN. Now RUN the program again to convince yourself that REM has no effect on the program.



ITS VALUE

You will find that the REM statement is of increasing use to you as you write programs, to remind you of the different sections of a complex program.

CHAPTER 5

Uses of the PRINT command

THE PRINT INSTRUCTION

The PRINT instruction is one of the most frequently used commands in programming. It is also one of the easiest to understand! You have already used it in Chapters 3 and 4.

EXAMPLE

Type this into your computer — (don't forget RETURN after each line)

```
1Ø PRINT "JOHN"
2Ø PRINT "SMITH"
3Ø PRINT "35"
4Ø PRINT "ACACIA AVE"
5Ø PRINT "ANYTOWN"
```

Now type RUN and press RETURN. This simply shows how to print a list of words or phrases under each other.

Note that each phrase is within speech marks.

```
10 PRINT "JOHN SMITH"
20 PRINT "35 ACACIA AVE"
30 PRINT "ANYTOWN"
```

A BETTER EXAMPLE

The above version of the same program is neater.

Type and RUN it after clearing memory.

Look at these extra lines:

```
15 PRINT
25 PRINT
```

The addition of lines 15 and 25 will leave a space between each line of the address — lines 15 and 25 don't PRINT anything. If you have a printer you could print out your own address labels from this program.

HOW LAYOUT CAN BE AFFECTED

Look carefully at the effects that slight differences in the use of the comma, the semi-colon and the space ▲ can have on a layout:

Type in the program that appears at the top of the next page. (Remember ▲ means press space bar.)

Interesting isn't it.

(a) Lines 10, 20, 40, 50 and 70 all print the numbers together with the last figure (4) at a point 5 columns across. Leaving a space (▲) after PRINT in lines 20, 40 and 70 has no effect, nor does the positioning of the semi-colon in line 50. Compare line 50 with

```
10 PRINT 1234
20 PRINT ▲1234
30 PRINT,▲1234
40 PRINT;▲1▲2▲3▲4
50 PRINT;▲1234
60 PRINT "1234"
70 PRINT▲1▲2▲3▲4
80 PRINT 1; 2; 3; 4
90 PRINT 1, 2, 3, 4
```

line 80 — here the semi-colon has resulted in the computer leaving two spaces between each number.

(b) Line 60 however is affected by the speech marks. The computer now displays the numbers tight up against the left hand margin. It displays it in the first available column.

(c) The comma after PRINT at line 30 instructs the computer to begin its display at column 12 while the commas in line 90 display the numbers in four columns. Your computer has four "fields" and the comma instructs it to display at 10 column intervals.

(d) It has been explained that spaces (▲) used on their own have no effect on the display. Add this line:

100 PRINT "1▲▲2▲▲3▲▲4"
and RUN.

What has happened this time? That's right, the spaces (▲) have been printed this time because they are within speech marks.

(e) Type this line:

110 PRINT "1,2,3;4;"
and RUN it.

Commas and semi-colons within speech marks have no effect on the display!

(f) That's enough to start with! Experiment yourself with different spacings. You can see how important being able to print in columns can be.

```
1Ø PRINT "WHAT IS 3 + 2?"
2Ø FOR T = 1 TO 3000 : NEXT T
3Ø PRINT "THE ANSWER IS" 3 + 2
```

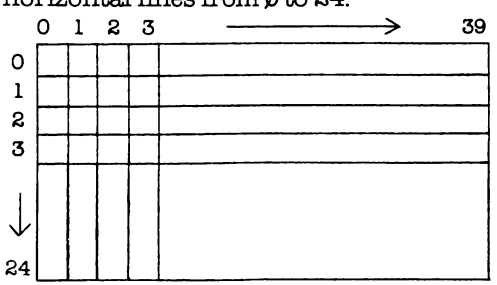
THE EFFECT OF SPEECH MARKS

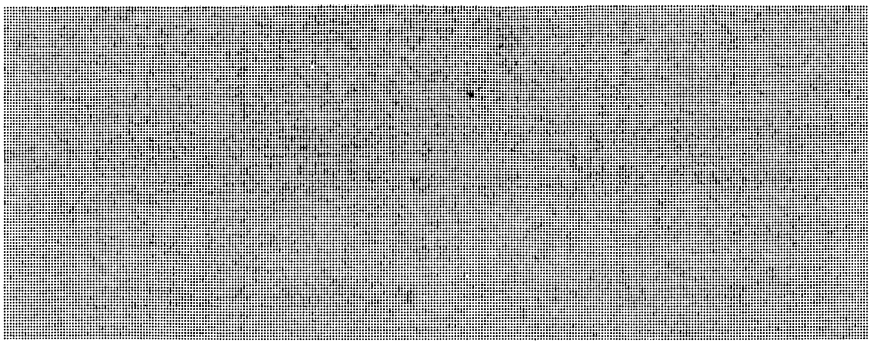
Type and RUN the above program.

Line 10 PRINTS the question. Line 20 waits for a short time (see Chapter 11 for explanation). Line 30 PRINTS "THE ANSWER IS" and completes the calculation 3+2. Note that when 3+2 is written within speech marks on line 10 no calculation takes place but when 3+2 is outside the speech marks the computer works out the calculation. You will see that the answer in line 30 is printed one character position away from "THE ANSWER IS" (like lines 10, 20, 40, 50 and 70 in the previous program). Edit the sums to 2 minus 7 and RUN the program again, now you know what that space is reserved for!

SCREEN LAYOUT

If you want to PRINT something in an exact place on the screen you must give the computer the correct instructions as we have seen. The screen is split into 40 vertical columns from Ø to 39 and 25 horizontal lines from Ø to 24.





POSITIONAL PRINTING

PRINTing the letter A in any given position is done by using the CRSR keys within a program line. To get this effect we have to use the CRSR keys as follows:

Type 10 PRINT "

Now press the CRSR ↑ ↓ key 10 times and then press the CRSR ← → key 19 times. Now close the speech marks and type ;"A". Your program line should look like this:

```
10 PRINT "QQQQQQQQQQ]]]]]]]]]]]]]]]]]]]]"
;"A"
```

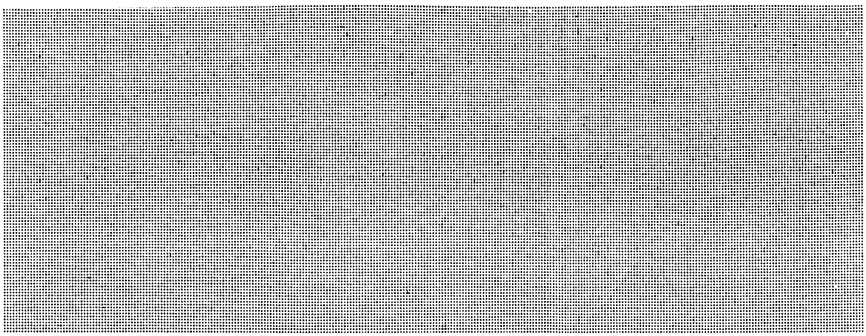
Now before typing RUN clear the screen — SHIFT and CLR HOME. When the screen is clear type RUN and press RETURN. Experiment with this feature.

THE TAB FUNCTION

There is another function which is useful when used with the PRINT command — this is TAB. Type and RUN the following line:

```
10 PRINT TAB(16)"HELLO"
```

It will PRINT "HELLO" 17 columns in from the left. Therefore TAB followed by a number from 1-39 will print whatever is asked starting in the column equal to that number from the left hand side of the screen.



(a) Type and RUN this line (use the ↑ CRSR ↓ key as explained above.

```
10 PRINT "QQQQQQQQQQ"TAB(16) 35
```

Clear the screen and then type RUN.

(b) Clear the screen and type:

```
10 PRINT "QQQQQQQQQQ "TAB(16)"35"
```

Clear the screen and then type RUN.

(c) Clear the screen and type:

```
10 PRINT "QQQQQQQQQQ"TAB(16)7*5
```

Clear the screen and then type RUN.

(d) Clear the screen and type:

```
10 PRINT"QQQQQQQQQQ"TAB(16)"7*5"
```

Clear the screen and type RUN.

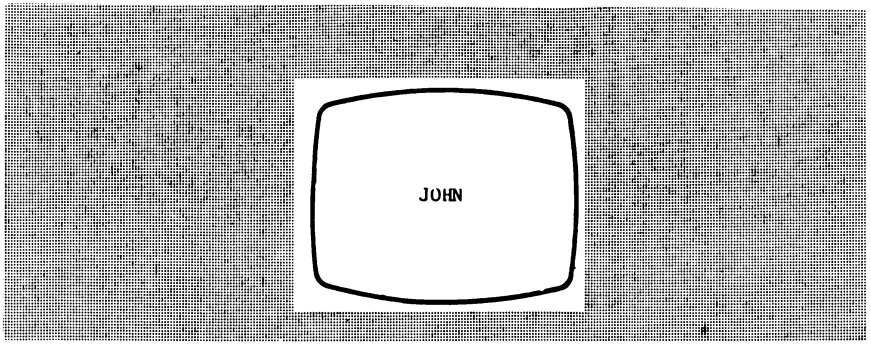
Notice that there is no difference between the program at (a) and (b) even though we used speech marks in the second example.

However look what happened when in example (c) we asked it to work out 7*5. When we left the speech marks off, the calculation was completed and the answer, 35, given. However when the calculation was placed inside speech marks, the computer printed exacly what was inside them, i.e. "7*5".

(e) Clear the screen and type:

```
10 PRINT"QQQQQQQQQQ"TAB(16)
"JOHN"
```

Clear the screen and RUN it.



(f) Clear the screen and type:

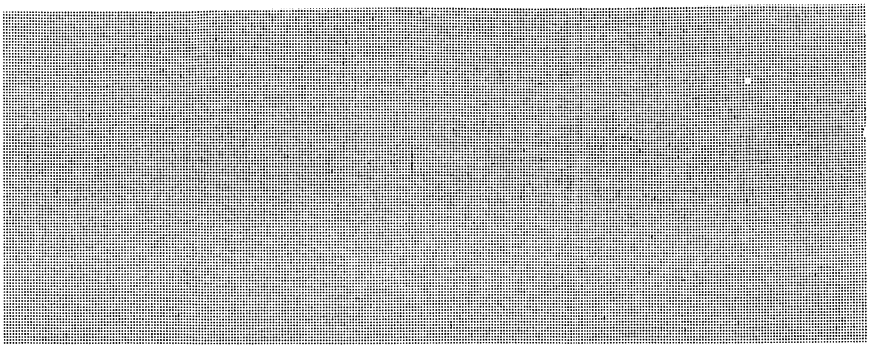
```
10 PRINT "QQQQQQQQQQ"TAB (16)JOHN
```

Note what happens when you RUN these lines. Program (e) PRINTs JOHN in the middle of the screen but program (f) simply prints RUN and READY followed by the flashing cursor. This is because there appeared to be nothing to print because there were no speech marks around JOHN. WHENEVER YOU USE LETTERS OR A COMBINATION OF LETTERS WITH A PRINT STATEMENT YOU WILL USUALLY HAVE TO USE SPEECH MARKS.

Note also that more than one TAB statement can be used in one line. Type:

```
10 PRINT TAB(10)"HI"TAB(20)"THERE"
```

Experiment with all the features and functions explained in this chapter.



CHAPTER 6

How to use END

INTRODUCTION

Once you have written a program you will want it to finish properly. We have already suggested that you should begin each program with a REM statement so that you know what the program is about. You also know that the REM statement does not affect the running of the program because the computer ignores it.

THE END COMMAND

However, the END command can affect the running of a program. Sometimes, unless you use it, your program could go on for ever!

Consider this program:

```
1Ø REM THIS IS AN END TEST PROGRAM
2Ø PRINT "I WANT THIS PROGRAM TO END"
3Ø END
```

Type and RUN the program. This is what will appear on the screen.

```
I WANT THIS PROGRAM TO END.
```

Now delete line 30 from the program. Type 30, press RETURN. Now RUN the program again. The result is the same. A simple program such as this, therefore does not need an END statement.

```
1Ø REM This is also an "END" test program.
2Ø PRINT "I want this program to continue."
3Ø GOTO 2Ø
```

EXAMPLE

Next, consider the above program. Type and RUN it.

I want this program to continue.
I want this program to continue.

Fills the screen.

WHY END CAN HAVE NO EFFECT

To break into the above program on the screen, press the RUN/STOP key. Type in LIST and press RETURN.

Now add the following line 40 to your program.

```
4Ø END
```

RUN the program again. The computer fills the screen exactly as it did previously. The END statement on line 40 has no effect at all. This is because line 30 sends the program back to line 20 before it ever gets to line 40! Press RUN/STOP and type NEW—RETURN.

ANOTHER EXAMPLE

Next, consider this program. Type and RUN it:

```
1Ø REM THIS IS A PROGRAM WHICH USES
       "END" - PROPERLY
2Ø LET C = Ø
3Ø PRINT "I WANT THIS PROGRAM TO
   RUN TEN TIMES AND THEN END"
4Ø LET C = C + 1
5Ø IF C = 1Ø THEN END
6Ø GOTO 3Ø
```

Line 20 LET C = Ø simply means, let the count begin at Ø (see Chapter 8 for more detail).

