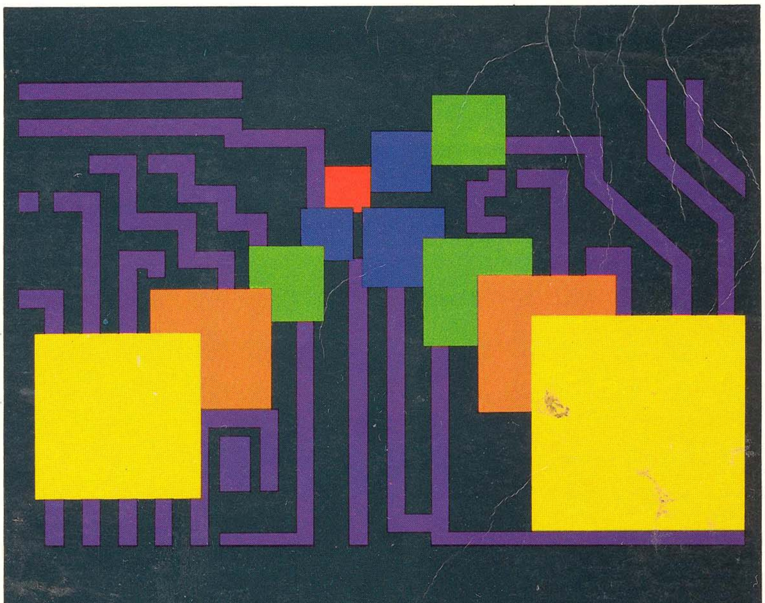
AMSTRAD

USING DATABASES ON THE AMSTRAD PCW8256 & PCW8512



Stephen Morris

GLENTOP

USING DATABASES ON THE AMSTRAD PCW8256 AND PCW8512

USING DATABASES ON THE AMSTRAD PCW8256 AND PCW8512

Stephen Morris

GLENTOP PRESS LTD

First published 1986

ISBN 1 85181 121 4

Printed by The Eastern Press Ltd
London and Reading

A Newtech production for
Glentop Press Ltd
Standfast House, Bath Place, Barnet, Herts EN5 5XE

CONTENTS

ILLUSTRATIONS

CHAPTER 1

INFORMATION AND ITS STORAGE

The world, it seems, revolves around information. To survive in the modern world requires an ability to handle information, as swiftly and effectively as possible. The means by which the information is stored is an overriding factor in determining efficiency. The manipulation of information is an ideal application for computers and, by harnessing their power, the wealth of information available becomes a manageable entity.

This books aims to show how information can best be managed with the aid of two computers in particular: the Amstrad PCW8256 and Amstrad PCW8512. In this first chapter we look at the concept of information and how it is handled in the non-computerised world, an essential prerequisite to the mastery of information-handling techniques by computer.

The need for information

Information has, in recent years, become one of the world's most important commodities. Every organisation and individual has a need for information. Multinational companies store information on a massive scale, ranging from employees' payrolls to the latest share prices, from market research results to stock lists.

Small businesses need to know what goods they have in stock, the current state of their bank balances and the present pricing policy. Clubs and societies must maintain membership lists and accounts. Individuals need to keep track of their personal finances, store details of future events in a diary and keep recipe books. The need for information permeates every level of society and every activity.

The efficiency of any organisation depends not only on the use that is made of the information, but also on how it is stored. Whether or not information is used wisely and to its best effect is a matter of good management and is not a major concern of this book.

However, before any information can be used it must first be stored away. This book describes some of the ways in which the Amstrad PCW8256 and Amstrad PCW8512 can help with the storage of large volumes of information.

Suggestions are also made on the way in which that information should be collected. Often the way in which information is recorded has a major impact on its usefulness and the initial preparations can be crucial to future effectiveness.

Manual filing systems

Before we can begin to look at the way in which computers handle information it is useful to consider the traditional, paper-based filing systems found in most offices and homes. Essentially there are two ways of storing information; these can be summarised as the disorganised and the organised approaches.

The disorganised approach

The simplest way of storing information is to adopt the disorganised method. All information that arrives, whether in the form of bills, correspondence, memos or lists of telephone numbers, is put in a single pile on the desk. Once this begins to fall over the pile is stored away in a box and a new pile is started. As far as storage goes this is the most economical method imaginable; it is fast and requires the minimum of space. However, for large quantities of information it is by no means practical!

Most people have tried this approach, especially for their personal information, and for a busy person it is only too easy to lapse into this style. The problems start when you want to retrieve some information. The only help you have in deciding where to look is that the individual items are in rough, chronological order, with the most recent at the top of the pile. There is little alternative but to start at the top and work down until you reach the relevant sheet. A momentary lapse of concentration and you may miss the document you want and have to start all over again. Any savings in time in storing the information are clearly outweighed by the massive amounts of time needed to retrieve anything from the system.

The seasoned user of such a system will, of course, evolve some refinements to the searching process. Knowing how much information arrives in a given time helps in judging roughly where to start looking; with many boxes of paper the dates on the top sheets (if dated) provide a form of index; there is even the Sherlock Holmes approach, dating a pile of papers by the thickness of the dust on top!

Once the papers become disordered in any way (by something being extracted and then put back on the top, for instance) the system begins to break down. Certainly any sophisticated use of the system, such as selecting all correspondence from a certain individual, sorting bills into alphabetical order or locating all memos, becomes an arduous and often almost impossible task. There can be no certainty that the required result will be achieved and there is no assurance of accuracy.

The advantages of the disorganised approach are:

- The system can store any type of paper information.
- The storage of information is very fast.
- The system is easy to learn.
- Minimum storage space is required.
- All information is safe from accidental destruction (though it may never again see the light of day!)

The disadvantages are:

- A very long time is needed to retrieve information.
- The selection of a group of items and sorting into different orders are almost impossible.
- Extracting lists is time-consuming and probably inaccurate.
- The system quickly fills up with redundant information.
- There is a need to be present in the room where the information is stored in order to retrieve it.

Clearly this approach to the filing system is suited only to limited amounts of information and, as we shall shortly see, is an absolute non-starter as a model for the conventional computer database.

The organised approach

The way in which most offices, and many individuals, store information requires much more discipline in the filing of the information in the first place, but makes the retrieval of any particular item fast and effective.

The initial storage of information is broken down into a number of stages. First, the information arrives at random and in no particular order. This may consist of the contents of somebody's 'out' tray or the morning's mail, for instance. Before anything can be done to store the bulk of the paper a preliminary sort is required. This takes the form of dividing up the main stack of documents into a number of separate piles, each containing information on one subject or topic. For example, one pile will contain invoices, a second correspondence, a third contracts and so on. Each of these piles of paper are destined to be stored in a particular filing cabinet, drawer or other receptacle. Redundant papers, e.g. answered messages, unwelcome circulars and empty envelopes, can be filed immediately – in the waste bin.

Depending on the volume of information, there may be further stages where each pile is further subdivided: correspondence into letters from customers, suppliers and the bank, for instance. Thus the filing cabinet will contain a number of separate sections, each containing information on one particular subject. Each section is generally referred to as a *file*. For large amounts of data, an individual file may be the entire contents of a drawer or completely fill a filing cabinet. The important point is that all the information in one particular place should be related in some way.

The next stage is to store each selection away, adding it to the existing contents of the file in some predetermined way. Within any one file the items will be sorted according to some easily identifiable key piece of information. Contracts may be stored in order of date, for example, while incoming invoices will be in alphabetical order of supplier's name; for any particular supplier the invoices will be sorted by date or invoice number. Any item added to the filing system is slotted into the appropriate position.

Clearly this can take some time; indeed, in many organisations the post of filing clerk can be a full-time job. But the benefits are quite substantial. To retrieve a particular item of information it is necessary to specify only two or three terms of reference. All invoices from a certain firm can be located by identifying the invoices file and searching through it until you reach the company name. An individual invoice is located by further specifying the invoice number or its date. In the well organised filing system any single document can be found from amongst several thousand pieces of paper in a matter of seconds.

The success of such a system depends upon the structure that is laid down for it. It is important that this structure is determined before any information is stored, and that it is understood and rigidly applied by anyone using the system, whether for storing or retrieving data. It must be obvious to all concerned where any particular item is to be stored; if it is not then the retrieval of any document is going to be long-winded, maybe almost impossible. If you cannot decide where to file a piece of paper, how can anyone else know where to look for it?

It is also essential that more than one person should understand the system. A great deal of time can be wasted if anyone who wishes to find an item has to wait while someone else does the looking; much confusion will arise when the one person who understands the system goes on holiday (only to be plagued by constant phone calls asking for directions or to return to a mountain of unsorted information).

The rules of filing must be as simple as possible and easily learnt. Much chaos can be caused by a new secretary adding correspondence at the back of the file rather than the front, or resorting the invoices by date rather than name. Inevitably, no matter how rigidly discipline is applied, items will go missing and then the only solution is to scan through all the likely files until it is found.

Selecting groups of items can also be quite fast under this method. For instance, all correspondence from one individual should be filed in the same place and can be easily removed; all letters from all correspondents on a certain date may take a little more finding. But there are some operations that are still going to be slow, and some that will be impractical even to start on. It may be useful, in certain circumstances to have all invoices in date order, but it will take a long time to sort them into this order. Once sorted, invoices cannot be accessed by name, unless the whole file is resorted or an extra copy of all items is kept in that order.

Reports can be compiled from the information, but this will be quite time-consuming. For example, it is possible to compile a list of invoices received, with amounts, dates and invoice numbers, but this will be a slow process. It requires someone to work through each invoice in turn, write down the details and then type it out. However, such a procedure is certainly feasible. Listing all correspondence in order of date could be almost impossible.

The removal from the filing system of redundant information should also be quite straightforward, as long as a reasonable structure exists. For example, all correspondence before a certain date can be consigned to the store room quite easily if all letters are in date order; if they are stored primarily by name then it will take a little longer to sort out. When any system is devised – whether for use with a computer or otherwise – you should always bear in mind the type of access that is likely to be needed.

However, there will always be conflicts; you may want to inspect invoices by company name yet store them in the archives by date and in these cases some priority must be worked out or a system of cross-referencing devised. When it is a question of access, the problem is sometimes resolved by keeping all files in duplicate; a library, for instance, will keep indexes of books in both subject order and alphabetically by author. The disadvantages here are that twice as much storage space is needed and it takes twice as long to file anything.

The advantages of the organised filing system are therefore:

- The system can store any type of paper data.
- Retrieval of data is fast.
- Selection of data according to some criteria is simple.
- Extraction of lists is feasible if the requirements are not complex.
- Redundant data can be removed with a minimum of effort.
- All information is safe from accidental destruction, providing adequate safeguards are followed.

The disadvantages are:

- The storage of data requires a disciplined approach.
- The system can become dependant on a very few people.
- Complex rules may need to be followed.
- Storage space may be wasted (to allow files to expand).
- There is a need to be present in the room where the information is stored in order to retrieve it.
- 'Lost' items of data may be impossible to find.

This approach is clearly much more feasible than the first, disorganised method. It is practical, fast, efficient and accurate. Provided that suitable safeguards are built into the system and a disciplined approach is adopted there should be no problems - apart from the need for plenty of storage space!

Standard forms

The filing systems described above were suitable for storing almost any type of information provided it came in paper form. Within any particular file there need be no specific format for any individual item. For example, the layout of all incoming letters will be totally different and a collection of publicity material may bear no resemblance from one item to the next.

For much of the information, however, there will probably be a standard layout that is followed throughout the file. All invoices sent out, for instance, will probably consist of a pre-printed blank form with various details filled in: customer's name, invoice date and number, description of goods and quantities, and so on. Similarly for employees' records it is often the case that a blank form is designed to suit all occasions. Even contracts may consist of a standard text with gaps to be filled in as appropriate.

In all these circumstances the storage of information is even more straightforward than before. Each file consists of a number of records (or invoices, contracts, etc.), each one of which has an identical appearance. All that differs is the individual details that have been added to the records. Almost invariably, such a record has a number of gaps or boxes that are to be filled with specific details. The contents of one or more of these boxes can be used to decide the order for storing the forms. Employees' records, for instance, may be stored first by surname, then by forenames and finally (for very rare cases where the full names are the same) by address.

Searching for a record is very easy, since the details by which the search is carried out always appear in the same place on the form. It is very easy for the eye to jump quickly from one record to the next. Any individual item of data can be identified by specifying just three attributes: the file name, the record name and the box. For example, an employee's address can be found by specifying the following:

File:	Employees
Record:	Bird, W
Item:	Address

In a similar way the amount of a particular invoice can be found by giving the instructions:

File:	Invoices
Record:	Southbury Electronics, 15/9/86
Item:	Total Amount

These three instructions can be given to anyone who understands the system. They would find the answer by first opening the Invoices drawer; next they would search through the Name box to find 'Southbury Electronics' and, having reached this far, check through the Date box for 15/9/86; finally, the answer is found by looking at the Total Amount box at the bottom of the sheet.

Although this approach requires data that can be stored in a very formal and structured way it can apply to a surprisingly wide variety of information. It is this sort of information that is best suited to storage by computer.

CHAPTER 2

COMPUTER DATABASES

For a manual filing system to cope with a large amount of information it must be carefully designed and the storage and retrieval of that information must follow specific, strict rules. For information stored on computer to be easily accessible a similar discipline must be followed. This chapter describes the form that information must take to be suitable for storage by computer and the types of activity that are generally available on most database programs.

Files, records and fields

When talking about information handled by computer the term *data* is generally used. The collection of all information stored on a computer is usually referred to as a *database*. This may consist of details of invoices, lists of customers and their telephone numbers, and many other categories of data. The programs that manipulate this information are known as *database programs*.

Database programs are able to carry out a wide range of tasks on their data; searching for specific items of data, sorting files according to given criteria and printing lists and labels are just a few of the possibilities. If the programs are to work successfully then the data must be stored according to certain predefined rules, in the same way as for the manual system. A rigid structure must be set up. This structure closely mimics that of the manual system for storing standard forms.

Files

First of all the database must be divided into separate *files*. Each file will contain information on one particular subject. For example there will be a file for invoices, another for employees and a third for contracts. If two sets of data have a very different structure then they must be stored in separate files. However, one set of data can be further subdivided into more than one file if the volume of information is large and if there is no need to cross-reference between the subsets. For instance, each department may have a separate file for its invoices.

Records

Within each file it must be possible to identify individual *records* and the records within a file must have identical structure. In the employees file, for instance, each record would contain information about one employee; a record in the invoices file contains the details of one invoice; each record in the contracts file holds details of an individual contract.

Fields

Each of these records is then subdivided into a number of areas, similar to the boxes in the standard form. For reasons which are now lost in computer history these individual areas on the record are called *fields*.

Each field can hold one item of information. Therefore on each customer record there will be fields for name, credit limit, customer number and so on; depending on the uses to which the customer's address is to be put it may be held in a single field or divided up into, say, five fields, one for each line of the address, with a separate field for the postcode. The fields should always be chosen with the ultimate use of the file in mind.

Each invoice record will have fields for company name, invoice number, date and so on. Meanwhile the contracts records will have fields that correspond to each of the gaps in the standard contract. Whether or not these last two applications are suited to computerisation depends upon the use that is planned for them. The applications also require a careful choice of database program. This aspect of computerisation is expanded on below.

To summarise, the entire database contains a number of files, each of which holds data on a certain subject. The files are subdivided into records, one for each item in the file, and the records are comprised of fields, each capable of storing one piece of information. Records from different files will have very different structures, but within any one file all records share the same format. The fields within a record will be of various types and sizes.

Suitability of data

The information that can fit into such a database is clearly of a limited type. Essentially it must be the sort of data that can be stored on standard forms or will fit into the format of a printed list, consisting of a number of headed columns.

Some information will not be suited to this sort of treatment. For example it is not possible to store incoming correspondence in a computer database. What can be stored, if required, are basic details of the correspondence. In this case there is one record for each letter, with fields for date received, author, subject, etc.

Some applications may be more suited to storage on a database then they initially appear. Outgoing correspondence could be a potential candidate, for example. Here there are fields for date, address, person addressed, main title and perhaps one field for each paragraph (or even a very large field to take the whole text). As long as the database program can accept text fields of a reasonable size and has an ability to design the layout for each record when printed then this can be a surprisingly useful approach. It has the advantage that each letter produced is in a standard format, with the possibility of constructing letters largely from standard paragraphs. Such an application is described in Appendix 1.

The ability to design your own layout and to include a variety of text is essential in many applications. Invoices can be created either by overprinting standard invoice forms (with blank boxes to match each field) or by completely printing from scratch, adding all the text and lines on each print run. In the first case all that is needed is an ability to position each field independently of the rest on the paper; in the second case an additional facility to print text between the fields is needed. If the program has calculation facilities then it can automatically provide totals, percentages for VAT and discounts.

Similar options are needed for printing contracts. The great advantage here is that typing is kept to a minimum, only the limited details that are different being necessary in each case. Many word-processor programs have a facility to merge a list of information with some standard text; the advantage of using a database is that your lists of data can also be used for other purposes, such as the preparation of invoices, or can be the basis for calculations.

Unstructured databases

It would be feasible to devise a database that was based upon the disorganised method of storing information, with all information jumbled together and typed straight in at the keyboard. The disadvantage is that any operation to search for information would take a long time (by computer standards) and many activities would be impossible. You could not, for example, expect the computer to reasonably distinguish between invoices, letters and details for contracts. This could lead to some rather surprising results and would be a waste of valuable computer resources.

Creating a database file

The first stage in setting up any new database file is usually to define its record structure. Before entering any data it is necessary to state what fields are going to be used in each record, the type of each field and, in some cases, their size.

This is the most crucial part of the exercise. Whatever structure is chosen here is often permanently fixed. All data must therefore be able to fit into the fields supplied. Take great care over choosing the record structure, making allowances for all the variations in data that you are likely to come across; it can be very frustrating to have to squeeze a set of data into a record whose layout is not really suitable.

Some programs will allow the record structure to be changed at a later stage while others permit all data to be transferred from one file to another. This overcomes the problem of unsuitable structures, in that the new file can be defined differently. In some cases there is no such option and all data must be re-entered if a different structure becomes necessary.

Field types

At its simplest the field type is either text or numeric. A *text* field can hold any selection of characters, including alphabetic, numeric and other symbols. Its main drawback is that it cannot be used in calculations. Text fields are also frequently referred to as *alpha* fields. A *numeric* field can hold only the numbers 0 to 9 and the symbols for plus, minus and decimal point. Anything else is not accepted and will lead to a request to re-enter the data. The contents of numeric fields can sometimes be used in calculations.

More sophisticated database programs will include a variety of other fields: for example, *date* fields for which the program automatically checks the validity of any dates entered, multiple-line text fields or fields that convert all text to upper case. Numeric fields may come in more than one variety: *integer* fields that can contain only whole numbers (positive or negative) and several types of *real* fields that can take decimals as well as integers. These may have a fixed or variable number of decimal places and the precision with which calculations are made may be definable.

Using a database

The actual entry of the data itself is generally a fairly straightforward process. In the first instance it may be necessary to *open* a file. This merely means that the file is put into a state in which it is able to receive new or changed data.

A blank record is then displayed on the screen and the empty fields can be filled with the specific data items. Usually you will wish to enter a group of records at a time.

In some cases the program provides the opportunity to design your own screen layout. This can be a useful facility if you want to set up the program to be used by someone unfamiliar with the system. Additional text can then be placed on the screen to provide instructions for the user.

Once entry is complete, various other options are available.

Inspecting records

A variety of options may be available for inspecting the contents of records. In all programs the records can be looked at individually, with the same layout as for the data entry. Alternatively an option may be provided to compare a number of records on the screen in a list format.

You can choose a record by giving its unique record number or reference, and you may be able to jump to the beginning or end of the file. Options are sometimes included to move on to the next record in the file or to move back to the previous record.

Editing records

There are always occasions when mistakes are made during the entry of data and therefore every program includes options for changing the contents of one or more records. This process, referred to as *editing*, allows any individual record to be displayed in the same way as when first entered with one or more values then being changed.

After entering, inspecting or editing records it may be necessary to *close* the file. This is a process whereby all data is saved permanently on disk; it also signifies to the program that no further changes are to be made for the time being. Opening and closing are complementary actions. In many cases a data file is only properly closed if an option to leave the program is selected; just taking out the disk and switching off may damage the contents of the file.

Searching, selection and sorting

Having set up your database file and entered the basic data the real power of the database program now comes to the fore. Some time will have been saved in the entry of the initial data by not worrying about the order in which the records were entered. There is usually no need to slot a record into the correct place since the program should have options that will deal with this.

Searching

To find any particular record you need only specify one of the fields and its value. The program will work through the file from the beginning to find a record that contains fields that match the values given. For example, by specifying that the Name field in the Customer file should be 'Southbury Electronics' the program will locate the first record which has this name. If more than one field matches the criteria given then there may be options to move on to the next suitable record, having inspected the first.

Any item of text is generally referred to as a *string*. An item of text that you are trying to find in the file is called a *search string*.

Great care must be taken when searching. If you ask a person to find a record for 'Southbury Electronics' then they will not think twice over selecting one with the name 'Southbury Electronics Ltd'. The computer, however, will treat these two items of text as being totally different and will not locate the desired record. When trying to find a match the program checks the two strings character by character. It only accepts strings as being the same if there are no descrepancies. Even 'Ltd' and 'Ltd.' are treated as being different. Care must therefore be taken both to ensure that the original entry is precise and that any string searched for is accurately stated.

This is perhaps one of the failings of a computerised system. The computer is utterly logical, totally unerring and completely lacking in imagination. There is no way that the machine can take the initiative to identify something that is close enough to that required; it is unable to make any association of ideas.

Some confusion can be avoided by searching for just part of the text. You are often able to specify a search string that starts, ends or comes somewhere in the middle of the actual field value. This overcomes some problems, but certainly not all of them.

When searching, all characters are considered, including full stops, commas, hyphens and even spaces. To the computer a space is just another character, resulting in a blank location on the screen, and two spaces are certainly not the same as one. Therefore missing a space between two words, separating them with a hyphen or accidentally including an extra space will all result in the desired record being missed. To the computer 'Southbury', 'South bury', 'South-bury' and 'Southbury.' are all different strings.

A great deal of confusion can be caused through two people having different ideas about the use of commas, full stops and spaces. 'AM Jones', 'A.M. Jones' and 'A M Jones' are all different people to the computer, so some standard policy is helpful. Even worse than variations in convention is the 'sloppy' computer user who takes no care over such vital details as spaces, full stops and the use of capitals, following no clear convention. The most important thing to learn for anyone who uses a computer (for virtually all applications, not just databases) is that attention to detail is important. In the long run a little care will save a lot of time.

Case-sensitivity Some additional problems may arise over the use of capitals (upper case) and lower-case letters. Some programs will treat the two as the same, others will regard them as being quite different. Therefore whether 'Jones' and 'JONES' are considered equal depends upon which program you are using. A program that distinguishes between upper and lower case is said to be *case-sensitive*.

Using conditions

Some of the problems described above can be reduced by the use of *conditions*. A condition is a statement that you wish the program to search for records that satisfy some given criteria, rather than just containing a specific value in a named field. For example, you might ask to see all invoices where the total amount is more than a certain value or all customers whose names begin with 'S'.

In order to make these searches it is necessary to include some extra symbols in the instruction. Those most commonly used are:

=	equal to
>	greater than
> =	greater than or equal to
<	less than
< =	less than or equal to
< >	not equal to, i.e. greater than or less than

Therefore you might ask for records that satisfy:

Amount > 100

The program should find all records in the file where the value of the Amount field exceeds 100. These conditions can sometimes be used for text as well as numeric data. Then the decisions are not made by comparing the size of the values, but by seeing how two strings compare in alphabetical order. For example, you may decide to ask for:

Name > 'South'

In this case anything that comes after 'South' in alphabetical order will satisfy the condition.

Case-sensitivity The case of the text can be even more important in these circumstances. With programs that are case-sensitive two possible approaches can be taken. Some programs treat lower-case letters as being the same as the corresponding capitals, so they see the alphabet as being 'AaBbCcDd. . .Zz'. Therefore 'South' comes in the same place as 'south'.

Others regard all upper-case letters as coming before all the capitals, treating the alphabet as 'ABC. . .Zabc. . .z'. Under this system, surprisingly enough the most commonly adopted, 'Weston' comes after 'Southbury' but before 'east well'. Before suggesting any comparisons you should therefore be absolutely certain of the rules applied by the package you are using.

Combining conditions Sometimes it is necessary to combine conditions in order to reduce the scope of the search, to search in more than one area at once or to exclude specific records. This is done with the following three *logical operators*:

AND	Combined condition is true if both individual conditions are true
OR	Combined condition is true if either or both individual conditions are true
NOT	Excludes records for which condition is true

For example, if you are not sure of the spelling of Southbury then this condition may not be enough on its own:

```
Name > 'South'
```

This will find many unwanted records that start later in the alphabet. To find just those that start with 'South' use:

```
Name > 'South' AND Name < 'Sp'
```

The scope of the search is increased by the use of the OR conjunction. For example:

```
Name > 'S' OR Amount < 100
```

This locates all records where the name starts with S or a later letter, or for which the amount is less than 100. The only records excluded are those for which both separate conditions are false.

The use of brackets changes the order in which conditions are calculated. Anything in brackets is calculated first. To find all records with either a name starting with S or an amount less than 100 use this condition:

(Name > 'S' AND Name < 'T') OR Amount < 100

Moving the brackets makes a big difference to the outcome:

NAME > 'S' AND (Name < 'T' OR Amount < 100)

In this case a record with the name coming before S and an amount less than 100 is excluded; in the first example it satisfied the condition.

To exclude specific groups of records use NOT. Records whose amount is outside the range 90 to 100 are found with:

NOT (Amount > = 90 AND Amount < = 100)

This is equivalent to:

Amount < 90 OR Amount > 100

The NOT logical operator is only rarely found.

These combinations can obviously get extremely complicated and it is advisable to stick to fairly simple conditions in the early stages of using any database program.

Selection
The obvious extension of the search capabilities described above, in which a group of records satisfying some criteria are viewed in turn, is to create a new file containing only those particular records. This selection process can often be repeated several times, so that the file is reduced at each stage, resulting in a group of records that satisfy a complex list of criteria. In some cases the new file is actually created on the disk, in others it is merely held temporarily in memory.

Sorting files
Perhaps the most exciting use for a database program is to sort the records into a different order, in a fraction of the time it would take to achieve manually. For example, if invoices have been entered in name order it will be useful to sort them into order of date or invoice number, or to alternate between the two arrangements according to circumstance.

Most database programs have fairly powerful facilities for changing the order of the records according to the contents of one or more fields. If more than one field is specified then the second field is used to decide the order when two records have the same value for the first sort field.

For each field it may be possible to choose between ascending or descending order. In the case of numeric fields *ascending* order means starting with the smallest value and working upwards; for text fields it is alphabetical order. *Descending* order reverses the records.

Once again there may be some confusion when sorting upper- and lower-case text, the same rules as for searches being followed by case-sensitive programs.

Some programs will not allow all fields to be used for the sorting process; only *key* fields that are specified when the file is first set up can be used. This should not cause too much of a problem if sufficient key fields are allowed, since it ought to be fairly obvious which fields you will be using for sorting. It is unlikely that a file will be sorted according to the first line of the address, for example, but it may be put in order of postcode.

In some programs you can create a duplicate file on disk, using the new order, while in others the sorted file is only held temporarily in memory. A frequent approach is to create an *index file* which contains just the record numbers in their sorted order rather than the data itself. This saves space in memory or on disk yet provides a fast means of displaying or printing the records in the new order.

Finally, there is often the chance to combine both selection and sorting instructions in order to create a subset of the rearranged file.

Reports

Most database programs will perform some form of search and sort routine, displaying the results on the screen. Frequently it is useful to have a printout of the list; for example, you may wish to print labels for all customers who have spent over a certain amount, listing them in alphabetical order of name.

All databases have some form of reporting facility, whereby the contents of the file can be listed on paper, but the flexibility of the reports varies quite considerably. A common approach is to print the file with one record per line and headings at the top of the sheet. In some cases you can design your own sheet headings; in others, you must be content with the field names to head the columns. Generally speaking you can list any selection of records from the file, arranged into any order. It is also usually possible to specify which fields will be listed.

An alternative is to print the records in the same format as they appear on the screen, resulting in a set of ‘cards’. Some programs allow both types of printout; others stipulate just one or the other.

In some cases rather more involved methods of varying the printout exist. For example, invoice values may be listed in order of customer name with a sub-total for each customer and a grand total at the end. With the more complex packages it is even possible to provide a partial total to carry forward at the bottom of each page.

Looking after files

One of the great advantages of a computer database is the potential it has for saving filing space. An inch high pile of A4 sheets may be stored on a single disk, resulting in great savings in space and a more manageable physical object. Certainly there is no danger that if the disk is dropped the individual records will be scattered all over the floor, taking hours to collect up and put back into order!

However, far greater dangers do exist and it is easy to overlook them until the inevitable disaster occurs. Although it is fairly easy to get a paper file confused or lose individual sheets, it is almost impossible to accidentally destroy large quantities of paper. With a computer file, however, the situation is very different. A single fingerprint on the exposed surface of the disk will completely destroy any chance of recovering the data that is recorded there. Placing a disk near a source of magnetism can be equally devastating. No matter how careful you may be in the care of your disks there will still be occasions when a previously perfect disk suddenly becomes totally unusable for no apparent reason. This *corruption* of the disk, as it is termed, may take the form of making one or more files unreadable; in extreme cases the computer will indicate that the disk is totally beyond recovery.

Often this sort of problem is due to some stray magnetic field altering the most important parts of the disk, where it stores details of the location of files, rather than any physical fault in the disk itself. If only certain files are affected then the other files on the disk may be recoverable and can be transferred to another data disk.

Backing up disks

The alarming prospect of losing all your data should not cause too much worry as long as certain basic precautions are taken. Most importantly you should make regular and frequent *backup* copies of all data disks. It only takes a few minutes to duplicate a disk and this can safeguard many hours' – possibly days' – work.

How frequently you copy your disks is a matter of personal preference. Certainly you should make a copy at the end of each working session but, during periods of intensive work, it may well be necessary to make interim copies. The decision really comes down to a question of how much work you are prepared to lose. Remember – if something goes wrong you will have to revert to your latest backup and any work carried out since then will have to be repeated.

Copying files

Corruption of disks is, fortunately, fairly rare and therefore the precautions taken during a working session need not be too stringent. A greater danger is the accidental destruction of part of a file by entering the wrong instruction; this can be avoided by making regular copies of individual files as the work proceeds. Many programs include facilities for doing this.

More likely to cause problems is the loss of data held in the computer's memory. Some database programs store their data on disk every time changes are made. In these cases the contents of your files are reasonably safe. In other cases the data is held in memory until the file is complete, and here the dangers are quite great. Anything held in memory is only there as a temporary measure, and only remains as long as the power supply is maintained. If the computer is switched off (or if there is some temporary interruption in the power supply or a sudden power surge) then everything held in memory is immediately lost. All unsaved data is beyond recovery and must be re-entered.

The problem can be even greater for those programs that require files to be opened and closed. Until a file is closed the data in it is insecure. If anything happens to the smooth operation of the program before the file is closed – whether a disk fault or an interruption in the power supply – then it may be impossible to close the file and the data then becomes useless. To overcome this danger it may be necessary to close the file from time to time, make a copy on another disk, then re-open the file and carry on working with it.

Some programs incorporate procedures for copying files and carrying out other file management operations. In other cases it is necessary to leave the database program and carry out these activities from the operating system. This is a far more long-winded process and is therefore likely to be carried out less frequently.

The important point to bear in mind whenever using any computer program is that until you have made at least one backup copy of the files your data is insecure and at the mercy of any unexpected hardware faults. If suitable precautions are taken, however, the storage of data by computer can be even safer and more reliable than the corresponding manual methods. It can also provide improvements in speed and efficiency which are otherwise unimaginable.

Programming

Database programs fall into two basic categories: those that include just some or all of the facilities described so far and those that go much further, incorporating their own mini-programming language. Four of the six programs described in this book fall into the first category. Their main task is to store, sort, select and retrieve any data that can be stored in a standard format.

The fifth program, Cambase, allows lists of instructions to be stored away for future use and includes a range of arithmetical operations.

The sixth program, Condor 1, provides a number of additional commands that allow almost limitless operations to be carried out on the data. These allow you to make decisions on how the data should be handled and to collate the data in a variety of ways.

For example, you may decide to produce statements from the invoices file by searching through for each customer in turn, listing the invoices sent and the payments received (from another file). To do this with the simpler programs requires the same sequence of instructions to be entered for each customer, a time-consuming process. With the programmable databases the instructions can be stored and carried out automatically by the program, time after time, tirelessly and accurately.

Procedures and programs

A sequence of instructions that perform some specific task is generally referred to as a *procedure*. For complex activities one procedure may *call* a second procedure, i.e. one of the instructions in the first procedure puts into effect the instructions in the second procedure. In this way procedures can be *chained*, with each procedure ending by putting the next into effect. Alternatively, one procedure may consist of a list of calls to other procedures, each of which has some specific task to perform.

A group of related procedures can be stored away as a *program*. These may be procedures that all depend upon each other to complete some major task (in such a way that all the procedures within the program are linked) or just a set of procedures that are all likely to be used at the same time. For example, there may be one program containing a set of procedures for producing statements while a second set of procedures prints address labels; a separate program may provide analyses for the end-of-year accounts.

Sequences, branches and loops

Procedures are made up of an ordered set of instructions, in the form of a single list. These instructions fall into three broad categories.

A *sequence* is a simple list of instructions. The program works through from the top of the list to the bottom, carrying out each instruction in turn, once only.

A *branch* is a point in the procedure where some decision is made. This decision will be based upon the outcome of some condition. If the condition is true then one sequence of instructions is executed, if it is false then a different sequence of events occurs. Alternatively, the instruction may simply say that if the condition is true a set of instructions should be skipped.

Finally, a *loop* is a set of instructions that are repeated. The loop may either be repeated a given number of times or until some condition is satisfied (or while a condition remains true). For example, if you know that there are ten customers the procedure may be repeated ten times, once for each customer. If the number of customers is likely to vary then it may be repeated for each customer in turn, until all records have been used up.

Flexibility of data

A further advantage of the computerised database is that it can, with the right equipment, remove the need to be physically present in the room where the data is stored. With a suitable computer terminal and modems the data can be accessed from anywhere that has a telephone.

The transfer of data from one place to another is also much more straightforward; using telephone links a large file can be copied from one computer to another in a matter of seconds. However, when engaging in this type of activity you should bear in mind the restrictions of the 1986 Data Protection Act, described later in this chapter.

The advantages of a computer database are therefore:

- The retrieval of data is very fast.
- Data can be selected according to limitless combinations of criteria.
- Data can be sorted into many different orders.
- The extraction of lists is simple and automatic.
- Redundant data can be removed quickly and accurately.
- Data is secure if adequately safeguarded.
- With suitable equipment, data can be inspected at a distance.

- The system can be made easy to understand for anyone using it.
- The physical storage space is very small.
- 'Lost' items of data can be rapidly located.

The disadvantages are few, though the first at least is severely restrictive:

- Only data that fits into a standard format is suitable for a computerised database.
- The data is extremely vulnerable if not carefully managed.
- Inspecting just one or two records is comparatively long-winded because of the time taken to start up and close down the program.

From the summary above it is clear that for the right type of data, and with suitable precautions, the computer provides the ideal answer for most information-handling problems.

Database programs reviewed in this book

Many different database programs have been produced over the last couple of years, a number of which have been converted for use with the Amstrad word processors. In this book six such packages are considered, between them covering almost the full spectrum of database facilities. They range from the very simple system, mimicking the traditional card index, to the fully fledged database, incorporating a complete programming language and capable of manipulating almost any type of data. Six packages are reviewed:

- *Matchbox* is a very simple program based on the card index, with only very limited search abilities and no sorting capability.
- *Cardbox* is a slightly more advanced program, with the ability to determine the position of the fields on the screen. Searches can be carried out at a number of levels but there is no sorting option.
- *AtLast* is a rather difficult program to master but with the advantage of the ability to link records in different files. Records can be sorted according to predefined key fields.
- *Retrieve* is an easy-to-use yet powerful program. It includes labelling and mail-merging facilities and data can be sorted according to any group of fields.

- *Cambase* has a limited programming capability. It is rather awkward to use, but reasonably powerful when set up.

- *Condor 1* is a comprehensive database package, including a good programming facility. Any operation can be carried out on every selected record in a file. The package includes full sorting capabilities and is easy to use.

Whatever application you intend to computerise there should be a package amongst these that satisfies your needs.

The Data Protection Act

If you are considering using a computer to store personal information about individuals then you should consider registering as a data user under the 1986 Data Protection Act. The Act covers all personal information, with a very few limited exceptions. You need to register if you are storing information yourself or processing data for someone else (in which case you are referred to as a 'Computer Bureau', for the purposes of the Act).

The Act also covers the uses to which the data may be put and the way in which it is stored. The uses of the data and those to whom it may be disclosed must be registered; there is a duty to ensure the accuracy and security of the data.

Further information and registration forms are available from the Post Office.

CHAPTER 3

DATA COLLECTION AND ACCURACY

The results produced by a database program are only as good as the data that is first entered into the computer. If the raw data is riddled with mistakes then the results that are achieved are likely to be unreliable and should be treated with great caution. On the other hand, precise, accurate data will lead to reliable, trustworthy results that can be quoted with confidence and will enhance your business.

This chapter explores ways in which the accuracy of computer data can be improved. It also discusses how the errors in any results can be accurately calculated to ensure a realistic appraisal of conclusions reached by the computer database.

Types of error

Mistakes can creep into your data from a number of sources and you should be aware of the potential harm they may cause. The main types of error include:

- *Collection errors:* mistakes made when collecting the initial data.
- *Entry errors:* mistakes made when entering the data into the computer.

- *Program faults:* mistakes in the logic of the program that result in inaccuracies in the data.

- *Hardware failure:* corruption of the disks or power surges that cause faults in the data in memory.

Of these the collection errors are the hardest to detect. If someone makes a mistake in filling out the form to start with then it may be impossible to check later on and the error may go unnoticed. Entry errors can be discovered (with a great deal of hard work) by comparing the raw data with the information held in the computer file.

Faults in the program are unlikely to be a major cause of problems, although even professional programs have 'bugs'. These faults are more likely to cause the program to 'crash' unexpectedly or corrupt a file, rather than making an arithmetical mistake. Of greater concern are the programs you write yourself, as in Condor 1. For example, incorrectly entering the formula for calculating VAT can have serious consequences. Such faults may not become obvious for some considerable time and can be very difficult to trace. These faults are discussed further in Appendix 1.

Hardware failures may result in a file being totally wrecked or may have no effect at all; such failures are unlikely to go unnoticed. It is extremely rare for a hardware failure to result in a minor miscalculation.

A well designed database should eliminate many of these errors and reduce the harmful consequences of those that do remain.

Data collection

If the original data is wrong then any conclusions drawn from it are suspect. It is therefore of vital importance that all data, whether for use in a manual or a computer system, should be as accurate as possible. At the very least anyone using a set of information should be fully aware of the maximum levels of error in any particular item.

For example, if dealing with weights you should be aware of the accuracy to which measurements have been taken. There is no point in calculating an average in fractions of an ounce for data measured to the nearest pound. It is also important to ensure that everyone uses the same units of measurements. (Invoices can reach alarming totals if goods are measured in ounces and their cost is calculated according to the price per kilogram!)

If goods are given reference numbers then checking procedures need to be built into the system to cope with possible transcription or typing errors.

It is also important to ensure that the system is storing the information that is really needed. It is no good recording people's ages in years if the date of birth is required; likewise, recording dates of birth can lead to unnecessary calculations if what is really needed is the age. Above all else, beware of storing unnecessary information; the magpie instinct is all very well when it comes to being able to produce that one vital statistic from a set of data that no-one thought would ever be used but it can also lead to a great deal of wasted time, effort, energy, money and space.

Preparing data collection forms

Many of the problems associated with data handling - from inaccurate data and confusion over units to missing and inaccessible statistics - can be avoided by making careful preparations before any data is collected.

If your data is ultimately destined to be stored by computer then it is probably in some standard format and therefore is recorded on a form. The design of this form can be critical in ensuring that the data is collected accurately, and that the intermediate stage of transferring the data to computer avoids new errors creeping in.

The form should be clearly set out, easily legible and its intent should be obvious. Instructions will be needed to ensure that the right information is gathered but these should not be so lengthy as to confuse the data collector.

If a survey is conducted, the questions should be clear and concise and the range of possible answers should be indicated. Avoid insisting on a simple 'Yes' or 'No' answer to questions for which most people will clearly want to qualify their answer. Asking 'Do you drink frequently?' may lead you to conclude that the majority of people do, but its validity in determining the national level of alcoholism is highly questionable. Unfortunately there is no easy way round such problems. As we shall see, a computer cannot accept the multitude of possible answers to such questions without losing the ability to produce results. In order to come to any sensible conclusion the set of responses must be limited and this inevitably leads to some answers being given with misgivings. The result is a degree of error in the final conclusions; the level of error will depend upon the level of flexibility in the questions and these errors are themselves open to varying interpretations and opinions.

For numerical answers make sure that the range of answers is limited. If asking for an age then specify whether it is age at present or at next birthday, and whether it should be age in years only or years and months (for children). If it doesn't really matter and only a rough estimate is needed then don't give long instructions since this will only confuse the issue.

Particular care must be taken if the forms are to be filled out by a large group of people, rather than a few trained individuals. Remember that people may not always apply the same rigorous attitude to filling out the details or may tend to supply optimistic estimates. If you want their age only if under 25 then avoid embarrassment and unreliable responses by saying so!

Dates can cause problems if not used carefully. Most people in Britain will supply numerical dates in the format day/month/year, but elsewhere you may find month/day/year or other variations. Avoid this by specifying the format or at least be aware of the variants you may find.

For any sort of measurement be sure to specify the units to be used and the accuracy required. It may also be necessary to know whether the data has been rounded up or down or to the nearest whole number.

In conclusion, ensure that your aims are obvious to the person filling the form, that they will result in data that is intelligible to the person transferring it to the computer and that they are the results that were really asked for.

Data entry

Having collected the data you are well on the way to making good use of it. In a computerised system the next stage is to transfer the data to a computer and it is here that futher errors will arise unless measures are taken to minimise the mistakes.

If the form has been designed properly then the information provided on paper should match the requests presented on the screen by the computer. It should be a fairly straightforward task to type the information in. In those cases where the program allows you to design your own data entry screen this can be made to closely resemble the collection form; if this is not possible then the screen display should be borne in mind when designing the collection form.

Problems will most likely arise if the person entering the data has to make any sort of judgement or carry out calculations. For example, if the form shows dates of birth and the computer requires an age in years then a mental calculation must be carried out for each record and, with large quantities of data, will lead to some errors.

Of course, there will always be simple typing errors when any volume of data is typed into a computer. Names will be misspelled, numbers mistyped, even whole records accidentally ignored when two pages stick together. Much of this can be avoided by ensuring that the person doing the typing is not under pressure to complete the task too quickly and that sessions at the keyboard are limited. The longer a person has been working at the computer, the greater the number of mistakes they make. Regular breaks from the machine can save a great deal of time in the long run. Ultimately the reliability of your data will depend upon the satisfactory nature of your working conditions.

Various checks can be built into the system to reduce errors. Many of these will depend upon the flexibility of the system being used and the degree to which the user can incorporate checks into the program. For example, with a list of figures it may be useful to have an additional box on the collection form for a total. If the program will allow calculations to be carried out then it can calculate its own total. If this disagrees with the total that is typed in then this suggests an error: perhaps one of the figures was copied across to the computer inaccurately or the total on the original sheet was wrong. Either way the discrepancy will be highlighted and can be corrected.

Validation

Computer databases generally have some form of data *validation* incorporated into the program. These are means by which the program tests to see whether or not a particular entry is a realistic one. At its simplest this merely means that the program will check that no alphabetic characters are typed into numeric items. An incorrect numeric value (containing other than the numbers 0 to 9, decimal point, plus and minus) will result in a request to re-enter the data, the computer may 'beep' or simply sit there stubbornly waiting for something acceptable to be entered. If integer fields have been specified then no decimal points will be allowed.

Some programs allow a field to be specifed as a date type, in which case it will only allow entries in the date format, e.g. dd/mm/yy, and will check each constituent part to make sure that it is within the permissable range. Such a program should test for the number of days in the month as well as complications such as leap years. You can always test the sophistication of programming in a package that is supposed to validate date fields by entering a date such as 29/2/87. If the program accepts the value then you are likely to be in for trouble.

Some database programs allow ranges to be specified for each field and test to see whether each entry lies within this range. For example, they do not allow a percentage outside the range 0 - 100. With those packages that incorporate some form of programming language you can include your own validation checks in the data-entry stage. For instance, it is a fairly simple matter to check that times are realistic. Some examples are given in Appendix 1.

Verification

Validation checks can identify some mistakes, but they will miss many simple typing errors. Some form of *verification* of the data may be needed to find these errors. If a set of data is particularly important then you may wish to consider some form of double-entry procedure. Here the records, or part of the records, are each entered twice. This is done either by the same person repeating the entire process or (preferably) by two different people. On each occasion the same data is typed into the computer so that the two sets of results can be compared. How the comparison is carried out depends upon the sophistication of the database program.

At the simplest level (for example, Matchbox) a printout must be produced for each set of data and these are then manually compared, item by item. This is generally easier than comparing the data with the original lists, since the format in the printed list is almost certain to differ from the way the data appears on individual written records. Comparisons between different formats can be quite a strain and many errors will be missed.

One of the advantages of the more advanced computer database programs is that a screen display can mimic a paper form but the data can then be condensed into a printed list, with each record only taking up one line on the printout.

It may be that only certain data needs to be verified, in which case the second entry can be restricted to just items of importance. With programs that allow you to design your own reports you can list only these items in each case, making the comparisons more straightforward.

Double entry becomes more practical with programs that allow totalling and other calculations. All that then needs to be compared is the overall results. Only if these show discrepancies is it necessary to compare the raw data. Depending on the nature of the data, it is unlikely that there will be two errors in a set of data that precisely cancel each other out, although this will happen very infrequently, of course. Whether you wish to take further measures to counteract such occasions depends on the nature and importance of the data. For example, a simple total of entries will not show up the errors in these two sets of data:

6 5 8 3 2
6 4 8 4 2

The total in each case is 24. Such an error will be found if a sum of squares of the entries is calculated, the values being 138 and 136, repectively. However, even this does not show the error in the data sets:

6 5 8 3 2
6 3 8 5 2

Further enhancements can be devised, as deemed necessary.

For the top end of the database scale (such as Condor 1), you can really put double entry to its most effective use. Your program can include a special verification routine that checks each second entry against the original and displays a warning if any differences arise. An example is given in Appendix 1.

Error calculations

Essentially there are two types of inaccuracy that can creep into your data. Firstly there are the genuine mistakes, described above, which are due to typing errors, collection errors or misunderstanding of instructions. Also in this category are programming faults and hardware errors.

Secondly, and perhaps harder to deal with, are the errors that arise from dealing with imprecise values. If an invoice refers to a number of units sold, the total charged will be a precise amount, accurately calculated and theoretically perfect. Any errors will be due to human error and can be overcome by suitable checks. There should be no doubt about the results that are calculated.

However, there are many occasions when such accuracy cannot be assured. You may be able to calculate the number of units sold but the weight of an item can never be measured precisely; the figure given can only ever be an estimate, no matter what degree of accuracy is attained. In most cases the inaccuracies will not be relevant to the results but there will always be instances when it is useful to know the level of accuracy involved. These values are fairly simple to calculate. For instance, if a weight is measured to the nearest ounce the value can be said to be correct to within half an ounce. The maximum error is at most 0.5 ounce. If a height is measured to the nearest tenth of a metre then any individual value is correct to the nearest 0.05 metre.

