

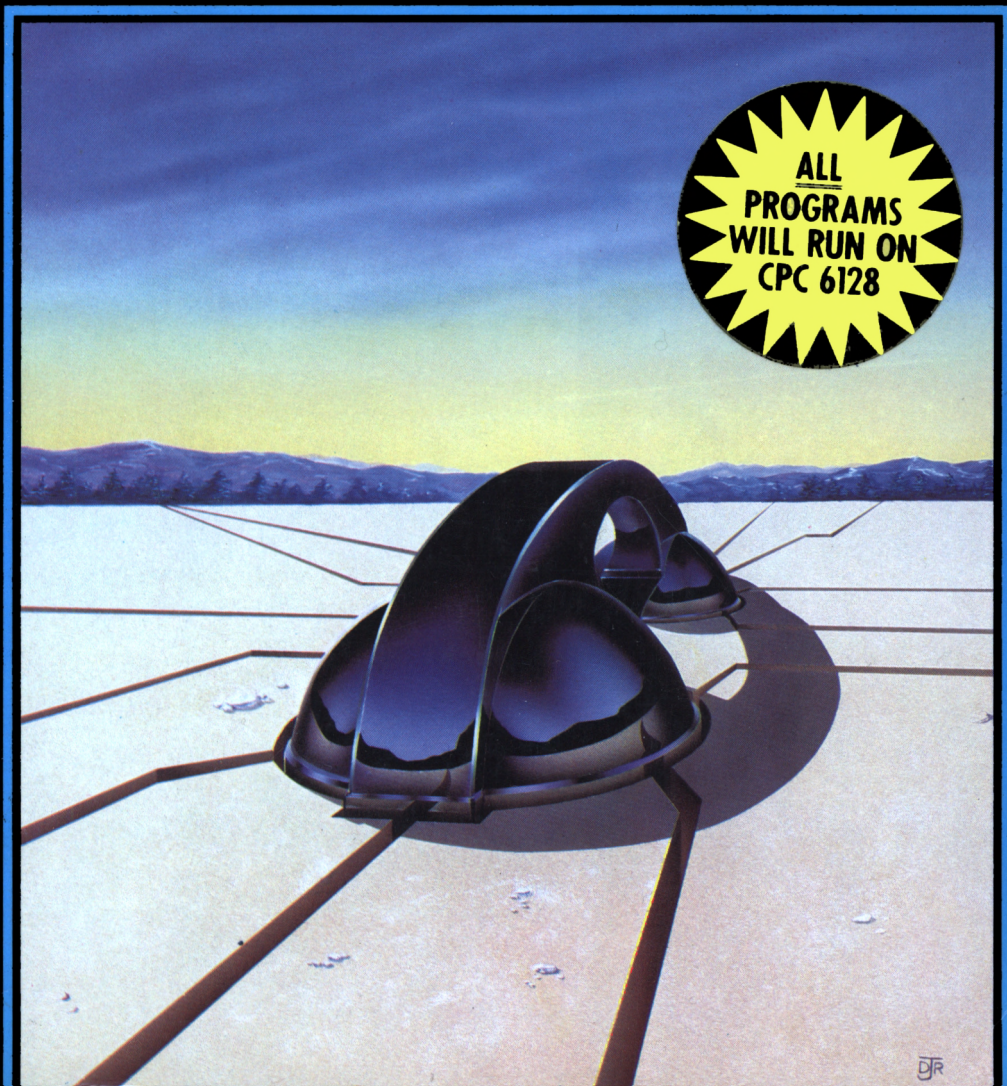
ARGUS BOOKS

APPLICATIONS FOR THE AMSTRAD

CPC464 & 664

GARRY MARSHALL

**ALL
PROGRAMS
WILL RUN ON
CPC 6128**



GR

Applications for the AMSTRAD CPC464 & 664

**Applications for the
AMSTRAD CPC464 & 664**

Garry Marshall

ARGUS BOOKS

Argus Books Limited
1 Golden Square
London W1R 3AB

©Argus Books Ltd 1985
ISBN 085242 853 7

All rights reserved. No part of this publication may be reproduced in any form, by print, photography, microfilm or any other means without written permission from the publisher.

Phototypesetting by
Contact Typesetting Limited,
14 Stoneleigh Park Road,
Ewell, Epsom, Surrey.
Telephone: 01-393 0998/9

Printed and bound by Whitstable Litho

Preface

What useful and worthwhile tasks can be carried out with an Amstrad computer? This is the question that this book sets out to answer.

Information can be handled in any of an unlimited number of ways with the aid of a program to tell the computer how to proceed. Word processing and the storage and retrieval of information are among the most useful information-related activities. The computer can be used to solve problems of almost any kind. The methods that are required for a computer to do this are rather different from those used otherwise. Some of them are demonstrated here. A computer can be used to control other equipment. The equipment can range from a printer, which is an obvious essential for word processing, to a robot, the uses for which are still being invented. Computers can also be used for communications, and a whole range of new services is being developed in this area. Giving a computer the ability to communicate allows it to be used as a sort of enhanced CB radio or for accessing information and computer services that are based almost anywhere in the world.

The applications for the Amstrad computers are described and explained under four headings: software-based applications, problem solving, hardware-based applications and communications. Many of the applications that are covered are, in their details, highly specific to the Amstrad computers because they describe what can be accomplished with software or hardware developed specifically for use with an Amstrad computer. The other applications, while they can, of course, be carried out from an Amstrad, could also be achieved with other computers, because they depend on the use of standard equipment. The Amstrad possesses the same sort of standard sockets to which equipment can be attached as many other kinds of computers. In this way, the applications consist of a mixture of some that are

tied to the Amstrad and some that are more general. When tackling a particular application, there may be advantages, perhaps in terms of its cost or of its performance, in dealing with it in a way that is Amstrad-specific or that is general.

No matter what uses you may have in mind for your Amstrad, I hope that you will find something of help in this book, and also that you may find some ideas for further uses.

Contents

1 Software-based applications **1**

What can the computer do? The major applications. Word processing. Who can benefit from word processing? A word processing session. Databases. A card index and a database. Spreadsheets. Examples of using a spreadsheet. Other applications. Summary.

2 The Amstrad software **25**

Word processing with Amword. The commands. Database. The Commands. Spreadsheets and Easi-Amscal. The commands. Other software. Summary.

3 Problem solving **40**

Networks and routing. Grazing problems. How to number pages. Classification and coding. Conclusions.

4 Hardware-based applications **70**

Cassettes and disks. Printers and plotters. Joysticks, light pens and mice. Robots. Educational robots. Personal robots. Other equipment. Summary.

5 Communications **96**

The hardware and the software. Applications. Prestel. Databases. Bulletin boards. Telecom Gold. Summary.

Glossary of jargon terms **106**

Index **109**

Introduction

This book is about the ways in which you can use your Amstrad computer. It is not about using it for playing games, but about using it in ways which will be beneficial and genuinely useful. The applications that are described are aimed at allowing the potential of the computer to be tapped in the home. The word that tends to be used for such applications is 'serious'. In a way, it is accurate, but it should not be taken to mean that there is no fun to be had from using the computer in these ways.

The applications in which the computer can be of immediate value include storing information, for instance, about the activities of the family and the situation in the home, in such a way that any of it can be readily recovered, whether it is a forthcoming appointment with the doctor, a recipe for a special occasion or a list of what is required to restock the freezer. They also include many activities that will be educational. These may complement the school and college studies of a member of the family, or they may provide a less formal up-dating of the skills of someone at work. Attaching a 'turtle' to the family computer can allow a child to use it as it would be used at school, but in an environment where he or she has sole and uninterrupted use of it, as opposed to the situation in school where even short periods of use may have to be shared. With a small robot attached to the home computer, industrial uses of computers can be investigated and simulated, and ways in which it can be used to advantage in the home can be explored. And there are also the unexpected problems that inevitably crop up from time to time, which can be solved as long as you know something about the special methods for problem-solving that are used with computers.

In this book, the applications for the computer have been selected in such a way that they are not trivial but are worthwhile and, at the same time, are within the range of what can be achieved with a personal computer. The underlying thrust is that

the computer should be able to enhance our lives. The applications described in this book all show, to a greater or lesser degree, how the computer can be used so that it does improve the quality of our lives.

The first two chapters of the book deal with applications that become possible by running the programs that can be bought for the Amstrad computers. By running a program, the computer is transformed from an essentially useless mass of electronics to a machine that is configured for a particular task and, further, is ready to carry out that task. The major programs that we examine are for word processing, for storing information and for planning. Word processors, databases and spreadsheets, as these types of programs are known, are widely used commercially, but they have a surprising number of uses in the home, and can provide improved ways of carrying out existing tasks as well as being tools with which new activities can be brought within the range of what can be successfully accomplished. We examine a number of such tasks and activities, and there are others waiting for you to discover them.

The final two chapters are about the equipment that you can attach to the computer to widen its range of application, and also about how the resulting combination can be used to advantage. We have already mentioned robots as one of the items that can be connected to and controlled by a computer. Others range from a printer, that will be essential if you use the computer for word processing, to the 'modem' that is needed if you use the computer to communicate over the telephone, either for passing messages or for accessing the information that is stored in other computers that are also connected to the telephone network.

One essential difference between the first two chapters and the last two is that programs for Amstrad computers are specific to them and, in general, will not run on another computer. In contrast, the equipment that can be attached to the computer can be attached to many other computers because it must be plugged in at a standard socket, and most computers possess these standard sockets. This leads to a distinct contrast between the treatment of the applications in the two pairs of chapters.

The coverage of the applications that become possible by running software is, in part, a description of how to use particular programs for particular purposes. To provide a framework within which these accounts can be placed, discussions of word processing, databases and spreadsheets precede the treatment of the specific programs. The general discussions describe the whole range of what can be accomplished, so that the capabilities of the individual programs can be compared with this to ensure that

they can be used to do all that you would like to do, or so that it is clear if some feature that you might need is missing. If the item of software discussed here does not meet all your requirements, there are probably others that will.

In the chapters covering the use of the equipment that can be attached to the computer, there is much less that is specific to the Amstrad machines. This is simply because attaching, for example, a printer via the standard socket will give entirely the same behaviour with an Amstrad computer as with any other having the same socket. Similarly, accessing Prestel with an Amstrad will be just the same, and will give the same results, as accessing it with another computer or, indeed, not accessing it with a computer at all but with the special Prestel adaptor.

The software chapters, then, spend a certain amount of time dealing with the specific way that programs written especially for the Amstrad are operated, although they also explain what can be done with them. In the hardware chapters, the emphasis is much more on what can be done than on how it is done, purely because these applications are always carried out in the same way.

In between these two pairs of chapters is a chapter on using the computer for problem-solving. Spreadsheets and databases, for example, can be used to tackle many and varied problems. But, life being the way it is, the next problem to come along is likely to refuse to be one of them. For this reason, it is a real asset to be familiar with the ways in which programs can be created for carrying out activities and for solving problems. Computer-based methods of solving problems are different from other methods, requiring a sort of lateral thinking. The examples in the problem-solving chapter have been chosen to try and illustrate a range of these methods. This chapter contains problems that are neither trivial nor extremely difficult but which show off a range of the solution methods that can be adopted when using a computer.

In this way, the book aims to show how programs can be used to good effect by the family, how the computer can be programmed to solve particular problems and meet special individual needs, and how the extra equipment that can be attached to a computer can be usefully applied. In all these ways, the computer can be used to enhance the life of the family that owns it, and so be of real value in the home. The applications that are described are intended to be of real use, and all the members of the family can benefit. And if you create your own program for some particular purpose, then the computer has become *yours* to an even greater extent. While one purpose of this book is to show that computers can be used for activities other than playing games, you should find that these applications are also great fun.

Software-based applications

1

A computer is a general-purpose machine; but from the point of view of its user, it is a 'blank' machine that is capable of nothing until it runs a program. When it is running a program, it becomes a special-purpose machine that is capable of performing a particular task. And as soon as this task is completed another program can be run to turn the computer into another special-purpose machine for performing another task.

It is the range of the software that is available for a computer that gives it its versatility. It is possible for a user to write the programs to make the computer carry out the tasks that he or she requires of it, but this can be a far from simple task. To write, let us say, a powerful and complex word processor requires a considerable degree of expertise. Besides, the people whose main interest in a computer, and whose reason for having it, is to use it to accomplish some specific task, or tasks, will have neither the time nor the inclination to program it themselves before it can be used for each and every activity.

The benefit of having a popular and widely owned computer such as the Amstrad is that a good deal of software is available for it with which it can be used to carry out worthwhile activities.

In this chapter, we shall first of all look at the sort of tasks that the Amstrad can help us to accomplish simply by running the appropriate software. In the next chapter we shall examine specific, but representative, items of the software that are available for each of these tasks, describing how the software is used and exactly what it can make the computer do.

What can the computer do?

The first questions that occur to people who have the opportunity to use a computer for the first time are almost invariably something like 'What can it do?' and 'What can I use it for?'. The

aim of this section is to answer these questions in general terms so that, armed with an appreciation of what a computer can do, it will be possible for the reader to begin to see how one of them can be used to meet particular, individual requirements.

It is generally known that personal computers can be used for education, although how this is done is not so widely known. (Actually, this is not surprising, because even the experts in computer-based education are far from certain about it!) Computers are also used for playing games. But, based in part on the knowledge that many offices successfully employ computers in their daily operations, there is a growing perception that it must be possible to use personal computers for commonly needed activities in the home, so that they can remove some of the drudgery and act to enhance life generally. It is fair to say that there is a desire, and even a demand, for computers to be used in this way.

To begin to explain how an Amstrad computer can be used in this way, we must examine what a computer does. Basically, a computer deals with information. It can accept information, store it, process and manipulate it, and it can communicate it.

This is fine as long as we know what is meant by 'information'. We can say that information is any data, facts, knowledge, or intelligence that are of interest and use to us and which we can pass to the computer for it to deal with. The information may take the form of a table of numbers, a paragraph of text or a picture. It is something that can be typed at the computer keyboard or, as with a picture, something which can be described from the keyboard. There are other ways for the Amstrad to acquire information but, as we are dealing with the use of the computer when it is doing nothing more than running a program, we will not mention them for the moment.

Some examples will probably help to illustrate what is meant by information and, while giving them, we can also describe some of the ways in which the computer can handle information. If the computer is to be used to create documents, whether they are memos, letters or books, it must be able to accept text and so, for this activity, the basic information, in the sense that this is what is typed at the computer's keyboard, consists of letters and punctuation marks. A program that enables the computer to create documents is known as a word processing program. It must be able to store the text, character by character, as it is entered and then to manipulate it so that the letters can be combined to give words, and then sentences and paragraphs. It must then arrange these various assemblages of letters so that they appear neatly on the computer's display screen and, if it is

required, on paper when the document is printed.

If we want the computer to keep information for us in such a way that we can recover it as and when we need it, then it can do this. Suppose that in a large family, the computer is used to keep the details of all the children's medical records, including their illnesses, their inoculations, their next appointments and so on. The information in this case consists of words, sentences, numbers, dates and much else. A program that can hold this sort of information is known as a database. It must be able not only to store the information but also to allow it to be recovered. This means that it must be able to associate items of information that it holds while taking advantage of the associations. For example, to find out which children have had mumps, the program must search its store for any occurrences of 'mumps' and then report the names of the children associated with these occurrences. To find if there are any doctor's appointments scheduled for today, it must search for occurrences of today's date and then report the associated appointment details.

As a third example, we can use the computer to keep the family's financial records. In this case, the information will be numbers that represent income and expenditure. The program must store the numbers and then perform arithmetic operations on them, such as subtracting expenditure from income to give the balance remaining. A spreadsheet is one kind of program that can accept and display the sort of table of numbers that is needed for this, and it can also perform the necessary arithmetic operations automatically.

The examples that we have given show that programs for handling information, no matter what its form, all conform to the same broad pattern and, in particular, must have the following three stages:

- 1 The program must enable the computer to accept information, usually from the keyboard, and store it. The information may consist of numbers, letters or facts. Without some information stored in it, so that it has some raw material to process, the computer can do very little.
- 2 The program must then tell the computer how to manipulate, or process, the information that it has been given. The way in which the information will be processed depends on its type. Arithmetic operations will be performed on numbers; letters will be combined into words, words into sentences, and so on. Facts will be scanned and searched to see if they match an item that someone is looking for.

3 The program must tell the computer how to communicate the results of what it has done. It is fruitless for the computer to accept and to process information unless it communicates to us the results of what it has done. The results will usually be displayed on the screen, although other equipment can be attached to the computer through which it can communicate with us in any of a variety of ways. The form in which the computer communicates its results is extremely important, for it can make them easy to understand or very difficult. Word processors and spreadsheets communicate their results most effectively, because a great deal of care has been spent on the parts of these programs that deal with the presentation of results. There is no need to present information in a certain form just because it was originally entered in that form. It may be much easier to understand the significance of a set of numbers, for example, if they are presented as a graph rather than as a table of numbers.

We can summarise all this by saying that a computer with the appropriate programs can carry out all the information-related tasks that occur in the home. It can keep the family accounts. It cannot rock the baby to sleep, although if you attached a robot to the computer and had a program that was the equivalent of a lullaby it could do so. It cannot spend your money, although with equipment that connects it to the telephone it can enable you to order goods from a shop without ever leaving home. We shall examine the applications that become possible after attaching extra equipment to the computer in Chapters 4 and 5, but for the moment we are concerned only with the information-related tasks that can be performed with a computer and the appropriate program.

The major applications

In this section, we shall look at the main information-related tasks for which the Amstrad can be used in the home. These are word processing, using a database and using a spreadsheet. We shall examine each in some detail, and then discuss who can benefit from them.

Word processing

We can say that word processing is using the capabilities of a computer to store, process and communicate text. The computer must do this in a way that satisfies anyone who has to write words, and so must make it possible to create items as diverse as a memo, a document of two or three pages, or a book. The

computer itself will carry out the same fundamental operations as it always does, whether it is being used to process words, to perform complex numerical calculations or to create graphics. As we have seen, it is by running a word processing program that it acquires the ability to accept the text that is typed at its keyboard and is then able to recognise words, sentences, paragraphs and pages in that text so that it can perform actions in which these are treated as basic units.

The computer cannot understand the text that is entered, and it would be to exaggerate the power of a word processing program to think that it gave the computer this ability. It could also lead to expecting more from a word processor than it could possibly deliver. A word processing program simply has in-built rules that allow it to recognise words, sentences and other linguistic units. One of these rules is that several consecutive letters followed by a space form a word. It is the 'trick' of being able to recognise that a space terminates a word that allows the word processor to distinguish a word. Another rule is that a word can end with a full stop. This will be the last word in a sentence, and so the word processor recognises a sentence as a number of words, the last of which ends with a full stop. There must be other, similar, rules to cover the use of question marks and exclamation marks.

A word processing program allows you to type at the keyboard just as you would with a typewriter. Pressing a letter key by itself gives a lower case letter. Holding down a SHIFT key while pressing a letter key gives a capital, or upper case, letter. The text is automatically stored in the computer's memory and is also displayed on the screen as it is typed. Storing the text in the computer's memory allows it to be processed or manipulated in any of a variety of ways which will vary with the particular word processor that is used, although there are certain operations that are provided by almost all of them. They allow alterations to be made to the text that has been entered, by deleting, inserting or replacing letters. This means that spelling mistakes can be corrected, for example. They also allow a block of text to be identified so that it can be deleted, copied or moved. This permits a larger amount of text to be treated as a single unit so that paragraphs or pages can be deleted or moved within a document.

The word processor also automatically arranges the text so that it is always neatly presented. The precise arrangement can be set by the user. The length of the lines can be given, as can the positions for the left and right margins and the indentation arrangement at a new paragraph in the text. Many word processors can display text on the screen, as it is typed in, in exactly the manner that has been prescribed. When printed on a

printer attached to the computer, it will appear in precisely the same form. This means that a document can be prepared and then checked on the screen to ensure that it is perfect in every way. It need not be committed to paper until both its content and its presentation are completely satisfactory. When the document is ready for printing, it only remains to issue the appropriate command.

It is stating the obvious to say that the text must be typed in before it can be processed by the word processor, but it is worth pointing out that the better the user is at typing, the faster the text can be entered. This brings out the importance of the 'proper' keyboard that is a feature of the Amstrad. It is comparable to the keyboards found on electric typewriters in its keys, the action of the keys and the comfortable angle at which the keys are presented. It will allow a trained typist to type just as quickly as on a typewriter, and it will also make it possible for the unskilled typist to reach a respectable standard quite quickly. But the untrained typist should, from the outset, try to learn to type properly. The 'hunt and peck' style of keying may be adequate for short sessions at the keyboard, but it will never be good enough for a word processing session of any length and, once acquired, it is a habit that is not easy to lose.

A further point that can be made is that, although we are probably accustomed to thinking of a document as something to be printed on paper, when using a word processor paper is not really needed, either for the preparation of a document or for its communication. Once a document is prepared it is more convenient to store it on a cassette or floppy disk than on paper. And because the computer can be linked to another machine by the telephone and the appropriate equipment, a document can be passed from one to another in electronic form. The receiving system can display the document on its screen and can store it, so that the document can be communicated without the need to commit it to paper at any stage. In this way, the word processor gives its users the ability to produce perfect documents and to communicate these documents as 'electronic mail' without the need to involve paper in the process at any stage.

Who can benefit from word processing?

Word processing makes the creation of documents easier in various ways, and these ways can benefit different kinds of users to different extents.

The speedy and straightforward production of accurate, even perfect, documents is one advantage of word processing that can benefit all its users. Any corrections and amendments can be

made before the document is finally printed. The appearance of a document can also be enhanced by using a word processor's ability to underline words, to emphasise words by printing them in heavier type than the others, to place headings centrally on a line and to arrange the columns of a table in alignment. This allows the production of documents of a quality and style that any user of word processing may consider to reflect his or her aspirations.

When typing a letter, its format can be altered after the text has been entered to ensure that it fits neatly on the page and does not require, for example, a final page consisting of perhaps a single line and a signature. The word processor can perform the rearrangement of the text automatically after being directed to make the line lengths in the document a little greater. All users of word processing will benefit from being able to make adjustments of this kind.

Word processing also brings a very real benefit to anyone producing letters that basically consist of standard sentences and paragraphs drawn from some fixed repertoire. The standard repertoire can be stored on a cassette or disk, and then any letter can be produced simply by recalling the paragraphs needed to make it up and placing them in the right order. There is no need to type each letter individually. Any small changes that may be needed in a particular letter can be effected by using the word processor's editing facilities. It is possible to go further, because names and addresses that have been stored on a cassette or disk can be recalled to address the letters.

The use of standard material to compose letters is much used in business by, for instance, lawyers when producing legal documents and estate agents advertising properties. It can be equally useful in the home for things like standard requests for advertising brochures and 'thankyou' letters.

The authors of books and magazine articles, and students writing essays, will be among those familiar with the process of creating polished text after amending and refining several hand-written draft versions. The process involves producing a first draft, and then proceeding to a second by crossing out parts, inserting others, and perhaps cutting out sections to move them by pasting them in somewhere else. When the whole becomes too untidy or just unreadable it must be rewritten. The entire process may then have to be repeated to give another draft. This is extremely time-consuming, and it leaves the writer high and dry if, having produced a further draft, he considers that it is worse than an earlier one. With a word processor, the amendment, revision and polishing of a document can be done much more

easily. There is never any need to retype a document as the word processor's editing functions ensure that, as one version is converted to the next, the new version is always neatly displayed. But the current version of the document can always be saved, so that if the result of revising it is not an improvement, then the new version can be discarded and the original recovered.

Anyone with a need to write polished text can benefit from this way of using a word processor. Those who have become accustomed to working with hand-written or typed drafts may take some time to adjust to the new ways of working that go with word processing, but the almost universal experience of those who have tried is that, once they have become accustomed to it, the use of a word processor is a much more convenient and rapid way to create text that expresses what they want to say.

A word processing session

This section gives a description of a typical word processing session with a view to showing what happens in practice. Word processing has its own terminology, and some of it will be introduced in the course of the description.

If you have some text that you want to turn into a document, you can sit down at the keyboard and type it as soon as your word processing program is running. Your text will appear on the screen as you type it. Nothing unexpected happens until a line on the screen becomes full. Then, the first word that makes the line of text too long to fit on the screen is automatically positioned at the beginning of the next line. This is known as *word wrap*. It should be stressed that there is no need to press the ENTER key at the end of each line of text as the equivalent of the carriage return on a typewriter; the word processor handles this itself. In fact, it is wrong to press ENTER for a line ending because it signifies something else as we shall see in a moment.

When a line is filled with text and a new line is started, the word processor also adjusts the line that has been filled so that its final word finishes exactly at the end of the line. In this way the document is given a neat vertical margin at the right as well as having one at the left which it would even when using a typewriter. This process is known as *justification*. It is achieved by inserting extra spaces between the words on a line to push the last letter across as far as is necessary. But the spaces are inserted with care: they are positioned in a symmetrical way so that their presence, while not drawing attention to itself, produces a pleasing appearance.

Continuing to type gives further lines, all of which are treated in the same way. The end of a paragraph is indicated by pressing

ENTER. In response to this, the word processor automatically creates the gap between paragraphs and makes the indentation for the beginning of the new paragraph. This is part of the word processor's document *formatting*: the positioning of the left and right margins and the line spacing are other aspects of it. At all times the document is displayed fully formatted on the screen as it is being entered. This shows the user the document that is being prepared in exactly the form in which it will be printed if it is committed to paper.

The word processor provides its users with a particular style of formatting by default. If the user prefers a different style, then any aspect of the format can be altered by giving the appropriate command to the word processor.

When sufficient text has been entered, the word processor will indicate that the end of a page has been reached and will start on a new one. The final document will be divided into these pages, which can be numbered, given a *header* and given a *footer*. A header is a line of text to be placed at the top of each page as with the running title that is found at the top of every page in many books. The footer, correspondingly, is a line of text to be placed at the bottom of each page.

If a word or phrase needs to be underlined or emboldened at any point in the text, this can be done by giving the appropriate direction to the word processor to cause it to use the required style. From then on all the characters that are entered will be displayed in this style until you indicate that you no longer want it or until you select another style.

When the entire document has been entered, or perhaps after typing only a certain amount of it, you will want to go back to check it. You can then ensure that you have typed everything correctly, and see whether there are any spelling mistakes, whether you want to make any changes and if the appearance of the document is to your liking. The process of correcting and amending a document is known as *editing*, and is one area where the word processor really shows to advantage over a typewriter.

If your document is shown on the screen in its entirety, small amendments, such as are needed to correct spelling mistakes or to make small insertions, can be made after using the cursor movement keys to move the cursor to the position on the screen where the change is needed.

Having done this, the letter immediately to the left of the cursor can be removed by pressing the key for deletion. A word can be deleted by positioning the cursor at its left and deleting its letters one by one. Text can be inserted at the position of the cursor, thereby adding it to the document, just by typing it. It is also

possible to make the text that is inserted replace, or over-type, the existing text. While any editing operation is in progress, the word processor re-formats the text at once to take account of the change.

If the document is too large to fit on the screen, the screen acts as a 'window' through which a part of the document can be seen, as shown in Figure 1.1. The screen window can be moved up and down the document in order to bring different parts of it into view. (Looking at it in another way, you could say that the document is scrolled up and down under the screen window.) Since a part of the document can be edited only if it is being displayed, a large document is edited by first bringing the part to be edited into the display window and then proceeding as before. To make substantial alterations to a document, such as deleting large blocks of text or moving a paragraph from one position to another, it is necessary to give the appropriate command to the word processor. Similarly, if the format of the document is not to your liking, it, too, can be changed after giving the necessary commands.