Line 40 LET C = C + 1 means let the new count equal the old count plus one.

Therefore the count works as follows:
C = Ø
PRINT
C = Ø + 1
PRINT
C = 1 + 1
PRINT
C = 2 + 1 etc.

WHEN IT STOPS

As the count continues the program runs through to line 50. IF C = 10 THEN END.

If the count doesn't equal 10, this instruction is ignored and the program continues to line 60 and goes back to 30.

```
1Ø _____ THIS IS AN END TEST PROGRAM.
2Ø PRINT "I LIKE SPORT"
3Ø _____ "I LIKE MUSIC"
4Ø _____
```

When the count = 10 the computer takes notice of the instructions on line 50 and stops.

EXERCISES

(1) Fill in the blanks in the above program and then RUN it. Try not to look back unless you have to.

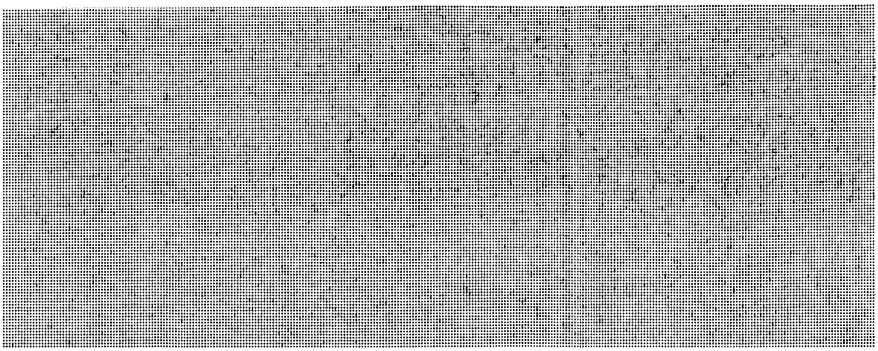
(2) Fill in the blanks in this program and then run it.

```
1Ø _____ SECOND END TEST PROGRAM
2Ø _____ "MY PROGRAM RUNS FOREVER!"
3Ø PRINT "OR, UNTIL I PRESS RUN/STOP KEY"
4Ø _____ 2Ø
```

(3) Write a program which prints your name and address 5 times and then stops. If you have a printer, you can print them out (see Chapter 14).

SECTION 2

BASIC PROGRAMMING TECHNIQUES



CHAPTER 7

How to use INPUT

GETTING INFORMATION

Let us now look at the keyword INPUT. This command word is of great use as it enables you to INPUT or feed information into a program which is already running.

USING INPUT

Type this program into your computer:

```
1Ø REM THIS PROGRAM DISPLAYS NUMBERS
       ON THE SCREEN
2Ø INPUT "WHAT IS YOUR NUMBER";N
3Ø PRINT N
4Ø GOTO 2Ø
```

Type RUN and press RETURN. You will see that the computer is asking you to give it a number. After the words "What is your number" you will see the flashing cursor. The computer is now waiting for you to INPUT any number of your choice — remember, it expects a number i.e. 1, 2 etc. not one, two. Press 1 and you will see the number 1 appear after the question mark. Now press the RETURN key — the number 1 has been printed on the next line.

The computer has followed the instructions in the program, i.e:

LINE 20 It has asked you to INPUT a number.
LINE 30 It has PRINTed that number
LINE 40 It has been sent back to line 20 — GOTO line 20 (GOTO is explained in more detail in Chapter 10).

WHEN YOU ENTER THE WRONG THING

This program will continue for as long as the instructions are followed. Now try to input a letter instead of a number. The computer does not understand your answer as it was expecting a number.

?REDO FROM START

In other words — try again!

LET'S EXPLORE DEEPER

The simple program that we have just examined is not very useful in itself. It can, however, be adapted in many ways. Let us go a little deeper. To break into this program press the RUN STOP key and whilst holding it down press RESTORE.

Consider this:

```
 1Ø REM MENTAL ARITHMETIC QUIZ
 2Ø INPUT "WHAT IS YOUR FIRST NUMBER";A
 3Ø INPUT "WHAT IS YOUR SECOND NUMBER";B
 4Ø PRINT "▲";A
 5Ø PRINT "+";B
 6Ø PRINT "____"
 7Ø FOR T = 1 TO 5ØØØ : NEXT
 8Ø PRINT "▲";A+B
 9Ø PRINT
1ØØ GOTO 2Ø
```

Type it in followed by RUN. Remember the techniques demonstrated in the previous program. They also apply to this quiz.

The computer asks you for your first number — INPUT any number you like, but remember line 10 of the program! You don't want the computer to beat you yet! After INPUTing your first number you are asked for your second number. After you have pressed RETURN the computer will print the sum on the screen; you have approximately 5 seconds to answer before the computer answers for you. Did you get it right?

Let us study the program:
LINE 20 — Asks you to INPUT a number

LINE 30 — Asks you to INPUT a second number

LINE 40 — PRINTS the first number; the spaces within the quotation marks position the number in the correct place on the screen.

LINE 50 — PRINTS the "+" addition sign (this can be altered to subtraction, multiplication or division) and the second number.

LINE 60 — PRINTS the line

LINE 70 — FOR T=1 TO 5000 NEXT acts as a stop watch — the time can be altered by making the number 5000 higher or lower as desired.

LINE 80 — PRINTS the answer as you would when completing the sum.

LINE 90 — PRINTS an empty line so that the next sum is set apart.

LINE 100 — GOTO 20 sends the computer back to line 20 to start again.

ANOTHER EXAMPLE

Let us now look at another simple arithmetical program, the basis of which has many possibilities.

```
 1Ø REM ARITHMETIC
 2Ø INPUT "WHAT IS THE FIRST NUMBER...";A
 3Ø INPUT "WHAT IS THE SECOND NUMBER...";B
 4Ø PRINT
 5Ø PRINT A;"+";B;"="
 6Ø PRINT
 7Ø INPUT "MY ANSWER IS...";C
 8Ø PRINT
 9Ø PRINT A;"+";B;"=";C
1ØØ PRINT "COMPUTER ANSWER...";A+B
11Ø GOTO 2Ø
```

Type, and RUN it. After INPUTing your two numbers as before the sum is displayed in a slightly different form and you are now asked to INPUT your answer. Once you have answered the question the computer then displays the correct answer.

Program explanation.

LINE 20 — requests a number
LINE 30 — requests second number
LINE 40 — empty print line
LINE 50 — PRINTS the problem
LINE 60 — empty print line
LINE 70 — requests the answer
LINE 80 — empty print line
LINE 90 — PRINTS the problem again and your answer
LINE 100 — PRINTS the correct answer
LINE 110 — sends the computer back to line 20 (GOTO)

YOU CAN USE THESE EXAMPLES

The previous two programs are adaptable and can be incorporated in longer and more sophisticated educational programs. It is our aim to help and encourage you to develop your own.

USING INPUT FOR LETTERS & WORDS

We have seen how the computer accepts numbers. You will, of course, wish to INPUT letters, or more importantly, words. BREAK into the program by using the RUN STOP and RESTORE keys. You will remember the first simple program in this chapter which began:

10 REM THIS PROGRAM DISPLAYS NUMBERS ON THE SCREEN
(Do not type this in.)

Well, we can change it slightly to meet our needs.

Type this into your computer.

```
1Ø REM THIS PROGRAM DISPLAYS WORDS.
2Ø INPUT "WHAT WORD DO YOU WANT PRINTED...";A$
3Ø PRINT A$
4Ø GOTO 2Ø
```

Type RUN and press RETURN.

The words "WHAT WORD DO YOU WANT PRINTED" followed by a question mark and the flashing cursor will appear.

Type in your name and press RETURN. Now type in your full address, including the post code. You will see that the computer will accept numbers as well as letters. In fact, it will print whatever is INPUT with the exception of commas and quotation marks. Break into this program as previously explained by using the RUN STOP and RESTORE keys.

A HANDY PROGRAM FOR CHRISTMAS

Here is a program which, with the aid of a printer, will greatly assist with the chore of writing Christmas thank you letters: (You can, of course, display this on the screen, even without a printer.)

```
 1Ø REM   THANK YOU LETTERS
 2Ø INPUT "DEAR"; A$
 3Ø INPUT "THANK YOU FOR THE "; B$
 4Ø INPUT "IT IS VERY"; C$
 5Ø INPUT "BEST WISHES"; D$
 6Ø PRINT "SHIFT CLR HOME"
 7Ø PRINT "DEAR ▲"; A$
 8Ø PRINT
 9Ø PRINT "THANK YOU FOR THE ▲"; B$
1ØØ PRINT "IT IS VERY ▲";C$
11Ø PRINT " WISHING YOU A VERY HAPPY NEW YEAR"
12Ø PRINT
13Ø PRINT TAB (6) "BEST WISHES, ▲"; D$
```

Type and RUN it. NOTE that in line 60 you should press the keys inside the quotes and not type the words (similar to what you did with the cursor control keys to position text on the screen).

THE CHRISTMAS PROGRAM EXPLAINED

You will see the word DEAR appear on the screen followed by ? and the flashing cursor. INPUT the name of the person to whom you are writing — don't forget to

press RETURN. The first line of your letter will appear for you to complete — then the second line and finally your name.

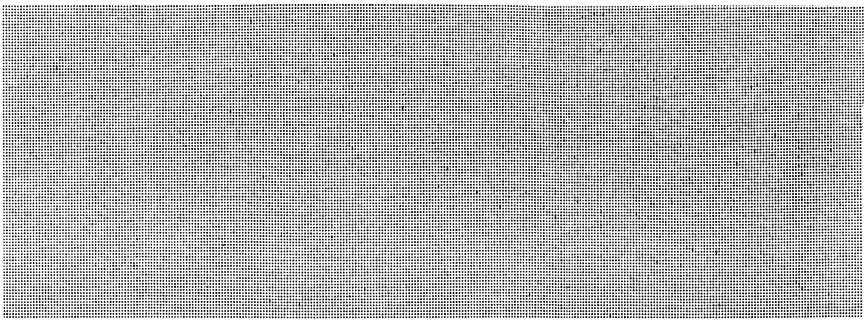
The program works as follows:

LINE 20 — Asks you to INPUT the name
LINE 30 — Asks for the description of the present
LINE 40 — Asks your opinion of the present
LINE 50 — Asks you to sign your name
LINE 60 — Clears the screen
LINE 70 — PRINTS the name
LINE 80 — Leaves an empty line
LINE 90 — PRINTS the thank you line
LINE 100 — PRINTS your opinion
LINE 110 — PRINTS the greeting
LINE 120 — Leaves an empty line
LINE 130 — PRINTS your signature

You can if you wish add:

```
140 GOTO 20
```

to start the program again or you can run it again by typing RUN and pressing RETURN. There is further information on "string variables" in Chapter 9, e.g. A$, B$, C$, D$.



CHAPTER 8

Numbers and your computer

INTRODUCTION

USING YOUR COMPUTER AS A CALCULATOR

Your computer can work as a calculator. Moreover, it will display calculations on the screen, show you if you have the correct answer and permit you to remedy your error.

The computer is quite capable of handling complex mathematical calculations but this chapter concentrates on handling some simpler functions.

ADDITION

+

Type this:
10 PRINT 3 + 4
RUN it and you will see 7 on the screen.

Type this:
10 PRINT "3 + 4"
RUN it and you will see 3 + 4 on the screen.

Type this:
10 PRINT "3 + 4 =▲"; 3 + 4
RUN it and you will see 3 + 4 = 7

(a) Therefore, if you simply want to see the result of adding two numbers, follow the first method.
(b) If you put the calculation within speech marks, it will print whatever is inside the speech marks and it will not complete the calculation.
(c) If you wish to print the calculation and its result then follow the last example!

```
1Ø PRINT 12-6
2Ø PRINT "12-6"
3Ø PRINT "12-6=▲"; 12-6
```

SUBTRACTION

Type and RUN the above program.

Subtraction follows the same rules as addition.

Type NEW followed by RETURN.

EXPLANATION

Type this:
10 PRINT 12−6 + 3+2
RUN this and you will get the answer 11. This is because the computer follows the signs − + as they appear. You will see the importance of this in the paragraph 'Order of Calculations'.

Type this:
10 PRINT 8 + 7 + 6 −3
RUN this and you will get the answer 18 for the same reasons as given above.

MULTIPLICATION

(Please note — don't use x.) The rules are similar to the paragraphs 'Addition' and 'Subtraction'.

Type this:

```
1Ø PRINT  4 * 6
2Ø PRINT "4 * 6"
3Ø PRINT "4 * 6 =▲"; 4 * 6
```

Now RUN it.

```
1Ø PRINT 10/2
2Ø PRINT "10/2"
3Ø PRINT "10/2 = "; 10/2
```

DIVISION /

Note that this is not the usual sign for division.

Type the above program.

RUN it.

Type NEW and RETURN.

ORDER OF CALCULATIONS

Where a calculation involves mixing signs, the computer deals with them in a definite order. That is, it deals with * and / before + and − NO MATTER WHAT ORDER THEY COME IN.

