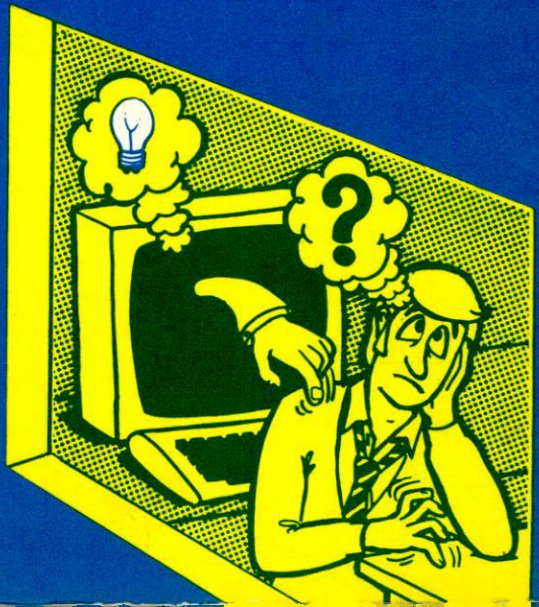
Idea Book

COMMODORE 64

A Data Becker Book from First Publishing Limited



1st
FIRST PUBLISHING LTD

IDEAS FOR USE ON YOUR COMMODORE

BY: Ranier Bartel

A DATA BECKER BOOK

PUBLISHED BY: FIRST PUBLISHING LTD
UNIT 20B
Horseshoe Road
Horseshoe Industrial Est.
Pangbourne
Berkshire, England

First English Edition, October 1984

Printed in Germany

Important Note

This book and the programs contained therein have been very carefully compiled by the author and checked during reproduction. Nevertheless, errors cannot totally be avoided. First Publishing Ltd are therefore obliged to point out that they cannot give any guarantee or accept legal responsibility for any consequences attributable to program errors or incorrect information in the book. We shall be pleased to learn of any errors at any time.

ISBN 0-948015-08X

Foreward

What can you really do with a Commodore '64 ? Many users ask this sort of question after the initial excitement of the purchase has worn off.

Rainer Bartel attempts to answer this question in his book by showing you numerous applications for the '64. The spectrum goes from file management to music programs, from diet to jogging programs. In fact, one of these applica - tions has proven very useful to Rainer. The original man - uscipt for this book was using a word processing package - FirstWORD on the '64, and printed on an Epson LQ 1500.

The best part, aside from the comprehensive information is that this book is interesting and easy to read - some- thing which cannot be said about many computer books.

We really hope you have fun while reading our books.

Table of Contents

0. Get your 64 out of the corner

Do you recognize the following dialog?

"So, why did you buy a computer?"

"Well, I want to use it to store all my addresses and I can use it for my studies. The games aside, there are all sorts of things I can do with it."

"You spent hundreds of dollars just to do that? What a waste!"

"Well, the way I see it, you just can't get by without a computer these days."

"Hmmph. You certainly won't catch me blowing my money on such a contraption."

Many conversations between computer owners and computer skeptics take place like this. If you happen to be in the uncomfortable position of having to justify your purchase of a C-64, you can quickly run out of things to say.

Of course you know very well how you are going to use your Commodore 64, don't you? Do you belong to that rather large group of impulse buyers that later put their computer under someone else's Christmas tree or to the group who is tired of just playing with their video games?

Let's try a little test just to see.

```
**************************************************************
*     How do you most want to use your C64?                  *
*                                                            *
*                                                            *
*     1. Only for games                          YES/NO      *
*     2. For hobbies                             YES/NO      *
*     3. In the household                        YES/NO      *
*     4. Professional or business Use            YES/NO      *
*     5. To be able to understand                            *
*        computers better                        YES/NO      *
*                                                            *
**************************************************************
```

If you answered question 1 with NO, then you're in the right place, particularly if you answered more than one other questions with YES. If, however, you want to use the computer primarily as a toy, then you should trade this book in for a game cartridge (assuming your dealer will go along with it.)

To make our point clearer: In this book you will find information about serious, useful applications that can even save you money. So get your '64 out of that corner!

1. Getting to know each other

Good morning. My name is Rainer Bartel. I am standing here on the starting line of the great computer race, ready to bring you a whole pile of new ideas. Before we get started however, I must get a chance to know you a little better, otherwise we'll never get a chance to really get moving.

The following questionnaire, by the way, is also intended to help you get to know yourself a little better. Perhaps you have never really given any thought to what you actually want to do with your computer. Once you finish the questionnaire, you'll know more.

So, on your mark, with pen in hand. The game begins.

****
* *
* Questionnaire *
* *
****
****
* Your hardware Configuration? *
* 1. Commodore 64 YES/NO *
* 2. Datasette (cassette recorder) YES/NO *
* 3. Floppy disk drive VIC 1541 YES/NO *
* 4. Second or third disk drive YES/NO *
* 5. Monochrome monitor YES/NO *
* 6. Color monitor YES/NO *
* 7. Dot matrix or daisy wheel printer YES/NO *
* 8. Plotter YES/NO *
****

Now you can begin with the first evaluation. But don't worry, you don't even have to turn on your '64.

NO YES ANSWERS : You should read this book from cover to cover. First, you'll learn about the fantastic possibilities of the Commodore 64 and also about its limitations. Second, you'll find many stimuli for future applications of your system. You can also skip over the rest of the questionnaire.

ONE YES ANSWER : That's strange. Either you picked up a peripheral real cheap which is now waiting for a computer or your '64 is still standing alone somewhere. My recommendation: First expand your system and then read further.

TWO OR THREE YES ANSWERS : You already own a system with which you can do many interesting things. There is also something in this book for you. My tip however: You should read section 1.2 and following, after which you'll probably seriously consider expanding your system.

MORE THAN THREE YES : Very good. You already have the most important things together. You should now answer the questions in the second questionnaire and then we can talk some more. In any event this book will help you also.

```
************************************************************
*                                                          *
*    Your software library                                 *
*                                                          *
*    1. Cartridges only                          YES/NO    *
*    2. Commercial programs on cassette          YES/NO    *
*    3. Programs you have written                YES/NO    *
*    4. Word processing software                 YES/NO    *
*    5. Data management (data base) software     YES/NO    *
*    6. Bookkeeping software                     YES/NO    *
*    7. Business accounting software             YES/NO    *
*    8. Graphics or synthesizer software         YES/NO    *
*                                                          *
************************************************************
```

Evaluation: If you didn't answer **YES** to any of these questions then something is wrong. Either you don't have a C-64 or you use it only as an ornament. But jokes aside.....

ONE YES ANSWER : You have a computer and are beginning to ask yourself what you can actually use the thing for. Buying this book was a step in the right direction. Read section 1.4.

TWO YES ANSWERS : Your software library has one or more ready-to-run programs. It should be expanded. Refer to sections 2.1 through 2.5 for information on what types of software your should purchase.

THREE OR MORE YES ANSWERS : Your selection of software is quite professional. It may be that you just need a couple of ideas for new applications. Please help yourself!

Good. Now we have gotten to be friends as far as our common areas of interest are concerned and possibly you have even been able to get used to my style of writing. If not, you are going to have bear with it for another 200 pages or so.

1.1 What can the C64 really do?

It would be very boring indeed to simply list the technical specifications of the '64 here. You can get that information out of the user's guide. What would be interesting, however, is to see how these specifications apply to daily life. To this end we are going to have to take a brief look at the insides of our computer. (* Note: technical experts please go immediately to the end of the chapter and read the paragraph there.) As the name implies, the memory capacity of your Commodore is 64 KB (kilobytes), a large portion of which is required for internal processing. You can see this when you turn on the machine and get the message

"64K RAM SYSTEM 38911 BASIC BYTES FREE"

Approximately 39k bytes are available for BASIC programs. Let's take the theoretical case in which all memory positions are occupied by a letter or digit. In this case you could put approximately 15 pages of text in the unused storage of the C-64. Admittedly, not a lot. On the other hand, our example is not very realistic. In the first

place you wouldn't be able to do anything with the stored text because no space would be available for a program which would process the information. Secondly, there are external storage media available, without which a computer is as good as useless. We'll say more about this later.

You might perhaps ask yourself what is actually contained in the other 25,000 storage positions. That is where the clever engineers and programmers have placed the things which give the '64 all its fantastic capabilities.

The first thing there is the BASIC interpreter, which, as the name implies, interprets or translates something. This portion of the system always has a lot to do when programs written in BASIC are running. BASIC stands for Beginner's All Purpose Symbolic Instruction Code. This language is well adapted to human and in particular the English language The problem is that the brain of the computer, the central processor, doesn't understand a word of BASIC. More exactly, it doesn't understand any words or letter, only digits. Not only that, but it knows only 2 digits: 0 and 1. This means that each command or instruction which the CPU (Central Processing Unit) is to execute must be represented with combinations of these two digits.

But don't be afraid when you sit down in front of the keyboard--you're permitted to do more than press 0 and 1. That was taken care of in advance. Not even the engineers that built the C-64 had to use this level of language. That difficult work was taken care of during the development of the central processing unit. The CPU is capable of following machine language commands--the code which the processor understands.

Later, when you understand a lot more about computers and are really interested, you can learn to program your '64 in machine language. You'll be amazed at the speed at which programs can be executed in machine language. But now back to the BASIC interpreter. This interpreter is responsible for telling the CPU what you want it to do when you issue certain BASIC commands.

The interpreter occupies approximately 8 Kbytes, approximately the same amount space that the operating system of the C-64 takes up. "What does all this mean?" you might ask. The answer is simple: Nothing will run without an operating system. The '64 relies on the operating system to tell it which key you have pressed or which data should be read from the diskette. It also wouldn't be able to write information to the monitor or print anything on the printer.

Two of the really big pluses of the C-64 certainly lie in the areas of graphics and music. A thorough discussion of these capabilities is beyond the scope of this book. We will, however, touch upon one or more programming ideas in these areas. We must also recognize the limits as far as ease of use is concerned; if you really want to use everything that the machine offers, it is necessary to use a plethora of PEEKs and POKEs. Many people would rather purchase finished software to do these things.

We have already spoken about the storage capacity of the Commodore 64 but now need to say a little more. It's capacity is ample for most applications. You should of course not get the idea that you can do all of the data

processing of a medium-sized company with such a computer, a couple of diskette drives, a good printer, and a little software. This is simply not the case. On the other hand, if we take a self-employed painter as an example, then the '64 is capable of eliminating or taking over a large amount of nerve-wracking and time-consuming work.

The memory capacity is more than adequate for household use, provided an external storage device such as the 1541 disk drive is used. The system is then capable of managing a phonograph collection file of say 500 titles without difficulty.

If you want to believe the public opinion polls on home computers, then a large portion of computer users use their devices for games. As the performance of reasonably priced computers increases, there is an enormous improvement in the quality of games for them, not only in sound and color but also in the content of the games themselves. At one time, only black and white "Pong" games or simple "shoot 'em up" games were possible. Now fantasy games are available which are so cleverly made that the player can transport himself or herself to a fairy-tale world and attempt almost insoluble puzzles. The C-64 offers all the prerequisites for enjoying such leisure-time fun.

Take the step from being a passive player to an active game designer. You might discover a new creative hobby. You can find some hints and suggestions in the last sections.

*) Remarks:

If you already know what happens inside the integrated circuit chips in a computer then you should read the following section knowing that the descriptions are as both somewhat inaccurate and incomplete. I believe that no beginner is helped by being told what RAM/ROM overlapping is or what the architecture of a CPU is. Anyone who seriously studies the machine will sooner or later become interested in its architecture and can then read any of the standard books on the internal workings of the 64.

1.2 Storing data

What data you want to store naturally depends on the type of program you are using. The internal storage of the C64 is adequate for small applications using a limited amount of data. But what you are going to do if you want to keep track of your color slide collection of over 200 boxes? "No problem," you say. "I'll just use my datasette for data storage and I'm all set." And you're right.

One thing is quite clear. An external storage medium is absolutely necessary, not only to store data but also to programs which you write yourself, since everything in memory is lost when you turn the computer off. In most cases, you will use special data cassettes with your data recorder. These cassettes, in contrast to music cassettes, have only 5, 10, or 20 minutes of playing time on each side. It is recommended that only one programs (including

data) be stored on each side. The reason for this is easy to see. Data cassettes can only be read sequentially, that is, from beginning to end. This means that if the data which we are looking for is at the end, we have to read the entire tape before we can get to it.

The only way to get around this problem is through very clever use of the features of the C-64. It is, of course, much easier to work with a floppy disk from the very beginning.

Ease of use is not everything. Of perhaps greater value is the large storage capacity of a diskette. On a Commodore VIC-1541 disk drive you have 170 Kbytes of storage available. Remember our storage capacity example from a previous chapter in which 38 KB was only 15 pages of text? You can place almost 5 times as many pages on a diskette. The entire text of this book would require only 4 diskettes.

Another very important (to many, the most important) reason for using a disk drive is the much higher access speed. This access speed is so much higher than that of a cassette in part because a diskette has more direct access to all the stored information. For the time being this applies only to programs. All data are stored on 35 tracks, which are further divided into sectors. Each sector on every track has its own name which is entered into the directory of the diskette. This name ensures that the entire diskette does not have to be read when searching for a particular program.

So there are three very important reasons why you need

a disk drive (and later a second drive). Such a storage medium is an absolute requirement for professional use of the C64 and the VIC-1541 is reasonably priced at around $250.

- Ease of use
- Capacity
- Speed

are the major advantages of a disk drive.

If these arguments haven't convinced you, the ideas in the rest of this book certainly will!

1.3 Printers

When you take the first step in the direction of a word processing or data management program, the question of a printer becomes very important since even the best address list program is of little value if it outputs only to the screen. For writing, some type of print-out device is a necessity. You might say that this is much too expensive. But let's take a look at the following simple example.

Let's assume that you own a C-64 and an external storage device. Let's further assume that your old typewriter is about ready for the museum and that you wish to purchase a new electric typewriter. Both assumptions are realistic. You then have two alternatives: a modern electric typewriter which costs between $300 and $450 or a printer which can be used with your '64 and which costs around $300. As a plus you are also getting something with which you can list the directories from your diskettes and which is also capable of printing some graphics.

The choices of a disk drive for the C-64 are very small, but the choices for a printer are enormous by comparison. If we just take the most popular devices into account we still have a choice of about ten printers with different capabilities from which to choose.

-print quality
-speed
-graphics capability

These are the most important criteria when purchasing a printer. In addition, the applications for which the printer will be used must be taken into account.

Let us take the most important possible choices one by one:

VIC-1525/MPS-801. This printer is sold as a companion for the Commodore 64. It is relatively inexpensive and gives fair quality output. It has a true dot-addressable graphics mode and produces readable text.

VIC-1526. The universal printer. This printer is also a companion for the C-64. It came on the market as "the printer" for the C-64 and because of many problems, had to have a new operating system installed. Since that time it can now be recommended because of its excellent price/performance ratio. It is not a true graphics printer and it not widely supported as such. It has somewhat better print quality than the VIC-1525.

EPSON RX/FX-80. The best looking. This printer not only has the best design (admittedly, a matter of taste) but also has exceptionally good print quality for a dot matrix printer. It is also fast and relatively quiet. Unfortunately it is not easy to use all of its 63 (!) print style combinations when it is attached to the C64.

VIC-1520. The plotter. This type of device has no print needles and no daisy wheel. It writes with 4 different colored pens selected through software. The advantage: lines are really drawn as continuous lines and not as rows

of points. Anyone who wants to make full use of the C64's graphic capabilities should consider purchasing a VIC-1520 (perhaps as a second printer).

There are, of course, many more printers but these are the most commonly used in connection with the Commodore 64. In particular, we have not mentioned any daisy wheel printers. These tend to cost significantly more than dot-matrix printers but are now coming down in price. If you need letter-quality (typewriter-quality) output, you should investigate current low-cost daisy wheels now on the market.

1.4 No software, No use

Every home computer owner needs software. We are already aware of that. The difficulty lies in the manner in which we will acquire programs for our various applications. There are three major alternatives:

1) Develop the program yourself
2) Type in a program from a magazine or book
3) Purchase finished software

The decision as to which of these alternatives is best is difficult since such a decision is based on the personal requirements of the user. In addition, these three areas are not totally independent of each other. Let's look at each in detail.

Develop the program yourself

As you become familiar with BASIC, you'll begin writing short programs to test you knowledge of the language and capabilities of your computer. Your first programs will probably be short--no more than 10 or 20 lines per program--and you'll be very happy if the computer does what you wanted it to.

Your expectations will soon rise, however. If you still feel motivated to continue to write programs yourself,

you're probably well on your way to becoming a BASIC programmer. No one can develop super programs completely in a vacuum. You'll require information and ideas from books and computer magazines as well as from other computer hobbyists and computer clubs. Many of the well-known programs were, however, developed in the living rooms of amateur programmers. There is always the danger that you will begin to program for the sake of programming and that the actual application will fall by the wayside.

Type in program listings

You can always type in programs listings that are printed in books and magazines. Some such listings are written for "generic" computers and others may require changes to run on your '64. The amount of programming knowledge to make these changes is slightly less than if you had written the program yourself. There are two reasons for this. First, BASIC is a language with many dialects. The command to PRINT text on the screen at a certain location differs for most home computers. If you wish to take a good program that was written for an ATARI and translate it for the '64, then you will have to inspect the program line-by-line looking for these PRINT commands. Second, many programs use PEEKs and POKEs. It becomes very complicated to translate these commands into equivalent ones for the '64. It may be well worth the effort to translate these programs, but the translation involves considerable effort nonetheless. There is no reason for every computer fan to reinvent the wheel and there also is

no reason not to use other ideas in this particular case. If you stick with magazines which deal only with the Commodore 64, you can of course avoid these problems.

Collections of programs can be found in many different forms. If you do not wish to directly use a program, you can have the chance to learn from the experience of others by studying them. But be careful. Published program listings sometimes contain printing errors which can cause problems for the user.

Someone who has spent a weekend debugging such programs will be much more inclined to purchase well-tested software from a well-known software house the next time. This of course brings us to the next point...

Purchase finished software

Anyone who must make a choice has a problem. The number of programs offered for the Commodore 64 is practically unlimited. Even if we ignore the game programs for this discussion, there are more than enough titles to deal with. There are word processing programs on diskettes, data base systems on cassettes, and graphic tools in plug-in cartridges, and the list goes on, and on, and on. How can anyone decide which package is for him?

Give it a try with the following checklist:

****

Checklist for purchasing software

****

A) Questions

1) For what purpose will the software be used? (see sections 2.1 to 2.6)
2) Of which type of should the storage medium be?
   - plug-in cartridge
   - cassette
   - diskette
3) What is the maximum price which you will pay?