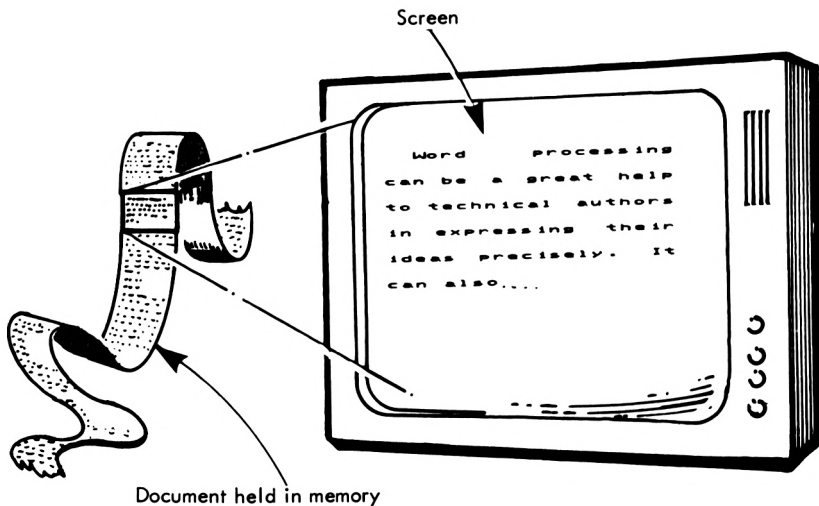


Figure 1.1 The word processor's screen as a 'window' on a document

The display maintained on the screen by a word processor usually shows not only the formatted text but also a certain amount of information about it. It shows the page and the line on which the cursor is situated so that you can always keep track of your position in the document, even during the most tortuous editing session. The position of the cursor along the current line is indicated on the *ruler* line. The ruler appears immediately above

or below the displayed text, and also shows the positions of the left and right margins and of the tab stops. The number of words in the text may also be shown.

When you have edited your document and are satisfied with its state, you will want either to save it on a cassette or disk, or to print it. Either can be done by giving the associated command. Before giving any command it is first necessary to enter a special mode so that the word processor will know that you are issuing commands rather than typing text for it. On entering the command mode, a *menu* of commands is often displayed from which one may be selected. The procedure for saving or printing a document, then, is to enter command mode to display the required command, and to issue the command for saving or printing. The commands may require you to give more information, such as the name of the file in which the document is to be saved, but you will be prompted for any information that you should give.

The range of commands possessed by a particular word processor gives a very good indication of its general power and capabilities. You will find that you do not usually need all of them. The commonly used ones, such as those we have mentioned and a few others, will be sufficient for most of the time. But it is a good idea to be aware of all the commands, for if you are not, you may not appreciate that your word processor is quite capable of fulfilling some unusual task, particularly when the need for it first arises.

Databases

A database is an organised collection of data. It is also something more than this, because data that is collected simply for the sake of it has no particular value: it must be possible to make some use of it. This means that the contents of a database must be organised so that we can recover them to make use of them and, what is more, recover them in any of the ways that we may need to. Operations typical of those that we may want to perform to make use of a collection of data are to select particular items from it, to search it for any items, or collections of items, that meet a specified condition, to update some of the items in it, and to sort the items into a special order.

The data to be placed in a database can be organised, essentially, by ensuring that all the entries are structured in the same fashion when they are placed in it. This structure should reflect, and even typify, the natural structure of the items that are recorded as entries. Using a standard structure ensures that the entries in a database all record instances of the same sort of item.

This gives a coherence to the database not only in the way that it records items but also in the type of items that it records. A well thought out structure will also serve to avoid any redundancy and duplication in the data that is stored in the database.

A database program can be of benefit to just about anyone, for we all have a need to store and retrieve information of some kind. Typical uses at the personal level range from cataloguing the details of a stamp collection or an accumulation of recipes, through keeping a record of the contents of the freezer, to keeping the details of all the financial transactions relevant to one's annual tax return and running an appointments diary. Activities of a kind usually associated with business, such as keeping customer records, maintaining address lists and stock control are all examples of applications that can usefully be adapted for use in the home, and not just for running a business from home. Address lists will be useful when sending the cards at Christmas; a record of the personal details of friends and relations can be used as a much more convenient replacement for an address book, and to ensure that birthdays are not forgotten; a small stock control program could automatically produce the shopping list each week.

When databases are used for these kinds of purposes, the ways in which the items in the database may need to be accessed will be broadly the same in all cases. When using a database to keep the details of a collection of recipes, a typical requirement might be to display the details of a particular recipe, or to find all the recipes for a beef dish for two people. A database holding data for an appointments diary will need to be able to locate a date and to display any appointments for that day. A database that is used for a tax return ought to be able to perform arithmetic operations on some of the entries, such as adding up all the items of income, for example. When used to keep a record of the contents of the freezer, it will be necessary to update the quantities held. If the levels at which items should be re-stocked are also held in the database, then re-ordering can take place automatically and without fail whenever it becomes necessary.

A database of names and addresses can be used to print its information on envelopes to address them. If the contents of a database containing information about friends and relatives can be passed to a word processing program then it can be added to a standard letter so that individual personalised letters can be produced for each one. By recording other information about each individual in the database it will be possible to send letters to a particular group of them. In this way letters can be sent to only the children, for example, by making the database perform the

task for an individual only if his or her age is below a certain figure.

From this discussion, we can see that a database program has two distinct parts. The first allows the data to be entered. The second allows the user to access the data in any way that may be required. In the entry phase, the program should allow the user to structure the entries in a way that is suited to the application. When it comes to making use of the data in the second phase, the program should provide for all the operations that the user is likely to require. These include, as we have seen, selection, searching and sorting. The database program should provide commands for all of these operations. In addition, if it is to be thoroughly useful, it should allow its users to modify these commands and even to design their own commands, for even the most ambitious program cannot anticipate every need that may occur. In general, we can say that the fewer restrictions a database program places on its users in the ways that they enter and retrieve data, the better it is.

As the final point in this introduction to databases, we can illustrate how a database program can give a considerable advantage over a collection of data presented in conventional fashion. Almost every house in the country contains a database, although it is printed on paper, in the form of the details of the week's television and radio programmes. This is a database in the sense that it is an organised, coherent collection of data about television and radio programmes. From the printed version, it is easy to find the programme that is on at a particular time, not quite so easy to find at what time a particular programme will be shown, and even less easy to find what programmes of a particular kind will be shown and when they will be on. But if the database were stored in a computer rather than being printed on paper, any query could be answered with equal ease. It would only be necessary to enter the query, no matter what it was, and the answer would appear straight away.

A card index and a database

By examining how information is stored in a card index system and subsequently recovered from it, we can begin to see how a database program operates because it is used in an entirely analogous fashion. Using a computer, rather than a card box full of file cards, to store the database makes it much quicker and more convenient to recover the information, but the principles underlying the use of both systems are the same.

To deal with a concrete example, suppose that a family keeps records showing details relating to the health of each child in the

family, and that for each child it is decided to record the following items of information: the child's first name, sex and age, the diseases that the child has had, the inoculations that he or she had, any forthcoming appointments, and finally any comments that may be relevant. For this purpose a file card such as the one shown in Figure 1.2 could be designed. Each card would carry the details of one child and, depending on the size of the family, the card box would contain a larger or smaller number of cards.

NAME	Anne	
AGE	10	
SEX	f	
DISEASES	Measles	Chicken Pox
	Mumps	
INOCULATIONS	Smallpox	Whooping Cough
APPOINTMENTS	17 May	
REMARKS	Health is good	

Figure 1.2 A file card to hold a child's medical record

The information to be stored about each child has been structured by deciding carefully which details it is important to record. All families will have different ideas about this, and it is clear that other items can be placed on the card just as easily as the ones that have been decided on, and that any item can be crossed out if it is no longer important. But time spent on getting the layout of the card right in the first place will be well spent. It is also obvious that a file card carrying the details of a stamp in a

stamp collection would not be kept in the same card box as the file cards containing a child's medical details. It would be placed in a separate box kept for the stamp collection and which would be full of cards each carrying the details of a stamp. The situation when a number of card boxes have each been filled with their own type of cards can be represented as shown in Figure 1.3.

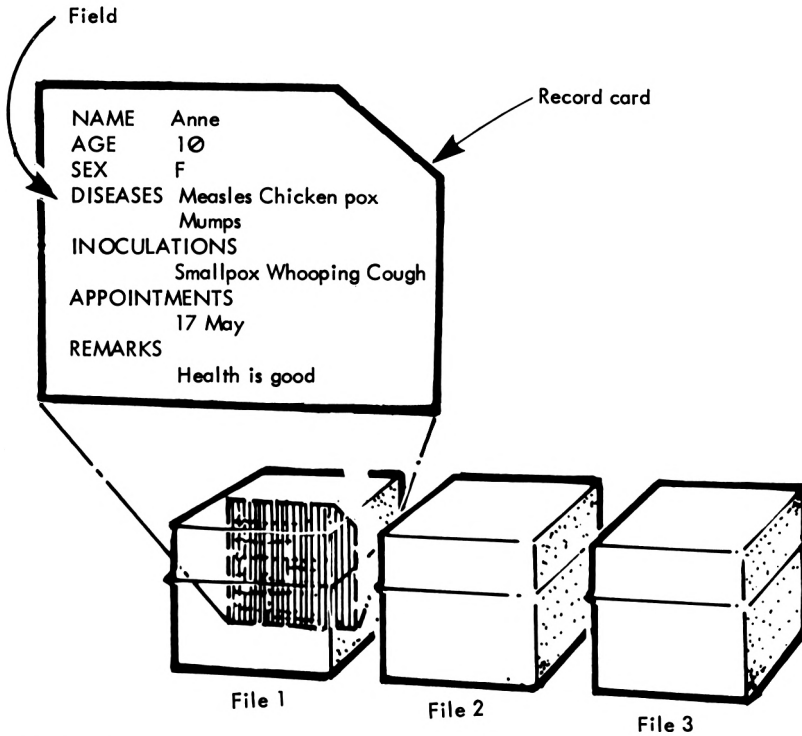


Figure 1.3 Several files, each containing record cards

Once the details of each child have been entered on a card, and all the cards have been placed in their box, the data entry stage has been completed and the database has been created. It can then be examined to recover information about children's medical details as it may be needed. The children who have had mumps, for example, can be found by flicking through all the cards in the box and noting the 'first name' on those for which the entry against 'diseases' includes 'mumps'. The girls with medical appointments in the next week can be found by flicking through the cards and noting those on which the entry against 'sex' is 'female' and the entry under 'appointments' contains a date between those of the beginning and the end of the week. Any child with a particular allergy could be found by searching for the

cards with appropriate remarks under 'remarks'. The family could decide to sort the cards into order using the alphabetical order of the first names to order them, or perhaps using the decreasing order of the ages to order them with, as a secondary consideration, females preceding males of the same age.

These operations will take some time involving, as they do, scanning all the cards. The sorting will be particularly lengthy as, besides scanning, it involves re-ordering. Although a computer database will operate in essentially the same way when it carries out these operations, the computer's speed ensures that the operations are performed much more quickly. In addition, the computer is doing all the work instead of us. (It can't drop the cards, either.)

When using a database program, the program first allows us, as the equivalent of designing a file card, to design the format of a *record*. When the format has been designed records can be entered, and this corresponds to filling in filing cards. Each item of information in a record is known as a *field* of a record, and is analogous to one of the entries on a card. When a record is entered it is stored in the computer's memory. A collection of such records is known as a *file*. With reference to Figure 1.3, a file is obviously the equivalent of a box of cards. And just as we can have a box full of cards that carry the details of one child's medical record and another box full of cards each carrying the details of a stamp, so a database program can be used to create various files each of which contains records designed specially for it. When a file has been created it can be permanently stored on a cassette or disk.

Spreadsheets

By running a spreadsheet program the computer provides its users with the electronic equivalent of a pencil, sheets of paper and a calculator for preparing tables and performing calculations on their numerical entries. The entries in the table can be numbers, text to provide headings and labels, and calculations involving the numbers in the table. After altering one value or another in a table, the effects of the changes will automatically be taken into account in the calculations and the new values displayed. In this sense the spreadsheet is an 'electronic worksheet'. Its advantages when compared to pencil and paper working are those of speed and convenience, plus the abilities to handle large amounts of data, and to perform calculations on this data with ease. By allowing the rapid and direct investigation of any alterations that are possible in a given situation, but

particularly in a complex one, the spreadsheet becomes an ideal tool for forecasting and planning.

The user of a spreadsheet program is presented initially with a blank sheet. The sheet is structured as a table, providing a large number of positions that are arranged in rows and columns. Each position is known as a cell, and an entry is placed in a cell simply by moving the spreadsheet's cursor onto that cell, typing the entry and pressing ENTER when it is complete. The entries themselves, as we have seen, can be numbers, text or a description of a calculation involving other numbers in the table. But, just by placing numbers and text in the appropriate cells, a table such as the one shown as Table 1.1 can be prepared. The headings and labels for the table will be entered as text, and the entries in the body of the table as numbers.

Table 1.1 Family budget for one quarter

	<i>Family budget (£)</i>		<i>Expenditure</i>
	<i>Income</i>	<i>Savings</i>	
Jan.	200	120	Water
Feb.	200	180	Electricity
Mar.	200	140	Gas
		100	Rates
Totals	600	540	
Balance	60		

The rapid and convenient way in which a table can be prepared only begins to hint at the spreadsheet's possible usefulness. Its ability to hold a formula describing a calculation in any of its cells is one factor that helps to make it such a useful tool. The formula will make reference to the contents of other cells in the spreadsheet but, when a formula is associated with a cell, the spreadsheet does not display the formula itself in the cell, it displays the *value* of the formula. This can be a help even in preparing a table as small as the one in Table 1.1, where the totals are the sums of the numbers in the columns above them, and the balance is the difference of the two totals. With a large table it is invaluable.

To prepare Table 1.1 by using formulae, each time the cursor is positioned on a cell where a total is to be displayed, we can enter a formula such as 'sum of the numbers in the column above'. For the balance, we require a formula such as 'total of the savings - total of the bills'. Of course, in practice a spreadsheet will have its

own way of expressing such formulae, and they will be more compact than the forms we have just given.

Table 1.1 could be prepared just by entering numbers and text, but this gives a once-and-for-all table that serves its purpose, but only that one purpose. If as many entries as possible are formulae then we have a much more flexible table, because when a number in the sheet is changed, the values of the formulae change correspondingly. No matter what figures are placed in the table for Savings and Bills, the spreadsheet automatically calculates the figures for the Totals and Balance.

All spreadsheets have the ability to update automatically the value displayed in a cell with which a formula is associated whenever a change is made to any of the cells involved in a formula. This means not only that the calculations are left to the spreadsheet, which will do them more reliably than we can, but also that all the consequences of a change are calculated without fail to be displayed at once.

When Table 1.1 is prepared with the fullest use of formulae, it can be used to present any quarterly budget. All that is necessary is to place the figures for the Savings and Bills in their two columns. In this way, it provides a general model for a quarterly budget.

Creating Table 1.1 with the help of formulae, also allows it to be used to investigate the effects of different figures in the columns where numbers, or data, are entered. If the amounts of the bills were to rise, for example, it could show the resulting balance, or could be used to find the savings that would be necessary to maintain the balance at the same level. This illustrates, in a small way, that a spreadsheet is an ideal vehicle for investigating the effects of changing data values and so for planning. It will provide immediate responses to the 'what if?' type of questions that often need to be answered when planning ahead to prepare budgets and forecasts.

In fact, with a spreadsheet it is possible to create a model of any of a wide range of things from the household finances through some proposed enterprise to an investment portfolio. Their possible future behaviour can then be examined quantitatively, and sensible plans for the future can be prepared with this analysis in mind. The creation of a model requires a decision about which entries are to be provided initially as data, but its essence lies in deriving the formulae to represent the situation.

In modelling any complex operation, the spreadsheet will have to support a very large table, and spreadsheets are quite capable of this. The blank sheet that is provided is much too large to be displayed on the screen in its entirety. The screen acts as a

'window' on the sheet through which a part of it may be seen in the way illustrated in Figure 1.4. The part of the sheet that cannot be seen is still safely stored in the computer's memory. At any time the window can be moved over the sheet to allow another part of it to be viewed.

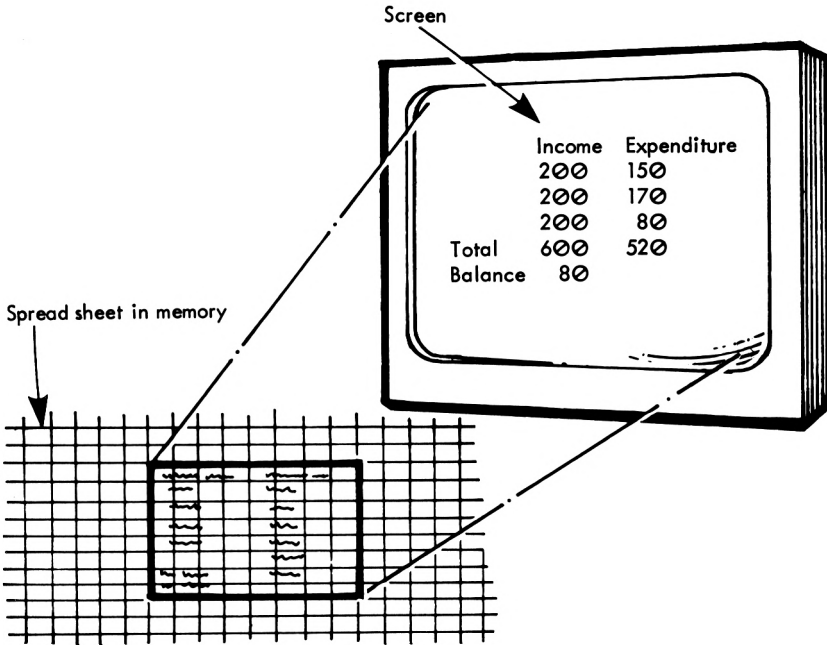


Figure 1.4 The screen as a 'window' on a spreadsheet

To summarise, a spreadsheet program maintains a table and can accept numbers, text and formulae for display at any position in the table. Numbers and text are displayed as they are entered, but the value of a formula is displayed. The table can be, and often is, much too large to display on the screen in its entirety. In this case the screen acts as a 'window' on the table. A spreadsheet has the ability to update the value of any formula it contains as soon as a new value is placed in one of the cells involved in the formula. This makes a spreadsheet an ideal tool for planning and forecasting because it allows all kinds of 'what if?' investigations to be carried out just by changing the numbers in the table and observing the effects of the changes.

Examples of using a spreadsheet

In this section two examples of how to use a spreadsheet are given. They are still relatively small-scale examples because it is not practical to develop large-scale ones in a book. A spreadsheet naturally comes into its own in a large application with a

correspondingly large table. But these examples serve to illustrate the sort of applications in which spreadsheets are valuable as well as the basic ideas behind their use.

The first example is the creation of the sheet for the quarterly budget that was shown in Table 1.1. This table can be entered, as we have explained, as text and numbers only. The entries in the columns headed Savings and Bills are the basic data, and will be entered as numbers, but the other numeric entries depend on those in these two columns, and will be entered as formulae.

	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					
8					

Figure 1.5 The initial sheet of a spreadsheet

When a spreadsheet program presents its initial blank sheet, the columns are labelled with letters and the rows with numbers, as shown in Figure 1.5. One of the cells in the sheet is identified by giving its column letter and row number, so that the cell at the top left of the sheet is cell A1, and the cell immediately to its right is cell B1, for example. If we choose a position for our budget table, such as the one shown below, we can go on to describe how to enter it.

	A	B	C	D	E
1			Family budget		
2					
3		Income		Expenditure	
4					
5		Savings		Bills	
6	Jan	200		120	Water
7	Feb	200		180	Electricity
8	Mar	200		140	Gas
9				100	Rates
10					
11	Totals	600		540	
12					
13	Balance	60			

The entries can be made by moving the cursor to the appropriate cell and typing the number or text for that cell, except that for cells B11, B13 and D11 we shall enter formulae. To make the entries in column A, we first move the cursor to cell A6, using the cursor movement keys, type 'Jan' and press ENTER; then the cursor is moved down one place to cell A7 and 'Feb' is typed followed by ENTER; and so on for the rest of the column. All text and numbers can be entered in the same way.

The formula to be entered in cell B11, giving the total savings, is for the sum of the numbers in cells B6 to B8. This is expressed by the formula $B6 + B7 + B8$. This formula can be entered by moving the cursor to cell B11, typing the formula and pressing ENTER. The figure for the total savings will then automatically be displayed in this cell. The formula for the total amount of the bills will be similar to this one. In fact, the formula for cell D11 is $D6 + D7 + D8 + D9$. The balance is to be displayed in cell B13, and as this is the difference between the total savings and the total bills, the formula for this cell is $B11 - D11$.

We can place a formula rather than a number in two more cells, for if the savings are to be the same in each month, we need only enter the amount once, and then the spreadsheet can copy it to the other cells. To do this, we can enter the monthly amount as a figure in cell B6 (for January), and the copy it for the next two months by placing the formula B6 in both cell B7 and cell B8.

Having developed this sheet using as many formulae as possible, it provides a model on which other quarterly budget calculations can be based. For example, if the gas and electricity bills increase, then the monthly savings that are required to maintain the same balance can be found by entering the new figures for the gas bill in cell D8 and the electricity bill in cell D7 and then trying new figures for the monthly saving in cell B6 until the spreadsheet shows the required figure for the balance.

The second example concerns an investment portfolio, and anyone with shares in several companies will be interested in a worksheet such as the one shown below as Table 1.2.

Table 1.2 An investment portfolio

	A	B	C	D	E	F
1	Company Holding	Price (p)	Value (£)	Cost (p)	Gain (£)	
2	BT	400	141	564	50	364
3	Bejam	200	168	336	148	40
4	Lloyds	100	522	522	446	76
5						
6			Total	1422		480

It shows the companies in which we hold shares, the number of shares we have in each (Holding), the currently quoted price per share in pence (Price), the value of our holding (Value), the original cost of the shares in pence (Cost) and, most importantly, the Gain in value of the shares.

The holdings and prices must be entered as numbers, but the value of the holding in a company is calculated by multiplying the number in the holding by the price per share and dividing by a hundred to convert it to pounds. The necessary formulae for cells D2 to D6 are $B2 \star C2 / 100$, $B3 \star C3 / 100$ and $B4 \star C4 / 100$. The total is obtained by placing the formula $D2 + D3 + D4$ in cell D6.

It is a common occurrence when creating a worksheet to find columns or rows of entries that are identical or, as with the formulae for the values, show a simple repetitive pattern. For this reason spreadsheets in general provide facilities that allow entries to be repeated systematically in the cells of a column or a row. Figures and text can be repeated in the same form while formulae can be repeated in a pattern such as the one required in our example. Following on from this, there is a need to be able to specify a range of cells in a column or in a row, in part so that it is possible to say over which cells an entry is to be repeated. The range from C4 to C6, which is where we would like to repeat our formula, is written as C4:C6.

When the price of a share changes, we can enter the new value in column C, replacing the old one, and its consequences will be reflected at once by corresponding adjustments in column D to the value of the holding in that company and to the total.

The original costs must be entered as numbers in column E, but the gains can be found from formulae. The gains can be calculated as their current price less their original cost multiplied by the number of shares in the holding, and divided by a hundred to convert to pounds. This gives the formulae that we need for cells F2 to F4 as $(C2-E2) \star B2 / 100$, $(C3-E3) \star B3 / 100$ and $(C4-E4) \star B4 / 100$. Again, there is a clear pattern to the formulae. After this, the total gain can be shown in cell F6 by placing the formula $F2 + F3 + F4$ in that cell.

Variations on this sheet can be made in many ways. For example, if shares in, say, BT are bought at different times and at different costs, a separate row can be used for each transaction. But the current price, which will be the same regardless of when they were acquired, need only be entered once as it can be copied to other locations. When shares are sold, it is only necessary to remove a row from the sheet. The table that has been created provides a model from which any portfolio can be evaluated. When the worksheet has been prepared it provides an ideal basis

for making decisions about buying and selling shares because, by taking a view about the way that share prices may move, the possible future situations can be examined to see which gives the best return.

Other applications

In this section we shall describe briefly a few of the other applications for which software is available and which will be of value in the home.

Programs for *graphics* can be valuable, particularly in trying to make sense of sets of numbers. They allow numbers to be typed in, as you would type text into a word processor, and then display the numbers graphically in ways which you can choose. Line graphs, bar charts, histograms and pie charts are among the commonly used forms of display, and each will come into its own in different circumstances. A pie chart is an effective way of showing relative proportions and a chart with horizontal bars can give a good impression of the sizes of the numbers in a set as well as of their relative sizes. A good graphics program will display the numbers it is given in any of a number of ways. This allows you to try different forms of display and then to select the one that suits you best.

You may have given the computer the numbers that you want to display already, perhaps by entering them in a database or a spreadsheet. It seems unnecessary to type the numbers again simply to display them graphically. This brings us to a topic the jargon word for which is *integration*. An integrated package is a group of programs that is provided with a uniform environment which allows the programs to exchange information with each other. The uniform environment ensures that the programs are used in the same way, as far as is possible, so that by learning to use one you learn something about the others. Because the programs can exchange information, a set of numbers that is given to one of the programs can be passed from it to another.

A word processor, a database, a spreadsheet and a graphics program are commonly to be found as the components of an integrated suite of programs. The component programs may not be as powerful as their individual, non-integrated, equivalents, but in some circumstances integration may be more important than having sophisticated features. Integration permits a set of numbers from a column in a spreadsheet to be passed directly to a graphics program for display. It would also allow a set of numbers to be taken from a database, passed to the graphics program for display, from where the graph or chart could be passed to the word processor for incorporation in a document.

A *program generator* is a computer program for writing programs. It allows you to describe to it in fairly non-technical terms what you want the computer to do, and then it writes a corresponding program in the language of the computer instructing the computer in how to perform the task in question. In effect, a program generator allows you to program the computer without being an expert computer programmer. If you cannot find the software to make the computer do what you want of it, then a program generator might be the way for you to create your own software for the task.

Finally, 'expert systems' are available at a price at which they can be considered for use in the home. All the other programs we have mentioned deal with data and information: an expert system deals with knowledge. The knowledge is extracted from an expert, and the expert system can process that knowledge to deliver advice and assistance of a standard equivalent to that of an expert. It is not difficult to see that an expert system can be invaluable in applications such as exploring for oil or diagnosing illnesses, but there are occasions in the home when advice can be sorely needed. The problems of bringing up a baby suggest themselves at once, and there must be many others ranging from advising with plumbing problems to helping to plan interesting but nutritional meals. Expert systems complete with knowledge about your particular problem may not be available, but it is possible to buy an 'expert system shell', which is an expert system containing no knowledge, to which you can add your own knowledge of a particular topic, after which it can offer advice on this subject to its users.

Summary

Word processors, databases and spreadsheets are the most commonly used of the programs that can allow the Amstrad computers to be used to good effect. This chapter gives general descriptions of the applications, orthodox and otherwise, of these types of programs, and describes the full range of facilities that they should possess. In this way, the chapter provides a yardstick against which the specific programs dealt with in the next chapter can be measured. It may be that these programs have all the features that you will ever need, or it may be that they lack a feature that is vital to the particular application that you have in mind. In either event, this chapter provides the criteria by which particular programs can be judged.

Other kinds of programs that have valuable applications, including graphics programs and expert systems, are also described, although more briefly.