Consider the following:
10 PRINT 3 + 2 + 4 * 5
If you saw this calculation you might think the answer would be 45. The computer disagrees! RUN it. It has given the answer 25, because it works out 4 * 5, then adds 3 to it, then 2, to give 25.

If you want the computer to calculate in the order you have written it down you have to use brackets, as follows:
10 PRINT (3 + 2 + 4) * 5
The computer gives priority (tackles first) to the calculation in brackets, then it tackles whatever is outside brackets.

```
1Ø REM THIS PROGRAM DISPLAYS NUMBERS IN SEQUENCE
2Ø LET N = 1
3Ø PRINT N
4Ø LET N = N + 1
5Ø PRINT N
6Ø GOTO 4Ø
```

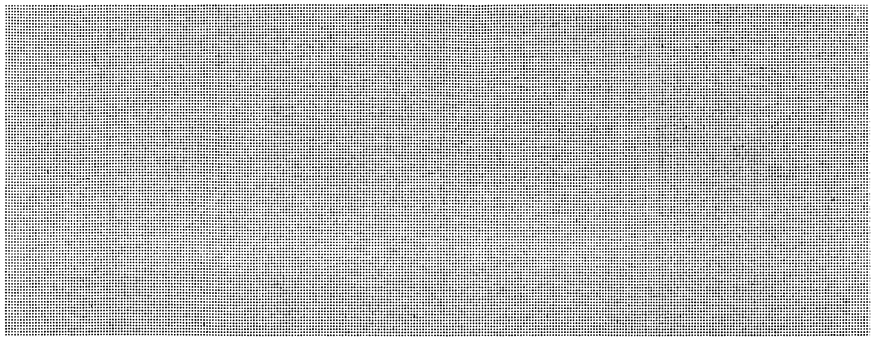
A PROGRAM TO LIST NUMBERS

Next look at the above program.
Type, and RUN it. You will see that the computer is printing a list of numbers in sequence down the left hand side of the screen — and it will continue to do so until it runs out of memory. Find the key CTRL and hold it down. This has the effect of slowing the computer — release the key and off it goes again! To BREAK into the program find the RUN/STOP key and press it. Now type CONT and then press RETURN — off it goes again! If you use the RUN/STOP key to BREAK into a program you can CONTinue from the place you stopped without having to go back to the beginning of the program by using RUN. BREAK into the program with the RUN/STOP key and read on:

EXPLANATION

LINE 10 — Remember the computer ignores this line because it begins with REM
LINE 20 — Means — let the first number = 1
LINE 30 — Prints N — that is 1
LINE 40 — Adds 1 to N to make it 2
LINE 50 — Prints the new N — that is, 2
LINE 60 — Sends the program back to line 40 which establishes a new N which is one more than the old N — this time it will be 3.

The computer continues doing this and printing a list of numbers.



N IS A "NUMERIC VARIABLE"

The N in the program is called a numeric variable — because the value of N varies — first it is 1, then 2, then 3 etc.

NAMING & GIVING A VALUE TO A NUMERIC VARIABLE

When we write LET N = 1, we talk about establishing a variable — saying what its value is.

Also, you will have noticed in the program that we have called the numeric variable 'N'. We do not have to. We can call a numeric variable by any letter of the alphabet from A — Z.

Note that a numeric (or any other) variable must always begin with a letter.

WHAT IS ALLOWED AS A NAME

You do not have to call your numeric variable by a single letter. You can, if you wish, use a mixture of letters and numbers — for example:

```
10 LET AB = Ø
20 PRINT AB
```

```
10 LET APPLES = 1
20 PRINT APPLES
```

```
10 LET R2D2 = 3
20 PRINT R2D2
```

```
10 PRINT 1
20 PRINT 2
    ↓
990 PRINT 999
```

Type and RUN them.

(a) However — you could not use words that the computer recognises as Command words e.g. RUN, LIST or even ORANGE because OR is a pre-set function. This is explained in more detail in the next chapter.

(b) 3D (it doesn't BEGIN with a letter)

(c) Hello! (! isn't a number or a letter)

MORE USES OF NUMERIC VARIABLES

You have seen one use of a numeric variable in the paragraph 'A Program to List Numbers'. It saves space in a program and in that particular program saves a long list of PRINT statements, as in the above example.

There are many other uses.

Consider this program:

```
10 REM CONTROLLING NUMBERS
20 FOR N = 1 TO 15
30 PRINT N
40 NEXT N
```

Type and RUN it:

EXPLANATION

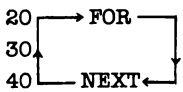
Line 20 sets the lower and upper limits of the list of numbers, that is, the lowest number will be 1 and the highest will be 15.

LINE 30 — prints the first number, that is 1.
LINE 40 — sets the next number on line 20, that is 2.
LINE 30 — prints 2.
LINE 40 — sets the next number on line 20, that is 3.
LINE 30 — prints 3, etc.

TWO NEW WORDS — FOR & NEXT

You will have noticed two new words FOR and NEXT. They are used together when you want to count in a program. The program LOOPS round from line 40 back to line 20. When it works through to line 40 it goes back to line 20. In the previous program it will loop 15 times because the loop is set to count 15 times!

To summarise, the numeric variable is n, and n varies between 1 and 15.



```
10 REM PROGRAM SHOWING USE OF FOR/NEXT LOOP
20 FOR N = 1 to 10
30 PRINT "THIS IS A FOR/NEXT LOOP"
40 PRINT 2 * 6
50 NEXT N
60 PRINT "THIS IS THE END OF THE PROGRAM"
```

TO REPEAT A GROUP OF INSTRUCTIONS

Of course, there can be more than one line between line 20 and 40.

Therefore, if you want to do something a certain number of times you simply include the instructions as part of a FOR — NEXT loop.
Consider the above program.

Type and RUN it.

EXPLANATION

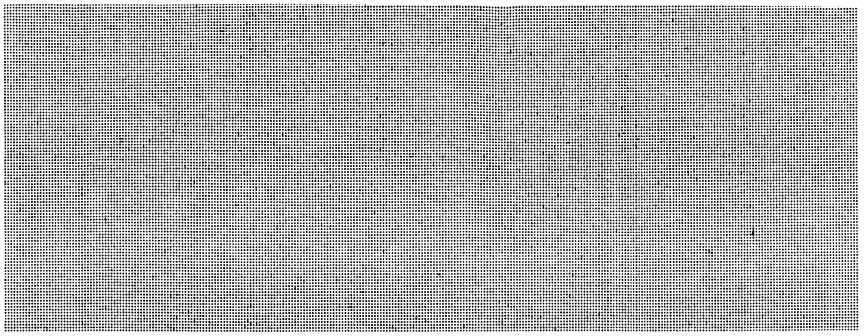
Lines 20 to 50 are the FOR/NEXT loop. It runs through 10 times, hence 1 to 10.
Lines 30 and 40 simply show two ways of using PRINT.

Note that when you want it to complete a calculation as in line 40, you don't use speech marks.

Line 60 — When the loop has run through ten times, it will continue through to line 60.

Remember that when you use a FOR/NEXT loop in this way, it doesn't actually print the numbers 1, 2, 3 etc.

What would you do to the program if you wanted it to actually number your statements? That's right, alter line 30 so that it reads:
30 PRINT N; "▲THIS IS A FOR/NEXT LOOP"



Notice that we have left a space ▲ at the start of the sentence inside the quotation marks. Try the program without leaving a space.

THE COMPUTER AS A CALCULATOR

Remember that we showed you how to use the computer as a calculator. Instead of typing in each calculation, the program can be written as follows:

```
1Ø REM USING NUMERIC VARIABLES
2Ø LET N=1
3Ø LET D=1
35 FOR G = 1 TO 1Ø
4Ø PRINT "WHAT IS▲";N"+";D
5Ø LET N=N+1
6Ø LET D=D+1
7Ø NEXT G
```

LINE 20 — means let the first number 'N' = 1
LINE 30 — means let the second number 'D' = 1
LINE 35 — means that there will be 10 loops between line 35 and line 70
LINE 40 — prints the calculation
LINE 50 — adds 1 to 'N'
LINE 60 — adds 1 to 'D'
LINE 70 — sends the loop back to 35

```
1Ø REM ANOTHER VARIATION
2Ø LET N = 1
3Ø LET D = 1Ø
35 FOR G = 1 TO 1Ø
4Ø PRINT "WHAT IS ▲";N;"*";D
5Ø LET N = N + 1
6Ø NEXT G
```

A PROGRAM FOR THE 10 TIMES TABLE

The above program asks the 10 times table.

Compare it with the previous program. Line 30 sets D to 10 because it is the 10 times table. Line 35 reads FOR G = 1 TO 10 because we want to go round the loop 10 times.

Line 40 has been altered to give * instead of +.

Line 60 has been missed out completely because we only want N to increase by 1 each time round. We do not want D to increase at all because in the ten times table there must always be a 10!

We can very quickly alter this program to give the answers too. Simply insert at line 45 PRINT N*D.

Type, and RUN both variations.

RANDOM NUMBERS & ROUNDING

There is a function, RND (type the letters R,N,D) which instructs the computer to choose numbers at random. This can be very useful in many types of programs especially those dealing with mathematics.

Type and RUN this program:

```
10 PRINT RND (0)
20 GOTO 10
```

You will notice that all numbers are between Ø and 1.

We now introduce a further function, INT(type the letters I,N,T). Using the INST DEL key insert INT into the above program at line 10:

```
10 PRINT INT (RND (0) + 1)
20 GOTO 10
```

Then RUN it.

You will now see a list of 1's.

This is what has happened. You have added 1 to all the random numbers i.e. (RND+1). The function INT (INTEGERISE) has then rounded all the random numbers produced by the computer DOWN to the nearest whole number, which is 1. e.g. 1.5 becomes 1, 1.8 becomes 1, 1.9 becomes 1 and so on.

Try this variation:

```
10 PRINT INT (RND (0) + 2)
20 GOTO 10
```

RUN it. The same principle applies. You will now have a list of 2s. Remember INT on its own always rounds down.

```
1Ø REM RANDOM NUMBERS 1 TO 1Ø
2Ø LET N = INT (RND (Ø) * 1Ø) + 1
3Ø PRINT N
4Ø GOTO 20
```

A LIST OF RANDOM NUMBERS USING RND AND INT

On its own, this would be of very little use, but the RND function together with INT enables you to print a list of random numbers between, say, 1 and 10. This is how we write it.

Type and RUN the above program.

A LIST OF CALCULATIONS IS EASILY PRINTED

Therefore you can see how it is quite easy to print a list of calculations adding numbers between 1 and 10 at random.

```
 5 REM Program to show use of RND
1Ø LET N = INT (RND (Ø) * 1Ø) + 1
2Ø LET D = INT (RND (Ø) * 1Ø) + 1
3Ø PRINT "WHAT IS▲";N;"+";D
4Ø PRINT "ANSWER IS▲";N + D
45 PRINT
5Ø GOTO 1Ø
```

Type and RUN the program. Use the CTRL key.

SOME USEFUL FORMULAE

Below are some useful formulae if you want to print certain lists of random numbers.

Numbers between 1 and 15 inclusive.

```
1Ø LET N = INT (RND (Ø) * 15) + 1
2Ø PRINT N
3Ø GOTO 1Ø
```

Numbers between 1 and 100 inclusive.

```
1Ø LET N = INT (RND (Ø) * 1ØØ) + 1
2Ø PRINT N
3Ø GOTO 1Ø
```

Numbers between 10 and 100 inclusive.

```
1Ø LET n = INT (RND (Ø) * 90) + 1Ø
2Ø PRINT N
3Ø GOTO 1Ø
```

Numbers between 100 and 1000 inclusive.

```
1Ø LET n = INT (RND (Ø) * 9ØØ) + 1ØØ
2Ø PRINT N
3Ø GOTO 1Ø
```

VARYING THE LEVEL OF DIFFICULTY

Once you are able to choose which numbers to print, you are then in a position to vary the level of difficulty of your calculations. For instance, if you are just beginning to learn multiplication, then the numbers you choose might be between 1 and 10. If you are more advanced then you might want long multiplication, say numbers between 99 and 1000.

DIFFERENT WAYS TO USE THE COMPUTER

Also remember that you can use the computer in different ways.

(i) to list questions — which you might answer yourself on paper

(ii) to ask a question and, using the Timing statement to give you the answer after you have again worked it out on paper (Chapter 11 — Timing).

(iii) to INPUT answers into the computer and then be told if you are right or wrong, perhaps after giving a specified number of attempts, say 3.

(iv) if you have a printer, you can print worksheets

(v) you can also set out your calculations attractively by using the simple techniques you have already learned. The following program is for long multiplication.

```
1Ø PRINT "LONG MULTIPLICATION";
2Ø PRINT TAB (26); "ANSWER"
25 PRINT "_____________________";
26 PRINT TAB (26); "_______"
27 FOR D = 1 TO 12
3Ø LET A = INT (RND (1ØØ) * 899)
4Ø LET B = INT (RND (1Ø) * 89)
41 LET C = A * B
5Ø PRINT " ▲▲▲▲▲";A
6Ø PRINT " ▲▲▲▲*";B
8Ø PRINT " ▲▲▲▲ __________";D
9Ø PRINT " ▲▲▲▲ __________ "
1ØØ PRINT " ▲▲▲▲ __________ "
1Ø5 FOR T = 1 TO 10000 : NEXT
11Ø PRINT " ▲▲▲▲ __________ "; TAB (26) ;C
115 PRINT
12Ø NEXT D
```

This is what the screen displays:

Long Multiplication **Answer**

```
  980
  *53
  ____________  1
  ____________
  ____________
  ____________          51940
  579
  *11
  ____________  2
  ____________
  ____________
  ____________          6369
```

etc. until twelve calculations have been printed. Note that you are given almost a minute and a half before you are told the answer. This gives you plenty of time to work out the answer on paper. Obviously if you are faster, you can alter line 105 accordingly.