Simple measurements should not cause any problem. Difficulties may arise when calculations are involved and the errors are combined. In these cases a few basic rules should be remembered. If two values are either added or subtracted then the errors should be added in both cases. For example, adding two heights, each measured to the nearest tenth of a metre, results in a value for which the maximum error is 0.1 metre. Adding ten such values produces a result that is accurate to only 0.5 metre. Subtracting one weight from another, when each is measured to the nearest ounce, results in a value that is correct to within one ounce.

For multiplications the formula for calculating errors becomes slightly more complex. If you are multiplying values A1 by A2, with respective errors of E1 and E2, the error of the combined result (A1 × A2) is given by:

$$(A1 \times E2) + (A2 \times E1)$$

If the error in each case is the same this simplifies to:

$(A1 + A2) \times E$

For division the formula is given by:

$[(E1/A1) + (E2/A2)](A1/A2)$

This is the *absolute error*. The formula for the *relative error*, i.e. the error as a proportion of the original value, is different again.

Where applicable you can get the program to perform these calculations for you. As with any error-handling procedures the lengths you go to will depend on the necessity for accuracy in the results. The vital factor is to be aware of the limitations of the computer and not to make unreasonable demands of the system.

CHAPTER 4

INITIAL PREPARATIONS

Before you start to use any application program there are some essential preparations to be made. Copies must be made of the original disk, work disks must be created and data disks prepared.

If you intend to use other than the standard Amstrad printer, or want to use it in a non-standard way, this must also be set up before you load a program. Some suggestions on these preparations are given in this chapter.

Preparing disks

You need to prepare three types of disk:

- Exact copies of the original disk.
- Work disks that contain the programs and some of the more useful CP/M Plus utilities.
- Data disks for storing the data files.

You may also want to make backups of the work disks and the data disks once files have been created.

Copying the original disk
The first thing you should do before even attempting to use any program is to make a copy of the original program disk. This is for your own use only; the programs are protected by copyright and you should not make a copy for anyone else to use, unless they are prepared to pay the full cost of the original disk.

In the case of the database programs described in this book there is no form of protection on the original disk, either to stop you illegally copying it or to prevent you from accidentally overwriting it. Therefore you should follow their exhortations to copy the disk and, having done so, store away the original in a safe place. If you fail to do so and inadvertently damage the original disk then you may pay dearly for a replacement.

The original disks do not contain any disk-copying routines. Therefore this must be carried out using your CP/M disk. Switch on the computer and load the CP/M Plus disk into drive A (the top drive on a PCW8512). (The loading and running of CP/M utilities are not a subject of this book; it is assumed that the user is already familiar with the basic operation of their computer. If the database program is the first application you have used on your Amstrad then you should refer to the computer's manual, or a good book, on how to set up the computer, 'boot up' the operating system and enter commands.)

When the A> prompt appears enter the DISCKIT command. On pressing RETURN the main menu appears and you can choose the option to copy a disk. Before starting, make sure that the original program disk has been write-protected. Follow the instructions that are provided to make an exact replica of the original disk on a blank or redundant disk. If the disk has programs on both sides, e.g. Retrieve and Condor 1, then you should also copy the second side.

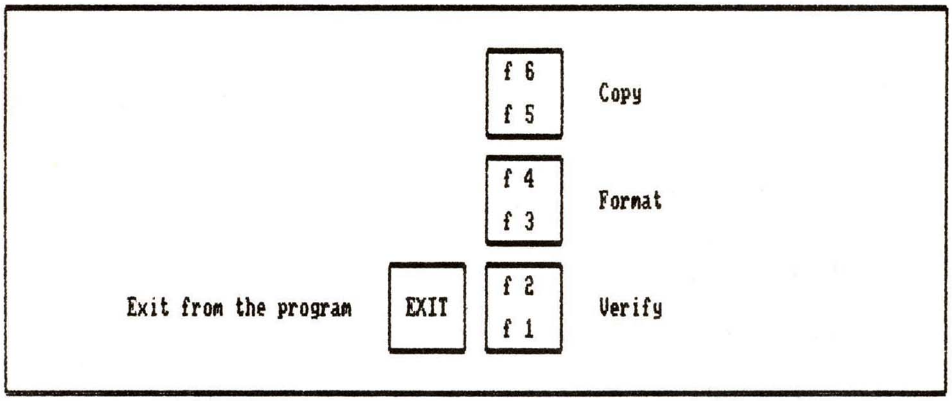


Figure 4.1 *The DISCKIT menu*

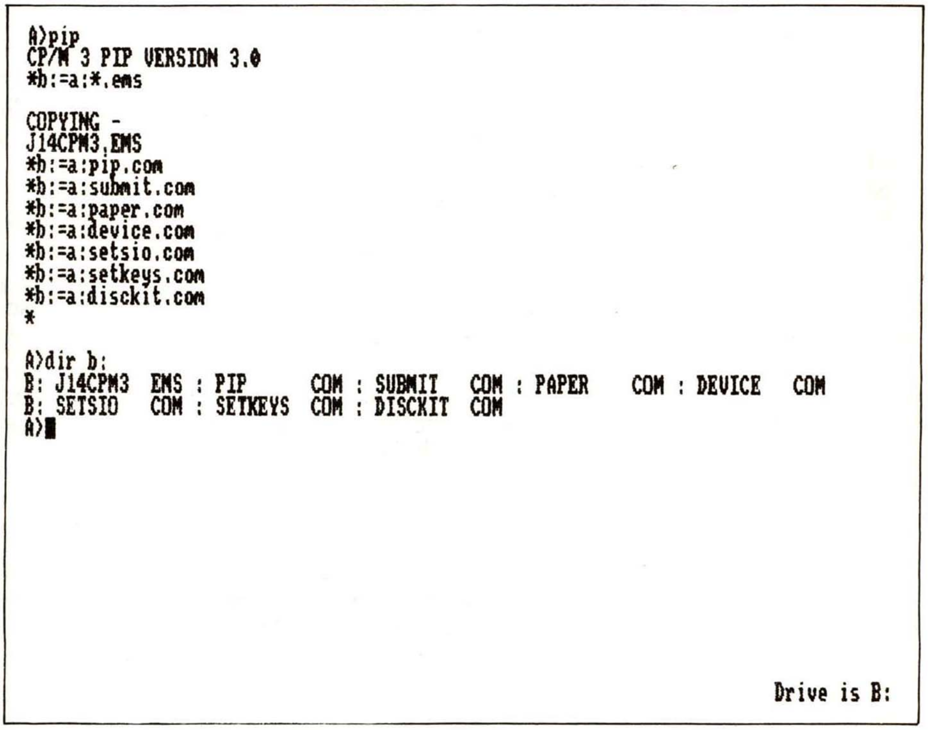
You should only need to use the original disk again if the *master copy* you have just created becomes unreliable for any reason.

Making a work disk

The next stage is to create a *work disk*. This differs from the master copy in that it should contain some of the more useful CP/M commands (such as DISCKIT). If there is sufficient space it is also worth adding the *system file*. This converts the disk into a *system disk* that can be used for starting up the system; the CP/M disk is not then needed each time you want to use your database.

The spare space could be used for storing your data files. However, this is not a particularly good idea since it means that there will be a danger of deleting the program files when carrying out housekeeping tasks. If the program files are included on every data disk then a lot of space will be wasted.

Start by making a further duplicate of the master copy. Then leave DISCKIT



```
A>pip
CP/M 3 PIP VERSION 3.0
*b:=a:*.ems

COPYING -
J14CPM3.EMS
*b:=a:pip.com
*b:=a:submit.com
*b:=a:paper.com
*b:=a:device.com
*b:=a:setsio.com
*b:=a:setkeys.com
*b:=a:disckit.com
*

A>dir b:
B: J14CPM3  EMS : PIP      COM : SUBMIT   COM : PAPER    COM : DEVICE   COM
B: SETSIO   COM : SETKEYS  COM : DISCKIT  COM
A>█

                                                                Drive is B:
```

Figure 4.2 *Copying files*

and load the PIP program by putting the CP/M disk in drive A and entering the command:

```
PIP
```

The * prompt is then displayed. To copy the system file to the work disk enter the instruction:

```
B: = A: *.EMS
```

The work disk should be in drive B (on a double-drive Amstrad). For a single-drive machine you are requested to swop disks.

Other utilities (such as PIP itself) can be copied across with instructions similar to:

```
B: = A:PIP.COM
```

The capacity of a disk (one side) is 173K. The system file takes up 40K. The amount of space left depends on the size of the program files. The utilities that are most useful include:

SUBMIT.COM	(6K)	Runs a batch file
PAPER.COM	(2K)	Sets page length, etc.
DEVICE.COM	(8K)	Changes details of printer data format (from the Programming Utilities disk)
SETSIO.COM	(2K)	Changes serial printer data format
SETKEYS.COM	(2K)	Sets up keyboard
PIP.COM	(9K)	Copies files
DISCKIT.COM	(7K)	Copies and formats disks

In some cases you may need a text editor (such as RPED.COM, which also requires the file BASIC.COM) or a word processor.

Further information on how much space is available on the work disk for each database program is given in the relevant chapters. If you have sufficient disks available it is probably also worth making a backup copy of the work disk.

Preparing data disks

Finally, you will need some blank disks on which to store your data files. A disk can only be used if it has been formatted for use with your particular computer. Therefore you should use the FORMAT option from the DISCKIT menu to prepare at least one new data disk. Remember that formatting is a very drastic process that destroys any existing data on a disk; if you intend to format a used disk be sure that it does not contain anything of importance.

Using the printer

The communication of data between the computer and a printer is usually the most difficult part of the system to set up.

Sending information out of the computer in the right order and according to specific conditions is generally no problem. Making sure that it arrives at the other end and is printed out in the manner intended is a very different matter. If you decide to use the standard printer supplied with the Amstrad PCW then you should not have too many problems. Certainly this printer works quite happily with each of the programs described in this book.

However, if you want to attach a different type of printer then life begins to get a little more complicated. First of all, of course, you must attach a suitable printer interface to the back of the monitor unit. A special cable is needed to connect the interface to the printer. Finally, instructions must be given to the computer so that it is aware of where to find the printer and the way in which it expects to receive data.

Printer types

Printers come in many different makes and models and unfortunately there is very little standardisation in the way they handle data. Essentially there are two main types of printer: serial and parallel. This refers to the way in which the data is transmitted between computer and printer. A few printers can be switched between the two types of data format.

Parallel printers are the easier type to cope with and should cause very few problems. It is simply a question of using the right type of cable to connect the printer to the computer and stating before you start the program that you wish to use parallel output (see below).

If using a *serial printer* the problems are far greater. The data is transmitted in a long stream of individual *bits* (0s and 1s: the binary code understood by the computer) and consequently various complications can arise. There are various speeds that can be used (the *baud rate*), as well as a variety of ways in which the printer may check for errors (the *parity*). There are also different methods of identifying the start and end of each item of data.

Some printers have small internal switches, known as *dip switches*, that allow you to vary the way in which data is treated; each switch can be either on or off and the combination of a number of switches determines the data format. Similarly, some programs allow you to decide the data format that is to be used; there is then a good chance that the two ends of the communication link can be made to match up and the printing will be successful. All too often, however, the program only allows a limited choice of printers and the printers are inflexible in the formats that can be used. The chances of producing a successful printout can sometimes be quite slim, if not altogether impossible. A great deal of time can be wasted in the frustrating business of persuading a printer to accept data from a particular program.

The best approach is to make sure that you see the combination of computer, program and printer all working successfully together before you buy them. If you buy the components separately then ask your dealer to make sure that the link-up can be achieved before you make a final decision, and ask him to show you how to do it.

Paper types and sizes

Before running the program - whether using the Amstrad printer or some other printer - you should tell the system what type of paper you are using. Whenever you start up the Amstrad it will be automatically set up for single sheets of A4 paper. For database programs in particular this is unlikely to be satisfactory and continuous paper is much more likely to be used. This can be done with the PAPER command by specifying the page length in inches. For example, if standard 11-inch continuous paper is used the command is:

```
PAPER 11
```

To change back to either A4 or A5 single sheets use one of these commands:

```
PAPER A4
PAPER A5
```

Printing now stops at the end of each page to allow you to load a new sheet. Other details that can be changed with the PAPER command include:

F n	Sets form (paper) length to n lines
G n	Sets gap length (number of lines skipped at end of page) to n lines
L n	Line pitch (number of lines per inch); either 6 or 8
S	Single sheets
C	Continuous

The default values that are used unless specified otherwise are:

Continuous	Form length as specified, gap length 0, line pitch 6
A4	Form length 70, gap length 3, line pitch 6
A5	Form length 50, gap length 3, line pitch 6

For example, to use 12-inch continuous paper, with 8 lines per inch and skipping 10 lines at the end of each page, enter the command:

```
PAPER 12, L 8, G 10
```

These details must be set each time the computer is switched on. It may be useful to include them in a batch file (see below).

The PTR key At any time, even when in the middle of using the database program, you can vary the printout by pressing the PTR key. This displays a list of options (which also appear whenever you change sheets of paper if set for A4 or A5). For example, you can move the paper to the top of the next sheet (FF) or advance it one or more lines (LF). You can also change between *draft quality* (which prints very quickly) and *high quality* (which is easier to read). RESET erases any data that is held in the printer's own internal memory waiting to be printed. Select any of these options by highlighting them with → and then press the + or – keys on either side of the space bar. Press EXIT to resume printing.

Data formats
If you are using the printer interface, rather than the standard Amstrad printer, you must inform the system of this fact.

Before starting the program you must have entered the appropriate DEVICE or SETSIO command to set up the printer for the correct *port* (the point to which the printer is connected). This is either CEN for the parallel port or SIO for the serial port. It may also be necessary to specify such details as baud rate, parity, data bits and stop bits. The DEVICE command is to be found on the UTILITIES disk. To set up a parallel printer use the command:

```
DEVICE LST: = CEN
```

For a serial printer, running at 1200 baud with no parity, 7 data bits and 1 stop bit, enter:

```
SETSIO 1200, P NONE, B 7, S 1
```

SETSIO can include the following *parameters*:

n	The baud rate, e.g. 1200, 9600
P	The parity: EVEN, ODD or NONE
B	Data bits: 5, 6, 7 or 8
S	Stop bits: 1, 1.5 or 2
X	Whether XON protocol is ON or OFF
H	Whether handshaking is ON or OFF

All the necessary parameters should be found in the printer manual. Remember that the values given in the printer manual can sometimes be changed by adjusting the printer's internal dip switches. A full discussion is beyond the scope of this book. The dealer who sold you the printer interface and/or printer should be able to provide the necessary information. Indeed, before spending a lot of money on the interface (which is fairly expensive compared to the initial cost of the Amstrad) it is a good idea to see the Amstrad demonstrated satisfactorily with your printer. A great deal of time can be spent setting up printers and there is no guarantee that any particular combination will be effective.

At the A> prompt you can return to the standard Amstrad printer with the command:

```
DEVICE LST:=LPT
```

Always make sure you have saved all your data before trying to use another printer in case you have to reset the system to recover from the situation.

Using the PROFILE.SUB batch file
The printer instructions can all be placed in a *batch file*. This is simply a list of CP/M instructions that can be put into effect with a single command. The batch file must have a .SUB extension. Its instructions are executed with the SUBMIT command. For example, a PRINTER.SUB file can be put into effect with:

```
SUBMIT PRINTER
```

Alternatively, you can use the special file, PROFILE.SUB. If this file exists on the system disk the commands it contains are automatically put into effect every time the system is reset. This file may also need to include a SETKEYS instruction to set up the keyboard and can automatically run the database program by ending with the appropriate command. For example, to set up a parallel printer, with 11-inch continuous paper, and then run the Condor 1 program, the following PROFILE.SUB file is required:

```
DEVICE LST:=CEN
PAPER 11
DBMS
```

This batch file can be created with a command in the form:

```
PIP PROFILE.SUB=CON:
```

The instructions are typed at the keyboard. Complete each one by pressing RETURN; move down a line with ALT-J. The file is terminated by pressing ALT-Z. Alternatively, such a file can be created or edited with a text editor such as RPED.

You should now be ready to run a database program. The remainder of the book describes some of the programs that are available for the Amstrad PCW8256 and PCW8512.

CHAPTER 5

THE MATCHBOX ELECTRONIC FILING SYSTEM

The first database program to be considered is the Matchbox Electronic Filing System from Quest Automation Plc. This is by far the simplest of the six programs described in this book. It is certainly easy to master, but its facilities are correspondingly lacking in sophistication.

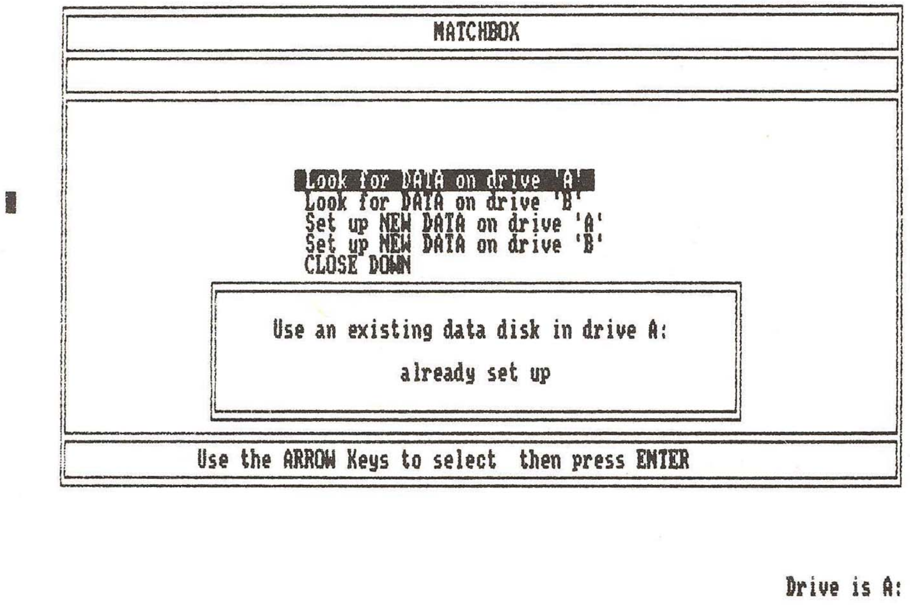
Initial preparations

The Matchbox program files take up very little space (only 47K) and therefore there is sufficient room on the work disk for all the CP/M utility files that you are likely to need. The program files are read into memory when the program is put into effect so there is no need to use the memory drive.

Any changes to the default printer specifications should be made before you start the program. No further preparations are needed before you use Matchbox. It is not even really necessary to read the manual!

Creating a data file

The Matchbox program disk is not a *system disk*. That is, it doesn't have the ability to load the operating system into memory, an essential prerequisite to



```
MATCHBOX

Look for DATA on drive 'A'
Look for DATA on drive 'B'
Set up NEW DATA on drive 'A'
Set up NEW DATA on drive 'B'
CLOSE DOWN

Use an existing data disk in drive A:
already set up

Use the ARROW Keys to select  then press ENTER

Drive is A:
```

Figure 5.1 *The initial menu*

run any application program. Therefore you must start the system with a suitable system disk, such as the CP/M Plus disk. When the A> prompt is displayed remove the system disk and replace it with your Matchbox work disk. Then enter the command:

```
M
```

The program is read into memory and put into effect. After the introductory copyright message has cleared from the screen an initial *menu* is displayed. It is a very simple set of five choices, two to inspect existing data files, two to create new files and a fifth one to leave Matchbox, returning to CP/M Plus.

Selection is achieved by pressing either the ↑ or ↓ keys, then pressing RETURN when the required option is highlighted.

A further explanation of each option is displayed in the box below the menu.

Defining a file
To create a new file, select either the third or fourth options from the menu, depending on the location of the data disk. Then put a blank, formatted disk in drive B (on a double-drive system) or replace the program disk with a blank disk in drive A (for a single-drive system).

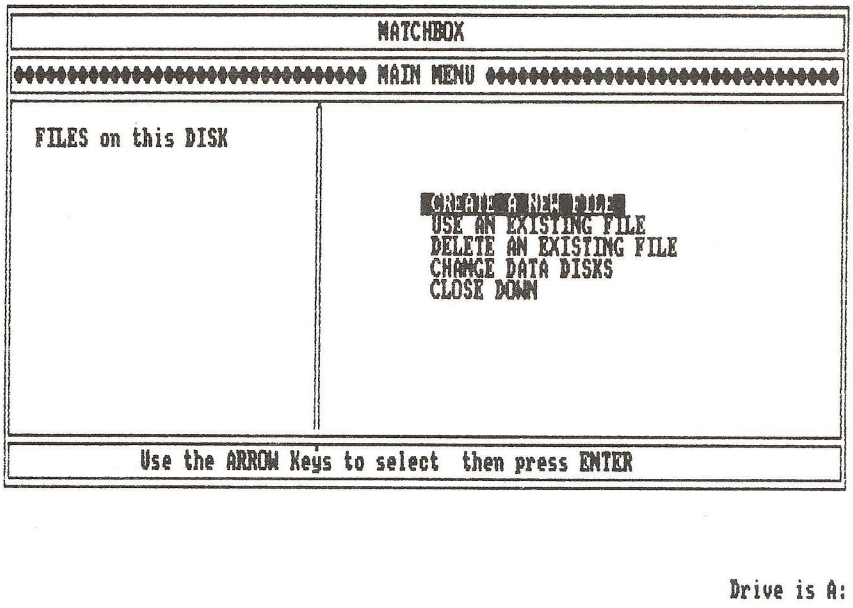

```
MATCHBOX
♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦ MAIN MENU ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦

FILES on this DISK

                                    CREATE A NEW FILE
                                    USE AN EXISTING FILE
                                    DELETE AN EXISTING FILE
                                    CHANGE DATA DISKS
                                    CLOSE DOWN

          Use the ARROW Keys to select  then press ENTER

                                                          Drive is A:
```

Figure 5.2 *The main menu*

The main menu is displayed, with five more options. If any data files already exist these are listed on the left of the screen. Select the top option to *Create a New File*. Now enter a name of up to eight characters. As for any CP/M filename you can use letters, numbers and certain symbols. A .DAT extension is automatically added by Matchbox.

For example, for a file to record a magazine's subscribers the file can be called SUBS and will be stored on disk as:

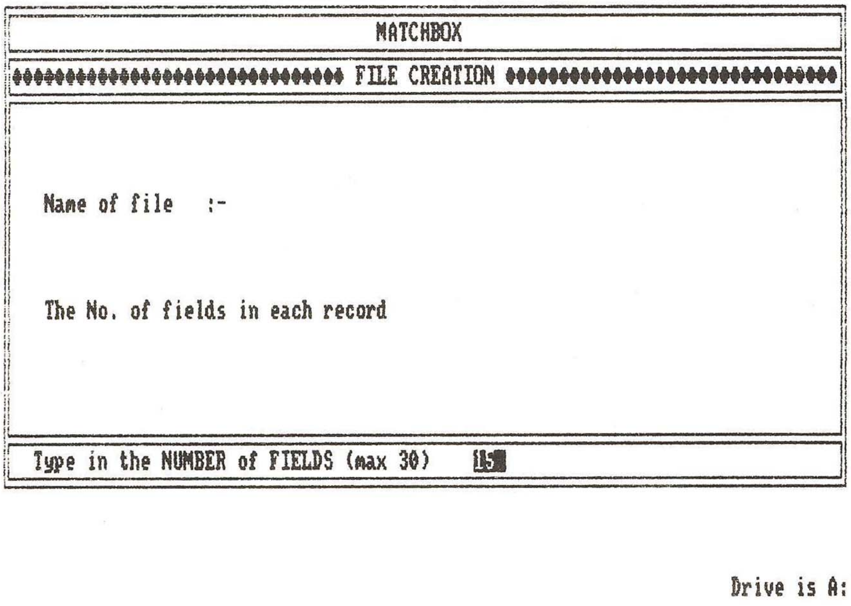
SUBS.DAT

Choose a name that will adequately reflect the contents of the file, without making it more complex than is necessary.

Defining the fields

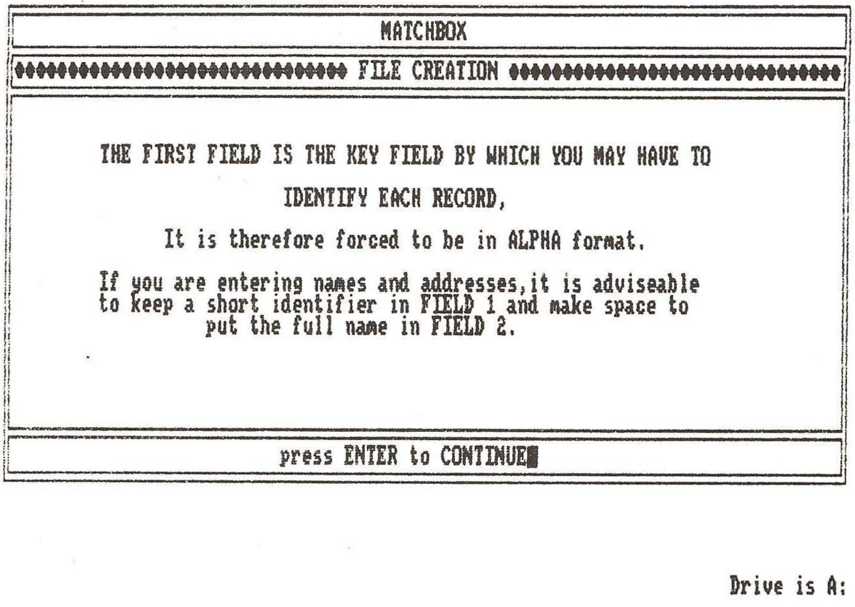
The next stage is to identify the number of fields that will make up each record. If you have already done your homework then this should be quite straightforward. You can have up to 30 fields for a record and you should allow for possible future expansion (as described below).

After entering the number of fields you must describe what types of field they are to be. In each case you must specify a name for the field, the



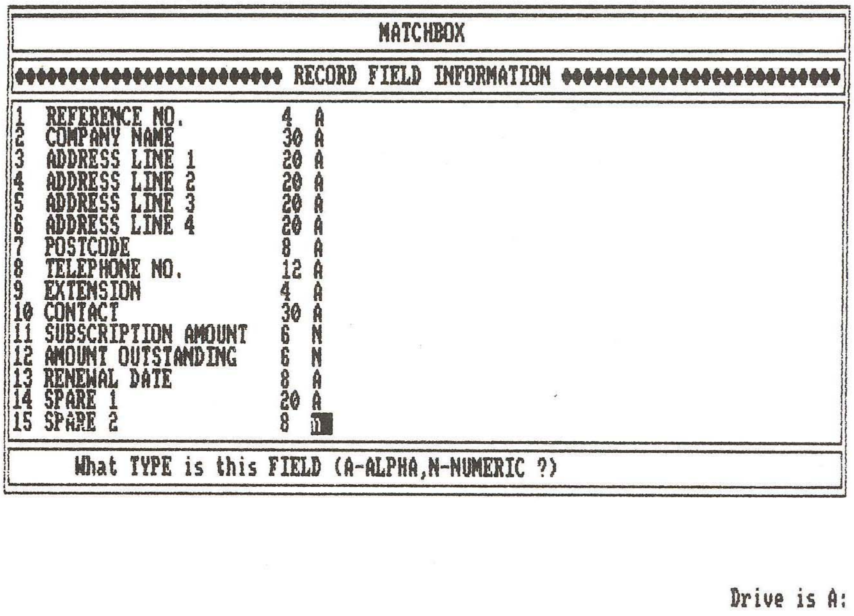
```
                              MATCHBOX
♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦ FILE CREATION ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦

  Name of file   :-


  The No. of fields in each record


 Type in the NUMBER of FIELDS (max 30)    15


                                                                Drive is A:
```

Figure 5.3 *Choosing the number of fields*



```
                              MATCHBOX
♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦ FILE CREATION ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦

      THE FIRST FIELD IS THE KEY FIELD BY WHICH YOU MAY HAVE TO

                        IDENTIFY EACH RECORD,

              It is therefore forced to be in ALPHA format.

      If you are entering names and addresses,it is adviseable
      to keep a short identifier in FIELD 1 and make space to
              put the full name in FIELD 2.


                       press ENTER to CONTINUE


                                                                Drive is A:
```

Figure 5.4 *Key field message*



```
                              MATCHBOX
♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦ RECORD FIELD INFORMATION ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
1  REFERENCE NO.          4  A
2  COMPANY NAME          30  A
3  ADDRESS LINE 1        20  A
4  ADDRESS LINE 2        20  A
5  ADDRESS LINE 3        20  A
6  ADDRESS LINE 4        20  A
7  POSTCODE               8  A
8  TELEPHONE NO.         12  A
9  EXTENSION              4  A
10 CONTACT               30  A
11 SUBSCRIPTION AMOUNT    6  N
12 AMOUNT OUTSTANDING     6  N
13 RENEWAL DATE           8  A
14 SPARE 1               20  A
15 SPARE 2                8  N
      What TYPE is this FIELD (A-ALPHA,N-NUMERIC ?)

                                                           Drive is A:
```

Figure 5.5 *Defining the fields*

maximum length for the data on the screen and the field's type. The name can be up to 20 characters long, and any name is acceptable. It should obviously relate to the contents of the field and should indicate to anyone using the system the data they are expected to enter. Names are always converted to upper case.

The length of the field is the maximum number of characters that can be used for the data. For text fields, you should ensure, where possible, that they can take the longest item of text you are likely to come across; otherwise some form of abbreviation will be necessary when the data is actually typed in. The maximum length is 40 characters. For numeric fields, remember to allow for a minus sign and decimal point (where applicable). For example, money fields up to £10000 require 8 characters (to accomodate -9999.99).

Note that a money field must consist only of numeric characters and cannot include the £ symbol. If you want to display a monetary symbol then the field should be set up as text. To make such fields line up on the decimal point they should be padded out with spaces when they are entered.

Although you will want to allow for the longest expected values there is a maximum total length for all fields of 255 characters. Therefore if you have a large number of fields there will have to be a trade-off, with some of the least important fields being reduced in size. For example, you may be able to reduce the length allowed for addresses by using suitable abbreviations, but you cannot reduce the size of surnames.

Only two field types are allowed for in Matchbox: numeric and *alpha* (another name for text fields). The differences between these types are very limited:

- *Alpha* fields always start on the left-hand side of the field; any characters are allowed.
- *Numeric* fields always end on the right-hand side of the field; only numeric characters can be included.

The first field must be an alpha field; it is used to identify individual records and therefore its value must be unique in each case. The value should also be kept reasonably short for convenience.

Mistakes cannot be corrected until all fields have been entered.

Example With magazine subscriptions it is necessary to record details such as name, address and telephone number, subscription amount and date due.

The fields for the SUBS file may be:

FIELD NAME	LENGTH	TYPE
Reference No.	4	A
Company Name	30	A
Address Line 1	20	A
Address Line 2	20	A
Address Line 3	20	A
Address Line 4	20	A
Postcode	8	A
Telephone No.	12	A
Extension	4	A
Contact	30	A
Subscription amount	6	N
Amount outstanding	6	N
Renewal date	8	A
Spare 1	20	A
Spare 2	8	N

Most of these are fairly self-explanatory. However, two additional fields have been added at the end. Matchbox does not allow any further changes to be made once the record structure has been confirmed. Therefore it is as well to allow for extra information, just in case you later decide to increase the scope of the data stored in a file.

Matchbox works by allocating 255 characters on disk for each record, regardless of the total length of the fields specified. Therefore the extra space at the end of the record can be set aside for future expansion. The problem of how to allocate that space is not easy to resolve and will depend on the type of data likely to be stored. However, you should note that numeric fields can only hold one item of data and therefore there should be enough such fields to satisfy possible future needs. Alpha fields can always be further partitioned; for example, the first 10 characters may be used for an additional reference number while the next 10 could take a second telephone number.

When all the fields have been entered you are given the choice of making changes or accepting the structure as shown. Use ↑ to change the response to 'YES'.

Matchbox is very unforgiving of those who make mistakes while designing the record structure. Once it has been accepted there is no other opportunity to change any of the fields or to add new fields. Nor is there any option for transferring data to another file. If you do make a major blunder all your data will have to be re-entered in a new file, so make sure that you do ample preparatory work before sitting down to enter the field details.

Using a data file

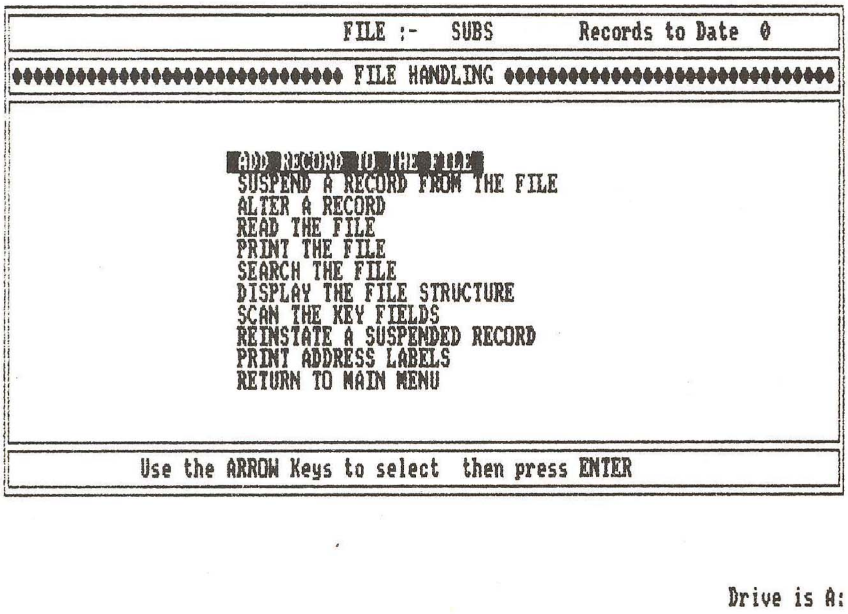
After completing the structure you are returned to the main menu. Select the option to *Use an Existing File*, choose the filename and a new list of options is displayed. All instructions in Matchbox are in the form of menu options, selected with ↑ or ↓ and RETURN. This makes the program simple to use, though limits its scope.

Entering data

Entry of records is very straightforward. Selecting the first item in the list results in a display of the fields you have chosen with their *attributes* (length and type). The data is then typed straight in against the field names. To accept the entry press RETURN; to leave an entry blank press RETURN on its own. All text is converted to upper case.

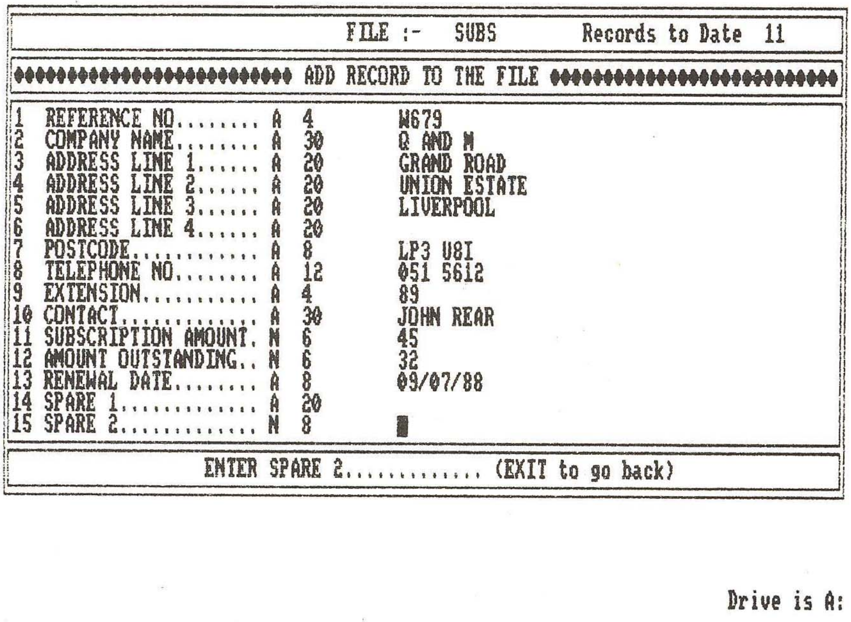
If you make a mistake you must wait until later and use the Alter option. When the last field has been entered the details are stored on disk; the data is then secure, even if the power is suddenly switched off. This is one of the major advantages of this type of database.

Further records can be entered or you can return to the main menu.



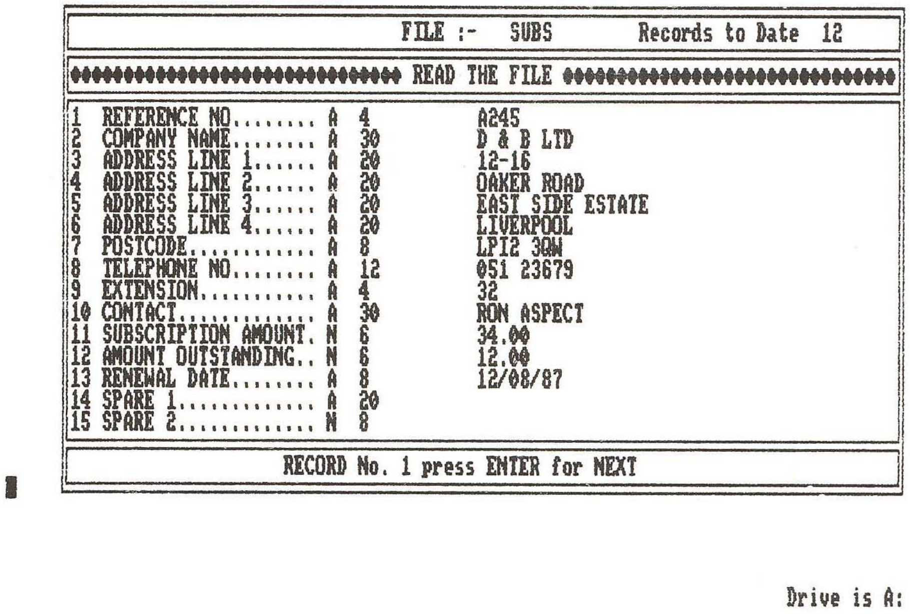
```
                          FILE :-   SUBS        Records to Date  0

♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦ FILE HANDLING ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦

                     ADD RECORD TO THE FILE
                      SUSPEND A RECORD FROM THE FILE
                      ALTER A RECORD
                      READ THE FILE
                      PRINT THE FILE
                      SEARCH THE FILE
                      DISPLAY THE FILE STRUCTURE
                      SCAN THE KEY FIELDS
                      REINSTATE A SUSPENDED RECORD
                      PRINT ADDRESS LABELS
                      RETURN TO MAIN MENU

              Use the ARROW Keys to select  then press ENTER

                                                          Drive is A:
```

Figure 5.6 *The File Handling menu*



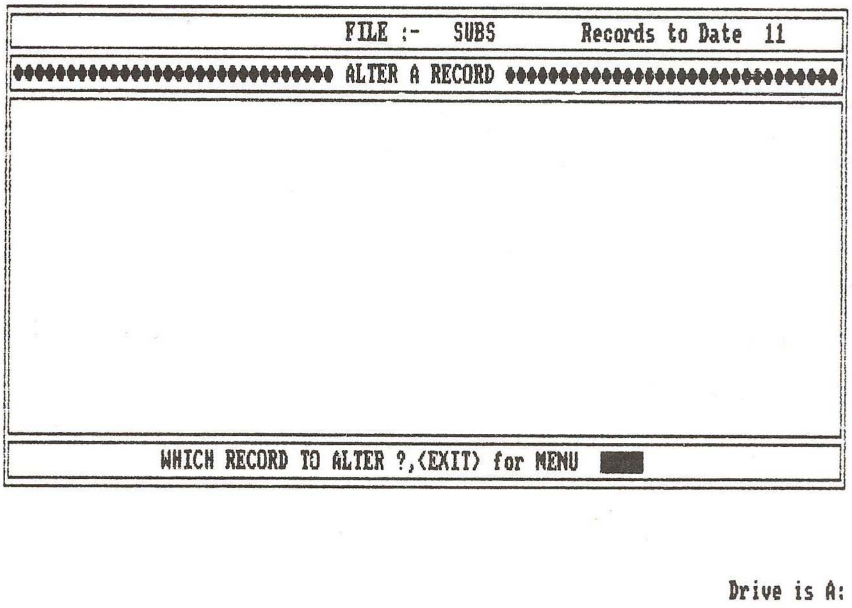
```
                          FILE :-   SUBS        Records to Date  11

♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦ ADD RECORD TO THE FILE ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
1  REFERENCE NO........ A  4      W679
2  COMPANY NAME........ A  30     Q AND M
3  ADDRESS LINE 1...... A  20     GRAND ROAD
4  ADDRESS LINE 2...... A  20     UNION ESTATE
5  ADDRESS LINE 3...... A  20     LIVERPOOL
6  ADDRESS LINE 4...... A  20
7  POSTCODE............ A  8      LP3 U8I
8  TELEPHONE NO........ A  12     051 5612
9  EXTENSION........... A  4      89
10 CONTACT............. A  30     JOHN REAR
11 SUBSCRIPTION AMOUNT. N  6      45
12 AMOUNT OUTSTANDING.. N  6      32
13 RENEWAL DATE........ A  8      09/07/88
14 SPARE 1............. A  20
15 SPARE 2............. N  8      █

              ENTER SPARE 2............. (EXIT to go back)

                                                          Drive is A:
```

Figure 5.7 *Entering a record*



```
                         FILE :-   SUBS        Records to Date  12

♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦ READ THE FILE ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦

1  REFERENCE NO........ A  4        A245
2  COMPANY NAME........ A  30       D & B LTD
3  ADDRESS LINE 1...... A  20       12-16
4  ADDRESS LINE 2...... A  20       OAKER ROAD
5  ADDRESS LINE 3...... A  20       EAST SIDE ESTATE
6  ADDRESS LINE 4...... A  20       LIVERPOOL
7  POSTCODE............ A  8        LPI2 3QW
8  TELEPHONE NO........ A  12       051 23679
9  EXTENSION........... A  4        32
10 CONTACT............. A  30       RON ASPECT
11 SUBSCRIPTION AMOUNT. N  6        34.00
12 AMOUNT OUTSTANDING.. N  6        12.00
13 RENEWAL DATE........ A  8        12/08/87
14 SPARE 1............. A  20
15 SPARE 2............. N  8

               RECORD No. 1 press ENTER for NEXT

                                                           Drive is A:
```

Figure 5.8 *Inspecting a record*



```
                         FILE :-   SUBS        Records to Date  11

♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦ ALTER A RECORD ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦




            WHICH RECORD TO ALTER ?,<EXIT> for MENU

                                                           Drive is A:
```

Figure 5.9 *Choosing a record to edit*

Inspecting records
To look at the contents of the file, choose the option to *Read the File*. You must enter the numbers of the first and last records you want to look at. This means that you need some idea of where in the file the records are located. The records are automatically numbered from 1, the total number of records being shown at the top of the screen. (It may be helpful to define the first field as the record number.)

The records are shown in the order they were entered. Your only option is to press RETURN to move on through the file. To return to the main menu before reaching the end of the selected range, press EXIT.

Editing records
Changes to specific records can be made with the option to *Alter a Record*. Enter the record number and then, for each field, either press RETURN to accept the details as shown or completely re-enter the value. No opportunity is given to edit any existing text or number.

After accepting the changes you are given the option to enter another number if you want to make further changes.

Searching, selection and sorting

Locating records according to specific criteria is rather primitive, though reasonably effective. With the *Search the File* option you can look at records according to the contents of one field only. Enter the field number and then the text or value that is to be located. This can be either the complete entry or just a string of characters. Next you need to decide whether the located records are to be displayed one at a time or printed out in a list format.

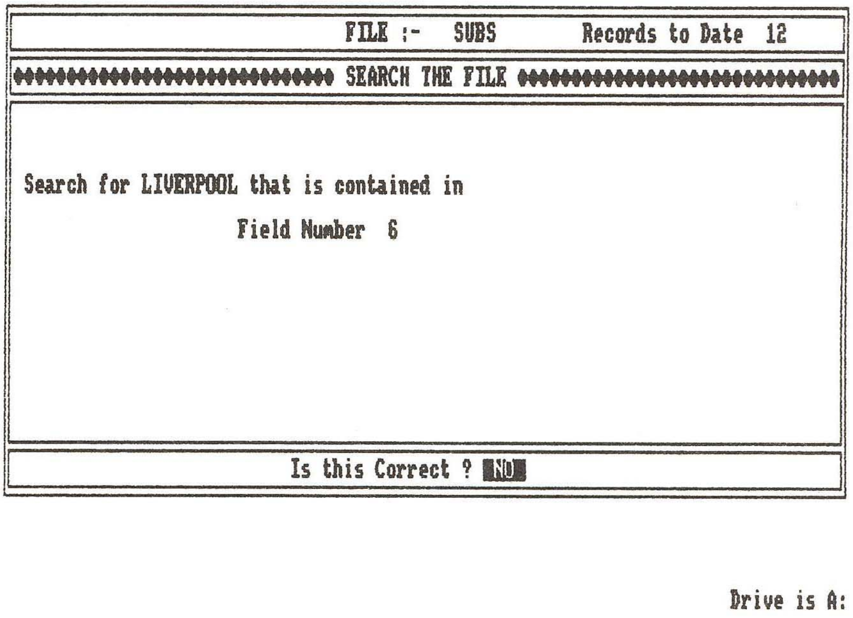
Finally, choose the range of records that is to be searched. This can provide an extra opportunity to reduce the selection that is made.

For example, suppose you want to select all records in the SUBS file in which subscriptions fall due on a given date. You can locate these records by using the Search options as follows:

Field:	13 (Renewal date)
Value:	01/11/86
Search type:	1 (Exact match)
Screen or printer:	S

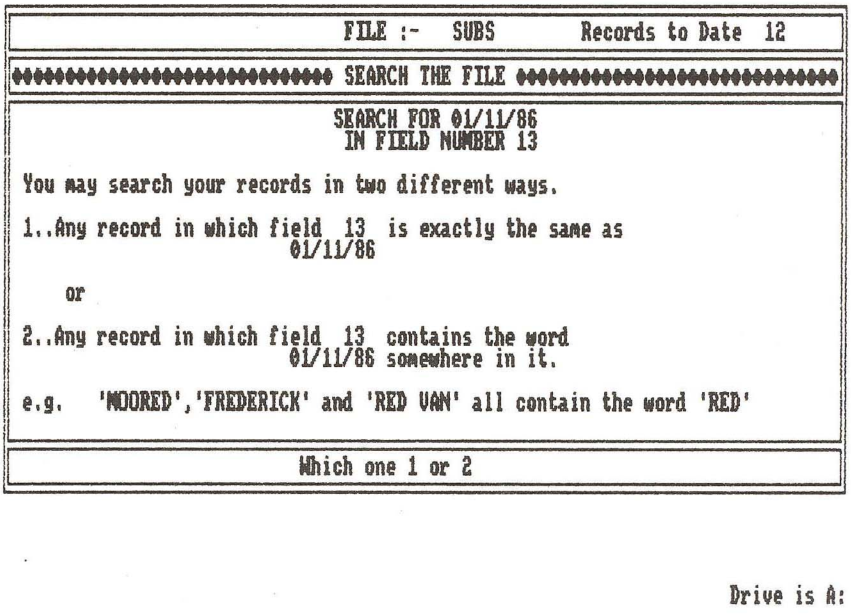
Keep pressing RETURN until the last record has been found.