The Amstrad software 2

This chapter is concerned with using software for the Amstrad of the kind that was described in the previous chapter. This software allows the Amstrad to be used for serious computing. The programs that we shall examine in some detail are:

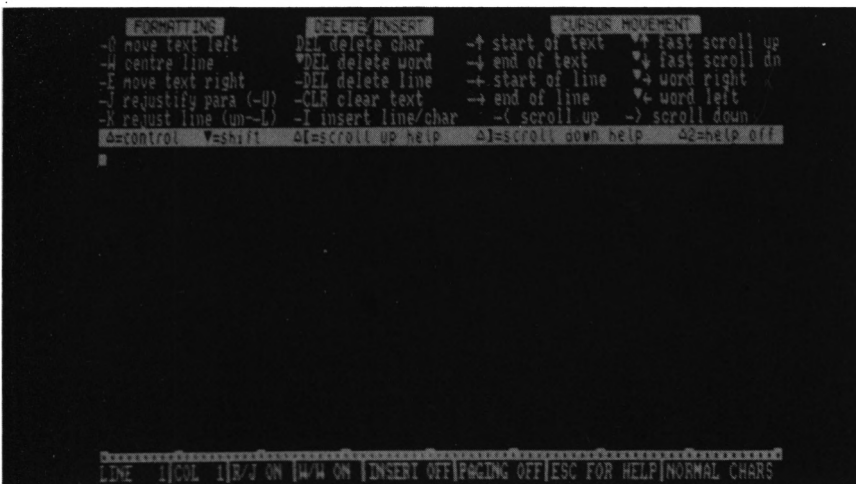
Amsword — the word processor by Tasman Software Ltd that is available from Amsoft.

Database — the database from Gemini Marketing Ltd.

Easi-Amscalc — the spreadsheet by Saxon Computing that is available from Amsoft.

Word processing with Amsword

The Amstrad is turned into a word processor by running the word processing program Amsword. The program is recorded on cassette and, when it is run, it gives the initial display shown in Figure 2.1. From this point, the user familiar with word processing can proceed to explore Amsword's capabilities.



```

      FORMATTING          DELETES/INSERT          CURSOR MOVEMENT
-0 move text left      DEL delete char      ← start of text      ↑ fast scroll up
-H centre line        ▽DEL delete word    → end of text        ↓ fast scroll dn
-R move text right    -DEL delete line    ↑ start of line      ↗ word right
-J rejustify para (-U) -CLR clear text     ↓ end of line        ↖ word left
-K rejust line (un-L) -I insert line/char  -( scroll up         -) scroll down
Δ=control  ▵=shift    Δ|=scroll up help  Δ|=scroll down help  Δ2=help off

```

LINE 1/COL 1/B/1 ON /W/W ON |INSERT OFF|PAGING OFF|ESC FOR HELP|NORMAL CHARS

Figure 2.1 Amsword's initial screen

The display is divided into three parts. The largest, central, area which is initially blank is for the display of the formatted text. The area at the top contains reminders about how to use Amsword. The bottom part shows the ruler and a line providing information about the state of the document.

The top part of the display is part of Amsword's facility for helping its users by providing the basic information about how to enter, edit and format text, how to control an attached printer, and, generally, how to make it do all the things that it is capable of doing. This display is only a part of the help that is available, which can be seen in its entirety by pressing the ESC (escape) key. This gives the display shown in Figure 2.2. The word processor's activities are all initiated by holding down either the green CTRL (control) key or a SHIFT key and pressing one of the letter or number keys at the same time. On the screen, and in the documentation, the CTRL key is represented by an upward pointing arrowhead filled in white and the SHIFT key by a downward pointing arrowhead filled with black. These key combinations are used so that the word processor can distinguish a command for it to perform some activity from text that is being entered.



Figure 2.2 Amsword's help screen

It is necessary to press ENTER to return to the normal display from the help screen. After seeing the complete help screen, it is

clear that the normal display shows only a small part of it, but different parts can be scrolled onto the screen by pressing CTRL and [or CTRL and] so that the part of the help screen containing the information that is required at any time can always be brought into view.

On becoming familiar with the functions of Amsword, it is no longer necessary to have the reminders in the upper part of the screen all the time, and they can be removed by pressing CTRL and 2. The part of the screen that was occupied by the reminders is then taken into the part for the document display, making it correspondingly larger. Figure 2.3 shows the screen when the reminders have been removed and the text of a document has been typed into Amsword in the way that we described earlier. The help area can always be displayed again by pressing CTRL and 1.

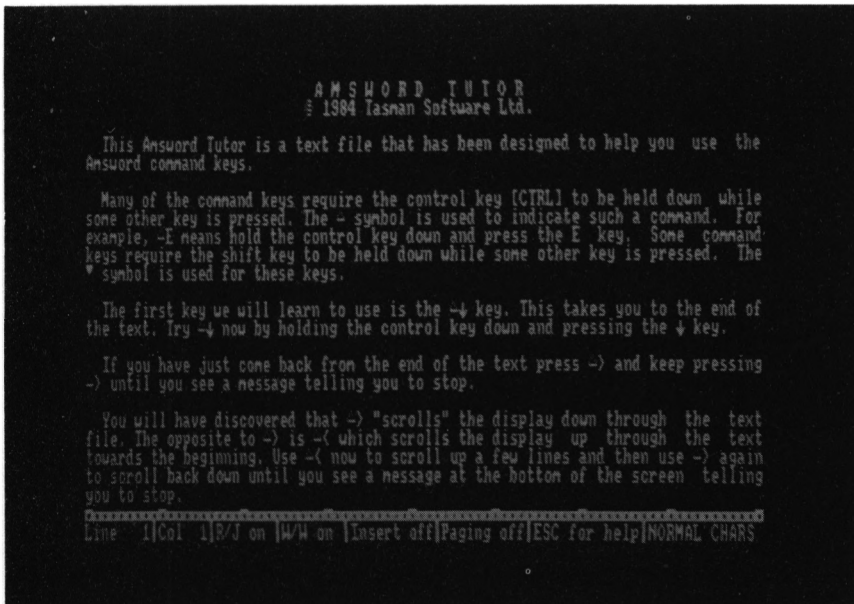


Figure 2.3 Amsword, without its helpful reminders, displaying a document

In the lower part of the screen, the ruler is one line from the bottom, and it shows the positions of the margins and the tab stops. The bottom line itself contains a surprising amount of information, including the position in the document of the cursor, whether right justification and word wrap are active, the form in which the characters that are being entered will be displayed, and more.

The power of Amsword comes from its repertoire of com-

mands. As mentioned earlier, you are unlikely to need them all in the course of its general use, but you can only acquire a complete appreciation of its capabilities if you are aware of them all. The commands are all displayed and explained on the help screen, as shown in Figure 2.2, although the explanations are necessarily rather abbreviated. In the next section, we give expanded explanations of the more commonly used commands.

The commands

This section groups together the most frequently used Amsword commands and explains their purposes. The commands are grouped together by function to show the way that Amsword corresponds with the general account of word processing given in the previous chapter.

Text entry. Text is entered simply by typing it. Word wrap can be turned on or off by CTRL and G. The positions for the margins can be set after first moving the cursor to the required position. After this, pressing CTRL and A sets the left margin so that the cursor is immediately inside it; the right margin is set in a similar way but by using CTRL and D. The end of a paragraph is indicated by pressing ENTER. The indentation for a new paragraph is achieved by the use of TAB. The tab stops are, by default, at every tenth column. A new stop can be created by placing the cursor at the required position and pressing CTRL and TAB.

Editing. The cursor movement keys can be used to move the cursor around in a document. By themselves, the keys move the cursor by one position in the direction indicated on the key. With CTRL they cause it to move to the edges of the document, that is, to the top or the bottom and to either end of a line. With SHIFT they cause it to move by one word along a line or by one screen up and down. Once in the required position, typing causes the new text to replace, or over-type, the existing text. Deletion is achieved with DEL: by itself, DEL deletes the character to the left of the cursor, CTRL and DEL delete the word on which the cursor is positioned, and SHIFT and DEL delete the entire line. Text can be inserted after pressing CTRL and I.

The replace function, initiated by CTRL and R, allows all the occurrences of a word in a document either to be found or to be found and replaced by another word.

Formatting and style. Right justification can be turned on and off with CTRL and F. The current line can be centred with CTRL and W. A paragraph can be re-justified, perhaps after editing it or

turning off the justification, by placing the cursor on the first line of the paragraph and pressing CTRL and J.

The position at which the end of each page will occur when a document is printed can be indicated on the screen by a dashed line. These indications can be introduced or removed by pressing CTRL and P. A header to be printed at the top of each page can be created by typing it on a single line and pressing CTRL and 6. A footer is created in similar fashion, but in this case CTRL and 7 must be pressed.

Special styles, such as underlining and emboldening, are invoked by pressing CTRL and the space bar. Characters with underlining and other styles do not appear on the screen. Instead, special inverse characters appear before and after the text that is to be treated in the specified way. The J key is used to signify that characters will be underlined when the document is printed, and cause an inverse J to precede and to follow the text in question. Similarly, the letter A indicates emboldening and I italic.

Printing and saving text. When a document has been created, it can be printed or saved after pressing CTRL and ENTER. This causes a menu of options to be displayed, with printing the document and saving it among the choices. Pressing P to select the printing option, followed by ENTER to confirm it, gives the display shown in Figure 2.4, and then pressing COPY will cause the document to be printed on a printer attached to the Amstrad.

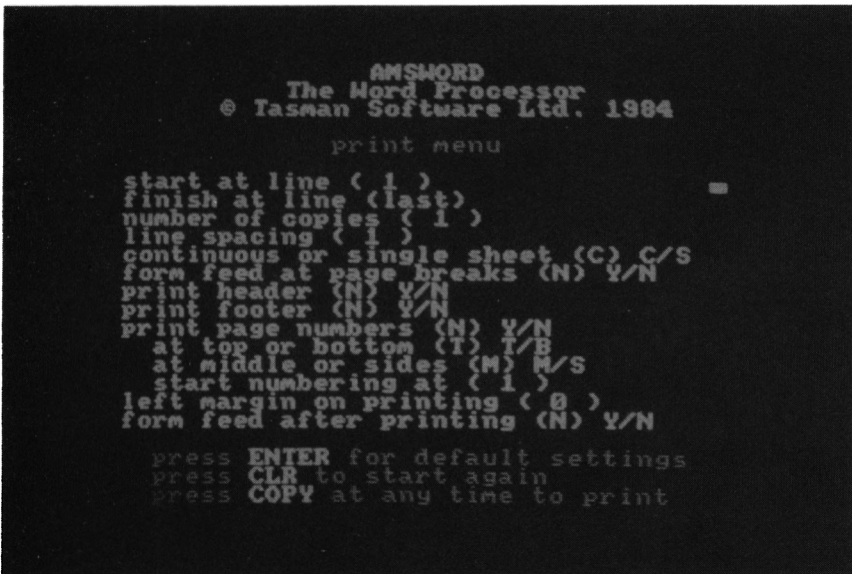


Figure 2.4 The options when printing a document

An examination of Figure 2.4 will show that there are many ways in which to affect the way in which the document is printed, and any of them can be chosen as they may be required before pressing COPY. Pressing S followed by ENTER when the menu is displayed initiates the sequence of actions, each of which is prompted on the screen, for saving the document.

A number of commands have not been mentioned here, but those that have should be sufficient to convey an impression of the capabilities of Amsword and to show that it is a thoroughly useful program that should meet the needs of anyone who wants to create text. In common with any good word processor, it encourages its users to enter their ideas as quickly as possible so that they, and their sequence, are not forgotten. The editing facilities make it so easy to go back to correct, polish and finish the text that one need have no inhibitions about the way that the text is entered in the first place as long as it is possible to understand it.

Database

We can make the Amstrad create and manage a database by running Gemini's program called 'Database'. When the program has been loaded from its cassette, it displays the initial screen shown in Figure 2.5. This is the program's main menu, and any activity from this card-oriented database is initiated by selecting it from this menu.

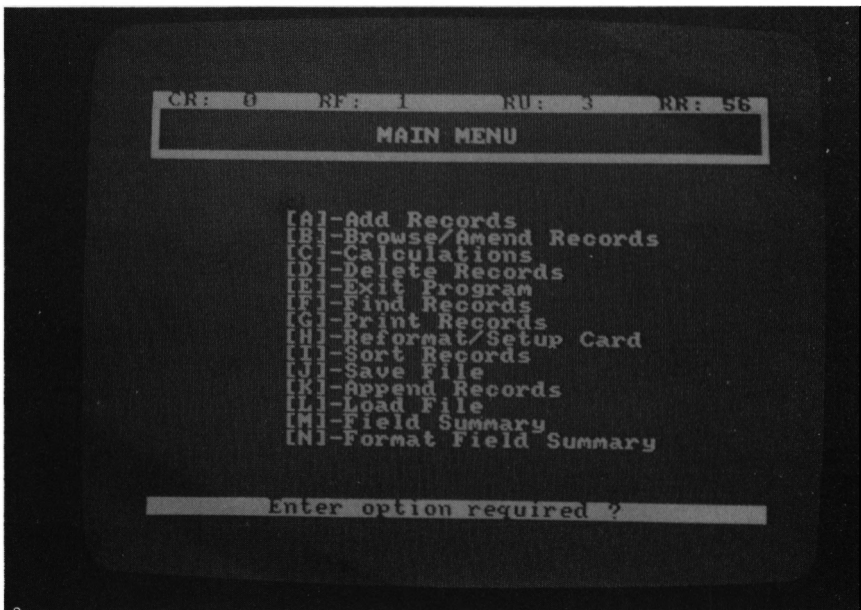


Figure 2.5 The initial screen of Database

To show how the program is used, starting from scratch, first to create and then to examine a database, we shall go through all the steps as they are needed for the database of children's medical records that we described in the previous chapter. We must begin by designing the 'card' on which the records in the database are entered. This is done by selecting option H from the main menu. This option presents us with a blank screen as the equivalent of a sheet of paper on which to design the card. Text and graphics can be typed anywhere on the screen to label the information and create an acceptable appearance for the card. Then the COPY key is used to indicate the positions in which the item of information for each field of a record is to be displayed. Each press of COPY creates a space that can be occupied by a character, and is indicated on the screen by a white rectangle. When the design has been created satisfactorily, with all the labelling and the gaps for the information, pressing TAB causes the database to ask for a name for each of the fields in the record, starting with the one at the top of the screen and progressing to the one at the bottom. It is sensible to use names that are similar to the labels for each field, and in this case the names were chosen as:

Field 1: Name

Field 2: Age

Field 3: Sex

Field 4: Diseases

Field 5: Inoculations

Field 6: Appointments

Field 7: Remarks

When all these names have been entered, the program displays the names and asks if they are correct. Pressing Y causes them to be accepted, while N gives you the opportunity to correct them. When the design and names for the fields are confirmed as satisfactory, the program accepts them and returns to the display of its main menu.

The next step is to enter some records, and this can be done after selecting option A. This causes a blank card to be displayed, and a record can be 'written' on it simply by typing the entries and pressing ENTER at the end of each. When a card is filled in, another blank card is displayed. A typical record is shown in Figure 2.6. The entry of records can be continued for as long as necessary; it is halted at any time by pressing TAB, which will cause the currently displayed record to be abandoned while all the ones entered previously are retained.

```

-----
                                RECORD 2
-----

First name: Anne
Age:      10
Sex:      f
Diseases:  Measles Chicken pox Mumps
Innoculations:  Smallpox Whooping cough
Forthcoming appointments: 17 May
Remarks:  Health is good

```

Figure 2.6 A record created and displayed with Database

Now that some records have been entered, the display in the top line above the main menu becomes useful. It tells us how many records we have entered against the initials RU (they actually stand for 'records used'). Against RR it shows the 'records remaining' unused in the file, so that we know how many more can be entered. The initials CR stand for the 'current record', and when a record is displayed its number will be shown here. The main thing that the number of a record tells us is its position in the file: the first record in a file is record number one, the next is number two, and so on. The fourth set of initials is RF. This stands for 'records found': it comes into play when a file is searched, and shows the number of records that were found when the file was searched for a particular type of record.

With a file in place, we can browse through it using option B from the main menu. The cursor-down and cursor-up keys, respectively, let us move towards the end or towards the beginning of the file. The number of the displayed record appears against CR at all times. The currently displayed record can be printed, amended or deleted just by pressing the appropriate key.

We cannot do anything creative in terms of examining the file simply by browsing through it, but we can with the Sort and Find options. Option I is for sorting the file, and when selected it gives us a display of the fields and their names, as listed above, and asks which field is to be used as the basis of the sorting. If the field contains words, the records are sorted so that the entries in this field appear in alphabetical order as in a dictionary. If it contains numbers, the records are sorted so that the numbers in the field are ordered from the lowest to the highest. When the sorting is complete, the program returns to the display of its main menu, after which a browse through the file will confirm that it has been sorted.

Option F allows us to search the file for any records meeting a particular criterion. We have to enter the criterion in a stylised

form, but it is not difficult to do this. If we want to find all the girls with records in the file then, after selecting option F and pressing the space bar to indicate that we are about to enter a search criterion, we must tell the database to search for all the records in which the entry under 'Sex' is 'f'. Because 'Sex' is field 3 of the record, we enter this criterion as:

F3 = "f"

When this has been entered the search is carried out and, when it is finished, the display of the main menu returns. The number of records meeting the criterion that were found during the search is displayed at the top of the screen against RF. If we browse through the file now any records that were found by the search will be marked by the message 'FOUND' in their top left-hand corner when they are displayed.

We can find all the children who have had mumps with:

F4 = "*Mumps"

The inclusion of the asterisk causes the database to be searched for records with the words 'Mumps' appearing anywhere in field 4, that is, under 'Diseases'. All the girls who have been inoculated against diphtheria can be found by:

F3 = "f" AND F5 = "*Diphtheria"

All the boys with medical appointments in April can be found by:

F3 = "m" AND F6 = "*April"

The children who have either had measles or been inoculated against it can be found with:

F4 = "*Measles" OR F5 = "*Measles"

These examples should serve to show that the database can be searched in a wide variety of ways. There is one unfortunate shortcoming which is that a search criterion cannot mix numeric and non-numeric parts.

The other options provided by the program are for activities such as adding records to a file and amending the records of the file. Option M can be useful in providing a shortened tabular summary of the contents of a file. It can only be used after Option N, which is needed to establish the style in which the table is to be presented.

All in all, Database is easy to use because it is so close to a card system, and with its pleasant display format it is convenient and flexible in the ways in which it can manage its database.

The commands

The commands that can be accepted by Database, and their purposes are as follows:

Add records. To add records to a file by filling in the form designed with the Setup Card option.

Browse/amend records. To browse through a file as described above. The current record can be amended, printed or deleted.

Calculations. To perform calculations on the numerical fields of records. The totals and averages of a particular numeric field can be found for all or some of the records in a file.

Delete records. To delete records from a file.

Exit program. To leave Database and go to BASIC.

Find records. To find the records that satisfy some criterion as described above.

Print records. To print some or all of the records in a file.

Reformat/setup card. To design a record card with which records can be added to a file, or to alter the design of a card.

Sort records. To sort the records in a file into a particular order.

Save file. To save a file on cassette or disk.

Append records. To append the records in a file to those in the current file. The records that are appended will inherit the card format of the current file.

Load file. To load a file from cassette or disk.

Field summary. To give a tabular summary of the contents of a file.

Format field summary. To design the layout of the table for the field summary.

Spreadsheets and Easi-Amscalc

Easi-Amscalc is a spreadsheet program that is intended to be easy to use, and as a result it lacks a few of the more sophisticated facilities possessed by some other spreadsheets. It can be used to create the sheets described in the previous chapter without any difficulty, though, and in this section we will show how this can be done.

When the program is first run it displays a blank sheet. The screen window shows part of a sheet with 30 rows, numbered from 1 to 30, and 26 columns labelled A to Z. The spreadsheet has a command with which the size of the sheet can be changed, but this size suits our needs and we will retain it. The sheet is

displayed on an 80-column screen, which allows quite a large part of the sheet to be shown in the screen window, but the display can be switched to a 40-column screen, in which a smaller part of the sheet is shown, although its cells are correspondingly larger. A second command allows the user to switch between the two forms of display at any time.

We have already mentioned two of the spreadsheet's commands, and a menu showing them all can be displayed at any time by typing M to give the Menu command. These commands are listed later in this chapter.

The commands include all the basic ones that we would expect in the light of the discussion of the previous chapter, and we will proceed to use them to construct the sheet for the family budget given in Chapter 1. All the items of text and the numbers in the columns under the headings 'Savings' and 'Bills' can be entered by moving to the appropriate cell, pressing E to give the Enter data command, and typing the text or number before finally pressing ENTER. When all the text and numbers are entered, it remains to enter the formulae. The formulae that Easi-Amscalc can support are relatively simple consisting, typically, of the difference of two entries in the sheet or the sum of the entries in a column. Simple as they are, they are sufficient for our needs. The formulae are not entered by typing them as formulae, but by indicating them to the spreadsheet. This is done by positioning the cursor on the cell where a formula is to be placed and pressing ENTER to confirm it, indicating the type of formula, using the minus sign for a subtraction, for example, and then using the cursor as before to indicate the two cells whose contents are involved in the subtraction. The formula will then be constructed in the form that we described in Chapter 1.

To give a concrete example, the formula for cell D11 is a column addition, and this is indicated by pressing C and then positioning the cursor on cell D6 to indicate the top of the column of figures that is to be added, followed by cell D9 to indicate the bottom.

Family budget (£)

	Income	Expenditure	
	Savings	Bills	
Jan	200.00	120.00	Water
Feb	200.00	180.00	Electricity
Mar	200.00	140.00	Gas
		100.00	Rates
Totals	600.00	540.00	
Balance	60.00		

Figure 2.7 The spreadsheet for the family budget

The completed sheet is shown in Figure 2.7. The formulae that have been entered can be displayed by pressing D to give the Display formulae command, and the result of this is shown in Figure 2.8.

FORMULA NO. 1
 D 11 = D 6 c D 9
 FORMULA NO. 2
 B 11 = B 6 c B 8
 FORMULA NO. 3
 B 13 = B 11 - D 11

Figure 2.8 The formulae used in the family budget spreadsheet

When a formula is first entered in this way, the formula does not appear in the cell, but neither does the value of the formula! Instead, the number of the formula appears in the cell. (The first formula to be entered is formula number one, the next is number two, and so on.) To make the *value* of the formula appear in the cell, we must give the Compute command, and this causes the spreadsheet to compute the values of all its formulae. Subsequently, if a number in one of the cells is changed, the Compute command must be given again to cause the spreadsheet to update the value of any formula involving that cell.

Company	Holding	Price(p)	Value(p)	Cost(p)	Diff(p)	Gain(p)
BT	400	141	56400	50	91	36400
Bejam	200	168	33600	148	20	4000
Lloyds	100	522	52200	446	76	7600
Total			142200			48000

Figure 2.9 Easi-Amscal's version of the shares portfolio spreadsheet

The sheet for the second example in Chapter 1, the share portfolio, cannot be created directly as described there because of the restricted nature of the formulae that Easi-Amscal can accept. An adapted form of this sheet is shown in Figure 2.9, with all values in pence and an extra column introduced for an intermediate calculation. The entries in the Value and Gain columns can, of course, be converted to pounds, but further intermediate col-

umns will be required. One further command that is useful in creating this sheet is *Block*, which allows a formula to be repeated, but with the appropriate adjustments, throughout a block of cells. To use it, all that is necessary is to indicate the formula that is to be repeated, and three of the corners of the block of cells in which it is to be repeated. This command was used to 'paint' the formulae in the Value and Gain columns.

Easi-Amscalc is, by intention, a rather rudimentary spreadsheet, but it is easy to use, and it provides a good entry point to spreadsheets and the exploration of their capabilities.

The commands

The commands provided by Easi-Amscalc and their purposes are listed below. Each command is given by keying its first letter.

Amend a formula. To change any formula in the current sheet.

Block allocate. To 'paint' a formula, with appropriate adjustments, throughout a rectangular block of cells.

Compute. To compute and display the values of the formulae in the spreadsheet.

Display formulae. To display the formulae entered in the current spreadsheet.

Enter data. To enter a number or text in a cell.

Formula entry. To enter a formula.

Graph. To display a row or column of numbers in the spreadsheet as a graph.

Jump cursor. To move the cursor to a specified cell.

Kill formula. To remove a formula from a cell. The formula still exists, and can be placed in another cell.

Load spreadsheet. To load a spreadsheet from cassette.

Menu. To display the menu of commands.

New spreadsheet. To create a new spreadsheet.

Print spreadsheet. To print all or part of a spreadsheet.

Quit. To leave Easi-Amscalc.

Replicate. To repeat the contents of a cell in a range of cells.

Save. To save a spreadsheet on cassette.

Titles lock. To lock onto the rows and columns in a spreadsheet that contain titles.

Utilisation of formulae. To show the allocation of formulae to cells in all or part of a sheet.

V. To set the number of decimal places with which numbers in the sheet are displayed.

W. To change the colours of the display.

Xchange. To swap between 80- and 40-column screens.

Other software

The programs described above are by no means the only ones of their kind for the Amstrad. There are other cassette-based versions and there are more sophisticated disk-based versions. There are programs that enhance the programs that we have dealt with, and it is interesting to mention a few of these, for they begin to show the full range of what can be achieved with the Amstrad and the available software.

There is a program known as The Stylewriter, from Tasman Software again, that can be used with the Amsword word processor to create print styles, or founts, when the word processor is used in conjunction with a dot matrix printer. The user can choose one of the founts provided by Stylewriter or can design one of his or her own. This adds an extra dimension to the personal style of presentation of documents that can be achieved with a word processor.

To go with Database, Gemini have produced Database Report Generator. This allows printed reports and summaries to be produced from a file created with Database. Further than this, addresses can be printed, on labels if required, for mail and information from the database can be merged into standard documents to personalise them.

Amsoft's Microspread is a disk-based spreadsheet that is much more advanced than Easi-Amscalc. It cannot accept sheets created with Easi-Amscalc, but it is operated in much the same way, so that any expertise gained from using Easi-Amscalc can be brought to bear when using the more sophisticated version.

These examples show that it is possible to start from quite a low level, in terms of the abilities of the software, and to achieve a good deal. If ambition or need then determine that more powerful

software is necessary, then it is possible to enhance what you have or to move on to acquire more powerful software.

Summary

One example each of word processor, database and spreadsheet program are closely examined in this chapter to show the way in which they are used, the range of facilities that they possess and the applications to which they can be put. The descriptions are intended to show the range of applications in which these programs can be used and, together with the previous chapter, to reveal any shortcomings that the programs may have. Programs that enhance or supersede them are also mentioned to demonstrate that it is possible to grow and develop as you become more confident about the application of your Amstrad. The possibilities of extending the area of usefulness of your computer system will almost certainly require attention to the hardware as well as to the software, though, and these matters are dealt with in Chapter 4.

3 Problem solving

One of the most creative ways to use your Amstrad is for problem solving. Complete books have been written on the ways in which one may go about solving problems, whether with the aid of a computer or not, but when you have read such a book, it is not entirely certain that you will be in a better position to set about solving a specific problem. In this chapter we shall proceed by posing a number of problems and then explaining how they can be solved with the help of the computer. The result of this will be a number of programs written in Amstrad BASIC each of which, in essence, instructs the computer in how to go about finding the solution to a problem. In this way, we can examine and illustrate various methods for solving problems and, in particular, methods that are suitable for use with computers, for they can be rather different from those that are used to solve problems in other circumstances.

We shall see that there is a certain benefit in writing programs with a clear structure and, preferably, with a structure reflecting that of the problem that is being solved. We shall also see that the way in which a program presents the solution that it finds to its users can be greatly improved by making use of graphics. The way in which the solution method works can also be made clear with the use of carefully designed graphics, and this can enhance considerably the value of a problem-solving program.

Besides this, the problems that we shall consider illustrate areas in which computers are commonly applied, so that if you are either already involved or are thinking of becoming involved in one of these areas of computing, the way in which the relevant problem is solved will give you a flavour of the way that the computer is used there.