WHEN THE COMPUTER USES DECIMAL

At this stage, if you are attempting division calculations, you cannot guarantee that the answer will be in whole numbers. The computer always uses decimals if you have a calculation like this:
10 PRINT 5/2
Type and RUN it.

RAISING A NUMBER TO A POWER

Next, in this chapter about numbers, a brief word about the sign ↑. This is the computer's way of showing that a number has been raised to a certain 'power'. For example if you type in:
10 PRINT 3 ↑ 2
and RUN it, you will get the answer 9 because 3 ↑ 2 means 3 × 3, or 3 squared.

Equally:
10 PRINT 2 ↑ 8 means 2 × 2 × 2 × 2 × 2 × 2 × 2 × 2
How many do you think this is? Try it!

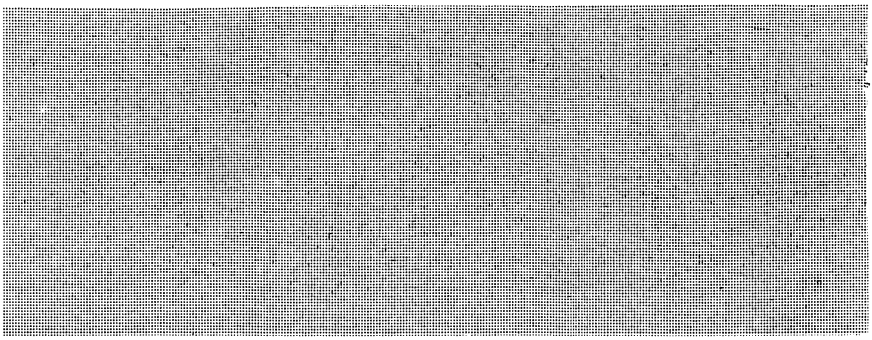
DEALING WITH FRACTIONS

Your computer will also deal with fractions but you must set it out as follows:
10 PRINT 2+1/2 + 3 + 1/2 is the only way you can ask it to add $2^{1/2} + 3^{1/2}$!

LET IS NOT COMPULSORY

Note that with your computer you do not HAVE to use LET. You can write 10 N=1.

Throughout this book we have continued to use LET. We feel this is clearer for beginners. Similarly we have not shortened any other commands.



CHAPTER 9

Strings and things!

INTRODUCTION

You have already read about numeric variables and now know about some of their uses. Defining string variables is rather more difficult and can perhaps best be understood by looking at some examples of how they work and some of their features.

HOW TO RECOGNISE A STRING VARIABLE

(a) You will always be able to recognise a string variable because it has the $ sign written after it. The $ sign is on the 4 key. To obtain $ hold down the SHIFT key and press 4.

(b) A string variable MUST begin with a letter and always end with the $ sign. The variable can be of any length — either a single letter e.g. A$, B$ or more than one letter e.g. AB$, ABC$.

(c) For the purposes of this book the value of the string variable (what it is equal to) must be typed within speech marks. A STRING is simply several letters or graphics or numbers. That is, the computer will print anything that is within speech marks, including numbers. It does not have to be of any particular length.

```
LET NL$    = "JOHN"
LET NL$    = "JOHN SMITH, 35, ACACIA AVE"
LET NLRP$ = "JOHN"
LET N8$    = "JOHN"
```

For example:
LET N$ = "JOHN" — This is called establishing a variable — you use the LET command as you did with numeric variables.
However you could also type the strings as seen above.

(d) Having established that a string variable can have any number of letters or numbers you must be careful in your choice because the computer only recognises the first two characters plus the $ sign of any variable name. For example type the following lines and RUN them:

```
1Ø LET TREE$ = "BRANCH"
2Ø LET TRAIN$ = "ENGINE"
3Ø PRINT TREE$; TRAIN $
```

As you see, the computer has displayed ENGINE twice because it treats line 20 as replacing the variable TR$ (set equal to "BRANCH" in line 10) with the new value ENGINE. Line 30 is then treated as an instruction to print TR$ twice.

(e) It is also important to know that a variable name cannot include letters that the computer will recognise as a command or function.

Type and RUN these two lines:

```
1Ø LET COST $ = "PRICE"
2Ø PRINT COST $
```

As you see this results in a SYNTAX ERROR because the computer recognises the first three letters of the variable as COS, one of its pre-set functions. Consult your Commodore manual for all commands, and function words.

AN EXAMPLE

Here is a program to show you a use of string variables on their own. (For simplification all string variables are single letters followed by the sign $.) Let us imagine that you wanted to display on the screen a number of items on a weekly shopping list. Your program might read as follows:

```
10 REM SHOPPING LIST
20 LET J$ = "JAM"
30 LET C$ = "COFFEE"
40 LET P$ = "POTATOES"
50 LET S$ = "SUGAR"
60 LET M$ = "MILK"
70 PRINT J$, C$, P$, S$, M$,
```

Type and RUN this program. You will simply get a list of five things you may wish to buy.

A USE FOR THE EXAMPLE

However, if you only want to buy two of the items this month, you would simply alter line 70 to read:
70 PRINT J$, S$

This particular program might not seem to have much value until you consider say the 20 or so items which might make up a household shopping list. If you SAVEd a program such as this on tape and, at the start of each week, scanned the program to see what you needed and then alter the last line, you would be able to print on the screen your requirements for that week. (See Chapter 15 for SAVE.) If you had a printer, it would then be a moment's task to print out the list! No more forgotten items at the supermarket!!

ADAPTING THE EXAMPLE

The program would need very little adaptation to print out the cost of your list, by using numeric variables as well as string variables. Look at this adapted program:

```
1Ø REM SHOPPING LIST
2Ø LET J$ = "JAM" : LET J = .4Ø
3Ø LET C$ = "COFFEE" : LET C = 1.2Ø
4Ø LET P$ = "POTATOES" : LET P = 2.ØØ
5Ø LET S$ = "SUGAR" : LET S = 1.6Ø
6Ø LET M$ = "MILK" : LET M = 5.ØØ
7Ø PRINT J$, C$, P$, S$, M$;
8Ø LET T = J + C + P + S + M
9Ø PRINT "▲"; T
```

In line 80, T stands for total.

Note that in lines 20-60, it is perfectly possible to have a STRING variable, say J$ and a NUMERIC variable, say J.

RUN this program. Now before RUNning it again, use the INST DEL key to get rid of J$, C$ on line 70 and J, C on line 80.

USING THE ADAPTATION

You will have seen that you get two different lists and two different totals. You could use this as a guide to see how much money you would need when you go shopping. Also, if you only had a limited amount of money, say £20 and a shopping list which came to £25, you could try different combinations to see which would be the best way to spend your money.

ANOTHER USE OF STRING VARIABLES

One of the uses of string variables is to save typing the same words time and time again.

This is particularly useful in a long program where you might want to use a particular string variable several times.

ANOTHER EXAMPLE — A QUIZ

Here is another example of string variables used in a quiz:

```
10 REM KINGS AND QUEENS
20 LET A$ = "WILLIAM 1"
30 PRINT "WHO WAS KING OF ENGLAND IN 1063"
40 INPUT "MY ANSWER IS ";B$
50 IF B$ = "WILLIAM 1" THEN PRINT "WELL DONE" :
                                    GOTO 70
60 PRINT "INCORRECT. THE CORRECT ANSWER IS ▲";A$
70 END
```

For explanations of INPUT see Chapter 7.

Type and RUN this. Type RUN to RUN again.

STRING VARIABLES DO VARY

One thing that often puzzles people is that string variables do not seem to vary! For example, in the program above, A$ = "WILLIAM I" — A$ however did not have to equal WILLIAM I — it is only equal to WILLIAM I because you gave it that value!

A TRUE STRING VARIABLE EXPLAINED

Now look at B$ — this, if you like, is a true string variable, because it is not equal to anything until you give it a value. You could give it the value ELIZABETH I or GEORGE III — its value could vary depending on your answer, what you INPUT.

WELL DONE!

Line 50 says that only if B$ equals WILLIAM I will you get the statement "WELL DONE".

USING THE QUIZ

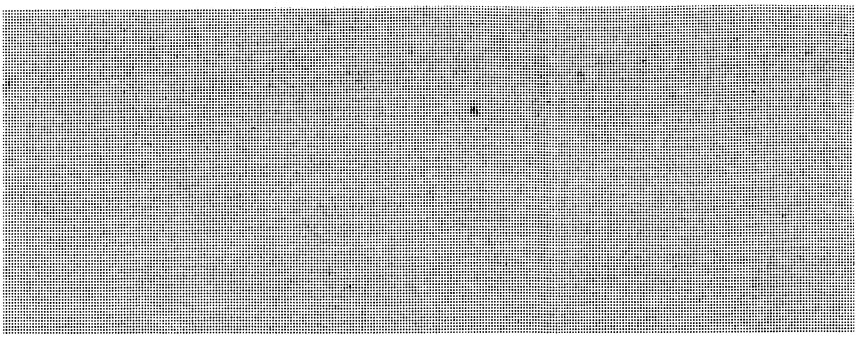
You could construct a series of questions from your exercise books on different topics and load them on tape to use for revision. It might be an idea to give yourself three tries at each question before you get the correct answer. If you want to do this, look at Chapter 6.

A PRIMITIVE WORD PROCESSOR

Let us now turn to look at a primitive 'word processor'. This program permits you to tell a 'story' using some of your own words for the characters and places. It is possible to get books printed with a child's name appearing in it as the hero all the way through. Perhaps this is how it is done.

```
1Ø REM THIS SHOWS THE USE OF STRING VARIABLES
      IN A STORY.
15 PRINT "I WENT TO SEE T$ AND THEN WE
   MET K$ BEFORE GOING TO THE Y$"
2Ø INPUT "MY WORD FOR T$ IS ";T$
3Ø INPUT "MY WORD FOR K$ IS ";K$
4Ø INPUT "MY WORD FOR Y$ IS ";Y$
45 PRINT
5Ø PRINT "I WENT TO SEE▲"; T$; "▲ AND THEN
   WE MET ▲"; K$; " ▲BEFORE GOING TO THE▲ ";
   Y$;
6Ø END
```

If you have young children you could use a program such as this to build vocabulary. The limits are only those of your imagination.



CHAPTER 10

Looping and branching

USING GOTO TO REPEAT

You can use GOTO in a program when you want to repeat something, e.g:

```
1Ø REM THIS IS AN EXAMPLE OF LOOPING
2Ø PRINT "THIS WILL HELP TO SHOW THE
   USE OF GOTO"
3Ø GOTO 2Ø
```

This program simply prints line 20 indefinitely because the computer prints line 20, arrives at line 30, obeys this line, goes back to 20 and so on. The flickering bottom line is proof that the computer is still printing line 20. Break into the program by pressing RUN STOP.

GOING ROUND IN A LOOP

This process is called LOOPING because the program goes round in a loop.

AN EXAMPLE OF OTHER USES OF GOTO

However, GOTO has other uses. Consider this program:

```
 1Ø REM FRENCH VOCAB. PROGRAM.
 2Ø PRINT "DO YOU WANT TO GIVE ANSWERS IN
    FRENCH (PRESS 1) OR IN ENGLISH (PRESS 2)?"
 3Ø INPUT "MY CHOICE IS ▲"; A
 4Ø IF A = 1 THEN GOTO 7Ø
 5Ø IF A = 2 THEN GOTO 11Ø
 6Ø GOTO 2Ø
 7Ø PRINT "WHAT IS FRENCH FOR THE TABLE"
 8Ø INPUT "MY ANSWER IS ▲ ";B$
 9Ø IF B$ = "LA TABLE" THEN GOTO 2ØØ
1ØØ GOTO 7Ø
11Ø PRINT "WHAT IS ENGLISH FOR LA TABLE"
12Ø INPUT "MY ANSWER IS ▲";C$
13Ø IF C$ = "THE TABLE" THEN GOTO 2ØØ
14Ø GOTO 11Ø
2ØØ PRINT "WELL DONE, CORRECT ANSWER"
21Ø END
```

OVER TO YOU

This is an unsatisfactory program in a number of ways, not least because it only tests one word! However, its main purpose is for you to see further uses of GOTO. There is a French vocab test in Chapter 18 which is of real use and can be SAVEd on tape.