****

B) Check points

1) Study the software tests in the computer magazines
2) Watch software advertisements
3) Pick out those products which might be of interest
4) Gather information about the software products of interest from your local computer shop or from the software house which offers the products
5) Talk to current users of the products
6) Read the handbooks and user manuals on the software
7) Have the computer shop or software house

```
*       demonstrate the product to you                          *
*    8) Compare prices                                          *
*    9) If possible, test the product at home                   *
*   10) Repeat steps 1 through 5 if necessary                   *
*   11) Make final choice of software products required         *
*   12) Take close look at service and support offered          *
*       by the company which is selling the software            *
*       -Can defective program diskettes or disks that          *
*        have been mistakenly erased be exchanged               *
*        either at no cost or for some small fee?               *
*       -Is there some type of telephone hot line               *
*        support that you can call to ask questions             *
*        about the operation of the software package?           *
*       -What type of maintenance plan is available to          *
*        ensure that your are able to receive the               *
*        newest version or release of the software -            *
*        either free or for some reasonable price               *
*   13) Select the software seller who has the highest          *
*       cost/benefit ratio                                      *
*   14) Purchase software                                       *
*                                                               *
*****************************************************************
```

You probably won't go through this list from beginning to end, particularly since many of the points were repeated just to be on the safe side.

We're confident that you will exercise the same care in choosing your software as you did in the purchase of your Commodore. Also consider that a readable and easily understood user manual adds considerably to the success that you have with any software product.

1.5 Television or Monitor?

That might sound like a silly question but it isn't. It's less of a question of preferences than a question of how many bottles of eyedrops your pharmacist is going to be able to sell to you. Of course, as long as you're not really going to do any serious computing, your television set will be more than adequate. But first let's take a look at all the different devices we can attach to our computer:

-black and white television
-color television
-monochrome monitor (generally without sound)
-color monitor with sound

Just a few words about working with a standard television set: The principle disadvantages of using a television stem from the fact that the picture is created in quite a different manner than is the case with monitors. There is no way, for example, to get a satisfactory high-resolution graphics and the picture is also often unsteady. With a normal television set, we have a case in which the displayed pictures constantly change. One type of picture has the visual information which we see and the others are black. You eyes are then continuously being confronted with alternating changes from dark to light, even though you are not consciously aware of it. Eye irritation and tiredness are the result.

This problem cannot be entirely eliminated with the available cathode-ray technology, but it can be drastically reduced. It's no longer possible to argue that monitors are too expensive. A good monochrome CRT display device can be purchased for slightly over a 100 dollars. Color monitors with sound capability start at around 250 dollars.

The Commodore 1702 monitor is one such color monitor which is specially adapted for the '64. The '64 works best with it because it has all the features that you need if you wish to use all the capabilities of the C64, both optically and acoustically.

There are of course many other monitors available and I can't recommend one over the others. Your best bet in choosing a monitor is to have your dealer demonstrate it using a Commodore 64. Which monitor is right for you depends very heavily upon the application for which you intend to use it. The program ideas throughout the book also contain information as to which equipment would be the best for the particular application.

1.6 Helpful Hints

Let's start with the most well known:

Joysticks and Paddles

It's not an exaggeration to say that anyone who has used the keyboard of a computer also knows how to use a joystick. We'll therefore save ourselves the long-winded explanations of what they look like and how they are used.

For our purposes a joystick is only really of interest if we can use it for something other than games. Even then, paddles are sometimes the better device. Joysticks and paddles can be used to select items from a menu.

Joysticks cost between 10 and 40 dollars each. Paddles are, as a general rule, less expensive.

Lightpen and graphics tablets

The lightpen is an input device with many capabilities. Depending on the software, you can use a lightpen to draw directly on the display screen or, as is the case with joysticks, select from a menu. Unfortunately, there is not much software available to make use of light pens, and good light pens typically cost 30 to 50 dollars.

A graphics tablet is an input device (pointing device) which offers the same sort of capabilities as a lightpen, except that one draws or points on a surface other than the screen. The KOALA PAD and the CHALK BOARD are two such devices which may be used with the C-64. Graphics tablets range in price from 50 to 100 dollars and are typically no more well supported than lightpens, although they often come with some type of drawing software. Both lightpens and graphics tablets may be used by your own programs, of course, and it is this aspect that we will be dwelling on in this book.

Care and maintenance

Do you have any idea what can happen when a dust particle gets onto a disk in the disk drive? No? Believe me, it can cause a good number of problems.

In any event, you should try to do everything you can to keep the dust particles from wreaking their havoc, whether it be in the disk drive, in the printer, or anywhere else for that matter. There are dust covers available for all these peripherals, and you should consider purchasing them. Diskettes should be stored in a dust-free (lockable, if you so desire) disk box.

The only two devices which really require maintenance are the disk drive and the printer. Once again, dust and dirt are the culprits.

The read/write head of the disk drive needs to be

cleaned from time to time because dust and dirt can collect there. A dirty read/write head can cause problems during disk accesses and destroy several days of programming or data entry work. Simply put, an ounce of prevention is worth a pound of cure. Purchase head-cleaning diskettes and use them regularly.

As far as the printer is concerned, the dust is created internally to a certain extent. The small fibers, found especially in the perforations of computer paper, are shaken loose by the print head and paper feed and then fall somewhere in the printer itself. There are no special maintenance procedures for printers unless specifically mentioned in your printer documentation. Aerosol cans of compressed air (usually available in photography shops) can be used to blow the dust out. Do not use these spray cans in the areas where the electronics of the printers are installed. The cleaning of these areas should be left to maintenance technicians who should also handle the repair of any defective equipment. A vacuum cleaner can also be used if a small enough attachment is available.

2. Basic recipes

If you've ever seen a good cook book, you may have noticed that in the back there is frequently a list of what are called basic recipes. In most cases, the recipes for the culinary masterpieces in the main part of the book are simply modified versions or combinations of these basic recipes. Now as far as I am concerned, things are no different in computing than they are in cooking: With small changes and skillful mixing we can cook up 1000 programs out of the 3 or 4 basic applications. That is why I chose to call this chapter "Basic recipes." These are the basic categories:

-word processing
-data processing
-computations and calculations
-games

Games are really combinations of the first three points plus sound and graphics, but finding the right combination and successfully creating the audio-visual component is itself enough of an art to rate a separate category. In any event, it is definitely a different application.

In choosing ideas to program, I have tried to follow the classification system outlined above. Only the chapter on music, painting, and design strays from the other ideas discussed. These are more or less a mixed form of the basic applications.

If you now have everything together--your C-64, your 1541 disk drive, your monitor, and your printer, plus a couple of finished meals (such as word processing, data management, or some other software) in the pantry, then grab this cook book, and let's see if we can put together a meal fit for a king. "Bon Appetit."

2.1 Word processing

The term "word processing" is in reality not a very good term to use to describe what is actually taking place. Text processing would be a better description of what actually takes place. Text can be defined as a set of words grouped together in a certain order so that they have meaning. This is still a little abstract, but I think if we see some of the things that we can do with a word processing system things will be much clearer.

-letters
-dissertations
-theses and other term papers
-books, stories, poems etc.
-notes

Each of these applications has its own unique characteristics and problems. One problem which they have in common is that of errors.

Let's go back a couple of years, or more exactly to the year 1 before CK (year 0 is the year in which the correction key for the electric typewriter was developed). Typewriter technology had made many advances. Cramps and strained muscles in the fingers of secretaries had more or less become a thing of the past after the electric typewriter was brought into use. The IBM Selectric typewriter was being delivered to offices throughout the world. At the same time, the world record for typing speed had reached new heights along with the use of correction fluid

because, of course, the faster you typed, the more mistakes you made. These little white bottles, which resembled bottles of fingernail polish to female users (and led to white fingernails now and then) were an absolute must for any desk.

The problem with this wonderful stuff was that it tended to dry out over time. So one had to shake it before using and if necessary add a little thinner. Shaking correction fluid bottles became a necessary skill for anyone who used a typewriter. The time that had been saved through use of the electric typewriter was used up by shaking bottles. Well maybe not quite, but almost.....

But then "Mr. IBM" let go of his bomb. A small orange-colored key made the little white bottle almost as out-dated as dinosaurs. If you found a mistake on the page you just moved the print head back to the mistake, pressed the orange key and afterwards hit the correct character and the mistake was gone. Now that was real progress.

Improvements seemed unthinkable, but in reality there was already something better. The electronic brains, as they was called at that time, had been installed in many large companies during the previous years, spoiled and lovingly tended by High Priests called programmers and system analysts, and were becoming more common. The excuse "Our computer is down" was used to cover up every deadline which was not met within the company. It didn't take long, however, for clever systems experts to come up with the idea that these enormous machines could be used to do something other than number-crunching. And thus began the era of electronic word processing. We can safely omit the

next few years when word processing software could not be purchased for anything less than a four-digit figure.

The important question was this: How would it be if, instead of typing text directly onto a piece of paper, we could display the text directly on a monitor. This proved to be an excellent idea since the electrons fired onto the cathode ray tube to represent the letters had the very excellent quality of being changeable or correctable at an enormous speed. This was not true of the letters or characters hammered onto a piece of paper. The smart inventors all agreed that this was the way it should be. Text should first be typed onto a display screen and after all the mistakes have been corrected should it then be printed on paper.

During this period of enlightenment, the inventors came up with a couple of other things that makes word processing so pleasant for us today:

-formatting
-insertion of text blocks
-search and replacement of texts
-storage of text

Formatting

The writer must at some point decide what the text is going to look like when it is finally printed on paper. You may ask "Don't you see that when you are typing it

in?" The answer is not simple.

There are two major philosophies around which word processing systems are designed. One philosophy is that the word processing system should show everything on the screen exactly as it will appear on paper. This is only reasonable if the monitor is capable of showing everything in exactly the same way as the printer can print it. And at this point we have to say that such programs are very nice, but unfortunately cannot be used on the C-64, whose monitor is not capable of showing superscripts, footnotes, or expanded text such as can be printed on an EPSON FX-80, for example. For the time being we will forget about this particular type of formatting.

The second philosophy, which is used in most word processing packages that can be purchased for the '64, is that the formatting is done during printing. This has one very large benefit which can best be shown through use of an example:

Let's say that you have written a letter with a word processing system. During the input you have paid no attention to the length of the line or to how words should be separated. Your text is now finished and you wish to print it out. The printer of course does not know what it is supposed to do. You will be asked to specify the document format with the following information:

- left margin
- right margin
- top line
- bottom line

- justification (yes/no)
- centering (yes/no)
- distance between lines
- distance between characters
- selection of character set

As soon as you have given the printer the required information (which will also be stored in the computer) you will be able to see the formatted text on the display screen. If what you see does not meet your approval, you can reformat and redisplay until you are satisfied with the results.

Remarks:

My more detailed descriptions relate to the word processing program, TEXTOMAT, which is available from ABACUS Software and which was used to write, format, and print the original version of this book on an EPSON LQ-1500 printer. All of the word processing programs which can be used with the C-64 are more or less similar except in price. Most of the things said about TEXTOMAT hold for other programs as well.

Processing blocks of text

A text block is a portion of text that you define. You can make any portion of text a block simply by marking off that section within the text. Once the text block has been defined, you can manipulate it in a variety of ways:

- delete
- store
- copy
- move

What all these capabilities actually mean and what they can do for us can perhaps best be shown through another example. Let's assume that you are going to use your word processing system to write cost estimates for potential customers. You can imagine that there are many lines and paragraphs which will be exactly the same in every letter which you write. For example the paragraph:

> "This offer, including prices, conditions of delivery, and discounts, is good for 14 days. Please request a new bid at the end of this time."

You can test for yourself how long it takes you to type this paragraph . In my case it took around 20 seconds. This means that if you write 100 letters times 20 seconds you have used approximately 30 minutes of time. But wait. Won't the time that I have saved be needed to read such a block of text in from the diskette? It's not worth my time to store such short blocks of text. Well then, what if we store the entire letter? Good, but what about the particular numbers and prices that are different for each cost estimate. What do we do then?

You can write a cost estimate which contains all the information which is normally included in all letters. This letter should be processed in the normal way and then printed out on the printer. Now let's take a look at the

letter on the display screen. Everything which applies only to this one particular offer should be deleted (including the date). The rest of the letter should be defined as a block and stored. The next time that you wish to make a bid you need only read in the block of text and fill in the blanks with the correct information. Your offer is then complete. If you write 100 letters, this procedure will save you at least 70 minutes.

We'll also learn to appreciate these block operations when working with large manuscripts. This was certainly true as I was writing this book. Often I liked a particular passage so well that I stored it away and later decided to toss the rest of the chapter into the electronic waste basket because I didn't feel it was good enough. What one has stored electronically on the diskette, can be used and reused.

By the way, these block operations don't always have to take place through an external storage medium. They can also take place within the text or on the display screen. Blocks can be moved, copied, and deleted. The possibilities are almost unlimited.

Search and replace

These capabilities are really fantastic! You can search through an entire document for a word or group of words and see in what context it was used.

Another example: Let's say you are working on a highly scientific historical paper on "Internal Navigation in South East Europe" (It's possible). One company played a really important role in this particular area the "Danube Steam Ship Company" (This is a lot worse in German: "Donaudampfschiffahrtsgesellschaft"). Let's say that this group of words, or one long word in the case of the original, would occur over 40 times within your text. In order to go easy on you nerves and the tips of your fingers you would only have to type in a short-hand form, let's say DD, everywhere this group of words would occur. As soon as you finish the first draft you then use the search and replace function of the word processor and let the system search for each occurrence of DD and substitute the name of the company. It's as easy as that. Likewise, in the first draft of this book, each occurrence of the word "computer" was originally only a "C" and was subsequently changed to "computer" in this manner. If you have the time and energy you can count how many times the word computer occurs.

A further application is to check the spelling of words, especially the spelling of foreign words. If you only knew how much trouble I have with that. My method is this: If when I am reading a document and am not quite sure of the spelling of a foreign word, say the German word for cathode ray tube, (Kathodenstrahlrohr), I simply write the character combination KAT* wherever this word occurs. The German word and what I have used as the short form are then written on a piece of paper and checked at a later time in the dictionary. I can then replace every occurrence of KAT* by the correct spelling of the word. One must exercise a bit of care in choosing the short

form to ensure that the same character combination will not be replaced in a word where we don't want it to be replaced.

Storing text

We should mention at this point that every piece of text which you suspect, however slightly, might be used again should be stored. It should also be clear that you need at least one disk drive (preferably two) to really do word processing effectively.

Here are some helpful hints in this area: Don't wait to store your text until everything is finished. Particularly in the case of very long document, it is recommended that you save it every 80 to 100 lines. More than one tired writer has turned off his computer at 2:30 in the morning after having forgotten to save the text that was typed into the computer.

Give filenames to your text that have some meaning. One possibility: Some short form of the actual title plus the word "text" plus the section number. For example, assume your paper is called "Electronic Switching Past and Present," and you have just written section 7.1.1, "Buddhism and Transistors." This would give us a block title of "ESPP.TXT 7.1.1". Write that down in your temporary table of contents which can then be revised from time to time.

Make a backup copy of absolutely everything!!! The

diskette is a very reliable storage medium but it is still magnetic and therefore fallible. Count on it failing. Do you have any idea what happens if you lay a pair of scissors on top of floppy disk? No? Scissors have the unfortunate quality of being slightly magnetic. This magnetic field can ruin any data which is already stored on the diskette. It would be very easy to destroy many weeks' work. You can make this backup copy either daily or when a diskette is full. If you have 2 disk drives it is extremely simple. With one you have to be a little bit of a "disk jockey." The process goes something like this: Insert the data diskette, read some data into the storage area, insert the backup diskette, write the data onto it, delete the data from memory, reinsert the original data diskette, read new data from the diskette, and so on. If this is too much work you should at least get in the habit of writing every bit of text that you have put on the diskette on paper. At least you will have a backup copy in this form.

2.2 Calculating and planning

The computer is right at home any time it can work with numbers. That is its real power: adding, subtracting, multiplying, dividing, sorting, evaluating statistical information, correlations, etc. These operations are performed with lightning speed by the internal workings of the machine. But there is a lot more to it than just that, if the results are to be given to the user in a readable form. It is at this point that we use our calculating programs.

Just to confuse the issue somewhat, it is particularly in this area that the technical terms are very complicated. We will not consider the really professional software packages that are now available for the 64, since anyone who wants to do financial analysis for a medium-sized company is unlikely to use a C-64.

What can calculation programs do?

- Create a spreadsheet
- Do computations within the spreadsheet
- Evaluate the spreadsheet
- Compute statistics
- Print out the statistics
- Print out diagrams
- Do limited forecasting

We then have 3 basic functions which build on each other:

- spreadsheets (bookkeeping tables)
- statistics
- diagrams

In the next section, if everything goes as planned, you will learn the basic functions of spreadsheet calculations and also what you can do with the results. Discussions on management-oriented planning software and other such complex packages will not be found there. Anyone who wishes to know more about these type of systems can read about them elsewhere.

The spreadsheet

Unfortunately the term spreadsheet or table is somewhat abstract. A spreadsheet is a multidimensional table in which certain fields are related to or derived from certain other fields.

A multiplication table, as a very simple example, is two-dimensional. It has a vertical direction, let's say from 1 to 4, and a horizontal direction with the numbers from perhaps 1 to 12. The relationship of the values of the table elements is that the values are the products of the row and column numbers. We then have the following picture

```
**********************************************************
*-----1---2---3---4---5---6---7---8---9---10--11--12*
**********************************************************
*---*
*-1-*
*---*
*-2-*
*---*
*-3-*
*---*
*-4-*
*---*
*****
```

We can then picture a checkerboard grid with 12 * 4 = 48 different fields. In every field from top left to bottom right, we have the values, which depend on the defined relationships. Let's take a look at the finished table:

```
**********************************************************
*-----1---2---3---4---5---6---7---8---9---10--11--12*
**********************************************************
*---*                                                  *
*-1-* 1   2   3   4   5   6   7   8   9   10  11  12*
*---*                                                  *
*-2-* 2   4   6   8   10  12  14  16  18  20  22  24*
*---*                                                  *
*-3-* 3   6   9   12  15  18  21  24  27  30  33  36*
*---*                                                  *
*-4-* 4   8   12  16  20  24  28  32  36  40  44  48*
*---*                                                  *
**********************************************************
```

There are many things you can do with this table:

- row-wise addition
- column-wise addition
- define subsections
- add subsections

It's not necessary to restrict ourselves just to addition. You could also do things like add up each row and calculate what percent of the total sum each row sum is.

You can also go into the third dimension. In two dimensions we had the x-direction (horizontal) and y-direction (vertical) standing at right angles to each other . Going into the third dimension we have to add a z axis which is in turn at right angles to the other axes. We then have an imaginary cube the corner of which is made up of the x, y, and z axes. Dependent on these three axes we

then have a large cube made up of a number of small cubes, each of which contains the result of an operation. Got it?

An actual example should clear things up a bit. Let us take our original matrix, which is two dimensional, defined by x=12 and y=4. We then replace some values. For example, x stays the same and y takes on the values of 5 through 10 instead of 0 through 4. If we now recalculate our table, we have a second page. Then we again take new values for the y-axis and recalculate the table, creating a third. We continue this a couple of more times and eventually we have a pile of pages on our theoretical desk. Let us then also assume that we take our pages to a book binder who makes a book out of them. What we then have in our hands is a 3-dimensional table having z-many pages. The advantages in comparison to a real collection of tables is that we can define relationships across the different pages.