So, each of the sections of this chapter contains a problem for solution with the aid of the computer. For each, we describe a way of solving it and then write and present a program

incorporating this solution method. After this, we pause to see if we can improve, first, the way that the program presents its solution and, second, the way that it demonstrates how it goes about finding it. This is usually done with the considered use of graphics. This approach gives us two programs for solving each problem. The first will contain the method for solving the problem: the second will be an expanded version of the first that considers the requirements of the users of the program. By presenting the two versions of the program alongside each other, we hope that it will be possible to see how the first is developed into the second and how, if the first one has a decent structure, the development of the second is quite natural.

Networks and routeing

Problems involving networks and finding routes through them can occur in quite different contexts. For example, a network can consist of the roads linking a number of towns together, and the problem of the motorist travelling from town to town is to find the best route between them. The airline passenger faces a similar problem in getting from one airport to another in the network of international airline routes. The public water supply authorities have the same problem in supplying water in sufficient quantities to all the reservoirs that are linked together by their network of water pipes.

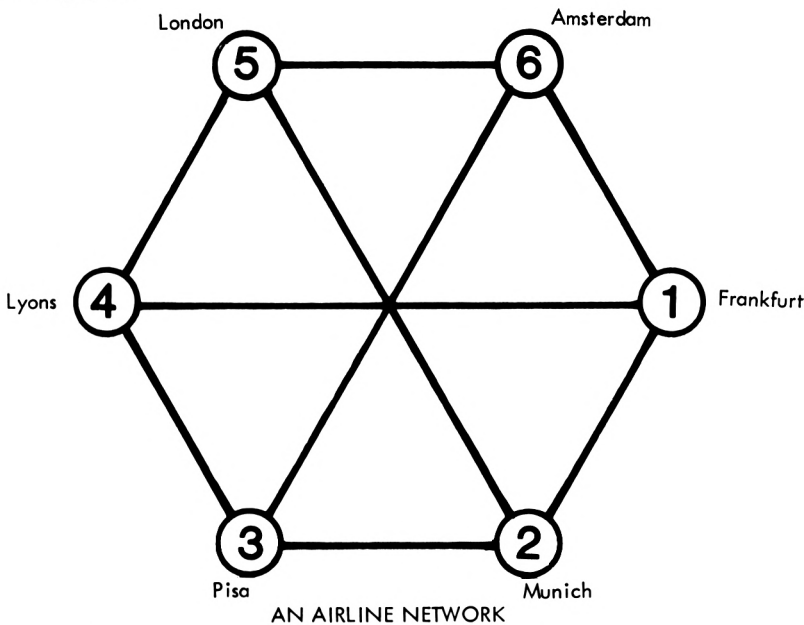


Figure 3.1 An airline network

The particular problem that we want to solve with the aid of the computer is this. Given the network of cities linked by airline routes as shown in Figure 3.1, what is the best route for a traveller to take from any city to any other? By 'the best route' we mean, to begin with at least, the route that requires the smallest number of stops between departure and arrival at the destination. Although the problem is stated in terms of an airline network and the routes that an airline passenger should take, it is probably clear that the problem could equally well have been posed for car travel, water supply or in any of a number of other contexts.

In beginning to solve our problem, the first thing that we need to do is to describe the network to the computer. To do this, we shall first give the computer the names of the cities in the network, and then tell it which cities are directly connected to each other. We could give the names of the cities by assigning them to string variables named, say, A\$, B\$ and so on, but we can do it systematically and at the same time help with the devising of a solution method if we assign the names to the elements of an array of string variables. We shall use an array named N\$ which is declared by

```
DIM N$(6)
```

A typical assignment to an element of this array is

```
N$(1)="FRANKFURT"
```

The airport names are associated with the particular elements of the array by using the number in the circle representing each airport in Figure 3.1 to give the number of the element of the array that is to hold the name of the airport.

To record the airports that are connected directly to each other, we shall use a method that can be used to advantage in many computer methods. We want to record, for example, that Frankfurt, airport 1, is connected to Lyons, which is airport 4, and that Pisa, airport 3, is connected to Munich, airport 2. This can be done by using a two-dimensional array named C, which is declared by:

```
DIM C(6, 6)
```

and using its elements so that, to state the general case, C(J, K) is assigned the number of connections between airport j and airport k. The number of connections will be 1 if there is a connection, or 0 if there is not. The two connections just mentioned can be recorded by the assignment:

$$C(1, 4)=1: C(3, 2)=1$$

As there is no direct connection between airports 4 and 6 we shall make the assignments:

$$C(4, 6)=0$$

By recording all the cities in the network and how they are connected to each other, we have made a description of the network that is suitable for the computer. It now remains to devise a way of finding routes through it. We can write the program so that a passenger will enter his or her point of departure and destination. Examination of Figure 3.1 shows that if two airports are not directly linked to each other, then there is always at least one route that involves only one intermediate airport. Our program should test to see if the departure and destination airports are directly connected, which it can do by referring to the two-dimensional array, C. If they are it can report the direct connection as the best route. If they are not it must find an intermediate airport through which the flight can be made. This can also be done with reference to C by finding an airport to which the departure airport is connected, and which is also connected to the destination. It would seem to be desirable to report all the possible intermediate airports in order to give the traveller a choice of routes.

The following program is the result of these ideas on solving the problem.

```

10 GOSUB 100: REM READ NETWORK DESCRIPTION
20 GOSUB 300: REM PASSENGER ENTERS AIRPORTS
30 GOSUB 400: REM FINDING THE ROUTE
40 END
100 REM NETWORK DESCRIPTION
110 DIM N$(6), C(6, 6)
120 FOR K=1 TO 6
130 READ N$(K)
140 NEXT K
150 FOR J=1 TO 6
160 FOR K=1 TO 6
170 READ C(J, K)
180 NEXT K
190 NEXT J
200 DATA "FRANKFURT", "AMSTERDAM", "LONDON"
210 DATA "LYONS", "PISA", "MUNICH"
220 DATA 0, 1, 0, 1, 0, 1
230 DATA 1, 0, 1, 0, 1, 0
240 DATA 0, 1, 0, 1, 0, 1
250 DATA 1, 0, 1, 0, 1, 0
260 DATA 0, 1, 0, 1, 0, 1
270 DATA 1, 0, 1, 0, 1, 0

```

```

280 RETURN
300 REM PASSENGER ENTERS AIRPORTS
310 CLS
320 PRINT "ENTER DEPARTURE AIRPORT"
330 INPUT A$
340 PRINT "ENTER DESTINATION AIRPORT"
350 INPUT B$
360 RETURN
400 REM FINDING THE ROUTE
410 PRINT "TO GET FROM "; A$; " TO "; B$
420 F=0
430 FOR J=1 TO 6
440 FOR K=1 TO 6
450 IF A$=N$(J) AND B$=N$(K) AND C(J, K)=1 THEN PRINT "THERE
IS A DIRECT CONNECTION": F=1
460 NEXT K
470 NEXT J
480 IF F=1 THEN RETURN
490 FOR J=1 TO 6
500 FOR K=1 TO 6
510 IF NOT(A$=N$(J) AND B$=N$(K)) THEN 550
520 FOR L=1 TO 6
530 IF C(J,L)=1 AND C(L,K)=1 THEN PRINT "THERE IS A ROUTE
VIA "; N$(L)
540 NEXT L
550 NEXT K
560 NEXT J
570 RETURN

```

A typical dialogue with this program produces the following output.

```

ENTER DEPARTURE AIRPORT
LONDON
ENTER DESTINATION AIRPORT
PISA
TO GET FROM FROM LONDON TO PISA
THERE IS A ROUTE VIA MUNICH
THERE IS A ROUTE VIA LYONS
THERE IS A ROUTE VIA AMSTERDAM

```

The arrays and variables used in this program and the purpose of each are summarised in the following table.

<i>Name</i>	<i>Item stored under this name</i>
Arrays:	
N\$	The names of the airports
C	The pattern of connections between the airports
Variables:	
A\$, B\$	The names of variables as input to the program
F	A value to indicate if a direct connection has been found

Now, although this program solves our problem, its output is not presented in a particularly appealing manner. It would be much more helpful if it could display a map of the network and show the possible routes on the map. There are several reasons for saying this, but the main one is that most people can absorb information more readily if it is presented pictorially than if it is in words. The contrast is not particularly marked with our network of six airports, but imagine the corresponding situation for a world-wide network. Also, any language problems, which will be bound to occur with the assortment of people travelling on international flights, can be minimised by avoiding the use of language as much as possible.

We propose to further develop our program to make it display the map of the network. (At the same time, this will prove that our description of the network is adequate.) Then, when a route between two airports is requested, we will make the program mark it on the map.

The map of the network can be drawn by a display subroutine that is called after the network description has been read. In essence, it deals with each airport in turn by drawing a point to represent it, and then drawing lines from it to all the airports to which there are routes from it. The subroutine needs to be told where on the screen each airport is to be positioned, but this is the only extra information that it needs.

When the airports at the start and end of a passenger's journey have been entered, the route can be highlighted on the map by calling another subroutine. If there is more than one route then the alternative routes will be highlighted in different colours so that the passenger will be able to distinguish them and then to choose between them. The resulting program, with its new subroutines, but not the ones already given above, is listed below.

```

10 DIM R(6), CO(6)
20 GOSUB 100: REM READ NETWORK DESCRIPTION
30 GOSUB 300: REM PASSENGER ENTERS AIRPORTS
40 GOSUB 1000: REM DISPLAY THE NETWORK
50 GOSUB 2000: REM FINDING AND MARKING THE ROUTE
60 END
1000 REM DISPLAY THE NETWORK
1010 FOR K=1 TO 6
1020 READ CO(K), R(K)
1030 NEXT K
1040 DATA 400, 200, 300, 300, 200, 300, 100, 200
1050 DATA 200, 100, 300, 100
1060 INK 0, 1: INK 1, 24: CLS: CLG
1070 FOR K=1 TO 6
1080 FOR J=1 TO 6
1090 IF J>K AND C(J, K)=1 THEN MOVE CO(K), R(K): DRAW CO(J),
R(J), 1

```

```

1100 NEXT J
1110 NEXT K
1120 FOR K=1 TO 6
1130 X=CO(K): Y=R(K)
1140 IF Y=300 THEN Y=60
1150 IF Y=200 THEN Y=240: X=X-60
1160 IF Y=100 THEN Y=340
1170 LOCATE X/16, Y/16: PRINT N$(K)
1180 NEXT K
1190 RETURN
2000 REM FINDING AND MARKING THE ROUTE
2010 F=0: P=2: INK 2, 0
2020 FOR J=1 TO 6
2030 FOR K=1 TO 6
2040 IF A$=N$(J) AND B$=N$(K) AND C(J, K)=1 THEN MOVE CO(J),
R(J): DRAW CO(K), R(K), 2: F=1
2050 NEXT K
2060 NEXT J
2070 IF F=1 THEN RETURN
2080 FOR J=1 TO 6
2090 FOR K=1 TO 6
2100 IF NOT(A$=N$(J) AND B$=N$(K)) THEN 2140
2110 FOR L=1 TO 6
2120 IF C(J,L)=1 AND C(L,K)=1 THEN MOVE CO(J), R(J): DRAW
CO(L), R(L), P: DRAW CO(K), R(K), P: P=P+1
2130 NEXT L
2140 NEXT K
2150 NEXT J
2160 RETURN

```

The program uses mode 1, and in this mode only four colours can be displayed. With two of the colours used for the background and the network itself, only two remain for highlighting the routes. This causes a problem when there are three possible routes to be shown. If the background colour is used again for a route, the effect is to remove one of the routes from the display as shown in the illustration. Although this is not entirely satisfactory, the alternative of using mode 0 is, despite its providing more colours, even less so.

The arrays and variables used by this program but not by the previous one, and their purposes, are given in the table below.

<i>Name</i>	<i>Item stored under this name</i>
Arrays:	
CO	{ The column and row positions for the display of the airport with its name in the corresponding element of N\$
R	
Variables:	
X	{ The column position (X) and the row position (Y) for the display of the name of an airport
Y	

As a final refinement, we can adapt the program so that when there is a choice of routes, it finds and displays the one that is the cheapest. To do this, we must give the cost of the flights between the various cities to the program. Suppose that the costs are as shown in the following table.

Flight costs in pounds

	<i>Frankfurt</i>	<i>Munich</i>	<i>Pisa</i>	<i>Lyons</i>	<i>London</i>	<i>Amsterdam</i>
<i>Frankfurt</i>	0	20	0	70	0	70
<i>Munich</i>	20	0	40	0	100	0
<i>Pisa</i>	0	40	0	40	0	100
<i>Lyons</i>	70	0	40	0	110	0
<i>London</i>	0	100	0	110	0	35
<i>Amsterdam</i>	70	0	100	0	35	0

The entries in the table not only give the cost of the flight between two airports when there is a direct link, but by taking the value zero they also indicate where there is no direct link. If we store this data in the two-dimensional array C so that $C(J, K)$ is assigned the cost of the flight between airports j and k , as given in the table, then we can interpret zero as before, but we can take any non-zero value both to indicate that there is a connection and to give the fare for that flight.

The following program reads the fares data into C , and uses it to find the cheapest route between any pair of airports.

```

10 DIM R(6), CO(6)
20 GOSUB 100: REM READ NETWORK DESCRIPTION AND COSTS
30 GOSUB 300: REM PASSENGER ENTERS AIRPORTS
40 GOSUB 1000: REM DISPLAY THE NETWORK
50 GOSUB 2000: REM FINDING AND MARKING THE CHEAPEST ROUTE
60 END
100 REM NETWORK DESCRIPTION AND COSTS
110 DIM N$(6), C(6, 6)
120 FOR K=1 TO 6
130 READ N$(K)
140 NEXT K
150 FOR J=1 TO 6
160 FOR K=1 TO 6
170 READ C(J, K)
180 NEXT K
190 NEXT J
200 DATA "FRANKFURT", "AMSTERDAM", "LONDON"
210 DATA "LYONS", "PISA", "MUNICH"
220 DATA 0, 20, 0, 70, 0, 70
230 DATA 20, 0, 40, 0, 100, 0
240 DATA 0, 40, 0, 40, 0, 100

```

```

250 DATA 70, 0, 40, 0, 110, 0
260 DATA 0, 100, 0, 110, 0, 35
270 DATA 70, 0, 100, 0, 35, 0
280 RETURN
1000 REM DISPLAY THE NETWORK
1010 FOR K=1 TO 6
1020 READ CO(K), R(K)
1030 NEXT K
1040 DATA 400, 200, 300, 300, 200, 300, 100, 200,
1050 DATA 200,100, 300, 100
1060 INK 0, 1: INK 1, 24: CLS: CLG
1070 FOR K=1 TO 6
1080 FOR J=1 TO 6
1090 IF J>K AND C(J, K)>0 THEN MOVE CO(K), R(K): DRAW CO(J),
R(J), 1
1100 NEXT J
1110 NEXT K
1120 FOR K=1 TO 6
1130 X=CO(K): Y=R(K)
1140 IF Y=300 THEN Y=60
1150 IF Y=200 THEN Y=240: X=X-60
1160 IF Y=100 THEN Y=340
1170 LOCATE X/16, Y/16: PRINT N$(K)
1180 NEXT K
1190 RETURN
2000 REM FINDING AND MARKING THE CHEAPEST ROUTE
2010 F=0: INK 2, 8
2020 FOR J=1 TO 6
2030 FOR K=1 TO 6
2040 IF A$=N$(J) AND B$=N$(K) AND C(J, K)>0 THEN MOVE CO(J),
R(J): DRAW CO(K), R(K), 2: F=1
2050 NEXT K
2060 NEXT J
2070 IF F=1 THEN RETURN
2080 C=0
2090 FOR J=1 TO 6
2100 FOR K=1 TO 6
2110 IF NOT(A$=N$(J) AND B$=N$(K)) THEN 2160
2120 FOR L=1 TO 6
2130 IF C(J,L)>0 AND C(L,K)>0 THEN D=C(J, L)+C(L, K)
2140 IF D<C THEN C=D: X=J: Y=K: Z=L
2150 NEXT L
2160 NEXT K
2170 NEXT J
2180 MOVE CO(X), R(X): DRAW CO(Z), R(Z), 2: DRAW CO(Y),
R(Y), 2
2190 RETURN

```

Grazing problems

There are some problems that cannot be solved exactly by using mathematics and logic. This does not mean that they cannot be solved at all, for approximations to their solutions can be found with a computational method which, naturally enough, makes the computer the ideal vehicle for determining them. One problem which, although it appears to be quite simple, turns out to need a computational method for its solution concerns a goat grazing in a field. This problem is not only a prime example of the

computer coming into its own for problem solving but also illustrates very well the sort of method that is likely to be used in computer-based problem solving.

If the idea of finding only an approximate solution seems none too acceptable it is, first of all, obviously better than having no solution at all. But an approximate solution may even be as useful as an exact one for, at the cost of an increase in the amount of computation that may be required, the accuracy of the approximation can usually be increased, so that the approximation to the solution that is finally obtained may be as close to the precise solution as is required.

This problem concerns a goat that is tethered to the edge of a circular field. When the field has a radius of ten metres and the goat is at the top of the field, the situation is as shown in Figure 3.2. The problem, as it is usually stated, is to find the length of the lead by which the goat is tethered when the area that the goat can graze is exactly half the area of the field.

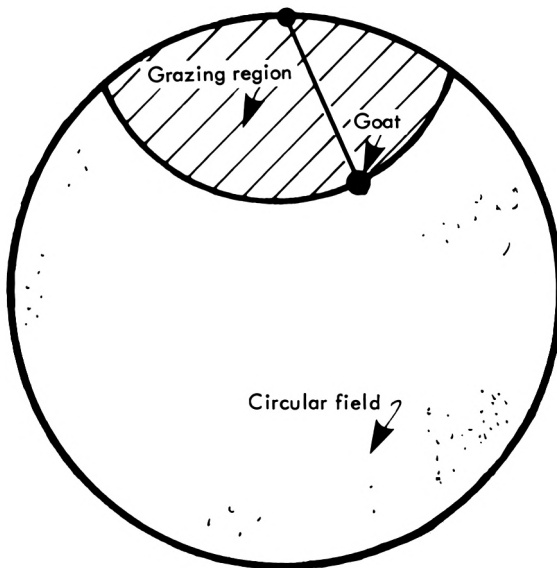


Figure 3.2 Goat tethered in a circular field

Rather than tackling the problem head-on, the general approach we shall adopt is to find the area that the goat can graze when the tether restraining it has a given length. Then by dividing this area by the area of the field, we shall have the fraction of the field that the goat can graze. By varying the length

of the lead in a systematic way, we can then find the length for the tether that makes this fraction more or less one half.

Before using this method to solve the problem, we need to be able to find the area of a region for which we know the boundaries. The basic idea of the way in which we shall do this is illustrated in Figure 3.3. The region, with its irregular boundaries, is laid on a square grid. To find an approximation to its area, we can count the squares that fall within it and add their areas. There is a slight difficulty over what to do about the squares at the edge of the region that fall partly inside and partly outside it. We shall count these only if the centre of the square falls within the region. This means that we are to count a square if most of it lies in the region and to neglect it otherwise. We might reasonably expect the amount from such squares that are counted almost to be balanced by the corresponding amount from the squares that we neglect.

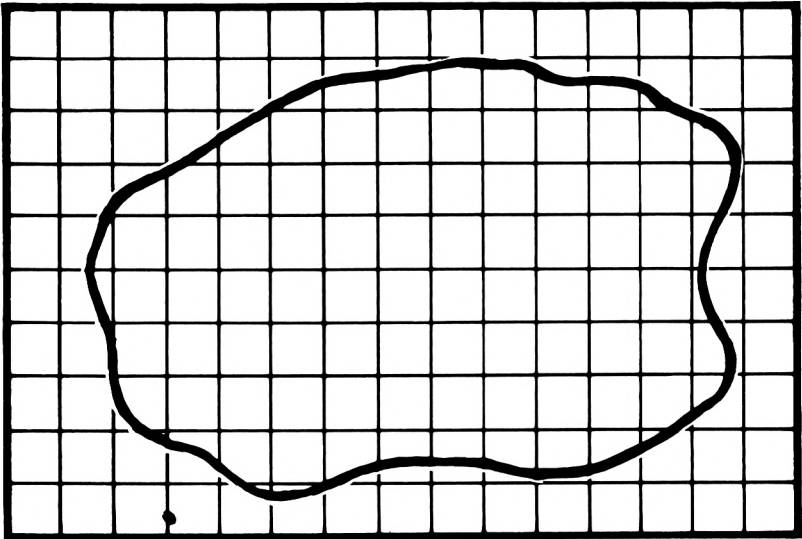


Figure 3.3 Finding the area of a region

We shall begin by writing a program to find the area of the circular field. (We know that the area is $\pi \cdot 10 \cdot 10$, so that we shall be able to check the answer that we get.) By using the same method, we can then find the area that the goat can graze when the lead has a given length. After this, we can vary the length of the lead in a systematic way until the area that the goat can graze is approximately half that of the field.

A program based on the idea for finding the area of a circular field of radius ten covered with squares of side one, is listed

below. It works by counting all the squares the centres of which lie within the distance R of the centre of the circle.

```

100 R=10: A=0
110 FOR J= -R TO R-1
120 FOR K= -R TO R-1
130 D=SQR((J+0.5)^2 + (K+0.5)^2)
140 IF D<R THEN A=A+1
150 NEXT K
160 NEXT J
170 PRINT "Area is "; A

```

The program gives the result as 316. Compared with the exact value of 314.16 this shows an error of 0.58%. The accuracy could be increased by reducing the size, and so increasing the number, of the squares in the grid.

Since the method and a grid of unit squares gives an acceptable accuracy, we can feel confident about going on to use both the method and this grid for finding the size of the area that the goat can graze. The solution program stores the length of the lead under L, assigning a value of one to L as a starting value, and then finds the area that the goat can graze. It then repeatedly increases the value assigned to L by one and finds the new value for the grazing area until the area exceeds half that of the field. The area that the goat can graze with a lead of a given length is computed by counting those squares with their centres both within a distance R of the centre of the circle and within a distance L of the point where the goat is tethered. The program is:

```

100 R=10: L=1
110 G=0
120 FOR J= -R TO R-1
130 FOR K= -R TO R-1
140 D=SQR((J+0.5)^2 + (K+0.5)^2)
150 E=SQR((R-J-0.5)^2 + (K+0.5)^2)
160 IF D<R AND E<L THEN G=G+1
170 NEXT K
180 NEXT J
190 IF G/(100*PI)>0.5 THEN 220
200 L=L+1
210 GOTO 110
220 PRINT "Length of lead is "; L

```

The program gives a length of 12 for the lead when the goat can graze half of the field. Again, the accuracy with which the length is determined can be improved by increasing the fineness of the covering grid.

The variables in this program and the previous one are listed in the table below along with the purposes for which they are used.

<i>Name</i>	<i>Item stored under this name</i>
R	The radius of the field
A	The area of the field
D	The distance from the centre of one of the squares of the grid to the centre of the field
L	The length of the lead
PI	The value of pi
G	The area that the goat can graze
E	The distance from the centre of one of the squares in the grid to the point at which the goat is tethered

Now, although this program solves the original problem, it does not really give any insight into the situation to a person who simply uses the program. We can adapt it so that it does so with the addition of some graphics. With graphics, we can illustrate the part of the field that the goat can graze with each different length of the lead. This turns out to be quite easy to do, because Amstrad BASIC provides us with graphics commands with which we can show the field as a circle filled with colour. The circle representing the field will naturally be filled with green, and the part that the goat can graze can be removed from it for each different length of the lead.

To draw a filled circle, we can draw a series of lines from its centre to points on its edge, radii in fact. If we draw such lines all round the circle, as long as they are close enough to each other, the circle will be filled. A sector can be filled in similar fashion by drawing the radii for only part of a circle.

The program with the additions to give the graphics is listed below, and Figure 3.4 shows a display that it produces.

```

100 CLS: DEG: INK 0, 24: INK 1, 18: INK 2, 24
110 LOCATE 1, 1: PRINT "Length      % grazing"
120 R=10: L=1
130 S=0: F=360: X=320: Y=200: W=18*R
140 Z=11: GOSUB 1000
150 G=0
160 FOR J= -R TO R-1
170 FOR K= -R TO R-1
180 D=SQR((J+0.5)^2 + (K+0.5)^2)
190 E=SQR((R-J-0.5)^2 + (K+0.5)^2)
200 IF D<R AND E<L THEN G=G+1
210 NEXT K
220 NEXT J
230 S=180: F=360: X=320: Y=380: W=18*L: Z=2: GOSUB 1000
240 IF G/(100*PI)>0.5 THEN 270
250 L=L+1
260 GOTO 150

```

```

270 LOCATE 9, 1: PRINT L
280 LOCATE 25, 1: PRINT G/PI
290 END
1000 FOR T=S TO F
1010 MOVE X,Y
1020 DRAW X+W*COS(T), Y+W*SIN(T), Z
1030 NEXT T
1040 RETURN

```

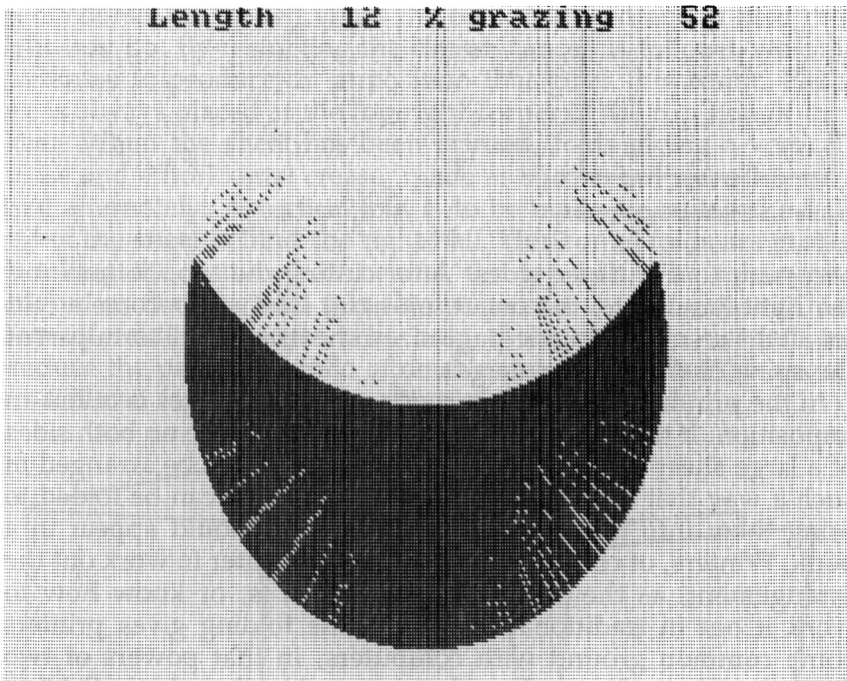


Figure 3.4 Grazing program

How to number pages

This section provides an example of how a computer can be used to solve a problem when many people cannot imagine how it might be possible even to begin to do so. The problem is to determine how to arrange pages when a number of them are to be printed on a large sheet of paper so that, when the large sheet is folded, the pages will come into their correct numerical order as they should appear when in a book or magazine. This is illustrated in Figure 3.5, which shows top views of a set of eight pages printed on one sheet which is then folded twice so that the resulting stack of eight pages starts with page 1 on the top and goes through to page 8 on the bottom. After the folding, the edges of the folded sheet must be trimmed to separate the pages from each other so that they can be turned, but it is only

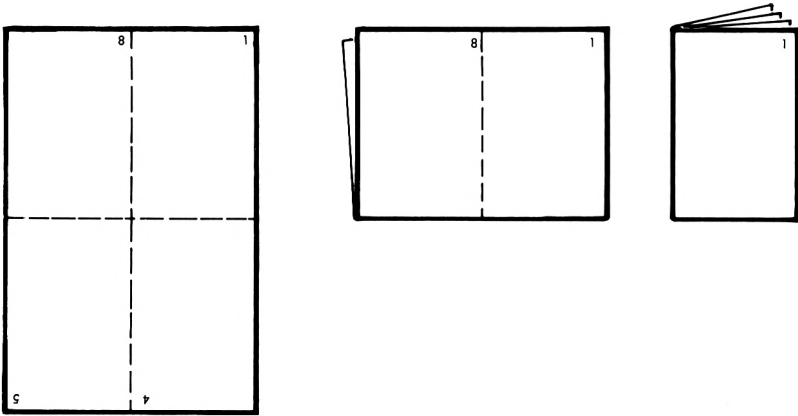


Figure 3.5 Folding eight pages

necessary to trim three sides. After cutting the top, bottom and the right-hand side, the uncut left-hand side will become a fold for all the resulting double-page-sized sheets and will form part of the spine of the book or magazine.