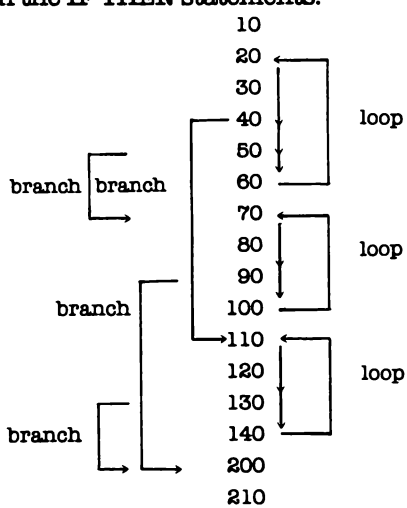
EXPLANATION

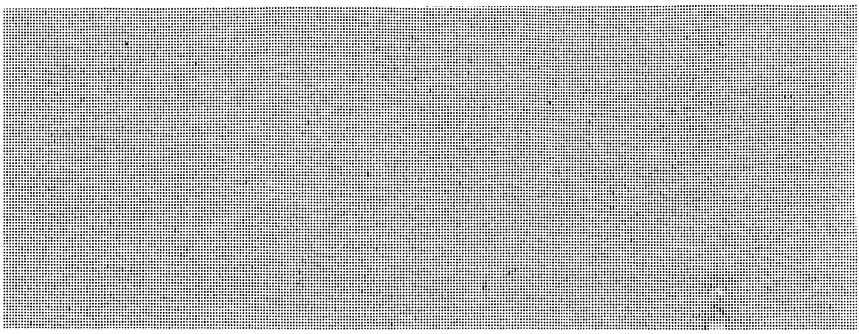
Program commentary:
LINE 20 — asks whether you want to answer in English or French
LINE 30 — asks you to press either key 1 or key 2 depending on the question required
LINE 40 — if you answer 1 the computer branches to line 70 for the English question and misses out 50 and 60.

LINE 50 — if you answer 2 the computer branches to line 70 for the French question (NOTE on lines 40 and 50, the construction IF something is so, THEN GOTO another line.) The IF THEN GOTO command is explained more fully in Chapter 16.
LINE 60 — is an interesting line. If you type in anything other than 1 or 2, line 60 will send you back to line 20. This will continue until you type in 1 or 2. This is an example of a loop.
LINE 90 — is another IF THEN GOTO line. IF your answer is correct THEN GOTO line 200 for some praise. This is a branch. This line enables you to miss out lines 100-140.
LINE 100 — is also an IF THEN line. IF you don't give the correct answer THEN line 70 and get the question again. This is a loop.
LINE 110 — begins the English section of the program.
LINES 130 and 140 — are also IF THEN lines, similar to 90 and 100. Line 130 branches, line 140 puts you into a loop.

HOW AN IF-THEN STATEMENT WORKS

This diagram shows you what is happening with the IF-THEN statements.





THINGS TO NOTICE

Lines 30, 40, 50 — all mention 'A', and also the numbers 1 and 2. 'A' is an example of a numeric variable. Simply, 'A' can either be the number 1 or 2. That is, it can vary — hence numeric variable! (See Chapter 8 for more explanation.)

STRING VARIABLES

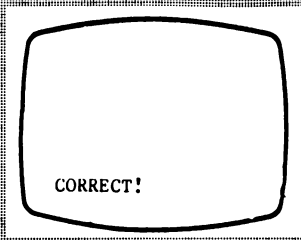
Lines 80 and 90 — both mention B$. When you see the $ sign together with a letter, you have come across a string variable — this has been covered in Chapter 9.

In this instance the string is going to be two words — hopefully la table! However, the answer you give may be LA TABLE, or something else. Therefore the answer can vary.

LINE 80 — invites your answer

LINE 90 — sends you to line 200 if your answer B$ is correct

LINE 100 — sends you back to line 70 if your answer is wrong, that is, anything other than "LA TABLE".

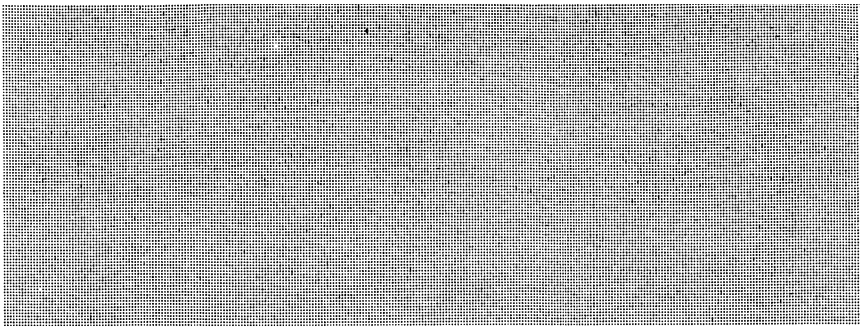


EXERCISES

(1) Write a program which asks the question "WHAT DATE IS CHRISTMAS DAY?" and expects the answer "DEC, 25TH". The program should tell you if the answer is correct and should continue to ask you if the answer is incorrect.

(2) Adapt the program given in the chapter to give you three attempts at the question. If you get the answers wrong three times then the program should STOP. Refer to Chapter 6, the paragraph 'Another Example'.

(3) If you feel you can do it, adapt the program so that it gives you the correct answer after three attempts.



CHAPTER 11

Screen clearing

INTRODUCTION

This chapter is about clearing the television screen for more effective displays.

TO RECAP

In a previous chapter it was explained that it is possible to clear the television screen by pressing CLR HOME whilst holding down SHIFT. It is also possible to clear the screen by switching off at the mains. This wipes out all the memory.

CONSIDER THIS PROGRAM

Type and RUN this program:

```
10 REM ADDITION
20 LET A = INT (RND (0) * 10) + 1
30 LET B = INT (RND (0) * 10) + 1
40 PRINT "WHAT IS...";A; "+";B
50 INPUT "MY ANSWER IS ";C
60 IF C = A + B THEN PRINT "CORRECT"
70 IF C <> A + B THEN PRINT "INCORRECT, THE
   ANSWER IS     ";A + B
80 GOTO 20
```

EXPLANATION

LINE 20 — chooses a random number between 1 and 10
LINE 30 — chooses a random number between 1 and 10
LINE 40 — asks you to add the two numbers
LINE 50 — asks for your answer
LINE 60 — will tell you if your answer is correct
LINE 70 — will tell you if your answer is incorrect and gives you the correct answer
LINE 80 — sends you back to line 20 to choose another random number.

THE UNTIDY RESULT

Before long you will fill the screen with questions and answers which can be rather irritating and is also a little untidy.

A NEATER APPROACH

Let us see if we can tidy up the program. Whilst holding down the RUN STOP key press RESTORE. Now type the following line:

35 PRINT "SHIFT CLR HOME"

To do this type PRINT and then the speech marks — now hold down the SHIFT key and then press the CLR HOME key — a heart will appear. Now close the speech marks.

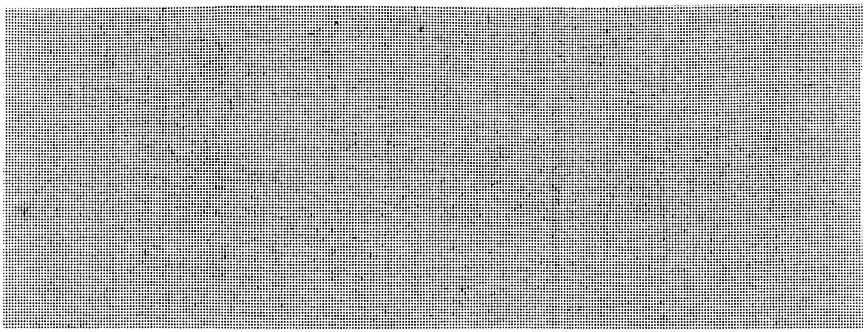
When you RUN the program you will see that after you have INPUT your answer, you are told whether it is correct or not but the screen then clears with remarkable speed and prints the next question!

A SLOWER, MORE READABLE RESULT

There is a way around this. Break into the program as explained previously and then type this line:
75 FOR T = 1 TO 5000: NEXT

When you RUN the program now you will see that line 75 gives you sufficient time to read the answer before it clears the screen.

If you like, line 75 acts as a clock. The statement stops the computer for a given length of time.
FOR T = 1 TO 5000 : NEXT = 7 secs approx.
FOR T = 1 TO 10000 : NEXT = 14 secs approx.



CHAPTER 12
Subroutines

INTRODUCTION

You have seen in Chapter 10 that the keyword GOTO instructs the computer to skip lines in a program. There is another keyword which has a similar function.

WHEN TO USE GOSUB

This keyword is GOSUB and is short for GO TO SUBROUTINE. It can be used when a number of identical lines are repeated at different stages within a program.

FOR EXAMPLE

Consider this program BUT DO NOT TYPE IT:

```
1Ø LET A = 5
2Ø LET B = 6
3Ø PRINT "WHAT IS ";A;"+";B
4Ø INPUT "MY ANSWER IS...";C
5Ø IF C = A + B THEN PRINT "RIGHT"
6Ø IF C <> A + B THEN PRINT "WRONG THE
   ANSWER IS ";A + B
7Ø LET A = 7
8Ø LET B = 4
```

```
 9Ø PRINT "WHAT IS ";A;"+";B
1ØØ INPUT "MY ANSWER IS...";C
11Ø IF C = A + B THEN PRINT "RIGHT"
12Ø IF C <> A + B THEN PRINT "WRONG, THE
    ANSWER IS "; A + B
13Ø STOP
```

EXPLANATION

LINES 10 and 20 — set up the variables
LINE 30 — asks the question
LINE 40 — you give the answer
LINE 50 — means that IF C, (see Chapter 16) your answer, is correct, the computer prints "RIGHT"
LINE 60 — means that IF your answer is wrong (<> means not equal to — see Chapter 16) the computer tells you so and prints the correct answer
LINES 70 to 120 — repeat the exercise and you can imagine the length of program necessary to ask 20 questions with answers!

AN IMPROVED EXAMPLE

This program is much better — type it in.

```
 5 REM SUBROUTINE
1Ø LET A = 5
2Ø LET B = 6
3Ø GOSUB 1ØØØ
4Ø LET A = 7
5Ø LET B = 4
6Ø GOSUB 1ØØØ
7Ø END
```

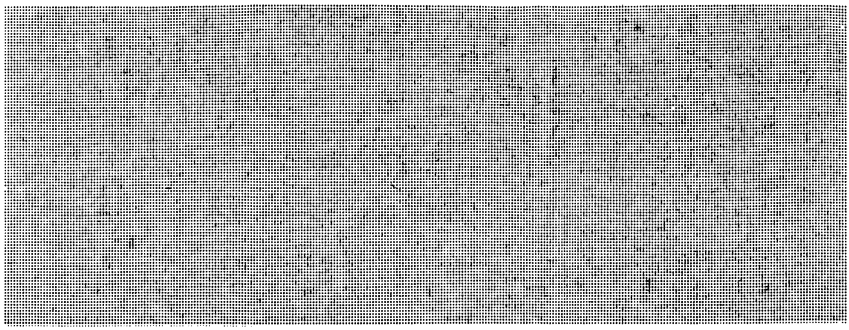
```
1000 PRINT "WHAT IS ";A;"+";B
1010 INPUT "MY ANSWER IS..";C
1020 IF C = A + B THEN PRINT "CORRECT"
1030 IF C <> A + B THEN PRINT "INCORRECT, THE
     ANSWER IS ";A + B
1040 RETURN
```

EXPLANATION

Let us examine the program as the computer obeys it.
LINES 10 and 20 — set up the variables
LINE 30 — sends the computer to the subroutine at line 1000
LINE 1000 — asks the question what is A+B?
LINE 1010 — you give your answer
LINE 1020 — IF your answer is correct, prints "CORRECT". The IF keyword is explained later in Chapter 16.
LINE 1030 — IF your answer is not correct, <> means does not equal (see Chapter 16) then the computer prints the correct answer
LINE 1040 — RETURN sends the computer back to the main program at the line IMMEDIATELY after the GOSUB line 30
LINES 40 and 50 — set up new variables
LINE 60 — sends the computer back to the subroutine
LINE 1040 — Returns computer to the line after GOSUB Line 60
LINE 70 — ENDs the program. If this line were not included the computer would continue to line 1010 where it would crash!

CONCLUSION

You can see from this program that it would now be quite easy to add further lines, like 40, 50 and 60, to instruct the computer to ask as many questions as you like.



CHAPTER 13
Summary

Before you go any further you should satisfy yourself that you have understood what you have read so far. The following is a useful checklist.

(1) Do you know the uses of the following?
RUN, CLR HOME, INST DEL, END, NEW, RUN STOP, LIST, CTRL, REM, PRINT, TAB, SHIFT, SHIFT LOCK, INPUT, LET, GOTO, RESTORE, GOSUB and RETURN.

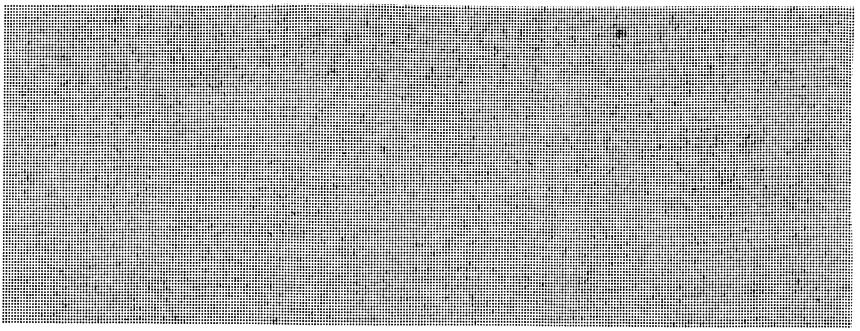
(2) Can you move the cursor around the screen?