To search for all records in a given month the search criteria are:



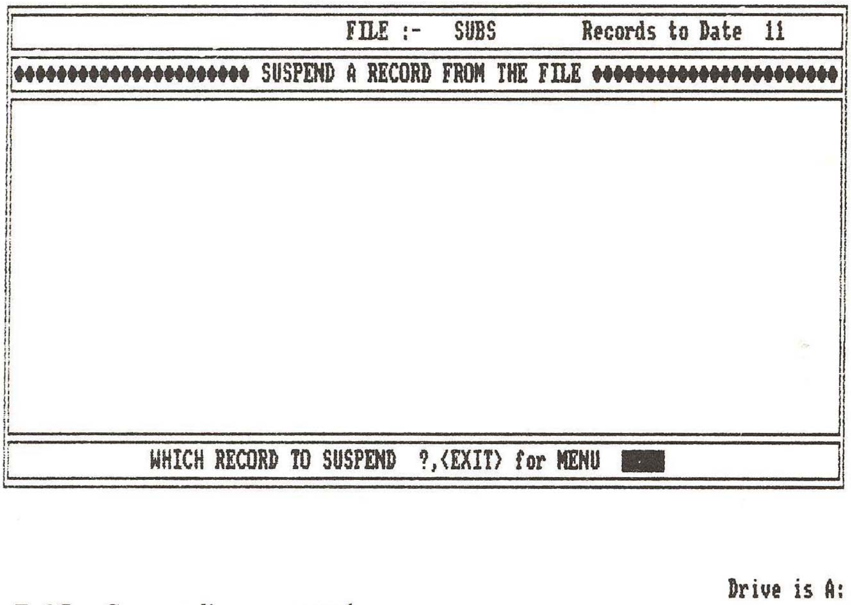
```
                              FILE :-   SUBS       Records to Date  12
♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦ SEARCH THE FILE ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦

Search for LIVERPOOL that is contained in

                    Field Number  6

                         Is this Correct ? NO

                                                                Drive is A:
```

Figure 5.10 *Choosing the search criteria*



```
                              FILE :-   SUBS       Records to Date  12
♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦ SEARCH THE FILE ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
                              SEARCH FOR 01/11/86
                               IN FIELD NUMBER 13
You may search your records in two different ways.

1..Any record in which field  13  is exactly the same as
                          01/11/86

   or

2..Any record in which field  13  contains the word
                          01/11/86 somewhere in it.

e.g.   'MOORED','FREDERICK' and 'RED VAN' all contain the word 'RED'

                          Which one 1 or 2

                                                                Drive is A:
```

Figure 5.11 *Choosing the search method*



```
                          FILE :-   SUBS        Records to Date  11
♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦ SUSPEND A RECORD FROM THE FILE ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦






            WHICH RECORD TO SUSPEND  ?,<EXIT> for MENU

                                                              Drive is A:
```

Figure 5.12 *Suspending a record*

Field:	13 (Renewal date)
Value:	11/86
Search type:	2 (Contains match)
Screen or printer:	S

The only problem here is that for a large file it can take some considerable time to move through all the selected records.

Careful selection of the search field can often make life much easier. For example, searching for subscribers in Liverpool would require these details:

Field:	6 (Address Line 4)
Value:	Liverpool
Search type:	1 (Exact match)

This will only be successful if the postal town is always placed on the last address line, regardless of the length of the address. Because some addresses are very short it may be better to conduct a search on the postcodes:

Field:	7 (Postcode)
Value:	LP
Search type:	2 (Contains match)

All addresses with Liverpool postcodes are now shown.

Selection
In the example above there may well be unwanted records. For example, anyone with a postcode such as HR1 3LP will also be found. In these cases such files can be specifically excluded from the list with the *Suspend a Record* option. Enter the record number and confirm that this is the one by selecting 'YES'. This record will not now show up in any operation to inspect, edit, search or print the file, until it is reinstated with the appropriate option.

Therefore the usual procedure when producing printed lists is to display all selected records on the screen, noting those that should be excluded, suspend the unwanted files and then repeat the search with the option to print. Don't forget to reinstate the suspended records afterwards, otherwise this may cause some confusion in later searches.

Sorting
There is no option to sort the file. If you want all records to appear in a certain order then they must be entered in that order.

Reports

There are a number of ways that Matchbox can be used to produce a printed list of records. The Search facilities described above can result in a printout of all selected records. Alternatively the option to *Print the File* will list all records in a given range.

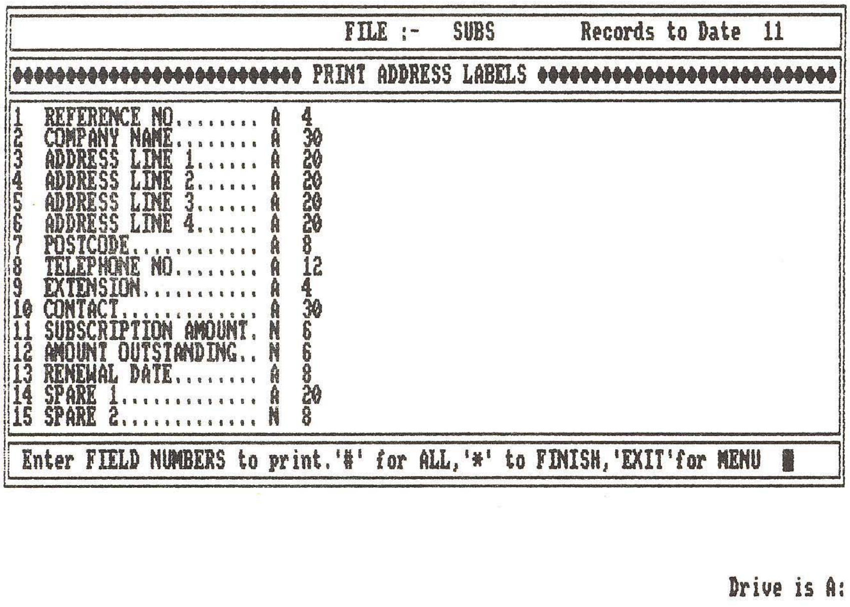
When printing the file you can select the fields that are to be printed. Supply the field numbers and end the list with an asterisk (*), or select all fields by pressing the # key.

The printout is in the same format as the screen displays.

Printed labels
One further method of printing records is provided which is a useful addition to the program. This is the *Print Address Label* option which can be used not only for printing labels but also for other purposes if a little imagination is applied.

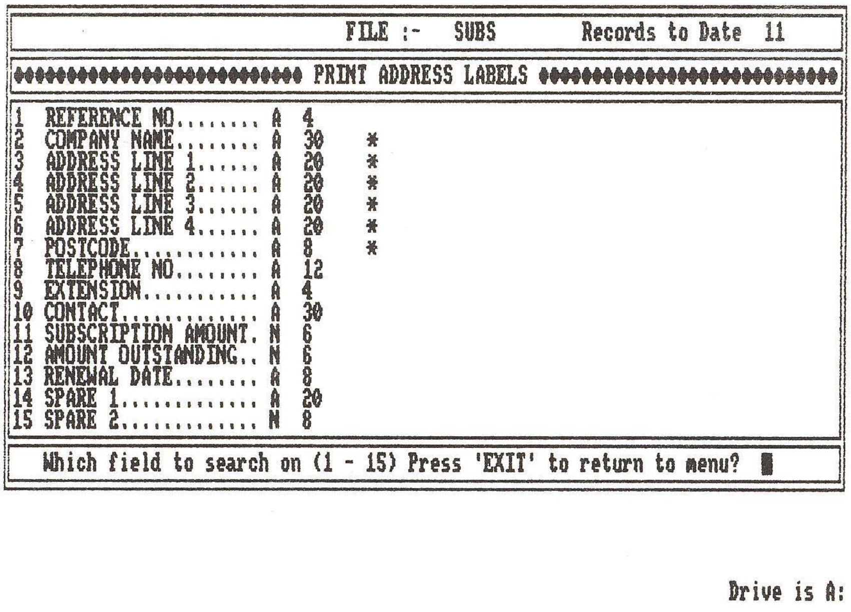
It is intended for the printing of address labels and is designed to work with rolls of labels with sprocket holes. These can be fed through the Amstrad printer if it has the tractor unit attached. Alternatively the labels can be printed on blank paper.

In the same way as for printing all records you must state the range of



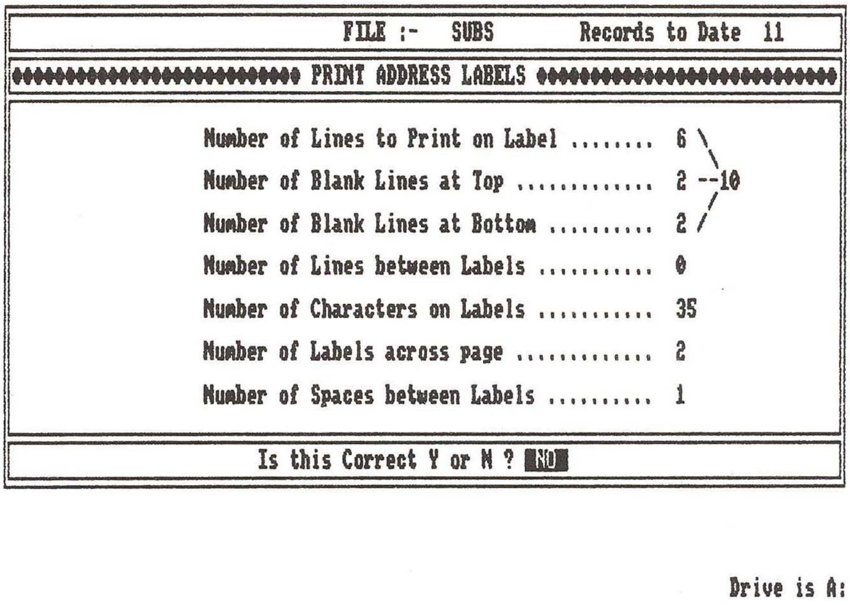
```
                                   FILE :-   SUBS        Records to Date  11
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆ PRINT ADDRESS LABELS ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
1  REFERENCE NO........  A  4
2  COMPANY NAME........  A  30
3  ADDRESS LINE 1......  A  20
4  ADDRESS LINE 2......  A  20
5  ADDRESS LINE 3......  A  20
6  ADDRESS LINE 4......  A  20
7  POSTCODE............  A  8
8  TELEPHONE NO........  A  12
9  EXTENSION...........  A  4
10 CONTACT.............  A  30
11 SUBSCRIPTION AMOUNT.  N  6
12 AMOUNT OUTSTANDING..  N  6
13 RENEWAL DATE........  A  8
14 SPARE 1.............  A  20
15 SPARE 2.............  N  8
 Enter FIELD NUMBERS to print.'#' for ALL,'*' to FINISH,'EXIT'for MENU █

                                                                 Drive is A:
```

Figure 5.13 *Choosing the fields to print*



```
                                   FILE :-   SUBS        Records to Date  11
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆ PRINT ADDRESS LABELS ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
1  REFERENCE NO........  A  4
2  COMPANY NAME........  A  30    *
3  ADDRESS LINE 1......  A  20    *
4  ADDRESS LINE 2......  A  20    *
5  ADDRESS LINE 3......  A  20    *
6  ADDRESS LINE 4......  A  20    *
7  POSTCODE............  A  8     *
8  TELEPHONE NO........  A  12
9  EXTENSION...........  A  4
10 CONTACT.............  A  30
11 SUBSCRIPTION AMOUNT.  N  6
12 AMOUNT OUTSTANDING..  N  6
13 RENEWAL DATE........  A  8
14 SPARE 1.............  A  20
15 SPARE 2.............  N  8
   Which field to search on (1 - 15) Press 'EXIT' to return to menu? █

                                                                 Drive is A:
```

Figure 5.14 *Selecting a search field*



```
                    FILE :-   SUBS        Records to Date  11

♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦ PRINT ADDRESS LABELS ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦

          Number of Lines to Print on Label ........  6 \
                                                         \
          Number of Blank Lines at Top .............  2 --10
                                                         /
          Number of Blank Lines at Bottom ..........  2 /

          Number of Lines between Labels ...........  0

          Number of Characters on Labels ...........  35

          Number of Labels across page .............  2

          Number of Spaces between Labels ..........  1

                     Is this Correct Y or N ? NO

                                                           Drive is A:
```

Figure 5.15 *Selecting the print format*

records to be printed and the fields to be used (for example, the company name, four lines of address and the postcode). You may also select records according to the values of the fields (enter 0 for field number if there is to be no selection).

You should specify the number of blank lines at the top and bottom of the page and the number of blank lines between labels. If using continuous sheets and labels you will have no gap at the top or bottom of the sheet; the number of fields and number of blank lines should add to the total number of lines from the top of one label to the top of the next. For blank sheets of paper the sizes are more flexible.

If you have sheets with more than one column of labels then you may specify this. In this case you must give the maximum width of each label (determined by the length of the longest field) and the number of blank columns between each label. Again, if using standard labels the number of characters per label plus the gap between labels should equate to the actual label width.

As an example, consider the use of a continuous sheet of labels, with each page 12 inches long, 6 labels per page and one column. The figures required are:

```
D & B LTD
12-16
OAKER ROAD
EAST SIDE ESTATE
LIVERPOOL
LPI2 3QW



HONEYWAY
HONEYWAY HOUSE
WHITEHOUSE LANE
MERSEY SIDE
LIVERPOOL
LP23 2FN



NCC
NIVEN GROVE
CHERRY STREET
HAVINGTON
LIVERPOOL
LP4 8JR
```

Figure 5.16 *The printed labels*

Number of blank lines at top	0
Number of blank lines at bottom	0
Number of lines between labels	6
Number of characters on labels	30
Number of labels across page	1
Number of spaces between labels	0

The number of lines between labels is calculated from:

Page length = 12 inches
Label height = 12/6 = 2 inches = 12 lines (6 lines per inch)
Lines between labels = Label height - No. of fields = 12 - 6 = 6

To print on A4 paper (which has 66 lines available after the paper has been fed into the printer, a margin at the top always being unusable) leaving an inch at the bottom and printing two columns, each with width 4 inches (at 10 characters per inch) requires the following details:

Number of blank lines at top	0
Number of blank lines at bottom	6
Number of lines between labels	4
Number of characters on labels	30
Number of labels across page	2
Number of spaces between labels	10

This will print two columns, each with six labels per page.

An additional use of this part of the program could be to overprint invoices with the name and address. In this case only one label is needed per page. If the invoices are on continuous 11-inch stationery the layout would be:

Number of blank lines at top	20
Number of blank lines at bottom	40
Number of lines between labels	0
Number of characters on labels	30
Number of labels across page	1
Number of spaces between labels	0

This assumes that the label is 2 inches from the top of the form and is to be printed on the extreme left.

With all these printing options it is probably best to have a practice run to set the spacing exactly right. Any print run can always be interrupted by pressing EXIT.

Using the printer

There are no additional commands in Matchbox for changing the way the printer behaves, although you can, of course, change between draft and high quality by pressing the PTR key.

Looking after files

Matchbox does not have any file-handling facilities as such, apart from one option in the main menu to delete a file. Therefore to make backup disks or copies of files, or to change file names you must use the CP/M Plus PIP, RENAME and DISCKIT facilities.

Support

The manual supplied with Matchbox is extremely short (only 13 pages, including a title page and contents list) but even this covers quite adequately the facilities of the program. Some of the finer details have to be worked out for yourself, but the program is not hard to use.

The second section of the manual, which is longer than the first, explains what the manufacturers will and will not do to help you, and what services you have to pay for. They explain in great detail why you should expect to pay for help and offer a support service which, in relation to the cost of the package, is quite expensive. However, the program is so simple that there should be little call for this extra help and if you do experience any problems your dealer should be able to help. Support for the first 30 days is free.

Quest do, of course, offer the standard guarantees. They will replace a faulty disk if returned within three months. If you have an unusual hardware set-up and have problems then they will sort it out (or ask your supplier to help), providing you made it plain what equipment you had before you purchased the program. Make sure that if you have any problem understanding the program it is in the first four weeks, otherwise you may have to pay for the privilege of finding out how to use the package.

Conclusion

Matchbox is simple to use and should not cause any great problems. It is, however, extremely limited in its facilities, especially for search capabilities. It is totally lacking in any form of sort routine. For certain applications it may well be sufficient and it provides a reasonable introduction to the theory of computer data storage.

CHAPTER 6

CARDBOX – THE ELECTRONIC CARD INDEX

The second database to be considered is Cardbox, published by Caxton Software Ltd. Although this is still based on the idea of the traditional card index it is very much more sophisticated than Matchbox, and correspondingly harder to use.

Initial preparations

The first operations to be performed, before starting to use Cardbox, are identical to those of Matchbox. A working copy of the program disk must be made straight away; since the original program disk is not a system disk then it is as well to duplicate all the files on a copy of the CP/M disk, creating an *auto-booting* work disk. Data disks are also needed and should be created now.

The Cardbox programs are not much larger than those of Matchbox (55K, including the terminal definition program), so there should be ample room for the most useful CP/M Plus utilities.

The program files are accessed while the program is running, so on a single-drive Amstrad they should be copied to the memory drive with the commands:

```
PIP M:=CARDBOX.*
```

```
CARDBOX
```

and

```
   PIP M: =TERMINAL.DEF
```

Then change to drive M.

On a double-drive machine you can use drive A for the program disk, drive B for the data.

Creating a data file

To start Cardbox, load the data disk into drive A on a single-drive machine, drive B for a double-drive Amstrad, and enter the instruction:

```
   CARDBOX
```

All the program files are read into memory; there is no need to use the memory drive.

The initial decision on what you want to do with the program is determined by selecting from two levels of menus; after that the activities are performed by entering two-letter commands.

The introductory screen shows the main (or 'primary') menu on the left, with the second-level menu on the right. As you change the option on the left, so the menu on the right alters correspondingly. Press P to move the pointer down through the primary menu, S for the secondary menu.

Defining a screen format

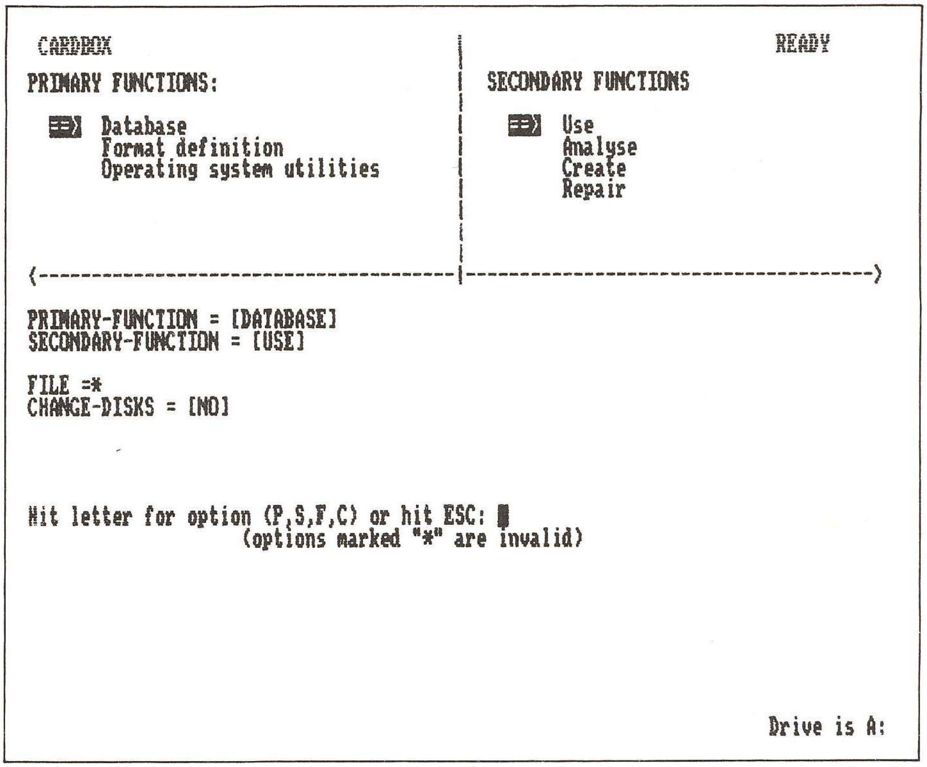
Initially you can create a file by selecting Format Definition from the first menu and Create from the second. Press F to enter a filename. The name can be up to eight characters, as usual. It should be preceded by the drive specifier. For example, on a single-drive Amstrad it can be entered as:

```
   A:SUBS
```

Cardbox creates two files, one with a .FIL extension to hold the data, the other with .FMT extension to determine the format of the screen layout and the record structure.

To put any part of the program into effect from the main menu press the rather curious combination of keys EXIT and G. (Note that the display refers to ESC, which is represented by EXIT on the Amstrad.)

Before going any further you should make sure you know exactly what you



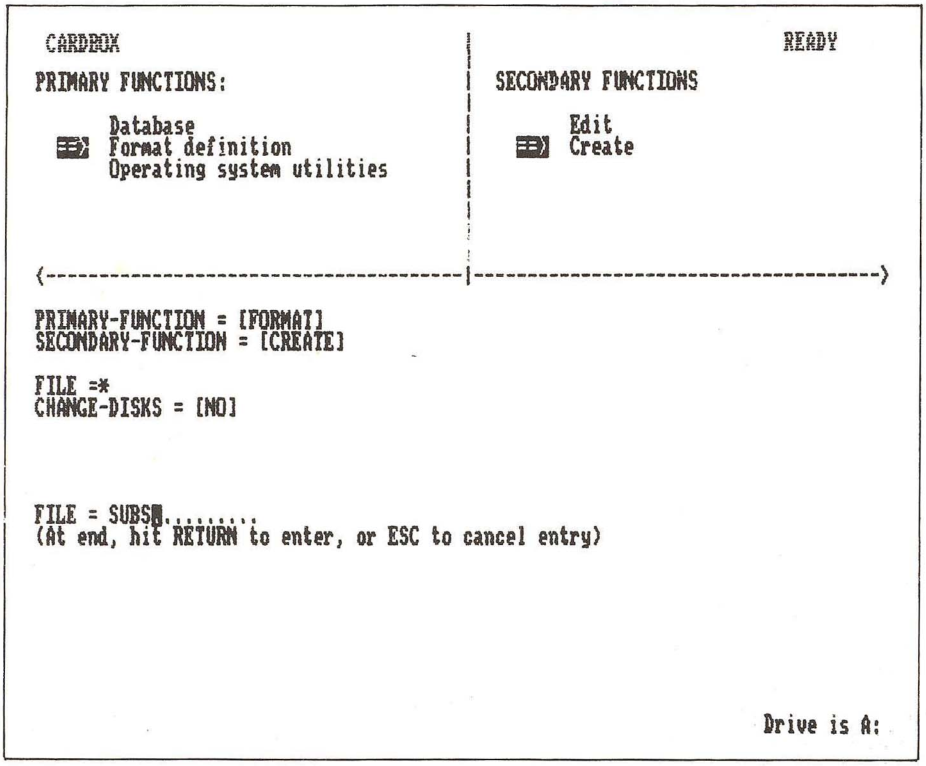
```
CARDBOX                                  |                                    READY
PRIMARY FUNCTIONS:                       |  SECONDARY FUNCTIONS
                                         |
  ==> Database                           |   ==> Use
      Format definition                  |       Analyse
      Operating system utilities         |       Create
                                         |       Repair
                                         |
                                         |
<----------------------------------------|----------------------------------->

PRIMARY-FUNCTION = [DATABASE]
SECONDARY-FUNCTION = [USE]

FILE =*
CHANGE-DISKS = [NO]


Hit letter for option (P,S,F,C) or hit ESC: █
                        (options marked "*" are invalid)




                                                                     Drive is A:
```

Figure 6.1 *The Cardbox main menus*

are going to do: you need to know where to draw lines and boxes, what explanatory text is to be added and how much space is needed for each field and each item of text. It is easiest, though perhaps a little time-consuming, to prepare your design on squared paper. You can make changes to your screen design after it has been entered but this can be quite a slow process. Of course, it's unlikely that whatever is produced on paper will look exactly the same when on screen and some adjustment is almost certain to be necessary. Note that the maximum size available on the screen is 20 lines each of 80 characters. (The full size of the Amstrad screen is not available.)

A blank screen is displayed with a number of single-letter commands shown at the bottom of the screen. Any one of these is selected by pressing the appropriate letter key.

To draw the outline boxes press E to *Edit Screen*. You can move around the screen using the 'diamond' of control keys that is common to many programs. The ALT key, in combination with E, X, S and D moves the cursor up, down, left and right, respectively.



```
CARDBOX                                                     READY
PRIMARY FUNCTIONS:                    | SECONDARY FUNCTIONS
                                      |
        Database                      |      Edit
  ==>   Format definition             | ==>  Create
        Operating system utilities    |
                                      |
(-------------------------------------|--------------------------------)

PRIMARY-FUNCTION = [FORMAT]
SECONDARY-FUNCTION = [CREATE]

FILE =*
CHANGE-DISKS = [NO]


FILE = SUBS.........
(At end, hit RETURN to enter, or ESC to cancel entry)



                                                        Drive is A:
```

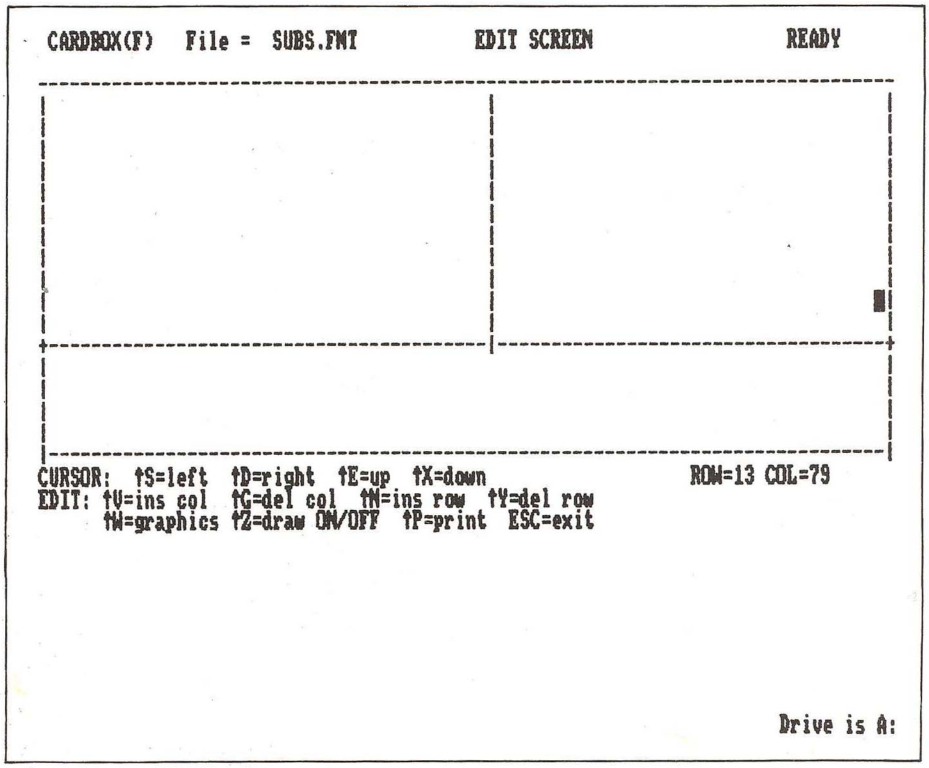
Figure 6.2 *Creating a new file*

These may seem somewhat unusual choices of commands at first sight but the diamond is used by many programs. The style was popularised by the WordStar word-processing program. From a point in the middle of the diamond the keys move the cursor in a direction that corresponds to the position of the key in the diamond.

Pressing ALT-Z allows you to 'draw' on the screen, any character selected being repeated at the cursor positions. ALT-Z again turns the drawing mode off.

In a similar way ALT-W switches on or off a line-drawing mode. When it is on, any cursor movement results in lines being drawn, either horizontally or vertically, and with any corners of boxes and crossing of lines automatically taken care of.

With ALT-Z on and ALT-W off, you can delete lines or text by typing spaces over the top of them. Explanatory text can also be added when ALT-



```
CARDBOX(F)   File =  SUBS.FMT          EDIT SCREEN                READY

CURSOR:  ↑S=left  ↑D=right  ↑E=up  ↑X=down                ROW=13 COL=79
EDIT: ↑V=ins col  ↑G=del col  ↑N=ins row  ↑Y=del row
      ↑W=graphics ↑Z=draw ON/OFF  ↑P=print  ESC=exit

                                                               Drive is A:
```

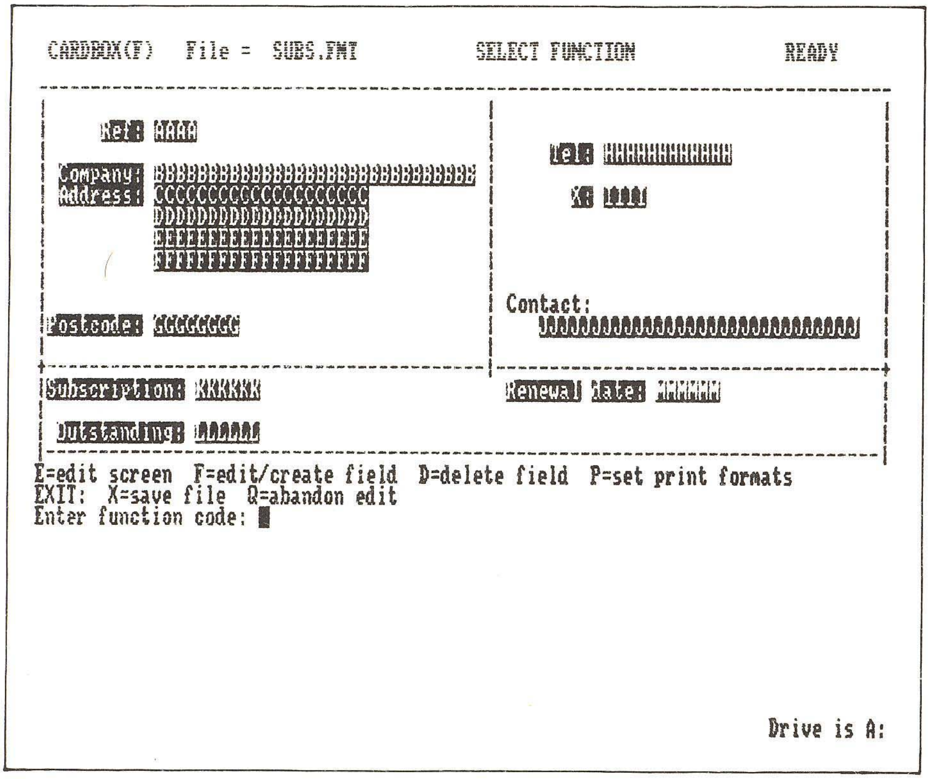
Figure 6.3 *Drawing the outline of a card*

Z and ALT-W are off. Press EXIT to return to the menu of single-letter commands.

Defining fields

The F command lets you add a field. Up to 26 fields are allowed for any one record and each of these can be of considerable size. The maximum total number of characters in a record is 1404. However, Cardbox differs from Matchbox in that it uses as many characters as are needed for each record, so there is nothing to be gained by allocating spare space as a precaution. The record structure can be changed at any time, and new fields can be added or deleted as necessary.

For each field you must specify a single letter to identify it on the screen layout (hence the maximum of 26 per file), along with a two-letter combination to be used for identification in commands. The single-letter identifiers should be in alphabetical order; this determines the order in which



```
CARDBOX(F)   File = SUBS.FMT          SELECT FUNCTION          READY
--------------------------------------------------------------------------------
|     Ref: AAAA                          |                                       |
|                                        |     Tel: HHHHHHHHHHHHH                |
| Company: BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB|                                      |
| Address: CCCCCCCCCCCCCCCCCCCC          |        X: IIII                        |
|          DDDDDDDDDDDDDDDDDDDD          |                                       |
|          EEEEEEEEEEEEEEEEEEEE          |                                       |
|          FFFFFFFFFFFFFFFFFFFF          |                                       |
|                                        | Contact:                              |
|Postcode: GGGGGGGG                      |   JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ  |
+----------------------------------------|---------------------------------------+
|Subscription: KKKKKK                      Renewal date: MMMMMM                  |
|                                                                                |
| Outstanding: LLLLLL                                                            |
|--------------------------------------------------------------------------------|
E=edit screen  F=edit/create field  D=delete field  P=set print formats
EXIT:  X=save file  Q=abandon edit
Enter function code: █




                                                                   Drive is A:
```

Figure 6.4 *Adding fields*

data is to be entered. The position of the field on the screen is determined by moving the cursor to the start of the field and pressing S, then moving it to the end of the field and pressing E. The field does not have to consist of a single line. The points at which you press S and E identify the top left-hand and bottom right-hand corners of a rectangle for entering data. Therefore a text field may stretch over several lines. The field is marked out with the identifying letter, except for the first and last characters which are reserved for Cardbox's own use; remember to take account of this when setting the length of the field.

Cardbox does not distinguish between different types of field. They are all treated as text. Any lining up of numeric values must be done manually by inserting spaces in front of the values.

For each field you can place a caption on the screen by pressing C and entering some suitable text. To change the length, move to a new position and press either S or E.

Finally you must choose the *index mode* for the field. During any search operation the program will only look through a separate index file, which consists of words in the data file that have been specifically marked as being part of the index. For each field you have to decide whether the data is likely to be used in a search or not, and set the index mode as ALL or NONE, respectively. Any data in a field with the NONE mode is excluded from the index file and cannot be located in a search. For the ALL mode any word entered in the field is included in the index. However, this takes up additional space on the disk and for lengthy records may be unnecessarily wasteful of valuable disk space. Therefore there are two intermediate modes, MAN (manual) and AUTO, in which individual words will only be included in or excluded from the index. Each word must be specifically marked for inclusion (MAN) or exclusion (AUTO) by pressing TAB when the word is entered. For example, lengthy items of text may be in MANual mode, with occasional words marked; addresses may be in AUTO mode, since you would not wish to include "Road" or "Street" but almost anything else may be searched for. To select the mode keep pressing I until the correct type is shown. Press EXIT and F to move on to the next field.

For the SUBS file described in the previous chapter the fields would be:

Identifier	*Command name*	*Caption*	*Size*	*Index mode*
A	RE	Ref:	4	ALL
B	CO	Company:	30	ALL
C	A1	Address:	20	MAN
D	A2		20	MAN
E	A3		20	MAN
F	A4		20	AUTO
G	PO	Postcode:	8	MAN
H	TE	Tel:	12	NONE
I	EX	X	4	NONE
J	CN	Contact:	30	MAN
K	SU	Subscription:	6	ALL
L	AO	Outstanding:	6	ALL
M	RD	Renewal date:	8	ALL

Any field can have some of its attributes changed or be moved on the layout by pressing F again and re-entering the field identifier. A field is deleted with the D option.

To leave this part of the program and save the format on disk, press X. At any stage you can return to make further changes by selecting Format Definition and Edit from the main menu.

Entering data

To enter data for the first time select Database and Create from the main menu. Press F and enter the drive specifier and filename (to match the format filename), then EXIT and G. The layout previously designed is shown, with blanks left where the field values are to be entered. A list of possible two-letter commands is shown at the bottom of the screen. To choose any command type the letters and press RETURN.

To add records, choose the AD (Add) command. The record can be entered in the same way as for Matchbox, but with some editing facilities available. There are a variety of options, such as ALT-A to move back through the field one word at a time. Wherever the cursor is, any new characters replace existing ones; to insert characters press ALT-V. Characters are deleted to left or right with the DEL keys. RETURN takes you through each field in turn.

To complete the record press EXIT, followed by S to store it on disk. E (Edit) makes further changes while Q (Quit) scraps the record.

If two records are similar you can start by giving the DU (Duplicate) command. The last record is repeated and can be edited, which saves typing in all the data from scratch.

When entering data the indexed words are highlighted; remember to press TAB at the start of any word that is to be included or excluded when in MAN or AUTO mode, respectively.

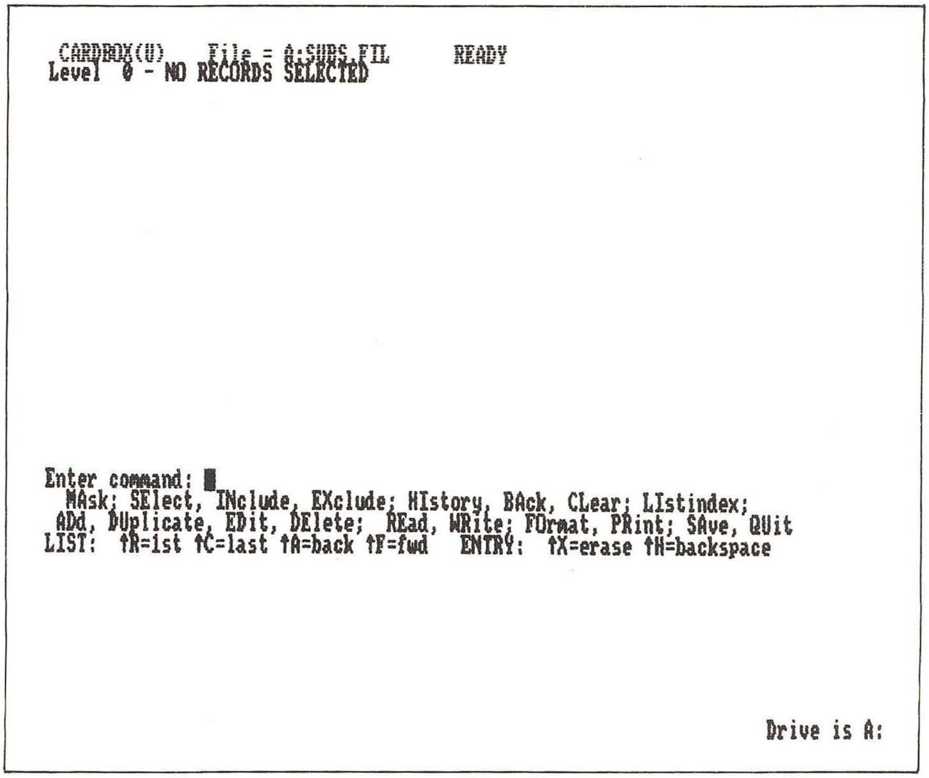
The QU (Quit) command takes you back to the main menu.

Inspecting records

From the main menu choose Database and Use and supply the filename. The most recent record is displayed and a range of commands is available (including those to Add or Duplicate a record). To move through the file, four options are available:

- ALT-A to move backwards through the file.
- ALT-F to move forwards through the file.
- ALT-R to move to the first record.
- ALT-C to move to the last record.

The keys are again based on the WordStar ‘diamond’; the key to the right moves you forwards and that to the left backwards, while the keys above and below take you to the top (or beginning) and to the end of the file,



```
  CARDBOX(U)    File = A:SUBS.FIL       READY
Level  0 - NO RECORDS SELECTED









Enter command: █
  MAsk; SElect, INclude, EXclude; HIstory, BAck, CLear; LIstindex;
 ADd, DUplicate, EDit, DElete;  REad, WRite; FOrmat, PRint; SAve, QUit
LIST:  ↑R=1st ↑C=last ↑A=back ↑F=fwd   ENTRY:  ↑X=erase ↑H=backspace

                                                            Drive is A:
```

Figure 6.5 *The Cardbox commands*

respectively. The use of these keys is rather more obvious in a file of text but even so, the principles, once accepted, are the same.

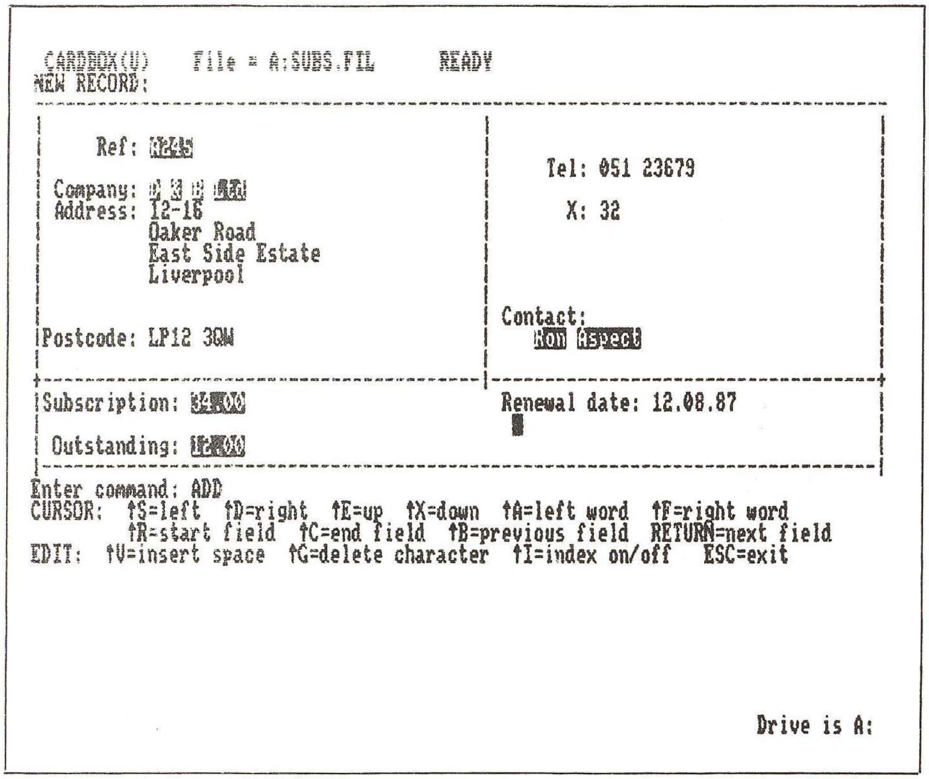
Editing records

Editing is carried out with the ED command and offers the same opportunities for changes as Duplicate. The record currently displayed is the one to be edited and so there is no need to enter a record number. The only difference between ED and AD is that on pressing EXIT the Q option returns the record to its original state, as it was before editing, rather than abandoning it altogether.

The current record can be deleted with the DE (Delete) command.

Searching, selection and sorting

You can select any group of records by entering a range of commands.



```
 CARDBOX(U)    File = A:SUBS.FIL      READY
NEW RECORD:
-------------------------------------------------------------------------
|                                        |                              |
|     Ref: 1245                          |                              |
|                                        |      Tel: 051 23679          |
| Company: [illegible]                   |                              |
| Address: 12-16                         |        X: 32                 |
|          Oaker Road                    |                              |
|          East Side Estate              |                              |
|          Liverpool                     |                              |
|                                        |                              |
|                                        | Contact:                     |
|Postcode: LP12 3QW                      |    Ron Aspect                |
|                                        |                              |
+----------------------------------------|------------------------------+
|Subscription: 54.00                     | Renewal date: 12.08.87       |
|                                        | █                            |
| Outstanding: 12.00                     |                              |
|-----------------------------------------------------------------------|
Enter command: ADD
CURSOR:  ↑S=left  ↑D=right  ↑E=up  ↑X=down  ↑A=left word  ↑F=right word
         ↑R=start field  ↑C=end field  ↑B=previous field  RETURN=next field
EDIT:  ↑V=insert space  ↑G=delete character  ↑I=index on/off   ESC=exit



                                                            Drive is A:
```

Figure 6.6 *Adding a record*

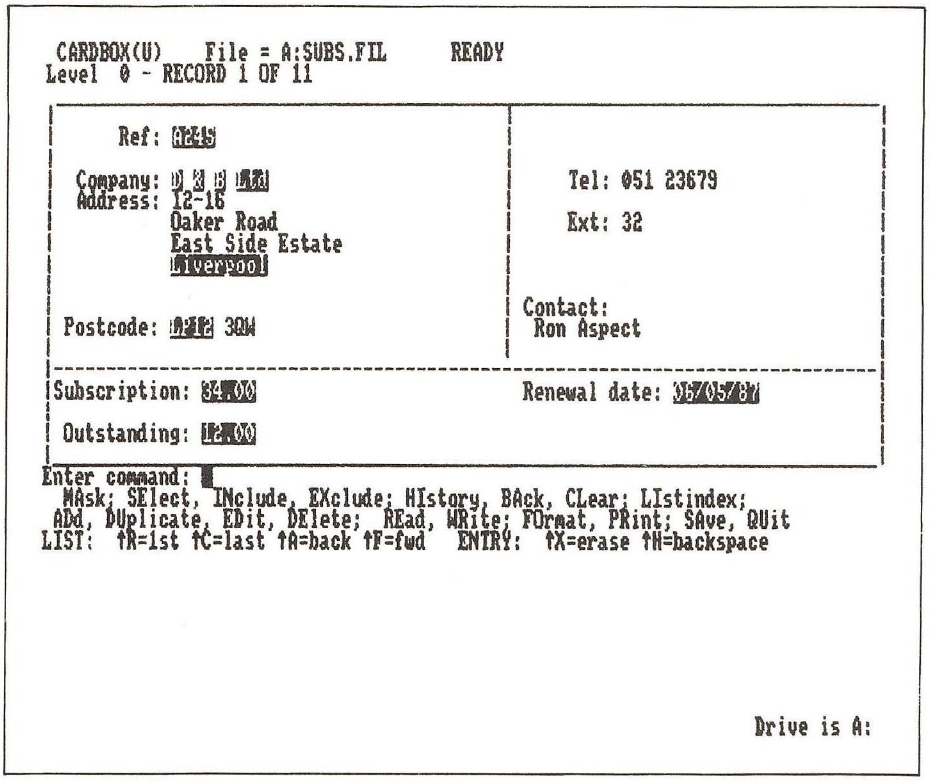
Unlike Matchbox you are not limited to a single field or a single criterion for selection. Any record not matching the criterion is automatically suspended from the file until specifically reinstated.

Reducing the file size

The main commands are SE (Select) and EX (Exclude), which select or exclude records, respectively, depending on whether or not they match the criterion given. Each command is followed by an oblique stroke (/) and the item of text to be searched for. Cardbox is not case-sensitive; capitals and lower case are treated as being the same. The search is carried out on all indexed words unless the field is specified after the command (using the two-letter abbreviation). All words, rather than just those that are indexed, are searched for if the command is preceded by the MA (Mask) command.

For example, to search for all addresses in Liverpool use the command:

```
SE /LIVERPOOL
```



```
 CARDBOX(U)    File = A:SUBS.FIL      READY
Level  0 - RECORD 1 OF 11

|--------------------------------------------|-------------------------------|
|     Ref: A245                              |                               |
|                                            |                               |
|  Company: [illegible]                      |    Tel: 051 23679             |
|  Address: 12-16                            |                               |
|           Oaker Road                       |    Ext: 32                    |
|           East Side Estate                 |                               |
|           Liverpool                        |                               |
|                                            |                               |
|                                            | Contact:                      |
| Postcode: L12 3QW                          |  Ron Aspect                   |
|                                            |                               |
|-----------------------------------------------------------------------------|
|Subscription: 34.00                          Renewal date: 06/05/87         |
|                                                                             |
| Outstanding: 12.00                                                          |
|-----------------------------------------------------------------------------|
Enter command: █
  MAsk; SElect, INclude, EXclude; HIstory, BAck, CLear; LIstindex;
 ADd, DUplicate, EDit, DElete;  REad, WRite; FOrmat, PRint; SAve, QUit
LIST:  ↑R=1st ↑C=last ↑A=back ↑F=fwd   ENTRY:  ↑X=erase ↑H=backspace




                                                            Drive is A:
```

Figure 6.7 *Inspecting a record*

This overcomes the problem of the town having to be in the last line of the address, but means that occurrences of the value in other fields will also show up. To find all addresses outside Liverpool the command is:

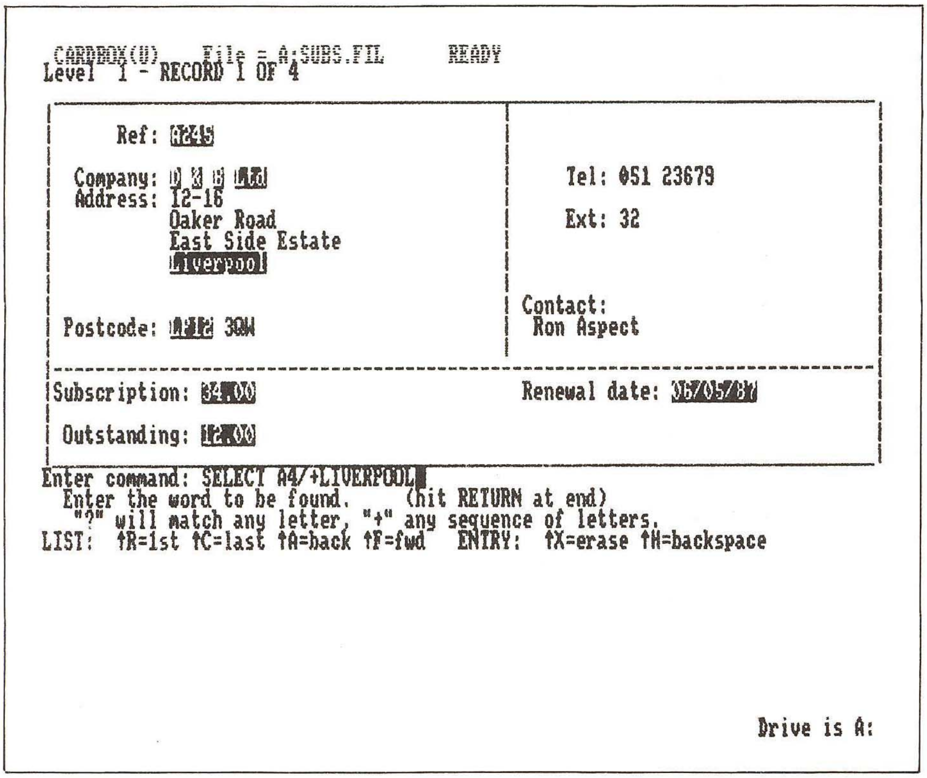
EX /LIVERPOOL

If the Address field has an index mode of NONE the command should be:

MA EX /LIVERPOOL

The disadvantage of the Mask option is that the search is very much slower. The advantage is that you can search for a whole phrase rather than a word.