And now to the fourth dimension--I can already hear the cries of protest, but in a computer the fourth dimension is child's play. Even the example we used above still functions. Let us assume that we made not one book but a whole row of books using the process described above. Everything clear? If not, the program ideas in chapter 4 will should make it so.

Statistics

The fact that we went into some detail in the previous section will now help us since tables and matrices are a

good preparations for statistics, and statistics, in their simplest form, are nothing other then a summation of rows and columns. Please note: in their simplest form.

Let's first define the term statistics with an unscientific but useful definition. Statistics is a way of bringing order to a collection of data. This is to our benefit, if after ordering the data, we are able to make some determination about the relationships among the them. For example, we can collect data on birth dates within our circle of friends and from their friends, their children, their grandmothers, their grandfathers and neighbors etc. In addition, we ask what everyone's favorite meal is. We then place all of this previously unordered information into a matrix. Now we can start our game. What percent of those questioned belong to which zodiacal sign? That is to say, which are Aries and which ones are Virgos? What percent of those questioned said that their favorite meal was Welsh rabbit, or what percent of those who have the zodiacal sign Aries like to eat cucumber salad? After our evaluation, we can try to answer the really important question. Is there really any relationship between a person's sign and their favorite food? At least that is what a statistician would do.

At this point you probably say that's nonsense, and in this particular case you're right. But let's replace the sign of the zodiac with age and the favorite meal with a hobby. Then let's expand the number of persons questioned. If you were the owner of a hobby shop, you could probably be able to think of a number of good ways to turn this information into increased sales.

A logical outcome of statistics is a prognosis or forecast. As the name indicates (Greek/Roman: pro=future and gnosis=belief or opinion) we here leave the land of hard facts, particularly if we don't have a large-scale computer of the size used by the public opinion pollsters when they predict election results. This area is worth devoting some time to, but it is not within the scope of this book and we simply do not have the room. Therefore a few comments:

-gather and enter the data
-order the data
-put the data into a statistical form
-evaluate the statistics
-make a prognosis based on the evaluation
-check your prognosis against new data as it becomes available

Charts

After all the difficult abstract things that we have talked about up to now, we are going to do something simple. A chart is nothing more than a graphical representation of data, statistics, or a prognosis. As the owner of a 1541 disk drive, you saw a bar chart containing the sales information of computer companies when the demo diskette was run. I was not surprised that the yellow Commodore bar, accompanied by all sorts of noises, surpassed the A-bar (A for Apple or Atari). In any event you can program such bar charts yourself without needing any special software and they are in many cases the best way to

show results.

Strangely enough, it seems that the marketing departments of the computer manufacturers believe that pie charts are the most effective. You hardly ever see an ad for a new personal computer that doesn't have some type of pie chart. It's very effective. The entire pie represents 100% and the individual pieces show the percentages relationship to the whole. Without additional graphics capabilities you should not try to show your household budget per pie chart. The required PEEKS and POKES would probably ruin your appetite.

Line charts are another favorite. They have the drawback that if you try to represent more than four or five lines at a time they are very difficult to read. Lines or curves do however lend themselves very well to showing changes and values which are dependent upon time.

If you are looking for some suggestions on how to set up your charts then you should take a look at television. There has been a good deal of drawing going on there for some time (see the remarks about the predictions for election results above). This type of graph can be represented very well on the 64.

Diagrams and graphs are the real show part of any dry program. With good graphics you can very often leave a good impression and make a real show out of relatively weak data. You know the old saying "There's no business like show business."

2.3 Storing and sorting data

What are data? In this case, let's just pretend that we can't even count to 3.

Why is it that we actually speak of data in the plural? What is it called in the singular? Let me tell you (a little tongue-in-cheek): Date. Ah, now we have a clue that might lead to a solution of our definition problem. Friday the 13th, April 1984, the 1st of May or 10/11/52 are all terms with which we are familiar. We have found the prototype for data.

The most important property of our example is that each date is different from the others. In the first place the word itself is different. This is clear; we have different days. In the second case the format is different. If we were to try to put dates in a programmable appointments calendar in this chaotic form, the computer would probably start coughing and tell us in no uncertain terms to "REDO FROM START." Really good programs eliminate this problem and accept every possible date format. We learned something in spite of this: data that are going to be processed must usually be in some type of predefined format.

Except for these, there are not too many rules we have to follow in data processing, except for one very important point: Never try to put in more data than the memory capacity can handle. In the worst case, this leads to garbage data. Everything is there, it's just a question of it being in a usable form.....

Oh, yes, what then are data? Let's first give a loose definition--everything which can be printed out in computer format through the use of alphanumeric characters (the letters of the alphabet plus numbers). These include names, addresses, titles, prices, dates (in the sense of day, month, year--a series of numbers), shoe sizes, hair colors, cubic inches, storage capacity, open invoices, notes, appointments, calories, etc., etc....

As you can see, there are few limitations. We now must bring some sort of order to this data, and not just in the input format. What we use to bring order into the situation is called a file. This is a universal word. Whenever we have more than two data items, we have a file. By using files, it is possible to store information onto a diskette, for example, and to retrieve it again. There are different types of files. It's easier to explain the different file types by using examples rather than from a theoretical standpoint.

Simple files built from one- or more dimensional arrays are easily set up and processed. Only simple programs which most people with a beginning knowledge of BASIC can write are required to handle these types of files.

The basic elements of file processing are as follows:

- file creation
- file maintenance
- sorting of files
- data selection

Even though these activities are more or less obligatory they may be best explained by looking at the workings of a commercial data management program such as DATAMAT. Other data management packages perform functions similar to those described for DATAMAT.

File creation

The first question that you should ask yourself is, "What kind of information do I need to keep track of?" The second is "What data do I need to give me this information?" Let's take an example. Say you wish to keep information about the birthdays of your extremely large circle of friends. You call your file "Birthdays" and decide, after giving the matter some thought, to include the following information in every record:

- relationship (uncle, aunt, son...)
- first name
- last name
- street number
- zip code
- city
- telephone
- date of birth
- ideas for presents
- affection level (rich uncle, evil mother-in-law, god child, etc.)

You check the list of data items to make sure that you haven't forgotten anything and then take a look at the

order in which they occur. The list could be arranged better but it will do for now. Next you have to design the screen input format.

Perhaps you wish to begin by placing an attractive border on the screen. Next you ask yourself--where am I supposed to put all the information from my old card file? I can make a suggestion. Let's take a look at the input form that I have designed for you:

```
************************************************************
*                                                          *
*                        Birthdays                         *
*                                                          *
************************************************************
*                                                          *
* 1) Date                 : 11/21/1950                     *
*                                                          *
* 2) First name           : Anne                           *
*                                                          *
* 3) Last name            : Smith                          *
*                                                          *
* 4) Address              : 99 Main Street                 *
*                                                          *
* 5) City/State/Zip       : Seattle WA 98011               *
*                                                          *
* 6) Telephone            :                                *
*                                                          *
* 7) Relationship         : cousin                         *
*                                                          *
* 8) Affection            :                                *
*                                                          *
* 9) Gift                 : red roses                      *
```

* *
****

The program will now ask you for the name of the input format as well as for the name of the file. It's usually a good idea to use the same name for both. You can, of course, use the same input format for two different files. This will probably not happen in our case unless you decide to trade in all your relatives. As soon as the input format has been saved, DATAMAT will ask which fields of the format are going to be index fields, i.e. which fields will later be used as search criteria. You should take three things into account when making this decision:

- A shorter index field allows more records to be stored in the file.

- An index field that is closer to the top of the input form allows for a faster search.

- A field to be indexed should contain information in each record.

At this point, DATAMAT will display the maximum number of records that can be stored in the file. You are then asked to enter the number of records that you are actually going to enter into the file. Try to be a little bit miserly here; give only the number that you think you are actually going to need. Don't say 650 just because the maximum is 657. No one has that many relatives.

File Maintenance

Your birthday file is defined and you can now continue with the processing. You can now begin to enter all your data which should take about an hour and a half. Now I can take a break.

Are you finished? Good, then I can continue with my explanation. You now have the capability to search through your file and select something of interest. You can do this by selecting "search" from the menu. A search goes quickest if you search using an index field. The index field in a birthday file could most reasonably be the month. Why the month? Just consider that the simplest use of the file is that on the last day of each month you take a look at who is going to have a birthday in the following month. Suppose that today is the 30th of May. You therefore search the file for records whose index = 06 (for June). After a couple of seconds the name of the first relative who has a birthday in June is displayed on the screen.

The program now wants to know if it should search further. If so, you probably have another uncle or aunt who has a birthday in June. Then the next record will come up on the screen and so on until everyone with a birthday in June has been displayed.

And how do change and delete work? If you have a record that needs to be changed or deleted already on the screen then you can simply select either change or delete from the menu. Change works in the same fashion as input.

Deleting you should try out when you have your own DATAMAT or other data management software.

Sorting files

Now things begin to get interesting. Up to now, everything that we have done with our computer file we could have done by hand on 3x5 cards. Let's say you want to select the records of everyone who has a birthday in November and who is supposed to get a gift of a new tie. "No sweat," you'll probably say. You just shuffle through the cards, stopping at those with a birth month of "11." Next you see if the gift idea is a "tie" and if so, you pull the card out of the stack since it meets your search criteria.

But what if I'd also like to see the name of all my aunts who have a birthday in June. What, still not finished? Right, you first have to replace the 3x5 cards that were picked from our first search. If your file is on the computer, you don't have to do that.

"So what?" you might say. "That's something I would very seldom have to do." With a birthday file this is probably true. But what would you do if you had a customer file of a 1000 names and needed similar searches? The advantage is that such selections can also be saved as a pointer file and called up again when required.

The sorting of a file is also handled much more efficiently with a computer. To sort a file, it may be done

in several steps.

- sorting by month
- within the month by days
- within the date by first names
- end

You can now sort on the first names by relationship. Because the computer sorts in alphabetic order it is also possible to get the following: date : 11/21, first name: Mary

1) cousin Mary
2) mother Mary
3) niece Mary
4) grandmother Mary
5) sister Mary
6) aunt Mary
7) great-grandmother Mary

This is not exactly what we wanted, but it can still be very useful to have a file in which the records can be sorted by all the different fields. We'll come back to this later.

Data selection

We will brush over this quickly. To select from a file means that we are going to print out only the selected

data records in a particular form.

Your printer will be able to give you a lot of help in getting the right format. You can for example use the full 80-character width as well as the control codes to get emphasized, small, or proportional letters, etc.

You will also be able to specify where on the page each field is to be printed. Titles can also be printed. For our birthday file we have chosen the following output format.

```
Birthday: (MM),(DD),(YY)
First name, last name
Address
City, zip
Telephone
Gift idea
```

Consider: You are going to use this format to print a new list every month. You can either use it to print the list for the following month or the entire year. This has the drawback, however, that if any changes are made you must print a whole new list for the year each time.

You can also use this part of the program to print out the address labels which are to be placed on the gift packages.

So far so good. DATAMAT can do all that but how about other data management packages? I think I can put your mind at ease. Most of the data management packages for the C-64 offer similar capabilities. They differ only in their

ease of use, processing speed, price, or availability. If you use the check list for the purchase of software (section 1.4), you'll be able to select the right one.

2.4 Music

To be really truthful, I love music in almost every possible form: from Wagner to Mozart, from Elvis to the Beatles, rock'n roll, reggae and funk. I like to listen to almost everything and in spite of that I am very unmusical. Playing "Mary had a Little Lamb" on a child's flute was about the best I could do and now I am supposed to make music with the Commodore 64. But have courage. I'll get started:

"The starting address of the SID is 54272 (Dec) = $D400 (hex). The low-byte can accept values from 0 to 255, the hi-byte from 0 to 255. You can adjust the tone of voice 1 in register 19 from loud (0*16) to soft (15*16)."

You just read "Music for Beginners", the first in a series called "The Commodore 64 as Synthesizer." This is more or less what happens to us when we try to learn to make music by using the C-64 user's guide. It looks more like the code used by a secret society whose members already know exactly what they want to do.

But don't let yourself be scared away. There are some fantastic aids that are available to make beautiful sounds with the 64. For example, packages such as ULTRABASIC-64 and VIDEO BASIC-64 contain simple commands for producing music. Or dedicated musical packages such as SYNTHY-64 and SYNTHIMAT and be used even by professional musicians.

2.5 Painting, drawing, and designing

I don't wish to repeat my previous comment about the readability of the C-64 handbook for beginners, so I'll only say that the same holds true for graphics applications.

You may have read that the '64 offers many capabilities in the area of graphics. For example the HIRES mode (high resolution) is capable of displaying 320 x 200 individual points. If my calculator is working correctly that's 64,000 dots on the screen. You should be able to draw some really fantastic pictures if you only knew how. This leaves us with two recommendations: either buy a graphics expansion package or a good book on '64 graphics.

Because painting and drawing are more in my line of work than music, we won't go quite as quickly as we did in the previous section. There are a couple of things about graphics which we need to discuss. However, they aren't so difficult that you'll need a university degree to understand them.

It is possible to draw some very attractive pictures in the standard character graphics mode (using the graphics characters on the front edge of the keys on your '64). Drawing like this takes practice and patience. Up to 1000 (40 characters per line x 25 lines) graphics characters can be displayed on the screen. There is an equivalent number of storage locations for these 1000 characters (called screen memory) and a corresponding color memory

location for each character.

The principle for screen display is very simple. The screen memory is similar to hotel keyboxes that contains rows and columns of individual boxes in which unused keys are placed. In this case, we have two sets of keyboxes with each set having exactly 1000 boxes. One set of keyboxes contains the character to be displayed and the other set of keyboxes contains the color of that character.

When the computer displays a picture on the screen, it does so very systematically. First it looks in box number 1 of the first set of keyboxes and takes the character that it finds there. Then it goes to the second set of keyboxes which contain small paint cans. The computer then takes the character which it brought, dips it into the paint can in box number 1 of this second set of keyboxes and finally attaches it to the inside of the monitor screen (we're beginning to get somewhat silly again). This process is repeated by the computer for box number 2, box number 3, and so on until all 1000 boxes have been completed.

The usual way of placing characters into the individual boxes is through PRINT commands in a BASIC program, but you can do it much faster by POKEing the box numbers with the character's code and color. Of course we don't want to make this too simple, so the boxes are not numbered from 1 to 1000 but from 1024 to 2023 (for the characters) and from 55,296 to 56,295 for the colors.

One question always asked by graphics fans is "How can I use high resolution graphics?" As is the case with many

other things in the computer, this is a storage problem. Since one byte (8 bits) is required to store the information for every screen character position, it is only possible to define a total of such 1000 locations. There is simply no room for any more. Unfortunately, a very simple character which displays only a single pixel still requires 1 byte to hold the information just as does a complex character. This is because each character really consists of an 8x8 bit format in the following form:

```
01234567
********  1
...**...  2
...**...  3
...**...  4
...**...  5
...**...  6
...**...  7
........  8
```

Every character of the character set can be represented as an 8 x 8 matrix, including the graphic symbols. But in hires mode, characters are not required. Each pixel on the screen is represented by one bit in memory. But wait! Let's do a little more arithmetic: 8 bits = 1 byte; 1024 bytes = 1 kilobyte and 64,000 bits = ? kilobytes. Right, a little less than 8 kilobytes. In standard character mode, 1000 bytes are required for the picture and 1000 bytes for the colors. So this storage area is too small for high resolution graphics. The storage area used for high resolution graphics move to a larger area.

More detailed information on the points discussed above can be found in the book The Anatomy of the Commodore 64.

2.6 Do-it-yourself programming

It's possible that there is nothing you would rather do less then discuss this topic. I can well understand that since I too must admit that I would rather work with finished software. In particular I don't want to write my own word processing or data management software.

A strange thing happened the other day. I was taking a look through a computer magazine when I saw a article which really caught my eye and in which one particular word really jumped out at me.

*** F O R T H ***

That sounds like force. It aroused my curiosity. I was really surprised when I found that the article was about a programming language. In spite of this, I read the article further and quickly came to the conclusion that this was something for me. I'm now going to leave the story at the most interesting part and talk about something else for a minute. The following programming languages are available for the 64:

-BASIC (built in)
-PASCAL
-LOGO
-FORTH
-ASSEMBLER

There are of course many many more programming lang-

uages (COBOL, FORTRAN, C, LISP, ADA, to mention only the most used). But only the ones listed above can be run on the C-64 (the ADA Training Course [with compiler and editor] is now available from ABACUS Software and a C compiler is forthcoming). We won't discuss BASIC any further since information on BASIC will be found in other chapters.

PASCAL

Pascal is by its very nature a learning language. That is to say, it lends itself very well to learning how to program. You don't have to suffer under the illusion that programming consists only of typing in lines of code. Quite the contrary, using Pascal, a greater portion of the work involved is in the preparation.

- program concept
- program design
- flow chart
- the design of input formats
- file format and list of variables
- output design
- program documentation

These are the principle points of program design that must be taken care of before typing in the code. Pascal supports this type of structure through the fact that a good portion of the activities listed above are a portion of the Pascal program itself.

You might even say that structured programming is a hallmark of Pascal. Programs retain their clarity because of the structure forced upon them by the programming language. For example: Pascal does very nicely without GOTO statements, something that can lead to a good deal of confusion in BASIC programs.

In addition to this, Pascal programs run quite quickly and are sometimes better for professional applications than BASIC programs.

LOGO

The name is clever, and so is the language. Anyone who has ever heard about LOGO usually thinks about it in terms of "children and computers" and this for good reason. LOGO is a language with very few commands. No one has to use short hand instructions as in BASIC such as CLR, LEN, SYS, etc. Instead you can use simple words to control the computer. The only drawback is that LOGO programs tend to be very long.

This is also the case with so-called "turtle graphics." Turtle graphics is not a real language. The turtle is nothing more then a fancy cursor which draws directly on the display screen. LOGO and turtle graphics actually go very well with the '64 but I have never tried them out.

FORTH

"At last!" you're probably saying now. And after you have read the next couple of lines you may be disappointed because you are not as enthusiastic about FORTH as I am. But let's wait and see.

FORTH is

- easy to learn
- fast
- and universally applicable

The heart of FORTH is its library of commands. It is very small. This means that FORTH does not require much room in the memory of the computer. On the other hand, because you can do very little with only a couple of instructions (or the programs must be unusually long) the designers came up with a simple trick. The user can expand the library of commands of FORTH at any time to meet the requirements of his program.

For example, FORTH has no command to raise numbers to a power. This command must therefore be defined externally. FORTH can multiply which means that we only have to define raising to power as a series of multiplications. It's as simple as that. If a program is going to do only arithmetic operations, your library will require very few output commands. If you're writing a word processing program, your library won't require many arithmetic commands. The FORTH program never has to scan through a large library to find a command.

It is in any event worthwhile to take a closer look at FORTH. FORTH programs can run up to 10 to 30 times faster than a similar BASIC programs.