(3) Do you know how the computer uses " " , ; ?

(4) Can you give examples of numeric variables, string variables? Can you write simple programs involving them?

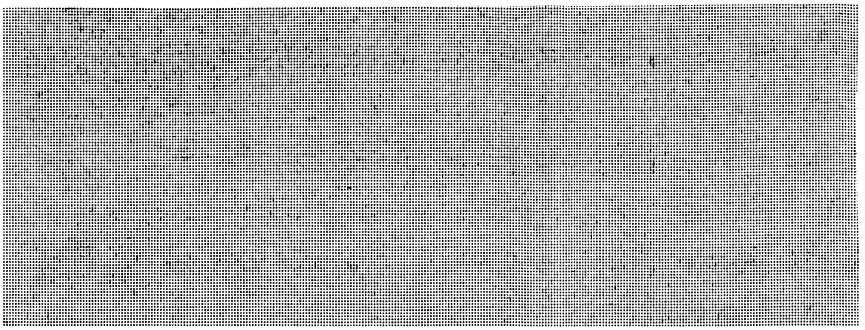
(5) Can you recognise looping, branching and subroutines? Can you write simple programs involving each?

If you feel unhappy about any of the above you should possibly go back and refresh your memory. If you feel confident, then go straight ahead and look at Chapters 16, 17 and 18 which contain rather more difficult ideas.



SECTION 3

USEFUL AIDS



CHAPTER 14

Using the PRINTER

THE PRINTER IS USEFUL

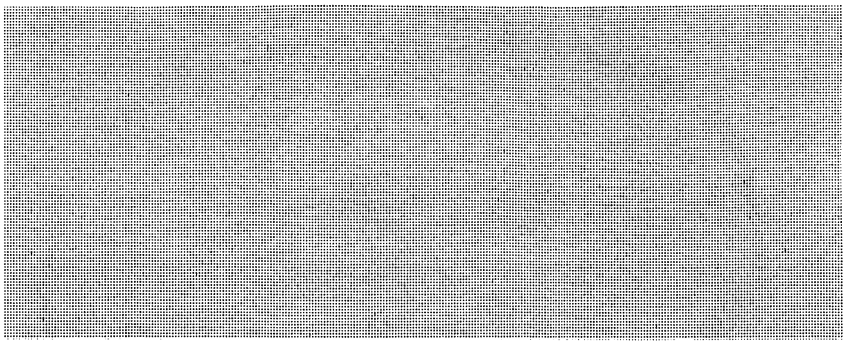
If you have a printer it should be connected to your computer whilst it is switched off. Plugging or unplugging can cause damage to the computer if it is switched on.

WARNING!

If you do not have a printer you will have to consider its value to you. It's certainly possible to save programs on tape, but it's often useful to have a "hard copy" available too. Loading a cassette takes time. Hard copies are immediately available. You don't risk losing them because of damage to your tape or accidental erasure.

You can then keep a file of your best programs. Browsing through these will often give you good ideas for other programs too and suggest improvements for your existing programs.

Further, if you hit problems with a program, a print-out of where you have got to will make it easier for you to discuss with other computer fanatics the problems you have faced in the hope of finding a solution. Exchanging programs by swapping print-outs is also a good deal cheaper than swapping by cassette.



COMMANDS FOR THE PRINTER

Several of the programs in this book are test programs. They need very little adaptation to be able to print, say, a score out of 20 in a vocabulary test together with a complete print out of the words in the test, together with the correct answers. This hard copy, in its turn, provides a basis for further future revision.

PRINTING A PROGRAM LISTING

You can also print out maths worksheets almost for ever with some of the maths programs which have random numbers. Obviously you can carry out a similar exercise with quiz programs.

A PRINT STATEMENT FOR THE PRINTER

This has been just a short insight into the possible uses of a printer. I hope it has been of use in assessing its value.

CHAPTER 15

Using the tape recorder

GET THE RIGHT TAPE RECORDER

A tape recorder is not quite indispensable, but it considerably extends the use of your computer. Without a tape you will not be able to use any of the "games tapes" or "educational tapes" available in you local computer store!

HOW TO USE PRE-RECORDED PROGRAMS

The introductory manual which accompanies your computer is fairly clear about how to use the recorder, so a few hints from us, we feel, will suffice.

HOW TO SAVE A PROGRAM ON TAPE

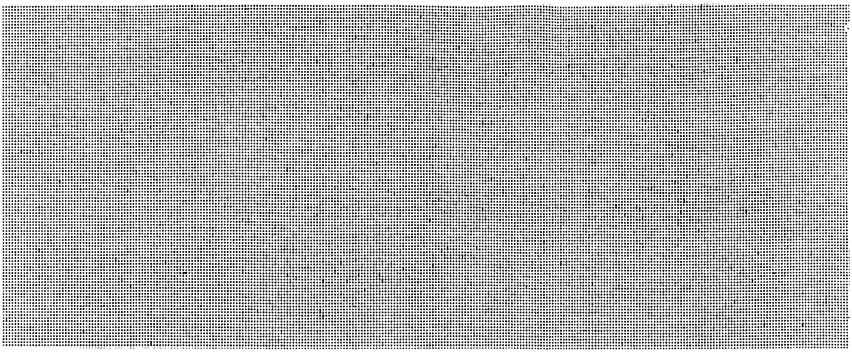
It is advisable when saving your own programs to bear in mind:

(a) Make sure that your program RUNs before SAVEing it.

(b) Use the VERIFY command to confirm that you have SAVEd the program listing as it appears on the screen.

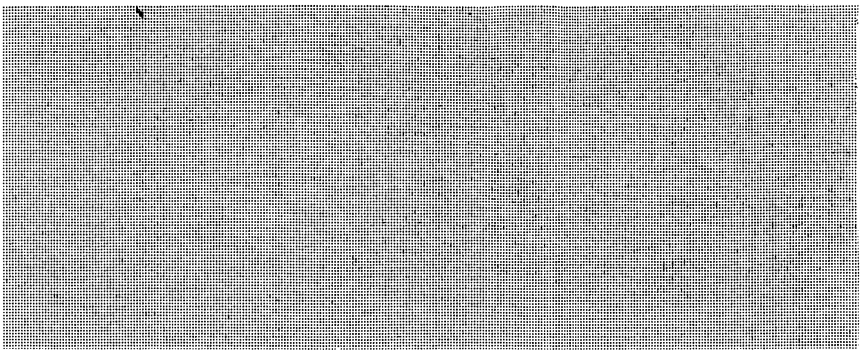
(c) SAVE each program twice in case anything goes wrong with one of the programs.

(d) Keep your tapes well away from speakers, the TV, Hi Fi or any magnetic field. They can 'corrupt' and programs may be lost.



SECTION 4

MORE ABOUT PROGRAMMING



CHAPTER 16
Conditional statements

IF IS A CONDITIONAL STATEMENT

The function IF is used by the computer as a conditional statement. That is the computer will act on the statement after considering the information it has received.

EXAMPLE

Type and RUN this program:

```
1Ø PRINT "WHAT IS 3 + 4"
2Ø INPUT "MY ANSWER IS...";A
3Ø IF A = 3 + 4 THEN PRINT "CORRECT"
4Ø IF A <> 3 + 4 THEN PRINT "INCORRECT"
```

NOTE: To type the sign <> hold down the SHIFT key and press the keys with the signs < and >. It does not matter if you type >< as the computer recognises both forms.

EXPLANATION

You will recognise the basis of this program from previous chapters. Answer the question both correctly and incorrectly. After you have given the computer your answer it will apply it firstly to line 30, (because that is the next line in the program) and IF the answer is correct, that

is IF A = 3+4, the command on that line will be effected, i.e. it will print the word "CORRECT". Should the answer be incorrect, i.e. IF A <> 3+4 the computer considers that line 30 does not apply and then passes to line 40 and prints the word "INCORRECT".

You will know that the sign = means "equal to" but you may not be sure about the sign <> at line 40. <> or >< means "is not equal to". Therefore whenever IF is used in this way you may well have to use both = and <> or >< as the computer's next move will be conditional on the information it has received.

= AND < > CAN ALSO BE USED WITH WORDS

Both of these signs can be used with words as well as numbers as you will have gathered from previous chapters where the conditional IF keyword has been used.

MORE SIGNS

There are other signs which are used in the same way as = and <> or ><. These are:

<, which means "less than"

>, which means "more than"

=< or <=, means "less than or equal to"

=> or >=, means "more than or equal to"

EXAMPLE

Here is a program which demonstrates the different functions of these signs:

Type, and RUN.

```
1Ø PRINT "THINK OF A NUMBER BETWEEN 3 AND 8"
2Ø INPUT "MY NUMBER IS...";A
25 PRINT
3Ø PRINT A
4Ø IF A< 5 THEN PRINT "LOW"
5Ø IF A<=5 THEN PRINT "FIVE OR UNDER"
6Ø IF A>6 THEN PRINT "HIGH"
7Ø IF A>=6 THEN PRINT "SIX OR OVER"
75 PRINT
8Ø GOTO 1Ø
```

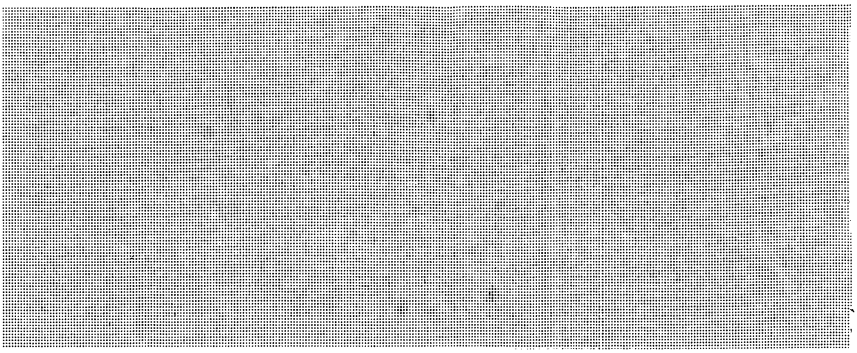
EXPLANATION

If you go through the numbers 3 to 8 in sequence you will see how the computer acts on the instructions in the conditional lines. The computer will consider the conditional lines (IF lines) in sequence; if the number input is either 3 or 4 it will act as follows:

LINE 40 — Is the number less than (<) 5? — answer yes — then print "LOW"
LINE 50 — Is the number 5 or less than (<=) 5? — answer yes — then print "FIVE OR UNDER"
LINE 60 — Is the number more than (>) 6? — answer no — no action
LINE 70 — Is the number 6 or more than (>=) 6? — answer no — no action

If the number INPUT is 7 or 8 the same principle applies but this time the computer will act on lines 60 and 70.

However if the number INPUT is 5 the computer will consider line 40 and ignore it because 5 is not less than (<) 5. It will, though, act on line 50 because 5 is equal to 5. Remember the sign <= means "is equal to or less than".



If the number INPUT is 6 the computer will only act on line 70. Why? . . .

That's right! >= means equal to or more than.

TWO IMPORTANT WORDS USED WITH IF

Let us now look at AND and OR. These are known as Logical Operators and are usually used with the keyword IF.

THE MEANING OF "AND"

AND means that one statement AND another statement must be true for the computer to comply with the command.

AN EXAMPLE OF ITS USE

Type and RUN this short program:

```
1Ø PRINT "GIVE ME THE LOWEST FOUR NUMBERS IN
   SEQUENCE WHICH ARE DIVISIBLE BY 5"
2Ø INPUT "LOWEST NUMBER IS...";A
3Ø INPUT "NEXT LOWEST...";B
4Ø INPUT "NEXT LOWEST...";C
5Ø INPUT "NEXT LOWEST...";D
```

```
7Ø IF A = 5 OR B = 1Ø OR C = 15 OR D = 2Ø
   THEN PRINT "GOOD TRY, THE CORRECT ANSWER
   IS 5, 1Ø, 15, 2Ø"
```

```
6Ø IF A = 5 AND B = 1Ø AND C = 15 AND D = 2Ø
   THEN GOTO 9Ø

7Ø PRINT "INCORRECT"

8Ø END

9Ø PRINT "CORRECT"
```

The conditional line 60 is dependent on the answers in lines 20, 30, 40 and 50 being correct to send the computer to line 90. A wrong answer at any of those lines will result in the computer executing line 70. Why? That's right — because all the answers must be correct.

Now change line 70 as per the above listing.

The condition in the new line 70 means that if ANY of the numbers are correct the computer will print the words "GOOD TRY, THE CORRECT ANSWER IS 5, 10, 15, 20".