Two *wildcards* can be used, the ? wildcard being used to replace any character in the word and the + wildcard replacing any group of characters. The difference between the Cardbox + wildcard and the CP/M * wildcard is that the + can come anywhere in the word, not just at the end. You may only use one ? wildcard in any search string.



```
 CARDBOX(U)    File = A:SUBS.FIL      READY
Level  1 - RECORD 1 OF 4
 -----------------------------------------------------------------------
|     Ref: A245                          |                             |
|                                        |                             |
|  Company: D M B Ltd                    |    Tel: 051 23679           |
|  Address: 12-16                        |                             |
|           Oaker Road                   |    Ext: 32                  |
|           East Side Estate             |                             |
|           Liverpool                    |                             |
|                                        |                             |
|                                        | Contact:                    |
| Postcode: L12 3QW                      |  Ron Aspect                 |
|-----------------------------------------------------------------------|
|Subscription: 34.00                       Renewal date: 06/05/87      |
|                                                                       |
| Outstanding: 12.00                                                    |
 -----------------------------------------------------------------------
Enter command: SELECT A4/+LIVERPOOL
  Enter the word to be found.     (hit RETURN at end)
   "?" will match any letter, "+" any sequence of letters.
LIST:  ↑R=1st ↑C=last ↑A=back ↑F=fwd   ENTRY:  ↑X=erase ↑H=backspace



                                                           Drive is A:
```

Figure 6.8 *Selecting a record*

Therefore to search for all addresses in Liverpool you can use the postcode (PO) as follows:

```
SE PO/LP+
```

To find all subscriptions due in November the commands could be:

```
SE RD/+11+
EX RD/11+
```

Note that you cannot use symbols such as / in the search string.

The second command is necessary to exclude dates such as 11/03/87. (The + wildcard can match no characters.) The ? wildcard must always match a single character.

As each new condition is entered, the number of records selected varies, as shown at the top of the screen, and the number of selections (or 'levels') increases. Entering another command is the equivalent of the AND logical operator described in Chapter 2.

Adding records to the selection

There is also an equivalent to the OR operator, which is the IN (Include) command. This is also followed by a search string and any record that matches is added to those already in the selection file.

For example, to find all subscriptions from Liverpool or Manchester that do not fall due in 1988 enter the commands:

```
SE /Liver+
IN /Manch+
EX RD/+88
```

The HI (History) command displays the last 18 selection commands on the screen. (The most you can have at any one time is 99.) BA (Back) undoes the effect of the last command while CL (Clear) reinstates all records so that you can start again from scratch.

The main drawback of the Cardbox approach is that there is no easy way to deal with numerical values. For example to find all amounts outstanding in the range £20 to £49 would require these commands:

```
SE AO/2+
SE AO/3+
SE AO/4+
```

Unfortunately this also includes amounts in the range £2 to £4.

There is no sorting facility in Cardbox.

Reports

The reporting facilities of Cardbox are much more sophisticated than those of Matchbox, and quite straightforward to set up. The print layout is designed at the same time as the screen layout. If you want to print with the layout that appears on the screen, for example to print a selection of cards, then choose the Format Definition and Edit options. When the original display reappears, press P to *Set Print Formats*. Similar details to Matchbox are required:

P	Page size - number of lines on page
N	Next page - determines how top of next sheet is selected
T	Top margin - blank lines at top of page

```
 -----------------------------------------------------------------------------
|                                          |                                  |
|     Ref; A245                            |                                  |
|                                          |                                  |
| Company; D & B Ltd                       |    Tel; 051 23679                |
| Address; 12-16                           |                                  |
|          Oaker Road                      |    Ext; 32                       |
|          East Side Estate                |                                  |
|          Liverpool                       |                                  |
|                                          |                                  |
|                                          | Contact;                         |
|Postcode; LP12 3QW                        |  Ron Aspect                      |
|                                          |                                  |
|-----------------------------------------------------------------------------|
|Subscription; 34,00                         Renewal date; 06/05/87           |
|                                                                             |
| Outstanding; 12,00                                                          |
|-----------------------------------------------------------------------------|
```

Figure 6.9 *Printing a card*

L	Left margin - blank columns to left of records
E	Entries per page - number of complete records on page
B	Blank lines - blanks between records

Remember that the number of lines needed for each record may be quite large, for example 20 if the whole screen is used in the display. On continous paper (66 lines) this allows for three records per page with 1 blank line between each, 2 blanks at the top and 1 at the bottom. The program warns you if the figures you supply don't add up. The N value determines whether or not the printer will perform a form-feed operation, taking the paper to the top of the next sheet. The alternative is for the paper to be advanced until the number of lines shown in the page size has been reached. By varying this number you can achieve some useful effects.

Of course, this layout will probably be unsuitable for most printouts. Therefore you can design a completely new format, for example one called LABELS.FMT for printing address labels or LIST.FMT to print the records in a list format. Enter all the details as before and space the fields and text as you want them printed. For instance, a label format would have all the details in a block at the top of the screen; to produce a list of records just put the fields you want to print in a single row on the top line. The page size and other details should also be set accordingly.

To produce the printout, load the data file with Database and Use. The first record will appear on the screen in the standard layout. Exclude those records that are not to be printed. Then use the FO command to load the new format. Press F and give the filename; press EXIT G to load the format. The screen display changes accordingly.

The command to Print is PR. Selecting S gives a choice between printing from the beginning of the file or from the record that is displayed. Press M to select the Mode, in this case the paper type. Finally, EXIT G starts the printing. This can be interrupted at any time by pressing STOP and then Q (though the printer may continue for some time while it finishes printing the data it holds in its own internal memory).

The printer must have been set up in CP/M before starting Cardbox if you are to use other than the Amstrad printer.

With a little care, a print format can be designed to overprint the gaps on a standard letter prepared by a word processor. (You could write the whole letter with Cardbox, but this would be extremely tedious.) When writing the letter you must be sure to leave gaps of adequate size and try to put any fields whose values are variable in length at the end of a line; otherwise you will have unsightly gaps between words. Note that you can have as many formats as you like for one data file; the same format can be used for more than one file if the files share common field names and attributes.

Looking after files

The *Operating System Utilities* option in the main menu has two items in the sub-menu: *Copy File* and *Erase File*. The Copy File option should be used frequently to make a backup of your data files (and complicated format files as well). You can copy onto another disk, even if you have a single-drive machine.

Whenever you edit a file the previous version is always automatically saved, as a precaution, with a .BAK extension.

Erase File is a rather curious option, since it will not allow you to delete files with either a .FIL or .FMT extension. Intended to avoid reckless destruction of important data files, no doubt, but such mistakes are far more likely to be made using the CP/M ERASE command with wildcards which is the only alternative.

Any renaming or disk-copying functions must be carried out when back in CP/M Plus.

Damaged files

Cardbox has an unusual facility (Database and Repair) that allows you to try and recover data from corrupt files. The procedure is rather complex but nevertheless can be extremely welcome if a large file of data is about to be lost. Of course, it should never be necessary if the much simpler backup procedures have been followed.

Export and import

The process of sending data to another program is generally referred to as *exporting*, while the reverse is called *importing*. Cardbox can achieve both of these. With the WR (Write) command a selection of records can be stored in a separate file. These can be in an internal format, ready to be imported into another Cardbox file with the RE (Read) command. Therefore any selection of records can be transported between files, a very useful facility.

Alternatively, files can be stored in an *ASCII* format - a simple data format understood by most programs - with an option for the indexed words to be marked.

Finally there is a format that can be accepted by WordStar and compatible programs. In this way the data can be transferred to a word processor to form part of a document and may be further edited there.

Support

The manual provided with Cardbox covers all that you need to know, but is short on hints as to how to get the best out of the system. Each short section is immediately followed by a reference section that covers more or less the same ground. However, the manual is easy to follow, though a little confusing if you don't want the bother of working through the example file that is supplied; the instructions on how to set up a new database don't appear until two-thirds of the way through the manual.

Caxton Software will replace a faulty disk and offer unlimited telephone support. Once again, though, the program is quite straightforward to use once you're familiar with it and extra help shouldn't really be necessary.

Conclusion

Matchbox and Cardbox deal with the same sort of data in a similar way. However, Cardbox has the additional facility to carry out searches at a number of levels and to print only the selected records. The layout of the report is fully within the control of the user. Therefore if you want to do more than just print lists or labels and require a little sophistication in your selections, Cardbox will be suitable.

Despite the lack of a sort facility, Cardbox is well suited to many applications.

CHAPTER 7

ATLAST 1

The third program follows the card index style previously described, with some modifications. AtLast 1 from Rational Solutions introduces the concept of sorting, allowing your records to be printed or displayed in a variety of arrangements.

Initial preparations

The usual procedures of copying the program disk and making and preparing data files should be carried out. AtLast is comprised of two separate programs, one for defining files (DBDEF) and the other for entering and manipulating data (DBUSE). The files total 147K and therefore there is insufficient room for all of them on a system disk. Probably the best approach is to put the DBDEF files on one side of the system disk and DBUSE on the other side.

Before running the program on a single-drive machine the relevant files should be copied to the memory drive and the program can be executed from there.

Before defining a new file the following instructions are needed:

```
PIP M:=DBDEF.*
SETKEYS ATLAST.KEY
```

You need to have the CP/M utilities and SETKEYS on both sides of the work disk.

You should also include SUBMIT.COM and PAPER.COM, as well as ATLAST.SUB, on the DBUSE side of the disk. (A full directory listing is given in Appendix 2.)

Creating a data file

The setting up of a data file is, to say the least, complicated. It is necessary to have a full understanding of the concepts behind the AtLast approach to databases before you create any files. Once you've struggled through the jungle of jargon and file structures, the actual entry of data and its manipulation is fairly easy.

AtLast is a menu-based program. This limits its facilities, but does mean that there are no complicated commands to remember. The package consists of two separate programs, DBDEF and DBUSE, for defining and using a database, respectively.

To create a database, load the operating system, put in your work disk and (for a single-drive machine) copy the program files to drive M. Then at the A> or M> prompt enter the command:

```
DBDEF
```

You are prompted for a filename, up to eight characters long, as before. If it is to be placed on another drive (which will normally be the case) precede it with the drive specifier (for example, A: or B:). On a single-drive machine replace the program disk with the data disk before going any further.

Note that the name of the data file can be specified in the initial command:

```
DBDEF A:SUBS
```
    (single-drive, run from drive M)
```
DBDEF B:SUBS
```
    (double-drive, run from drive A)

The file is given a .DEF extension.

Defining a file

The program takes you straight into the file definition stage. AtLast works with groups of files in separate databases, rather than have a single large collection of independent files. The reason for this is that you are able to link together two files in a database, a process described later. The filename you have given relates to the database as a whole.

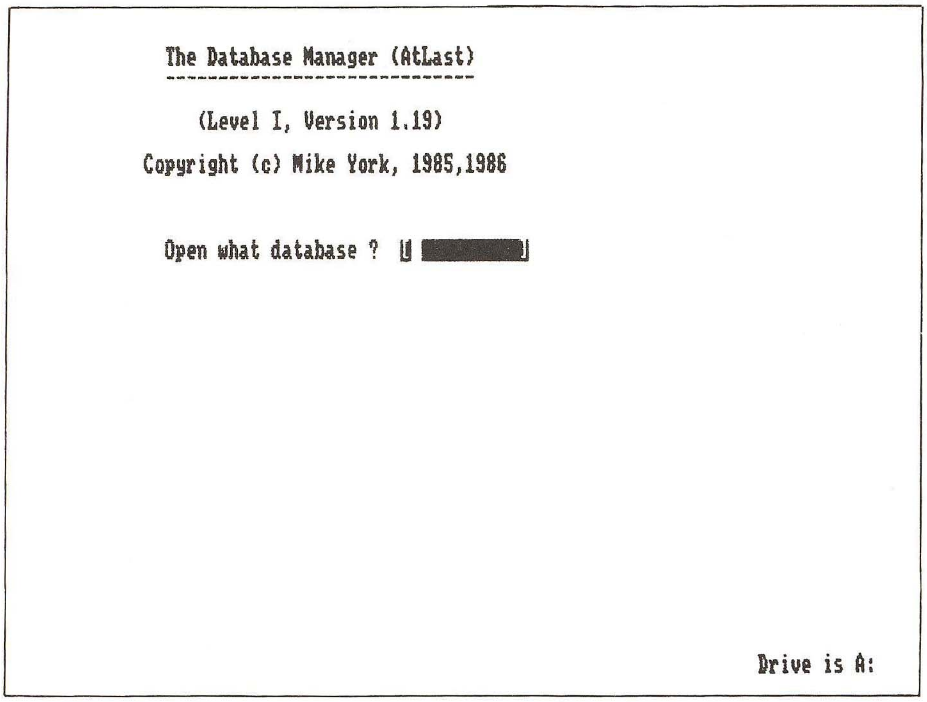

```
                The Database Manager (AtLast)
                -----------------------------

                   (Level I, Version 1.19)

              Copyright (c) Mike York, 1985,1986


                Open what database ?



                                                           Drive is A:
```

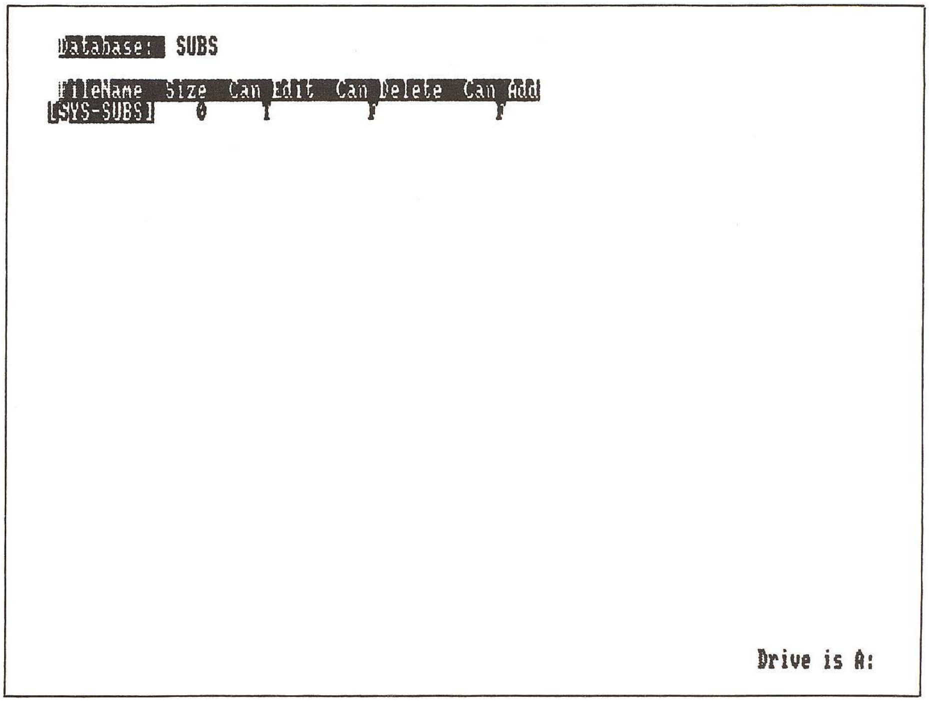
Figure 7.1 *The introductory display*

The first stage is to add the names of the files to be used in this particular database. Start by entering a single name and then build up the database by adding extra files when needed. The first file shown is used to store general information about the database. Its name begins with 'SYS' (for example, SYS-SUBS) and it contains just one record. For example, you may use it to store subscription rates, start of the financial year or total number of customers. Initially it is probably better to ignore this file. Press ALT-X to move down and enter a name for the main data file (such as SUBSDATA).

The data itself is held in a .DAT file, e.g. SUBSDATA.DAT.

For each file you must specify several items of information: the name of the file, details of the fields (see below) and whether or not individual records can be edited, deleted or added. 'TRUE' indicates that changes, deletions or additions are allowed, 'FALSE' that they are not. Generally the answer will be TRUE in each case but there will be some circumstances where you may wish to ensure that records cannot be changed in any way.

On pressing EXIT you can further edit the list, save the changes on disk or print the file structure. Press EXIT again to return to the main menu.



```
Database:  SUBS

FileName  Size  Can Edit  Can Delete  Can Add
[SYS-SUBS]   0     T          F           F

                                              Drive is A:
```

Figure 7.2 *Entering filenames*

Defining fields

After entering each filename you must define the fields. Only 20 fields are allowed, which can be rather limiting. Even though AtLast has the unusual feature that more than one value can be specified for each field (for example, the Address field may have four *elements*, one for each line) the nature of most databases is that each item tends to be different. To make best use of this feature each record would need to contain lists of information. For example, the SUBS file could be used to record all previous subscriptions, rather than just the most recent one.

You must supply a field name (used for identification, a maximum of eight characters long) and the data type. AtLast excells itself in the range of field types available. These include:

Alpha	Text fields
Upper	Text fields with all letters converted to upper case
Integer	Whole numbers (positive or negative)
Fixed	Decimals, with a fixed number of decimal places after the decimal point
Real	Decimals with no fixed number of decimal places

Date	Dates in the form day/month/year (the year may be two or four digits)
HMS	Times in the form hours:minutes:seconds

Two other field types can be included for calculations by the program:

Constant	A field built up from the contents of other fields
Serial	A field that is automatically increased by 1 for each new record

For each field you must specify the number of elements, i.e. the number of values allowed, and the length of each element.

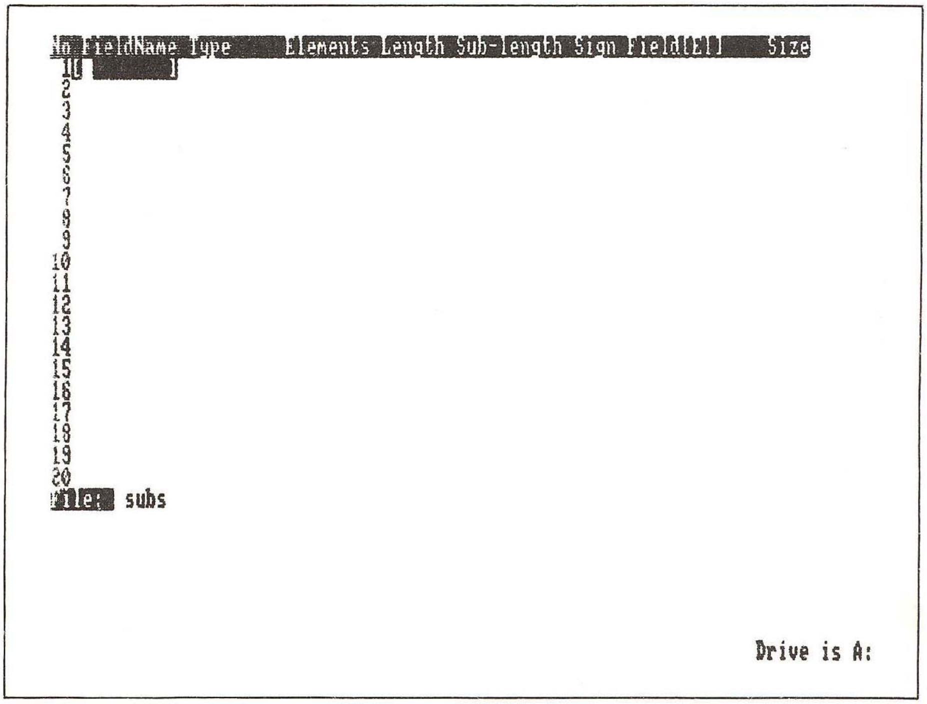
For the SUBS example described in previous chapters the record structure can be as follows:

Name	*Type*	*No. of elements*	*Length*	*No. D.P.*	*Signed*
Refernce	A	1	5		
Company	A	1	30		
Address	A	4	20		
Postcode	A	1	8		
Tele	A	1	10		
Ext	A	1	4		
Contact	A	1	30		
Subscrip	F	10	6	2	FALSE
Ren-date	D	10	8		
Outstand	F	1	6	2	FALSE
Date-due	D	1	8		

On pressing EXIT you are taken to the next stage (to define key fields).

Note that the space allocated on disk for each file is determined by the total field sizes. Therefore once any data has been entered you cannot extend the file size and changing any of the field types or sizes can lead to serious problems. It is as well to leave any file unchanged once data has been added. Some problems can be avoided by including spare fields (as for Matchbox); as an alternative all data can be transferred to a new file at a later stage if it is discovered that the present structure is adequate.

Key fields When sorting your records the program uses special *key* fields. These are fields whose contents are held in separate *index* files (with a .FIX extension) that are constructed from the contents of the normal fields in the file. Each key field can be made up of the contents of up to three other fields; any sort routine is carried out by ordering records on the first part and then, for those records which have the same value, on the second part. Finally, if the key is still the same for two or more records, they are ordered according



```
No FieldName Type       Elements Length Sub-length Sign Field[El]     Size
 1[         ]
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
File: subs

                                                          Drive is A:
```

Figure 7.3 *Defining the file*

to the third part. For each field used you must specify how much of its value will be used for the sort.

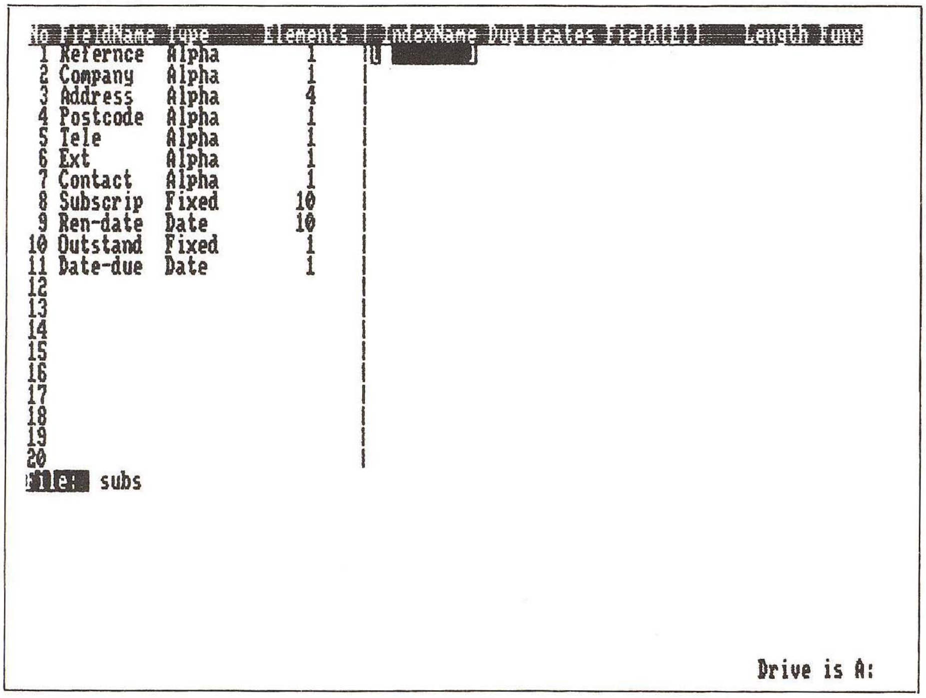
For Alpha fields you must also supply a *function*, which takes the values:

0: Value is as shown
1: All letters are converted to upper case for sorting
2: All numbers and symbols are removed

Up to five key fields can be defined for any file. In each case you must specify whether or not these may duplicate another key. Having to define the keys at this stage can be a drawback, since it means that you have to decide what type of sorting procedures you are likely to need. However, you can add new key fields at a later stage by running the database definition program again (as long as you have not used up your five keys).

Some of the keys that may be useful for the SUBS file are:

Index name:	Overdue
1st field:	Date-due, length 5
2nd field:	Company, first 10 characters, function 1



```
No FieldName Type      Elements | IndexName Duplicates Field[El]     Length Func
 1 Refernce  Alpha       1      |[          ]
 2 Company   Alpha       1      |
 3 Address   Alpha       4      |
 4 Postcode  Alpha       1      |
 5 Tele      Alpha       1      |
 6 Ext       Alpha       1      |
 7 Contact   Alpha       1      |
 8 Subscrip  Fixed      10      |
 9 Ren-date  Date       10      |
10 Outstand  Fixed       1      |
11 Date-due  Date        1      |
12                              |
13                              |
14                              |
15                              |
16                              |
17                              |
18                              |
19                              |
20                              |
File: subs




                                                              Drive is A:
```

Figure 7.4 *Adding index names*

Index name:	Area
1st field:	Postcode, first two characters only, function 1
2nd field:	Date-due, length 5
3rd field:	Company, first 10 characters, function 1

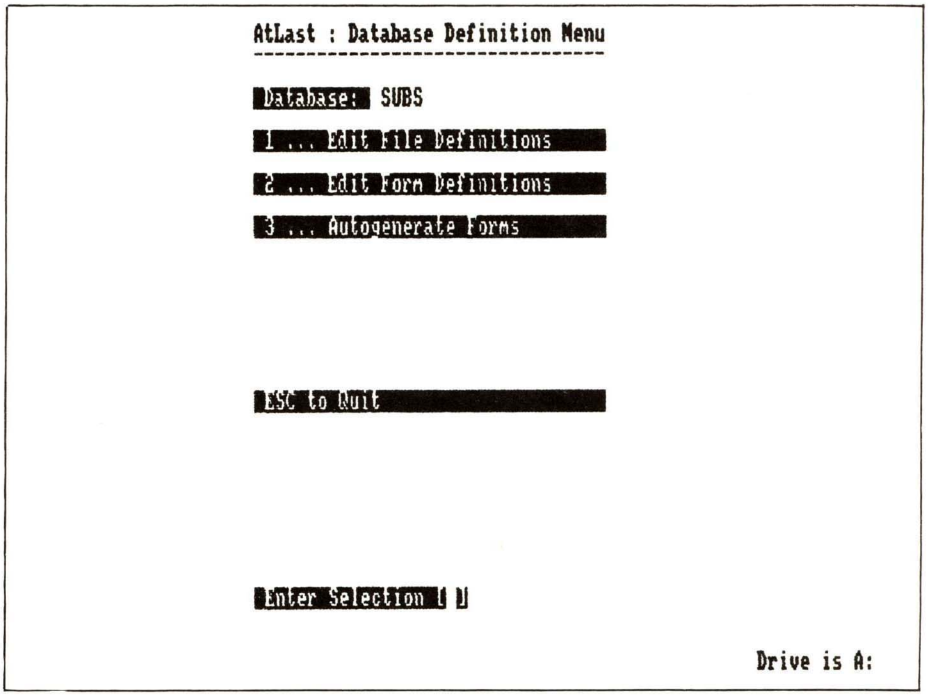
In the first case records are listed by the date on which the subscription is due, with the records for each date being in alphabetical order of company name. In the second case the records are first sorted into areas according to postcode.

After entering the fields for a key press EXIT and then ALT-X to move down to the next key. Each time a record is added or changed the key fields are calculated and automatically stored in the index file in the correct position.

The index file provides a pointer to the records in the main file, so that lists of sorted records can be produced in a very short time.

Designing a format

The second option from the main menu allows you to design forms for



```
AtLast : Database Definition Menu
---------------------------------

Database: SUBS

1 ... Edit File Definitions

2 ... Edit Form Definitions

3 ... Autogenerate Forms




ESC to Quit




Enter Selection [ ]

                                      Drive is A:
```

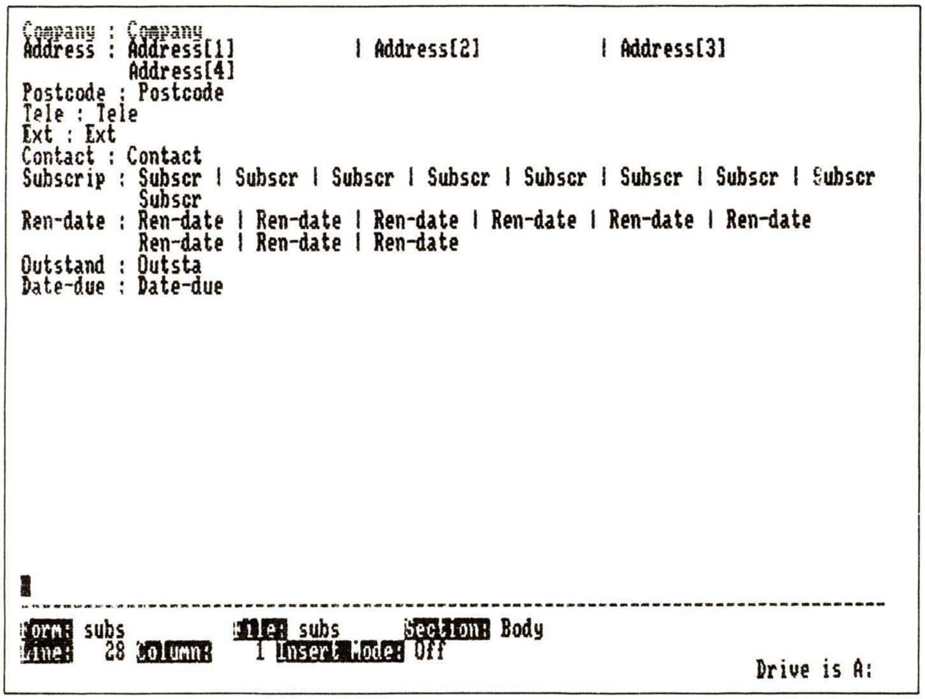
Figure 7.5 *The Database Definition menu*

display on the screen and printing on paper. The third option automatically generates simple formats.

You can either create a new format from scratch or edit an existing format. If creating a format you must specify the file with which it is to be associated. You cannot use the same format for more than one file though you can make a copy of it and give it a new name. The format name can be up to 12 characters long.

Each format is in three parts: a 'head' that provides title text, headings for columns and so on but may not contain field values; a 'body' that contains the bulk of the data, repeated for each record; a 'tail' to print further text at the foot of the printout and totals of numeric fields. If a format is to be used for a screen display then only the body is used.

Select the option to create a format, supply a name of up to 12 characters and specify which data file it is for. The program shows a blank screen onto which you can place the body of the format. Press E from the menu to edit the selected part of the format. The cursor can be moved around the screen and text entered at any position.



```
Company : Company
Address : Address[1]          | Address[2]          | Address[3]
          Address[4]
Postcode : Postcode
Tele : Tele
Ext : Ext
Contact : Contact
Subscrip : Subscr | Subscr | Subscr | Subscr | Subscr | Subscr | Subscr | Subscr
           Subscr
Ren-date : Ren-date | Ren-date | Ren-date | Ren-date | Ren-date | Ren-date
           Ren-date | Ren-date | Ren-date
Outstand : Outsta
Date-due : Date-due

-------------------------------------------------------------------------------
Form: subs          File: subs       Section: Body
Line:   28 Column:    1 Insert Mode: Off
                                                                  Drive is A:
```

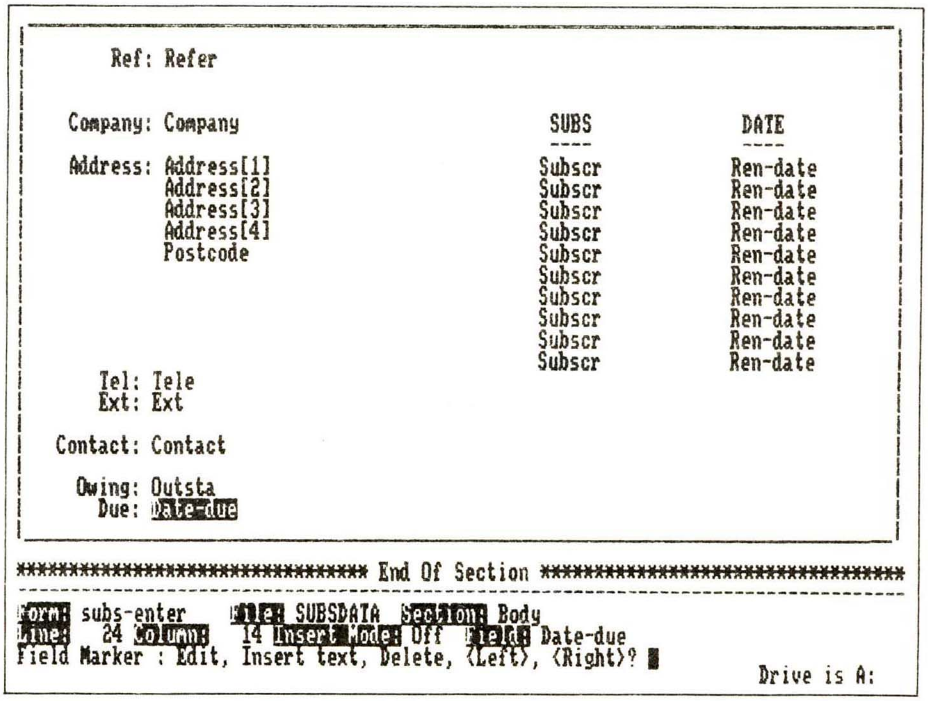
Figure 7.6 *An auto-generated form*

To place a field on the format, press PASTE. Press E to edit the field and you are prompted to enter the field name and, where appropriate, the element number. Finally, you must specify whether leading and trailing spaces should be removed when the format is printed. If you select the TRUE option then any following text is moved along to fill the gaps. This means that the length of the line will vary. Therefore this option is useful for names and addresses placed on a single line, but is not suitable for columnar lists.

To delete a field marker, press D when the cursor is in the field; to leave the field, press L or R (to move to left or right).

By moving the cursor down below the last line of the screen the format can be made larger than the screen display. To extend a printed format to allow more than 80 characters per line you must end each part-line with the backslash (\) character, achieved by pressing EXTRA and ½ simultaneously.

Press EXIT when the format is complete; a variety of options are available for editing the three parts of the format.



```
        Ref: Refer


    Company: Company                                   SUBS             DATE
                                                       ----             ----
    Address: Address[1]                               Subscr           Ren-date
             Address[2]                               Subscr           Ren-date
             Address[3]                               Subscr           Ren-date
             Address[4]                               Subscr           Ren-date
             Postcode                                 Subscr           Ren-date
                                                      Subscr           Ren-date
                                                      Subscr           Ren-date
                                                      Subscr           Ren-date
                                                      Subscr           Ren-date
                                                      Subscr           Ren-date
        Tel: Tele
        Ext: Ext

    Contact: Contact

      Owing: Outsta
        Due: Date-due

******************************** End Of Section *********************************
---------------------------------------------------------------------------------
Form: subs-enter    File: SUBSDATA  Section: Body
Line:   24 Column:    14 Insert Mode: Off   Field: Date-due
Field Marker : Edit, Insert text, Delete, <Left>, <Right>? █
                                                                 Drive is A:
```

Figure 7.7 *Defining a form*

When the formats for the database are complete you should leave the definition program and return to CP/M. If you have been using drive M: you may need to make space by erasing the DBDEF files before continuing.

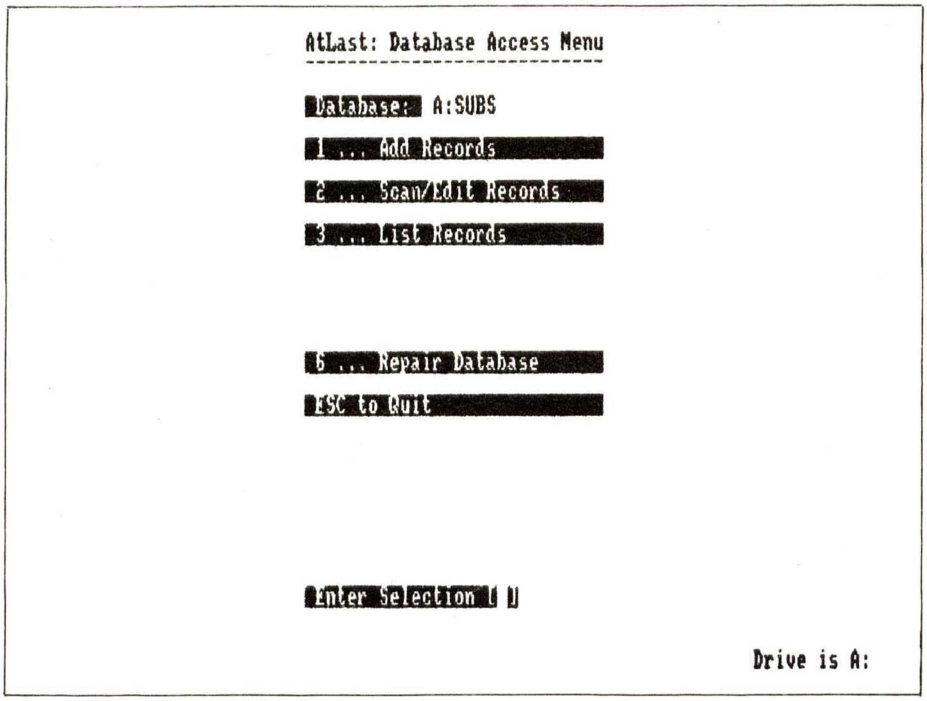
Using a data file

Any manipulation of the actual data itself must be done through the second program, DBUSE. To add data to the record you have just created enter the commands:

```
SETKEYS ATLAST.KEY
PAPER 11
DBUSE B:SUBS
```

On a single-drive machine the relevant files can be copied to the memory drive and the program can be put into effect with the command:

```
SUBMIT ATLAST B:SUBS
```



```
AtLast: Database Access Menu
----------------------------

Database:  A:SUBS

1 ... Add Records

2 ... Scan/Edit Records

3 ... List Records



6 ... Repair Database

ESC to Quit



Enter Selection [ ]

                                        Drive is A:
```

Figure 7.8 *The Database Access menu*

This program copies the program files to memory, sets up the keyboard, prepares the printer for continuous paper and then runs DBUSE from drive M.

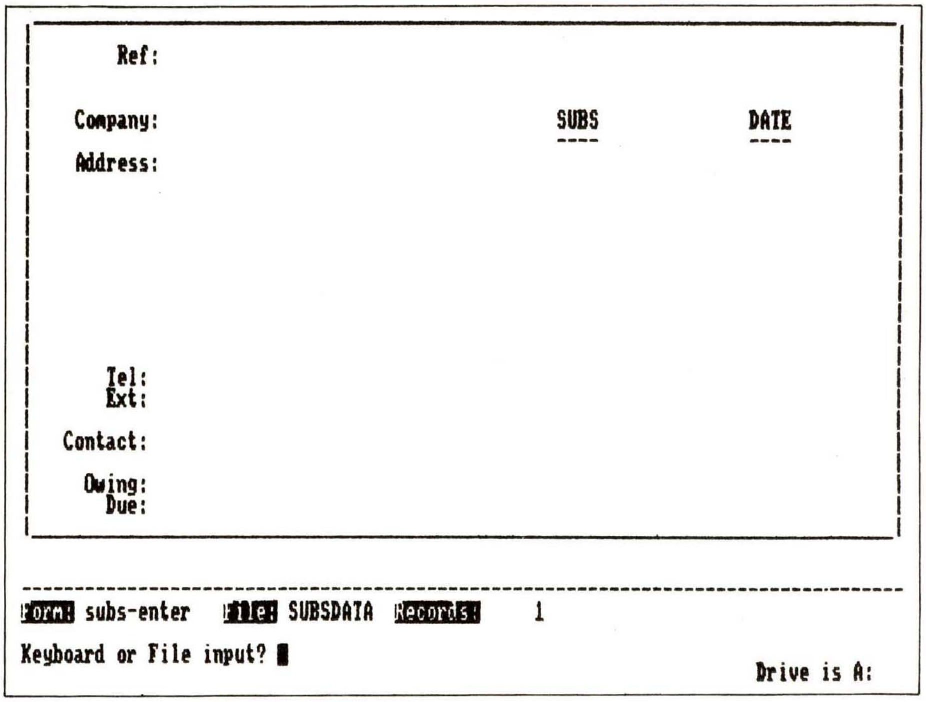
In either case the program is loaded and a new menu is displayed.

Entering data

The first option allows new records to be added (if the TRUE attribute was set). Enter the format to be used for the data entry and press K to enter data from the keyboard.

The layout previously designed appears on the screen and you can enter each of the fields in turn. The ↑ and ↓ keys can be used to move around the record to make corrections. The ← and → keys take you left or right in any field. New characters can be inserted by pressing the CHAR key, existing characters can be deleted with the DEL keys.

Every entry, completed by pressing RETURN, is validated by the program. For example, it will not allow text entry in numeric fields and all dates are



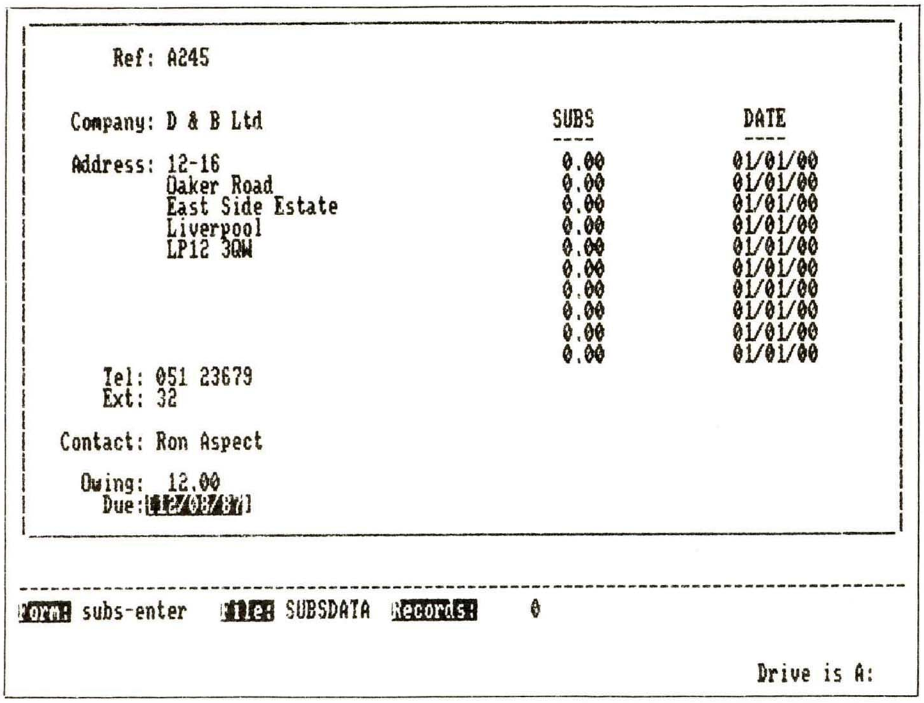
```
Ref:

Company:                                SUBS            DATE
                                        ----            ----
Address:






     Tel:
     Ext:

  Contact:

    Owing:
      Due:

-----------------------------------------------------------------------
Form: subs-enter   File: SUBSDATA   Records:     1

Keyboard or File input? ▮
                                                        Drive is A:
```

Figure 7.9 *Selecting input type*

checked to ensure that each part is within the required range. Times are also checked for hours in the range 0–23, minutes and seconds in the range 0–59. The program cannot check that text has been correctly spelled or that numbers are in the right range, of course.

Press EXIT to finish the record or RETURN when in the last field. New options are provided to re-edit or save the record. When it is saved the corresponding index files are automatically updated as well. Further records can then be entered.

Inspecting records
Option 2 from the main menu allows you to inspect any record. You must choose the format (which determines which file is looked at) and the key on which to search. All records are presented in the order of this key. You can choose the starting point by giving specific values for each part of the key, or just press RETURN in each case to start at the beginning of the file, i.e. the first record in that key order. It is sometimes useful to have a key consisting of just the reference number (which may be a serial field that is automatically updated).



```
        Ref: A245


    Company: D & B Ltd                               SUBS          DATE
                                                     ----          ----
    Address: 12-16                                   0.00        01/01/00
             Oaker Road                              0.00        01/01/00
             East Side Estate                        0.00        01/01/00
             Liverpool                               0.00        01/01/00
             LP12 3QW                                0.00        01/01/00
                                                     0.00        01/01/00
                                                     0.00        01/01/00
                                                     0.00        01/01/00
                                                     0.00        01/01/00
                                                     0.00        01/01/00
        Tel: 051 23679
        Ext: 32

   Contact: Ron Aspect

     Owing:  12.00
        Due:[12/08/87]

Form: subs-enter    File: SUBSDATA  Records:      0

                                                            Drive is A:
```

Figure 7.10 *Adding a record*

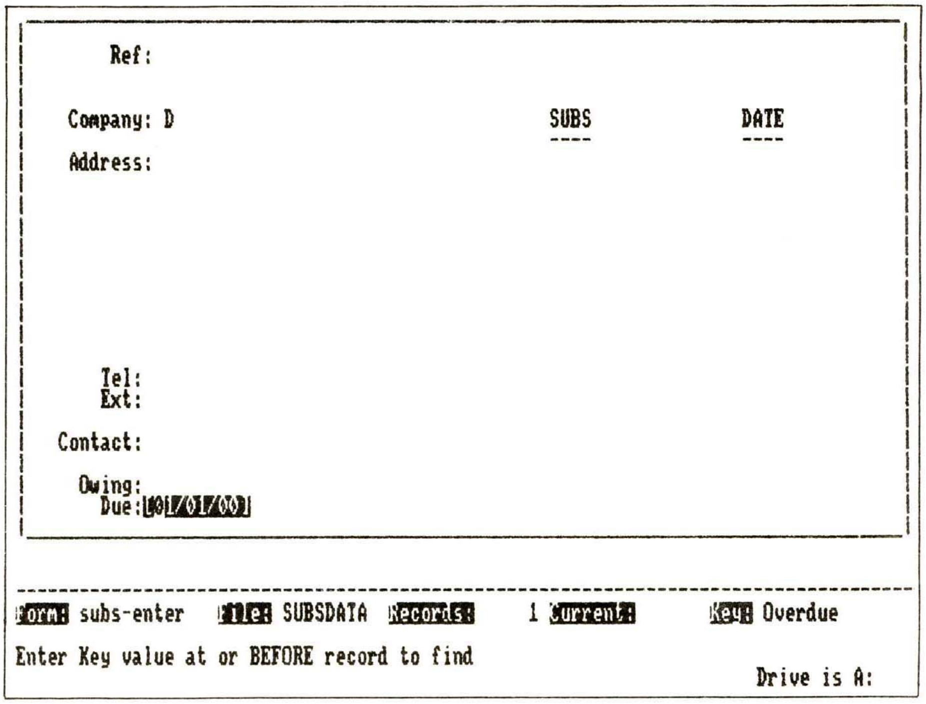
Editing records
When any field is displayed you can edit it by pressing E. As soon as the changes are completed the new version is saved on disk and the index file is updated. You can only edit or delete a record if the TRUE attribute was set when the file was defined. These attributes can be changed by running DBDEF again.