In the printing trade, this sort of assembly of pages is called an imposition. If the pages are to be printed all at once on both sides of a large sheet of paper, it is clear that they must be arranged in such a way that after they are printed the sheet can be folded to bring them into their proper order as they should appear in a book. Printers, of course, do not need a computer to work out the arrangement, as they either know what it is, or know how to work it out in traditional ways. But the folding gives printers some common ground with computers, in that powers of two figure largely in the numbers that are used by them both. Folding naturally leads to a power of 2 for the number of pages in an imposition, as the number of pages that eventually result from each sheet doubles with each fold. 32-page and 64-page impositions are commonly used. Computer memory sizes also come in powers of two as a result of the binary nature of the fundamental process involved in their operation.

We will develop an Amstrad BASIC program to compute the arrangement of the pages for an imposition containing any number of pages as long as that number is a power of 2. If the number of pages that is required is not an exact power of 2, then an imposition for the smallest power of 2 that exceeds the number must be made, and it will include some blank pages. In practice, it is said that a sheet of paper cannot be folded more than seven times, and seven folds would give an imposition of 2^8 or 256 pages. For this reason, impositions do not contain more than this number of pages and, as we have said, usually contain rather

fewer. But this need not prevent us from making our program completely general.

There is more than one way to arrange the pages of an imposition so that, after folding, a stack of pages is obtained all of which are the right way up and which are in order. A book is traditionally made with page one, and all the odd-numbered pages, on the right-hand side when a book is opened. The spine is, of course, on the left when the book is closed and is held the right way up. By restricting the circumstances as follows, we can guarantee to produce an imposition that gives this result. Page one is always placed at the top right corner of the top side of the sheet. When the sheet has been printed, folds giving a vertical crease and a horizontal crease, made as shown in Figure 3.6, are used alternately. A vertical fold is to be made by taking the left half of the sheet behind the right half. Horizontal folds are made by taking the bottom half behind the top half. The folding is ordered so that the final fold always gives a vertical crease. This form of folding ensures that only three cuts are needed to produce the pages as a folio of double sheets, and that the uncut left side can form part of the spine of the book.

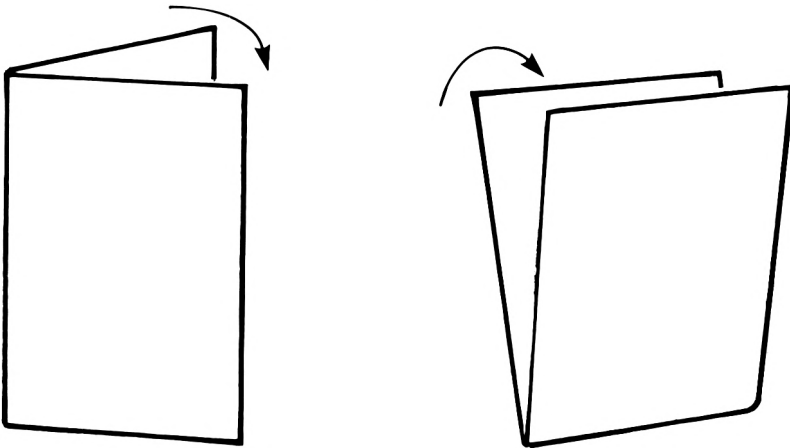


Figure 3.6 The vertical and horizontal folds

Having explained the problem, it remains to write a program to solve it. Unless you are absolutely clear about the problem it may be a good idea to try folding a few sheets of paper and numbering the 'pages' that result from the folding to get a feel for what is happening. This is worth doing in any case, for it gives us the basic idea that we shall use to solve the problem.

We shall approach the problem of finding the arrangement of pages in an imposition by starting with the stack of pages in

order, and 'unfolding' them to obtain their original arrangement. This means that we shall need one subroutine for 'unfolding' a fold that gives a vertical crease, and another for 'unfolding' the fold that gives a horizontal crease. When we have written them, we can apply them alternately, starting with the vertical one, until our stack of pages has been unfolded into a single sheet.

To visualise what is required, it may help to refer to Figure 3.6 again and to consider how a stack of eight pages can be unfolded. We can consider the original stack as consisting of pages arranged in one row and one column, and stacked to a depth of eight. Unfolding the first vertical fold will give us one row with two columns of stacked pages, and the pages will be stacked to a depth of four in each stack. The right-hand stack will be the top half of the previous single stack. The left-hand stack will be the bottom half of the original stack, but in reverse order. After this, unfolding the horizontal fold gives us two rows each with two columns of stacked pages. The pages in the upper row are those from the top half of the previous row. Those in the second are from the bottom half of the previous row, but reversed in order and turned upside down. The pages are now stacked to a depth of two, and this means that we have reached the original sheet as it was printed, for the depth of two pages corresponds to the pages printed on both sides of the sheet.

In this way, by starting with a known arrangement of pages, we can keep track of where each page goes as unfolding takes place, and we can tell when all the unfolding is finished to give the imposition.

After this, if we begin by thinking about just where the pages with a particular number should be, the main thing that the program has to do is to store and rearrange the page numbers. For an eight-page imposition, we can store the page numbers in an array declared by

```
DIM P(8)
```

and initialised with the page numbers by

```
FOR K=1 TO 8: P(K)=K: NEXT K
```

Since the pages are arranged in piles in a number, C , of columns, a number, R , of rows and each with a depth, D , we can keep track of what happens in an unfolding by having a rule that tells us whereabouts in the array P a page with a given column, row and depth is to be found. As we have just demonstrated, unfolding a vertical fold doubles the number of columns in which the piles are

arranged, while halving their depth. Unfolding a horizontal fold doubles the number of rows of piles and halves their depth. The rule we shall use to give the position in the array for the page in the pile in column CO, row RO at a depth DE is

$$1 + R*D*(CO-1) + D*(RO-1) + DE-1$$

With this preamble, the program for finding the imposition for any number of pages that is a power of two is:

```

10 PRINT "ENTER NUMBER OF PAGES. THIS MUST BE A POWER OF 2"
20 INPUT N
30 DIM P(N), T(N)
40 C=1: R=1: D=N
50 FOR K=1 TO N: P(K)=K: NEXT K
60 GOSUB 1000: REM VERTICAL UNFOLDING
70 IF D=2 THEN 110
80 GOSUB 2000: REM HORIZONTAL UNFOLDING
90 IF D=2 THEN 110
100 GOTO 60
110 GOSUB 3000: REM PRINT RESULTS
120 END
1000 PR=R: PC=C: PD=D
1010 D=D/2: C=2*C
1020 FOR CO=1 TO C
1030 FOR RO=1 TO R
1040 FOR DE=1 TO D
1050 L=1 + R*D*(CO-1) + D*(RO-1) + DE-1
1060 LR=RO
1070 IF CO>PC THEN LC=CO-PC: LD=DE: GOTO 1090
1080 LC=PC+1-CO: LD=PD+1-DE
1090 PL=1 + PR*PD*(LC-1) + PD*(LR-1) + LD-1
1100 T(L)=P(PL)
1110 NEXT DE
1120 NEXT RO
1130 NEXT CO
1140 FOR J=1 TO N: P(J)=T(J): NEXT J
1150 RETURN
2000 PR=R: PC=C: PD=D
2010 D=D/2: R=2*R
2020 FOR CO=1 TO C
2030 FOR RO=1 TO R
2040 FOR DE=1 TO D
2050 L=1 + R*D*(CO-1) + D*(RO-1) + DE-1
2060 LC=CO
2070 IF RO>PR THEN LR=R+1-RO: LD=PD+1-DE: GOTO 2090
2080 LR=RO: LD=DE
2090 PL=1 + PR*PD*(LC-1) + PD*(LR-1) + LD-1
2100 T(L)=P(PL)
2110 NEXT DE
2120 NEXT RO
2130 NEXT CO
2140 FOR J=1 TO N: P(J)=T(J): NEXT J
2150 RETURN
3000 CLS
3010 FOR DE=1 TO D
3020 IF DE=1 THEN PRINT "TOP PAGES": GOTO 3040
3030 PRINT "BOTTOM PAGES"

```

```

3040 FOR RO=1 TO R: FOR CO=1 TO C
3050 L=1 + R*D*(CO-1) + D*(RO-1) + DE-1
3060 PRINT P(L); " ";
3070 NEXT CO: PRINT: NEXT RO
3080 NEXT DE
3090 RETURN

```

For an initial input of 32 the program gives the following output:

```

TOP PAGES
8  25 32  1
9  24 17 16
12 21 20 13
5  28 29  4
BOTTOM PAGES
7  26 31  2
10 23 18 15
11 22 19 14
6  27 30  3

```

These numbers give the order in which the numbered pages must be arranged on the large sheet. The numbers for the pages on the top of the sheet show the arrangement of the pages as they would be seen there. The numbers for the pages on the bottom of the sheet show the arrangement as it would appear when viewed from the top, that is, page 2 is under page 1, page 7 is under page 8 and so on.

The program only shows where the pages appear; it does not show which must be printed the right way up and which upside down. The program can be amended to show this in schematic form by printing a layout in which the number of a page that is the right way up is printed in one colour and that of a page that is inverted in another. It is only a horizontal fold that causes a page to be turned upside down, so that only the subroutine starting at line 2000 needs changes other than those that simply keep track of the unfolding actions. The amended program is listed below, and a sample of its output for a 32-page imposition is shown in Figure 3.7.

```

10 PRINT "ENTER NUMBER OF PAGES. THIS MUST BE A POWER OF 2"
20 INPUT N
30 DIM P(N), T(N), Q(N), S(N)
40 C=1: R=1: D=N
50 FOR K=1 TO N: P(K)=K: Q(K)=0: NEXT K
60 GOSUB 1000: REM VERTICAL UNFOLDING
70 IF D=2 THEN 110
80 GOSUB 2000: REM HORIZONTAL UNFOLDING
90 IF D=2 THEN 110
100 GOTO 60
110 GOSUB 3000: REM PRINT RESULTS

```



```

120 END
1000 PR=R: PC=C: PD=D
1010 D=D/2: C=2*C
1020 FOR CO=1 TO C
1030 FOR RO=1 TO R
1040 FOR DE=1 TO D
1050 L=1 + R*D*(CO-1) + D*(RO-1) + DE-1
1060 LR=RO
1070 IF CO>PC THEN LC=CO-PC: LD=DE: GOTO 1090
1080 LC=PC+1-CO: LD=PD+1-DE
1090 PL=1 + PR*PD*(LC-1) + PD*(LR-1) + LD-1
1100 T(L)=P(PL): S(L)=Q(PL)
1110 NEXT DE
1120 NEXT RO
1130 NEXT CO
1140 FOR J=1 TO N: P(J)=T(J): Q(J)=S(J): NEXT J
1150 RETURN
2000 PR=R: PC=C: PD=D
2010 D=D/2: R=2*R
2020 FOR CO=1 TO C
2030 FOR RO=1 TO R
2040 FOR DE=1 TO D
2050 L=1 + R*D*(CO-1) + D*(RO-1) + DE-1
2060 LC=CO
2070 IF RO>PR THEN LR=R+1-RO: LD=PD+1-DE: GOTO 2090
2080 LR=RO: LD=DE
2090 PL=1 + PR*PD*(LC-1) + PD*(LR-1) + LD-1
2095 IF RO>PR THEN Q(PL)=1-Q(PL)
2100 T(L)=P(PL): S(L)=Q(PL)
2110 NEXT DE
2120 NEXT RO
2130 NEXT CO
2140 FOR J=1 TO N: P(J)=T(J): Q(J)=S(J): NEXT J
2150 RETURN
3000 CLS
3010 FOR DE=1 TO D
3020 IF DE=1 THEN PRINT "TOP PAGES": GOTO 3040
3030 PRINT "BOTTOM PAGES"
3040 FOR RO=1 TO R: FOR CO=1 TO C
3050 L=1 + R*D*(CO-1) + D*(RO-1) + DE-1
3055 IF Q(L)=0 THEN PEN 1 ELSE PEN 3
3060 PRINT P(L); " ";
3070 NEXT CO: PRINT: NEXT RO
3080 NEXT DE
3090 RETURN

```

```

Top pages
8 22 1
9 17 16
12 20 13
5 19 4
Bottom pages
7 31 2
10 18 15
11 19 14
6 30 3

```

Figure 3.7 Folding program

The following table lists the variables and arrays used by the programs in this section, along with the purpose of each.

<i>Name</i>	<i>Item stored under this name</i>
-------------	------------------------------------

Variables:

N			The number of pages in the imposition
C	{		The number of columns (C), number of rows (R) and the depth (D) of the pages in a folded arrangement of pages
R			
D			
CO	{		Counter variables for the columns (CO), rows (RO) and depth (DE) of a folded arrangement of pages
RO			
DE			
PC	{		The number of columns (PC), rows (PR) and the depth (PD) of a previous arrangement of pages that is being unfolded to give a new one
PR			
PD			
LC	{		Counter variables for the columns (LC), rows (LR) and depth (LD) in the previous arrangement of pages
LR			
LD			
L			The position in the ordering of the page numbers of the number of a page at a given row, column and depth
PL			The position in the ordering of the page numbers of the number of a page at a given row, column and depth in the previous arrangement

Arrays:

P	To hold, in order, the numbers of the pages in an arrangement
T	To accumulate, in order, the numbers of the pages in a new arrangement

Classification and coding

The processes of classification and coding are closely related. Classification of sets of similar objects is usually achieved by providing a label for each. The label, whether or not it is a number, can be regarded as a code. The point of assigning the code is to facilitate the systematic storage and tracing of the objects.

A tree-like structure such as the one shown in Figure 3.8 is usually used for classification. From a tree such as this, the classification code for each item mentioned in the tree can be read

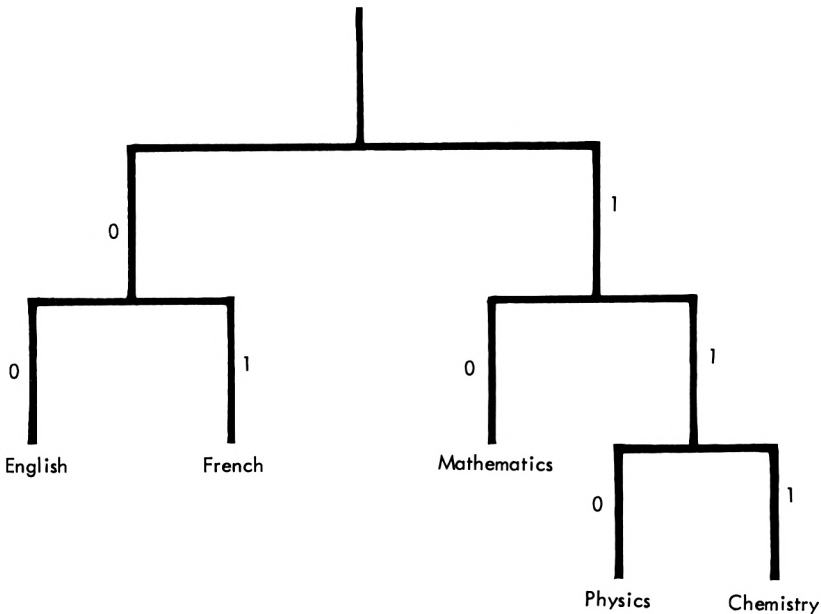


Figure 3.8 Tree for classification

by starting at the root of the tree and accumulating the digits or, more generally, characters of the code as they are passed in taking the path to the name of the item. The classification code for Physics, for example, can be read from Figure 3.8 as 110. By convention, the root of the tree is usually placed at the top, in contrast to the position of that of a natural tree. But the names of the parts of the tree still correspond with those of a natural tree, with the root leading to a trunk, which divides into branches. The branches divide further until at their ends they reach 'leaves' which are labelled with the objects to be classified.

The classification system by which the books in libraries are assigned their classification numbers is essentially of this tree-structured form. Similarly, the way in which an item of information is traced in a large publicly accessible information bank such as that of Prestel is by tracing the appropriate path through a tree of this kind. Always starting from the root, the user can repeatedly make a choice of branch, with guidance from the system, until the required leaf is reached.

We can write a program in Amstrad BASIC to store the information shown in Figure 3.8 with its tree structure. Once it is stored, we should be able to enter the name of an item and to have the program read its classification number from the tree which it has stored. To store the tree, we can start at the root of the tree and use pointers to point either to a branch or to an item

to be classified. After this, each branch must be given pointers of a similar kind until all the leaves of the tree have been dealt with. When the tree has been stored in this way, we can find the code for each of its items by tracing the path between the root and the corresponding leaf and noting the characters that label the route.

Since our tree grows exactly two branches at the end of its trunk and from all the other branching points, we need two arrays of pointers to mark the routes from the root through the tree. (Our tree is an example of what is known as a binary tree.) We shall call the arrays LB and RB to indicate that the elements of the first point to the left branch following a branching point and those of the second to the right branch. (The senses of 'left' and 'right' are as they would be from the point of view of a person viewing the tree as it is shown in Figure 3.8.) With the branching points numbered as shown in Figure 3.9, we can implement the pointers in the following way. From branching point number 1, the left branch leads to branching point 2 and the right branch to branching point 3. To represent this, we can make the assignments

$$LB(1)=2: RB(1)=3$$

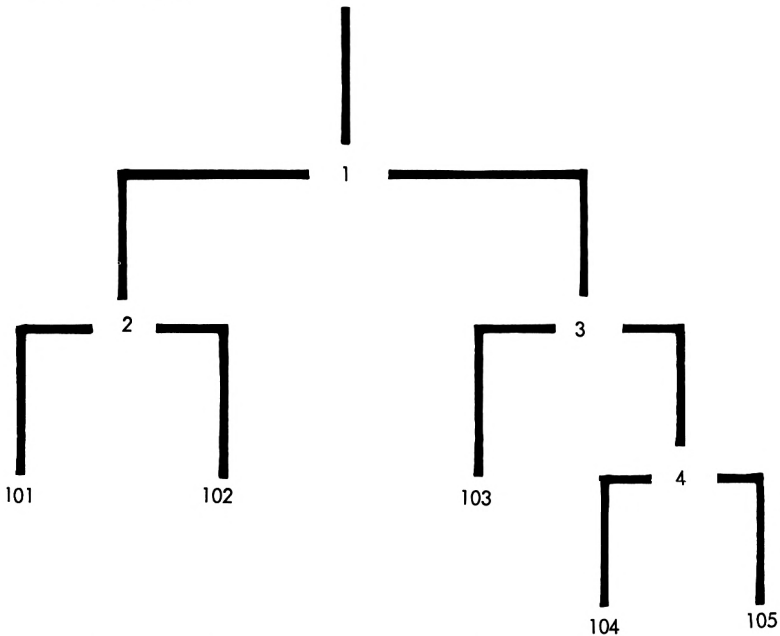


Figure 3.9 Tree with its branching points numbered

All the branches following a branching point can be pointed to in this way, but if a branching point is followed by a leaf rather than

another branch, we must have a way of showing this. The subjects to be classified can be stored in an array called S\$, so that each subject can be pointed to by assigning the number of the element of S\$ that holds it to LB or RB. In this way, after the assignment

$$S\$(3) = \text{"MATHEMATICS"}$$

we could write

$$LB(3) = 3$$

But, as things stand at present, this could also mean that after branching point 3 the left branch leads to branching point 3. To distinguish leaves from branching points, we shall add one hundred to their pointer numbers. There is no chance of confusion if we do this because there are far fewer than a hundred branching points so that a number greater than a hundred can never really represent a branching point. When a number greater than a hundred occurs, then, we know that we must subtract a hundred from it and that then it will point directly to a leaf. This will make the assignment for the pointer to the leaf labelled "MATHEMATICS"

$$LB(3) = 103$$

Once the tree is stored in this way, with the arrays LB, RB and S\$, the part of the program for finding the classification code for one of the items labelling the tree can be written. This will operate by tracing the path from the leaf labelled with that item to the root of the tree. The code will be accumulated as the route is traversed, with the first number to be encountered treated as the right-hand symbol or, if we are thinking numerically, the least significant digit. Since a zero is always associated with a left branch and a one with a right branch, we can add the appropriate character to the left of the character string that accumulates the code according to which branch is encountered.

The outline of the procedure for finding the code for a subject is:

Find the pointer, K, to the subject

Initialise code

Repeat

 Find J such that either $LB(K) = J$ or $RB(K) = J$

 Add 0 or 1 to the left of the string for the code accordingly

 Set K to J

Until the root is reached

The program based on these ideas is:

```

10 DIM S$(5), LB(9), RB(9)
20 LB(1)=2: RB(1)=3
30 LB(2)=101: RB(2)=102
40 LB(3)=103: RB(3)=4
50 LB(4)=104: RB(4)=105
60 S$(1)="ENGLISH": S$(2)="FRENCH"
70 S$(3)="MATHEMATICS": S$(4)="PHYSICS"
80 S$(5)="CHEMISTRY"
90 INPUT "SUBJECT"; T$
100 K=1
110 IF S$(K)=T$ THEN 140
120 K=K+1
130 GOTO 110
140 K=100+K: C$="": J=1
150 IF LB(J)=K THEN C$=C$+"0"+C$: K=J: J=0: GOTO 170
160 IF RB(J)=K THEN C$=C$+"1"+C$: K=J: J=0
170 IF K=1 THEN GOTO 190
180 J=J+1: GOTO 150
190 PRINT "CODE IS "; C$

```

The arrays and variables used in this program are listed, together with their purposes, below.

<i>Name</i>	<i>Item stored under this name</i>
-------------	------------------------------------

Arrays:

S\$	The names of the subjects to be classified
LB	The numbers of the branching points at the ends of the left branches of the tree
RB	The numbers of the branching points at the ends of the right branches of the tree

Variables:

T\$	The name of a subject as input to the program
K	The position in the array S\$ of the subject
C\$	The code for the subject
J	A counter variable to hold values for testing as candidates for K

Having started by writing a program to store a given tree and to read the codes from it, we can go a step further by writing a program which, when given subjects for classification, will construct their classification tree. A procedure for doing this is illustrated in Figure 3.10. The subjects are first written down in a

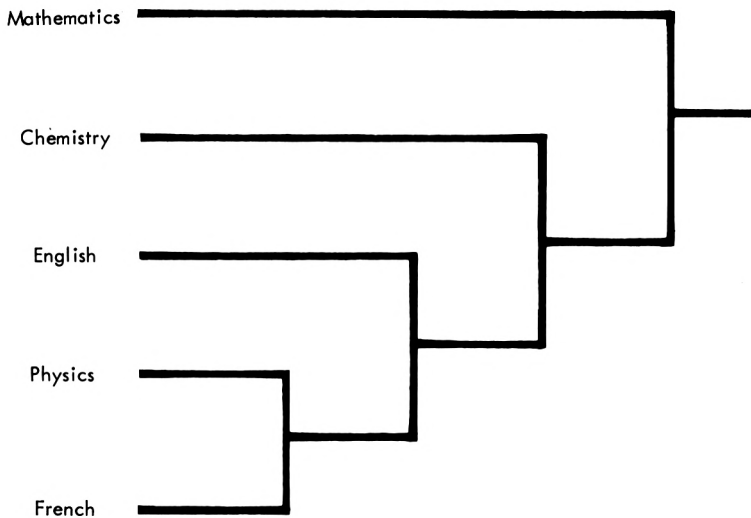


Figure 3.10 Classification tree for a set of subjects

list, then the bottom two are combined to give a new composite subject, and this is done repeatedly until all the subjects have been combined. A program that works by doing just this and which draws the resulting classification tree is listed below.

```

10 DIM S$(5), X(5), Y(5)
20 FOR K=1 TO 5: READ S$(K): NEXT K
30 CLS
40 FOR K=1 TO 5
50 X(K)=200: Y(K)=75*K
60 LOCATE 1, 25-Y(K)/16: PRINT S$(K)
70 MOVE X(K)+30, Y(K): DRAW X(K)+80, Y(K), 1
80 X(K)=X(K)+80
90 NEXT K
100 FOR J=4 TO 1 STEP -1
110 MOVE X(J+1), Y(J+1): DRAW X(J+1), Y(J), 1
120 Y(J)=0.5*(Y(J+1)+Y(J))
130 FOR K=1 TO J
140 MOVE X(K), Y(K): DRAW X(K)+50, Y(K), 1
150 X(K)=X(K)+50
160 NEXT K
170 NEXT J
180 DATA "ENGLISH", "FRENCH", "MATHEMATICS"
190 DATA "PHYSICS", "CHEMISTRY"
200 GOTO 200

```

The last line is needed to prevent the program from ending, so keeping the display on the screen.

The procedure on which this program is based is not the only way to produce a classification tree. An alternative method is shown in Figure 3.11. The question that naturally arises is which of the possible trees is the best one. The answer to this question is that it is best to give the shortest codes to the subjects that occur

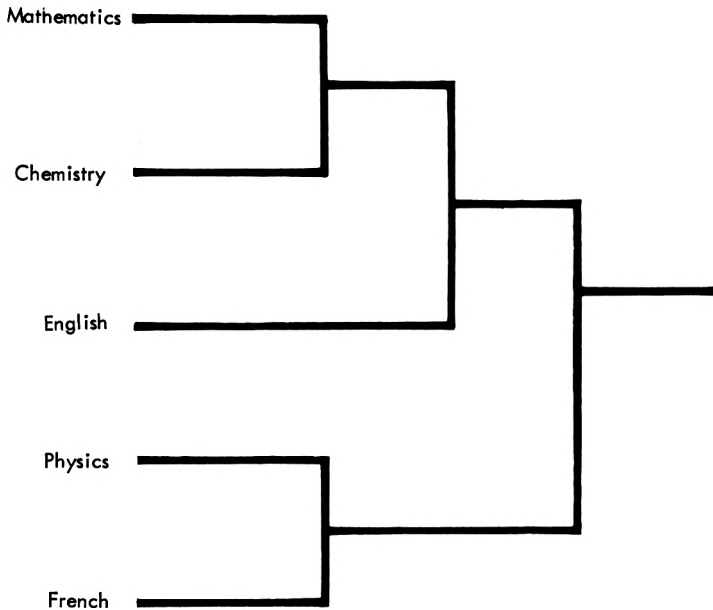


Figure 3.11 Alternative classification tree

most frequently and the longest to those that are the least common. This is clearly the best strategy when devising codes for storing information, rather than simply for classifying it, for giving the shortest codes to the items that must be stored most often ensures that the smallest amount of memory will be needed. With classification codes, it is also better to give the shortest codes to the most common items, for this will make the codes easier to remember and quicker to enter.

If we use the probability of occurrence of a subject as the measure of its frequency of occurrence, or commonness, there are two different ways of going about finding the resulting tree. One is to sort the subjects into decreasing order of their probabilities before using the procedure that we have just written. This works because the least common subjects are combined first and therefore are given the longest codes. The second method is to write a program to write the subjects down in the order in which they are held in the array $S\#$ and then to combine repeatedly the least probable, no matter which they are.