LOGICAL OPERATORS CAN BE USED WITH WORDS

As with the signs = and <> the operators AND and OR can be used with words as well as numbers. Their use can be very effective where your computer is being used as a teaching aid. Consider the following:

```
1Ø PRINT "WHAT IS THE CAPITAL OF FRANCE"
2Ø INPUT "MY ANSWER IS...";A$
3Ø IF A$ = "PARIS" THEN PRINT "CORRECT"
4Ø IF A$ <> "PARIS" THEN PRINT "INCORRECT,
   THE ANSWER IS PARIS"
```

AN EXAMPLE OF THE USE OF "OR"

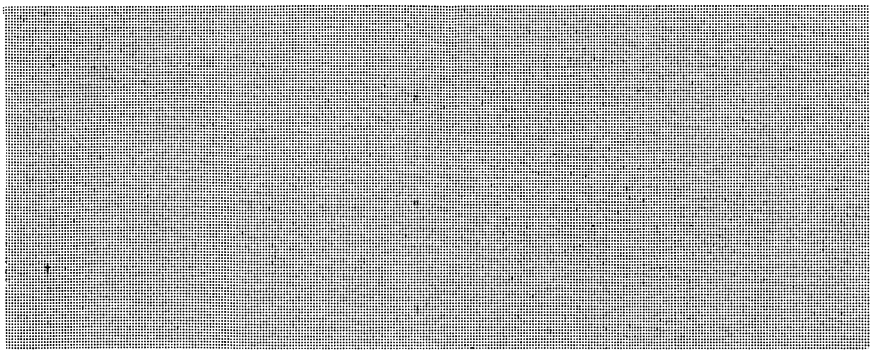
Type and RUN the above program.

For the sake of argument let us suppose that the word PARIS is incorrectly spelt, e.g. PARRIS. The answer is almost correct, but the computer only recognises PARIS as the correct answer. We could, therefore, change lines 30 and 40 to include as many combinations of spelling, as we realistically think possible, for example:

```
3Ø IF A$="PARIS" OR A$="PARRIS" OR A$="PARISS"
   THEN PRINT "CORRECT"
4Ø IF A$ <> "PARIS" AND A$ <> "PARRIS" AND
   A$ <> "PARISS" THEN PRINT "INCORRECT, THE
   ANSWER IS ";"PARIS"
```

Leave this program on the screen whilst we consider the final word to be looked at in this chapter — NOT. This like AND and OR is a Logical Operator and again is usually used with the keyword IF.

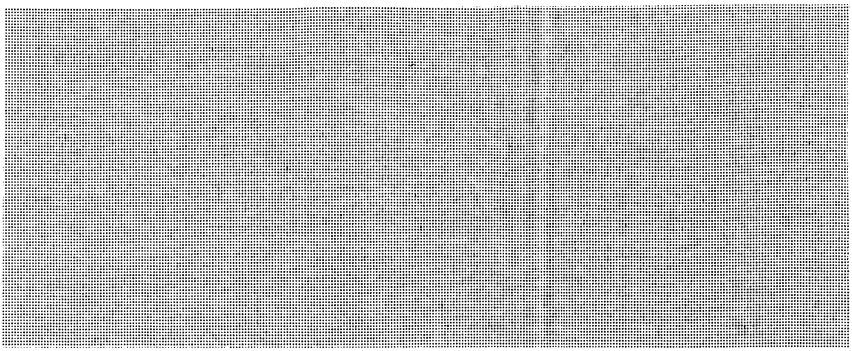
It is, however, unlikely that you will use this function as it is easier to obtain the same result by using the sign <>.



AN EXAMPLE OF THE USE OF "NOT"

EDIT line 40 in the PARIS program to:
40 IF NOT A$="PARIS" AND NOT A$="PARRIS" AND NOT A$="PARISS" THEN PRINT "INCORRECT, THE ANSWER IS ";" PARIS"

RUN the program again — making sure you get the answer wrong of course — the result is the same as before.



CHAPTER 17

How to handle DATA

AN EXAMPLE OF "READ" & "DATA" EXPLAINED

This chapter is meant as an introduction only. It concentrates on showing examples of how you can begin to use READ, DATA and DIM.

The first program shows you how to use READ and DATA. Look at the program explanation carefully.

Type and RUN:

```
 5 REM AVERAGES USING READ AND DATA
1Ø LET T = Ø
2Ø FOR K = 1 TO 5
3Ø READ N
35 PRINT N
4Ø LET T = T + N
5Ø NEXT K
6Ø LET A = T / 5
7Ø PRINT "THE AVERAGE IS";A
8Ø DATA 6, 4, 7, 2, 1
```

In the program above the numeric variable T stands for total, and on line 10, T is set at Ø initially.

LINE 20 — begins a FOR — NEXT loop — this continues until line 50 — NEXT K. Within this loop line 30 READs N, which is the first number in the DATA statement on line 80, which is 6.

LINE 35 PRINTS N.
LINE 40 — then establishes a new total T = T + N. The new total is T = 0 + 6, that is, 6.
LINE 50 — sends the program back to 20 for the second run through the loop. This time line 30 READs N, which is the second number in the DATA statement, 4. Line 40 establishes a new total T = 6 + 4, that is 10.

The program runs through the loop a third, fourth and fifth time. After the fifth time through the loop, the program runs through to line 60 which establishes a new variable A, (for average).

Average = total ÷ 5. After five loops the total is now 20, therefore 20 ÷ 5 will be the average.

The program then runs through to line 70 and prints the answer 4.

NOW YOU GO ON

Can you adapt the program to find the average of 3 numbers, then 10 numbers, then 20 numbers?

ANOTHER EXAMPLE EXPLAINED

READ and DATA statements are very commonly used in computer programs. Programs can be written which use string variables instead of numeric variables. The program below uses both string and numeric variables in the same program.

Type and RUN the following:

```
 5 REM TO PRINT A PASS/FAIL LIST
1Ø FOR K = 1 TO 5
2Ø READ N$, M
3Ø PRINT N$, M
4Ø IF M>5Ø THEN PRINT "PASS"
5Ø IF M<51 THEN PRINT "FAIL"
6Ø DATA "FRED",25,"JOAN",44, ALAN",63,"RITA",
   89,"BRIAN",67
```

This is what the screen displays:

```
FRED       25
FAIL
JOAN       44
FAIL
ALAN       63
PASS
RITA       89
PASS
BRIAN      67
PASS
```

As in the previous program, there are five items of data, five names (N$) and five numbers representing the marks they obtained (M).
LINE 20 — READs the name, then the number.
LINE 30 — prints the name, then the mark obtained.
LINE 40 — prints PASS if the mark is higher than 50.
LINE 50 — prints FAIL if the mark is less than 51.

Again the FOR—NEXT loop (lines 10-60) runs through five times.

AND IT'S OVER TO YOU

Can you adapt the program to PRINT a PASS/FAIL list for twelve pupils, where the pass mark is 80 and the marks are 32, 94, 26, 83, 70, 55, 99, 37, 85, 10, 91, 70?

AN EXAMPLE OF ARRAYS

READ and DATA can also be used together with the keyword DIM. Look at the following program:

```
  5 REM Simple Bubblesort
 1Ø DIM A (5)
 2Ø FOR K = 1 TO 5
 3Ø READ A (K)
 4Ø NEXT K
 5Ø FOR C = 1 TO 4
 6Ø FOR J = 1 TO 4
 7Ø IF A (J)<A (J + 1) THEN GOTO 11Ø
 8Ø LET S = A (J)
 9Ø LET A (J) = A (J +1)
1ØØ LET A (J + 1) = S
11Ø NEXT J
12Ø NEXT C
13Ø FOR K = 1 TO 5
14Ø PRINT A (K)
15Ø NEXT K
16Ø DATA 6, 3, 7, 2, 1
```

When we use the keyword DIM (Dimension) it is a way of reserving space in the computer which can be used later. We talk about DIMensioning an array. Therefore, a DIM A (5), line 10 tells the computer to reserve five spaces for a numeric variable, A.

The numeric variable A is represented on line 160 by the five numbers 6, 3, 7, 2, 1.

The program works as follows:

LINE 50 — sets up the number of passes through the DATA, that is, 4.
LINE 60 — controls the number of comparisons in each pass that is, 4. e.g. 6 with 3.
The computer READs the DATA on line 160 and compares the first two numbers.
LINE 70 — If the first number compared is smaller than the second then the program goes to the next comparison and compares the second number with the third number.

If the statement line 70 isn't true then lines 80, 90 and 100 change the first two numbers around.

Lines 130-150 — finally print the list of numbers in ascending order e.g. 1, 2, 3, 6, 7.

This program is called a bubblesort.

A DIFFERENT USE OF DIM

This program shows a different use of DIM plus READ and DATA:

```
 5 REM TYPING OUT CHOSEN SENTENCES
1Ø DIM A$ (5)
2Ø FOR J = 1 TO 5
3Ø READ A$ (J)
4Ø NEXT J
5Ø PRINT "HERE IS A LIST OF SENTENCES
   TYPE IN THE NUMBER OF THE SENTENCE
   YOU REQUIRE
```

```
 6Ø FOR Q = 1 TO 5
 7Ø PRINT Q, A$ (Q)
 8Ø NEXT Q
 9Ø INPUT "MY ANSWER IS ";B
1ØØ PRINT A$ (B)
11Ø DATA "THE WEATHER IS SUNNY.", "THE WEATHER
    IS WINDY.", "THE WEATHER IS RAINY.",
    "IT IS SNOWING.", "IT IS FOGGY."
```

This is what the screen displays:

```
HERE IS A LIST OF SENTENCES. TYPE IN THE NUMBER
OF THE SENTENCE YOU REQUIRE.

1            THE WEATHER IS SUNNY.

2            THE WEATHER IS WINDY.

3            THE WEATHER IS RAINY.

4            IT IS SNOWING.

5            IT IS FOGGY.
```

In this example the DIMensional array in line 10 is set to reserve five spaces, 10 DIM A$(5).

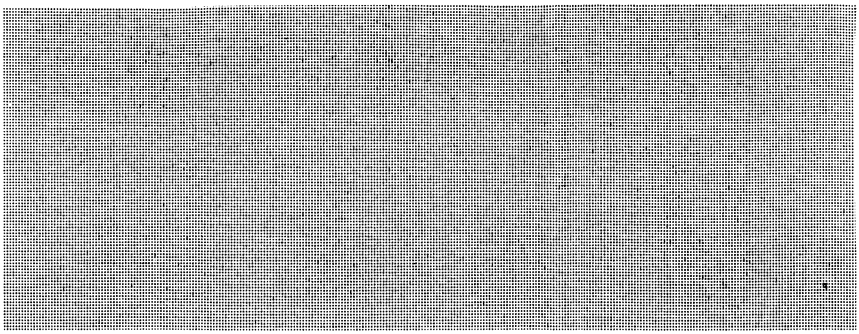
Lines 20, 30, 40 — READ the DATA in line 110
LINE 50 — PRINTS your heading and lines 60, 70, 80 PRINT the sentences in the DATA statement on line 110.
LINE 90 — asks you to choose your sentence and line 100 PRINTs it.

On its own, this program may not seem very useful. However, where you have to choose a selection of comments from a large number of comments, or where you might be writing many letters with slight variations, you can see how useful an adaptation of such a program might be.

Finally, a simple alphabetical bubblesort. Compare this with the numeric bubblesort. NOTE: The computer will take approximately 10 seconds to RUN this program.

```
 5 REM Simple alphabetic bubblesort
1Ø DIM A$ (26)
2Ø FOR K = 1 TO 26
3Ø READ A$ (K)
4Ø NEXT K
5Ø FOR B = 1 TO 25
6Ø FOR J = 1 TO 25
7Ø IF A$ (J) < A$ (J + 1) THEN GOTO 11Ø
8Ø LET S$ = A$ (J)
9Ø LET A$ (J) = A$ (J + 1)
1ØØ LET A$ (J + 1) = S$
11Ø NEXT J
12Ø NEXT B
13Ø FOR K = 1 TO 26
14Ø PRINT A$ (K)
15Ø NEXT K
16Ø DATA Q, W, E, R, T, Y, U, I, O, P, A, S, D, F,
    F, H, J, K, L, Z, X, C, V, B, N, M
```

Not only does the computer have the ability to distinguish between higher and lower NUMBERS, but it can also order LETTERS. That is, it is able to rank in alphabetic order — or reverse alphabetic order!



NOW YOU CAN EXPERIMENT

The preceding programs have served as an introduction to simple uses of DIM, READ and DATA. Experiment with some ideas of your own.