To move through the file, press N for the next record, B to move back to the previous record. A record is deleted (where allowed) by pressing D.

You can change to another format by pressing F. If you choose a format from another file then a record from that file is displayd. It is here that the linking of files becomes useful, since a shared key will result in a record from the second file being displayed.

Searching, selection and sorting

To search for a specific record, according to the key value, press S and enter the search values for the key. You only need to give a part of the field for text



```
      Ref:

   Company: D                                    SUBS          DATE
                                                 ----          ----
   Address:








      Tel:
      Ext:

  Contact:

    Owing:
      Due:01/01/00

Form: subs-enter   File: SUBSDATA   Records:   1 Current:       Key: Overdue
Enter Key value at or BEFORE record to find
                                                                Drive is A:
```

Figure 7.11 *Selecting key values to inspect a record*

fields; the program seaches for fields that start with this string. The first record to match the key values is displayed. Press N to move through the following records with the same key value. (They will all be together since the file is sorted according to the same key.) At any time you can select one of the other keys by pressing K.

Selections can only be made when printing the file, and the details of the conditions available are given in the next section.

Sorting is automatically taken care of, since records are always displayed in a specific order as a matter of course.

You can add a further sort key at a later stage by repeating the definition procedures. The new index file is not automatically created but can be made up to date by selecting the Repair option from the DBUSE main menu. This is quite a long process so should be used only as a last resort.

Key fields can only be deleted through the definition program, and the corresponding index file should also be deleted from disk.

Reports

The reporting facilities of AtLast are probably its most exciting feature, involving quite a lot more sophistication than either of the programs discussed previously. However, there are some drawbacks and limitations to the facilities.

Selecting option 3 from the main menu results in a request for the format and key. The records are automatically printed in the order defined by the key. You can choose the start and end of the print run by entering the first and last values for the key.

Before printing you must choose between printing all records or selecting records according to given criteria. These criteria are determined by giving the field name, the selection symbols and the value with which the field values are to be compared. The selections use abbreviations rather than the standard symbols, as follows:

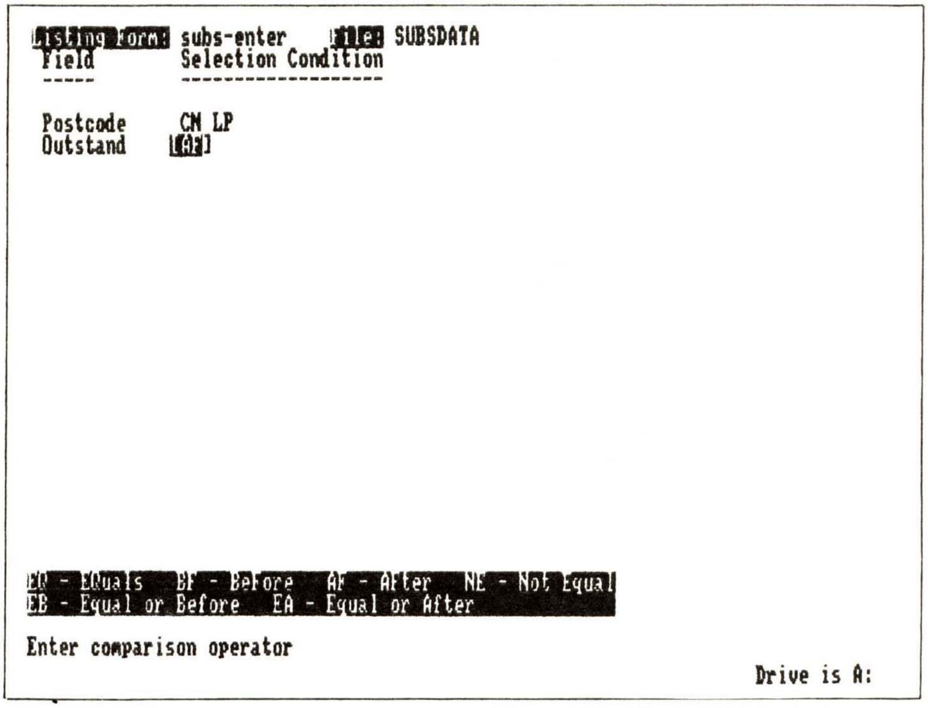
BF	Before (<)
EB	Equal to or before (< =)
AF	After (>)
EA	Equal to or after (> =)
EQ	Equal to (=)
NE	Not equal to (< >)

The method of selection is as described in Chapter 2 but with lower-case letters coming before all upper-case letters. For date fields the problems that arose for the Cardbox selections are overcome by the fact that all values are compared in date order.

Any number of selections can be made. A record is only printed if all conditions are satisfied. Thus this is the equivalent of the AND logical operator; there is no equivalent of the OR or NOT operators.

The list can be printed in either normal (ascending) or reverse (descending) order. It can be directed to the printer, the screen or a file.

You need to specify the number of lines to be printed per page and the page length. To produce a continuous listing, supply a value of 1 for each. Note that you can define a format for printing single columns of labels, but it is not possible to print more than one column at a time. (You could, of course, change the format after the first print run so that the left margin is much wider and put the roll of labels through the printer again. It is probably easier, however, to use specially prepared, single-column label sheets.)



```
Listing Form: subs-enter   File: SUBSDATA
Field         Selection Condition
-----         -------------------

Postcode      CN LP
Outstand      [GE]

EQ - EQuals   BF - BeFore   AF - AFter   NE - Not Equal
EB - Equal or Before   EA - Equal or After

Enter comparison operator
                                                    Drive is A:
```

Figure 7.12 *Entering conditions for the printout*

Looking after files

There are absolutely no file-handling facilities in AtLast. All this must be done from CP/M. There is only the Repair option to recover data from a corrupt file. This may be necessary if there is an interruption in power supply, if the disk is removed before you have worked back through the menu structure or if there is a stray fault in part of the disk. As usual, a properly backed up database will not have need of this rather complex process.

For each database the file definitions are held in a file with a .DEF extension, the formats are in a .FRM file and there is a corresponding .FIX index file. Each data file has a .DAT extension and the key data is held in an .IDX index file.

Data can be transferred between databases with a little patience. You must design a special format consisting only of field values (no extra text), with each field on a separate line and no gap between records. The file should then be printed, using this format, and specifying a disk file as the receiving device rather than the printer. The data can then be read into another file (by presing F rather than K when choosing option 1 to add records). The great advantage is that the data can be read into any file; the field names do not have to be the same though some care must be taken over the type (numeric data can be read into an alpha field, but the reverse is not true).

Support

The manual contains most of what you need to know about AtLast, but has to be read two or three times to become familiar with the jargon and general concepts. Only then can you begin to create a database. This is one of the few occasions when it can be extremely advantageous to use the sample data as practice, at least to try the complex file definition procedures; here, however, a description of the example files has been relegated to an appendix. The manual is almost unique in containing absolutely no sample screen displays so you really have to sit down in front of the machine with the program running to get a clear idea of what is going on.

Free telephone support is available and likely to be frequently needed.

Conclusion

AtLast provides comprehensive facilities for searching and sorting, though these are complex to master and confusing for the new user of computer databases. For many applications AtLast will provide essential extra facilities (in particular the ability to sort records) that are not available in Matchbox and Cardbox. If these are not really necessary then it may not be worth tackling the complexities of this program. You can really only make the best use of AtLast if you operate it frequently enough to become familiar with its structure and capabilities. If you have several files which are simple in their design, but need to be linked in some way, then AtLast could possibly provide the answer.

CHAPTER 8

SAGESOFT RETRIEVE

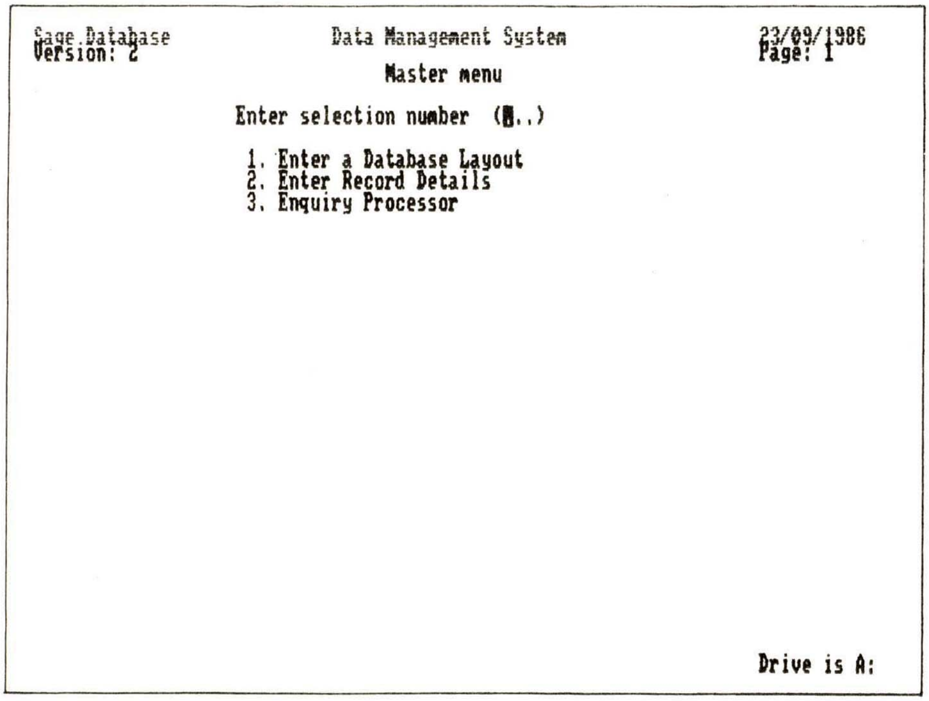
The fourth program under consideration is simple to use yet provides a range of advanced search, sort and print routines. Sagesoft Retrieve, by Sagesoft Plc, is a sophisticated program, capable of producing printouts to suit most applications.

Initial preparations

The first thing to do, as always, is to make a working copy of the program disk. In this case there are programs on both sides of the original disk so both sides should be copied. There is insufficient room to make side 1 a system disk and the files on this side are too large (152K) to fit into the memory drive on a PCW8256.

Side 1 contains the main program files while side 2 has the special printing routines.

The program always looks for the data files on drive B. On a single-drive Amstrad you will be asked to insert the disk for drive B or drive A when the program needs to access the data or program files, respectively. A great deal of disk swopping is involved and the use of Retrieve on a single-drive system is likely to be extremely frustrating.



```
Sage Database            Data Management System            23/09/1986
Version: 2                                                 Page: 1
                              Master menu

                     Enter selection number  (█..)

                      1. Enter a Database Layout
                      2. Enter Record Details
                      3. Enquiry Processor




                                                           Drive is A:
```

Figure 8.1 *The Master Menu*

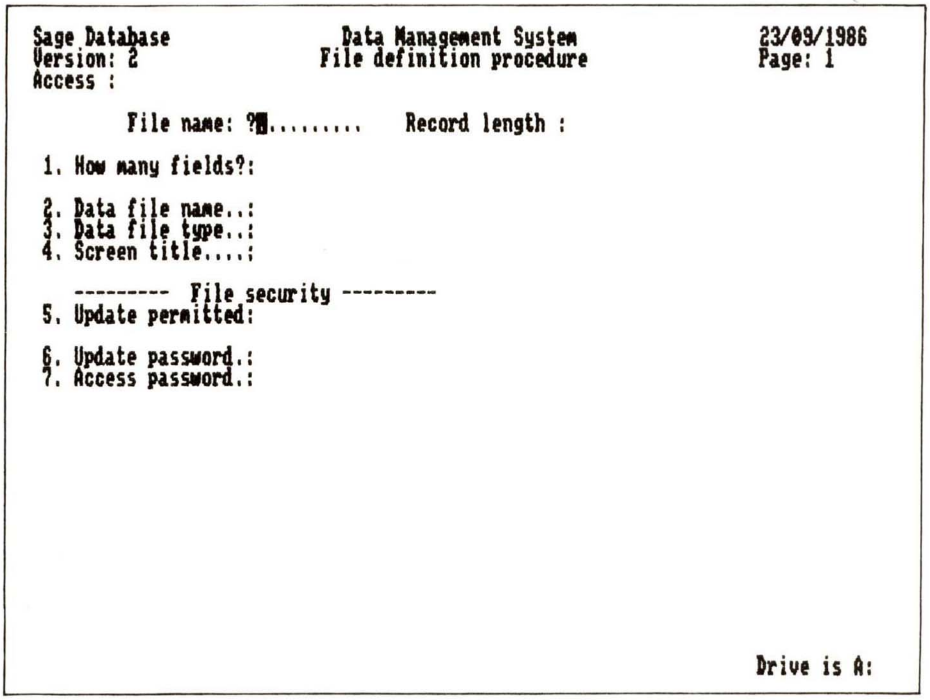
Data disks should also be formatted before you start. You will also need to prepare for continuous paper with the PAPER command. No other preparations are necessary, unless you want to use other than the standard printer.

Creating a data file

To create a new data file, load side 1 of the program disk and enter the command:

```
DATABASE
```

After you have entered the current date the Sagesoft 'Master Menu' is displayed, with three options available. The first allows you to design the structure of a file, the second provides for data entry and the third option leads to the powerful selection, sorting and printing routines. To end the program enter 'END'.



```
Sage Database              Data Management System          23/09/1986
Version: 2                 File definition procedure       Page: 1
Access :

          File name: ?█.........     Record length :

 1. How many fields?:

 2. Data file name..:
 3. Data file type..:
 4. Screen title....:

    --------- File security ---------
 5. Update permitted:

 6. Update password.:
 7. Access password.:







                                                           Drive is A:
```

Figure 8.2 *Defining a file*

Defining a file

On choosing option 1 you are required to enter a name for the *header* file, which stores details of the data file structure. As usual, this can be up to eight characters long. The program adds a .DCT extension. (Enter an asterisk to return to the menu.) You are instructed to load the disk for drive B, if you have not already done so.

The main database details are determined by a question and answer session. You must first decide how many fields there are to be; make sure you include a field for the record number.

The data file name should be left the same as the header file; this is done by pressing RETURN. The data file holds the data itself.

The data file type should always be given as K. This ensures that the program creates an index file for searches. The other possibilities are not detailed in the manual and therefore remain a mystery.

Next enter the screen title, which is an item of text to be displayed at the top of the screen during data entry. The next question asks whether updates of the file are to be permitted; that is, it needs to know whether or not the contents of the data file can be changed. There are three possiblities:

Y	The contents of the file can be changed. This is the option that should be selected when data is first entered.
N	The file can be inspected but cannot be changed. This answer can be changed if further modifications are needed.
Z	As for N, the file cannot be changed. In this case, however, the answer is permanently fixed and cannot be changed back to either Y or N.

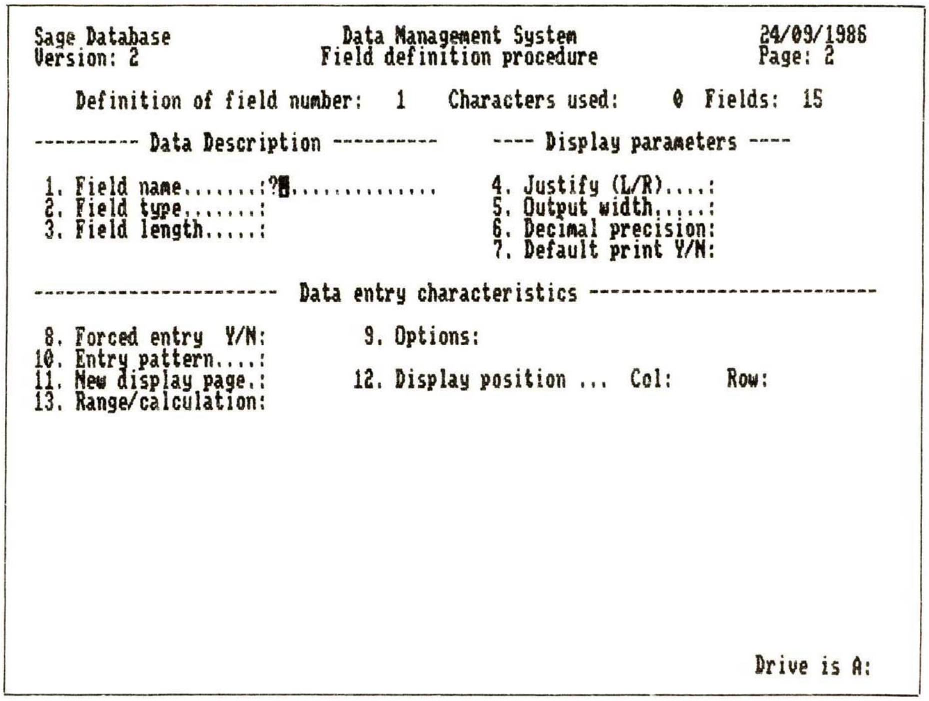
Two passwords are provided for security. If an Update password is entered the file can only be changed if the correct password is supplied, but the file can still be inspected without knowing the password. More drastic is the Access password, which must be given before the contents of the file can be either changed and viewed.

The passwords can be left blank by pressing RETURN. If a password is entered then it can be up to 8 characters long. Make sure you don't forget the password. There is no way of finding out what it is and the data becomes totally useless. Losing a password is as effective as deleting the file from disk.

On completing this screen of information you have a number of options. Any answer can be changed by entering the relevant number; entering an asterisk (*) takes you back to the main menu, ignoring all information; RETURN takes you on to the next part of the file definition.

Field definition
A similar question-and-answer session is needed to define the attributes of each field in the record. The field name can be up to 15 characters in length and a variety of field types is available:

KY	A key field used in searches (must be used for first field only).
TX	Text field.
IN	Integer field.
SP	Single precision (a decimal of up to 6 digits).
EP	Extended precision (a decimal of up to 14 digits).
VL	A numerical field, up to 14 digits, that is stored as text.
DT	Dates, with validation checks, in the form day/month/year.
DA	A date field that is compacted into two characters for storage. (This is obviously more useful than DT since it saves disk space; the existence of the DT option seems rather unnecessary.)
DS	A date stamp field. The value for this field is automatically set for each record by using the current date that is entered each time the program is started.
CK	A clock field, storing times in the form hours:minutes:seconds.



```
Sage Database              Data Management System              24/09/1986
Version: 2                Field definition procedure           Page: 2

   Definition of field number:  1   Characters used:    0  Fields:  15

---------- Data Description ----------      ---- Display parameters ----

 1. Field name.......:?█..............       4. Justify (L/R)....:
 2. Field type.......:                       5. Output width.....:
 3. Field length.....:                       6. Decimal precision:
                                             7. Default print Y/N:

---------------------- Data entry characteristics -------------------------

 8. Forced entry  Y/N:          9. Options:
10. Entry pattern....:
11. New display page.:         12. Display position ...  Col:     Row:
13. Range/calculation:




                                                              Drive is A:
```

Figure 8.3 *Defining a field*

A further five 'Maths' field types are available. These are fields that are to hold the results of calculations involving the values given in other fields. The Maths field type determines the way the result is presented after the calculation is complete. Full precision is used for the calculation itself, regardless of the type of fields used. The five types are:

MI	Integer
MS	Single precision
ME	Extended precision
MV	Value field
MD	Display field: this field is not shown during data entry but is shown when the file is being inspected

This is a very useful selection of field types. The key field can only be used for the first field in the record and you should always include such a field. The next questions are as follows:

Field Length	The maximum length for text and value fields only.

Justify (L/R)	Determines whether the text is on the left of the field or the right. Usually text is on the left and numeric fields are on the right.
Output width	The width of the field when printed. Press RETURN to accept the default.
Decimal precision	The number of decimal places printed on reports. This does not affect the accuracy of the calculations.
Default print	The field is automatically printed on all reports if this is set to Y.
Forced entry	A value must always be entered for the field if this is set to Y.
Options	One or more additional attributes may be specified as follows:

A	Automatic completion of the field when it is full without the need to press RETURN.
B	Block field which must always be completely filled.
C	Converts all characters to upper case.
H	Hidden field, whose value is never displayed or printed.

Entry pattern	Text fields can be forced to follow a specific pattern of entry with each character being specified as alphabetic (A), numeric (N) or either (X).
New display page	If this is set to Y the program moves on to a second (or subsequent) entry screen after this field has been entered.
Display position	The position on the screen of the first character in the field must be given in terms of column and row number. If the previous field was the last of a page then there is no need to worry about fields apparently overlapping.
Range/calculation	If you want the value given to be validated, then the range can be entered here. For example, for a percentage the range would be 1–100. For text you

can specify a list of options, which must be given in quotes, separated by oblique strokes; for example, the Title field in a personnel file may be limited to "Mr/Mrs/Miss/Ms".

This part can also be used to enter the formula for calculated fields.

Having entered the details for a field, various options are given to change any item (confusingly referred to as a field number), ignore all details or move on to the next field.

After entering all fields you can define a new file. Enter an asterisk for the file name to return to the Master menu.

Calculated fields For fields of type MI, MS, ME, MV or MD a formula must be given. This should refer to the various fields in the form F1, F2, F3, . . ., where the number is the field number. The formula may also include numeric values and the following arithmetical symbols:

+	Addition
–	Subtraction
*	Multiplication
/	Division

You can change the order of the calculation by putting part of the formula in brackets. Anything in brackets is calculated first. For example, to find the average of fields 5 and 7 will require the formula:

```
(F5 + F7)/2
```

This can be an extremely useful facility.

Example The fields that may be used in the SUBS example are:

No.	*Name*	*Type*	*Length*	*Justify*	*Width*	*Precision*	*Print*
1	Number	KY		R	2		
2	Reference	TX	4	L	4		Y
3	Company	TX	20	L	20		Y
4	Address1	TX	20	L	20		
5	Address2	TX	20	L	20		
6	Address3	TX	20	L	20		
7	Address4	TX	20	L	20		
8	Postcode	TX	7	L	7		
9	Telephone	TX	10	L	10		
10	Extension	TX	4	L	4		
11	Contact	TX	30	L	30		
12	Subscription	SP		R		2	
13	Renewal-date	DA		L			
14	Outstanding	SP		R		2	
15	Paid	MS		R			

No.	*Forced*	*Options*	*Pattern*	*Col*	*Row*	*Range/ Calc*
1				1	4	
2	Y	ABC	ANNN	35	4	
3	Y			1	6	
4				1	8	
5				1	9	
6				1	10	
7				1	11	
8		C		1	14	
9		A	NNNNNNNNNN		35	12
10			NNNN	35	13	
11				35	15	
12				1	18	10-70
13				1	19	
14				35	18	10-70
15				35	19	F12-F14

The database has now been fully defined. Note that there is very little control over the appearance on screen of the blank data forms, apart from the position of the individual fields.

Redefining the file The file definition can be changed at any time, even if data has been entered. Choose the option to enter a layout, select the file and press RETURN to advance to the field definition stage. You can make changes to where the field appears on the screen or enter a new value for the type or length. For more drastic changes, select any field and press EXIT. You then have options to insert, delete or move fields.

After each major operation the file is *restructured*. Before carrying out any such operation make sure you have a copy of the file in case any data is unexpectedly lost; changes in field type in particular may result in damage to data. You will also need to ensure that there is sufficient free space on the disk for a duplicate file to be temporarily created.

Using a data file

From the main menu select option 2 to *Enter Record Details*. The blank card is displayed for the first time. If it is unsatisfactory you can press RETURN to go back to the main menu and then make changes.

Entering data

To start entering values type '+' for the record number. Values can be entered for each of the fields. Suitable validation is carried out and the person making the entry is forced to enter certain fields, to completely fill others or to enter according to some pre-determined pattern (such as a reference consisting of a letter and three numbers).

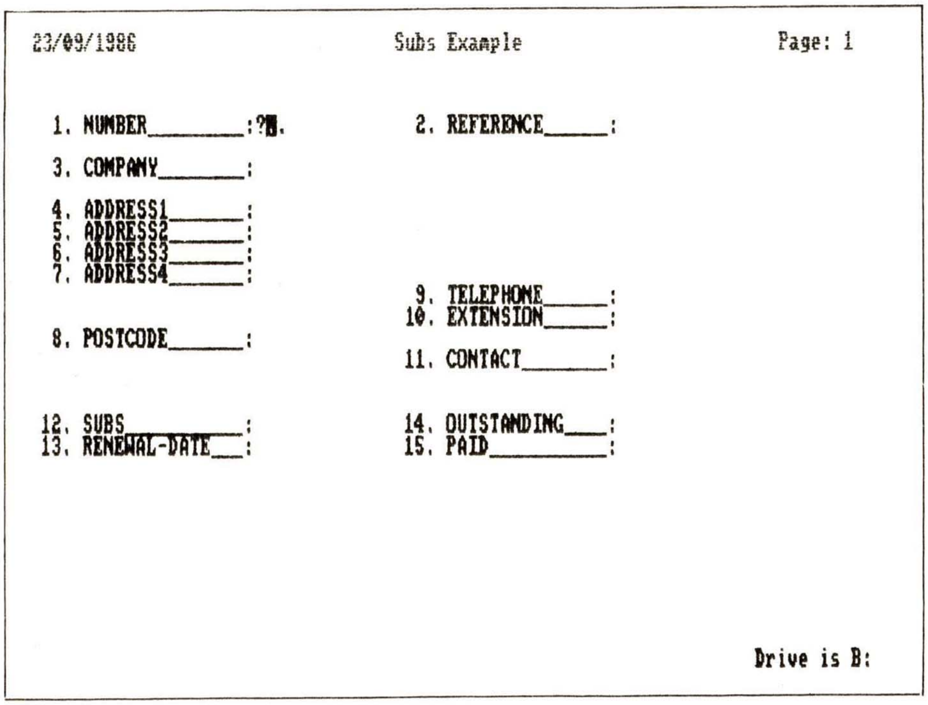
After entering a record you can enter a field number to change its value, press * to cancel all entries, or press RETURN to accept the details and 'post' the details to the data file. The program then moves on to the next page (for those files with more than one entry screen) or starts on the next record. You need to enter + again to add another record, or RETURN to work on a different file.

Inspecting and editing records

To look at any particular record choose option 2 again from the main menu and, instead of pressing +, enter the appropriate record number. The details are displayed as before and entering a field number allows you to make changes. To delete a record entirely, press D.

Searching, selection and sorting

Once the data has been entered the real power of Sagesoft Retrieve comes to the fore. By selecting the third option from the main menu, the *Enquiry Processor*, you are given the opportunity to interrogate the database using fairly complex instructions. These are entered in the form of 'sentences' that can be constructed in a variety of formats. Although they look like some



```
23/09/1986                       Subs Example                       Page: 1

  1. NUMBER________:?█.          2. REFERENCE_____:
  3. COMPANY_______:
  4. ADDRESS1______:
  5. ADDRESS2______:
  6. ADDRESS3______:
  7. ADDRESS4______:
                                 9. TELEPHONE_____:
                                10. EXTENSION_____:
  8. POSTCODE______:
                                11. CONTACT_______:

 12. SUBS__________:            14. OUTSTANDING___:
 13. RENEWAL-DATE__:            15. PAID__________:




                                                              Drive is B:
```

Figure 8.4 *A blank record*

strange, incomprehensible form of pidgin English the construction is very logical and quite easy to compile once you become familiar with it.

The program indicates that it is waiting for an instruction with the prompt:

```
Ready:
```

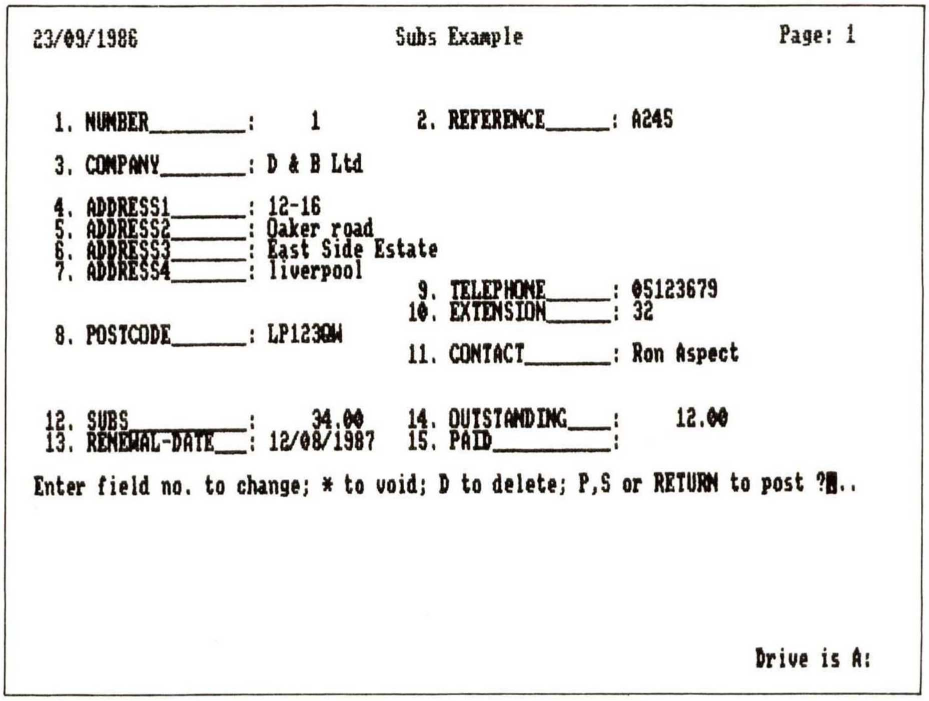
When you want to leave this part of the program, enter the MENU instruction. Each instruction can be entered in any combination of upper and lower case letters.

Listing records

The first 'verb' is LIST, which displays a series of records on the screen, in the original record order. To look at all records enter the instruction followed by the filename:

```
LIST SUBS
```

Only those fields for which the default print is set to Y are shown.



```
23/09/1986                     Subs Example                     Page: 1


  1. NUMBER_________:     1       2. REFERENCE______: A245

  3. COMPANY________: D & B Ltd

  4. ADDRESS1_______: 12-16
  5. ADDRESS2_______: Oaker road
  6. ADDRESS3_______: East Side Estate
  7. ADDRESS4_______: liverpool
                                   9. TELEPHONE______: 05123679
                                  10. EXTENSION______: 32
  8. POSTCODE_______: LP123QW
                                  11. CONTACT________: Ron Aspect


 12. SUBS___________:       34.00 14. OUTSTANDING____:      12.00
 13. RENEWAL-DATE___: 12/08/1987  15. PAID___________:
Enter field no. to change; * to void; D to delete; P,S or RETURN to post ?█..




                                                                Drive is A:
```

Figure 8.5 *Entering a record*

To only look at specific fields, their names or numbers must be given in a SHOW instruction. This should follow on from the LIST command. For example, to list the company name, subscription and renewal date the instructions would be:

```
LIST SUBS SHOW COMPANY SUBSCRIPTION RENEWAL-DATE
```

A faster way of writing this, if the field numbers are known, is:

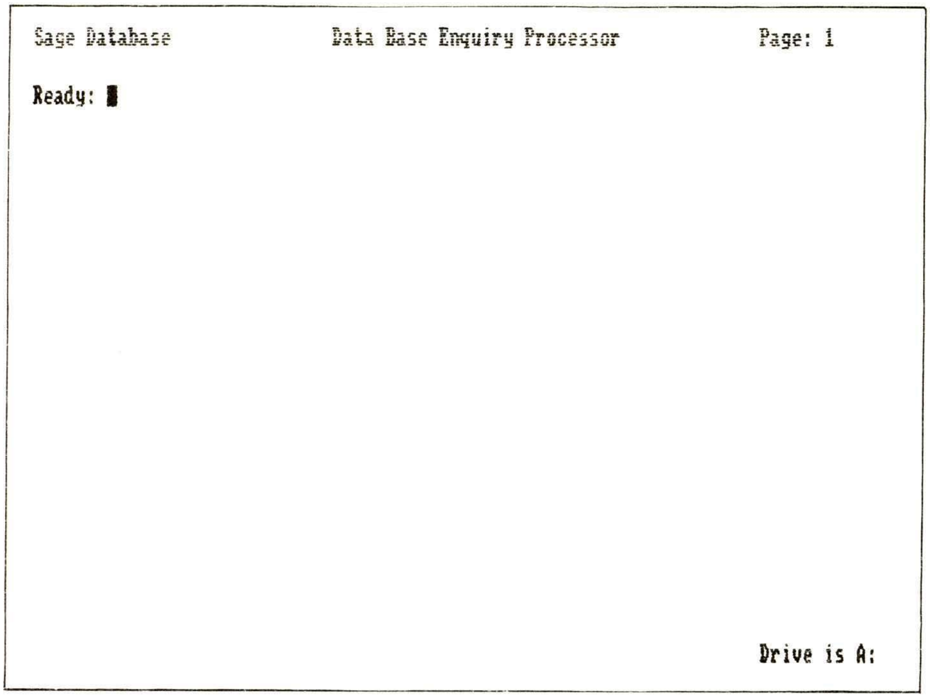
```
LIST SUBS SHOW 3 12 13
```

Any number of fields can be given and a field can be repeated.

Selection

To find records that satisfy a condition or to list records that fall within some category, the WITH *modifier* can be added to the LIST instruction:

```
LIST SUBS SHOW COMPANY WITH OUTSTANDING > 50
```



```
Sage Database              Data Base Enquiry Processor              Page: 1

Ready: █




                                                                 Drive is A:
```

Figure 8.6 *The Enquiry Processor*

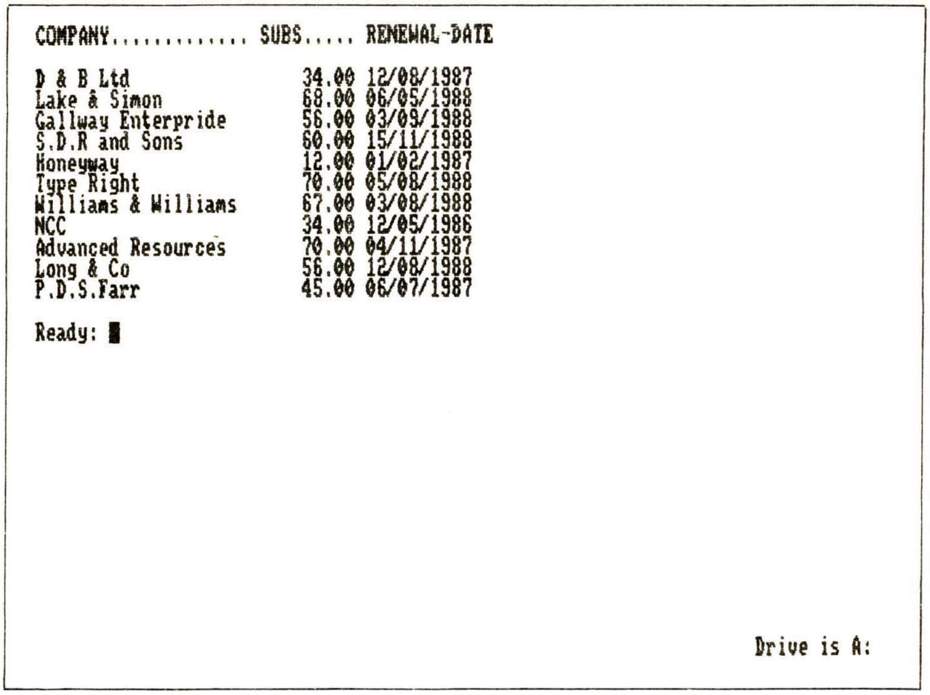
This lists all companies who currently owe more than £50. All the standard conditional symbols are available. Any text to be compared must be in quotes. For example:

```
LIST SUBS SHOW REFERENCE WITH ADDRESS4 = "Liverpool"
```

Note that the case of instructions, filenames and fields is unimportant. The only time it matters is when referring to text in quotes.

If searching for a part of a string, square brackets must be used. An opening bracket ([) indicates that the string does not have to start the field, while a closing bracket (]) means that the string does not have to be at the end of the field. For example, the following strings may be used in a condition when searching for addresses:

"LIV]"	All text starting with "LIV", e.g. Liverpool, Livingston
"[POOL"	All text ending with "POOL", e.g. Liverpool, Ullapool
"[VER]"	All text containing "VER", e.g. Liverpool, Vernwood, Andover



```
COMPANY............. SUBS..... RENEWAL-DATE

D & B Ltd                34.00 12/08/1987
Lake & Simon             68.00 06/05/1988
Gallway Enterpride       56.00 03/09/1988
S.D.R and Sons           60.00 15/11/1988
Honeyway                 12.00 01/02/1987
Type Right               70.00 05/08/1988
Williams & Williams      67.00 03/08/1988
NCC                      34.00 12/05/1986
Advanced Resources       70.00 04/11/1987
Long & Co                56.00 12/08/1988
P.D.S.Farr               45.00 06/07/1987

Ready: █

                                                      Drive is A:
```

Figure 8.7 *Listing records*

Conditions can be combined with the AND and OR operators:

```
LIST SUBS SHOW REFERENCE WITH SUBSCRIPTION > 10 AND +
RENEWAL-DATE < 01/03/87 OR OUTSTANDING > 50
```

Note that sentences extending over more than one line must end with a + sign to indicate continuation on the next line; words must not be split.

Conditions are calculated from left to right. The example above yields two categories of records: those that satisfy both conditions of a subscription more than £10 with a renewal date before March 1987; and those who owe more than £50 (regardless of subscription level or renewal date). All the records are shown as one selection in order of record number.

You can find out the number of records that satisfy a condition, rather than paging through them all, with the COUNT instruction:

```
COUNT SUBS WITH SUBSCRIPTION > 30
```

This figure can also be produced as a proportion of the total number of records:

```
RATIO SUBS WITH 12 > 30
```

Any number quoted on the left of a condition is always a field number. Only values, not field names or numbers, can end a condition.

Selecting records for editing

Further instructions are given to select specific records for editing:

```
CHANGE SUBS WITH RENEWAL-DATE < 01/01/87
```

Each record with a renewal date prior to 1987 is displayed in turn and can be edited. Alternatively, you can go straight into the main editing mode with:

```
AMEND SUBS
```

In a similar fashion, the file structure can be adjusted without the need to go through the main menu:

```
DEFINE SUBS
```

After making the changes you are returned to the main menu.

A selection of records can be permanently deleted, though this rather powerful command should be used with extreme caution:

```
DELETE SUBS WITH 13 < 01/01/87
```

All records with a pre-1987 renewal date are erased from the file.

Sorting

The sort routines are much easier to use than those of AtLast. There is no need to decide how to sort until you actually need to rearrange the records. The sentence begins with the SORT command and filename; it is followed by each field to be used in sorting, preceded by the word BY. For example:

```
SORT SUBS BY 12 BY 13 BY 3
```

In this example, the records are displayed in order of subscription rate, with those on the same value being sorted by renewal date; any that have both of these values the same are shown in alphabetical order of company name. Sorting is always done alphabetically for text fields, in ascending order for numeric fields or in increasing order of date or time.

A SHOW instruction can be added at the end to reduce the fields displayed.

Reports

The generation of reports is very straightforward, though the number of formats you can devise is very limited. For a columnar list format there are some simple instructions. Using PRINT instead of LIST results in the information being directed to the printer:

```
PRINT SUBS SHOW 1 2 3 9 10 11
```

This produces a list of telephone numbers and related information.

Alternatively, a selection of records or the re-arranged file can be directed to the printer rather than the screen by adding ON PTR (or ON PRINTER) to the LIST or SORT command:

```
SORT SUBS BY 3 WITH 12 > 20 ON PTR
```

Totals of specified fields are printed at the bottom of the list by adding a TOTAL instruction. For instance, to produce a total of the amount outstanding you could use:

```
PRINT SUBS TOTAL OUTSTANDING
```

To produce sub-totals in sorted lists use BREAK-ON with the field to be sub-totalled:

```
SORT SUBS BY 13 BREAK-ON 13 TOTAL 14 ON PTR
```

This produces a list on the printer in order of renewal date with a sub-total of amounts outstanding for each date.

The printer must have been suitably set up before entering the program.

Print utilities

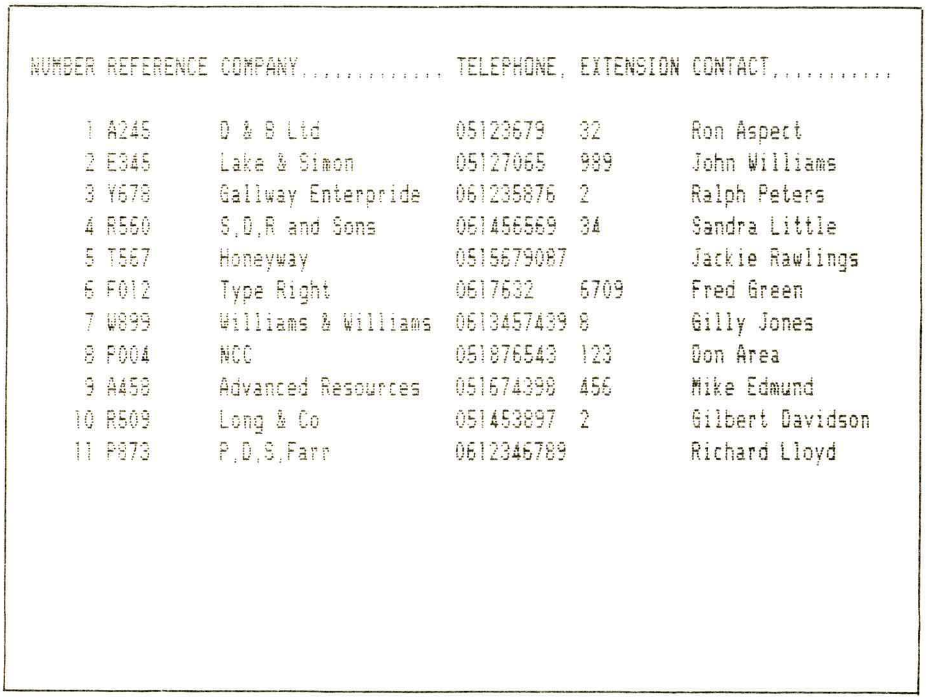
The second side of the Retrieve disk contains a set of utility programs for producing standard letters and documents or address labels. To run any of these you must start the program again from the CP/M A> prompt. With the second side of the disk facing the monitor screen enter the same command as before:

```
DATABASE
```

A new menu of four options is displayed.

The text editor

The first provides access to the text editor. This allows you to create reasonably large volumes of text: up to 220 lines, with 75 characters per line.



```
NUMBER REFERENCE COMPANY............. TELEPHONE. EXTENSION CONTACT..........

     1 A245      D & B Ltd            05123679   32        Ron Aspect
     2 E345      Lake & Simon         05127065   989       John Williams
     3 Y678      Gallway Enterpride   061235876  2         Ralph Peters
     4 R560      S.D.R and Sons       061456569  34        Sandra Little
     5 T567      Honeyway             0515679087           Jackie Rawlings
     6 F012      Type Right           0617632    6709      Fred Green
     7 W899      Williams & Williams  0613457439 8         Gilly Jones
     8 P004      NCC                  051876543  123       Don Area
     9 A458      Advanced Resources   051674398  456       Mike Edmund
    10 R509      Long & Co            051453897  2         Gilbert Davidson
    11 P873      P.D.S.Farr           0612346789           Richard Lloyd
```

Figure 8.8 *A database report*

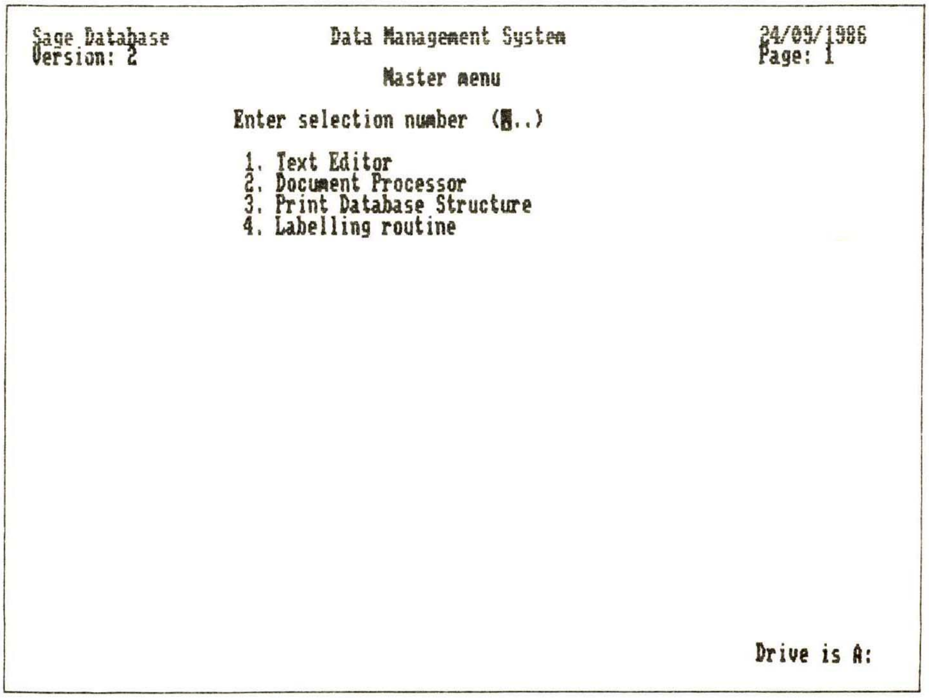
After selecting option 1, specify the filename (giving it a .WP extension). Then enter the text. You must press RETURN at the end of each line. The program does not allow particulary fast typing, especially at the beginning of a line.

You can move around the text with the cursor control keys. ALT-C and ALT-R, respectively, take you forward and back a page at a time. Anything typed is either inserted at the cursor position or overwrites the existing text; change between these two states by pressing ALT-V. Text is deleted with the DEL keys.

A completed document can be printed by pressing ALT-P. Documents should be regularly saved during editing by pressing ALT-Q.

Merging data The main use of a text editor with a database program is to produce standard documents.

Before running the text editor you should store in a separate file the data that is to be merged with the document. For example:



```
Sage Database                 Data Management System                    24/09/1986
Version: 2                                                              Page: 1
                                   Master menu

                         Enter selection number  (█..)

                          1. Text Editor
                          2. Document Processor
                          3. Print Database Structure
                          4. Labelling routine



                                                                        Drive is A:
```

Figure 8.9 *The Utilities menu*

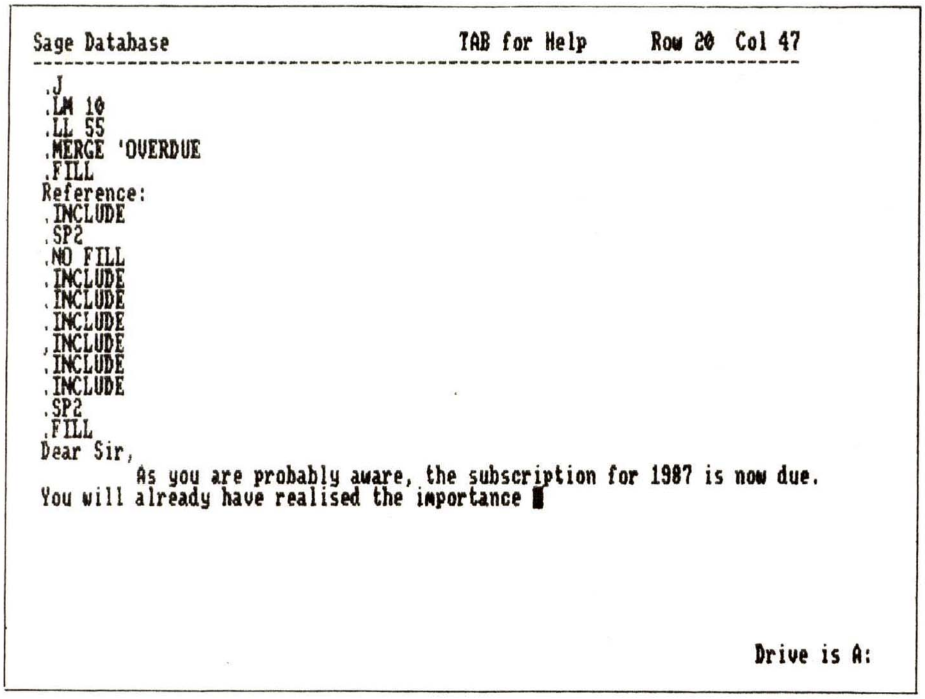
```
LIST SUBS WITH 13 < 01/03/87 SHOW 2 3 4 5 6 7 8 13 14 +
ON OVERDUE
```

A file called OVERDUE.DAT is created containing details of overdue subscriptions. Make sure you include all the fields that will be needed in the standard document and that they are in the right order.