ASSEMBLER

Programming in assembly or machine language is probably as difficult as trying to determine a precise definition for the word. You have probably heard or read that some people program in machine language and have a slight suspicion that that has something to do with assembly language. It is probably more correct to say machine language but let's not worry about semantics.

There is no such thing as a machine language in a global sense. A machine language is always for a specific microprocessor. There is a 6502 machine language, a Z80 machine language, and a 68000 machine language. Machine language contains instructions that the processor can directly understand or interpret. This of course leaves something to be desired as far as the understandability for us humans, and that's the big problem. Another problem is that you can only really use a machine language if you know every little detail of the processor. Programs written in machine language do, however, run faster than in any other language.

If you are going to write your own programs you should at least be watching machine language out of the corner of your eye. Even when programming in BASIC you can make use

of small machine language routines (these are sub-programs in machine language). PEEKs and POKEs will also take you in the right direction.

Computer users who write their own application programs, independent of which language they use, always have to ask themselves the question: Is the time required to develop my own programs really worth the effort? Of course you get more into the routine with experience. You can also develop standard subprograms which can be used again in other programs. This reduces the amount of time required. Until you reach this point however, it can take a couple of weeks, months, or years. If you really want to fully utilize your Commodore 64 from the first day on, you should probably go both ways, packaged software for day-to-day use and programs which you have developed yourself for learning, fun, and later, more specialized applications.

3. Writing and printing

Before continuing, I would just like to say a couple of words about monitors. If you are going to do a lot word processing with your '64, I highly recommend that you get a monochrome monitor. You'll save yourself a lot of trouble with your eyes, particularly if you stick with the green or amber monitors. The resolution is much better than the color monitors and monochrome monitors are considerably cheaper.

Some word processing packages allow you to change the colors of the text and background on the monitor. This may be a suitable alternative to a monochrome monitor if you already have a color monitor. At the present time I am writing with yellow on red. You'll have to try it out to find out which combination you like the best.

One last word about the requirements: I have always shown very economical configurations. With word processing it is particularly difficult to give any general sort of recommendations about printers. The old saying still goes: No matter how good your printer prints, there is always another one that prints better. And the question of daisy wheel printers, yes or no, is really a question of how thick your wallet is.

3.1 Party invitations

```
############################################################
*            Requirements                                  *
############################################################

############################################################
*            Hardware                                      *
############################################################
*                                                          *
*            Commodore 64                                  *
*            Datasette, disk drive is better               *
*            Monochrome monitor                            *
*            Dot-matrix printer                            *
*                                                          *
############################################################

############################################################
*            Software:                                     *
############################################################
*                                                          *
*            None                                          *
*                                                          *
############################################################

############################################################
*            Prerequisites                                 *
############################################################
*                                                          *
*            Good knowledge of Commodore BASIC             *
*                                                          *
############################################################
```

"Hey, let's throw a really fantastic party!"

"Yeah! Who should we invite?"

"Well, let's see. Fred and Joan, Bob and Eva, the Smiths and their son..."

"Wait a minute. Let's go about this a little bit more systematically. Get a piece of paper."

"By the way just where does Fred live these days?"

Later:

"Well, that's over one hundred people. If we are going to invite all of them there is going to be an awful lot of writing to do."

Stop!!

What do we have a Commodore 64 for anyway? Writing a lot of very similar letters is a classic application for a home computer.

Question: Do you already have your address book in the computer? If so, then you can skip the next section. Otherwise, read on.

3.1.1 A simple address file

Just a few words before beginning. This program is primitive and not particularly user friendly. It's meant to get you started and give you some ideas.

First question: How many addresses are going to be

entered?

Second question: How many fields will each record contain?

Let's say we are going to have 100 records (addresses). Each record will have the following fields in our example:

-first name
-last name
-street address
-city, zip code

That is a total of 4 fields per record. One record will be called D$ in the program. All together this gives the line of code:

```
100 DIM D$(100,4)
```

First thing we need to do is to write our input routine:

```
100 DIM D$(100,4)
200 REM INPUT
210 Z=1
220 INPUT "FIRST NAME";  D$(Z,1)
230 INPUT "LAST NAME ";  D$(Z,2)
240 INPUT "ADDRESS   ";  D$(Z,3)
250 INPUT "CITY/ZIP  ";  D$(Z,4)
260 Z=Z+1
270 PRINT  :  INPUT "MORE INPUT (Y/N)"; A$
280 IF LEFT$(A$,1) = "Y" THEN 220
290 END
```

A short explanation: Z counts the number of records and is incremented by 1 after each record is entered. INPUT of fields 1 to 4 is restricted to a total length of 255 characters. The shorter the fields, the faster the program will run later. The END statement at line 290 is just temporary. Later we will write the statements for saving the data.

The following program module "DATA STORAGE" is for use with a datasette. The changes necessary to use this code with a disk drive will be presented later.

```
290 REM DATA STORAGE
300 OPEN 1,1,1 "ADDR FILE"
310 FOR N=1 TO Z
320 :    FOR M=1 TO 4
330 :        PRINT#1,D$(N,M)
340 :    NEXT M
350 NEXT N
360 CLOSE 1
```

The OPEN command should not be completely new to you. If it is, you should review it in user's guide. The FOR/NEXT loop with index variable N counts the records records and loop M the fields. The OPEN command for the disk drive is as follows:

```
300 OPEN 1,8,2,"ADDR FILE,S,W"
```

Good. At this point, I will to let you start entering your names and addresses while I go and get a cup of coffee. But don't forget to rewind your tape if you are storing data onto a cassette.

If you want to be able to get at your data again you need a routine which differs in only two ways from the input program. There must be a new OPEN command:

```
400 OPEN 1,1,0,"ADDR FILE"
```

and where the PRINT# command is in line 330, should be the line

```
430  :    INPUT#1, D$(N,M)
```

Try it out. As you can see, you won't see anything. The addresses have been read from the cassette tape, but no one told the computer that they should be displayed on the screen. This is something that we can have the computer do with the following simple loop.

```
490 REM OUTPUT/DISPLAY
500 FOR N=1 TO Z
510 :    FOR M=1 TO 4
520 :        PRINT D$(N,M)
530 :    NEXT M
540 :    PRINT  :  PRINT
550 NEXT N
```

As you can see we are always doing very similar things. The N/M loops for storing, loading, and displaying are nearly identical. The OPEN commands are the primary difference. Since the program section for printing data onto the printer also contains a loop, we might just want to make some improvements.

Wouldn't it be nice if these individual sections of code could RUN one after the other. That would be great. So let's design a menu with this in mind.

```
90 REM MENU
100 PRINT "MENU"
110 PRINT : PRINT : PRINT
120 PRINT "INPUT ADDRESSES    =1"
130 PRINT "STORE ADDRESSES    =2"
140 PRINT "LOAD ADDRESSES     =3"
150 PRINT "DISPLAY ADDRESSES =4"
160 PRINT "END PROGRAM        =5"
170 PRINT : PRINT : PRINT
180 INPUT "YOUR CHOICE";C
```

A menu looks good and is easy to program. I have also added the routine END PROGRAM in order to have an emergency exit to which I can go without losing data or the program.

Without having said too much about it, I have included a couple of improvements and also written the subroutine that begins on line 500. I hope this is all right with you. The following listing contains the entire program for use with a 1541 disk drive.

If you're using a datasette you can use the above program modules for storing and loading. You will see that inputting data from the diskette is not so simple. We have to check the internal status variable ST. This shows the end of the file. If we were to use the normal loop (N,M) the disk drive would not know where the end of the file

was and sooner or later would stop, not knowing what it was supposed to do next. No error message would be given.

The procedure should then be used as given. It is also universally applicable for every type of sequential file.

And now at last the program listing

```
10 rem * * * * * * * * * * * * * *
20 rem *      address file       *
30 rem * * * * * * * * * * * * * *
40 rem
50 dim d$(100,4)
60 poke 53281,08 : poke 53280,08
70 poke 53272,23 : print chr$(158)
80 rem
90 rem
100 print "" : print "MENU" : print
110 print : print
120 print "Input Addresses   =1"
130 print "Store Addresses   =2"
140 print "Load Addresses    =3"
150 print "Display Addresses =4"
160 print "End Program       =5"
170 print : print
180 input "Your Choice >";w
190 on w gosub 200,300,600,400,700
195 goto 100
200 rem input
210 print ""
220 z=z+1 : print "Number :         ";z
230 input"First name ";d$(z,1)
240 input"Last name  ";d$(z,2)
```

```
250 input"Addresses  ";d$(z,3)
260 input"City/Zip   ";d$(z,4)
270 print : print : input"More Input";a$
280 if left$(a$,1) ="y" then 220
300 print ""
310 open 1,8,2,"addr file,s,w"
340 for n=1 to z
350 :   for m=1 to 4
360 :       print # 1,d$(n,m)
370 :   next m
380 next n
390 close 1
395 return
400 print ""
410 input "Printer or display (P,D) ";a$
420 if a$="p" then open 4,4,2 : cmd 4
430 print ""
440 for n=1 to z
450 :   for m=1 to 4
460 :       print d$(n,m)
470 :   next m
480 :   print :print
490 next n
500 if a$<>"d" then 520
510 close 4
520 get a$ : if a$="" then 520
530 goto 100
600 open 1,8,2, "addr file,s,r"
610 n=1
620 input #1,d$(n,1),d$(n,2),d$(n,3),d$(n,4)
630 if st<>64 then n=n+1 : goto 620
640 closel
650 z=n
```

```
660 return
700 rem end
710 print ""
720 print "end"
730 for i=1 to 1000 : next i
740 end
```

Just a couple of further comments. The color for this display will, as you know, be set by the POKEs in line 60. The POKE in the next line will select the upper and lower case letters. A safety question can also be used in place of the delay loop in line 730:

```
720 input "should program really be ended (y/n)" ; in$
725 if in$="n" then 100
730 end
```

If you answered the question with no, control goes back to the menu. Now let's take a look at a word processing system to go along with what we have already done.

3.1.2 A very simple word processor

That's short, sweet, simple and particularly user friendly. In other words really easy. Let's give it the name:

****Easy Text 64****

With Easy Text we will be able to write a maximum of ten pages, each with 20 lines per page. You will be able to correct and look at your text as well as be able to read from or store onto a cassette or diskette. You will also be able to print your text out. Easy Text contains an interface to the address program shown above. It is written in the form of modules so you can easily modify and improve the program with a bit of imagination. I would be more than happy to see any improved versions.

```
10 rem* * * * * * *program start* * * * * * *
20 poke 53280,0 : poke 53281,0
30 poke 53272,23 : print chr$(30)
35 dim t$(200)
40 print chr$(147)
50 print "* * * * * * * * * * * * * * * *
60 print "*                               *
70 print "*       EASY TEXT 64            *
80 print "*                               *
90 print "* * * * * * * * * * * * * * * * *
99 rem  ********
100 rem * menu *
101 rem ********
```

```
110 print : print : print
120 print "Input text   =1"
140 print "Change text  =2"
150 print "Store text   =3"
160 print "Load text    =4"
170 print "Display text =5"
180 print "Print text   =6"
190 print : print :"End          =7"
200 print : print  :print
210 input "Your Choice  ";w$
220 w=val(w$) : if w>0 and w<8 then 240
230 print "": goto 220
240 on w gosub 300,600,800,1000,1200,1300,9000
250 goto 40
299 rem ***********
300 rem *  input  *
301 rem ***********
310 clr : print chr$(147);
320 print "";
330 input "Name (max. 15 characters) ";tn$
340 if len(tn$)>15 then print "" : goto 330
350 input "maximum line length  40";zm
360 if zm<10 or zm>40 then print "": goto 350
370 print "A page may have at most 20 lines."
380 input "max # of pages (10)";sm
390 if sm<1 or sm>10 then print "": goto 380
400 bl=1: zl=1 : sp=1
420 print chr$(147);:print "Page:";spc(7);"Line:"spc(8);"Column:"
430 print "#####################################" : print;
440 print "F1 = new line/new page ";"F2 = stop";
450 p1=1030 : p2:1043 : p3=1058
460 poke p1,(int(bl/10))+48:poke p1+1,(bl-(int(bl/10)*10))+48
470 poke p2,(int(zl/10))+48:poke p2+1,(zl-(int(zl/10)*10))+48
```

```
480 poke p3,(int(sp/10))+48:poke p3+1,(sp-(int(sp/10)*10))+48
490 print chr$(95);chr$(157);
495 get in$ : if in$="" then 495
500 if in$=chr$(137) then 590
505 if in$<>chr$(20)then 525
510 print chr$(32);chr$(157); : sp=sp-1
515 print chr$(157);chr$(32);chr$(157);
520 t$(zl)=left$(t$(zl),len(t$(zl))-1) : goto 450
525 if in$=chr$(133) then 560
530 if in$=chr$(13) then 550
535 t$(zl)=t$(zl)+in$
540 print in$;
545 sp=sp+1 : if sp<zm then 450
550 print chr$(13); : t$(zl)=t$(zl)+chr$(13)
555 zl=zl+1 :sp=1 : if zl<20 then 450
560 print : print : print "    *** page full ***"
570 get g$ : if g$="" then 570
580 bl=bl+1 : if bl<sm then 450
590 got 40
599 rem *************
600 rem *  changes  *
601 rem *************
610 print chr$ (147);"";
620 print "";
630 input "which page    ":in$
640 bl=val(in$) : if bl>20 then print "" : goto 630
645 print chr$ (147);: print"Text: ";tn$; : print " page: ";in$
650 zl=((bl-1)*20)+1
655 print t$(zl)
660 if t$(zl)<>"" then zl=zl+1
670 if t$(zl)="" or zl>20 then 690
680 goto 655
685 print : print
```

```
690 input "which line " ;in$
695 in=val(in$)
700 if in=0 then 790
705 zl=((bl-1)*20)+in : k$=""
710 poke 214,14 : sys 58640
720 print t$(zl) : print
725 print chr$(95),chr$(157);
730 get in$ : if in$="" then 730
735 if in$=chr$(137) then 785
740 if in$<>chr$(20) then 760
750 print chr$(32),chr$ (157)
755 goto 725
760 if in$=chr$(133) then 780
765 k$=k$+in$
770 print in$;
775 got 725
780 t$(zl)=k$
785 print
790 input "Other changes (y/n) n";in$ : if in$="y"then 600
795 return
799 rem **********
800 rem *  save  *
801 rem **********
805 u$=""
810 print chr$(147),
820 input "drive ready  n";in$
830 if in$="n" then print "": goto 820
840 print " ";tn$;" will be saved "
850 open 1,8,2,u$+tn$+"s,w"
860 for n=1 to zl
870 :   print #1,t$(n)
880 next n
890 closel
```

```
900 rem
910 rem
915 open 15,8,15
920 input#15,ff,fb$,sp,se
925 close15
930 if ff=63 then u$="@:":goto 850
940 print "Save/load complete" : for i=1 to 1000 : next i
950 return
999 rem  **********
1000 rem *  load  *
1001 rem **********
1010 clr : print chr$(147);
1020 input "Name (max. 15 characters) ";tn$
1040 print " ";tn$;" being loaded "
1050 open 1,8,2,tn$+"s,r"
1060 z1=1
1070 input #1,t$(z1)
1080 if st<>64 then z1=z1+1 : goto 1070
1090 close 1
1100 print "Save/load complete" : for i=1 to 1000 : next i
1110 goto 40
1199 rem *************
1200 rem *  display  *
1201 rem *************
1210 print chr$(147);"";
1220 input "which page  1";in$
1225 b1=val (in$) : if b1>20 then print "": goto 1220
1230 print chr$(147);:print"text:";tn$;:print" page: ";in$
1235 print
1240 z1=((b1-1)*20)+1
1250 print t$(z1)
1255 if t$(z1)<>"" then z1=z1+1
1260 if t$(z1)="" or z1>20 then 1280
```

```
1270 goto 1250
1280 input  "Other pages (y/n) n";in$ :  if in$="y"  then 1200
1290 return
1299 rem ***********
1300 rem *  print  *
1301 rem ***********
1310 print chr$(147) ;
1320 print "The text with name"
1330 print " ";tn$;" ";"is printing."
1340 print
1350 input "Printer finished n";in$
1360 if in$="n" then print "": goto 1350
1370 open4,4,2
1380 zl=1
1390 if t$(zl)="" then 1440
1410 print #4,t$(zl)
1420 zl=zl+1 : goto 1390
1430 print#4
1440 close4
1450 return
8999 rem *********
9000 rem *  end  *
9001 rem *********
9010 print ""
9020 print "*** E N D ***"
9030 print : print : print
9040 input "Text saved   n";in$
9050 if in$<>"y" then 800
9060 end
9999 rem * * * * * * * program end * * * * * * *
50000 open15,8,15
50100 input#15,ff,fb$,sp,se
50200 close15
```

```
50300 printff;fb$sp;se
```

3.2 The professional thesis

```
##############################################################
*            Requirements                                    *
##############################################################

##############################################################
*            Hardware                                        *
##############################################################

##############################################################
*            Commodore 64                                    *
*            1541 disk drive                                 *
*            Monochrome monitor                              *
##############################################################

##############################################################
*            Software                                        *
##############################################################
*                                                            *
*            Word processing program                         *
*            (i.e. TEXTOMAT)                                 *
*                                                            *
##############################################################

##############################################################
*            Previous requirements                           *
##############################################################
*                                                            *
*            Experience in use of the                        *
*            the C64 keyboard and disk drive                 *
*                                                            *
##############################################################
```

Just so there are no misunderstandings, you can exchange the word thesis for any other type of written work which you may have. It doesn't necessarily have to be any kind of a scientific paper or written presentation. Even a report on the trip taken by your bowling club may be large enough to require the use of a (half) professional word processing program. From my own experience I can say that once you have gotten used to working with a word processing system you won't be able to imagine how you ever did it before.

This chapter is not meant to explain every move you have to make to be able to use a word processing system. That is to say no sentences such as "Take the program diskette in your left hand and open the door to the disk drive with your right." Information pertaining to the operational details of the word processing program are to be found in the program's manual. What I am trying to accomplish here is to pass on a couple of tips and tricks with which you should be able to get around problems that might occur, and also how to make your work a little bit easier. Detailed discussions are concerned with TEXTOMAT from Abacus Software because that is the program with which I wrote this book and with which I have the most experience. Most information does however apply to other word processing systems for the C64.

Familiarization phase

When you insert the program disk into the disk drive for the first time, you will not be very surprised by what happens and will be easily able to manage it--if you have your user's manual. Go quickly to the write mode and start typing.

Before you begin with serious writing however you should know the main functions of the program. You should have tried:

-correcting a single character
-deleting a line
-inserting a line
-moving around in the text
-use of control characters

Stop! Control characters are a good bridge to the next practice session. The control characters are there to control the format of the text when it is printed. If you are a TEXTOMAT user you can practice formatting. It is not necessary to print a finished and formatted text each time. The display function allows the written text to be displayed on the monitor in the same format as it will later be printed. You should already know about the important parameters for formatting from our discussion in section 2.1. My tip is: Try out every possible combination. Then you can store the formats for later use.