A program based on the first scheme is:

```

10 DIM S$(5), P(5), X(5), Y(5)
20 FOR K=1 TO 5: READ S$(K), P(K): NEXT K
30 GOSUB 1000: REM SORT ROUTINE
40 CLS
50 FOR K=1 TO 5
60 X(K)=200: Y(K)=75*K
70 LOCATE 1, 25-Y(K)/16: PRINT S$(K)

```



```

80 MOVE X(K)+30, Y(K): DRAW X(K)+80, Y(K), 1
90 X(K)=X(K)+80
100 NEXT K
110 FOR J=4 TO 1 STEP -1
120 MOVE X(J+1), Y(J+1): DRAW X(J+1), Y(J), 1
130 Y(J)=0.5*(Y(J+1)+Y(J))
140 FOR K=1 TO J
150 MOVE X(K), Y(K): DRAW X(K)+50, Y(K), 1
160 X(K)=X(K)+50
170 NEXT K
180 NEXT J
190 DATA "ENGLISH", 0.2, "FRENCH", 0.1, "MATHEMATICS", 0.3
200 DATA "PHYSICS", 0.15, "CHEMISTRY", 0.25
210 GOTO 210
1000 S=0
1010 FOR M=1 TO 4
1020 FOR N=M+1 TO 5
1030 IF P(N)<P(M) THEN 1070
1040 T#=S$(M): S$(M)=S$(N): S$(N)=T#
1050 X=P(M): P(M)=P(N): P(N)=X
1060 S=1
1070 NEXT N
1080 NEXT M
1090 IF S=1 THEN 1000
1100 RETURN

```

A program based on the second scheme is:

```

10 DIM S$(5), P(5), X(5), Y(5)
20 FOR K=1 TO 5: READ S$(K), P(K): NEXT K
30 CLS
40 FOR K=1 TO 5
50 X(K)=200: Y(K)=75*K
60 LOCATE 1, 25-Y(K)/16: PRINT S$(K)
70 NEXT K
80 FOR J=1 TO 5
90 FOR K=1 TO 5
100 IF X(K)>0 THEN MOVE X(K), Y(K): DRAW X(K)+50, Y(K), 1:
X(K)=X(K)+50
110 NEXT K
120 IF J=5 THEN 240
130 M=1: Q=1
140 FOR K=1 TO 5
150 IF P(K)<M THEN M=P(K): Q=K
160 NEXT K
170 M2=1: P2=1
180 FOR K=1 TO 5
190 IF P(K)<M2 AND K<>Q THEN M2=P(K): P2=K
200 NEXT K
210 MOVE X(Q), Y(Q): DRAW X(P2), Y(P2), 1
220 P(Q)=P(Q)+P(P2): P(P2)=2
230 Y(Q)=0.5*(Y(Q)+Y(P2)): X(P2)=-10
240 NEXT J
250 DATA "ENGLISH", 0.2, "FRENCH", 0.1, "MATHEMATICS", 0.3
260 DATA "PHYSICS", 0.15, "CHEMISTRY", 0.25
270 GOTO 270

```

The arrays and variables involved in the programs of this section following the first are summarised below.

<i>Name</i>	<i>Item stored under this name</i>
-------------	------------------------------------

Arrays:

X	{	Column (X) and row (Y) numbers for positioning the lines for drawing a tree on the screen
Y		
P		The probabilities associated with the subjects

Variables:

T\$	A subject name during swapping
X	A probability during swapping
S	A value to indicate whether swapping occurred
M	The smallest value in the array P
Q	The position in P of its smallest value
M2	The second smallest value in the array P
P2	The position in P of its second smallest value

Conclusions

In this chapter, we have posed and solved a number of problems each of which displays some degree of difficulty. One of the points that emerges, almost self-evidently, is that they can all be solved with Amstrad BASIC. Often, Amstrad BASIC possesses a feature that is exactly what we require to facilitate the solution of a particular problem.

Clearly, the addition of graphics to problem-solving programs can enhance their worth considerably, and Amstrad BASIC is quite well equipped to allow its users to create the necessary graphics.

If a problem is tackled in a systematic fashion so that the solution program has a clear and strong structure, then the addition of graphics to the solution program is straightforward, as is the addition of further refinements to enhance the usefulness and generality of the program.

Sometimes, Amstrad BASIC does not lend itself to the structuring of programs as well as other computer languages do. Languages such as Pascal are available for Amstrad computers and, if it is important to be able to find the solution of considerable problems in a way that can be extended and adapted, the investigation of such a language is well worth while.

It may seem that the use of a computer may not be necessary for some of the problems that are presented in this chapter. A computer is not really needed to classify five subjects, for example. But there can be no doubt that a computer would be a great help in classifying five thousand subjects. The program

presented here will do this, with simple and straightforward changes, although the demonstration of this would be extremely difficult in a book. So try to keep in mind the idea of 'scale': if a computer program will solve a problem with small amounts of data then (perhaps after minor modification) it will solve the same problem with any amount of data.

The main purpose of the chapter, though, is to demonstrate the sort of techniques that make the computer such a powerful tool for problem solving. The methods are undoubtedly different from those used in other areas, and their application requires a certain amount of ingenuity, but with their use the scope of the problems to which the Amstrad can be successfully applied is enormous.

4 Hardware-based applications

The range of applicability of the Amstrad can be enlarged enormously by attaching items of equipment to it. The equipment can range from something as obvious as a printer, through items such as video equipment which may already be in the home to, less obviously, a household robot. A printer is as necessary as the computer itself for the user whose application is word processing, although it has other uses. Attaching a computer to video equipment enhances the applications of both. A personal robot opens up a new area of interest in which the possibilities are only now becoming apparent, indeed are only just being imagined. Items of equipment such as these are attached to the Amstrad at one of the sockets on its back panel. The sockets are labelled to indicate their basic purposes. Each has its own pattern of electrical connections that makes it suitable for dealing with particular items because the item for which it is intended will have its own socket with a corresponding, matching pattern of connections. In this way, the socket marked 'PRINTER' on the Amstrad is of a standard kind that is known as a *Centronics* socket or port. Many of the types of printer that are manufactured include this type of connection, and so any of them can be connected directly to the Amstrad and operated with a minimum of fuss. (Other kinds of printer incorporate a different socket. While it is still possible to attach them to the Amstrad, the connection is not so straightforward.) The connection marked 'USER PORT' is not a widely recognised standard socket in the same way as the printer socket but, rather, is specific to the Amstrad. Nevertheless, it provides a well-defined pattern of electrical connections, and any item possessing the corresponding pattern can be plugged in here. Such items include the Amstrad joysticks.

Many of the items that can be attached to the computer will simply be controlled by it. The control will be achieved by passing

electronic signals from the computer to the attached equipment. This is not a particularly difficult departure for the computer. Signals are being passed around inside it all the time between its various component parts, and to lead some of these signals to a connection socket is as easy as to lead them anywhere else. But this does reveal the importance of having a well-defined pattern of connections between the computer and the item attached to it. If the computer sends a signal to one of its sockets for a specific purpose, then that signal must be conducted to the necessary place in the attached equipment to achieve that purpose. The use of standard sockets is naturally a great help in ensuring successful cooperation.

The one-way passage of signals, so that they only pass from the computer to the item attached to it, may be sufficient for a computer to control a printer. In this case it could be sufficient, for example, for the computer to indicate to the printer when it is to start printing, to pass the text to be printed, and then to indicate that the printer must stop. But if the computer is to guide a robot around the house signals must pass both from the computer to the robot and in the opposite direction. The computer must send the robot its directions as to how to proceed, but the robot must send signals to the computer to indicate that it has encountered some unusual situation with which it must cope. For example, if the computer has been provided with a description of the arrangement of the furniture in a room then it can guide the robot from place to place without its bumping into anything. But if there is a cat in the room and the robot is not to bump into it, then the robot must be able to detect when there is an obstacle in its path and, when there is, to request directions for avoiding it. The two-way transmission of signals can be achieved with success in the same way as one-way transmission, by agreeing on the pattern of the electrical connections that link the computer and the robot, or whatever, and the purpose of each.

The word *peripheral* is the jargon term for an item of equipment that can be attached to a computer. If you have a peripheral that you want to attach to the Amstrad and it does not have a socket matching one of those that is on the Amstrad, then it obviously cannot be plugged in directly, but there are still ways of attaching so that it operates successfully. The peripheral clearly cannot be plugged in if its plug is the wrong shape. But this is only a symptom of the more fundamental mis-match that we have been discussing, namely that the patterns of electrical connections do not match. The solution to this is to build an electronic circuit that takes the signals on the connections from, say, the computer and converts them to the signals needed by the

peripheral before routing them to connections that match the requirements of the peripheral. For two-way communication, it will also need to do the same thing for the signals that pass from the peripheral to the computer. An electronic circuit that performs these conversions is known as an *interface*. An interface to make it possible to connect a particular peripheral to a particular socket at the back of the Amstrad will incorporate two cables, emerging from the circuit in opposite directions, as it were, with one cable terminating in a connector to plug into the peripheral and the other in a connector that fits the particular Amstrad socket. For this reason, any peripheral for which an appropriate interface exists, or can be constructed, can be attached to the Amstrad and used in conjunction with it.

The peripherals that we shall examine and discuss in this chapter can be classified, fairly loosely, into four categories. The categories do overlap, but the purpose of giving them is to provide a clear indication of the sorts of applications that become available by adding extra equipment to the computer. The categories are:

- 1 Upgrading the computer. Certain peripherals can be seen as improving the capability of the computer as a computer. That is to say, they bring an improvement in kind, but not in the scope of the applications to which it can be applied. Adding a disk drive to the CPC464 is an example of this, in that it has improved the ability of the computer to do something that it could do before. It allows it to store more information and to store it more quickly. Adding a second disk drive to the CPC664 provides an upgrade in a similar way, by facilitating the manipulation or copying of substantial amounts of information, as anyone who has had to do it will rapidly confirm.

- 2 Enhancing the computer. Some peripherals can enhance the computer by bringing to it abilities that it did not have before. A printer is one example, and a graph plotter is another.

- 3 Making the computer easier to use. Many peripherals are available that make the computer easier to use. They often do this by providing alternatives to the keyboard for communicating with the computer. Joysticks are one example of this, and the light pen and the mouse are others. But all of these can be used as more than keyboard alternatives.

- 4 Opening up new applications. Although the creative use of items for enhancing the computer and of those for making it easier to use can be seen as making new areas of application, the addition of a robot to a computer opens up vistas of opportunities that represent much more radical departures. Similarly, attaching

video equipment to the computer brings promises of interesting and exciting applications.

Cassettes and disks

Cassettes and disks are the media on which information is stored. The information may be a computer program, data that has been placed in a database, text from a word processor or any of a great many other things. No matter what its origin, when stored the information is represented as a succession of magnetic marks on the magnetisable surface of a disk or of the tape in a cassette. The magnetic patterns representing information are recorded on a cassette by a cassette player and on a disk by a disk drive.

The Amstrad CPC464 has its own built-in cassette unit, while the CPC 664 has a built-in disk drive. We have seen in Chapter 2 that the CPC464 can be used for word processing, and for database and spreadsheet applications, but the ability to be able to use only cassettes for storage does impose some restrictions. As applications programs come to offer more, and more sophisticated, facilities, they become larger and, in consequence, take longer to load from cassette. As the amounts of information that are generated increase, they also take longer to save on a cassette, and further complications arise when too much information is created for it all to fit on a single cassette. Thus, as the demands that the user places on the system increase, there will be a natural desire, in the interests of convenience and time-saving, to upgrade the system to include a disk drive. Disk drives allow information to be stored and recovered much more quickly than a cassette player ever can. This is partly because they are inherently faster in operation, but also, as far as recovering information is concerned, because they can access any items of information as quickly as any others, which a cassette player cannot. A disk drive reads the information on a disk by using a reading head that is mounted on an arm which operates in much the same way as the arm carrying the playing head and stylus of a record player does when playing an LP. Just as a record player can play one track on an LP as easily as any other, so a disk drive can retrieve any information on a disk with equal ease. By contrast, a cassette player must wind through all the unwanted information that precedes the information that is required from a cassette before it can begin to recover it.

A disk drive can be attached directly to the CPC464 via the socket marked 'FLOPPY DISK'. Figure 4.1 shows Amstrad's own disk drives attached to a CPC 464. When this has been attached the system can, of course, run disk-based software. But it is worth

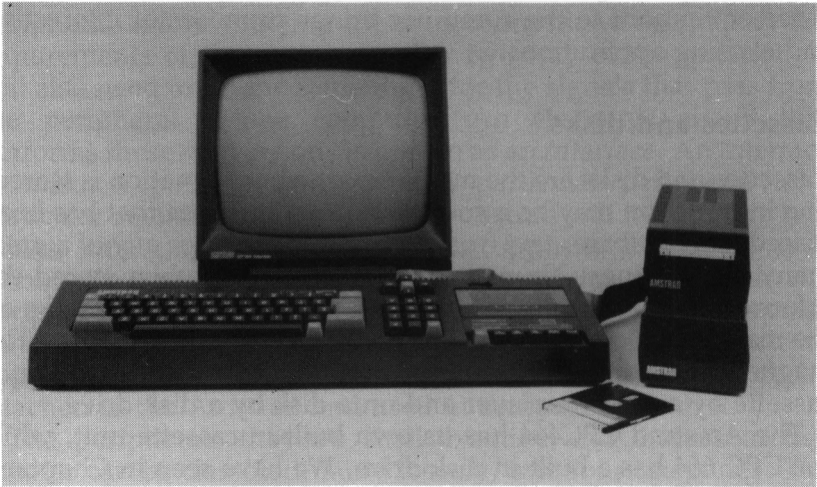


Figure 4.1 Two Amstrad disk drives attached to a CPC464

taking care, when up-grading in this way, to ensure that the disk-based software that is purchased is compatible with the cassette-based software so that all the work done previously is not lost. If the new software is compatible with the old, then it will be possible to load information held on cassette with the use of the cassette unit, and then to save it on disk, so that there is no problem in incorporating information stored previously into the new mode of working. Above all, there need be no hiatus when starting with the new system while the information created with the old system is re-created with the new one.

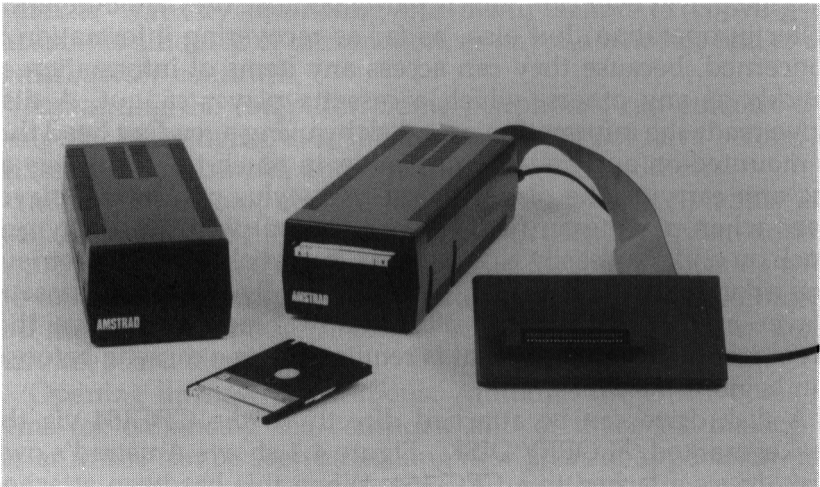


Figure 4.2 The 3 inch disks, drive units and interface used with the Amstrad CPC 464

The CPC664 has its own disk drive which, in common with the stand-alone drive, accepts 3 inch diameter disks of the type shown in Figure 4.2. It has a socket at which a domestic tape recorder can be attached, so that up-grading from a CPC464 to a CPC664 can be achieved with the same safeguards as have just been described for up-grading a CPC464 by adding a disk drive to it. But once the user is familiar with the CPC664 and starts to become more and more ambitious with its application, even a disk drive will be found to have its limitations. Part of the reason for this is that it is good practice to ensure that an applications program is held on one disk and the data it produces on another. This ensures, for example, that the applications program is never accidentally overwritten by some data, and that if one data disk is filled it is only necessary to start on another one. But with a single disk drive, the user will often have to remove one of the disks to replace it by the other, particularly if the applications program is too large to reside in the computer in its entirety. In this case, the computer will call in the parts of the program from the disk as they are needed. But if the data disk is in the drive at the time, it will be necessary to swap disks.

It is probably not difficult to imagine from this discussion that it is very useful to have two disk drives so that one can hold the disk with the applications program and the other the data disk. Another much-needed operation in which two disk drives come into their own is in making a copy of a disk. This will be needed, for example, with any new disk to make a back-up copy of it for security purposes in case any harm comes to the version that is in use. Because the capacity of a disk far exceeds that of the computer's memory (which will not all be available anyway because it must also contain the program that does the copying), the computer must repeatedly take chunks of information from the disk and copy them to the other disk. With the original disk in one drive and the disk on which the copy is being made in the other this is straightforward, but with only one drive disks will have to be inserted and removed rather too often for comfort.

Figure 4.3 shows the CPC664. With a second disk drive attached to it, this represents the culmination of what can be based on the Amstrad as far as external storage requirements are concerned, and it is undoubtedly a powerful one. Because the CP/M operating system can be used with the CPC664, a vast range of software is available for it from the CP/M library in addition to that which has been developed especially for the Amstrad. Having twin disk drives allows the users of the system to take full advantage of the capabilities of this software. By the route that we have described, there is an Amstrad system with



Figure 4.3 The Amstrad CPC664

adequate storage facilities to meet the needs of any user from the beginner to the most demanding.

Printers and plotters

As we have said, any printer with a Centronics socket can be connected directly to the Amstrad. Printers in this bracket cover the entire range of what is available, so that although a printer with another type of socket can be connected via an appropriate interface, there is not too much point in doing so because such a printer can offer nothing that is not possible with a Centronics printer. In fact, a Centronics printer has one positive advantage over its main rivals, which stems from the way that signals are passed between the computer and the printer. A Centronics connection passes eight signals simultaneously, or *in parallel*. Since each signal can represent a bit, eight bits, or one *byte*, can be sent at a time. And one byte can represent a *character*, that is, a letter, number, punctuation mark or special symbol, to be passed from the computer to the printer. By contrast, the connection used by other printers passes only one signal at a time, so that bits

are sent one at a time in a continuous stream. (They are sent *in serial*.) This means that the characters making up information can be sent to a Centronics printer more quickly than to other printers and, correspondingly, that it is possible for such a printer to print the information more quickly.

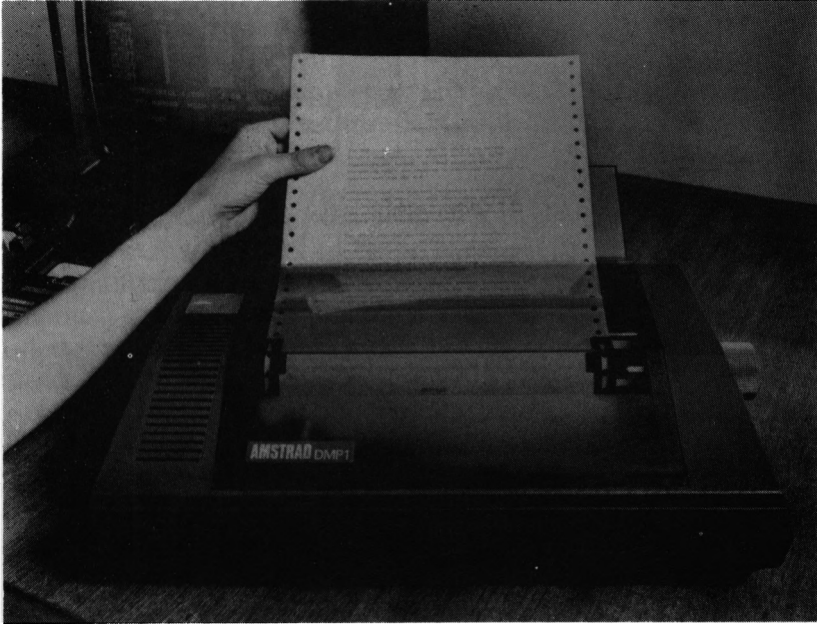


Figure 4.4 The Amstrad DMP-1 dot matrix printer

The printers available are basically of two kinds, known as dot matrix and daisy-wheel printers after the way that they print their characters. A dot matrix printer, such as the Amstrad DMP-1 shown in Figure 4.4, prints its characters as patterns of dots. A number of small needles arranged in a rectangular array can strike the print ribbon. By allowing some to strike it and not others, a character is printed as the resulting pattern of dots as shown in Figure 4.5. A daisy-wheel printer, such as the Silver Reed model shown in Figure 4.6, prints complete letters by striking their moulded shapes against the print ribbon. The printer operates by rotating the wheel to bring the required character into position and then striking the character against the print ribbon so that it makes its imprint on the paper.

The two kinds of printer produce results of different qualities, and also have a few distinct operating characteristics. It is sensible to review the uses to which a printer may be put to help in deciding which printer is more suitable for the different applications.

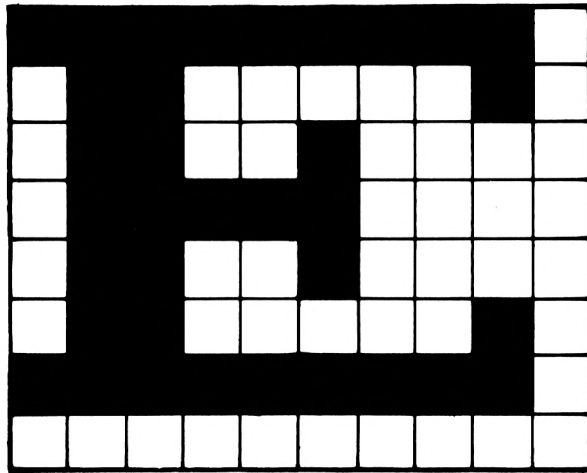


Figure 4.5 A character produced as a pattern of dots



Figure 4.6 A daisy-wheel printer

Word processing

The most widespread practical use of a printer is in word processing. Both dot matrix printers and daisy-wheel printers will produce perfectly acceptable and readable notes, letters and reports. For purposes of prestige, the user may feel that in some circumstances the letters and documents that he or she produces should have an appearance comparable to that which results from

using a good electronic typewriter. In this case, a daisy-wheel printer must be used to give the high quality, or so-called 'letter quality', standard of printing. The prejudice that has existed towards the results produced by dot matrix printers is breaking down, however, and as the quality of their printing improves, it becomes increasingly difficult to distinguish between the output of a good dot matrix printer and a daisy-wheel printer.

When using a daisy-wheel printer, it is possible to use different founts for the printing simply by changing the daisy-wheel to one that supplies the fount required. A range of daisy-wheels will be available giving different founts, such as Courier, Elite and Gothic Mini, as well as for italics. There may also be daisy-wheels providing the Greek alphabet and mathematical symbols. This means that documents can be produced with an appearance and of a quality that meet the highest requirements of the user. Technical documents can also be printed.

A dot matrix printer can also offer different founts, although it does not do so with the same flexibility as a daisy-wheel printer. By changing the pattern of dots that is used to create each character, a dot matrix printer can vary the appearance of the characters that it prints. To call this 'changing founts' would be an exaggeration, because the inherent limitation of a rectangular array of dots scarcely permits the creation of the letters of an accepted fount. But the change is possible. In contrast to the physical changing of a daisy-wheel, the change is usually made by software.

Another feature of daisy-wheel printers is that they can provide what is known as 'proportional spacing'. In essence, this means that they can provide spaces of various widths to separate letters. Such spacing is used in producing the text of a book or newspaper, so that it contributes to what we associate with the 'professional appearance' of such text, whether or not we are aware of its use. Proportional spacing comes into its own when text has to be right justified. It allows the extra space that must be inserted to push the right-hand end of the line far enough across to be uniformly distributed throughout the line. In the same circumstances, a dot matrix printer can only justify its text by inserting as many standard character-width space characters as are necessary. This can lead on occasion to lines of text acquiring an ungainly and obtrusive appearance that is distracting to the reader.

It is probably fair to summarise all this by saying that daisy-wheel printers are superior for the production of high quality letters and documents, although dot matrix printers are entirely adequate. Daisy-wheel printers are also more expensive, and so

the decisive basic point in choosing a printer for word processing may be whether you can afford to buy a printer that produces documents that meet your aspirations.

Programming support

As soon as programs of a certain complexity are being developed, a printer is an almost essential aid to the programmer. To develop or debug a lengthy program it is useful, and even necessary, to be able to see the entire program. The screen can show only a limited amount of it, while all of it can be printed. Besides this, a printed copy of a program can be taken away from the computer and studied at leisure. If a program can only be read on the computer's screen while sitting in front of it, there is a great temptation to tinker with the program by trying out anything that comes to mind. If a copy of the program can be taken away from the computer, it can be examined coolly and without the immediate pressure of the computer's presence. The second of these practices is much more likely to result in the development of a coherent and well-structured program. The computer will also be free for someone else to use.

With any kind of printer attached to the Amstrad, the program stored in it can be listed on the printer simply by giving the command:

LIST#8

The results given by a program can also be printed. This will preserve them in an easily accessible form, in contrast to their ephemerality if they are simply displayed on the screen. Before a program is developed to a completely satisfactory state, it will also be of help in correcting it, in the same way as is the ability to print the program itself.

Again, no matter what kind of printer is attached, the output of a program may be directed to it with commands such as:

PRINT#8, "Results for display on the printer"

On the face of it, there will be nothing to choose between the two types of printer for this usage and, indeed, for most of the time there will not be. But there are slight differences. The user may need some of the special characters that are possessed by the Amstrad or may design his own characters for some reason in an application. These characters can be printed by a dot matrix printer, after giving it the necessary directions, because the characters are created using a dot matrix on the screen (this time a

character is displayed by brightening certain dots and not others) as well as on a printer. A daisy-wheel printer cannot display such characters because it can only print the fixed repertoire of characters that exists on its daisy-wheel. The same applies to graphics characters, but we shall discuss them next.

The Amstrad provides a considerable repertoire of special characters and graphics characters. They are all illustrated in Appendix 3 of the Amstrad manual, but Figure 4.7 shows one example of each. The provision of some Greek characters and some mathematical symbols does weaken the advantage of daisy-wheel printers over dot matrix printers argued in the preceding

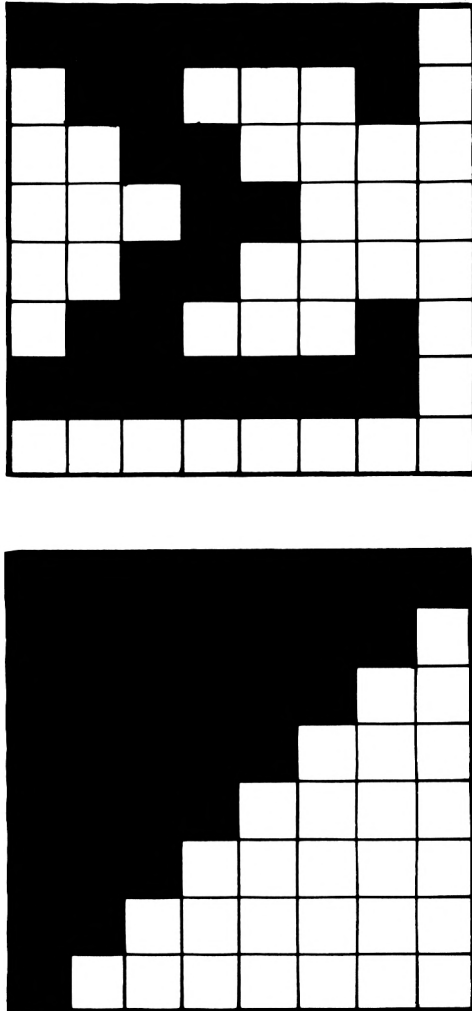


Figure 4.7 A special character and a graphics character

discussion of word processing. But the sets of characters supplied are not complete and, things being what they are, the one that you need is likely to be among those not provided!