Instructions can be included in the document for varying the layout and format, or for including values from the data file. These all take the form of 'dot' commands, each of which must be placed on a separate line. For example, to justify the text (so that it has a straight right-hand margin), with a left margin of ten characters and line length of 55 characters requires the commands:

```
.J
.LM 10
.LL 55
```

You can tell the program which data file to use with a .MERGE command.



```
Sage Database                       TAB for Help     Row 20  Col 47
---------------------------------------------------------------------
.J
.LM 10
.LL 55
.MERGE 'OVERDUE
.FILL
Reference:
.INCLUDE
.SP2
.NO FILL
.INCLUDE
.INCLUDE
.INCLUDE
.INCLUDE
.INCLUDE
.INCLUDE
.SP2
.FILL
Dear Sir,
          As you are probably aware, the subscription for 1987 is now due.
You will already have realised the importance

                                                        Drive is A:
```

Figure 8.10 *Creating a document*

To include a data value use the .INCLUDE command; the program works through each item in the data file in turn without any regard for field names or the ends of records. Therefore the heading of a standard letter could be:

```
.MERGE 'OVERDUE
.FILL
Reference:
.INCLUDE
.SP2
.NO FILL
.INCLUDE
.INCLUDE
.INCLUDE
.INCLUDE
.INCLUDE
.INCLUDE
.SP2
.FILL
Dear Sir,
```

The .SP instruction is used to leave blank lines (pressing RETURN on its own is effective). The .FILL instuction tells the program to use the data values and any following text to fill the line so that there are no gaps in the text, while .NO FILL ensures that each new data value starts a new line.

After saving the text, select option 2 from the menu to print the documents. Supply the filename, number of copies required, starting point and whether or not the program should pause after each sheet (to allow for single sheets or continuous stationery). Printing continues with each record in the data file until it runs out of data or you press EXIT.

Database structure

The third option in the Utilities menu produces a printout of the database structure, listing details of fields and so on.

Labels

The printing of address labels (or other occasions when a vertical list of details for each record is required) is very straightforward. Select option 4 from the Utilities menu, specify the data file name, e.g. OVERDUE.DAT, and the number of fields that were selected from each record. The labels are printed in a vertical list. There is an option to print a test pattern first to line up the labels.

Looking after files

Retrieve includes a backup procedure as part of the Enquiry Processor, which should be entered in the form:

```
BACKUP SUBS.DAT TO A:SUBS.DAT
```

Deletion and renaming of files must be carried out with the relevant CP/M commands, as must disk copying.

Any list can be directed to an ASCII file by specifying a filename in a LIST or SORT instruction:

```
LIST SUBS WITH 3 > "MJ" ON SUBS.MZ
```

All details of companies whose names start with letters in the second half of the alphabet are stored in an ASCII file called SUBS.MZ and the data can be read into a word processor. There does not appear to be any method of reading the data into another Retrieve file.

A selected list can be saved on disk for future use in the same file by using the

SAVE-LIST command after the SELECT command. SELECT is identical to LIST except that the records are not actually displayed or printed, but are prepared for a SAVE-LIST command. For example:

```
SELECT SUBS WITH POSTCODE = "LP]"
SAVE-LIST LIVERPOOL
```

A file called LIVERPOOL.LIS is created. This can be recalled at any time and then inspected with the command:

```
GET-LIST LIVERPOOL
```

A redundant list is erased from disk with:

```
DELETE-LIST LIVERPOOL
```

A list can be copied to a backup file or another disk with the command:

```
COPY-LIST LIVERPOOL A:LIVERPOOL
```

The MENU command takes you back to the main menu. At any stage you can leave the program and return to CP/M either by entering the SYSTEM command or by working back through the menu levels. All data on disk is secured when the system closes down. If you interrupt the program accidentally, some of the more recent data may be lost.

Support

The manual supplied with Sagesoft Retrieve is clear and concise. It succeeds in mixing instructions and examples so that it is constructive without the necessity for the impatient user to process endless sample data. (There is a tutorial section at the beginning of the manual, but the main manual is so instructive you can happily skip that part. There is also a tutorial cassette tape if you really feel in need of extra help.)

Free telephone support is provided for 90 days, after which time you must expect to pay an annual charge. Whether or not you consider it worthwhile will depend on the nature and importance of the data you are storing. If you have stored a complete mailing list, and this is the lynch-pin of your business, then it may be wise to subscribe to Sagecover. This helps to guard against those occasions when the only copy of the file gets corrupted two days before you are due to do a mailshot. Of course, there is no guarantee that the data will be recoverable, but Sagesoft should be able to give some helpful advice on how to make the best of such a situation.

Conclusion

Sagesoft Retrieve is simple to use, yet highly effective. It allows almost limitless sorting and selection criteria, with data stored in virtually every field type you could ask for. It is limited in its display and print-formatting procedures, but the mail-merging and labelling facilities compensate for this.

The main drawback of Retrieve is that it is not really suitable for use on a single-drive Amstrad PCW8256.

The manual is well-designed, easy to understand and completes a good, professional software package that is a pleasure to use.

CHAPTER 9

THE CAMBASE DATABASE

The Cambase Database, from Camsoft Plc, introduces the idea of procedures and programming. It also incorporates most of the facilities described in earlier chapters.

Initial preparations

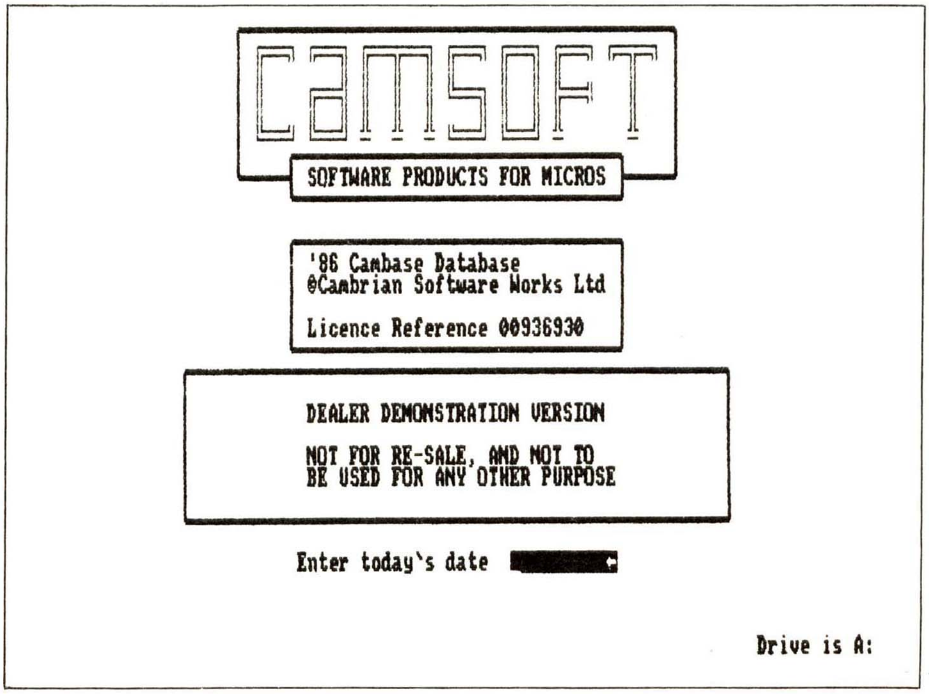
You should prepare suitable work disks and some blank data disks, as with all the other databases. There is sufficient room on the program disk to include some of the more useful CP/M utilities (for instance DISCKIT, PIP, SUBMIT, SETSIO, DEVICE and PAPER) as well as the main system file. If you have only a single-drive Amstrad then PIP.COM and SUBMIT.COM must be included on the work disk so that you can use the CAMUSE.SUB batch file.

The printer should be set up before you start the program. No other installation procedures are needed.

Creating a data file

Cambase is put into effect on a double-drive machine by entering the command:

```
CAMBASE
```



```
'86 Cambase Database
©Cambrian Software Works Ltd

Licence Reference 00936930

DEALER DEMONSTRATION VERSION

NOT FOR RE-SALE, AND NOT TO
BE USED FOR ANY OTHER PURPOSE

Enter today's date

Drive is A:
```

Figure 9.1 *The title screen*

On a single-drive Amstrad you can copy the relevant files to memory and automatically start CAMBASE by entering:

```
SUBMIT CAMUSE
```

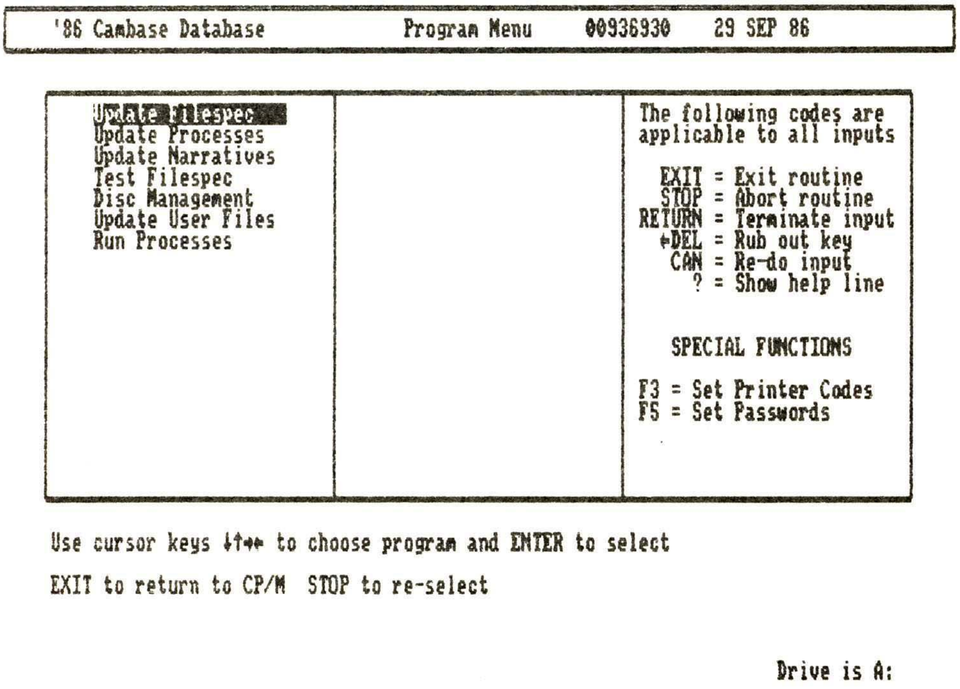
A title screen is displayed and you are required to enter the current date. Cambase does not use the standard date format adopted by most programs, but insists on an entry in the following format:

```
DD MMM YY
```

The day (DD) is a two-digit number, the month (MMM) is a three-letter abbreviation and the year (YY) is two digits. Suitable dates include:

```
20 MAY 87
01 JAN 88
```

All dates are validated immediately. The date always appears on printouts. However, you may not always want to use the current date (for example, if preparing a report to go with a document at a later date or producing a set of



```
'86 Cambase Database          Program Menu     00936930    29 SEP 86

Update Filespec                          The following codes are
Update Processes                         applicable to all inputs
Update Narratives
Test Filespec                              EXIT = Exit routine
Disc Management                            STOP = Abort routine
Update User Files                        RETURN = Terminate input
Run Processes                              ←DEL = Rub out key
                                            CAN = Re-do input
                                              ? = Show help line

                                           SPECIAL FUNCTIONS

                                         F3 = Set Printer Codes
                                         F5 = Set Passwords

Use cursor keys ↓↑←→ to choose program and ENTER to select

EXIT to return to CP/M  STOP to re-select

                                                       Drive is A:
```

Figure 9.2 *The main menu*

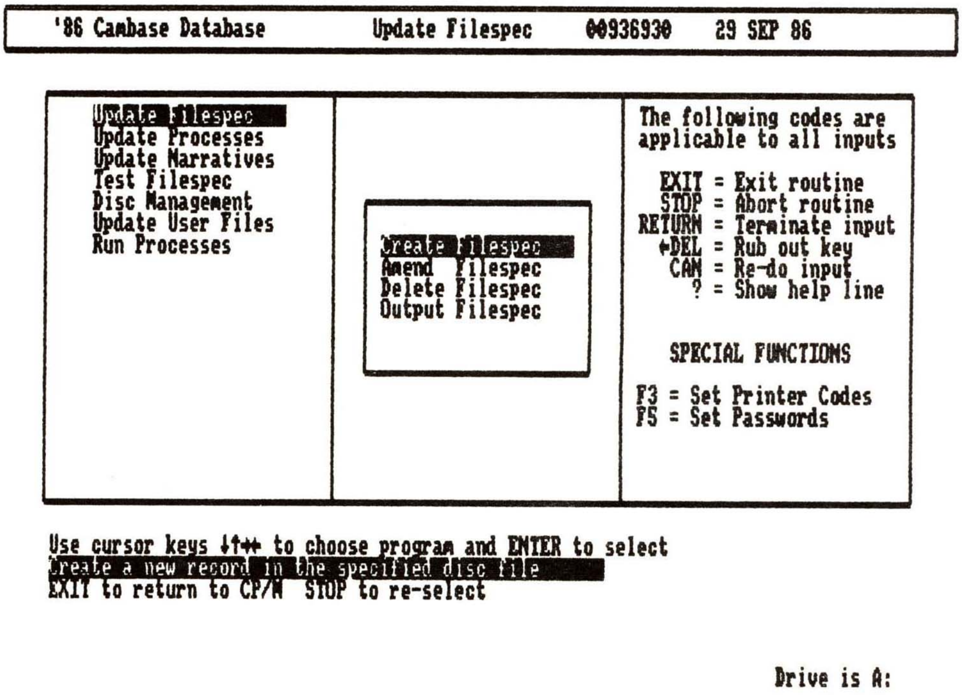
printouts over a number of days). If this is the case you can always 'cheat' by entering a different date at this stage.

Cambase uses a menu system, with question-and-answer sessions for determining the basic structure of the file. Any option can be selected from the menu by highlighting it with the cursor keys and pressing RETURN. Initially, to create a new file structure, select the first item *Update Filespec*.

File attributes

A sub-menu is displayed in the centre of the screen, the first option of which is *Create Filespec* (short for 'file specification'). Select this option and you will be expected to enter the main details for the file. Firstly you must give the file a number. Each data disk can hold, at most, nine data files and each of these has a unique number. If you want to edit an existing file's structure then you can enter its file number here. Otherwise, enter a new number between 1 and 9.

An option is given to copy from an existing file and then edit the specification. This is a useful alternative if you already have a file whose



```
'86 Cambase Database        Update Filespec    00936930    29 SEP 86

Update Filespec                                        The following codes are
Update Processes                                       applicable to all inputs
Update Narratives
Test Filespec                                            EXIT = Exit routine
Disc Management                                          STOP = Abort routine
Update User Files                                      RETURN = Terminate input
Run Processes               Create Filespec              ←DEL = Rub out key
                            Amend  Filespec               CAN = Re-do input
                            Delete Filespec                 ? = Show help line
                            Output Filespec
                                                         SPECIAL FUNCTIONS

                                                       F3 = Set Printer Codes
                                                       F5 = Set Passwords

Use cursor keys ↓↑←→ to choose program and ENTER to select
Create a new record in the specified disc file
EXIT to return to CP/M  STOP to re-select

                                                            Drive is A:
```

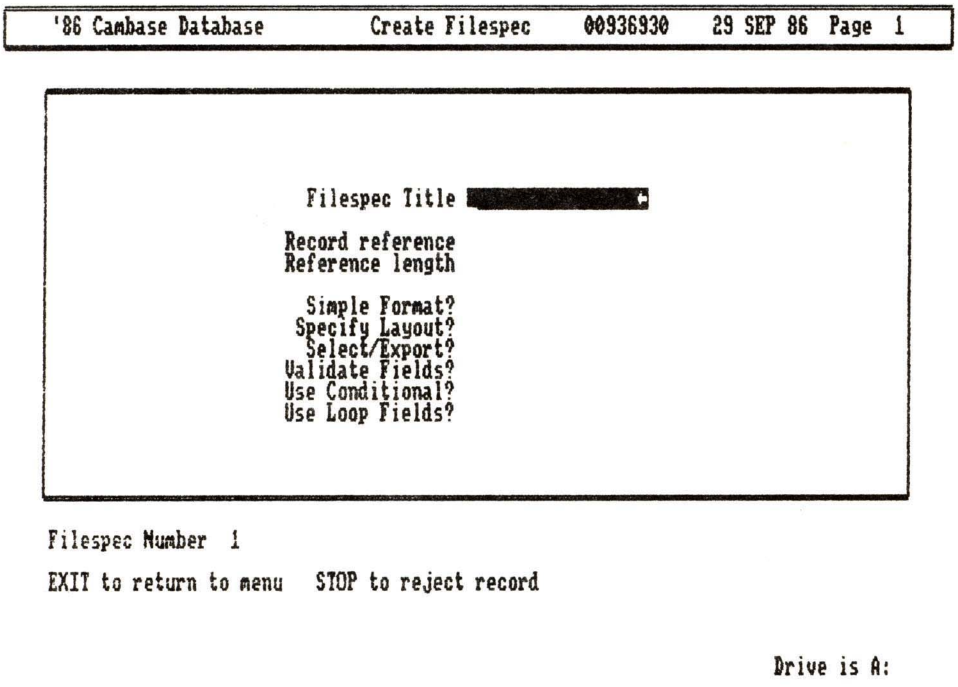
Figure 9.3 *The Update Filespec menu*

structure is similar to the one you wish to create. To start from scratch, just press RETURN.

Next enter a name for the file. This is not a standard 8-character filename; any name can be used, up to 15 characters in length and including spaces. This is simply the name by which the file will be identified. Using our previous example, the name given is SUBSCRIPTIONS.

Each record must have a unique reference number or string to identify it. In the case of the SUBSCRIPTIONS file you can use the customer reference number (Reference). Alternatively, you may just decide to use a simple record number. The maximum length of the record reference must be given.

Next you are asked whether you want to use a simple display format. This results in a straightforward card-type approach, where the fields are determined by a simple choice of types and lengths. While this will suffice for some simple applications, there are other possibilities provided by Cambase that are well worth using. Replying N to the simple format query results in these questions:



```
 '86 Cambase Database        Create Filespec    00936930    29 SEP 86  Page  1

                         Filespec Title

                       Record reference
                       Reference length

                         Simple Format?
                        Specify Layout?
                         Select/Export?
                       Validate Fields?
                       Use Conditional?
                       Use Loop Fields?

 Filespec Number  1

 EXIT to return to menu   STOP to reject record

                                                           Drive is A:
```

Figure 9.4 *Creating a file*

Specify layout?	Answer Y if you want to determine the position of each field on the display
Select/Export?	Answer Y if there are fields that will *not* be used in selections or will not be exported to another program
Validate fields?	Answer Y to introduce validation and range checks
Use Conditional?	Answer Y if there are fields that are entered only if some other field has a given value
Use Loop fields?	Answer Y if there are fields that can take more than one value, i.e. with more than one element

Defining files
Following this you must determine the fields for each record. Up to 39 fields are allowed for any file. The field type must be selected from the following:

C	Text (character field)
N	Numeric field
D	Date field (in Cambase's format)
Y	Yes/No (can only take values Y or N)

The range of field types is rather limited, but the Yes/No field type is a useful addition. This can be linked to a conditional field, where an entry would be made only if the value of the previous field is Y, for example.

A field prompt should be entered. This appears on the screen layout next to the field value and doubles as a field name.

With the exception of the character count and the number of decimal places the remaining attributes are only shown if the appropriate entry on the main file definition was given a Y answer:

Selection?	Generally this will be Y to allow selection or sorting on the field.
Export?	This is usually also Y to allow a field to be exported.
Character count	The size of text fields.
Code entry?	Enter Y if only a limited number of characters can be entered for a text field, and then specify the codes that are valid.
Decimal places	The number of decimal places for a numeric field. If Y was given for field validation, then enter the minimum and maximum values that can be taken for that field. Any decimal places should be included, but the decimal point must be ignored. This is an unusual method of entry and requires careful thought. For example, for two decimal places and a range of 85–90, the minimum and maximum are entered as 8500 and 9000.
0 Delete?	Answer Y if you want to ensure that the record can only be deleted when the value of this field is 0.
Conditional?	If you enter Y here then whether or not an entry is made in each record depends upon the value of some other field. You must give the number of the field that is to determine whether or not this field is to be entered (the test field must be a text or Yes/No field). Also give the characters that are to be searched for; only the first character of a text field is considered.

	Finally state Y for True? if the field is to be used when a match is found, N if the field is to be used when no match is found.
In loop?	If a field is to have several entries, answer Y. In this case you must also state the first and last fields in the loop and the number of times the loop is repeated. The same answers must be given for all fields in the loop.
Loop in prompt?	Answer Y if you want the loop number to appear as part of the prompt.
Layout line	The display line number on which the field appears. There are 18 lines per screen, so numbers greater than 18 indicate a position on subsequent pages.
Prompt column	The position of the start of the prompt text. Up to 80 columns are available, numbered from 1 on the left of the screen.
Input column	The position of the start of the actual field value. Cambase automatically checks that this does not overlap any other part of the display.
Loop adj.	If the field is part of a loop, successive entries should be placed to the right of the first and this attribute determines the distance between start positions. For example, you could enter values every 10 columns. The program counts over the end of the line to the start of the next so you can put all the values in a column by specifying an adjustment of 80.

On completing the details for the last field, enter E for the type. When editing the file definition on a subsequent occasion you can also enter X to delete a particular field.

Example file The fields in the SUBSCRIPTIONS example could be:

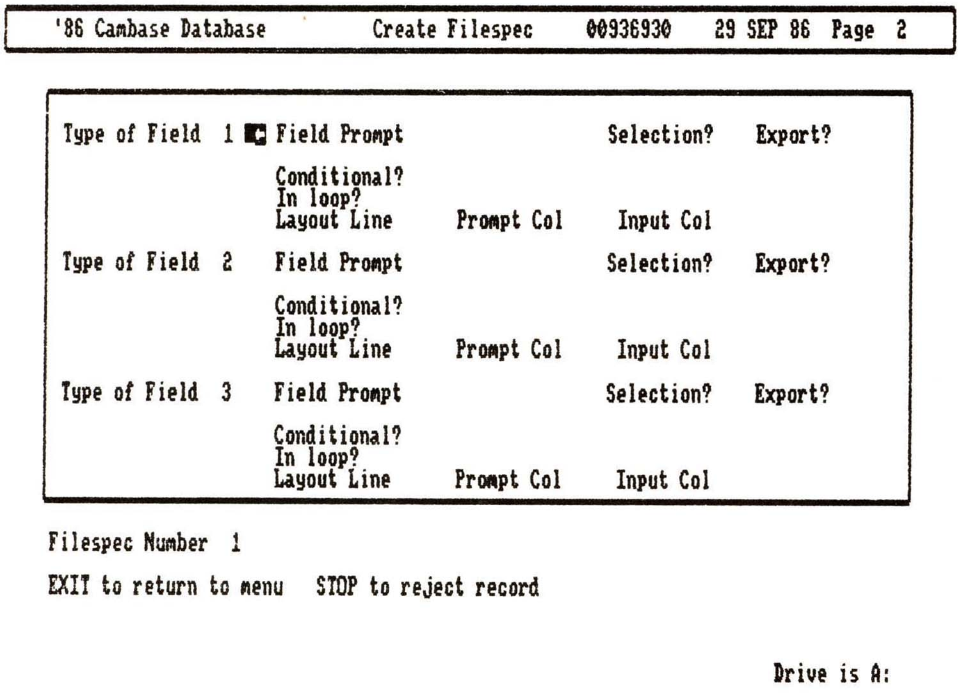
No.	*Type*	*Prompt*	*Selection?*	*Export?*	*Count*	*Code?*	*Valid*	*Decimal places*	*Min*	*Max*	*0Del?*
1	C	Reference	Y	Y	4	N					
2	C	Company	Y	Y	30	N					
3	C	Address	Y	Y	20	N					
4	C	Postcode	Y	Y	7	N					
5	C	Telephone	Y	Y	10	N					
6	C	Extension	N	Y	4	N					
7	C	Contact	Y	Y	30	N					
8	Y	Previous subs?	Y	Y							
9	N	Subscriptions	N	N				2	1000	7000	N
10	D	Renewal dates	N	N							
11	Y	Overdue?	Y	Y							
12	N	Amount owed	Y	Y				2	1000	7000	Y
13	D	Date due	Y	Y							

No.	*Conditional?*	*Field*	*Being*	*True?*	*Loop?*	*From*	*To*	*Max*	*Prompt?*
1									
2									
3					Y	3	3	4	Y
4									
5									
6									
7									
8									
9	Y	8	Y	Y	Y	9	10	10	Y
10	Y	8	Y	Y	Y	9	10	10	Y
11									
12	Y	11	Y	Y					
13	Y	11	Y	Y					

No.	*Line*	*Prompt*	*Input*	*Loop Adj*
1	1	10	22	
2	3	12	22	
3	4	12	22	80
4	8	11	22	
5	10	10	22	
6	10	35	47	
7	11	12	22	
8	15	10	25	
9	20	10	27	80
10	20	35	51	80
11	17	10	19	
12	17	25	40	
13	18	28	40	

Testing the file structure

You should next check that the structure follows all the rules laid down regarding total length (for which the maximum is 2048 characters) and the



```
 '86 Cambase Database         Create Filespec    00936930    29 SEP 86  Page  2

   Type of Field  1 C Field Prompt                 Selection?    Export?
                      Conditional?
                      In loop?
                      Layout Line      Prompt Col     Input Col
   Type of Field  2   Field Prompt                 Selection?    Export?
                      Conditional?
                      In loop?
                      Layout Line      Prompt Col     Input Col
   Type of Field  3   Field Prompt                 Selection?    Export?
                      Conditional?
                      In loop?
                      Layout Line      Prompt Col     Input Col

  Filespec Number  1
  EXIT to return to menu   STOP to reject record

                                                              Drive is A:
```

Figure 9.5 *Defining the fields*

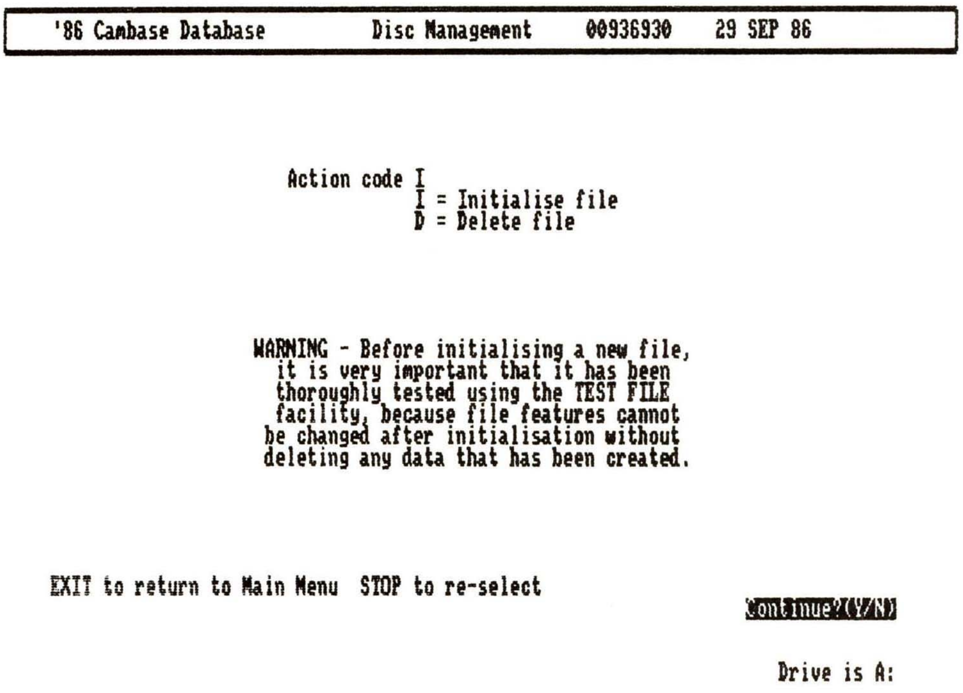
way in which fields are used. For example, a field can only be used in a single loop.

You can do this by selecting the *Test Filespec* option from the main menu. After entering the number of the file, the program carries out various operations and displays suitable messages if any errors are found. If there are errors then you need to return to the file definition stage and make the necessary changes.

After the main tests have been satisfactorily completed, a sub-menu is displayed and you can perform several operations (such as creating and editing records) on a test file that is temporarily created for this purpose.

File initialisation

If all is well you can go on to choose the *Disc Management* option. Two choices are available: I to *initialise* the data file, D to *delete* an existing file. Initialisation involves the setting up on the disk of sufficient records for the file. Once this is done you cannot make any structural changes to your file definition without losing all data that has been entered. Therefore, by this



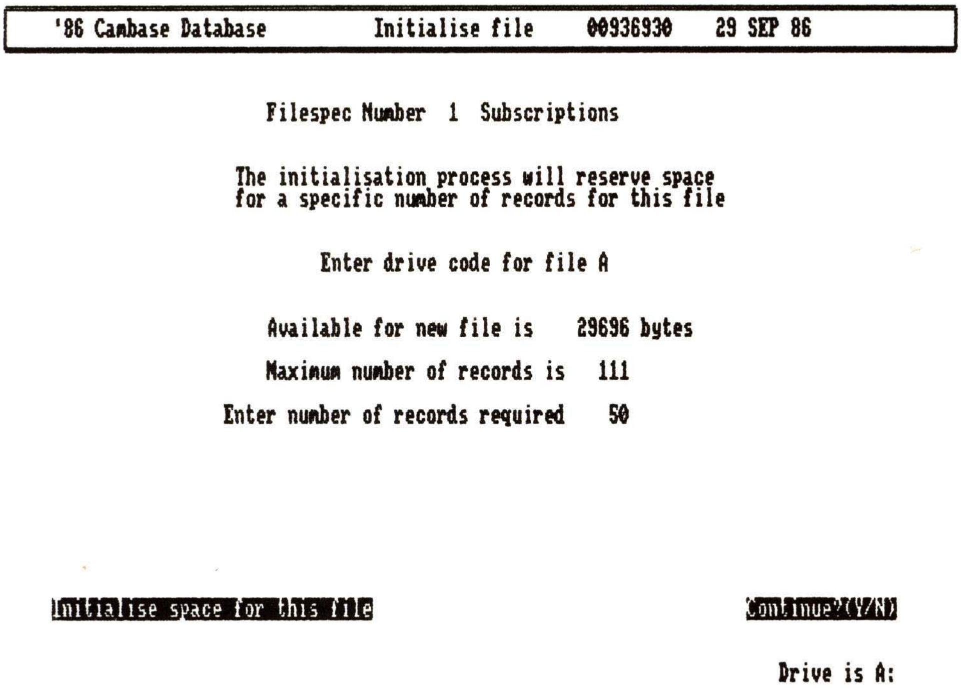
```
'86 Cambase Database          Disc Management     00936930     29 SEP 86


                    Action code I
                                I = Initialise file
                                D = Delete file


                 WARNING - Before initialising a new file,
                   it is very important that it has been
                   thoroughly tested using the TEST FILE
                   facility, because file features cannot
                  be changed after initialisation without
                  deleting any data that has been created.


  EXIT to return to Main Menu  STOP to re-select
                                                          Continue?(Y/N)

                                                             Drive is A:
```

Figure 9.6 *Initialising a file*

stage you should have thorougly tested the file to be certain that it is what you need.

You must specify the file number, the drive containing the data disk and the number of records for the file. The minimum number is two and the program calculates the maximum number that can fit on the disk. Having to do this is rather a drawback, since it means you have to decide how many records you think the file will need, often a very tricky judgement to make. (If you do get it wrong and run out of blank records you can always create a new file on another disk, copying the previous file's structure, set up a greater number of records and transfer all existing data; this is feasible but to be avoided where possible by making allowance for possible future expansions.)

The blank file is then created on disk, following confirmation that you are ready for this action to be taken.



```
 '86 Cambase Database          Initialise file    00936930    29 SEP 86

                    Filespec Number  1  Subscriptions

                The initialisation process will reserve space
                for a specific number of records for this file

                      Enter drive code for file A

                   Available for new file is    29696 bytes

                   Maximum number of records is   111

                 Enter number of records required   50



  Initialise space for this file                          Continue?(Y/N)

                                                           Drive is A:
```

Figure 9.7 *Setting the file size*

Using a data file

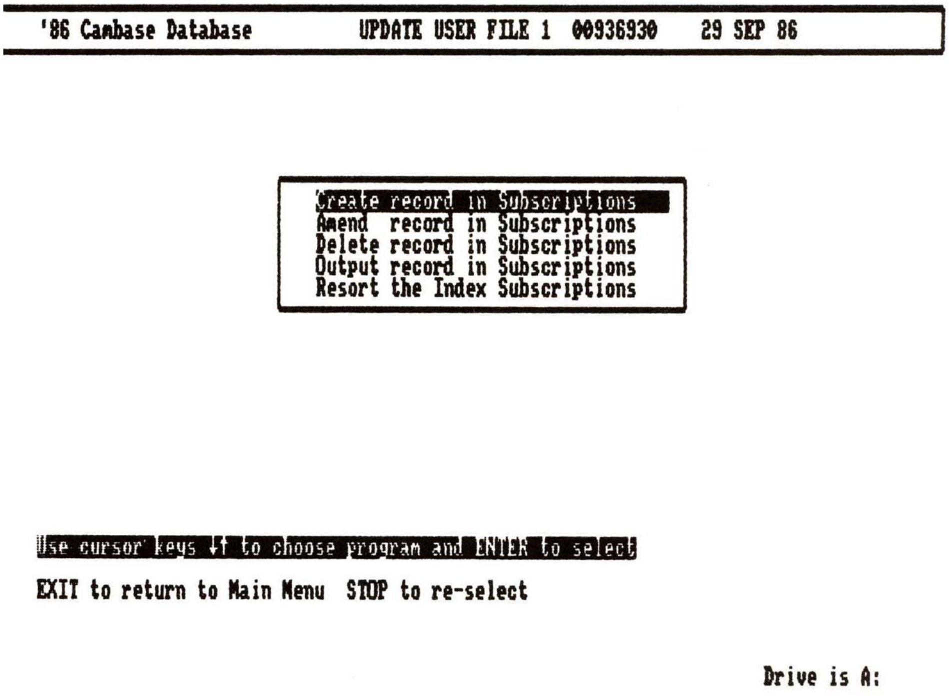
You need to return to the main menu once the file structure has been set up. A variety of options are then available.

Entering data

To enter new records select *Update User Files* from the main menu, enter the file number and choose *Create Record* from the sub-menu. For each record, enter a unique reference. To save yourself time, when consecutive records contain a lot of similar details, each record can be a copy from a previous record. Validation and range checks are performed on each field where they have been specified. Wherever a range has been set it can be displayed by entering a ?. Fields such as dates are always validated by the program.

Inspecting and editing records

The second option from the sub-menu allows you to choose a record to inspect or change by giving its reference. References are used a great deal so



```
'86 Cambase Database          UPDATE USER FILE 1  00936930    29 SEP 86

                    Create record in Subscriptions
                    Amend  record in Subscriptions
                    Delete record in Subscriptions
                    Output record in Subscriptions
                    Resort the Index Subscriptions

Use cursor keys ↓↑ to choose program and ENTER to select
EXIT to return to Main Menu  STOP to re-select

                                                Drive is A:
```

Figure 9.8 *The Update User File menu*

should be kept as short as possible; it is much easier to enter a record number rather than a complete company name.

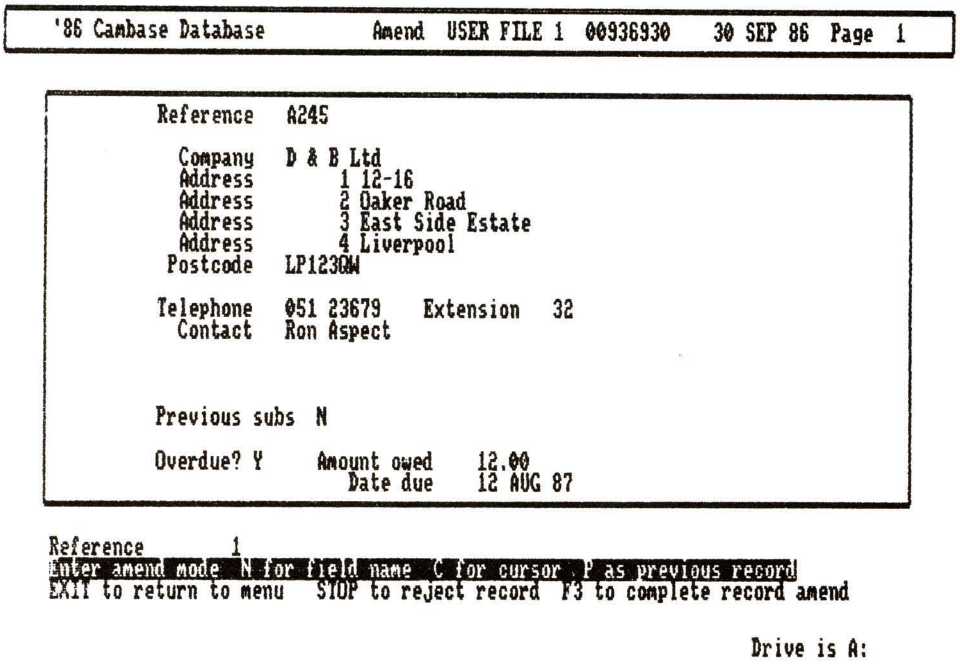
When the record is displayed you can choose to make changes by moving the cursor to the appropriate fields by pressing C (the easiest approach); or you can enter N to give the field name. An abbreviation of the name will do, but even so this is quite a clumsy way of making changes.

The third option in the sub-menu allows the deletion of specified records.

Searching, selection and sorting

The processes of selecting records according to given criteria and of sorting a file into a different order are performed through the *Output record* option in the sub-menu.

All records are indexed in alphabetical order according to the reference code



```
 '86 Cambase Database          Amend  USER FILE 1  00936930    30 SEP 86  Page  1

            Reference   A245

              Company   D & B Ltd
              Address        1 12-16
              Address        2 Oaker Road
              Address        3 East Side Estate
              Address        4 Liverpool
             Postcode   LP123QW

            Telephone   051 23679    Extension    32
              Contact   Ron Aspect


            Previous subs  N

            Overdue? Y     Amount owed     12.00
                              Date due     12 AUG 87

Reference        1
Enter amend mode  N for field name  C for cursor , P as previous record
EXIT to return to menu   STOP to reject record  F3 to complete record amend

                                                               Drive is A:
```

Figure 9.9 *Adding a record*

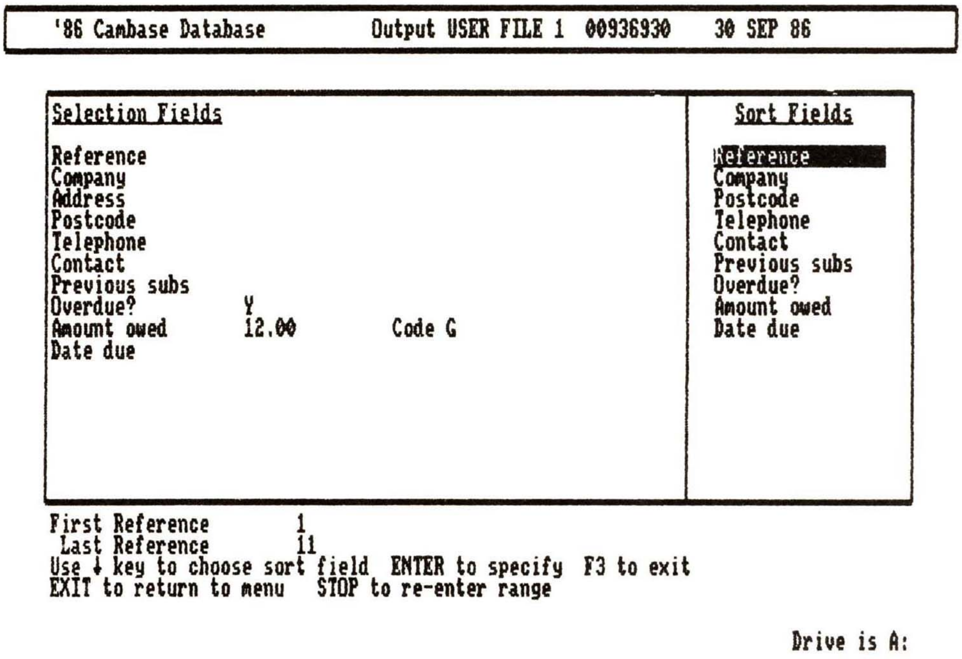
given them and will appear in this order unless specified otherwise. Therefore, if referencing by record number, make sure that small numbers are padded with zeros or spaces. Each reference is compared character by character rather then numerically. If numbers are not padded the order of records will be:

1, 10, 100, 101, 102, . . ., 109, 11, 110, 111, . . ., 119, 12, . . .

For the records to be in the correct order they must be numbered 001, 002, 003, etc. You should state the first and last records to be output, by giving their references, or press RETURN in each case for the whole file.

Selection
Conditions can be entered for each field to determine which records are selected. All fields are listed and any one can be highlighted with the cursor keys. Press RETURN and a condition can be given. The conditions available are somewhat limited. For text fields a single string can be entered. If it is only part of the field then you should enclose it in one or more asterisks (in the same way as square brackets are used in Retrieve).

After specifying a numeric field and entering a value you must decide



```
 '86 Cambase Database          Output USER FILE 1  00936930    30 SEP 86

Selection Fields                                              Sort Fields

Reference                                                    Reference
Company                                                      Company
Address                                                      Postcode
Postcode                                                     Telephone
Telephone                                                    Contact
Contact                                                      Previous subs
Previous subs                                                Overdue?
Overdue?           Y                                         Amount owed
Amount owed        12.00         Code G                      Date due
Date due

First Reference         1
 Last Reference          11
Use ↓ key to choose sort field  ENTER to specify  F3 to exit
EXIT to return to menu   STOP to re-enter range

                                                             Drive is A:
```

Figure 9.10 *Selection criteria*

whether to look for the exact value (by pressing RETURN), a value that is greater (G) or less (L), or a range (R). In this last case you need to enter the upper value for the range, the first value being taken as the lower end of the range.

Sorting

After entering sufficient search conditions, press F3 and you can select the field for sorting. Only one field can be used for the sort. Only the first six characters of a text field are used. The values used in the sort are all stored in memory, so for large files the memory requirement can be quite high. This is obviously less if the number of records have been reduced by selections.

The program calculates the maximum number of records it can sort and displays this number on the screen if it is less than the total number of records in the file. It then asks if you wish to proceed. You need to estimate how many records will be selected. If this is less than the number given, the sort should be successful; otherwise you must either reduce the range of records or increase the selection criteria.

Finally, you need to press D to display the records found, P to print or E to export to a data file.

Reports

Cambase is unusual in that it has no special reporting facilities. The choice of records to print, including selection criteria and ordering, are the same regardless of whether the file is to be displayed or printed. The format used in printing is identical to that of the screen display with one important exception; any blank lines are ignored to save space on the printout. The relative positions across the page are the same as for the display. No allowance is made for paper with more than 80 characters per line.

It is possible to design new print layouts and to print labels. One method is to redefine the file, using *Update Filespec,* so that a new format is supplied. The other approach is to use the programming capabilities of Cambase, as described below.

Using the printer

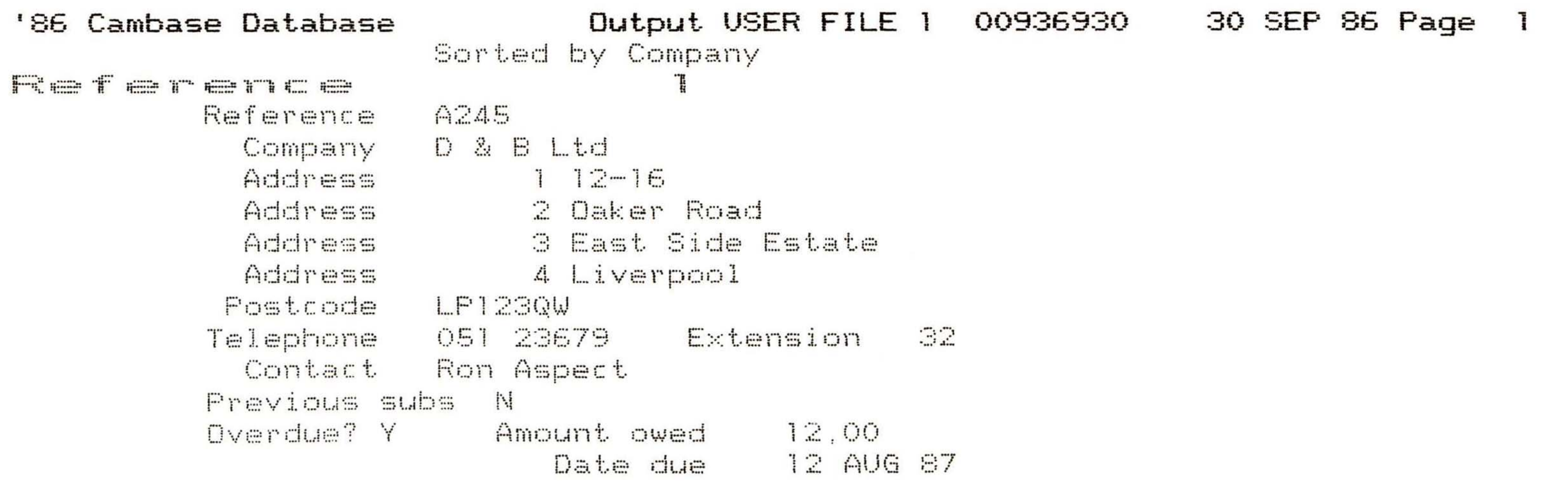
Cambase is one of the few programs available that allows you to make use of your printer's special typeface capabilities. You cannot vary the data format - the features such as baud rate and parity must be set before you enter the program - but you can use features such as bold type, italics and condensed printing.

Each different typeface is usually put into effect by sending a special sequence of *control* characters to the printer. The characters represent the 'unprintable' keys, such as EXIT, ALT and RETURN. For example, to make an Amstrad printer start printing in bold type you need to send the codes for ESCAPE E. ESCAPE is the function associated with the EXIT key on the Amstrad; it is one of the standard ASCII codes. An ASCII conversion table is given in Appendix 3.

Your printer manual should supply the codes needed for any variations in typeface that it can generate. (It is no good trying to send a code for italics if your printer is incapable of such a style.)