We now need to concern ourselves with the printer. You may perhaps have purchased the device for some work that

you are going have to do immediately. That makes it very simple. You learn how to use your printer along with the word processing system. But this is only the case if you are in the lucky position of having a word processing system for which you do not (independent of which type of printer you have) have to create a table. TEXTOMAT is from a software standpoint already prepared for the Commodore printer VIC 1525, and, in case you have one of these printers, you need do nothing more. For the VIC 1526 and other compatible printers there is also little to do. You only have to store a table which is prepared in the program (look in the user manual under utility programs). For all other printers, every character of the Commodore character set must be brought in line with the respective printer character. Fortunately this is a one-time thing.

In addition to this initial adjustment, a list of control characters must be filled out and the secondary addresses of the printer have to be set. The RS 232 interface may need to be adjusted (if you already have a 64 you know what is meant by this).

Once you have tried everything out we now come to:

The preparation phase

One must be well prepared for big jobs. You know that just as well as I do. The actual work with text requires some effort. In comparison to working with paper and pencil or typewriter, there are some differences when using electronic word processing. I don't wish to make any

hard and fast rules. I know only 2 basic types of word processing users. Those who do the entire job at the terminal and the others who write out the text by hand and then enter it into the computer when it is finished.

Both types have their good and bad points. The last method may save a good deal of difficult work in front of the display screen. The other method uses much less paper and you can make a hard copy of the text any time you wish. I myself belong to the last group but this is not the main reason why this method is the first one which should be considered. It's just that this is the only way in which the entire spectrum of possibilities can really be fully utilized.

Before you begin to write, do you already have clear in your head what you wish to accomplish? Or at least a clear concept of the order in which things are to be written? This can be extremely helpful. You should also write a preliminary table of contents. It is a particularly good idea to do this in its final form, that is to say, with the correct chapter numbers and the correct order of the main points. This table of contents then grows with the completed work. Any changes in chapter headings or among the different parts can then be immediately changed in the table of contents. The idea is that when you have finished inputting your text, your table of contents is also finished. This type of index is also a good structure to use for naming the different files on the diskette.

You probably have your glossary and your references in the form of a card file. Have you ever thought of putting these in the computer? Take a look at section 5.11. The

footnote file is actually nothing more than a bunch of notes. TEXTOMAT, for example, has an interface to sequential files. You can take the remarks in your text directly out of such a file.

One last point: use your computer as a tool and continue to use your normal work methods. If you don't, you'll be spending more time with the computer than you do with your actual work.

The work and writing phase

How you write, what rhythm you have, whether you prefer to write at night or in your bathing suit, which radio program you like to listen to while you are working, are all dependent on you (I myself prefer country). But there is one absolute in word processing, one thing which you must do, and that is to save your text and save it often.

Here I am really speaking from experience: One not so happy day I was really in good form and my fingers were just flying over the keyboard. Everything that I was writing had style and class (that's enough of complimenting myself...). In any event I sat at the terminal from noon until about 2 o'clock in the morning and had only saved the first 10 to 12 pages. What came next, of course, was bound to happen. Night-blind, tired, and not thinking quite straight, I wrote the last few words, took a look at what I had done using the display function, and turned off the computer, completely certain that I had saved everything for posterity on the diskette. And of course every-

thing was gone. Frustration, anger at the computer and myself. Since then I have gotten in the habit of doing 3 things:

- saving the document about every 100 lines.
- creating a new text file every 200-300 lines if this agrees fairly well with the chapter divisions.
- making sure that at the end of each sitting, everything up to that point has been saved and then creating a backup diskette. This backup diskette then gets a write protection tab placed on it. These are the foil strips which are fastened over the little rectangular holes on the edge of the diskette.

I would like to give a short explanation of this procedure. The storage methods which I first mentioned should overlap. That is, the first 100 lines are saved and then the 2nd 100 with the first 100 under the same name and so on. It is then almost impossible for anything to be lost. There are two reasons for my recommendation about the length of the files: First, 300 lines are approximately 100 blocks of the diskette or 6 files per diskette. This is a good usable amount. Enough was mentioned in section 2.1 about backup copies. As far as the order of the points mentioned above is concerned, it is just one method of eliminating errors in handling. I use the following procedure:

- opening ceremonies, up to going into the write mode.
- write documents with saves at regular intervals, continue to write.
- save, look at what I have done and make corrections.
- repeat the above until everything is okay.

- make final save, print out, check the contents of the diskette, if everything is okay make a backup copy
- Put a write protect on the backup diskette
- Put the program, text and backup disks in a safe place. Then when all that is done, quit.

You might be thinking that this is a good deal of work, but what sounds like hours of moving diskettes back and forth is in reality relatively easy. And when you become accustomed to it you can do it without having to think about it a great deal. It is in any event a lot easier than rewriting an extremely long document. Did you notice by the way that the work in progress stays in memory until the computer is turned off? This is also an important safety factor.

End phase

The text is complete, you are satisfied, and now you want to print out your text. Two things may now be different depending on which type of word processing system you have. The one is the automatic numbering of the pages, the other is the insertion of footnotes and remarks. The page numbering can be processed through the use of printer control characters in some programs. In other cases the numbers must be placed on the pages by hand (very frustrating). Footnotes can also be written by hand. A better method is of course to use a footnote file. This is possible in TEXTOMAT through use of a control character which is used to chain files. The text, which is then stored on the same diskette, can be loaded and appended to the main

text during printing by use of this function. A word of warning: If you wish to see the footnotes on the display screen, you have to be careful that their total length does not exceed the room which is available in the computer's memory.

If you have your document finished and printed out (several copies), you can relax and enjoy the fruit of your labors. There is however something that you should still do if you wish to save your work for posterity, not only on paper but also on diskettes. Go through the backup procedure one more time and in the following order:

1) printer parameters
2) the formats and the order in which they were used
3) the text files from first to last

That can, depending on the length of the job, also take a couple of hours. But it's the best way to ensure that your text has been saved in such a fashion that even after everything that has been printed out has turned to dust, you still have a way to correctly reconstruct your work.

Now I'd like to wish you a lot of inspiration and good grades. If you have any questions, please write to me. I am thankful for any criticism (compliments) and every new idea.

3.3 Form letters

```
#########################################################
*            Requirements                               *
#########################################################

#########################################################
*            Hardware                                   *
#########################################################

#########################################################
*                                                       *
*            Commodore 64                               *
*            1541 disk drive                            *
*            Monochrome monitor                         *
*            Correspondence-quality printer             *
*                                                       *
#########################################################

#########################################################
*            Software                                   *
#########################################################
*                                                       *
*            Word processor                             *
*            Data management program                    *
*                                                       *
#########################################################
```

```
#######################################################
*           Previous Requirements                     *
#######################################################
*                                                     *
*           Use of keyboard and disk drive            *
*                                                     *
#######################################################
```

And now you are again going to be asked to play a role, something which will probably not cause any great difficulty for you, even if you have gone directly from the front of the book to this chapter. You are to pretend that you have own a toy shop specializing in model railroads.

By the way, did you really start with this chapter? If so then please go back and read section 2.1. You'll save yourself and me a lot of repetition.

In your new role, direct correspondence with your customers has a very important function. It often occurs that you have been able to dig up collections of rare locomotives or old time models. You also know that some of your customers will do almost anything to get one of these rare models. What you must do is write to them when you have something to sell.

Using conventional methods you have 2 possibilities:

- You write each customer individually (possibly writing to each with an personal letter).
- You write your offer, copy it and then fill in the address by hand.

The first possibility has the advantage that you can offer the addressee exactly what he is looking for. You must on the other hand do an awful lot of typing (or have it done). The second variation has certain advantages as far as speed is concerned. How would it be if we took the advantages of both methods and combined them? It's possible.

The prerequisite is a good, up-to-date, customer file. That's what we are going to talk about first.

3.3.1 Multifunctional customer file

You can create such a file using DATAMAT from Abacus Software. First, the input format (if you want to see exactly how this is done, take a look at section 2.3). For our example I would recommend the following format:

```
------------------------------------------------------
Lukas-Games and Toys                Customer File
------------------------------------------------------
( 1) Last name          :  .Smith                .
( 2) First name         :  .John                 .
( 3) Title              :  .Mr.
( 4) Street address     :  .11 Main St.          .
( 5) City               :  .Seattle              .
( 6) State              :  .Washington           .
( 7) Zip                :  .98011.

        Customer Information

( 9) Last order on     .30/.2/.84.
(12) Last offering on .29/.6/.83.
(15) Open invoice numbers .       .
(17) Specialization: Pre-war Maerklin HO-Locomotives
(18) Sells          : Pre-war HO-Locomotives of  every
                      type
```

The block "customer information" is of course dependent upon the particular type of business. In the case of our shop it is interesting to know what customers might themselves sell particular pieces of equipment. Date of the last order or last offer can also be used as a search criterium. DATAMAT will now begin to set up the file. The input format shown has a total of 208 characters, index field is 1 (the last name). All in all we can have 432 records as the maximum for our customer file. That of course is not enough if we assume 2000 customers. All we have to do then is to set up 5 files of 400 records each using the same input format. To keep things simple we will distribute the customers through the different files al-

phabetically:

- A to F
- G to K
- L to P
- Q to U
- V to Z

This distribution may prove to be impractical in daily use. It could be that the distribution by area of interest would be more practical.

The selection of customers according to their special wishes will also be taken care of. And now we can begin with the input of data. We have 2 possibilities to get the first list: First we can get a hard copy, using the f8 key (print screen), of every record which was input or we can get a complete list through the appropriate menu selection.

Let us assume that for the time being, a list of names and addresses of the customers will be sufficient. We will then make the selection so that only the required fields are shown, with 2 lines per record and 60 characters per line:

```
*Smith                  John
*11 Main St.            Seattle 90811
```

With enough blank lines between the records, the list is quite easy to read. Using this list we now check the data to make sure that it is correct.

The most important work for full use of the customer file will be done by using the menu selection "Sorting." You set up and store as many pointer files as you wish.

The main target of our selection will be the different areas of interest of our customers. For example, we can place all the customers who are interested in purchasing pre-war Maerklin models in one file. In another we can place all of those who wish to complete their collection of N-gauge models, etc.

We can also place those with payments due in one file and those who generally buy very large orders in another. The only limits here are your requirements and your imagination.

I recommend that when you begin to use DATAMAT you try out everything that could possibly be of interest to you.

3.3.2 A variable text file

"A letter is a letter," I can hear you say. "Why do we need a file?" I'll let you in on a secret. There is no exact, reliable information, but my own personal experience has shown that out of 100 business (perhaps advertising) letters, at least 30 are made up of standard paragraphs. Not only "yours truly" pops up all the time, but also longer paragraphs (see section 2.1). This is true even if we select our letters from totally different firms operating in different areas. How many portions of your

business letters are repetitive? There are secretaries who know the text of the letters their bosses are going to dictate before they have even opened their mouths. You might protest that you want to send individualized letters to your customers, but that is exactly what a good word processing system can do.

Let's stay in our role and take a look at a typical offer that a model railroad shop might send out:

****
John Smith
11 Main St.

Seattle, Washington 98011

Dear Mr. Smith,

During our last visit to the international model railroad exchange in Luzern, Switzerland, it was possible for us to purchase a large collection from a Swiss hobbyist.

Included in the many items of the collection are some pre-war Maerklin locomotives and cars which will certainly interest you. In particular, the following items:

1 XZY, serial no. 4711
2 ZYX, serial no. 1510 and 8623
15 freight cars of different types
11 second class passengers cars from the 009

series

All pieces are in mint condition.

Our previous correspondence with you leads us to believe that the items offered would fit very well in your collection.

If your are interested in purchasing the items offered, you may, without obligation, request photographs of the above items.

Because of possible interest in the purchase of the entire collection, this offer is only good until the 15th of July. We will be looking forward to your reply.

Yours truly,

Lukas Model Railroads

[Signature]

****

Let's divide the letter into its individual parts. First of all we have the address, which can be taken directly from the customer file. The first portion of the letter is too specific to be stored. This is also the case with the next 3 paragraphs. The fifth portion of the letter is the first one that might occur in other letters, but it is almost too short to make it worth the trouble. You might be thinking that the last paragraph is also not one which would occur very often. The bottom line: This

letter is not one which will be used again.

But wait. Let's assume that the collection also contains a couple of HO Old-Timer car models in which other customers might be interested. To let them know, you simply take the +letter, delete the third paragraph and replace the information on the pre-war Maerklins with a sentence about the Old-Timers. Then make corrections to the list and to the information on the models and you have your second letter finished. Let's take a look:

****
Henry B. Miller
705 Columbia Street

Alexandria, Virginia 22314

Dear Mr. Miller,

During our last visit to the international model railroad exchange in Luzern Switzerland, it was possible for us to purchase a large collection from a Swiss hobbyist.

Included in the many items of the collection are some !!HO-Car models from 1937 to 1940 which will certainly interest you!! In particular, the following items:

!!Plus:
Cars, Buses and trucks, some with historical ads--and of special interest--a 1940 VW Beattle from Pirsche in Camstatt.

All pieces are in good condition and require only slight restoration!!
etc. etc.

Yours truly,

Lukas Model Railroads

[Signature]

****

It won't be necessary to go through the rest of this in detail. But I think you have seen how few changes will be required to the original letter. In spite of this, the second letter is as individual as the first one (The changes are indicated by the exclamation points).

What we have here is a classic example of what I would call the variable extended letter, in which a form letter is changed to meet the requirements of the addressee. The other case in which word processing software comes into use (in addition to the standard letter which we are not going to discuss here) is the phrase file.

Everyone uses phrases in repetitive forms of address. In some cases we are forced to do so (for example where there is a question of liability or legal requirement to do so). The question which we must ask ourselves is: "Are these blocks of text numerous enough to warrant storing them as an individual file?" Now we have to leave our role playing. Something that is important for our model rail-

road dealer Lukas might be completely useless for your purposes. You must yourself know which phrases and text come up often enough to make it worthwhile to save them. It can be worthwhile to maintain a file of phrases and standard terms for form letters that can be modified to meet individual requirements.

3.3.3 Bringing text and data together

Let's stay with our example of the information letter about the collection which we purchased. The first step would be to write a standard letter. There are two possible strategies: Either you write a form letter with free areas that can later be filled in to meet individual requirements or a letter which is intended for a particular group of customers and which can later be modified for another interest group. How you do it is up to you. Many people do not like to write without something to go by.

You should now select the customers to whom you wish to send the offering. You either do that through the pointer file or, if your customer file is not too large, directly out of the file.

A prerequisite for the first method is of course that the pointer file can be easily identified. The name P1 does not give much indication of what its contents are. A name such as "old maerklinHO" communicates much more in-

formation.

And now we can begin to use our printer. The manner in which the name and address are inserted in the letter is explained in the TEXTOMAT user manual. Simply go through the chosen pointer file and select (or change) the required information.

The rest consists of folding, placing your letters in the envelopes, and mailing. By the way: if you use envelopes with windows, you can avoid sending letters to the wrong address, a not uncommon occurrence when using labels. This type of error can be very embarrassing in the case of individual letters.

3.3.4 Conclusion

Our model railroad dealer Lukas has a large need for individual advertising letters; you perhaps do not. It could be however, that you send very similar letters to many addresses. If this is the case, the use of a C64 with word processing and database management programs may be worthwhile. A couple of additional ideas:

- Create your standard forms (invoices, dunning letters, bids, or offers) using the word processing program.

- Maintain your library of standard letters with the data management program.

- Keep an up-to-date hard copy of your standard letter library with exact information on where the text files are stored.

- Keep copies of the letters that at first glance seem a bit too individual to save. It may be that you can use parts of them later.

Since this chapter is considerable longer than originally planned, I will not make any additional comments. But you will certainly gain experience through day-to-day use that goes beyond the scope of what we have discussed here.

4. Computations and planning

In section 2.2 we discussed the particular advantages of the computer as far as speed, ease of use, and accuracy go. As in every case where the machine can do something particularly well, the question always arises, "How can I best utilize these capabilities?" This chapter will give you 5 answers to this questions. As you probably imagined, money will play an important role. It is simply this: All control of financial matters is a matter of talent. I don't have this talent and that's one reason why I have a C64 which helps me compensate for my deficiency.

In cases such as the construction costs calculations shown in section 4.3, the computer can actually pay for itself. Imagine that through the use of this program you are able to get financing which saves you 2000 dollars over the next few years. Wouldn't that be worth it?

The popular household bookkeeping systems are also real savers. Anyone who knows where his money is going will soon know where he can begin to save. This is also the case with automobile costs. Using information from tests in the car magazines, you can compute the approximate cost per mile for the various models that you are considering buying. This can be an important help in making a decision.

One more thing: All programs are intended as suggestions. If you can, you should experiment and change them until you have something which exactly meets your personal

requirements.

4.1 Household bookkeeping - Hints for designing your own system

You are probably shaking your head and saying "Here comes another bookkeeping system for the 64." There is actually no excuse except that there are many different starting point for an electronic household bookkeeping system. Most of the program listings you can get your hands on act as if the normal household should be administered in the same way as a small company. There are account numbers and discussions of credits and debits to the point where you begin to get dizzy. And most of these programs are based on accounting software. This is one possibility, but let's take a look at another.

The simple way to a financial overview:

The main point is this! Household bookkeeping is there so that we know where every dollar came from and where it went. The principle is simple but there are individual differences. Below you will find a check list which will help you to design your individual household bookkeeping system.

```
**********************************************************
*                                                        *
*          Check-list for household bookkeeping          *
*                                                        *
**********************************************************
*                                                        *
* A) Questions                                           *
*                                                        *
* 1) Number of regular sources of income?                *
*    (5 or less/more than 5)                             *
*                                                        *
* 2) Number of regular payments?                         *
*                                                        *
* 3) Number of sources for intermittent income           *
*    (10 or less/more than 10)                           *
*                                                        *
* 4) Number of payments made on a regular basis          *
*    (10 or less/more than 10)                           *
*                                                        *
* 5) Should all sources of income and payments           *
*    be summed for the total year?                       *
*    (Yes/No)                                            *
*                                                        *
* 6) Will amounts be entered more than once per          *
*    month?                                              *
*    (Yes/No)                                            *
*                                                        *
* 7) Are statistics required                             *
*    for the monthly sums? (Yes/No)                      *
*    for all payments? (Yes/No)                          *
*                                                        *
* 8) Are the statistics to be represented in the         *
*    form of graphs? (Yes/No)                            *
```

```
*                                                    *
* 9) Are the bank accounts to be automatically       *
*    controlled? (Yes/No)                            *
*                                                    *
******************************************************
```

Before actually beginning with the check list you should first evaluate the questions. A decision for the smaller values in question 1 through 4 should be counted as No.

Nine **Yes** answers: You are in the wrong place here. You don't need a household bookkeeping system. What you need is a full-fledged accounting system. In your case it would be best to purchase a standard software package.

Four or five **Yes** answers: If the number of your yes answers is primarily a result of the number of transactions, you should continue reading. Information on how to make your own simple household bookkeeping system can be found in this section. On the other hand, if you answered the questions 5 through 9 with yes, then you should continue with section 4.1.2 after you have finished the check points below.