The conclusion here must be that a dot matrix printer is likely to be preferable.

Graphics

The Amstrad can, as we know, display detailed graphics. The images created on the screen can be printed on paper by a dot matrix printer, but not by a daisy-wheel printer.

One way to create a graphics display on the screen is to PRINT the graphics characters that are required to make it, thereby creating an image in much the same way as a paragraph of text is created by printing its individual letters. The characters in Figure 4.7, for example, can be displayed by:

```
PRINT CHR$(190)
```

and

```
PRINT CHR$(212)
```

An image created in this way can be printed by a dot matrix printer in the same way as it prints text. It simply needs to print the characters as they are sent to it. No special arrangements are needed at all.

Even when the high-resolution graphics commands, such as DRAW and PLOT, have been used to create a display, it can still be copied by a dot matrix printer, although some software will be needed. The display on the screen is composed of dots, and if the screen is divided into blocks of eight by eight dots (the size of the dot matrix for the characters) it can be regarded as being composed of characters. These characters will, in general, not be those that are in the standard character set. But if software is provided that can describe each character into which an image can be divided and can instruct the printer as to how to print it, then high-resolution images can be printed in the same way as images that were originally constructed from characters.

For graphics, then, a dot matrix printer is capable of printing any of the images that can be created on the screen of the Amstrad.

However, serious graphics programmers wanting a permanent copy of their results will undoubtedly find that a graph plotter, such as the one shown in Figure 4.8, will be necessary to do full justice to their images. A graph plotter operates by moving a pen

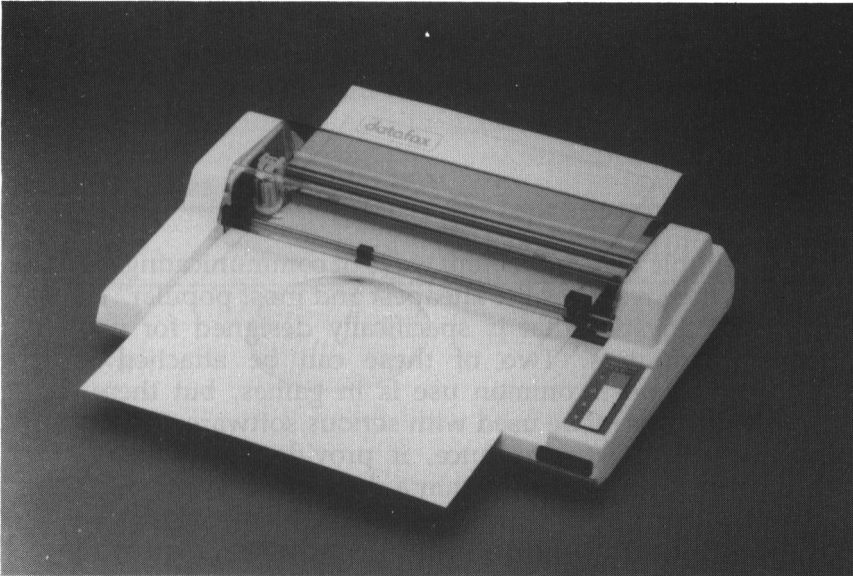


Figure 4.8 A graph plotter

over a sheet of paper. The pen is mounted on an arm: the arm can move across the paper and the pen can move up and down the arm. This allows the pen to be positioned at any point on the paper. If the pen is down while the movements take place it will draw a line. If it is raised, it can be moved to a new position without drawing a line and, when lowered, can start to draw a new line from there. If the plotter has a number of pens of different colours, then it can draw multi-coloured graphics by placing the pen of the appropriate colour in the pen holder at any time.

The ability to reproduce multi-coloured graphics gives a graph plotter an advantage over an ordinary printer, which can only produce its results in black and white. The plotter can also produce results superior to those displayed on the screen, for its resolution is superior to that of the screen. On the screen the positions of the dots which can be illuminated to create an image are fixed. So when plotting a straight line, for example, there may not always be a dot that is positioned directly in the path of the line. When this happens the dot that is nearest to the path must serve. This can lead to 'straight lines' that have a noticeably jagged appearance: the effect is particularly noticeable with lines that are nearly vertical and nearly horizontal. While a graph plotter does not draw smooth lines, apart from in certain exceptional cases, it does draw lines that are less jagged in appearance. (In fact, a graph plotter is operated by stepper

motors, with one to move the arm holding the pen by a small step at a time, and a second to move the pen along the arm in small steps. These steps are far smaller than the distance between adjacent dots on the screen.)

Joysticks, light pens and mice

Of all the simple and convenient ways of communicating with the computer, the joystick is the cheapest and most popular. There is an Amstrad joystick that is specifically designed for use with Amstrad computers. Two of these can be attached to the computer. Its most common use is in games, but there is no reason why it cannot be used with serious software. When using a word processor, for instance, it provides a natural way of moving the cursor around within a document.



Figure 4.9 A joystick

The Amstrad can detect whether the joystick has been moved left, right, up or down, and can also tell whether one of the two buttons on the joystick has been pressed. The joystick is treated by the Amstrad as part of the keyboard, so that it is scanned 50 times a second, and its activities can be tracked by using the INKEY command. The special function JOY is also provided,

though, and this function returns the values given in the following table when the different actions that are possible have occurred.

Table of values returned by JOY

Activity	Value of JOY(0)
Stick up	1
Stick down	2
Stick left	4
Stick right	8
Button 2 pressed	16
Button 1 pressed	32

Using this table, we can write a BASIC instruction to cause the screen to be cleared when the joystick is moved upwards as:

```
IF JOY(0) = 1 THEN CLS
```

Clearly, instructions similar to this can be included in any program, so that the joystick *can* be used in any application

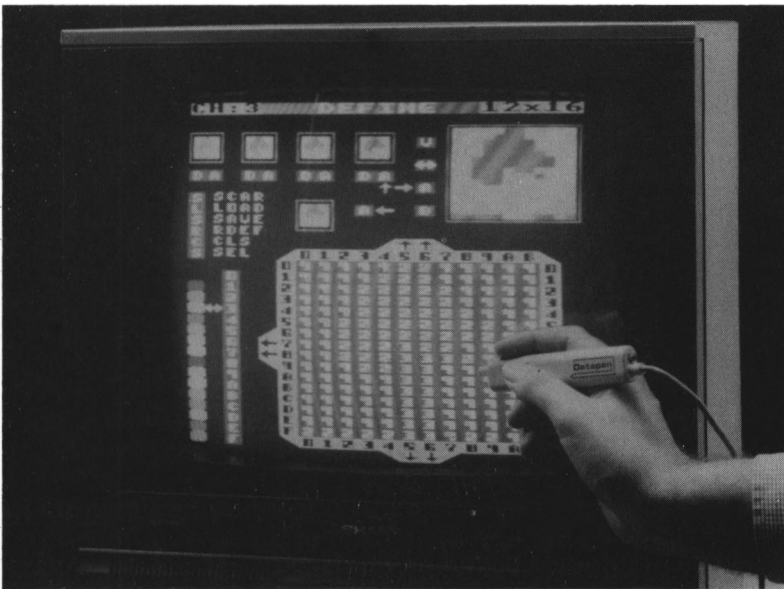


Figure 4.10 A light pen

although, with only six distinct states, its useful range may be rather limited. It could be used, for example, to select an item from a list, or *menu*, of up to six choices, but it must be admitted that this does not suggest that it provides a means of doing so that can be described as natural. But if the joystick is used to move an indicator progressively down a menu and then to show that the current item is to be selected, its use becomes rather more natural and appropriate.

A light pen such as that shown in Figure 4.10, provides a more natural means of interacting with the computer in that its use corresponds quite closely to familiar activities such as pointing and writing. Used in conjunction with the necessary software a light pen can be used to select an item from a list displayed on the screen by pointing at it, and can also be used to 'draw' on the screen in a way that is entirely analogous to the way that we draw on paper with a pencil.

One difficulty with using a light pen is that the very act of holding it to the screen inevitably obscures some part of the screen. One alternative that overcomes this problem is the graphics tablet, illustrated in Figure 4.11. This consists of a pad and a stylus which, again with the appropriate software, can be used in the same way as a light pen with a screen. By using overlays designed to accompany a particular applications program with the tablet, the use of the programs can be made particularly easy. The overlays can, at their simplest, have a box for each activity that the program provides, and any activity can then be selected and activated by placing the stylus in its box.

Another alternative means of interaction, which again avoids the problem associated with the light pen, is the mouse,

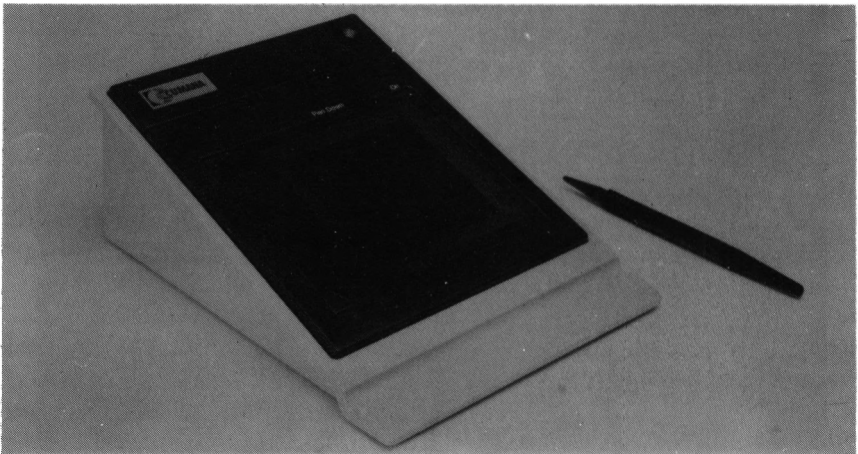


Figure 4.11 A graphics pad

illustrated in Figure 4.12. A mouse is only one part of a complete approach to making the computer easy to use. Other components are *icons* and *windows*. An icon is a graphical representation of an activity that the computer can carry out. A small picture of a folder can be used to represent the activity of filing information, and an image of a waste paper basket can be used to represent the discarding, or throwing away, of unwanted information. A mouse is used by moving it around on a flat surface at the side of the computer to which it is attached. As it is moved, it controls a cursor that moves in a corresponding fashion on the screen. By placing the cursor on an icon and pressing the button on the mouse, the activity represented by the icon is both selected and activated. With the cursor placed on the waste paper basket icon, for instance, clicking the mouse will cause the information that is currently being worked with to be thrown away.

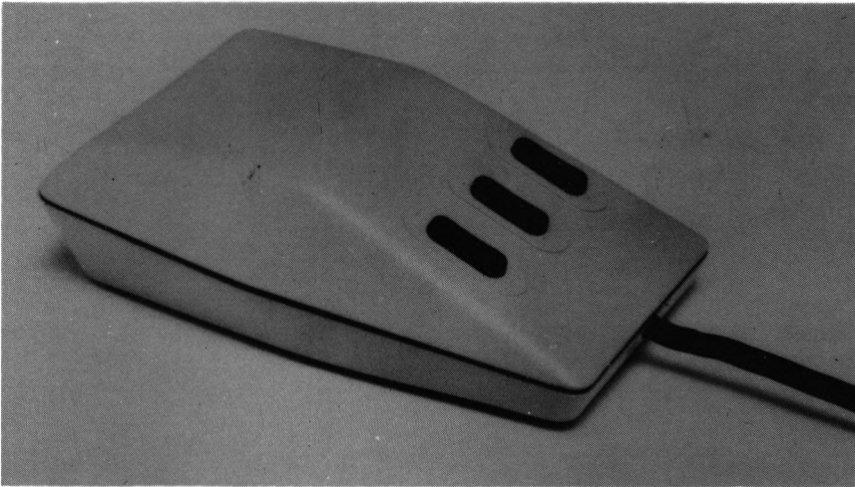


Figure 4.12 A mouse

A window is a rectangular region on the screen in which information can be displayed independently of the rest of the screen. Creating a window will cause it to overwrite a part of the display already on the screen, and removing it will cause the original display to reappear. The use of a window with a mouse and icons can be illustrated in the following way. Suppose that we are in the middle of the activity of creating a file of information, and we need to include some information that has already been stored in another file. What we need to do is to initiate the action of transferring information from another file to our current one. This will be an option in a menu provided by the program. So we place the cursor on the menu icon and press the

button on the mouse. The menu then appears in a window overwriting part of the display. Then we select the activity that we require, this time by placing the cursor on its name and pressing the button. We will have to give the name of the file containing the information that we need, but then the system will do the rest. When this activity is completed, we can place the cursor on the 'put away the menu' icon, and then the window containing the menu will be removed, leaving us with the display that we had before we began this activity.

This discussion is intended to show that there is a whole range of peripherals all of which can make the computer easier to use in varying degrees. They replace the keyboard as our basic means of communicating with the computer and, generally speaking, the more natural they are to use, the easier they make it for use of the computer.

Robots

In many ways, a robot is the most attractive of all the peripherals that can be attached to the computer. It provides something that the user can control, and which can immediately be seen to respond to the instructions it is given. The robots that can be operated in conjunction with personal computers may not yet be capable of enormously useful activity, but they do have their uses, and their usefulness is certain to increase in the future. At present, with a certain amount of ingenuity, there are some



Figure 4.13 The 'turtle'

worthwhile applications for robots that can be devised and implemented.

The robots that are available for use with home computers range from the 'turtle', shown in Figure 4.13 through the 'buggy' to the Zero 2 in Figure 4.14. We will discuss the capabilities of the turtle and the Zero 2, for in their applications they represent two very different approaches to robotics. The turtle is intended as an educational aid, and is found mainly, if not exclusively, in schools and other educational institutions. The Zero 2 is marketed as a personal robot for use in the home.

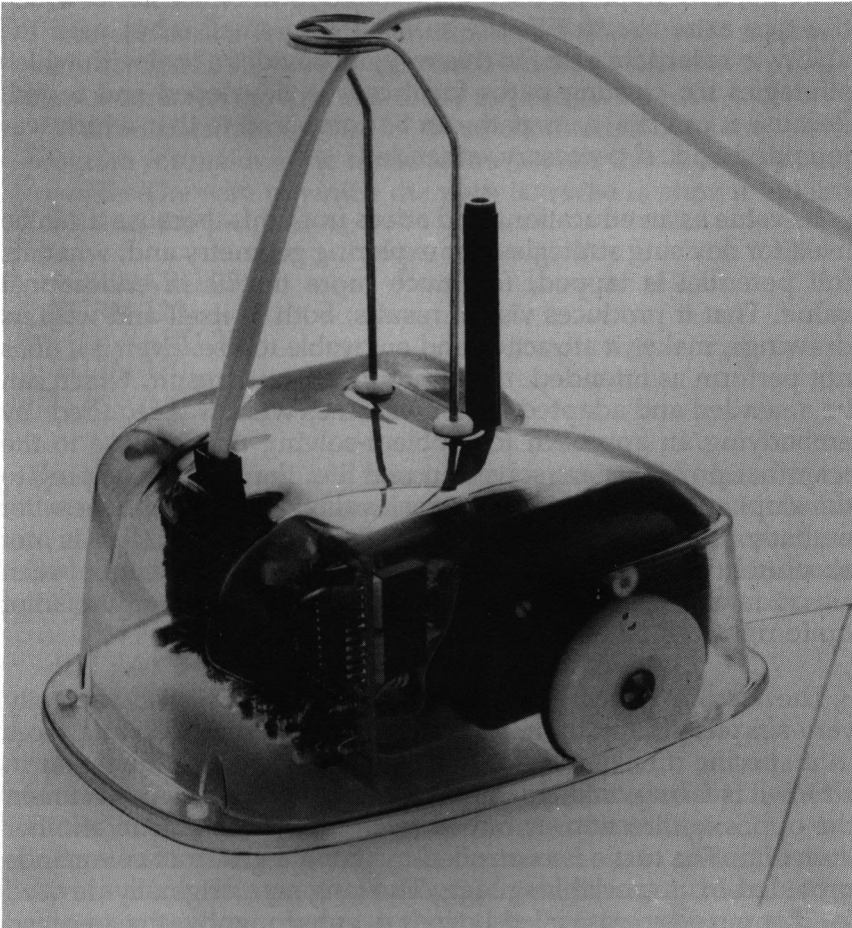


Figure 4.14 Zero 2

Educational robots

Examination of Figure 4.13 will show that, under its dome, the turtle possesses a pair of wheels, which enable it to move

forwards and backwards, and to turn. It also has a pen, which can be raised and lowered. When the turtle is placed on a large sheet of paper, it can draw on the paper when its pen is lowered, and so can draw lines and curves that mark its path.

The turtle, then, can be moved around under the control of the computer, perhaps following some path devised by its user. But it can also draw as it goes, thereby creating a record of the path that it has taken. This allows the turtle to be used, at one extreme, as a small robot whose movements can be controlled or, at the other extreme, as a graph plotter. Its intended form of use lies between the two extremes, for it can be used as a small robot with the ability to record its path. In this way, it provides a tool with which strategies for creating paths for it can be developed and tested. Because it can draw, its path can be compared to that which was intended and, if necessary, amended.

Its value as an educational aid arises from this, because it can be used for devising strategies, for exploring geometry and, when its full potential is tapped, for much more that is of educational value. That it produces visible results, both in itself and with its drawings, makes it attractive and enjoyable to use. Even if it does not perform as intended, it still produces some result, which can be amended and adapted until it becomes what was required. By embodying an approach to problem-solving that is close to the way that problems are solved in real life, the turtle is helping to develop and teach a skill of real value. Further, it does not embody the often-found attitude that something that is not absolutely right is, therefore, wrong: it deals with the in-between situation and encourages the attitude that something that is not quite right can be fixed so that it becomes right.

The way in which the turtle can be moved is fundamentally very simple. At any time, it occupies a known position and faces in a specific direction. It can move forwards in the direction in which it is facing, taking 'steps' of a certain length. It can retire in the opposite direction. It can be made to turn to face in another direction. The turtle is controlled by using a group of commands provided by a special language. The language originally devised for this purpose was called Logo but, subsequently, the so-called 'Turtle graphics' commands have been incorporated in other languages. No matter what language they are provided by, the commands for controlling the turtle are, with slight variations from language to language and dialect to dialect, as shown in the following table.

Table of turtle commands

<i>Command</i>	<i>Effect of command</i>
FORWARD n	To move the turtle forwards by n steps
BACK n	To move the turtle backwards by n steps
TURN a	To turn the turtle clockwise through a degrees
PENUP	To raise the pen
PENDOWN	To lower the pen

There are also a few other commands with obvious meanings, such as REPEAT, which can be used to make the writing of instructions for moving the turtle more convenient.

We can illustrate the use of these commands by giving a short program for making the turtle follow a path that is an equilateral triangle. The way in which the path is traced is shown in Figure 4.15, and the program is:

```
FORWARD 25
TURN 120
FORWARD 25
TURN 120
FORWARD 25
TURN 120
```

The last TURN command is not strictly necessary as far as tracing the triangular path is concerned, but it does bring the turtle back to its original state in that it occupies the same position and faces in the same direction as it did before it started. It also means that the program corresponds exactly to the following shorter version:

```
REPEAT 3
  FORWARD 25
  TURN 120
```

In both cases, the program should be preceded by the PENDOWN command if the turtle is to draw the triangle.

As a second example, we can consider how to make the turtle draw a circle. The previous program can give us a clue as to how this may be done, because the turtle has traced an equilateral triangle by advancing and turning three times. To move forward and keep turning by a constant amount is also the recipe for tracing a circle, for a circle has the same curvature at any point on its circumference. If the steps are as small as possible, and the amount of turn is also the smallest that it can be, then the circle

will be as smooth as is possible. This means that the turtle should repeatedly FORWARD 1 and TURN 1. How many times should it do this? Well, it turns through one degree with each turn, and there are 360 degrees in a circle, so 360 times ought to be enough. This gives the program for drawing a circle as:

```
PENDOWN
  REPEAT 360
    FORWARD 1
  TURN 1
```

This illustration shows the sort of reasoning that is required to make a turtle do what you want it to. The result of creating a shape by telling a turtle how to move along a path of that shape leads to a new approach to geometry which can be seen as a computational geometry. But it also provides an attractive test bed for trying out ideas and, if they do not work at first, for amending them. In this, it creates a powerful educational environment.

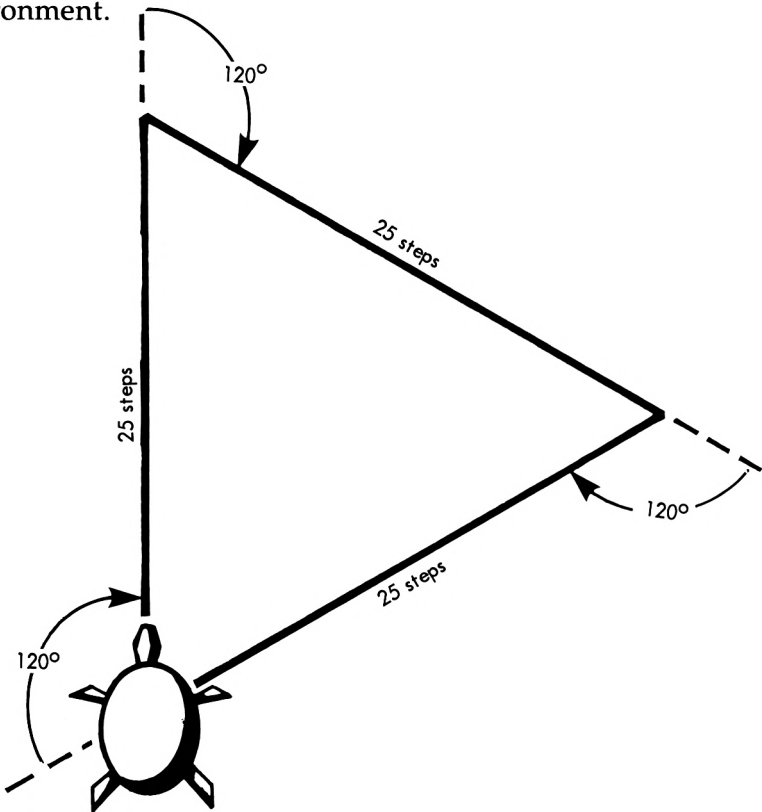


Figure 4.15 The way the turtle draws a triangle

The buggy is basically the same sort of robot as the turtle, with the same intended area of use. It is equipped with more facilities than the turtle, having an armoury of sensors, including touch-sensitive ones with which it can detect an obstacle it has run into and light-sensitive ones which can measure the brightness of a surface. This allows the buggy to be programmed to do more than the less well endowed turtle. It can be programmed to find its way around an obstacle in its path; to explore its environment, and create a map of it by recording where all the obstacles are located; and to follow a painted line by staying on a path on which the brightness of the surface under the light sensor remains constant.

Personal robots

There are a great many tasks that a robot can usefully do in the home, and even more that the harassed housewife might like one to do. The reason that robots are not yet commonly to be found doing jobs around the house soon becomes apparent on inspection of those jobs. Washing up, making the beds, Hoovering, mowing the lawn and so on are all tasks not only of some difficulty in themselves but also quite different from each other. If you doubt that, say, doing the washing up is a complex task, then consider how you would go about telling a robot how to do it. The robot's program that gives it the ability to wash up must do just this. And even when the robot can do the washing up, this ability will not help it very much when it comes to making the beds.

The tasks that a small robot can do at present include roaming round the house while the owners are away to give the impression that the house is occupied, to deter burglars; cutting material to a shape from a pattern, by following a path with a cutting tool lowered rather than the turtle's pen; feeding the cat, by dispensing its food at regular intervals; and playing a tune on a piano. These are fairly trivial tasks, in the context of a useful household robot, but they represent the first steps in the journey towards such a thing.

The most cursory glimpse at the Zero 2 in Figure 4.14 will show that it has little chance, in the form illustrated there, of carrying out any worthwhile task at all. In fact, the illustration shows it in its basic form in which it can do approximately as much as a turtle can. The idea behind this robot is that it can have modules added to it, each of which will provide it with an extra capability. So, the module for a particular task will consist of the mechanical

appendage required to carry out the task, be it an arm, a gripper, a sensor, or anything else, and the accompanying control circuitry and the software. From the lack of similarity shown by any household task to any other, it is clear that this modular approach to providing a robot with extra abilities is eminently sensible, and probably even essential.

Other equipment

Other equipment that the computer can be made to control to good effect includes a number of items that can be found in the home. Video recorders can be controlled to real advantage, allowing the user to edit and create his or her own films. The computer can also be used to add sub-titles to a film. Individual pictures can be acquired, as they can from a television set, for incorporation in a program. Whether for educational purposes or a game, a television frame provides a more realistic and detailed image than it is possible to produce with graphics programming without the expenditure of a very great deal of time and effort. In the future, compact disks are likely to be used in the same way as video can be and, with their enormous storage capacity, disks are likely to provide almost unlimited amounts of external storage.

Hi-fi equipment can also be linked to the computer to improve the quality with which computer-generated sound and music are reproduced. Electronic keyboards can be operated in conjunction with the computer. In general, any item of electronic equipment can be connected to the computer which can then be used to enhance its performance in some way.

Summary

There is a wide variety of hardware that can be attached to the computer. This can extend the power of the computer for the applications in which it is already used, make the computer easier to use and, most importantly, open up new areas of application.

The ability to link the computer to other electronic goods in the home, such as video and hi-fi equipment gives a way of getting more from that equipment. All the items that can be attached to the computer have considerable educational potential, whether intentionally, as with the turtle, or incidentally by incorporating and demonstrating the principles of modern industrial practice as based on the innovations of information technology.

As your use of the computer becomes more ambitious, and as you become increasingly aware of what it is able to do, it is almost inevitable that extra hardware will have to be added to your computer to allow it to perform these activities.

5 Communications

Using personal computers for communications, or *going on-line*, is one of the applications of personal computing that has achieved popularity only comparatively recently. All the same, it is one of the most exciting and liberating things to happen to them.

Computers have been able to communicate either with terminals or with each other practically from their inception. Now it is becoming increasingly easy to use a home computer as a communications terminal and, just as importantly, there are worthwhile reasons for doing so. These include stores of information that can be accessed, services provided by other computers that can be used despite the remoteness of the computers providing them, and the ability to communicate with the owners of other personal computers.

From a technical point of view, it is not difficult for a computer to communicate. It requires little more than to lead the electronic signals flowing within it to the outside and to conduct these signals to their destination. It is not quite as simple as this, of course, for two main reasons. One is that no two types of computer work in the same way, so that some conversions are necessary but, as this is the kind of thing that can be done by a computer anyway, it is not an insuperable problem. The other is that if an existing communications network is used to convey the signals, it will almost certainly be necessary to convert the signals from the form in which a computer provides them to the form in which the network requires them. This is particularly true of the telephone network which is both the most readily available network and the most widely used.

The benefits that result from enabling computers to communicate with each other are considerable and they follow from the fact that the linked computers can share their resources with each other. When a network of linked computers has been established, the user of one of the computers can make use of the facilities of

any computer in the network. Besides this, he can also communicate with anyone using one of the other computers, be it to discuss problems, to share ideas, or simply to chat. Not all the benefits that can follow from this may be apparent at first sight. A number of computers linked together by good communications and with the necessary communications software can become, in effect, one very large computer. By communicating and interacting with each other, the users of the network can produce co-operative results that can exceed anything they could accomplish individually.

Today, the operations of banks, airlines, police forces and, indeed, governments depend heavily on the use of interlinked computers and would be practically impossible without them. Personal computer users will not be making the same uses of their interlinked computers as these organisations, but they will know what can be achieved, and will be able to borrow the expertise that they need to do what they want to.