The codes may appear in the manual as the actual characters, e.g. ESCAPE E, the corresponding ASCII codes, e.g. 27 and 69 or their *hexadecimal* equivalents, e.g. 1B and 45. Hexadecimal is a means of representing numbers in a form that can be converted to the *binary* code (strings of 0s and 1s) that is used by the computer. Various conventions are used for representing hexadecimal values, such as '1B, '45 or 1Bh, 45h. Hexadecimal values can be converted to their ASCII or character equivalents using the table in Appendix 3.

To change the codes used by Cambase (which are initially suitable for the Amstrad printer), press F3 when at the main menu. For some reason this is only effective when you first start the program and not when you return



```
'86 Cambase Database           Output USER FILE 1  00936930    30 SEP 86 Page  1
                          Sorted by Company
Reference                                1
          Reference   A245
            Company   D & B Ltd
            Address        1 12-16
            Address        2 Oaker Road
            Address        3 East Side Estate
            Address        4 Liverpool
           Postcode   LP123QW
          Telephone   051 23679    Extension   32
            Contact   Ron Aspect
          Previous subs  N
          Overdue? Y     Amount owed    12,00
                            Date due    12 AUG 87
```

Figure 9.11 *A printed record*

from some other menu. Also, it only works if the program is run from drive A and not if it is run from drive M. Therefore if you do want to change the printer codes (for a single-drive system) you should run the program from drive A, make the changes, and exit. Cambase can then be run from drive M as usual. The decimal codes currently in use are displayed and new values can be set for each. The codes available are:

Printer Initialisation	Codes needed to set the page length and other initial details.
Expanded print	Two codes, one to start printing in expanded (wide) characters, the other to return to normal type.
Bold print	Two codes to turn bold - or emphasised - type on and off.

With a little bit of trickery it is quite easy to persuade the program to work with other type styles, such as italics or condensed print. If you enter the code for turning italics on when asked for Expanded Print On, and the code to turn italics off for Expanded Print Off, the program will not be aware of the fact. Whenever it tries to print in expanded print it will send the code to turn on italics and the text will appear italicised. In this way you can select any two type styles instead of bold and expanded print.

The Printer Initialisation can be used for a variety of purposes. For example, if you include within it the code for condensed print then the entire printout appears in condensed type; you still have the option to specify two other styles for use within the text. Similarly, you can supply codes to vary the pitch of the characters on each line, the number of lines per inch, the spacing between lines and any other special feature your printer may have.

Unfortunately, there is no way of determining where in the printout the extra styles are used.

Looking after files

There are no instructions in the program for making copies of files or disks so this is best done from CP/M. It is important to realise that Cambase does not use the standard method of storing files. All data is stored in a single file; output text is in another file and so on. The files used include:

SYSTFILE.DAT	System file
FDBFILE.DAT	Details of whole database
FILESPEC.DAT	Details of data file structures

PROCESS.DAT	Processes
NARRFL.DAT	Narratives (see below)

For each data file the following files are held:

IDATAF01.DAT	Data file for file number 1
LDATAF01.DAT	Linked index file
HDATAF01.DAT	'Hashed' index file (for fast access)

When making backups, it is safer to copy the entire data disk, rather than just some of the files, to ensure that you have all the relevant information.

Exporting data
At various stages in the program you can specify that data should be exported to an ASCII file. This file can be edited by a word processor, for example, but cannot be imported back into Cambase.

Programming

Cambase is the first program considered in this book to introduce the idea of procedures and programming. In this case the procedures are referred to as 'processes'.

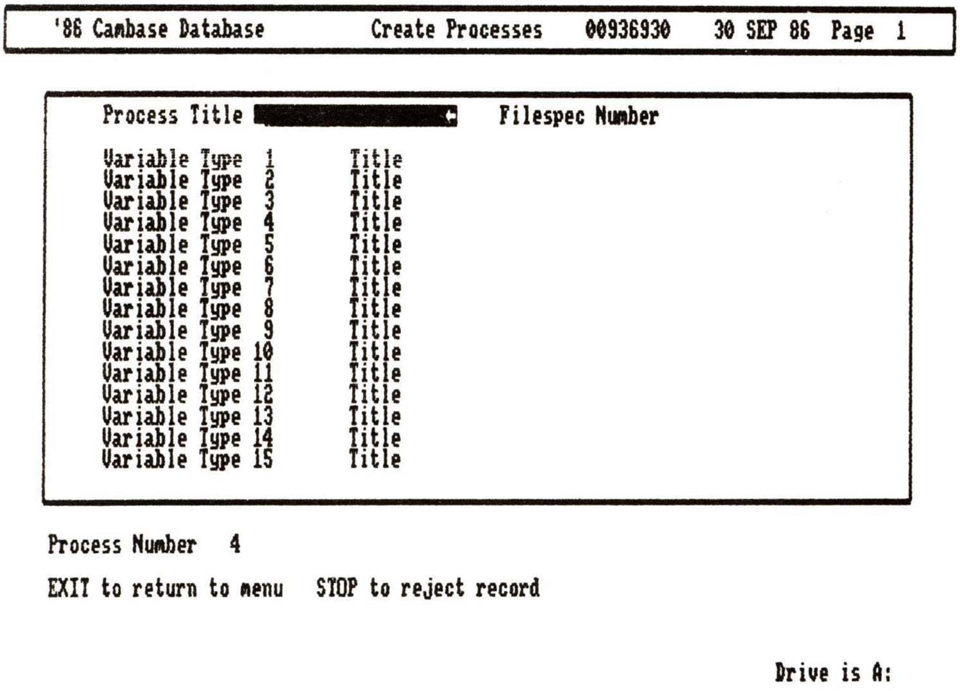
The programming capabilities of Cambase are extremely limited, but they allow the automatic execution of a set of instructions on a number of records, without any further interference by the user. The user can specify the first and last records, and apply the usual selection procedures; the instructions are then carried out on every record selected.

The instructions are supplied through the *Update Process* option. Unlike most programmable databases the procedures do not consist of a list of instructions, but are given in a question-and-answer session.

You must supply a name for the process and identify the file it is to relate to; each process can only be associated with one file.

Variables
When running the process you may want to store information as you go along (for example, totals). This extra data is held in *variables*. A variable is simply a temporary field that holds a single value and exists only as long as the process is running. You can consider the variables to be fields in a special file that has only one record.

You can enter information during the running of a process and display or



```
 '86 Cambase Database           Create Processes   00936930   30 SEP 86  Page  1

    Process Title                  Filespec Number

    Variable Type  1     Title
    Variable Type  2     Title
    Variable Type  3     Title
    Variable Type  4     Title
    Variable Type  5     Title
    Variable Type  6     Title
    Variable Type  7     Title
    Variable Type  8     Title
    Variable Type  9     Title
    Variable Type 10     Title
    Variable Type 11     Title
    Variable Type 12     Title
    Variable Type 13     Title
    Variable Type 14     Title
    Variable Type 15     Title

 Process Number   4

 EXIT to return to menu   STOP to reject record


                                                        Drive is A:
```

Figure 9.12 *Defining a process*

print results. You may also need to print items of additional text, called 'narratives'. These can be stored as entries in a separate file on disk, using the *Update Narratives* option. Up to 64 narratives are allowed per disk, each up to 60 characters long.

A variety of different instructions can therefore be used. Each instruction is given a number and a name. Up to 15 instructions of each type are allowed. The following variables and instructions are available:

Variables	Standard variables for storing any type of data (labelled V1, V2, etc.). Where applicable state the number of characters or decimal places.
Input Item	Values to be entered during the process. You must specify entry to a field or variable, giving the reference number (for example, F2 for the second field or V8 for variable number 8). Also specify whether the input is to be made for each record or just once at the start of the process.

Derived Item	Formulae, consisting of fields (F1, F2, . . .), variables (V1, V2, . . .,) and the symbols + – * /. No brackets are allowed; all calculations are carried out left to right. The result can be placed in a field or a variable.
Output Item	Values to be output, which can be the contents of fields, variables or narratives. You must specify the co-ordinates of the output in terms of the line number and column numbers for field name and data. Up to 40 output items can be used.
Total Item	The process will automatically provide totals and averages for fields or variables; alternatively, specify R for total number of records. Totals are numeric except the following: for character fields, number of non-empty fields, for Yes/No fields number of 'Yes' fields; for date fields total number of days since 1/1/1900.

There is also an option to use the process to print labels. If labels are not printed you can decide:

- Whether to print each record as it appears on the screen or just the Output items.
- Whether to include the record reference and/or standard heading.
- Whether to print each record on a new page.

If labels are to be printed you must choose:

- Number of labels across the page (1 to 10).
- Print position of first label.
- Distance across page from start of one label to start of next (the *offset*).
- Line number for first label.
- Number of lines from top of one label to top of next.

Before running the process use *Update User File* to enter the range to be processed; give any necessary selection and sort instructions. Next use *Run Process* to put the instructions into effect. You must indicate whether the output is to be displayed, printed or stored in a disk file. Enter values for any Input Items that are to apply to the whole file. The process is then carried out for each record that satisfies the range and selection criteria, in the order given by the sort instructions.

The operations carried out on each record are:

- The record is read.
- Input Items are entered.
- Calculations are performed and totals updated (based on the new values of the fields).
- Any changed field values are stored on disk.
- The output is displayed, printed or stored on disk.

When all records have been processed the totals and averages are output.

Example The example given below illustrates the use of Cambase processes, using the SUBSCRIPTIONS file. It considers the case where subscriptions fall due. If the amount outstanding for the previous year has all been paid then the new subscription needs to be entered and the date on which it is due must be updated.

The aim is to select all records where no money is overdue and the date due falls within a specified range (for example, within a particular month). Labels should be printed for all those who are now due for renewal. The label should also show the amount due and the date. It is assumed that all subscriptions have risen by 10%.

The process takes the form:

Variable 1:	Type: Y Prompt: Leap Year? Value: Y for Leap Year, N for non-leap year
Variable 2:	Type: Y Prompt: Continue? Value: Y
Input Item 1:	V1 Whole file?: Y
Input Item 2:	V2 Whole file?: Y
Derived Item 1:	F11 = V2
Derived Item 2:	F12 = F12 * 1.1 (Add 10% to Amount Owed)

Derived Item 3:	F13 = F13 + 365 + I1 (Add 1 year to date)
Label Print?:	Y Labels across: 2 Start at column: 5 Columns offset: 40 Start at line: 1 Lines offset: 8

Output Item		Line	Title col	Data col
1:	F1 (Reference)	1	0	20
2:	F2 (Company)	2	0	1
3:	F3 (Address)	3	0	1
4:	F4 (Postcode)	6	0	1
5:	F12 (Amount owed)	8	0	15
6:	F13 (Date due)	8	0	22

Total Item1:F12 (total amount now due but not yet overdue)

Total Item2:R (number of records)

The records should be selected according to 'Overdue?', taking the value N and 'Date due' being within the required range. There is no need for them to be sorted.

The first input should be Y if the number of days to be added is 366; the second input should alway be Y (this value is placed in field 11, 'Overdue?', replacing the previous N value). Hence, a 'dummy' prompt of 'Continue?' is given to ensure that this is always the case (though the user should be warned to press EXIT rather than N to abandon the process – otherwise, the records would be updated, but a value of N would be placed in all 'Overdue?' fields).

Support

The manual for Cambase is detailed, though slightly confusing in that it contains much information that is general to all Camsoft programs. Examples are relied on to provide instruction on how to use the program; the reference sections only give a brief outline.

The usual support facilities are available and Camsoft do seem genuinely keen to hear from anyone with comments, suggestions, problems and details of unusual applications. Telephone support is free for 90 days, after which there is an annual subscription.

Conclusion

Cambase is a fairly comprehensive program. Its structure is rather complex and unwieldly but it provides some interesting features, such as arranged files and Yes/No fields. Its output facilities are limited, but it includes very flexible labelling routines.

The use of processes for carrying out a standard procedure on all records satisfying a set of conditions is extremely useful for many applications, although the rigid structure limits its effectiveness. Overall, Cambase is a useful and advanced, if somewhat clumsy database program.

CHAPTER 10

CONDOR 1: THE BASIC PRINCIPLES

Condor 1, published by Caxton Software Ltd, is an advanced database, capable of a high level of programming, yet still simple to use.

In this, the first of two chapters that look at the package, the basic principles needed to create, inspect and print files are described, along with the methods available for selection and sorting. The final chapter introduces the programming capabilities of Condor 1.

Initial preparations

In the same way as for all the other programs you should start by preparing a work disk and a sufficient number of data disks. There is very little room on the Condor 1 disk for adding the system file or CP/M utilities. However, PIP and SUBMIT are already included there. Therefore side 1 of the work disk should contain all the necessary CP/M files while side 2 can be a copy of the Condor 1 disk.

If you have a PCW8512 then you may find it useful to copy all the program files to drive M with the command:

```
PIP M:=A:*.*
```

If you have only a single-drive PCW8256 then a great deal of disk swapping can be saved by storing just some of the program files in drive M. In this case not all files will fit into memory so you must copy the relevant ones with a specially prepared batch file. Enter the command:

```
SUBMIT PREP1
```

When this has finished you can replace the program disk with a data disk and enter:

```
SUBMIT PREP2
```

This copies the files that won't fit into memory to the data disk. In a single-drive system you are requested to load the disk for drive B, i.e. the data disk, after the files have been copied into memory from the program disk.

This procedure need only be carried out the first time you use Condor 1.

Starting Condor 1

If Condor 1 is being run from drive A enter the command:

```
DBMS
```

If the program is to be run from drive M load the program disk and enter:

```
SUBMIT CONDOR
```

The program files are copied across to the memory drive and you will be able to replace the program disk with the data disk.

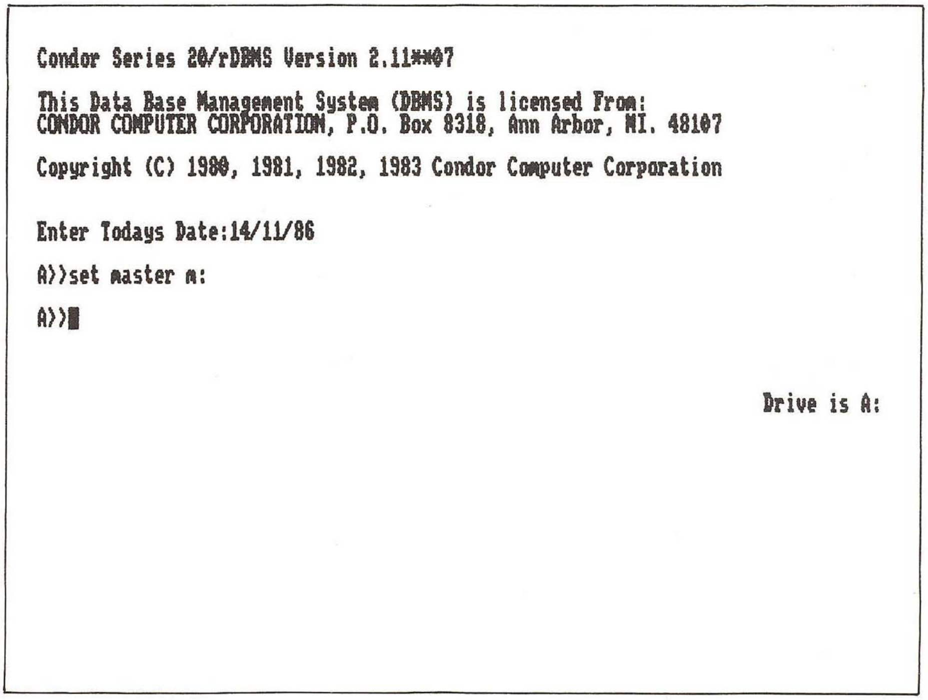
In either case you are asked for the current date (in the standard format). The Condor 1 prompt (A>>) is displayed. You need to specify the data drive, if this is not drive A, by entering:

```
B:
```

Whichever machine you have, you must always tell Condor where to look for the program files with a command in the form:

```
SET MASTER M:
    (single-drive system)
SET MASTER A:
    (double-drive system)
```

Condor 1 is now ready to receive data.



```
Condor Series 20/rDBMS Version 2.11**07

This Data Base Management System (DBMS) is licensed From:
CONDOR COMPUTER CORPORATION, P.O. Box 8318, Ann Arbor, MI. 48107

Copyright (C) 1980, 1981, 1982, 1983 Condor Computer Corporation


Enter Todays Date:14/11/86

A>>set master m:

A>>


                                                        Drive is A:
```

Figure 10.1 *The initial display*

Creating a data file

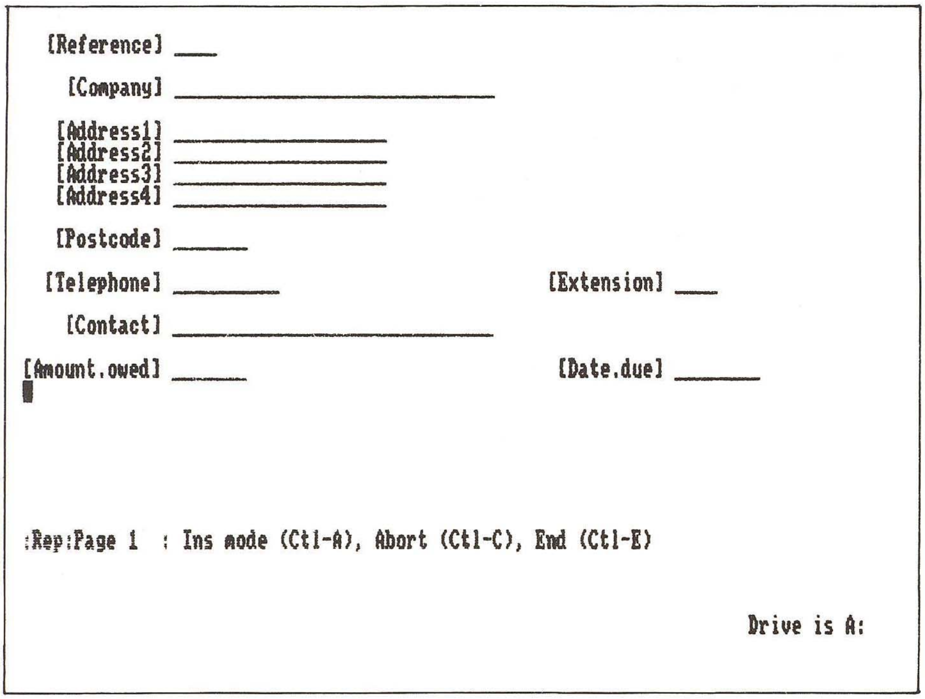
Condor 1 operates through commands typed in at the command line, in the same way as the CP/M operating system, rather than through menus. There is only a limited number of commands but, as for CP/M, each command must be typed accurately and with the correct components.

To define a new data file (or 'dataset') enter a command in the form:

```
DEFINE SUBS
```

This will create a file called SUBS.DAT on the data drive. If the file already exists then the same command can be used to make changes to its structure.

Condor 1 combines the basic field definition and the design of screen layout in a single operation. You are presented with a blank screen onto which you can type any headings, text, boxes, lines or instructions. The cursor is moved with the following keys:



```
      [Reference] ____

        [Company] ______________________________

       [Address1] ____________________
       [Address2] ____________________
       [Address3] ____________________
       [Address4] ____________________

       [Postcode] _______

      [Telephone] __________                        [Extension] ____

        [Contact] ______________________________

    [Amount.owed] _______                            [Date.due] ________
    █




    :Rep:Page 1  : Ins mode (Ctl-A), Abort (Ctl-C), End (Ctl-E)


                                                              Drive is A:
```

Figure 10.2 *Designing the layout*

ALT-H	Left
ALT-J	Down
ALT-K	Up
ALT-L	Right
ALT-R	Top of screen

You can change between insert and overwrite modes by pressing ALT-A.

Fields are defined by entering their names in square brackets. The only limitations on field names are that they may not contain spaces and cannot exceed 15 characters. The names must start with a letter and can contain most characters. Numeric field names should not include numbers or numeric symbols. Following the names (with a gap if you wish) you should enter a string of underscores to specify the position at which the field values will appear. These underscores also define the output size of the field, regardless of type.

The usual editing keys are available. When this part of the program is complete you can either save it on disk by pressing ALT-E or abandon it with ALT-C.

Defining field types
After the display layout has been defined the program requests the field type for each field. This is a very simple procedure and requires no complex preparations. The field types supplied by Condor 1 are:

A	Alphabetic, including spaces and the characters '",.-
AN	Alphanumeric, including all alphabetic and numeric characters and other printable characters
N	Numeric (integer)
N.n	Decimal, with a fixed number of decimal places (for example N.3 for 3 decimal places)
$	Money, with 2 decimal places
J	Date (Julian) in the standard format

For each field you can also add R to the type to indicate that an entry is always required. Following the type you can enter the following details:

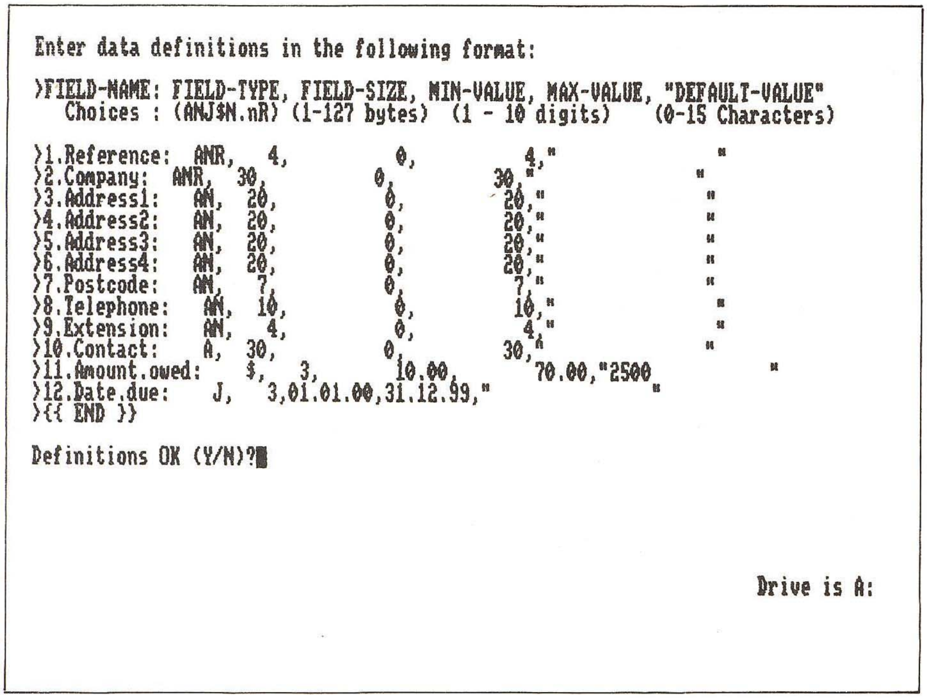
Field-Size	For text fields this is the number of characters, for numeric fields the size in bytes (see below)
Min-value	The minimum value allowed
Max-value	The maximum value allowed
"Default-Value"	The value provided as a default and stored as part of the record unless otherwise specified. The default must be enclosed in quotes even if it is a numeric value

The field size for numeric values will depend upon the range of values likely to be encountered. The program sets a default value unless you specify otherwise. To determine a suitable size yourself, use the following table to calculate the number of bytes from the number of underscores or vice versa:

Underscores for each type/byte combination

		Type		
		N	N.n	$
Bytes	1	1–2		
	2	3–4		4–5
	3	5–6		6–7
	4	7–11		8–12
	8		2–20	

Date fields require 3 bytes



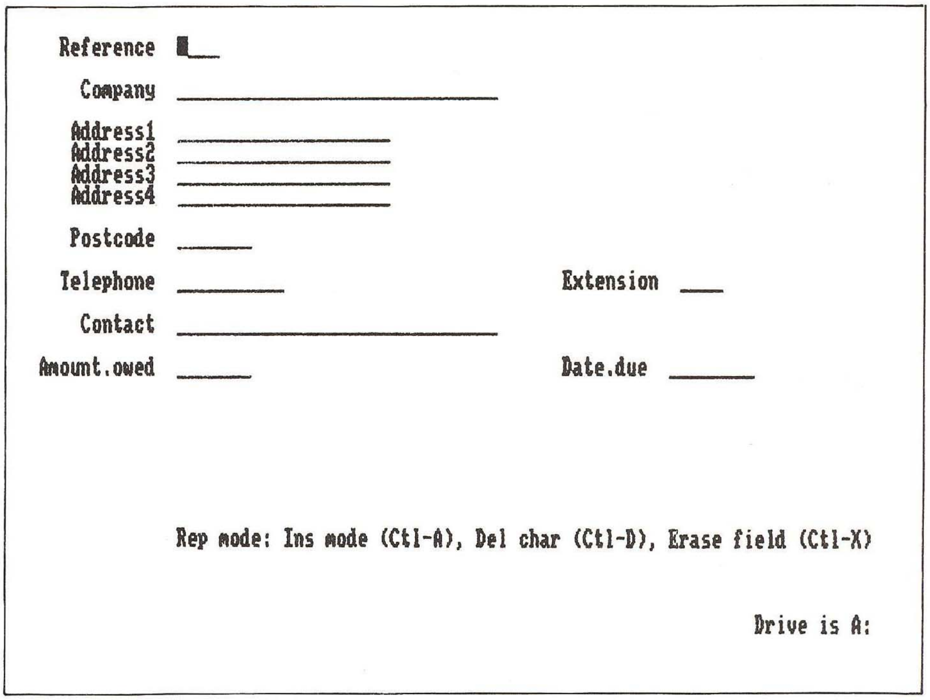
```
Enter data definitions in the following format:

>FIELD-NAME: FIELD-TYPE, FIELD-SIZE, MIN-VALUE, MAX-VALUE, "DEFAULT-VALUE"
   Choices : (ANJ$N.nR) (1-127 bytes)  (1 - 10 digits)    (0-15 Characters)

>1.Reference:  ANR,   4,           0,          4,"              "
>2.Company:  ANR,  30,           0,          30,"             "
>3.Address1:   AN,  20,           0,          20,"             "
>4.Address2:   AN,  20,           0,          20,"             "
>5.Address3:   AN,  20,           0,          20,"             "
>6.Address4:   AN,  20,           0,          20,"             "
>7.Postcode:   AN,   7,           0,           7,"             "
>8.Telephone:   AN,  10,           0,          10,"             "
>9.Extension:   AN,   4,           0,           4,"             "
>10.Contact:    A,  30,           0,          30,"             "
>11.Amount.owed:    $,   3,         10.00,      70.00,"2500        "
>12.Date.due:    J,   3,01.01.00,31.12.99,"          "
>{{ END }}

Definitions OK (Y/N)?█




                                                            Drive is A:
```

Figure 10.3 *Entering field details*

The attributes of the field must be separated by commas. If you just enter the type code the program supplies suitable default values for the other attributes.

For our example SUBS file, the fields can be:

Name	*Underscores*	*Type*
Reference	4	ANR
Company	30	ANR
Address1	20	AN
Address2	20	AN
Address3	20	AN
Address4	20	AN
Postcode	7	AN
Telephone	10	AN
Extension	4	AN
Contact	30	A
Amount.owed	7	$,3,10,70,25
Date.due	8	J



```
    Reference ▮__

      Company ______________________________

     Address1 ____________________
     Address2 ____________________
     Address3 ____________________
     Address4 ____________________

     Postcode _______

    Telephone __________                    Extension ____

      Contact ______________________________

  Amount.owed _______                       Date.due ________



              Rep mode: Ins mode (Ctl-A), Del char (Ctl-D), Erase field (Ctl-X)

                                                                 Drive is A:
```

Figure 10.4 *A blank record*

After entering the details you have the chance to correct them. You are asked if you want to create an index, to which you should reply N; this option is apparently not available in Condor 1.

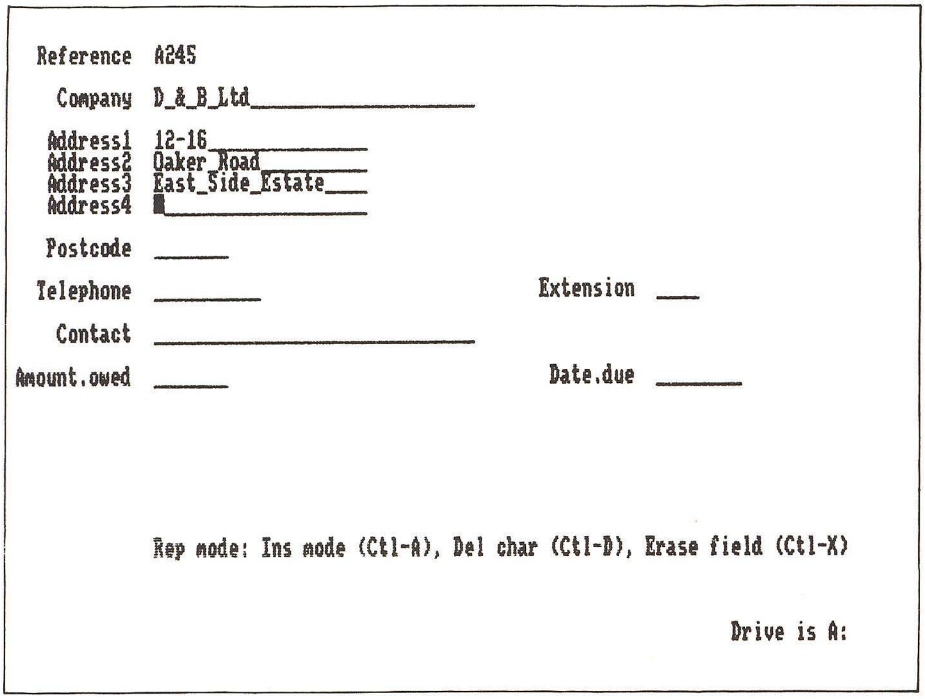
At any time you can redefine the file by adding or deleting fields, using the REORG command. The file is reorganised as a temporary file called RESULT and then, if this is satisfactory, it replaces the original. The FORMAT command is used to change field names and output sizes.

Entering data

Data can be entered by giving the command:

```
ENTER SUBS
```

A blank form is presented and can be filled in the usual way. The ALT keys move the cursor around the screen and the DEL key can be used to delete characters.



```
   Reference  A245

     Company  D_&_B_Ltd____________________

    Address1  12-16_______________
    Address2  Oaker_Road__________
    Address3  East_Side_Estate____
    Address4  ____________________

    Postcode  _______

   Telephone  __________                      Extension  ____

     Contact  ____________________________

 Amount.owed  _______                          Date.due  ________




              Rep mode: Ins mode (Ctl-A), Del char (Ctl-D), Erase field (Ctl-X)


                                                             Drive is A:
```

Figure 10.5 *Entering data*

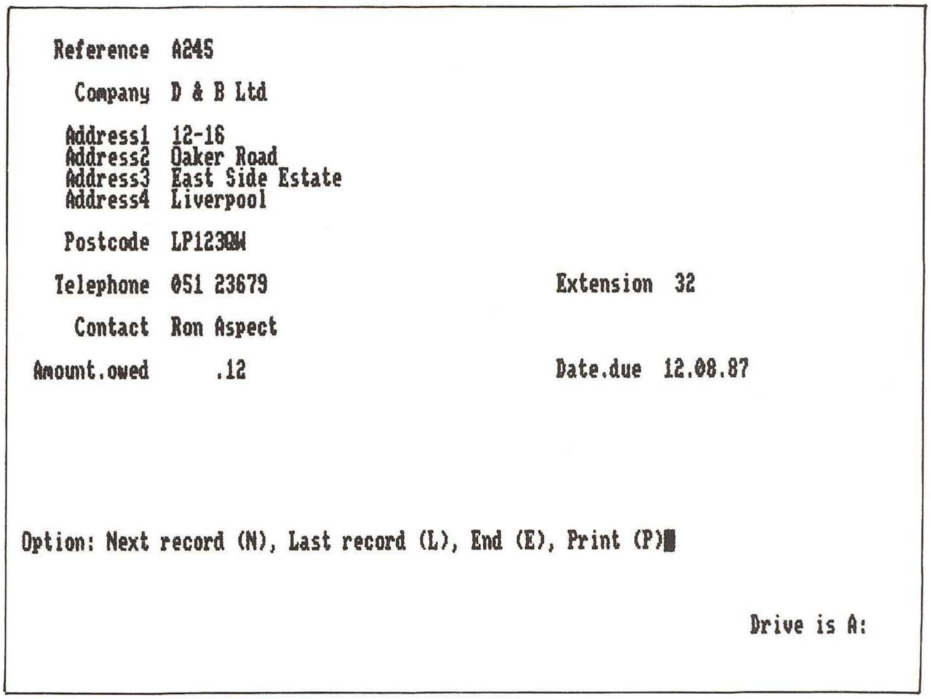
If the original ENTER command has an [F] parameter added then fields are automatically assigned their default value, if any. This value can be overridden by pressing ALT-K. To use each record as a basis for the next, use [R] at the end of the command.

After each record there are a number of options, each one of which is selected by pressing the appropriate key:

Abort	End entry of records and do not save current record
Continue	Save record and enter next record
Delete	Delete current record and re-enter
End	Save record and return to main prompt
Print	Save record, print it and continue with next record
Revise	Make changes to current record

On completion of the file you are returned to the A>> prompt.

Inspecting records
You can look at the records in the order in which they were entered with the



```
    Reference  A245

      Company  D & B Ltd

     Address1  12-16
     Address2  Oaker Road
     Address3  East Side Estate
     Address4  Liverpool

     Postcode  LP123QW

    Telephone  051 23679                        Extension  32

      Contact  Ron Aspect

  Amount.owed       .12                         Date.due  12.08.87




 Option: Next record (N), Last record (L), End (E), Print (P)

                                                          Drive is A:
```

Figure 10.6 *Inspecting a record*

command:

```
LIST SUBS
```

After each record has been displayed, you can move to the next record (by pressing N), move to the last record (L), print the record (P), or end this part of the program (E).

You can also display summaries of the data on the screen by specifying field names after the LIST command:

```
LIST SUBS BY REFERENCE COMPANY AMOUNT.OWED DATE.DUE
```

In this case the program produces a columnar listing which can be interrupted and restarted with the STOP key. (Note the similarities between this command and the corresponding Retrieve commands.)

Editing records
To make changes to the records enter the command:

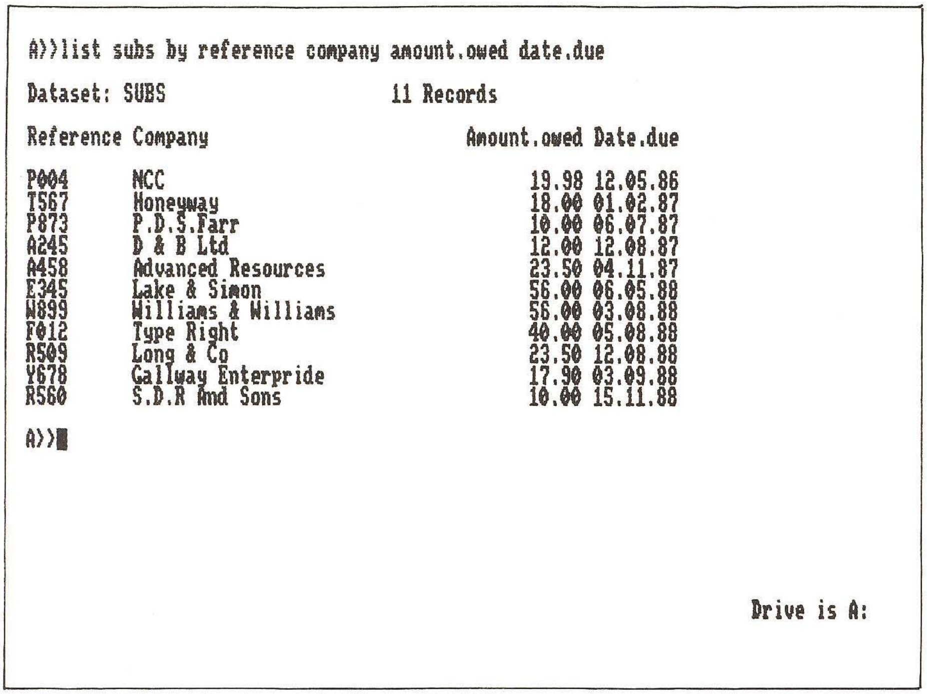

```
A>>list subs by reference company amount.owed date.due

Dataset: SUBS                           11 Records

Reference Company                              Amount.owed Date.due

P004      NCC                                        19.98 12.05.86
T567      Honeyway                                   18.00 01.02.87
P873      P.D.S.Farr                                 10.00 06.07.87
A245      D & B Ltd                                  12.00 12.08.87
A458      Advanced Resources                         23.50 04.11.87
E345      Lake & Simon                               56.00 06.05.88
W899      Williams & Williams                        56.00 03.08.88
F012      Type Right                                 40.00 05.08.88
R509      Long & Co                                  23.50 12.08.88
Y678      Gallway Enterpride                         17.90 03.09.88
R560      S.D.R And Sons                             10.00 15.11.88

A>>█

                                                                  Drive is A:
```

Figure 10.7 *A file in the list format*

UPDATE SUBS

The program works through each record in turn. You can revise (edit) the record, delete it, make no change (N) or press E to end. If you decide to edit you can move around the screen in the usual way and press ALT-E to accept the changes. If you only want to change certain records then you must use the selection criteria described below.

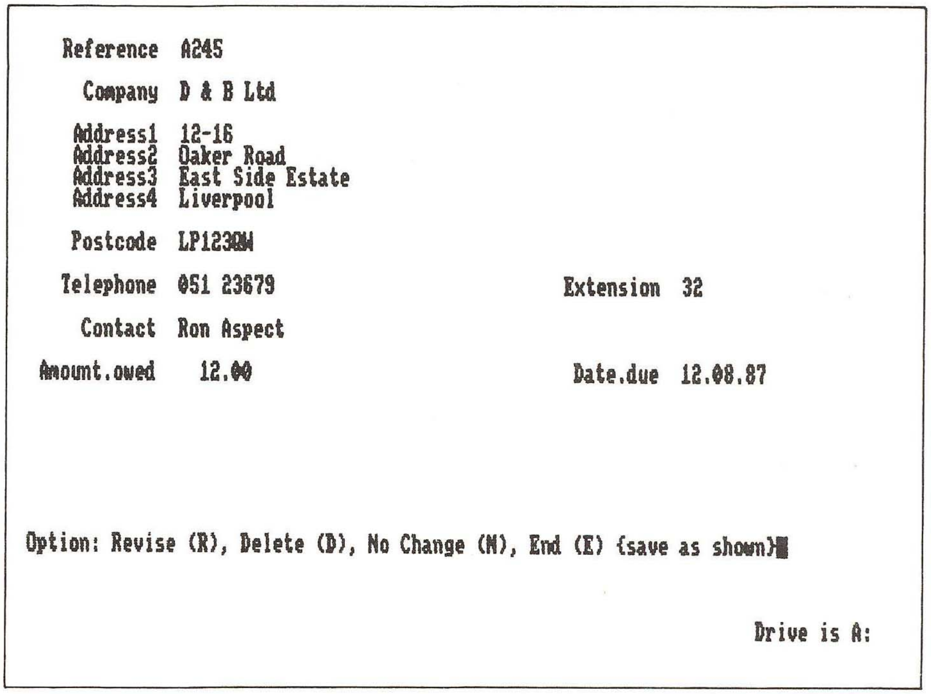
Searching, selection and sorting

To search for particular records or select groups of records, either for viewing or for editing, you can use a variation of the UPDATE command. The command must be followed by a condition in the form:

WHERE (field) = (value)

For example to select all companies in Kent enter the command:

UPDATE SUBS WHERE ADDRESS4 = KENT



```
     Reference  A245

       Company  D & B Ltd

      Address1  12-16
      Address2  Oaker Road
      Address3  East Side Estate
      Address4  Liverpool

      Postcode  LP123QM

     Telephone  051 23679                        Extension  32

       Contact  Ron Aspect

   Amount.owed    12.00                          Date.due  12.08.87



 Option: Revise (R), Delete (D), No Change (N), End (E) {save as shown}█

                                                              Drive is A:
```

Figure 10.8 *Selecting records*

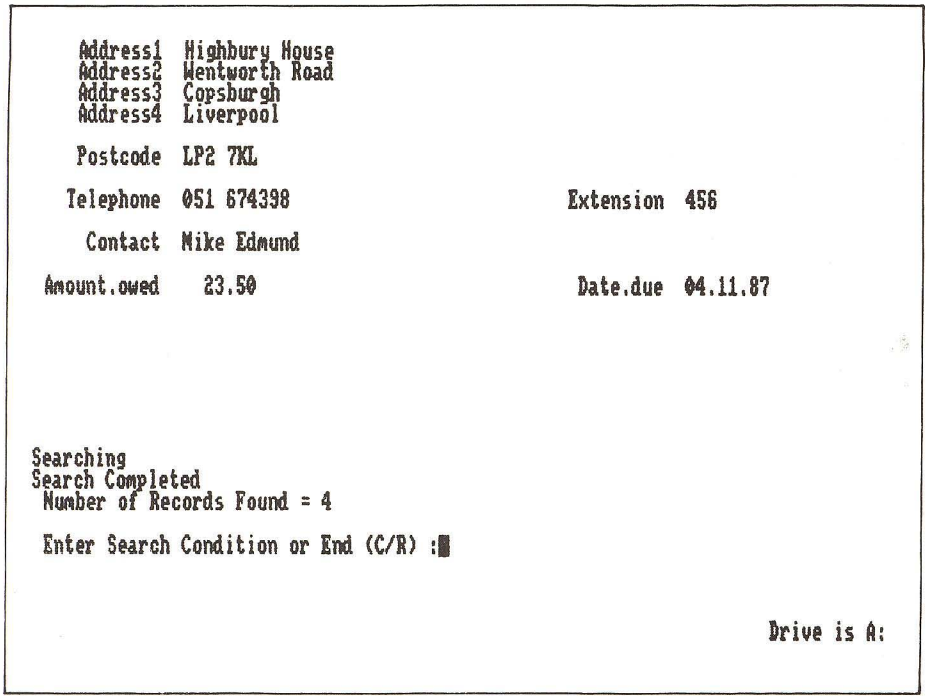
Note that you could equally well have used IS, ARE, EQ or EQUALS in place of the = symbol. Condor 1 is very flexible in this way, but unless you stick to one convention it can become confusing.

Each record displayed can be edited. After all records have been scanned you are shown the number of records found and requested to enter a new condition; this time only the part after 'WHERE' should be entered. For example:

```
DATE.DUE = 01/01/87
```

To end this part of the program, press RETURN without entering a condition.

To select a group of records, rather than displaying them, use SELECT instead of UPDATE. The form of the commands is identical. This time the program creates a new file on disk called RESULT. This file is only held temporarily but while it exists its structure is identical to that of the original file. The only difference is that it contains only those records that match the criteria given.



```
        Address1  Highbury House
        Address2  Wentworth Road
        Address3  Copsburgh
        Address4  Liverpool

        Postcode  LP2 7KL

       Telephone  051 674398                      Extension  456

         Contact  Mike Edmund

     Amount.owed    23.50                          Date.due  04.11.87




    Searching
    Search Completed
     Number of Records Found = 4

     Enter Search Condition or End (C/R) :


                                                         Drive is A:
```

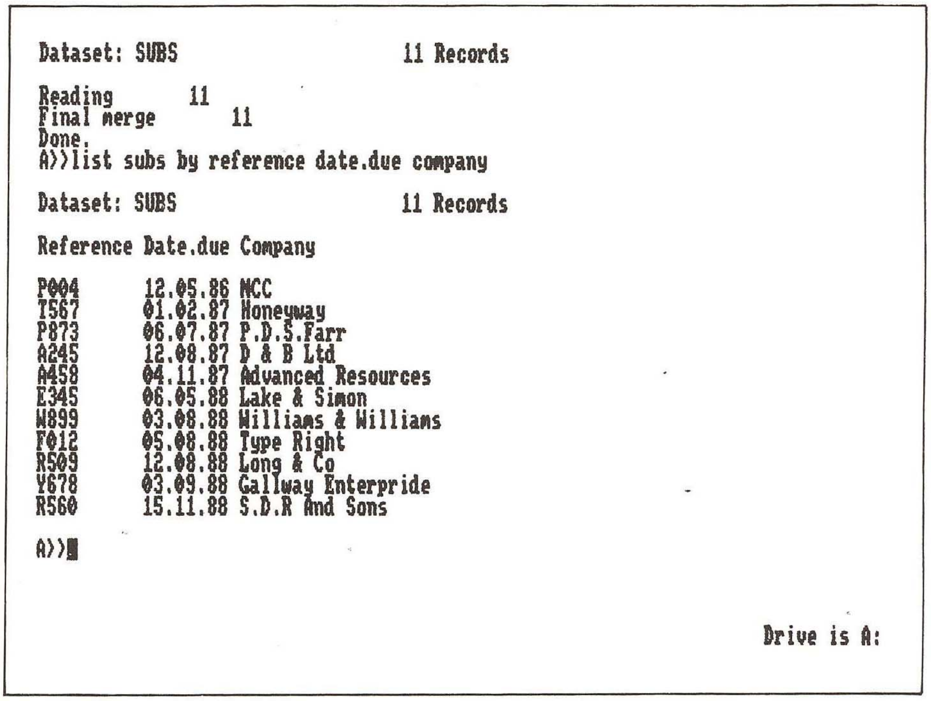
Figure 10.9 *Completing the selection*

This temporary file can be listed or updated as if it were a normal data file. To store it permanently on disk use the SAVE command, giving the file a new name. For example, after selecting all records with a Date.due value before a given date, these can be saved in the file OVERDUE.DAT with the command:

```
SAVE OVERDUE
```

This file is a proper data file and can be changed, added to or reduced by further selections. The program also saves the corresponding .DEF field definition file and the .FMT screen format file.

When selecting records the standard symbols can be used for numeric fields: =, >, >=, <, <=, <>. Each of these has a number of equivalent conditions such as GT, IS GT, IS > and IS NOT LE for the > symbol. For text fields you can use asterisks at the beginning or end of a word (or both) to indicate that the program is to search for a string of characters within a field. You can also use the ? wildcard to replace individual characters. For example, to find all customers in Liverpool you may use:



```
Dataset: SUBS                          11 Records

Reading        11
Final merge         11
Done.
A>>list subs by reference date.due company

Dataset: SUBS                          11 Records

Reference Date.due Company

P004      12.05.86 MCC
T567      01.02.87 Honeyway
P873      06.07.87 P.D.S.Farr
A245      12.08.87 D & B Ltd
A458      04.11.87 Advanced Resources
E345      06.05.88 Lake & Simon
W899      03.08.88 Williams & Williams
F012      05.08.88 Type Right
R509      12.08.88 Long & Co
Y678      03.09.88 Gallway Enterpride
R560      15.11.88 S.D.R And Sons

A>>█

                                                                    Drive is A:
```

Figure 10.10 *Sorting a file*

SELECT SUBS WHERE (ADDRESS4 = L* AND
ADDRESS4 = *POOL)

This will find those with addresses that do not include a postcode, including those where an abbreviation (such as "L'pool") has been entered in the last line of the address. Always take care with such combinations; the instruction will not find those with only three-line addresses.

Sorting records

Any file can be rearranged by using a SORT command:

SORT SUBS BY DATE.DUE COMPANY

The records now appear in any LIST command in order of the date due and, for each date, in alphabetical order of name. This order is used for all commands until there is another SORT command or the program is restarted.

Reports

At any time you can produce a printout that is identical in layout to the screen displays. For example:

```
PRINT SUBS
PRINT SUBS BY REFERENCE COMPANY DATE.DUE
AMOUNT.OWED
```

In the first case the complete records are printed in the same format as the screen displays; in the second case a columnar format is adopted.