Less than four **Yes** answers: You are very modest and your program will accordingly be easy to write. Take a look at the check points below. The following check list is based on the idea that household bookkeeping can be built as a relative file. This method seems to me to be the most universal. You will find a complete listing in the Anatomy of the 1541 Disk Drive written by Englisch & Szczepanowski (an Abacus Software book). The program is

completely documented and well explained. "What do we need a check list for?" you might say. The answer: In order to give you the possibility to individualize your book keeping system. For example you should be able to decide for yourself whether you want to do without all the extras and just have a simple input/output module as a program or if you want to design a book keeping system with all the bells and whistles (statistics, graphics, forecasting, etc.).

Points 1), 2) and 5) are, independent of form, mandatory for every household bookkeeping program. Point number 3 enhances the perspective and point 4 makes the results more readable. If you opt for the basic version, you do not even need a menu. The order in which the individual parts of the programs are executed is built into the programs themselves. You will see how you can put all this together in the next section.

```
**********************************************************
* B) CHECK POINTS                                        *
*                                                        *
* 1) Determine individual items                          *
*    - determine number (maximum approx. 40 chars)       *
*    - determine name (maximum 20 characters)            *
*    - define the maximum length of the amounts          *
*      (for example the largest amount is                *
*      $100,000.00 = a field length of 9 chars)          *
*    - determine the input and output items              *
*    - number the items                                  *
*    - create a list of the items with their #'s         *
*                                                        *
* 2) Monitor input/output                                *
*    - input item number                                 *
*    - the item name will be output                      *
*    - amounts for all 12 months will be output          *
*    - input month                                       *
*    - input amounts (positive or negative)              *
*    - new amount will be shown                          *
*    - all items for each month will be output           *
*    - year total for each item will be output           *
*    - the month total for each item will be output      *
*    - the year total for all months will be output      *
*                                                        *
* 3) Statistics and graphics module                      *
*    - the output shown above will be shown in           *
*      the form of graphs                                *
*    - as line charts                                    *
*    - as bar charts                                     *
*    - the amounts will be shown as a percentage         *
*      of total sums in the form of graphs               *
*      - as pie charts                                   *
```

```
*        - as bar charts                                    *
*     -  future values will be forecasted and extrapo-*
*        lated in the line charts                           *
*                                                           *
* 4) Printer output                                         *
*     -  as in the display input/output                     *
*     -  additional output for all items                    *
*     -  all months and all sums in one print pass          *
*                                                           *
* 5) Storage/recall                                         *
*                                                           *
* 6) Menu                                                   *
*                                                           *
*************************************************************
```

4.1.1 And this is how it looks in practice

```
#########################################################
*            Requirements                               *
#########################################################

#########################################################
*            Hardware                                   *
#########################################################
*                                                       *
*            Commodore 64                               *
*            1541 disk drive                            *
*            Monochrome monitor                         *
*                                                       *
#########################################################

#########################################################
*            Software                                   *
#########################################################
*                                                       *
*            None                                       *
*                                                       *
#########################################################

#########################################################
*            Previous requirements                      *
#########################################################
*                                                       *
*            The more, the better                       *
*                                                       *
#########################################################
```

You know the threat hanging over your head? That I will attempt to explain to you what a relative file is. (We are just starting with a little joke). A relative file is something different than a sequential file. You have seen the term relative file in this book several times. You know what sequential means--one after the other. Let's take a look then at the following example of a sequential file:

```
***Fruit Price List***

1) Apples             0.65 /lb
2) Pears              1.03 /lb
3) Strawberries       2.85 /lb
4) Dates              4.20 /lb
5) Grapefruit         1.30 /lb
6) Raspberries        2.70 /lb
7) Ginger             1.85 /lb
8) Red currants       2.50 /lb
9) Cherries           2.65 /lb
10) Mandarines        0.50 /lb
11) Nectarines        1.05 /lb
12) Regular oranges   1.25 /lb
13) Plums             2.15 /lb
14) Quince            3.05 /lb
15) Gooseberries      1.05 /lb
16) Grapes            3.00 /lb
17) Walnuts           1.00 /lb
```

After we have created this list with a small program,

we come to the really dramatic portion: storage on a diskette. The line numbers are only there for output and display and will not be saved. You want to see how it looks when stored on a diskette?

```
apples*0.65pears*1.03strawberries*2.85*dates*4.20*
grapefruit*1.30*raspberries*2.70*ginger*1.85*red
currants*2.50*cherries*2.65*mandarines*0.50*
nectarines*1.05*oranges*1.25*plums*2.15*
quince*3.05*  etc,  etc.(the "*" indicates the RETURN at
the end of each input)
```

The file "Fruit Price List" is in sequential form. We don't just want to store the file but we also want to be able to bring it onto the display screen. Simple. Within the program is contained the instruction to print the name of the fruit in column 4 and the price in column 20. No sooner said than done. The list shown is displayed on the CRT. Now let's say you wish to change a price--the one for grapes. You switch to input mode, move the cursor to column 20 and change the price and then save what you have just done. Simple.

Now consider the following situation. The price list contains not 17 but 1000 different articles and they don't fit on the display screen - possibly not even in the computer's memory. The file must then be brought into the storage one piece at a time. Each record must be compared to the search criteria and then the next piece of the file read and so on.

Unfortunately, if grapes are the 999th record, the search can take quite a while.

Is it possible to tell the computer that it only has to look from record x to record y? This idea has just occurred to you. You are on the right path. This is the principle of a relative file. Every record has its own position within the file which is indicated by its relative distance from the beginning of the file (that is the source of the name). "But what happens if the records have different lengths?" you might say, "How can that be accounted for?" A valid point. This is why a relative file may not have different record lengths. If necessary, all records and fields must be brought up to the standard length by padding with blanks.

The fruit price list for a relative file on a diskette then looks something like this:

```
*01*apples        *0.65*02*red currants  *2.50*
```

The records and the fields which they contain all have the same number of characters and are numbered. Every record can then be called up by use of its record number. The record number is equal to the distance of that record from the beginning of the file since all the records are of the same length. To make that clear we will show one more example:

Let us assume that the diskette is like a board in a shelf and on this board there is a line drawn every five inches. The departments which we have thus created are numbered:

```
/01----/02----/03----/04---- /05----/
```

We are now going to put bananas in these different compartments. Sequential bananas, which all have different lengths.

```
Banana 1 = /++++++++++++/
Banana 2 = /++++++/
Banana 3 = /++++++++++++++++++++++++++++/
Banana 4 = /++++++++++++/
etc.
```

Only the second banana fits into a single compartment, the others are all too long. Let's say that the computer is now to look for banana number 3 and we have told it that every banana is in a compartment by itself (we have a relative file). The computer will go to compartment number 3 and you can see for yourself what happens.

```
Contents of compartment 3 = llllll (approx. 4 in. banana)
```

The contents of compartment number 3 are approximately 4 inches of banana. This may not be tragic for the banana but as far as the amounts in our household bookkeeping system are concerned we can get some interesting results such as $5.00 July D5 (or something similar).

The basic principle of a relative file is clear, as are the advantages. How then is the interested user to install such a file? At first glimpse it does not seem to be possible, since the BASIC 2.0 that is implemented on the C64 does not have any commands for the management of relative files. We can still do it, however. It is a little complicated, but possible.

The Anatomy of the 1541 Disk Drive mentioned above shows exactly how a relative file can be built and managed. I will concentrate here only on the basics.

First we must consider how we are going to define the individual records of our bookkeeping file (we will call this file HHB from now on). We have to choose between 2 variations:

- one record = one month
  resulting in 13 records
  (one for each month plus one for the year sum)
- one record = one account

The method chosen has little effect either on the speed of access or on the ease of use. For our example let us take the second variation. The record will then be defined as follows:

1) Name of item	(20 characters)
2) January amount	(10 characters)
3) February amount	(10 characters)
4) March amount	(10 characters)
5) April amount	(10 characters)
...	
11)October amount	(10 characters)
12)November amount	(10 characters)
13)December amount	(10 characters)
14)Year total	(12 characters)

The record length is then 153 (20 + 12*10 + 12 + 1 for RETURN). The total number of records is equal to the

number of accounts. Just for the sake of argument let's say that we have 36. We then open a relative file with the following command:

```
OPEN 1,8,2,"HHB,L"+CHR$(153)
```

The L indicates to the operating system of the disk drives (DOS = Disk Operating System) that HHB is to be opened as a relative file. The CHR$(153) indicates the length of the record. To free all the records to be written, the last record must contain CHR$(255). Without this procedure each record must be individually freed after input.

In the case of HHB the records do not have to be written in order from 1 to 36, but can be written in any order. To ensure however, that the rent for January isn't put in the salary for March, the write head must first be properly positioned. That is, a pointer must be set to the correct field and the correct record. This is done with the following command.

```
PRINT#2,"P"+CHR$(2)+CHR$(0)+CHR$(20)+CHR$(1)
```

Let's explain these fields one after the other. "P" is the command for positioning; The value in the first CHR$ is the channel that will be used for transferring the data, in this case 2, because channel 1 will be required for the file. The next 2 CHR$'s contain the record number. It must be distributed between 2 CHR$'s because the largest value which a CHR$ can contain is 255 and a relative file may contain up to 65535 records. Since we will remain under 255, the first of these 2 CHR$'s does not concern us, it remains 0. In the second we place a 20. This means

that we wish to do something with record number 20. The last CHR$ contains the number of the character within the record that we wish to read or write. It should always contain 1 since we can only process one entire record if we start with the first character. Let's make a practical program out of our theoretical explanation:

```
100 NR=36: REM NR = NUMBER OF RECORDS
110 OPEN 1,8,2,"HHB,L,"+CHR$(153)
120 OPEN 2,8,15  : REM COMMAND CHANNEL FOR POSITIONING 130
130 PRINT#2,"P"+CHR$(2)+CHR$(0)+CHR$(36)+CHR$(1)
140 PRINT#1,CHR$(255)
150 CLOSE 1 : CLOSE 2
```

The file is opened and 36 records of 153 characters each have been freed. The game starts.

I am now going to leave you with this information. If you have read the entire chapter up to this point you are probably capable of writing a program which will meet your requirements. If this is not the case I would again recommend the Anatomy of the 1541 Disk Drive.

4.2 Figuring auto costs

```
#########################################################
*            Requirements                               *
#########################################################

#########################################################
*            Hardware                                   *
#########################################################
*                                                       *
*            Commodore 64                               *
*            1541 disk drive                            *
*            Monitor                                    *
*                                                       *
#########################################################

#########################################################
*            Software                                   *
#########################################################
*                                                       *
*            None                                       *
*                                                       *
#########################################################

#########################################################
*            Previous requirements                      *
#########################################################
*                                                       *
*            Good to very good knowledge                *
*            of BASIC                                   *
*                                                       *
#########################################################
```

It is a good thing that I don't drive my car as chaotically as I write BASIC programs, otherwise my car insurance payments would be the size of a small mountain. Confused listings may have made a few people mad, but no one has ever been injured by them.

A short time ago, during a trade of software, I proudly gave a fellow computer hobbyist a program I had written myself, called "Auto Costs". After many tries at getting the thing in gear, he gave up and hasn't spoken to me since. That's too bad; he was really a nice guy. But how could he know that the program had an allergic reaction to an input of 0 and would always answer with a "DIVISION BY ZERO ERROR." The only way to get around this was to use an input of 0.001 instead of 0.

I have however, learned from my mistakes. First I read some books on professional programming and secondly, I took what I read there and put it into practice. You can see the results below.

The program idea

The program "Auto Costs" is intended for the collection and storage of all costs associated with a private automobile. Additionally, the total cost and the cost per mile will be computed on a monthly and yearly basis. The results will then be stored in a sequential file.

Problem analysis

What should the program accomplish?

1) Data input
2) Data computation
3) Data output
4) Data storage

The program flow is then as follows:

1) Data input
2) Input month number
3) Output month data
4) Input item number
5) Input amount
6) Compute new data
7) Output new month data
8) Compute total amounts for year
9) Store data
10) Exit program

A menu is not required. The sequential file is organized as follows:

-13 records (for the 12 month and the total year)
-each record containing 9 fields

1) Gasoline cost
2) Repair
3) Care and maintenance
4) Reimbursements

5) Total mileage
6) Tax
7) Insurance
8) Total costs
9) Cost per mile

Input mask:

```
 00 ******************************************************
 01 *                                                    *
 02 *              Automobile Costs                      *
 03 *                                                    *
 04 ******************************************************
 05
 06  Monthly output for (month)
 07
 08
 09 (1) Gas                    =$  ..............
 10 (2) Repair                 =$  ..............
 11 (3) Care/maintenance       =$  ..............
 12 (4) Reimbursement          =$  ..............
 13 (5) Total mileage          =Mi ..............
 14
 15 (6) Tax                    =$  ..............
 16 (7) Insurance              =$  ..............
 17
 18 (8) Total costs            =$  ..............
 19 (9) Cost per mile          =$  ..............
 20
 21
```

```
 22 (( Input Line ))
 32
 24
```

File Design:

Filename	: dat.autocosts
Program name	: autocosts
File type	: seq.
Records	: 13
Fields	: 9 each
Record length	: 65 characters

The fields:

No.	Contents	Type	Length	Name
1	Gas	N	8	ak(m,1)
2	Repair	N	8	ak(m,2)
3	Care/maintenance	N	8	ak(m,3)
4	Reimbursement	N	8	ak(m,4)
5	Total mileage	N	6	ak(m,5)
6	Tax	N	7	ak(m,6)
7	Insurance	N	7	ak(m,7)
8	Total costs	N	8	ak(m,8)
9	Cost per mile	N	5	ak(m,9)

Required formulas:

1. Tax and insurance

Question:
"Amount of tax = $?" and "Method of payment (12,6,4,1)?"
Formula:

```
Amount of tax      = is
Method of payment  = zw%
Portion per month    ak(m,6) = is/zw%
Amount per year      ak(13,6)= 12*(is/zw%)
```

The formula is the same for insurance

2. Total costs and mileage costs

```
Input amount          =b
Total mileage         =ak(m,5)
Total costs            ak(m,8) = ak(m,8)+b
Costs per mile         ak(m,9) = (ak(m,8)/ak(m,5))*100
```

The same goes for year-total formulas.

In addition we need routines to round the dollar amounts and to convert the values into numbers with two places after the decimal.

The program listing

The lights are dimmed, the curtain is raised: the program ******** AUTO COSTS ********** will now be shown.