Personal computers have had the ability to communicate from the first, and a few of their owners discovered this at a very early stage. A fair amount of electronic expertise was necessary to construct the equipment needed to connect them to the telephone network, but once this was made the computer could communicate with any other that was attached to the same network. We may hear most about the activities of the 'hackers' who illicitly break into the computers of large institutions, but there is a good deal of innocent and perfectly legitimate activity going on, as there has been for some time.

In their way, the activities of the hackers illustrate very well the way that communications liberates the computer. There is a lot to find out about the inner workings of any personal computer, and the best way for the owner or user of one to find out is to explore them for himself. When the computer stands alone, this perfectly natural expression of curiosity can do no harm to anyone else. When a computer is linked by a communications network to other computers, there is much more to be curious about. And the difficulty is that the computer can give its user the ability to delve into things that are not his own. To experiment with a public communications network is to risk the disruption of a public service. To explore the workings of another computer without permission is to intrude on software and information belonging to other people, and even to jeopardise their integrity.

Just as communications gives extra scope to the hacker, so it provides extra services and facilities to the orthodox user. It is sometimes argued that the operators of computers and computer-based services to which access can be obtained should make their

systems secure so that the hacker can do no more than the orthodox user. But the systems are complex, and they are still in their infancy, so that taking preventive measures is not quite as easy as this.

The aims of chapter are to describe the hardware and software that are needed to equip a personal computer to go on-line, and then to describe the applications that can be carried out by an on-line computer.

The hardware and the software

The telephone network, which allows us to speak to each other no matter where we may be in the world, is used by personal computers as their ready-made means of communication. In the same way as one person with a telephone can speak to another who also has a telephone, so one computer that is connected to the telephone network can communicate with another computer that is also connected to it. Because the telephone system was designed to carry human speech, and not computer signals, the signals produced by a computer must be converted so that they resemble speech signals before they can be carried successfully by it. The device that carries out this conversion is known as a *modem*. This jargon word is a contraction of the words modulator and demodulator. When used for sending information from a computer, the modem accepts the signal from the computer. Because it represents a stream of binary data, this signal consists at any time of either one voltage level or another, with each level representing one of the binary states. The modem operates by changing each voltage level to an audible tone, thereby changing the computer's signal to a voice-like signal suitable for transmission down a telephone line. (This process is the modulation of a tone by the computer's signal. The reception process is, correspondingly, the demodulation of the telephone signal to recover a signal acceptable to the receiving computer. From this come the two words that give the modem its name.)

Since only one telephone line will be used for communication, the computer's signals must be sent one bit at a time, in serial, so that a modem must be connected to the Amstrad via a parallel-to-serial convertor. There are two main kinds of modem, and they are known as acoustic couplers and direct connection modems. An acoustic coupler provides two cups into which the ear-piece and the mouthpiece of the telephone handset are placed. A direct connection modem (Figure 5.1) is plugged in directly at a telephone jack plug.

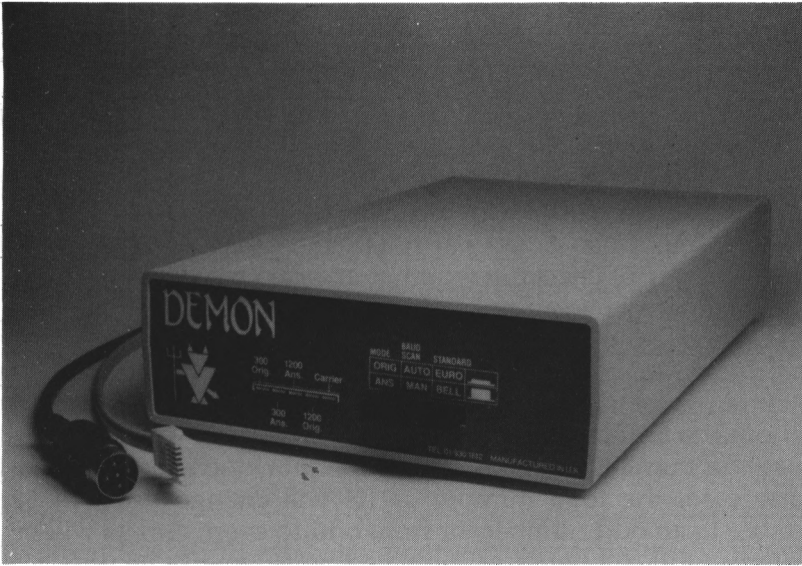


Figure 5.1 A direct connection modem

There are, quite naturally, advantages and disadvantages associated with the two kinds of modem. An acoustic coupler is portable and can be used with a telephone wherever it may be. (It should be noted, though, that the handset of a Trimphone will not fit into the cups of an acoustic coupler.) A direct connection modem is usually plugged in at one location. Because the communication process depends on the transmission of sound, it is essential to prevent external sounds entering the telephone. This is not problem with a hard-wired modem, but if the cups of an acoustic coupler provide a poor fit, or if it is used in a noisy environment, then extraneous sounds can intrude to corrupt the signals. Besides this, hard-wired modems tend to be more sophisticated, offering more facilities and being capable of sending signals at greater speed.

Since a telephone line is intended to convey information at the rate at which it is contained in speech, there is naturally a limit to the speed at which it can communicate information between computers. Typical transmission rates lie in the range from 300 to 1200 bits per second and, by and large, the reliability of transmission increases as the rate decreases. One unusual aspect to do with transmission rates that should be mentioned concerns Prestel and other Viewdata systems. These systems use different rates for sending and receiving data, sending it at 1200 bits per second but accepting it at 75 bits per second. This pair of rates is written as 1200/75.

When two computers are exchanging information, they must

clearly agree on the speed at which they are to exchange information with each other, but there are other matters on which agreement is also needed. Although data is transmitted serially, it is still parcelled up in groups, in an attempt to ensure the integrity of its transmission. The natural basis for a group is the character, and the number of bits needed to represent any character taken from the repertoire that is normally used is seven. To these is added an eighth so-called *parity* bit, the purpose of which is to provide a way of checking whether errors occur during transmission. The parity bit can be even or odd, meaning that its value is determined by ensuring that the number of binary 1's in the resulting group of eight bits is even or odd. This arrangement can detect an error occurring during transmission because an error will cause a 1 to be received as a 0, or a 0 as a 1. Either error will cause the number of 1's to be increased or decreased by one. In either case, the total number of 1's will change from an even number to an odd number, or from odd to even, and so will be at variance with the parity as indicated by the parity bit.

After the parity bit has been added, the group is completed by adding a 'start' bit, or bits, at its beginning and a 'stop' bit, or bits, at its end. These start and stop bits are always the same in each group, and their purpose is to orient the receiving computer to give it every chance of recognising the groups properly and of not receiving incorrect information by failing to detect the beginning and end of each character that is sent to it.

To give an example to illustrate all this, the code for the character 'U' in one of the most commonly used codes (the ASCII code) is, in decimal notation, 85. When written as a seven-bit binary number, it becomes 1010101. Adding an eighth bit of even parity to this gives 10101010. Preceding it by the single start bit set at 1 and following it by a single stop bit set to 0 gives the ten-digit group 1101010100. This is the group of digits that will be sent in these circumstances to convey the character 'U'.

In addition to agreeing on these matters, the transmitter and receiver must be at one concerning whether they may send data in both directions simultaneously or whether they can only send it in one direction at a time, in which case a return message cannot be sent until the transmission of the incoming message has been finished. These forms of communication are known, respectively, as full-duplex and half-duplex communication.

All the matters on which two computers must agree before they can communicate successfully with each other are known as a *protocol*. The components of a protocol are summarised in the following table.

The components of a protocol

<i>Component</i>	<i>Purpose of component</i>
Data rate	To standardise the rates at which data is exchanged in both directions between two computers. Typical rates are 300/300 and 1200/75
Parity bit	To allow transmission errors to be detected. The parity bit will be even or odd.
Start	To help the receiving computer to isolate the groups of bits representing characters in the serial stream transmitted to it. One or more start and stop bits will be used and they will be set as fixed binary digits.
Two-way exchange	To establish whether two computers will be able to exchange information either simultaneously or in only one direction at a time.

The protocol to be used for communicating with a particular computer or computer-based service may be set by selecting the appropriate settings using dials on the hardware, but it is more usual to set it from software. When done from software, the individual components can be selected from menus, although it may be simpler than this in some cases, for the software may contain the complete description of, say, the Viewdata protocol, which can be selected as a single entity. The use of software in preparing a computer for communications may be preferable because of its value in simplifying the process, particularly if it provides sensible defaults, there is even more to the preparations than selecting the protocol, for the computer's memory must be made available in a suitable way for storing any information that is transferred to it. Different services send their information in different formats, with Viewdata sending it by the screenful, in *pages*, and other services sending it in a simple serial stream. By dealing automatically with all these matters, or even by just providing a prompt for each together with a menu of alternatives, software can make the configuration process much simpler than if all the relevant matters must be dealt with individually.

Applications

With your computer connected to the telephone via a modem and the necessary 'terminal emulation' software running on it, you

are ready to communicate. This section deals with the various computer-based services that can be dialled, and describes the usefulness of the facilities that they have to offer. But there is one point that must be made before going any further. Most of the on-line services and databases that can be contacted are operated on a commercial basis. It is necessary to subscribe to them before you can use them, and this involves an initial joining or registration fee. When using them, there is often a charge for the information that is acquired and the services that are used must be paid for. Then there are charges for the use of the telephone. These costs can mount up, and it is certainly necessary to keep an eye on them.

The computer-based facilities that are provided by one or more of the services can be categorised, very broadly, as follows:

- 1 Viewdata, which is dealt with below under Prestel, but which is the generic name for a family of interactive information services originally based on the use of an enhanced television set.
- 2 Databases containing detailed and comprehensive information on their various and specialised areas.
- 3 Bulletin boards, which are run by enthusiasts for enthusiasts, and which provide the electronic equivalent of a bulletin board where messages of any kind can be posted.
- 4 Office services providing facilities that can carry out the information-handling tasks that are likely to be required in an office, such as electronic mail, electronic diary facilities and word processing.

Prestel

Prestel is one of a general class of systems, known as Viewdata systems, originally designed to provide an interactive information service in the home. Viewdata systems were based on the idea that many homes would already possess a television set and a telephone, so that it would be sensible to base a system on them. One consequence of this approach, and in particular of using the television set as the display device, is that Prestel presents its information by the screenful. This mode of presentation is not shared by all information services, and others rely on the receiving computer being able to retain in its memory the information it is sent. This underlines the need to configure a computer appropriately in order to communicate successfully with a particular service.

Communication with Prestel can be established, as with any other computer-based service, by dialling its number and then giving the identification code assigned to you on registration. The successful entry to Prestel is signalled by the appearance of

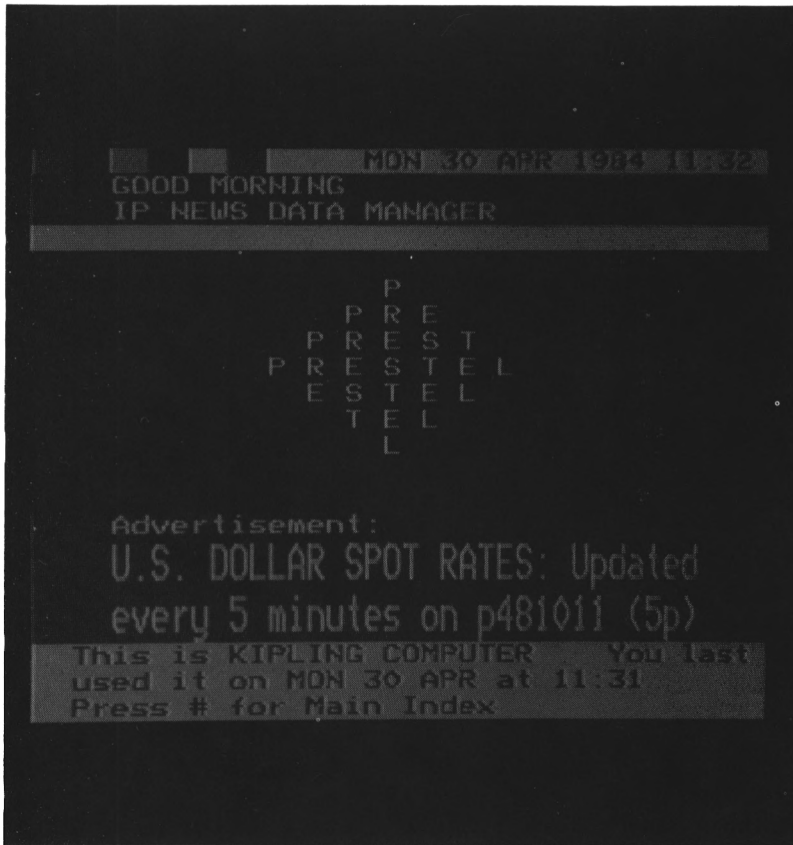


Figure 5.2 Prestel's welcome screen

Prestel's initial welcome screen, shown in Figure 5.2. There is no particular difficulty in finding any information that may be required, because Prestel uses a system of menus to guide its users to any item within the vast amount of information that it holds.

Prestel can be regarded as an electronic publishing medium. It contains some hundreds of thousands of pages that give information such as, for example, news, theatre and cinema guides, commodity prices, company results and analysis, and travel reports. Because Prestel's information can continually be kept up to date by the concerns providing it, the 'information providers', it is particularly valuable in areas where information changes rapidly, as it does with foreign exchange rates, airline seat availability and sports results. You can find any of this information by keying the numbers to make the appropriate selections from successive menus. The sequence of numbers gives the page number for the page on which the information is

displayed. If you know the page number in advance, as you will if you use a page regularly, the number can be keyed directly to avoid the trek through all the menus.

It is also possible to use Prestel to order goods, reserve seats at the theatre and book a room in a hotel. In these cases, the final page reached by the user is a 'response page', rather than an information page, in which the details of the transaction can be entered and, after their correctness has been confirmed, forwarded.

Private areas accessible only to authorised users can be established within Prestel. This allows those with access to one of these areas to exchange confidential information with each other, besides providing an area that contains only the information in which they are interested so that they do not always have to contend with all the information held by Prestel.

Micronet 800 is another kind of closed area within Prestel. It is intended for those interested in microcomputers and computing, and the expected access device is a microcomputer whereas for the rest of Prestel it is a keypad used in conjunction with a specially enhanced television set. Naturally, Micronet provides information directed to the special interests of its users and allows them to exchange messages, but it also supplies telesoftware, that is, computer programs that it can send to the users' microcomputers.

Databases

It is possible to dial directly into a number of databases that provide information of a more specialised kind than that supplied by Prestel, and databases in the USA as well as in this country can be accessed. Two of the databases and the kind of information that they provide are:

1 BLAISE (British Library Automated Information Service), which gives access to the British Library's catalogue of published works. The record for every book includes its title, author, publisher and a list of keywords, while some provide a summary of the contents. For researchers, writers and academics, this database is an unequalled source of information.

2 DIALOG supports a huge store of specialist information comparable in its scope to that of the largest encyclopaedias. A subset consisting of its more popular files, and known as the Knowledge Index, has been established to meet the information needs of micro users.

Bulletin boards

Bulletin boards are usually operated by individuals using their

own telephone and computer to provide a forum for other enthusiasts. This means that their use is free, although it is accepted practice that users make a contribution, perhaps in the form of news or software that is of general interest, whenever they access a bulletin board. Most boards can be accessed by only one user at a time, so that it can be difficult to contact a popular one, particularly at a peak time.

A bulletin board provides an electronic equivalent of a notice board on which messages can be left and from which they can be read. This allows its users to converse with each other, perhaps about microcomputer matters, which will naturally be a common interest. It can also provide an electronic version of the 'small ads' found in a newspaper, and can be used to hold software, a copy of which can be sent from the board to a user (the jargon word is 'downloaded').

Telecom Gold

British Telecom supports Telecom Gold, which is a computer-based service providing the information-handling tasks that are likely to be needed in the modern office. These include electronic mail, word processing facilities, an electronic diary and telexing facilities. Although aimed at business users, it does show the services that can be provided on-line and that are likely to be made more generally available in the future.

Summary

With a modem to link your Amstrad computer to the telephone network, it is possible to communicate with other computers and with information services. By communicating with other computers you can exchange mail and ideas with their owners. By contacting an information service, it is possible to obtain information about, for example, holidays and travel as well as information on more highly specialised topics. Some information services cater especially for microcomputer users in that they maintain special interest groups for them, such as Prestel's Micronet, and can transmit software directly to a microcomputer.

Glossary of jargon terms

Acoustic coupler. A device for connecting a computer to the telephone network via the telephone handset.

Back up. A copy of a file made for purposes of security.

BASIC. Beginners' All-purpose Symbolic Instruction Code. A computer programming language.

Bit. Contraction of binary digit. A *binary digit* is one of the two digits, represented by 0 and 1, used by the binary number system.

Bulletin board. A computer-based service accessible by telephone that is operated by an enthusiast for other enthusiasts.

Byte. A group of eight bits.

Cell. In a spreadsheet, a location at which text or a number can be displayed.

Centronics. A standard means of connecting a device, but mainly a printer, to a computer that employs parallel communication.

Character. Any symbol that can be displayed by a computer, including letters, numbers and other symbols.

Cursor. A marker on the screen of a computer showing where the next item will appear.

Cursor movement keys. Keys marked with arrows which change the position of the cursor when they are pressed. The cursor moves in the direction marked by the arrow on the key.

Database. An organised collection of information from which items can be retrieved in any way that the user requires.

Defaults. The settings which determine how a system or a program operates by default. The default settings can usually be changed to suit the user.

Dot matrix. A rectangular array of dots from which characters can be created by selecting certain of the dots.

Downloading. Passing information from a central computer to a personal computer with access to it.

Duplex communications. Two-way communication. Full duplex

means that communication can take place in both directions at the same time. Half duplex communication means that it can only take place in one direction at a time.

Editing. Correcting and polishing text.

Electronic mail. Passing messages in electronic form either directly from computer to computer or via an intermediate computer where they can be stored.

ENTER. The key which is pressed to indicate the completion of an entry typed at the keyboard.

Expert system. A program that can store knowledge extracted from a human expert and deliver advice based on this knowledge. It should also be able to explain the reasoning on which its advice is based.

Field. An entry in a record.

File. A collection of information that is in some sense complete and is treated as a single entity. In particular, it is stored as a single entity.

Fount. A complete collection of designs for characters.

Graphics. The creation and display of pictures and images by a computer.

Hacker. Person who, perhaps illicitly, uses on-line facilities to explore the structure and capabilities of any attached system.

Icon. A visual representation, to be displayed on its screen, of an activity that the computer can provide.

Information provider. An organisation supplying and updating information that is available commercially on Prestel.

Information technology. The technology resulting from the convergence of computing and communications.

Integration. The linking of two or more facilities or programs to create a whole that can offer more than the sum of its parts.

Interface. An electronic circuit that allows two basically incompatible devices to communicate with each other.

Joystick. A device, modelled on a pilot's joystick, for communicating with a computer.

Justification. To adjust text by the introduction of spacing, usually to give it a neat margin at the right in addition to that at the left.

Menu. A list of activities from which one is selected by keying the number associated with it.

Modem. A device to allow computers to communicate over the telephone network.

Mouse. A device for communicating with a computer. When rolled over a flat surface, it controls a cursor that moves in a corresponding way on the screen.

On-line communication. Direct communication with another computer or with a computer-based service.

Page. A screen of information, as used by Viewdata systems.

Parallel transmission. The simultaneous transmission of several, but usually eight, bits.

Peripheral. Any device that can be attached to, and controlled by, a computer.

Program generator. A program that can accept a description of a task in lay terms and from it can create a program enabling a computer to perform the task.

Protocol. The collection of matters on which two devices must agree before they can communicate successfully.

Record. A collection of fields. The basic entry in a database.

Sensor. A device for sensing some aspect of its environment.

Serial transmission. The transmission of bits one at a time in succession.

Spreadsheet. A program that displays and maintains a table in which the entries can be text, numbers or formulae.

Telesoftware. Computer programs which can be communicated over a telephone line.

Terminal emulation software. The software that converts a computer to a communications terminal.

Turtle. A small robot used in education.

Viewdata. The generic name for systems based on the television set and telephone that give access to computer-based information services. Prestel is one such service.

Window. The screen display as a window through which part of a worksheet or of a document can be seen.

Word wrap. In word processing, the automatic movement of the first word that is too long to fit on the current line to the beginning of the next line.

Index

- Acoustic coupler 98
- Amstrad BASIC 40, 54, 61, 68

- BASIC 40, 68-9
- Bulletin board 104

- Cassette 73-4
- Cell 18, 35
- Communications 96-105
 - applications of 101-5
 - benefits of 96-7
 - software for 100-1

- Database 3,11-16, 30-4
 - on-line 102, 104
 - uses of 12
- Disk 73-6

- Editing 7, 9, 28
- Electronic mail 6
- ENTER key 8, 26
- Expert system 24

- Field 16
- File 16
- Footer 9

- Graph plotter 72, 82-4
- Graphics pad 86
- Graphics 23, 82
 - programming 41, 45, 52, 65

- Hacker 97
- Header 9

- Icon 87
- Information 2
 - handling of 3-4
- Integration 23
- Interface 72

- Joystick 84-6
- Justification 8, 28-9

- Keyboard 3, 6
 - proper use of 6

- Light pen 86
- Logo 90

- Menu 11,86
- Micronet 104
- Modem 98-9
- Mouse 87

- Pascal 68
- Peripheral 71
- Prestel 99,102-4
- Printer 70, 76-7
 - daisy-wheel 77, 79
 - dot matrix 77, 78
 - uses for 77-82
- Problem solving 40-69
- Program generator 24
- Protocol 100-1

- Record 16
 - design of 31
- Robot 72, 88-94
 - educational 89-93
 - personal 93-4

- SHIFT key 5
- Spreadsheet 3, 16-23, 34-8
 - formula 17, 21, 35-6
 - as model 18, 22

- Telecom Gold 105
- Text 4
- Turtle 89-93

- Viewdata 99, 102

- Window 87-8
 - in word processing 10
 - uses of 87-8
 - with spreadsheet 19
- Word processing 2, 4-11, 25-30, 78-80
 - benefits of 6-7
- Word wrap 8

BOOKS FOR AMSTRAD COMPUTER OWNERS

Working Graphics on the Amstrad CPC 464 and 664-

James, Gee & Ewbank

Explains Amstrad graphics and how you can use them. Covers sprites, animation, computer assisted painting, two and three dimensional graphics, and charts and graphs. A practical book that gives enough information for you to convert the programs for your own purposes - or use them as they stand. All listings are taken from working programs.

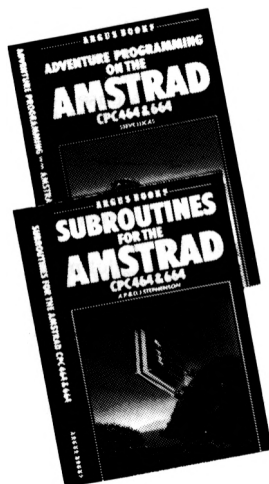
Illustrated, 234 x 156mm,
192pp £7 95
085242 874 X
PRODUCT CODE No. 170087

Applications for the Amstrad CPC 464 and 664 -

Garry Marshall

The book describes, demonstrates and illustrates the full range of useful applications for the Amstrad computers. From word processors, databases and spreadsheets to problem solving, from 'bolt-ons' like cassette and disc drives, printers, plotters, joysticks, light pens and mice, to communications uses - Prestel, Micronet 80, databases, private bulletin boards and Telecom Gold.

Illustrated, 234 x 156mm,
128pp £7 95
0 85242 853 7
PRODUCT CODE No. 170011



Adventure Programming on the Amstrad CPC 464 and 664 -

Steve Lucas

The book to teach you how to write your own adventure programs, including developing the plot, drawing the map, and translating the objects in the game into DATA statements. High-resolution graphics and sound are also described, and listings for three typical adventure games are also included.

Illustrated, 234 x 156mm,
224pp £7 95
0 85242 856 1
PRODUCT CODE No. 170044

Subroutines for the Amstrad CPC 464 and 664 -

Stephenson & Stephenson

The book to show you how to put your Amstrad computer to serious use. More than 50 fully tested subroutines in a wide variety of areas - graphics, maths, music, data processing etc. Major listings include a 3 graph function plotter, an index compiler, and a music sequencer - each being well worth the purchase price of the book in its own right!

Illustrated, 234 x 156mm,
224pp £7 95
0 85242 855 3
PRODUCT CODE No. 170036

Assembly Language Programming for the Amstrad CPC 464 and 664 -

A P Stephenson and
D J Stephenson

Clearly written and readable introduction to Z80 machine code on the Amstrad CPC 464 and 664. It explains binary and hexadecimal arithmetic and contrasts the pros and cons of machine code against BASIC. The book includes a hex loading program, for those working without an assembler, and the Amstrad Assembler/Disassembler.

Illustrated, 234 x 156mm,
160pp £7 95
0 85242 861 8
PRODUCT CODE No. 170060

Available through good book shops and specialist outlets or from

ARGUS BOOKS LTD. Wolsey House, Wolsey Road, Hemel Hempstead,
Herts HP2 4SS. Telephone: 0442 41221

Please add 10% of the total cost ordered to cover postage and packing (minimum 50p)

ARGUS BOOKS

APPLICATIONS FOR THE AMSTRAD CPC464 & 664

This book aims to describe, demonstrate and illustrate the full range of useful applications for the Amstrad CPC464 and 664 computers.

The first part of the book deals with software based applications — word processors, databases and spreadsheets — with a full explanation of what they are and what they can do. This is followed by a chapter that gets to grips with some commercial versions of these packages, in particular the word processor *Amword* from Tasman, *Gemini's Database*, and the *Easi-Amscal* spreadsheet from Saxon Computing.

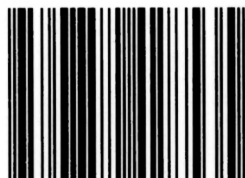
Garry Marshall then moves onto problem solving, and shows how your Amstrad can tackle a variety of tasks from selecting the best route through a complex network, planning the pagination of a book, to designing a coding and classification system.

No book on applications would be complete without some reference to hardware, and all standard 'bolt-ons' that might be needed are covered in the next chapter — cassette and disc drives, printers, plotters, joysticks, light pens and mice. The book closes with a look at communications and how you can get connected (legally!) to Prestel, Micronet 80, general purpose databases like DIALOG, private bulletin boards and Telecom Gold.

Here is a book that will help every Amstrad owner put his computer to serious practical use that will continue to bring results long after the games have been consigned to the rubbish bin.

The publishers would like to thank the following companies for their assistance in the preparation of this book: Amstrad Consumer Electronics, Gemini Marketing, Kuma Software, Saxon Computing, Silver Reed (UK) and Tasman Software.

ISBN 0-85242-853-7



9 780852 428535

ADVANCED TECHNOLOGICAL SOLUTIONS
FOR THE FUTURE OF BUSINESS
INNOVATION AND GROWTH
DRIVING SUCCESS THROUGH
DIGITAL TRANSFORMATION
AND CLOUD COMPUTING
SERVICES. CONTACT US
FOR A FREE CONSULTATION
AND DISCOVER HOW WE
CAN HELP YOUR BUSINESS
THRIVE IN THE 21ST CENTURY.

AMSTRAD

CPC



MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA



<https://acpc.me/>

[FRA] Ce document a été préservé numériquement à des fins éducatives et d'études, et non commerciales.

[ENG] This document has been digitally preserved for educational and study purposes, not for commercial purposes.

[ESP] Este documento se ha conservado digitalmente con fines educativos y de estudio, no con fines comerciales.