All printer output can be stored in a disk file with commands in the form:

```
PRINTER FILE TEXT.DOC
```

Other printer commands include:

PRINTER OFF	All printer output is ignored
PRINTER ON	Output is printed again, cancelling previous PRINTER commands
PRINTER STATUS	Displays current destination for printer output.

As before, the printer must have been set up in CP/M before the program was started.

Looking after files

Condor 1 contains a range of file and disk commands that are useful in day-to-day housekeeping tasks. These include:

DIR	Lists all files on a given disk
DIC	Lists the *data dictionary*, comprising the data file names with their corresponding format and definition files. (Two additional columns are provided but not used.)
DATE	Changes the current date (e.g. DATE 1/4/87)
LOG	Informs the system that the data disk has been changed
SYSTEM	Ends the program and returns to CP/M

Files can be copied with commands in the form:

```
COPY B:SUBS=SUBS
```

To delete a data file and its corresponding format and definition files (if the names are the same) enter:

```
DESTROY SUBS
```

Alternatively, the contents of the data file can be deleted while leaving the structure unharmed, ready for a new set of data, with a command such as:

```
EMPTY SUBS
```

If you have a file, e.g. SUBS2, whose contents are to be added to some other file then this can be achieved with the APPEND command:

```
APPEND SUBS SUBS2
```

A file can be renamed as follows:

```
RENAME SUBS87 = SUBS86
```

This changes the name of the data file SUBS86.DAT to SUBS87.DAT. It also renames the format and definition files. To change just one of these files the extension should be specified.

Export and import
Condor 1 allows the transfer of data between files and between programs. The WRITE command stores a copy of the data in an ASCII file. The new file does not contain any information apart from the actual data. For example, to save the SUBS data in the file SUBSDATA.TXT use the command:

```
WRITE SUBS SUBSDATA.TXT
```

An extension must be given for the ASCII file. Various options are available for choosing special formats so that the ASCII file can be imported by BASIC programs and other applications.

The data from an ASCII file, created either by Condor 1 or some other program, can be read into a Condor 1 data file with a command in the form:

```
READ SUBS2 SUBSDATA.TXT
```

Since the ASCII file does not contain any definition information, the data can be read into any file that has a suitable structure. For example, the numeric data could be read into text fields without causing problems. However, the reverse is not true and any attempt to do so would result in the offending records being ignored.

You must therefore be very careful if editing a file with a word processor. Accidentally including a blank field at the beginning of the record could result in the record being skipped; at best all the data would be in the wrong fields. It is advisable to check that the number of new records in the Condor 1 file matches the number of records in the ASCII file to ensure that none have been missed in this way.

Programming

The direct commands described above are only the beginning of Condor 1. The program also incorporates extensive facilities for storing procedures comprised of lists of instructions, to be put into effect as and when required. Included are many of the instructions found in various forms within programming languages. For example, there is the ability to choose between two or more courses of action depending on the values of existing fields and variables; and the option to repeat some of the instructions either a given number of times or until some condition is true.

The programming facilities of Condor 1 are briefly outlined in the next chapter.

Support

Condor 1 is supplied with a 300-page book, describing its operation in detail. This is not as daunting as it sounds. The book works carefully through the wide-ranging facilities of the program, starting at the very elementary level and only introducing the more advanced options after the basics have been covered. For many applications these additional features will not be needed and they can be ignored until they become essential. The book ends with large, comprehensive reference sections.

The book is well presented, with just the peculiarity that the sections are not actually numbered, despite frequent references to section numbers within the text.

Once you have registered with Caxton Software by returning the registration card, telephone support is free and limitless. However, in order to save their time you should obviously thoroughly check the procedures you are using before contacting them. This will also help their technical staff identify the problem. The willingness to offer unlimited support is an indication of confidence and the sign of a good program.

Conclusion

Condor 1 is a comprehensive, advanced, yet simple to use, database program. Even if only the commands described in this chapter are used, the opportunities provided encompass nearly all the facilities of the programs described in the previous chapters. It is, of course, correspondingly more expensive. For those new to computers the complexities of the program may be somewhat confusing and therefore - for simple applications at least - Cardbox or Retrieve may provide a better introduction to databases. For anyone who is vaguely familiar with computer database programs, Condor 1 should be sufficient for most needs and is readily understandable.

For most advanced database requirements the programming capabilties of Condor 1, described in the next chapter, should be sufficiently powerful.

CHAPTER 11

CONDOR 1: PROGRAMMING

The previous chapter described the essential ingredients of Condor 1 for the storage and retrieval of data, based on the card index approach. Condor 1 contains many more commands that extend the program's scope to cover some highly sophisticated programming features. This chapter provides a brief insight to the way in which a procedure can be constructed to maximise the benefits of computer data storage. The principles covered here are applicable to most database packages that have a programmable capability.

Creating a program file

The prospect of writing and running a program from within Condor 1 is often seen as a rather daunting task. However, there is no need for this to be so, especially if you are already thoroughly familiar with the basic Condor 1 commands and are well versed in their use.

At its simplest a Condor 1 procedure is a list of instructions that are executed in turn. The instructions are stored in a text file and Condor 1 puts them into effect when you enter the appropriate command. The advantages of this approach are:

- A set of instructions can be repeated by re-entering a single command. This avoids having to retype all the instructions each time they are required.

- If a mistake is made or there is a flaw in the logic behind the instructions then this can be corrected quite simply and you can be satisfied that no other errors will creep in.

- A complex procedure can be set up in such a way that it can be used by someone who is unfamiliar with Condor 1.

- For complex processes that take a long time to complete the instructions are put into effect and carried out without the need to have someone permanently on hand to enter commands.

The combination of all these advantages can lead to some very sophisticated programs. For example, an accounts program can be devised that presents a new user with a menu of options for producing invoices, printing statements and calculating stock levels. This menu is displayed by entering just one command and can be used as often as necessary. The program can be made to ask for any information that is needed and can be happily left to print a stack of invoices or statements (although there should always be someone near enough to be able to take fast action if something goes wrong, such as the printer getting jammed.)

Setting up the program file
The file containing the procedure is an ASCII file, with one instruction on each line, terminated by a carriage return. Condor 1 does not contain facilities for creating such a file but any text editor or word processor should be able to create a file in a suitable format. If a word processor is used then the option to save the text as an ASCII file must be used, rather than the special coded file normally used by the word processor.

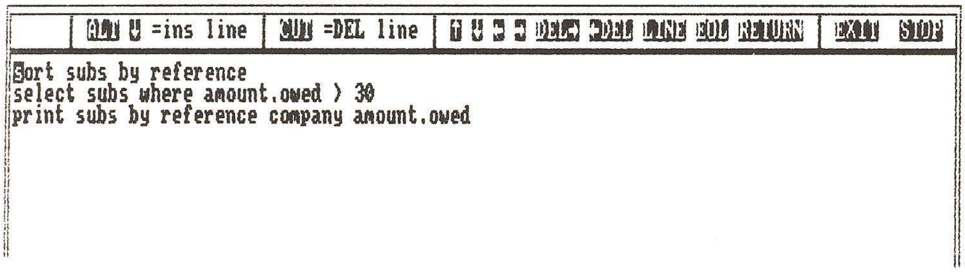
The easiest approach is probably to use the RPED text editor supplied on the CP/M Plus disk. This program can be put into effect with the command:

```
SUBMIT RPED
```

Choose the option to create a new file and supply a filename. The filename must have a .CMD extension. When the blank screen is displayed type the instructions as if they were being entered at the Condor 1 prompt. Complete each instruction by pressing RETURN.

Make sure that the instructions are in a logical order. They should be entered in precisely the same sequence as if you were typing them at the command line. In any particular file you can have up to 127 instructions, each of no more than 124 characters.

For example, suppose that the procedure is to sort the file according to reference number, select all records where there is an amount outstanding



```
Sort subs by reference
select subs where amount.owed > 30
print subs by reference company amount.owed
```

Figure 11.1 *Typing a procedure*

and print the records in each of these cases. Start the text editor, put in the data disk and give the new file the name UNPAID.CMD. Then enter the instructions:

```
SORT SUBS BY REFERENCE
SELECT SUBS WHERE AMOUNT.OWED > 0
PRINT SUBS BY REFERENCE COMPANY AMOUNT.OWED
```

Any mistakes can be corrected and the procedure is then saved on disk. When you run Condor 1 again this sequence of events can be put into effect with the command:

```
RUN UNPAID
```

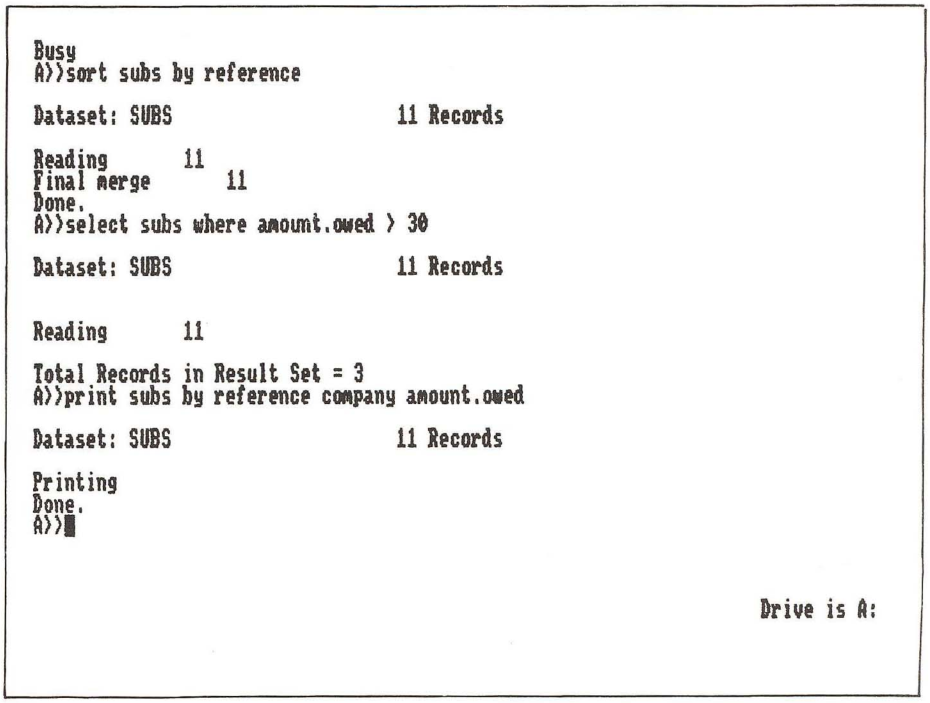
All the instructions are carried out automatically. If anything is wrong you can return to the text editor and make the necessary corrections. When the procedure is running satisfactorily it can be repeated as often as needed, without modification. Thus a frequently repeated task can be put into effect without worrying about individual instructions being mistyped or missed altogether.

A procedure can end with a RUN instruction to put another procedure into effect. Any other instructions after the RUN command are ignored.

Cosmetic text

There are other instructions in Condor 1's command language; these are generally identified by starting with an asterisk. Instructions starting with an asterisk are always carried out before other instructions, regardless of the order in which they appear.

In order to make your database operations more understandable it is useful to add explanatory texts. This can take the form of comments in the



```
Busy
A>>sort subs by reference

Dataset: SUBS                      11 Records

Reading        11
Final merge        11
Done.
A>>select subs where amount.owed > 30

Dataset: SUBS                      11 Records


Reading        11

Total Records in Result Set = 3
A>>print subs by reference company amount.owed

Dataset: SUBS                      11 Records

Printing
Done.
A>>█



                                                                  Drive is A:
```

Figure 11.2 *Running the procedure*

procedure listing, helpful messages to the user on the screen or headings and other text on printouts.

Comments

In a long procedure it can be useful to include special *comment* lines, as a reminder of what each part of the program is intended to do. Comments must always start with a semi-colon; this ensures that they are ignored by Condor 1 when the procedure is run. In the example above a title may be given to the procedure to help identify it later on:

```
;Program to select all unpaid subscribers
;Written on 8/2/87
```

It is surprisingly easy to forget the purpose of your procedure and an item of text like this helps to identify the procedure's aims. It is useful to include the date, especially if more than one version of the procedure exists. If this line is updated each time the file is changed then it also provides a means of checking that the most recent backup copy is also up-to-date.

If you want to separate different sections of the procedure to make them easier to read then you can place a semi-colon on its own on a line; you cannot leave blank lines in the procedure.

Screen messages
Even more useful than the comments facility is the ability to display messages to the user on the screen while the procedure is running. Any piece of text is displayed if preceded by the *MESSAGE command. For example, messages can be used to indicate the current state of the program:

```
;Procedure UNPAID2
*MESSAGE Printing addresses
PRINT SUBS BY COMPANY ADDRESS1 ADDRESS2 ADDRESS3
ADDRESS4
POSTCODE
RUN PROC2

;Procedure PROC2
*MESSAGE Printing amounts outstanding
SELECT SUBS WHERE AMOUNT.OWED > 0
PRINT SUBS BY COMPANY AMOUNT.OWED
RUN PROC3

;Procedure PROC3
*MESSAGE Printing completed
```

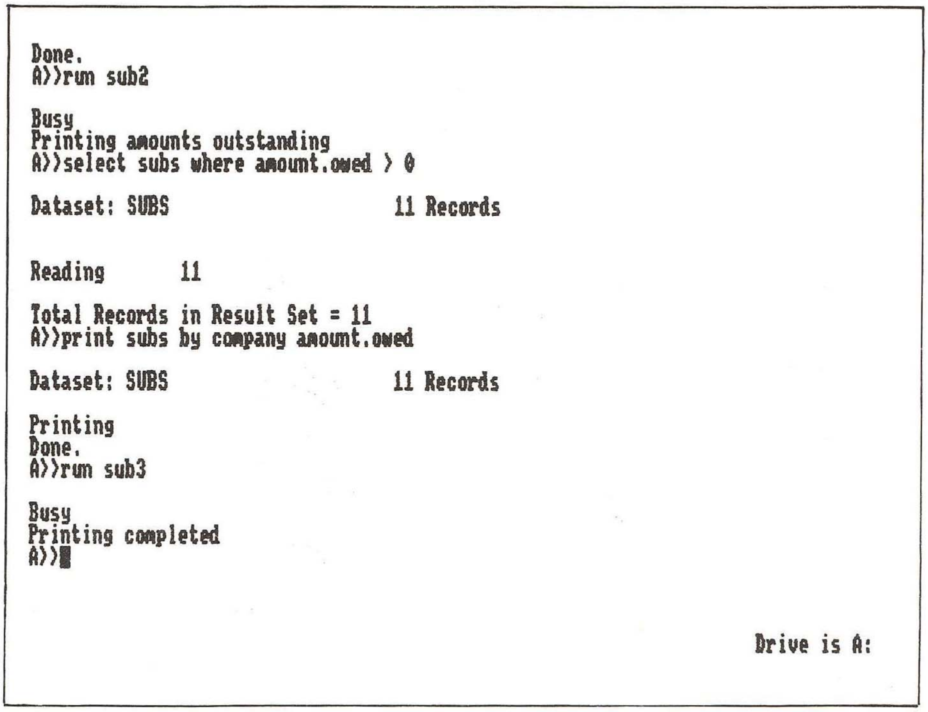
As a general rule the comands being executed and their results are automatically displayed on the screen. These displays can be switched off and on again with the commands:

```
SET ECHO OFF
SET ECHO ON
```

After a procedure has finished the display is always turned back on again.

Titles and print formatting
The TITLE command places an item of text at the top of each page. The text must be in quotes. Other instructions to the printer can be included in the TITLE command:

LINE	Starts a new line
TOP	Starts a new page
SKIP	Misses a line
SKIP,n	Misses a line after every n lines
LENGTH=n	Sets number of lines per page
WIDTH=n	Sets number of characters per line
DATE	Includes the current date in the title
PAGE	Includes a page number in the title



```
Done.
A>>run sub2

Busy
Printing amounts outstanding
A>>select subs where amount.owed > 0

Dataset: SUBS                     11 Records


Reading        11

Total Records in Result Set = 11
A>>print subs by company amount.owed

Dataset: SUBS                     11 Records

Printing
Done.
A>>run sub3

Busy
Printing completed
A>>█



                                                       Drive is A:
```

Figure 11.3 *Running a program with screen messages*

Each item must be separated by commas and there can be more than one TITLE instruction. For example, the following format may be adopted:

```
TITLE LENGTH=50, WIDTH=60
TITLE " Subscriptions list: printed on ",DATE," ",PAGE
```

This results in pages of 50 lines, each with a maximum of 60 characters, and individual pages headed as follows:

```
Subscriptions list: printed on 05/04/87                PAGE 1
```

Note that spaces were used to shift the title to the right, to put a space in front of the date, and to set the page number to the right.

Assigning values

Various means exist for assigning values to fields while running a program and for entering information from the keyboard.

The most obvious way of doing this is to include an UPDATE instruction.

When the program is run the user is required to change the selected records in the usual way.

Variables
It is often necessary to store for use with a procedure, some item of information, whose value will be different each time the procedure is run. For example, a procedure may be devised to print all records where the subscription is due on a given date; each time the procedure is run a different date must be used for the selection. This can be achieved by placing the value in a *variable*. A variable is a temporary field that is held in memory only as long as the program is running.

Field variables The first method is to use the actual values of fields in a condition rather than some constant value. For example, all overdue accounts can be listed with the command:

```
PRINT SUBS WHERE DATE.DUE < 01/01/87
AND AMOUNT.OWED > 0
```

However, if you only want to print those where the full amount is outstanding, this approach is suitable, since the value to be selected depends on the original subscription. To indicate that you are comparing a field's value with another field rather than a specific value the second field is preceded by an @ symbol. The command now becomes:

```
PRINT SUBS WHERE DATES.DUE < 01/01/87 AND
AMOUNT.OWED =@ SUBSCRIPTION
```

The two fields are compared and the record is printed only if the two values are the same. All the usual symbols can be used in such a condition.

Program variables The second method of using variables is to define special fields that exist only temporarily.

The program variables are given the names \$1, \$2, . . ., \$9. Their contents can be specified when the program is actually run by placing values after the filename. The first value given is assumed to be the value for \$1, the next is the value for \$2, and so on.

For example, a short procedure to print all subscribers falling due on a given date, sorting them in order of company name, could be as follows:

```
;Print all subscribers for date $1
SORT SUBS BY COMPANY
PRINT SUBS WHERE DATE.DUE = $1
```

If this procedure is in a file named SUBSDUE.CMD all subscriptions due on

1 January 1987 can be printed with the command:

```
RUN SUBSDUE 01/01/87
```

The variable $1 is replaced by the value 01/01/87.

Assigning variable values It is sometimes useful to give a program variable a specific value from within the procedure with an *assignment* statement. In the case of Condor 1 this is done with the *LET instruction, as follows:

```
*LET $2 = 18
```

This assigns the value 18 to variable $2. Any previous value is replaced. The new value is effective until another *LET instruction changes $2 or the procedure ends.

Inputting values Another method of changing the value of a program variable is to type it in from the keyboard while the program is running. The *GET instruction prints a colon on the screen and then waits until a value has been typed in. The value is stored in the specified variable. This command is often used in conjunction with a screen message.

For example, to enter an amount for use in a selection process the following instructions may be included:

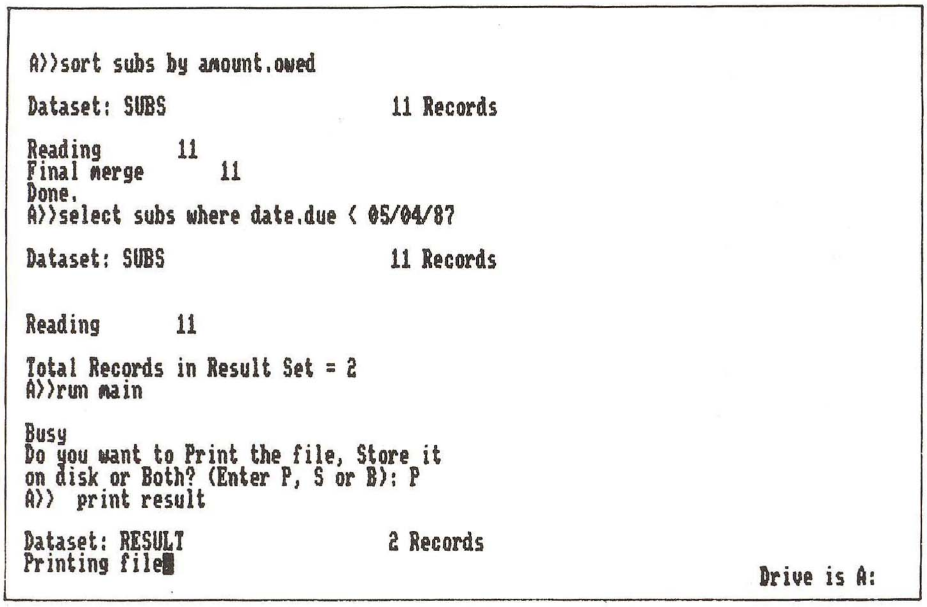
```
;Print list of subscribers to most expensive magazines
*MESSAGE This program will list all subscribers
*MESSAGE who have spent more than a specified amount
*MESSAGE
*MESSAGE Enter the minimum value
*GET $1
PRINT SUBS WHERE SUBSCRIPTION > $1
```

More than one value can be input at a time by separating the variables with commas.

Branching

One of the main advantages of any database command language is that you can usually build decision-making processes into the program. In the case of Condor 1 this is done with the *IF. . .*ENDIF construction.

The *IF instruction must be followed by a condition, usually referring to a variable. If the condition is true the instructions that follow are carried out until an *ENDIF statement is reached; if the condition is false the program skips to the instruction following the *ENDIF command.



```
A>>sort subs by amount.owed

Dataset: SUBS                       11 Records

Reading       11
Final merge       11
Done.
A>>select subs where date.due < 05/04/87

Dataset: SUBS                       11 Records


Reading       11

Total Records in Result Set = 2
A>>run main

Busy
Do you want to Print the file, Store it
on disk or Both? (Enter P, S or B): P
A>>  print result

Dataset: RESULT                     2 Records
Printing file
                                                      Drive is A:
```

Figure 11.4 *A branching program*

For example, following a selection process an option may be given to the user to decide whether or not to print the results. The following procedures would achieve this:

```
;Procedure START
SORT SUBS BY AMOUNT.OWED
SELECT SUBS WHERE DATE.DUE < 05/04/87
RUN MAIN

Procedure MAIN
*MESSAGE Do you want to print the file (Y/N)?
*GET $1
*IF $1 = Y
  PRINT RESULT
   *ENDIF
*MESSAGE Do you want to store the results on disk (Y/N)?
*GET $1
*IF $1 = Y
  SAVE RESULT APRIL87
   *ENDIF
```

The conditional instructions are indented for improved legibility; the extra spaces are ignored by the program.

Another option would be to ask a single question, with several alternative courses of action:

```
*MESSAGE Do you want to Print the file, Store it
*MESSAGE on disk or Both? (Enter P, S or B)
*GET $1
*IF $1 = P or $1 = B
  PRINT RESULT
  *ENDIF
*IF $1 = S or $1 = B
  SAVE RESULT APRIL87
  *ENDIF
```

Some database programs allow the use of an 'ELSE' clause; the instructions following this statement are executed only if the condition is false.

Ending a procedure

Before a procedure is run, Condor 1 copies all the instructions to a special work file. It continues until one of these situations arises:

- The end of the program file is reached, in which case the procedure is put into effect.
- An *END instruction is encountered, in which case all commands up to that point are executed.
- An *ABORT instruction is found, in which case none of the procedure is executed.

For example, you may include the following instructions in a file:

```
*MESSAGE Do you wish to continue (Y/N)?
*GET $1
*IF $1 = Y
  PRINT SUBS BY $2
  *END
  *ENDIF
*IF $1 = N
  *ABORT
  *ENDIF
```

In certain circumstances, for example when HELP screens are used (as described below), the *ABORT instruction returns you to an earlier level of the procedure rather than ending the program altogether.

Loops

Program *loops*, in which a set of instructions is repeated more than once, are useful in several circumstances:

- To repeat the same instruction for every selected record in a file.
- To repeat a sequence of several instructions for every selected record in a file.
- To repeat part of the procedure a given number of times or until a condition is true.
- To repeat part of the procedure for a series of values supplied by the user.

The first case is taken care of with instructions such as LIST and PRINT. If these follow some suitable SORT and SELECT instructions, they will repeat the instruction for all selected records, providing listings, totals, etc.

The second and third cases are generally dealt with by a WHILE. . .ENDWHILE or REPEAT. . .UNTIL construction. For example, you may wish to print the contents of a record along with some other text or values, repeating this for the whole file. Alternatively you may want to repeat some part of the procedure until a variable reaches some specific value. Neither of these constructions is available in Condor 1 and therefore this sort of operation is not available.

The final case is catered for by the use of HELP screens.

Displaying menus
Condor 1 includes an extremely useful and very unusual feature that allows you to display a menu of options on the screen. The user selects one of these options and, when its functions are complete, is returned to the menu to choose another option. For example, you may wish to supply the following options:

1 Print names and telephone numbers
2 Print overdue accounts
3 Sort and selection options
4 Disk utilities
5 End program

For the first two options, a simple Condor 1 instruction needs to be executed. In the third case the program needs to supply a further list of options while the fourth option must run the CP/M PIP utility. The final option takes you back to the main Condor 1 command line.

The options would be taken care of by these instructions:

```
1: PRINT SUBS BY COMPANY TELEPHONE EXTENSION
2: PRINT SUBS WHERE AMOUNT.DUE > 0 BY COMPANY
   AMOUNT.DUE
3: RUN SORTSEL
4: $PIP B: = *.*
5: ABORT
```

Any procedure can include a RUN instruction to put another procedure, e.g. SORTSEL.CMD, into effect; when this procedure has finished, the calling procedure resumes with the next instruction.

Any CP/M command can be put into effect from the Condor 1 command line by preceding the command name with a $ symbol.

Creating a menu The menu display must first of all be stored in a .HLP file with the FORMAT command, as follows:

```
FORMAT MENU1.HLP
```

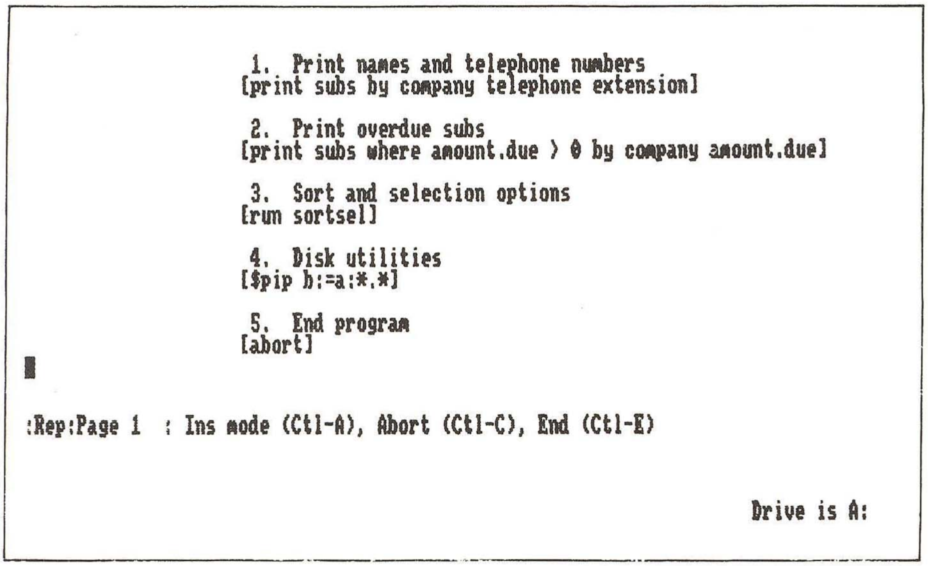
You are presented with a blank screen on which you can type any menu options, instructions to the user and other titles, boxes, lines and cosmetic text. All of this is displayed when the appropriate HELP instruction is encountered.

The .HLP file also has a second purpose, which is to indicate to the program the instructions that are to be carried out as a result of an option being selected. Each Condor 1 instruction must be typed onto the screen, in the same way as the text but enclosed in square brackets. It is usual to place the instruction near the corresponding menu option, to ensure there is no confusion, but this is not essential.

When the program is run, only the text is displayed on the screen; the instructions in square brackets are not shown. The program instructs the user to enter an option number and then executes the corresponding command. The relative position of the options on the screen makes no difference to the option that is assigned to them, in that the first instruction encountered is assumed to be option 1, the next is option 2, and so on.

Therefore, if space is short in the main body of the menu all the instructions could be placed in a long string at the bottom:

```
[PRINT SUBS BY COMPANY TELEPHONE EXTENSION][PRINT
SUBS. . .]
[RUN SORTSEL][$PIP B: = *.*][ABORT]
```



```
                    1.  Print names and telephone numbers
                   [print subs by company telephone extension]

                    2.  Print overdue subs
                   [print subs where amount.due > 0 by company amount.due]

                    3.  Sort and selection options
                   [run sortsel]

                    4.  Disk utilities
                   [$pip b:=a:*.*]

                    5.  End program
                   [abort]
█

:Rep:Page 1  : Ins mode (Ctl-A), Abort (Ctl-C), End (Ctl-E)


                                                           Drive is A:
```

Figure 11.5 *Designing a menu*

The menu is put into effect with the following command:

HELP MENU1

Every time an instruction has been completed the menu is redisplayed and a new option can be selected. Each ABORT command returns the user to the previous menu or to the Condor 1 command line.

Calculations

A range of instructions are available for performing calculations.

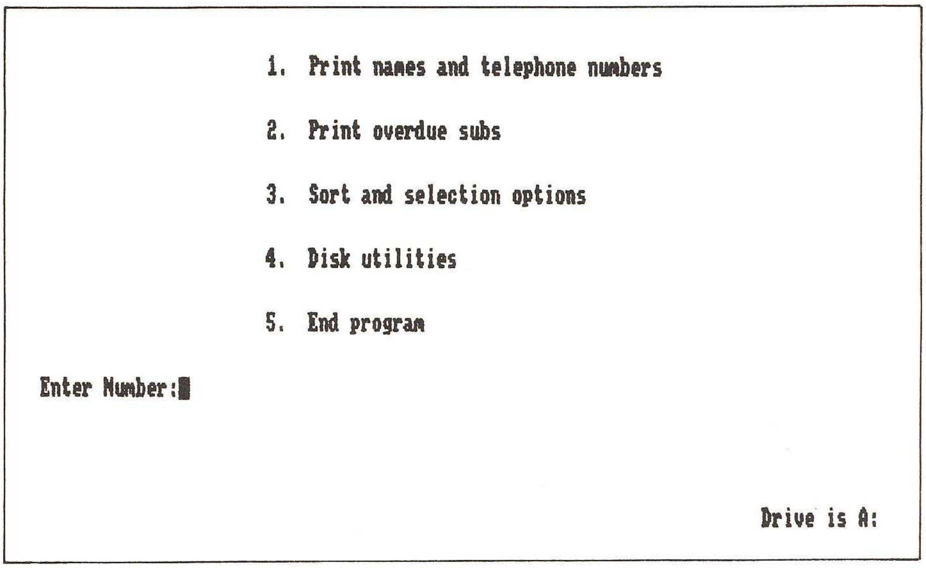
Statistical summaries For any field the STAX command calculates the total, minimum, maximum and average for all values over the file. For example, these values can be calculated for the two money fields as follows:

STAX SUBS BY SUBSCRIPTION AMOUNT.OWED

These details are printed if [P] is added to the command. The instruction can also include a selection condition:

STAX SUBS WHERE AMOUNT.OWED > 0 BY SUBSCRIPTION

This prints the total and average subscription for those records where there is



```
                         1. Print names and telephone numbers

                         2. Print overdue subs

                         3. Sort and selection options

                         4. Disk utilities

                         5. End program

  Enter Number:█



                                                          Drive is A:
```

Figure 11.6 *Using a menu*

still an amount oustanding and also finds the maximum and minimum values of the SUBSCRIPTION field.

Calculated fields The value of any numerical field can be calculated from the values of other fields with the COMPUTE command. The four arithmetical operators, plus (+), minus (–), multiply (*) and divide (/), can be used. The command affects every record unless a selection condition is included. For example, to increase the subscription by 10% for all records where the date due is after 1 May 1987, use the following:

```
COMPUTE SUBS WHERE DATE.DUE > = 01/05/87 ST
SUBSCRIPTION = SUBSCRIPTION * 1.1
```

(ST stands for "such that".) All subscribers who are due to receive a renewal notice after 1 May have now had their accounts updated to the new value. Obviously great care needs to be taken with this sort of operation; accidentally entering the instruction twice could have rather unfortunate consequences!

Dates can also be used in the calculations. For example, to update those subscribers who are paid up-to-date you could use:

```
COMPUTE SUBS WHERE DATE < 01/05/87
AND AMOUNT.OWED = 0
ST DATE.DUE = DATE.DUE + 365
```

Complex formulae can be built up, using a number of fields. The order of calculation can be changed by using brackets. More than one calculation can be carried out at a time by placing the formulae on separate lines.

Tabulation The TABULATE command produces a table, listing all the values found for a specific field. Against each value is shown the number of records that hold that specific value. For example, to count the number of subscribers for each subscription rate the command is:

```
TABULATE SUBS BY SUBSCRIPTION
```

Various other calculations can be carried out using the POST command described below.

Combining files

There are many occasions when you have more than one file and wish to compare their contents or combine values in some way. On other occasions you may wish to have a simplified form for entering just specific items of data.

Using an entry file

With large files there is always a danger that during any editing operation errors will creep into the data. Obviously, for database management to be a success, it is important to keep any errors to a minimum and to build in as many checks as possible to ensure that the data is accurate. If new entries are added to the main file then any errors may go unnoticed unless the new records are printed and carefully checked.

To overcome this danger it can be helpful to set up a special entry file. The data file is given a new name, e.g. NEWSUBS, but the format and definition files are the same as for the main file. The new file can be used to enter all new subscribers with the command:

```
ENTER NEWSUBS
```

When this is complete you will need to check that the unique field used to identify the records, e.g. REFERENCE, has not been duplicated. This is done with the COMPARE command:

```
COMPARE NEWSUBS SUBS MATCHING REFERENCE
```

The special RESULT file will hold all NEWSUBS records for which the value of REFERENCE is also to be found in SUBS. This temporary file can be listed with:

```
LIST RESULT
```

If no errors in reference numbers are found then RESULT should of course be empty. The next stage is to print the contents of the entry file and, having checked that it contains no errors, add the new records to the main file. You may also wish to make a copy of the new data (as an added precaution) and then erase the contents of the new file, ready for the next session. The commands used are:

```
PRINT NEWSUBS
APPEND SUBS NEWSUBS
COPY 04-03-87.DAT NEWSUBS.DAT
EMPTY NEWSUBS
```

Note that the current date is used as the filename for the security file.

This whole procedure could be automated as follows:

```
;Procedure START
HELP CHOICE1
;Procedure ADDSUBS
ENTER NEWSUBS

;Procedure DUPLICATE
COMPARE NEWSUBS SUBS MATCHING REFERENCE
PRINT RESULT
EMPTY RESULT
RUN UPDATES

;Procedure UPDATES
*MESSAGE Make changes to new file (Y/N)?
*GET $1
*IF $1 = Y
  UPDATE NEWSUBS
  RUN ENDPROC
   *ENDIF

;Procedure ENDPROC
*MESSAGE Choose option again: press RETURN to continue
*GET $2

;Procedure ACCURACY
PRINT NEWSUBS
RUN UPDATES

;Procedure COMBINE
APPEND SUBS NEWSUBS
RUN COMBINE2
```

```
;Procedure COMBINE2
*MESSSAGE Enter today's date
*GET $3
COPY $3 NEWSUBS.DAT
EMPTY NEWSUBS
```

The menu file CHOICE1.HLP contains the following options:

```
1  Enter new records
   [RUN ADDSUBS]
2  Check for duplicate records
   [RUN DUPLICATE]
3  Check accuracy of new data
   [RUN ACCURACY]
4  Add to main file
   [RUN COMBINE]
5  Exit
   [ABORT]
```

The user should be instructed to use each option in turn, repeating any option if errors are found.

Updating a file
If you want to make changes to specified fields only, then it can be rather tedious to work through the entire data file, and once again may be a source of unnecessary error. Condor 1 allows you to overcome this by entering the changed details into a new file and then transferring this revised information to the main file.

For example, you may wish to enter details of subscriptions received and update the AMOUNT.OWED field. A new file, e.g. RECEIPTS.DAT, can be created, with fields for REFERENCE and AMOUNT.OWED. It is important that the main field used to identify the record should be included. Any data that is to be transferred must also be in a field that exists in the main file. You can also include other details, such as the date on which the payment was received.

The new data can then be entered. All that is needed is the reference number and amount paid, plus any other details that are to be stored in this file for future reference. A COMPARE command can be used to check that all the references given actually exist in the main file. The new details can be transferred across to the main file with the POST command:

```
POST SUBS RECEIPTS MATCHING REFERENCE AND SUBTRACT
AMOUNT.OWED
```

The values of AMOUNT.OWED in RECEIPTS are deducted from the corresponding records in SUBS. RECEIPTS is unchanged and can be stored away for future checks.

As well as the SUBTRACT option you can choose to ADD the new value or REPLACE the existing value with the new one. More than one field value can be posted at a time by separating the items with commas, as in:

```
POST SUBS NEWSUBS MATCHING REFERENCE AND SUBTRACT
AMOUNT.OWED, REPLACE SUBSCRIPTION
```

It is always as well to check that the results you have achieved are sensible ones. For example, you can test that no-one has overpaid with the command:

```
COMPARE SUBS RECEIPTS MATCHING REFERENCE
PRINT SUBS BY REFERENCE SUBSCRIPTION AMOUNT.OWED
```

In this way it is possible to link several files; different methods of linking can be devised by including more than one unique reference in a file. Further examples of Condor 1 procedures are given in Appendix 1.

Conclusion

The Condor 1 command language is not difficult to master, though care is needed when dealing with more than one file at a time. Some of the features you might expect to find in such a comprehensive command language are missing (WHILE. . .ENDWHILE for instance), but there are few processes that cannot be achieved with careful handling of the commands that are available.

Condor 1 provides a good introduction to the principles of command languages and, if your database requires anything other than standard editing and reporting facilities, is well worth the investment of time and money.

APPENDIX 1

SAMPLE CONDOR PROCEDURES

Validation and verification

Validation of individual fields can cut down the number of errors in your data.

Range checks
To check for values within a given range, e.g. hours, minutes and seconds, during data entry include the minimum and maximum values in the field definition.

Alternatively, an entire file can be validated after entry of all records. For a file called TIMES with fields REF and HOURS, the following procedure requests minimum and maximum values for the file and lists those records where there are values outside the range:

```
;Procedure HOURTEST
*MESSAGE Enter minimum value for HOURS
*GET $1
*MESSAGE Enter maximum value for HOURS
*GET $2
SELECT TIMES WHERE HOURS < $1 OR HOURS > $2
TITLE 'Errors found in TIMES on ', DATE, LINE
PRINT RESULT BY REF, HOURS
```

Using totals

For a file called STOCK with entries made for three quantities in STORE1, STORE2 and STORE3 you can check that the values are correct by also entering a TOTAL1 value and then checking this against a calculated TOTAL2 value:

```
;Procedure TOTALS
COMPUTE STOCK ST TOTAL2 = STORE1 + STORE2 + STORE3
SELECT STOCK WHERE TOTAL1 <> TOTAL2
TITLE 'Errors in stock values or total', LINE
PRINT RESULT BY REF, STORE1, STORE2, STORE3, TOTAL1
```

Double entry

If data has been entered twice, in files FILE1 and FILE2 with fields FIELD1, FIELD2, etc., then the records where there are discrepancies can be identified with:

```
;Procedure DOUBLES
COMPARE FILE1 FILE2 NOT MATCHING FIELD1, FIELD2,. . .
LIST RESULT
```

Tracing programming errors

If a program has a fault then there are several ways of dealing with this.

Tracing the program path

You can trace the way in which the program uses the procedures and the way in which branches are dealt with by displaying messages at appropriate points. These messages can be deleted once the fault has been corrected. (Remember that all * instructions are executed first.) For example:

```
;Procedure A
*MESSAGE Starting procedure A - press RETURN
*GET $1
. . .
```

Displaying field values

If calculations are incorrect then you can list the values of fields at appropriate points:

```
LIST FILE1 BY REF
SELECT FILE1 WHERE FIELD1 > FIELD2
LIST RESULT BY REF, FIELD1, FIELD2
```

Standard documents

You can prepare data for inclusion in standard documents in a number of ways:

- If you have a word processor that can merge a standard document with a data list, the data can be prepared in a suitable format with the command:

  ```
  WRITE FILE1 [B]
  ```

  The [B] parameter ensures that fields are separated by commas, records are separated by carriage returns and all text is in quotes. Other formats are available.

- To produce the documents entirely from within Condor a new file must be created. The screen format should consist of the standard document with spaces where the values are to be inserted. Each record should consist of just those fields that are to be used in the document and they should be placed on the screen format in the appropriate gaps.

If the standard document is in the file STANDARD and has fields FIELD1, FIELD2, etc., and the original file is ORIGINAL (including all the STANDARD fields) the documents can be printed with the instructions:

```
TITLE TOP, 30
POST STANDARD ORIGINAL MATCHING REF REPLACE
FIELD1, FIELD2,. . .
PRINT STANDARD
```

It is assumed that the document is 30 lines long and that all the REF values already exist in STANDARD before posting takes place.

APPENDIX 2

WORK DISKS

This appendix suggests the contents of a single-sided work disk for each of the database programs described in this book. For double-sided disks there is sufficient room to hold the entire contents of the original disk plus all the necessary CP/M Plus utilities.

Matchbox	
Program files	47K
J14CPM33.EMS	40K
SUBMIT.COM	6K
PAPER.COM	2K
DEVICE.COM	8K
SETSIO.COM	2K
SETKEYS.COM	2K
PIP.COM	9K
DISCKIT.COM	7K
ERASE.COM	4K
RENAME.COM	3K
DIR.COM	15K
TYPE.COM	3K

Cardbox	
CARDBOX.COM	5K
CARDBOX.OVR	48K
TERMINAL.DEF	2K
J14CPM33.EMS	40K
SUBMIT.COM	6K
PAPER.COM	2K
DEVICE.COM	8K
SETSIO.COM	2K
SETKEYS.COM	2K
PIP.COM	9K
DISCKIT.COM	7K
ERASE.COM	4K
RENAME.COM	3K
DIR.COM	15K
TYPE.COM	3K

AtLast 1

Side 1:	
ATLAST.KEY	2K
DBDEF.000	3K
DBDEF.001	28K
DBDEF.COM	40K
J14CPM33.EMS	40K
DEVICE.COM	8K
SETSIO.COM	2K
SETKEYS.COM	2K
PIP.COM	9K
DISCKIT.COM	7K
ERASE.COM	4K
RENAME.COM	3K
DIR.COM	15K
TYPE.COM	3K
Side 2:	
ATLAST.KEY	2K
ATLAST.SUB	1K
DBUSE.000	4K
DBUSE.001	6K
DBUSE.002	10K
DBUSE.003	13K
DBUSE.COM	40K
J14CPM33.EMS	40K
SUBMIT.COM	6K
PAPER.COM	2K
DEVICE.COM	8K
SETSIO.COM	2K
SETKEYS.COM	2K
PIP.COM	9K
DISCKIT.COM	7K
ERASE.COM	4K
RENAME.COM	3K
DIR.COM	15K
TYPE.COM	3K

Sagesoft Retrieve

Side 1: Program files	152K
SUBMIT.COM	6K
PAPER.COM	2K
SETSIO.COM	2K
DISCKIT.COM	7K
Side 2: Program files	110K
J14CPM33.EMS	40K
SUBMIT.COM	6K
PAPER.COM	2K
PIP.COM	9K

Cambase

Program files (excluding ?DATAF??.DAT)	112K
J14CPM33.EMS	40K
SUBMIT.COM	6K
PAPER.COM	2K
SETSIO.COM	2K
DISCKIT.COM	7K

Condor 1

Side 1: CP/M Plus system disk 170K

Side 2: Condor 1 program disk 166K

APPENDIX 3

ASCII CONVERSION TABLE

Decimal	*Character*	*Hexa-decimal*	*Decimal*	*Character*	*Hexa-decimal*
0	NUL	00	32	space	20
1	SOH	01	33	!	21
2	STX	02	34	"	22
3	ETX	03	35	#	23
4	EOT	04	36	$	24
5	ENQ	05	37	%	25
6	ACK	06	38	&	26
7	BEL	07	39	'	27
8	BS	08	40	(	28
9	HT	09	41	)	29
10	LF	0A	42	*	2A
11	VT	0B	43	+	2B
12	FF	0C	44	,	2C
13	CR	0D	45	−	2D
14	SO	0E	46	.	2E
15	SI	0F	47	/	2F
16	DLE	10	48	0	30
17	DC1	11	49	1	31
18	DC2	12	50	2	32
19	DC3	13	51	3	33
20	DC4	14	52	4	34
21	NAK	15	53	5	35
22	SYN	16	54	6	36
23	ETB	17	55	7	37
24	CAN	18	56	8	38
25	EM	19	57	9	39
26	SUB	1A	58	:	3A
27	ESC	1B	59	;	3B
28	FS	1C	60	<	3C
29	GS	1D	61	=	3D
30	RS	1E	62	>	3E
31	US	1F	63	?	3F

ASCII CONVERSION TABLE

Decimal	*Character*	*Hexa-decimal*	*Decimal*	*Character*	*Hexa-decimal*
64	@	40	96	‘	60
65	A	41	97	a	61
66	B	42	98	b	62
67	C	43	99	c	63
68	D	44	100	d	64
69	E	45	101	e	65
70	F	46	102	f	66
71	G	47	103	g	67
72	H	48	104	h	68
73	I	49	105	i	69
74	J	4A	106	j	6A
75	K	4B	107	k	6B
76	L	4C	108	l	6C
77	M	4D	109	m	6D
78	N	4E	110	n	6E
79	O	4F	111	o	6F
80	P	50	112	p	70
81	Q	51	113	q	71
82	R	52	114	r	72
83	S	53	115	s	73
84	T	54	116	t	74
85	U	55	117	u	75
86	V	56	118	v	76
87	W	57	119	w	77
88	X	58	120	x	78
89	Y	59	121	y	79
90	Z	5A	122	z	7A
91	[	5B	123	{	7B
92	\	5C	124	|	7C
93	]	5D	125	}	7D
94	^	5E	126	~	7E
95	_	5F	127	DEL	7F

Notes Codes 1 to 26 represent the codes for CONTROL-A to CONTROL-Z. Code 35 is generally used to represent both the # and £ symbols.

INDEX

USING DATABASES ON THE AMSTRAD PCW8256 & PCW8512

Stephen Morris

When the Amstrad PCW was first launched onto the market it was sold purely as a word processing machine. Since then the thousands of owners have realised its power and potential as a microcomputer. This book is designed to help the PCW owner utilise the capabilities of their investment to organise, store and retrieve information.

The book starts with a general introduction to information storage and contrasts manual and computerised systems. It then outlines the general principles behind computer databases, the Data Protection Act, data entry and file management. This is followed by a detailed discussion of six widely used database programs — Matchbox, Card Box, At Last 1, Sagesoft Retrieve, Cambase and Condor 1. These programs are examined in turn, with descriptions on how to prepare and create a data file; searching, sorting and selecting information; producing reports and looking after files.

The book thus offers both a general introduction to using databases on the Amstrad PCW's and specific descriptions of how to use a number of popular packages.

£8.95

ISBN 1-85181-121-4

GLENTOP

Glentop Press Ltd
Standfast House
Bath Place
High Street Barnet
Herts EN5 5XE

9 781851 811212