CHAPTER 18

How to adapt programs

INTRODUCTION

We feel that many educational programs do not fit the individual's needs. We have shown already how to alter simple number programs according to the level of difficulty required. Here is a more complex program, which you may not fully understand yet but you can adapt to you own needs.

```
1ØØ REM MULTI CHOICE (RANDOM) PROGRAM
14Ø DIM A$ (10)
15Ø DIM B$ (10)
16Ø DIM T (5)
17Ø GOSUB 83Ø
18Ø PRINT " [SHIFT] [CLR HOME] "
21Ø FOR N = 1 TO 5 : LET T (N) = Ø : NEXT N
26Ø FOR N = 1 TO 5
27Ø LET R = INT (RND * 1Ø) + 1
31Ø FOR Z = 1 TO 5
32Ø IF T (Z) = R THEN GOTO 27Ø
33Ø NEXT Z
34Ø LET T (N) = R
35Ø NEXT N
38Ø LET Q = INT (RND * 5) + 1
41Ø PRINT "QUIZ"
```

```
411 PRINT
42Ø PRINT "WHAT IS THE CHEMICAL SYMBOL FOR▲
    ";A$ (T (Q))
43Ø FOR N = 1 TO 5
44Ø PRINT N;"                                ";B$(T (N) )
45Ø NEXT N
46Ø PRINT "ANSWER (1-5) =";
5ØØ GET R$ : IF R$ <>"1" AND R$ <> "2" AND
    R$ <> "3" AND R$ <> "4" AND R$ <> "5"
    THEN GOTO 5ØØ
51Ø PRINT R$
56Ø IF VAL (R$) = Q THEN PRINT "CORRECT": GOTO 66Ø
57Ø PRINT "INCORRECT-WAIT FOR CORRECT ANSWER!"

571 PRINT
64Ø PRINT "ANSWER WAS ";B$ (T (Q))
65Ø FOR N = 1 TO 25ØØ: NEXT N
66Ø PRINT "ANOTHER GO? YES/NO"
661 FOR N = 1 TO 25: NEXT N
67Ø INPUT G$
68Ø IF G$ = "YES" THEN GOTO 18Ø
81Ø PRINT "END OF QUIZ"
82Ø END
83Ø FOR N = 1 TO 1Ø " READ A$ (N), B$ (N):
    NEXT: RETURN
84Ø DATA HYDROGEN, H, POTASSIUM, K, CARBON,
    C, NITROGEN, N, OXYGEN, O
85Ø DATA SULPHUR, S, HELIUM, HE, COPPER, CU,
    PHOSPHORUS, P, ZINC, ZN
```

This is a copy of what appears on the screen:

```
QUIZ
What is the chemical symbol for PHOSPHORUS
1              HE
2              H
3              ZN
4              P
5              C
Press a number between 1 and 5 - 4
Correct
Another go? YES/NO

QUIZ
What is the chemical symbol for ZINC
1              ZN
2              C
3              N
4              HE
5              P
Press a number between 1 and 5 - 3
INCORRECT-WAIT FOR CORRECT ANSWER!
Answer was 1
Another go? YES/NO
```

EXPLANATION

(As the computer is printing capital letters it is important to remember that the second letter should be a small one.)

What the program does:

(a) It stores in its DATA statement ten chemical symbols and also their names — see line 840.

(b) The computer then chooses the full word for the symbol at random and asks "What is the chemical symbol for ... Hydrogen " (for instance). You are given a choice of the symbols to choose from and asked to press a number between 1 and 5. The correct answer is always one of the choices.

(c) If you are correct the computer tells you, and if you are incorrect, you are given the right answer.

(d) You are then asked if you want another go. If you type YES, you get another go. If you type NO the program ends.

Type it carefully, and RUN it.

THE PROGRAM TEACHES AS WELL AS TESTS

The beauty of a program such as this is that it teaches as well as tests. You could easily learn these symbols without any knowledge beforehand.

Because the computer is choosing the words at random, some words will be repeated perhaps three or four times before every word has been presented once.

HOW TO ALTER THE PROGRAM

Let us say you wanted to ask the capitals of ten countries in Europe — this is what you would alter:

LINE 420 to read "WHAT IS THE CAPITAL OF ▲"; A$ (T(Q))

LINE 840 DATA "FRANCE", "PARIS", "RUSSIA", "MOSCOW", etc. to a total of ten countries and ten capitals.

NOTE: It is not necessary to use speech marks in a DATA line unless you are using a SPACE, a COLON or a COMMA. See DATA lines in the next program as an example.

ANOTHER EXAMPLE

The next program is shorter and simply sets up two arrays on lines 10 and 15, one for French words and the other for the English equivalent. You are asked to give the French for the English word which is presented on the screen in the form of a question. You then INPUT your answer — line 90:

```
 5 PRINT " [SHIFT] [CLR HOME] "
10 DIM F$ (5)
20 DIM E$ (5)
30 FOR K = 1 TO 5
40 READ F$ (K), E$ (K)
50 NEXT K
60 LET T = 0
70 LET R = INT (RND (0) * 5) + 1
80 PRINT "WHAT IS FRENCH FOR ▲"; E$ (R)
81 PRINT
90 INPUT "MY ANSWER IS ▲"; A$
91 PRINT
110 IF A$ = F$ (R) THEN GOTO 210
120 PRINT "INCORRECT TRY AGAIN.

130 LET T = T + 1
```

```
14Ø IF T = 3 THEN GOTO 16Ø
15Ø GOTO 9Ø
16Ø PRINT "THE ANSWER IS ▲"; F$ (R)
161 PRINT
17Ø GOTO 6Ø
21Ø PRINT "WELL DONE. YOUR ANSWER IS CORRECT"
22Ø PRINT: GOTO 6Ø
5ØØ DATA "LE JARDIN", "THE GARDEN", "LE CHIEN",
    "THE DOG", "LE CHAT", "THE CAT"
51Ø DATA "LA ROBE", "THE DRESS", "LA CRAVATTE",
    "THE TIE"
```

If your answer is correct, it is confirmed on line 210. If your answer is incorrect you are asked the same question again. After three tries, you are told the correct answer. This is simply a clumsy way of having three quick attempts if you do not know the answer!

The program randomizes the words so that it is possible to be asked one word twice before all the words have been asked once!

Note too that this program is very particular. You must enter your answer EXACTLY as it is in the DATA statement. Forgetting the space or even one spelling mistake will give you the response INCORRECT!

HOW TO ADAPT IT

If you want to adapt this program for more questions you will need to look at lines, 10, 20, 30, 70, 80, (if you want to ask a different question) and 500. You don't have to change F$ or E$. Remember that E$ and F$ can stand for anything. They stand for English and French in our program merely to help you to understand the program.

To summarise, this program can handle material where there is one initial question such as "What is French/German/Arabic for ..." or "Who was monarch in ..." and the DATA is in the form of two blocks of information e.g. "the house", "Das Haus", or "1627", "Charles I".

VARIATIONS ON A THEME

The next two programs are variations on a theme, multiplication tests, one giving three attempts, the other one giving you the choice of whether to stop after the first attempt.

```
 5 LET A = INT (RND (Ø) * 12) + 1
1Ø LET B = INT (RND (Ø) * 12) + 1
15 PRINT "WHAT IS ";A" TIMES ";B;"?"
2Ø FOR N = 1 TO 3
25 LET C = Ø
3Ø INPUT C
35 IF C <> A*B THEN PRINT "TRY AGAIN"
4Ø IF C = A*B THEN GOTO 6Ø
45 NEXT N
5Ø PRINT "WRONG, THE ANSWER IS; A*B
55 GOTO 5
6Ø PRINT "CORRECT"
65 GOTO 5
```

```
 1Ø PRINT " [SHIFT] [CLR HOME] "
 2Ø LET A = INT (RND (Ø) * 12) + 1
 3Ø LET B = INT (RND (Ø) * 12) + 1
 4Ø PRINT "WHAT IS ";A; "TIMES";B
 5Ø INPUT "MY ANSWER IS ";C
 6Ø PRINT
 7Ø PRINT "YOUR ANSWER IS ";C
 8Ø PRINT
 9Ø IF A*B = C THEN PRINT "CORRECT.
    WOULD YOU LIKE ANOTHER GO?
    TYPE IN YES OR NO"
 95 IF A*B <> C THEN GOTO 1ØØØ
1ØØ INPUT A$
11Ø IF A$ = "YES" THEN GOTO 1Ø
115 IF A$ = "NO" THEN GOTO 2ØØ
135 INPUT B$
14Ø IF B$ = "YES PLEASE" THEN GOTO 4Ø
15Ø IF B$ = "NO THANKS" THEN PRINT
    "THE CORRECT ANSWER IS "; A*B
2ØØ END
1ØØØ PRINT "WOULD YOU LIKE ANOTHER GO?
    TYPE IN YES PLEASE OR NO THANKS."
1Ø1Ø GOTO 135
```

A TEST OF LOGIC

This is a number guessing game which is also a test of logic! Who can guess the number in the least time?

```
 1Ø REM NUMBER GUESSING GAME
 2Ø PRINT " [SHIFT] [CLR HOME] "
 4Ø LET N = INT (RND (Ø) * 100) + 1
 45 LET T = Ø
 5Ø PRINT "GUESS A NUMBER BETWEEN 1 AND 1ØØ"
 6Ø IF T = 1Ø THEN GOTO 5ØØ
 7Ø LET T = T + 1
 8Ø INPUT G
 9Ø IF G = N THEN PRINT "WELL DONE.
    YOU GOT THE ANSWER IN ";T; "GOES":
    GOTO 51Ø
1ØØ IF G<N THEN PRINT "TOO LOW.
    YOUR GUESS WAS ";G:GOTO 5Ø
11Ø PRINT "TOO HIGH. YOUR GUESS WAS
    ";G
12Ø GOTO 5Ø
5ØØ PRINT "SORRY, NO MORE GUESSES LEFT"
51Ø PRINT "THE ANSWER WAS :";" N
```

You can adapt this to give you any number of attempts by altering LINE 60 and you can also alter the limits of the numbers by altering LINE 40.

INDEX

Adaptation of Programs 114, 116, 119
Addition 45
AND keyword 98

Brackets (using them) 47
Branching 71-79
Bubblesort 100

Calculations 45, 53
CLR HOME key 78
Comma, uses of 24, 25
Command keys 10, 18, 27
Command words 50
Conditional Statements 95-101
CONT Command 48
CRSR keys 10, 18, 27
Counting 33, 34
Cursor 7, 8, 9

DATA Statement 103-110
DIM Statement 103
Division 47

END Command 31
ENTER 10
Errors — typing 15-19
Example programs.
— Addition 77
— Alphabetic Bubblesort 109
— AND Demonstration 98, 100
— Averages 103
— Christmas Thank You Letters 42
— Conditional Statements Demonstration 95, 97
— Correcting Mistakes 17, 19
— Division 47
— END Demonstration 31-33
— END Test Exercise 34
— FOR/NEXT Demonstration 52
— French Vocabulary 72
— GOSUB Demonstration 73, 74
— GOTO Demonstration 71
— History Quiz 64
— Input Demonstration 37, 38, 40, 42
— Kings & Queens 67
— Mental Arithmetic Quiz 38, 40
— Multiplication 46
— Multiplication Tests 117, 118
— Multi-choice Chemistry Quiz 111, 112
— Multi-choice French Quiz 115, 116
— Number Guessing game 119
— Numeric Bubblesort 106
— Numeric Variables Demonstration 50, 53, 54
— OR Demonstration 99, 100
— Pass/Fail Marks 105
— Primitive Word Processor 69
— Print a List of Numbers 48
— PRINT Demonstration 23, 24, 25, 26
— REM Demonstration 21
— RND Demonstration 55, 56, 57
— Shopping List 65, 66
— Subroutine Demonstration 82
— Subtraction 46
— Typing Chosen Sentences 107

Fields 25
FOR/NEXT Command 51
Formula 57
Fractions 60

GOSUB 81
GOTO 71-79
Graphics Characters 9

IF/THEN Statements 73, 95
INPUT Command 37, 41
INST/DEL key 15
INT Function 55, 56

Keys 8, 9, 10

LET 61
Line Numbers 11
Logical Operators 99-101
Long Multiplication 58
LOOPS 51, 71

Memory 11, 12
Multiplication 46

NOT keyword 101
Numeric Variables 49

OR keyword 98, 100
Order of Calculations 47

Plugging in 7
Powers, raising numbers to 60
PRINT Command 23
Printer
— uses of 89
— adapting for 89

Quotation marks ..... see speech marks

Random Numbers 54, 56, 57
READ Statement 103-110
READY 7, 8
REDO FROM START 38
REM Statement 15, 21
RESTORE key 38, 41
RETURN key 10
RND function 51, 56, 57
Rounding of Numbers 55
RUN Command 11
RUN/STOP key 32, 38, 41
RVS ON/OFF key 9

Semi-colon 24, 25
Signs
— + ... 45
— − 46
— * 46
— / 47
— ↑ 60
— < 96-98
— > 96-98
— <> 96-98
— <= 96-98
— => 96-98
Spacing (spaces) 25
Speech Marks 17, 23, 28, 29, 39, 63
String Variables 63
SYNTAX ERROR 65

TAB Command 27
Tapes 91
Timing Statement 39, 58, 60
Typing Errors 15

VERIFY Command 91

▲ 13, 24, 25, 52

This book is designed to guide the new Commodore 64 user through the first few weeks of programming – from the moment the machine is plugged in to a proficiency in basic programming.

Topics include:

- ★ Introducing the keyboard
- ★ Useful computing aids such as recorders and printers
- ★ Using easy commands
- ★ How to build a program
- ★ Programming techniques with advice on variables, looping, branching and counting
- ★ Practical applications

This book is essential reading for any first-time Commodore 64 user. It manages to avoid technical jargon and is also great fun to use. All example programs are presented in quiz, game or self-questionnaire form, fully illustrated for easy reference.

ISBN 0 7126 0412 X
CENTURY PUBLISHING CO. LTD



This was brought to you

from the archives of

http://retro-commodore.eu