Sit down in a quiet place and go through the lines of code one at a time (with a glass of wine at your fireplace, perhaps?). The comments are at the end of the program.

```
1 rem ***************************************
2 rem ***  auto costs  ****  ver. 3.10    ***
3 rem ***  from the    ****  date 5/28/84 ***
4 rem ***  data becker / abacus software  ***
5 rem ***  idea book   ****               ***
6 rem ***************************************
7 rem
8 rem
9 print chr$(147)                : rem clear screen
10 poke 53272,23                 : rem upper/lower case
15 poke 53280,5 : poke 53281,5   : rem crt color=green
20 print chr$(149)               : rem letters = brown
25 dim ak(13,9)                  : rem process data
30 gosub 11000                   : rem input line
35 rem
40 input "is this the first run? ";w$
45 if w$<>"y" and w$<>"Y" and w$<>"n" and w$<>"N"then 30
50 if w$="y" or w$="Y" then 50000
55 rem
60 dim d$(13,9)                  : rem store data
65 u$="@:"                       : rem overwrite file
70 print chr$(147)               : rem clear screen
80 print chr$(149)               : rem color of letters=brown
90 rem
95 rem
100 rem +++++++++++++++ start program +++++++++++++++++
110 rem
120 gosub 30000                  : rem read data
130 gosub 10000                  : rem input/output format
```

```
140 gosub 11000                  : rem set input line
145 rem
150 input "Which month (0=End)  ";m
155 rem
160 if m=0 then 340              : rem store data
170 if m>13 then 140             : rem new input
180 gosub 12000                  : rem name of month
190 gosub 20000                  : rem print month
200 if m=13 then gosub 22000     : rem input year
210 if m=13 then 140             : rem input month
220 gosub 11000                  : rem input line
225 rem
230 input "Which category (0 to 5 / 0=end)   ";r
235 rem
240 if r>5 then 220              : rem new input
250 if r=0 then 140              : rem input for month
260 cs=1026+40*cz(r)             : rem category pointer
270 poke cs+34,188               : rem set pointer
275 rem
280 if r<5 then input "Input amount  = $  ";b
290 if r=5 then input "Input mileage = Mi ";b
295 rem
300 gosub 21000                  : rem compute month total
310 gosub 20000                  : rem output month data
320 poke cs+34,32                : rem clear category pointer
330 goto 220                     : rem input section
340 gosub 31000                  : rem store data
350 gosub 11000                  : rem input line
355 rem
360 input "Exit program";w$
370 if w$<>"y" and w$<>"Y" and w$<>"n" and w$<>"N"then 450
380 if w$="n" or w$="N" then 100
385 rem
```

```
390 end
400 rem ++++++++++++ program end ++++++++++++++++++++++++
500 rem
505 rem
9000 rem ++++++++++++ subroutines ++++++++++++++++++++++
9005 rem
9010 rem
10000 rem ++++++++++++ input/output format +++++++++++++++
10005 rem
10010 print chr$(147);
10020 print "*****************************************";
10030 print "*                                       *";
10040 print "*       Automobile Costs                *";
10050 print "*                                       *";
10060 print "*****************************************";
10070 print
10080 print "Monthly costs for                        ";
10090 print : print
10100 print "(1) Gas              = $  ........      ";:
      cz(1)=9
10110 print "(2) Repair           = $  ........      ";:
      cz(2)=10
10120 print "(3) Care/maintenance = $  ........      ";:
      cz(3)=11
10130 print "(4) Reimbursement    = $  ........      ";:
      cz(4)=12
10140 print "(5) Total mileage    = Mi ........      ";:
      cz(5)=13
10150 print "(6) Tax              = $  ........      ";:
      cz(6)=15
10160 print "(7) Insurance        = $  ........      ";:
      cz(7)=16
10170 print "(8) Total costs      = $  ........      ";:
```

```
      cz(8)=18
10180 print "(9) Mileage costs    = $  ........      ";:
      cz(9)=19
10300 return
10990 rem
10995 rem
11000 rem +++++++++++++ input line +++++++++++++
11005 rem set input line
11010 poke 214,22 : poke 211,0 : sys 58640
11020 return
11040 return
11990 rem
11995 rem
12000 rem +++++++++++++ month names +++++++++++++++++
12005 rem
12010 data "January" , "February", "March"
12020 data "April", "May", "June", "July"
12030 data "August", "September"
12040 data "October","November"
12050 data "December", "The year 84"
12060 rem
12070 for n=1 to 13
12080 :   read n$
12090 :   if n=m then 12120
12100 next n
12120 mo$=n$ : restore
12130 poke 214,6 : poke 211,26 :sys 58640
12140 print "           ";
12150 poke 214,6 : poke 211,26 : sys 58640
12160 print mo$;
12170 return
12990 rem
12995 rem
```

```
20000 rem ++++++++++++ output monthly data ++++++++++
20005 ff$="          "              : rem empty string
20010 for r=1 to 9
20020 :   ak=ak(m,r)
20040 :   if r=5 then dz$=str$(ak) :dz%=len(dz$): goto
          20060
20050 :   gosub 40000
20060 :   poke 214,cz(r) : poke 211,28 :sys 58640
20070 :   print ff$
20080 :   poke 214,cz(r) : poke 211,36-dz% :sys 58640
20100 :   print dz$
20110 next r
20120 return
20990 rem
20995 rem
21000 rem ++++++++++++ compute monthly data ++++++++++++
21005 if r=4 then b=b*(-1)          : rem reimbursement
21010 ak(m,r)=ak(m,r)+b
21015 if r=5 then 21030             : rem miles driven
21020 ak(m,8)=ak(m,8)+b             : rem total costs
21030 gk=ak(m,8) : mi=ak(m,5)       : kk=gk/mi
21040 ak(m,9)=int(kk*100)/10        : rem costs per mile
21050 ak(13,r)=ak(13,r)+b           : rem year total
21055 if r=5 then 21070
21060 ak(13,8)=ak(13,8)+b
21070 gk=ak(13,8) : mi=ak(13,5) :kk=gk/mi
21080 ak(13,9)=int(kk*100)/10
21090 return
21990 rem
21995 rem
22000 rem ++++++++++++ year input ++++++++++++++++++++++++
22005 rem
22010 gosub 11000                   : rem input line
```

```
22020 input "Input tax or insurance ";w$
22025 rem
22030 if  w$<>"y" and w$<>"Y" and w$<>"n" and w$<>"N" then
      22010
22040 if w$="n" or w$="N" then return
22100 rem input tax
22105 print "                                                "
22110 gosub 11000              : rem input line
22120 input "Tax amount =$  "; is
22130 gosub 11000              : rem input line
22140 input "Method of payment (12,6,4,1) = ";zw%
22150 ms=is/zw%                : rem monthly portion
22160 for m=1 to 12
22170 :   ak(m,6)=ms
22175 :   ak(m,8)=ak(m,8)+ms
22180 next m
22190 ak(13,6)=12*ms           : rem yearly tax
22200 rem input insurance
22205 print "                                                "
22210 gosub 11000              : rem input line
22220 input "Insurance amount =$  ";iv
22230 gosub 11000              : rem input line
22240 input "Method of payment (12,6,4,1) = ";zw%
22250 mv=iv/zw%                : rem monthly portion
22260 for m=1 to 12
22270 :   ak(m,7)=mv
22275 :   ak(m,8)=ak(m,8)+mv
22280 next m
22290 ak(13,7)=12*mv           : rem yearly insurance
22300 return
22990 rem
22995 rem
30000 rem ++++++++++++ read data ++++++++++++++++++++++++
```

```
30005 print "One moment please"
30010 print : print "File being loaded"
30015 open 1,8,2, "dat.autocosts,s,r"
30020 m=1
30030 for r=1 to 9
30040 :    input#1,d$(m,r)
30050 next r
30060 if st<>64 then m=m+1 :goto 30030
30070 close 1
30080 for m=1 to 13
30090 :    for r=1 to 9
30100 :        ak(m,r) = val(d$(m,r))
30130 :    next r
30140 next m
30150 print chr$(147);
30160 return
30990 rem
31995 rem
31000 rem ++++++++++ store data ++++++++++++++++++++++++
31005 print chr$(147) : print"One moment please"
31010 print : print "File being stored"
31015 open 1,8,2,u$+"dat.autocosts,s,w"
31020 for m=1 to 13
31030 :    for r=1 to 9
31035 :        pr$=str$(ak(m,r))
31040 :        print#1,pr$
31050 :    next r
31060 next m
31070 close 1
31080 print chr$(147);
31090 return
31990 rem
31995 rem
```

```
40000 rem ++++++++++++ rounding & decimal display +++++++
40005 rem                          : rem example ak=1.2345
40010 d1  = int(ak*1000)           : rem d1=1234
40020 d2  = d1/10                  : rem d2=123.4
40030 d3  = int(d2)                : rem d3=123
40040 d4% = (d2-d3)*10             : rem d4=4
40050 :     if d4%>=5 then d5%=1 : rem round up
40060 :     if d4% <5 then d5%=0 : rem round down
40070 d3 = d3+d5%                  : rem d3=123
40080 ak = d3/100                  : rem ak=1.23
40090 rem decimal format
40100 dz$= str$(ak)  : dz%=len(dz$)
40110 for p=1 to dz%
40120 :   p$=mid$(dz$,p,1)         : rem find decimal point
40130 :   if p$="." then 40160
40140 next p
40150 dz$=dz$+".00"                : rem dz$ contains no point
40160 if p=dz%-2 then 40180        : rem dz$ 2 positions after
      decimal
40170 if p=dz%-1 then dz$=dz$+"0"
40180 dz%=len(dz$)
40190 return
50000 rem ++++++++++++ initialize file ++++++++++++++++++
50005 rem
50010 for m=1 to 13
50020 :   for r=1 to 9
50030 :       ak(m,r)=0
50035 :       ak(m,5)=1
50040 :   next r
50050 next m
50060 gosub 31000
50070 open2,8,15
50080 input#2,ff,fb$,sp,se         : rem read error channel
```

```
50090 close 2
40100 print chr$(147)
50110 print chr$(144);ff;fb$;sp;se
50120 if ff=0 then 40              : rem everything o.k.
50130 if ff<> 63 then end          : rem file exists
50140 print chr$(147);             : rem clear screen
50150 print "A file already exists"
50155 rem
50160 print "with name  AUTOCOSTS   "
50170 print : print "If file should be deleted"
50180 print "press the * key."
50190 get g$ : if g$="" then 50190
50195 rem
50200 if g$= "*" then u$="@:" :  print chr$(147) : gosub
      31000
50210 get g$ : if g$="" then 50210
50220 if u$="@:" then 60
50230 end
```

Comments

Line 00001 to 00095: The program introduces itself and does its housekeeping. In addition it will want to know if this is the first time the program is being run. If yes, any existing files will be overwritten.

Line 00100 to 00390: This is the main section. Everything will be controlled from here.

Line 10000 to 10995: Input format. It is universal enough to be used both for input and output. The variable cz(n) controls the cursor during the input and output.

Line 11000 to 11995: Input line. Cursor goes to line 22 when this subroutine is called up. The required input can then be typed in and control jumps back.

Line 12000 to 12995: The names of the month. This is a small luxury in which the number of the month will be translated to the full name and placed in the correct position of the input/output mask.

Line 20000 to 20995: The data for the month will be taken out of the table, rounded, and reformatted to have two digits after the decimal point. Everything will then be placed in the input/output mask.

Line 21000 to 21995: The calculations. This is where the required calculations are performed.

Line 22000 to 22995: Input of yearly information. The tax and insurance amounts will be requested and prorated over the months.

Line 30000 to 30990: The data is loaded from the diskette. In the second half of the subroutine, the input data will be placed into the working storage area.

Line 31000 to 31990: The storage of the data. This should no longer be unfamiliar to us.

Line 400000 to 40190: Rounding up and rounding down. The rounding routine is a bit long because I tried to make the procedure clear. This would normally be done with a standard rounding formula. In the second part of the

subroutine all amounts which either have one place after the decimal or which have no places after the decimal are reformatted to have 2 positions after the decimal point.

Line 50000 to 50230: Control is passed to this portion of the program only when it is run for the first time. It is here that the file is prepared and it is also here that additional utility programs for formatting, reading the directory, etc. can be built in.

One final word: The program is written in such a way that it can be shortened almost as much as you want. You can of course build in additional subprograms. You will find interesting tools in all of the previously mentioned Abacus Software books.

4.3 How high will the monthly payments be?

```
#########################################################
*            Requirements                               *
#########################################################

#########################################################
*            Hardware                                   *
#########################################################
*                                                       *
*            Commodore 64                               *
*            1541 disk drive                            *
*            Monochrome monitor                         *
*            Dot-matrix printer                         *
*                                                       *
#########################################################

#########################################################
*            Software                                   *
#########################################################
*                                                       *
*            None                                       *
*                                                       *
#########################################################

#########################################################
*            Previous requirements                      *
#########################################################
*                                                       *
*            Good to very good knowledge                *
*            of BASIC                                   *
*                                                       *
#########################################################
```

We are now going to ask a portion of the readers to leave the room. Anyone who just won the lottery, whose rich uncle just died, or who has just received a handsome portfolio of Commodore stock should please leave the room. You are probably never going to build a house.

If you are one of those remaining I will assume that first, you plan to build a house at some time in the future and that, second, you are neither rich nor poor. In other words: You are probably going to have to finance your new house and one question is plaguing you: "Can I really afford to? How high are my monthly payments going to be?"

This calculation is actually not very complicated. You will need a big pad of paper, a pencil and a lot of time. It takes a couple of hours, even with heavy use of a pocket calculator to get the monthly information for, let's say, the next 25 years. Your C64 can do it a lot quicker.

You will now jump ahead 2 or 3 pages and look for the listing. But I am going to have to disappoint you. No listing. I simply thought that in the meantime you are probably far enough along that you can write such a program yourself. Section 4.2 in particular contains a lot of the information that you need.

No fear, however, I am not going to leave you totally alone. We'll proceed in the same way as we did with the program "Auto Costs."

The program idea

The program "construction cost" will calculate the monthly payments which are required to pay off the interest and principle of a house loan. Tax advantages have not been considered. Results can be shown either on the display or printed out and can also be stored on the diskette.

Problem analysis

What should the program be able to do?

1) Collect data
2) Calculate data
3) Output data

The program should proceed in approximately the following manner:

1) Input of the loan amount
2) Input of a down payment, if any
3) Input of interest
4) Input of the beginning date of the payments
5) Input of the ending date of the payments
6) Input of the beginning payment
7) Computation of monthly interest cost
8) Computation of monthly payment
9) Computation of total debt at the beginning of

the month
10) Computation of total debt at the end of the month
11) Computation of payment to principle for the year
12) Computation of total interest payments per year
13) Computation of the annuity (=interest + principle per year)
14) Computation of debt at the beginning of the year
15) Computation of debt at the end of the year
16) Output of the data from 8) and 11) thru 15)
17) Print out of the data from 16)
18) Storage of data

Calculations which are to be saved can be re-run with different parameters, if required. Particularly because of this option, the program should have a menu.

MENU Construction Costs

(1) Data input
(2) Change parameters
(3) Recalculate
(4) Print results
(5) Store results
(6) End the program

Display format

We need several input masks for our program. We will have to develop two for the actual data.

Format of data input

```
*--------------------------------------*
*   Construction Costs Calculations    *
*--------------------------------------*

                Input

Loan                    = $  .........
Down payment            = $  .........
Interest rate           = %      .....
Amortization rate       = %      .....

Loan starts month    ..  year   ....
Loan ends   month    ..  year   ....
```

Format for data output

```
*--------------------------------------*
*   Construction Costs Calculations    *
*--------------------------------------*

          Output of results

*----------------------(1)--------(2)-*

Year            =       ....        ....
```

```
Beginning debt  = .........   .........
Annuity         = .........   .........
Monthly payment =   .......     .......
Interest/year   =  ........    ........
Amortization    =  ........    ........
Remaining debt  = .........   .........

*--------------------------------------*
```

My idea was that the year number would be entered in the output format in the program. We then fill in the rest of the format for this year and for the previous year. The values for the next two years would then be shown after any key had been hit. Unfortunately the 64 can show only 40 character per line. Otherwise you could make columns out of the lines and show the values of all the years underneath each other.
We also require a format for saving and loading:

```
*--------------------------------------*
*   Construction Costs Calculations    *
*--------------------------------------*

            (Saving/loading)

Name of the calculation : [filename]....

        One moment please


[filename]....is being loaded (or saved)
```

List of variable names and file design

Name	Size	Type	Length	Contents
db	$ 0-999999.99	N	max. 9	Amount of loan
gh	smaller than db	N	max. 9	Down payment
zs	% 0.01-19.99	N	max. 5	Interest rate
aj	1900-2099	N	max. 4	Start year
am	1-12	N	max. 2	Start month
ej	1900-2099	N	max. 4	Ending year
em	1-12	N	max. 2	Ending month
mb		N	max. 8	Monthly payment
ms		N	max. 4	Monthly interest
tr		N	max. 4	Beg. amort. rate
x	50	N		Max no. years
jz(x)		N	max. 9	Yearly interest
jt(x)		N	max. 9	Yearly amortization
an(x)		N	max. 9	Annuity
ja(x)		N	max. 9	Beg. debt
je(x)		N	max. 9	End debt
y	12	N		Month
mz(y)		N	max. 8	Monthly interest rate
mt(y)		N	max. 8	Monthly amortization
ma(y)		N	max. 9	Month beg. debt
me(y)		N	max. 9	Month end debt
md$		A	max. 15	Filename
dd$(1)	Read file	A	14	Title of format
dd$(2)	Store file	A	15	Title of format

er	5	N		Input section
ce(5)		N	1	Section line
ar	7	N		Output section
ca(7)		N	1	Section line

In order to avoid "REDIM'D ARRAY ERROR" a file will be dimensioned for 50 data fields. If your bank will make loans of longer than 50 years, you can change the value of x.

Required formulas for calculation

We need quite a few formulas. First the recalculation of the input values. Theoretically it would be nice if we had a constant monthly payment and could choose an input at the beginning for the amortization rate. It is even simpler if only 1 of the 2 values is input and the other one is calculated. We will however keep with the fact that the amortization rate can be chosen and that the monthly payment is variable. And now the formulas:

```
zs = zs/100
tr = mt/100
ms = zs/12
mb =db*(zs+tr)/12
ja(aj) = db-gh
jz(aj) = ja(aj)*zs
jt(aj) = 12*mb-jz(aj)
an(aj) = jz(aj)+jt(aj)
ma(am) = ja(aj)
```

```
mz(am) = ma(am)*ms
mt(am) = mb-mz(am)
me(am) = ma(am)-mt(am)
ma(am+1) = me(am)
mz(am+1) = ma(am+1)*ms
...etc., up to 12
je(aj) = me(12)
jz(aj) = mz(am)+mz(am+1)+...+mz(12)
jt(aj) = mt(am)+mt(am+1)+...+mt(12)
je(ej) = 0
me(em) = 0
```

And that's it. You may have wondered why there are two different calculations for the yearly interest and yearly amortization rate. The reason: The beginning and ending year of the loan have (from a financial standpoint) less than 12 months and must therefore be calculated differently than an entire year. You can of course act as if every year has a different number of months, the number of which must be input for the calculations. You then need a formula which adds the monthly values together.

So now you can turn on your computer and start with the coding. If you should have any problems, grab some of the books that can give you a helping hand (such as the C64 books from Abacus Software).

4.4 First, second, third

```
#########################################################
*           Requirements                                *
#########################################################

#########################################################
*           Hardware                                    *
#########################################################
*                                                       *
*           Commodore 64                                *
*           1541 disk drive                             *
*           Monochrome monitor                          *
*                                                       *
#########################################################

#########################################################
*           Software                                    *
#########################################################
*                                                       *
*           DATAMAT                                     *
*                                                       *
#########################################################

#########################################################
*           Previous requirements                       *
#########################################################
*                                                       *
*           None                                        *
*                                                       *
#########################################################
```

Strange: an unexpectedly high percentage of computer fans have an active sport as a first or even second hobby, even though they are normally considered to be the stay-at-home type. Tennis, bicycle riding, and jogging seem to be the favorites. Sport clubs are always glad to have computer owners as members because of the many areas in which they can be of help: membership files, form letters, and dues are all areas in which they can assist. One of the high points in the life of a hobby programmer might be the annual sport festival, possibly a marathon run. This gives our hobby computerist a chance to show what he can do.

Teenage computer hackers who are members of various clubs have also been able to make a little money on the side by writing programs to take care of some of the administration. Although there is a danger that we might cause this source of income to dry up, we at least want to present an easy problem solution for our sports festival.

The case

The Colorado Sports Club has 300 members and offers training in gymnastics, judo, track and field, as well as soccer and handball. The club has always tried to be a trend setter. Even before jogging got to be an "in sport," there was a long tradition of jogging at the club and this year is the 10th marathon run in which men, women, and children of all ages will be taking part in a 42 kilometer run. In order to make the event more interesting,

the club is going to try to get some prominent marathon runners to participate. The local television station has also agreed to broadcast the event and it is hoped that this will make the name of the club and the city better known.

The statistics

-approximately 1200 runners
-40% of which are women and girls
-5 different age categories

The problem:

Not only because of the television but also in order to make the whole event more professional, the club has promised that all results will be available within one hour after the last marathon runner has crossed the finish line. Every participant is also going to receive a certificate which shows, in addition to the total running time, his overall placement and the placement within the various groups. There are then approximately:

-13 result lists
-and over 1000 certificates

The solution

One of the club members, John Key, is a senior at a local high school and also the owner of a C64. At one of the meetings on the marathon run, he maintained that with his computer the administrative work of the marathon could be easily handled.

"All participants will be stored in a file and the start number which is given at registration will be used as an index field. This start list has the following purposes:

- to allow the participants to check their own data
- to allow the time keepers and judges to collect the various data"

...as John explained this to the other club members, no one understood a word. Since no one wanted to admit he didn't know what John was talking about, they made him the assistant chairman of the organization committee and decided to give him free rein with the data processing. John then began with the preparation. The first thing he did was to design the input format.

This had to contain the following fields:

- starting number
- last name
- first name
- sex
- age group

- running time

There were no other special requirements for the input format, as it was to be used only with the monitor.

```
Sporting Event
-----------------------------------
start number: 00012
name        : Kaminiski
first name  : Judy
m/f         : f
class       : senior
result      : .. h .. min .. sec
```

The format requires 55 characters which means that theoretically we can get 600 records on one diskette. Since we are sure that the thing is really going to go, we set up 1200. John has chosen to call the format "Sporting Event," because he feels he may be able to use it for other events.

The first job of the format is to collect data from the registrants. The Colorado Sports Club has been doing a lot of advertising and as a result, 80% of the participants have registered by mail. This has been a big help in that all the data can be input and checked in peace and quiet. The result field will be filled in with periods. This is only necessary in the first record as it can be easily included in the subsequent records by hitting RETURN.

The first afternoon at the terminal convinces our friend that the input of data is sort of a dumb job and

that some of the other club members should help him. Shortly afterwards, a couple of friendly helpers are found and after 3 sittings of about 1 and a half hours each, all 1000 participants have been entered.

The pointer file

How are we going to set up the 13 results lists? Quite simply: Through use of menu selection file sorting we will filter the required list, in which only registrants who fulfill particular criteria will be selected. We will have pointer files for:

1) all participants (the total list)
2) all female participants
3) all male participants
4) youth group B, male
5) youth group B, female
6) group A children, male
7) group A children, female
8) age group 20 to 30, male
9) age group 20 to 30, female
10) age group 30 to 40, male
11) age group 30 to 40, female
12) seniors, male
13) seniors, female

These pointer files should have names with which they can be identified, i.e. "startmale30" or "startfemalesen ." These somewhat strange looking names are caused by the fact that names can have a maximum of 50 characters. In addition, an alphabetically sorted total list should be printed out. John now needs to invest about 3 hours in the

creation of the pointer files. This investment in time will ,however, give him the capability of printing out the evaluation of the marathon within several minutes after its completion.

The print-out

This must also be prepared before the beginning of the race. John gives some thought as to how the result list for the starters should look in the menu selection (file evaluation). He comes to the conclusion that an evaluation is enough. The information about the individual participants can then be printed on a single line. This looks as follows:

```
-------------------------------------------------------------
Marathon Run 84                          Colorado Sports Club
-------------------------------------------------------------
Number   First name   Last name      Class      Results
-------------------------------------------------------------
00001    Christa      Vahlensiek     F 30-40    ........
00002    Heliodoro    Salazar        M 30-40    ........
00003    Adi          Pumann         M 20-30    ........
00004    Erika        Pumann         F 20-30    ........
00005    Sabine       Pumann         F B-Youth  ........
00006    Karl         Norpoth        M Senior   ........
00007    Norbert      Stampfer       M A-Youth  ........
00008    Corinna      Pietsch        F B-Youth  ........
00009    Rainer       Bartel         M 30-40    ........
00010    Patrick      Schiefelbein   M B-Youth  ........
```

00011	Paul	Schiefelbein	M Senior	.....
00012	Edeltraud	Kaminski	F Senior	.....
00013	Petra	Petel	F 20-30	.....

As you can see, the starting list is sorted by starting numbers and in the results column, we only have a place holder at the present time. This will be replaced when the time keepers fill in the times.

The printer is now going to get into the action. With help of the evaluation, multiple copies of the total list and the different age group listings will be printed. You can of course print as many as you wish for posting or for distribution to the press and television or for the list of events, etc. For the actual marathon, only the total lists are relevant.

Time keeping

We now come to the problem of big numbers. We have to realize that the system would break down, with or without a computer, if the results of 1200 participants had to be sequentially (one after the other) entered in the system. This is why whole companies of time keepers are normally used to control a particular segment of a run.

In the case of marathon runs which are evaluated without a computer, it can easily happen that the time keepers cannot keep up with the work because too many participants stumble over the finish line at the same time. This is why in the case of standard events of this type, re-

serve time keepers are available who can be called upon in an emergency.

John has thought up something special to get around this problem. His time keepers all have (loaned) printing pocket calculators. Nothing is going to be computed on them, but the starting number and the total time can be printed out. The strips of paper will then be brought by courier to the computer and further processed.

After all the runners who have not given up finally cross the finish line and their times have been written down, we go to the:

Results lists

In order to save the index, the DATAMAT menu selection "exit program" will now be chosen. We then proceed to the sorting of files. Sort criteria number 1 will be total elapsed time and a list will be created, for the moment only in the computer, which is sorted from the fastest runner at the beginning to the slowest at the end.

The results list can now be created using the old (hopefully, stored) evaluation files, exactly as we created the starting list at the beginning. This takes a little time but is considerably faster than any method which does not use a computer.

And now we have things under control. The lists are more or less finished and only need to be printed out. The

total time between arrival of the last runner and the handing out of the first results: around 45 minutes.

The certificates

Now comes the part which requires the most manual labor. This comes from the fact that DATAMAT cannot automatically compute the placings of the different runners. But when such good looking forms are available (see the next page) that's not really a problem. A certificate can be written with the word processing system TEXTOMAT, which has an interface to DATAMAT. TEXTOMAT reads the DATAMAT files and puts the data in the correct positions in the text. These certificates should of course be printed on good paper with a club emblem and all that goes with it.

P.S.: The 10th marathon run was a total success. The weather was good and an enormous crowd fenced the road. The television station broadcasted a 5 minute report in the sport show.

Everything went to everybody's satisfaction. The only breakdown in the computer sector was when John forgot to run the subprogram "END" after the last participants data had been entered. The index file was destroyed and had to be created again the next day, but aside from that, everything went well. The list of results was available within after 1 hour after the end of the marathon and the approximately 1050 runners had reached the finish line. Everyone received their personal certificate 3 days later in the mail. Next time John is going to have 2 computers,

4 floppies and 2 printers. One will be a daisy wheel printer for the certificates. In this way the printing of the certificates can be done at the same time as the list are being printed.

```
**********************************************************
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
*+******************************************************+*
*+*    10th Colorado Sports Club Marathon Run          *+*
*+*    8888888888888888888888888888888888888888        *+*
*+*                                                    *+*
*+*    ................  ................              *+*
*+*    Successfully participated in the Marathon       *+*
*+*    with an elapsed time of..h.../min.../sec        *+*
*+*    and placed..............                        *+*
*+*                  ////////////////                  *+*
*+*                  ////////////////                  *+*
*+*                  ////////////////                  *+*
*+*                  ////////////////                  *+*
*+*                  ////////////////                  *+*
*+*                                                    *+*
*+*   in class ..............   ............           *+*
*+*                                                    *+*
*+*                                                    *+*
*+*                  ////////////////                  *+*
*+*                  //////////////// place           *+*
*+*                  ////////////////                  *+*
*+*                       overall                      *+*
*+*                                                    *+*
*+*         Aspen, Colorado 14 May, 1984               *+*
*+*                                                    *+*
*+*  ---------------          --------------           *+*
*+*  Club President           Secretary                *+*
*+*                                                    *+*
*+******************************************************+*
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
**********************************************************
```

4.5 A calculator, too

```
############################################################
*            Requirements                                  *
############################################################

############################################################
*            Hardware                                      *
############################################################
*                                                          *
*            Commodore 64                                  *
*            Storage medium                                *
*            Monitor                                       *
*                                                          *
############################################################

############################################################
*            Software                                      *
############################################################
*                                                          *
*            None                                          *
*                                                          *
############################################################

############################################################
*            Previous requirements                         *
############################################################
*                                                          *
*            Knowledge of Commodore BASIC                  *
*                                                          *
############################################################
```

Do you have a pad of paper or even a pocket calculator sitting next to your C64? Rather aggravating now that you have this expensive computer with practically unlimited calculating power in front of you. But if you want to compute "cos (90-(36*1024)/49)" you have to do it by hand. Frustration can be a great incentive and in this case we have been very creative.

Many roads lead to Rome. And there are a lot of roads which lead us to a software calculator on the 64. The methods are different but the problem is always the same: The input/output and calculation of arithmetic calculation with their required values should be performed in a simple and user-friendly manner.

The different solutions

1. The keyboard simulates a pocket calculator.

2. The keyboard is displayed on the monitor and the input is accomplished through the movement of the cursor.

3. Subprograms in machine language can be written and called up while the BASIC program is running

1. Keyboard - Hand calculator

This calculator is capable of doing the basic arithmetic calculations. It can round the results and it can store them as constants. Individual inputs into the storage area can be deleted.

```
10000 rem ***c64 as a calculator
10010 poke  53281,0  : poke 53280,0
10020 poke  53272,23 : print chr$(150)
10030 u$=" -+-+-+-+-+- Calculator -+-+-+-+-+-"
10040 print chr$(147);
10050 print u$ : print
10060 print "To which position after the decimal
      should";
10070 print "rounding take place"; (1-8 ; 9= no
      limit)";
10080 input in$ : in=val(in$) : if in>0 and in<10
      then 10100
10090 goto 10040
10100 print chr$(147); : print u$ : print
10110 sm$=""
10120 get t$ . if t$="" then 10120
10130 t=asc(t$)
10135 if t=43 or t=45 or t=42 or t=47 then 10310
10140 if t$=chr$(136) and sm$="" then sm=0 : k=0 :
      goto 10100
10150 if t$=chr$(133) then k=sm : gosub 10500
10155 if t$=chr$(134) then t$=str$(k) : goto 10300
10600 if t$=chr$(135) then gosub 10500
10170 if t=46 then 10300
```

```
10180 if t<48 or t>57 then 10110
10300 print t$; : sm$=sm$+t$ : goto 10120
10310 op=val(sm$) : print chr$(t); : print tab(10)"=";
10320 if t=43 then sm=sm+op
10330 if t=45 then sm=sm-op
10340 if t=42 then sm=sm*op
10350 if t=47 then sm=sm/op
10360 if in<>9 then gosub 10400
10370 print sm
10380 got 10110
10400 sm=(int(sm*10^in+0.5))/10^in
10410 return
10500 print tab (20)"Constant=";k
10510 return
```

Proceed as follows: Input the first operand and hit the "plus" key, input the second operand and the operation (+,-,* or /) and you will see the results on the screen. The results will be placed in the constant storage if you push the F1 key. The F3 key is used to read and the F7 key to delete the contents of the constant storage area. Incorrect results can also be set to 0 with the function key 7.

2. Monitor-cursor-calculator

A couple of details just to whet your appetite: A graphic representation of a pocket calculator will be displayed on the monitor and the different fields can be chosen using the cursor keys in much the same way as a

"mouse" is used. Results are then presented in a simulated display. The whole thing looks very professional. It is, however, considerably too complicated to explain here or to give a program listing.

3. Machine language routines-calculator

Surprise! The 64 already has a built-in pocket calculator. "But where?" you might ask. The answer: The BASIC interpreter has a routine called "FRMEVL," which exactly fulfills the requirements of our pocket calculator. You can't, of course, simply type in GOTO $AD9E. You have to include the operating system routine in a machine language program. In addition you need the addresses of the different routines for the arithmetic calculations. How this can be accomplished can be read in two Abacus Software books, The Anatomy of the Commodore 64, which requires a good knowledge of the computer up to and including the operating system, and The Machine Language Book of the Commodore 64, which provides information on programming in assembly language.

5. Data of all types

You can fill in all kinds of forms, full of data of all types. You can then number the forms and fill shelves full of files with them. You can of course also keep track of your data with your Commodore 64.

From chapter 2.3 you know all the things you can do with data management software. Actual examples will increase our knowledge in this area. Just so that you're prepared in advance, the following program ideas are based entirely on the use of the software package DATAMAT from Abacus Software. I have simply found no comparable product for a similar price. You can of course purchase other data management programs, but why buy a Rolls Royce when a VW bus has more room for luggage?

All the applications discussed are concerned with amounts of data which meet the requirements of the average household. If your requirements are greater, you will sooner or later have to move up to other hardware and as a result, more powerful software. Parallel to this of course is the increase in costs.

DATAMAT is relatively inexpensive and requires only a 1541 disk drive and a printer. Since these prerequisites are applicable to most program ideas, we will be able to do without a block title "Requirements."

There are probably a couple of ideas here for you that you have been subconsciously waiting for. Browse through

this chapter and see what you find of interest.

5.1 Recipe file for hobby cooks

I don't know whether it's good or bad, but one of the first ideas that I had for an application of DATAMAT was to make a personal cook book file. You should know that my culinary art is well known and that my homemade noodles are really sensational. So I got a fresh start and for my first step designed an input format. Much to my dismay however the program answered with "Record too long." Okay, a couple of lines shorter won't be a big problem.

```
----------------------------------------
*              RECIPE FILE             *
----------------------------------------
----------------------------------------

(01)  RECIPE : ¦Spaghetti w/ meatsauce ¦

      INGREDIENTS

(02)  ¦Hamburger.vegetables           ¦
(03)  ¦broth.red wine.tomatoes        ¦
(04)  ¦cream cheese.onions            ¦
(05)  ¦garlic.spaghetti.oil           ¦
(06)  ¦salt.pepper.basil              ¦

(07)  ¦Preparation : fry meat.        ¦

(08)  ¦Pour liquid on                 ¦
(09)  ¦onions, garlic and             ¦
(10)  ¦all vegetables and tomatoes    ¦
```

(11) For ¦04¦ persons

(250) Characters/record: 150 records;
index: field 1)

As you can see, this is no way to write a recipe. A limit of 253 characters per record is simply too small. What are we are going to do now? We simply have to find another starting point which we will now explain in steps.

A recipe normally contains two parts: A list of the ingredients and a description of how to put them together. How are we going to divide these two parts? An example:

Your wife (husband) brought a couple of colleagues home after the office party. They are all hungry and know that there is always something good to eat at your place. Your honor as a chef is on the line. In the refrigerator you're able to find two lonely eggs, a little butter and a piece of cheese. In the pantry is a can of asparagus and one of pineapple. There is also bread.

You go quickly to your C64, load DATAMAT and the file "ingredients." A selected list of all the recipes that you can make with what you have on hand is displayed. "But I can't go to the computer every time a guest comes," you might now be saying. You don't need to. The work described above can of course be done beforehand. We will come back to this point again.

You pick a recipe out of the list and write down the

name. Now you load the file "preparation" and print out the required information.

It is of course nonsense in such a situation to have to go to the computer. Let's now however take a look at how we can set up a recipe file.

File "ingredients"

The ingredients can be elected quite easily using the following input format:

```
-------------------------------------------
*         Recipe (1) Ingredients          *
-------------------------------------------

(01) NAME :Spaghetti with meatsauce       :
(02) FOR  :04: PERSONS                    :

     INGREDIENTS:        AMOUNTS

(03) Spaghetti/oil       :2.2 lbs         :
(05) Hamburger Steak     :.44 lbs         :
(07) onions/garlic       :approx. 2/1     :
(09) tomatoes            :1 can           :
(11) carrots/leeks/celery/ :              :
(13) broth/red wine        : 3/4 l.       :

     CONDIMENTS
```

```
(15) SALT y PEPPER y MUSTARD n
(18) PAPRIKA n NUTMEG y
(20) OTHER                  :                  :
(21) HERBS basil/parsley    :                  :
(22) and oregano            :                  :

(251 Characters/Records; 100 Records; Index: Field 1)
```

The input fields for the condiments require little room. If they had to be fully written out, you would need approximately 20 more characters per record and we would have again reached our limit. Change the input format to meet your taste (in both meanings of the word).

File "preparation"

We are going to store the actual recipes in this file. The input format:

```
-----------------------------------------------
*          Recipe    (2)  PREPARATION         *
-----------------------------------------------

(01) NAME :Spaghetti                          :
(02) FOR :04:  PERSONS

          PREPARATION

(03) :bring water to boil                     :
(04) : add salt                               :
```

```
(05) ¦and a little oil                    ¦
(06) ¦cook spaghetti                      ¦
(07) ¦and then rinse off                  ¦
(08) ¦toss spaghetti in pan               ¦
(09) ¦with a little melted butter         ¦

          TIMES

(10)  ¦    ¦ FOR PORTION ¦   ¦
(11)  ¦    ¦             ¦   ¦
(12)  ¦    ¦             ¦   ¦
(13)  ¦ 8-9¦ TOTAL
```

(231 Characters per record; 100 records; index: field 1)

And here again we have a room problem. You can figure out yourself how to get around it.

Selection and printing

You have hopefully already entered your recipes because--as you can probably imagine--we are now going to get some use from then. There is no need to sort anything in advance as we don't have any fields the sorting of which would give us any benefit. The selection however may be worthwhile. You can for example set up an "egg file." This should contain all meals in which eggs play the main role. You can also select by portions of the name, for example basic recipes or salad, soups etc.

All these pointer files (if you don't know anymore what a pointer file is take a look at section 2.3) can be printed now, in any order with other selection criteria. (This is also a term which was used in section 2.3). Another pointer file can be used to create your table of contents.

As in all other data processing problems, it is better to try it out than to just read about it.

5.2 Getting organized in the deep freezer

A little creative chaos never hurt anyone, say many people. And I must say that they are right, in my opinion. In our day-to-day activities, however, things are a little bit different.

Let's take for example our deep freezer. "John, where did you put the pork chops in the deep freezer?" "At the bottom on the right-hand side, Mary." And Mary digs through the cold packages, gets her fingers frozen, and loses her appetite for pork chops. It would of course be good if we had a little bit of organization, at least in our freezer. But you know how it is with good intentions....

This type of search is not particularly pleasant. The following DATAMAT example can give us a some help, however.

Let's imagine the average deep freezer owner. Let's say we have a household with 5 persons, located in the countryside on a small farm. The head of the household is middle-aged and isn't afraid to butcher a pig himself once in a while. Since we also have a good-sized vegetable garden, the deep freezer is usually full.

Two situations are typical: the butchering of the pig and the month of May (with strawberries and asparagus). We then have a good deal to put in the freezer. Up till now

this was done as follows: Everything was placed in the appropriate plastic bags. A label was written and put on and then everything went into the freezer. All information on this label is also in the input format which we are now going to present.

```
=========================================
=              DEEP FREEZER             =
=========================================

= (1) Number    : ¦0030¦                 =
= (2) Contents  : ¦asparagus          ¦ =
= (3)             ¦in bunches         ¦ =
= (4) Amount    : ¦010¦  Weight: ¦   ¦   =
= (5) Frozen on  : ¦05¦¦01¦¦84¦          =
= (6) Good until : ¦11¦¦01¦¦84¦          =
=     Location in Freezer: ¦ 5A ¦        =
=========================================
```

Now we have half a pig, 20 pounds of strawberries, and a lot of asparagus well preserved. Four months later we have the discussion between John and Mary described above.

Let's say that John has a '64. Then everything is a little different. First of all, the labels can be written by the computer and secondly, John can do a lot of other things with his "freezer file."

Let's say John sets up a newly sorted file of the freezer contents which is sorted by date of expiration. This could be printed in a monthly list and hung up in the

kitchen. Mary will always know what has to be eaten next.

And how does it work? As follows:

Prerequisites

- File "freezer contents"
- Stored index (see DATAMAT USER MANUAL!!)

Procedure:

- Load DATAMAT
- Select "Sort file"
- Enter name of file
- Enter name of format
- Select "Sort" from the command line
- Enter "01" in the field expiration date/year
- Enter "02" in the field expiration date/month
- Enter "03" in the field expiration date/day
- Press f1
- Select "run" from the command line
- Select name for sorted file
- Save new file
- Select "File selection"
- Enter file name
- Enter mask name
- Answer the question "Should an already existing..." with "n"
- Enter line length 80
- Delete and move fields as required
- Enter title

- Press f1 when done
- Answer the question "Should an already existing pointer file..." with "y"
- Enter name of pointer file
- Print as required

You now have the list of the contents of your freezer sorted by the expiration date.

```
**********************************
          Freezer contents
**********************************

11 15 84   002 lbs  Asparagus     03a
09 21 84   001 lbs  Strawberries  03b
09 28 84   008 pcs  Pork chops    01a
06 01 84   004 prtn Beef stew     01a
```

Oh yes, in my input format I still have the field "location." I probably should explain what I mean by that. It was my idea that the freezer could be divided into sections according to its coordinates and the packages, cans, and bags placed accordingly. As my freezer only consists of the top part of my refrigerator I haven't been able to try this out. Perhaps you can test the principle for me and let me know your experience.

5.3 Vegetable garden calendar

Garden and computer--they really don't go together too well, do they. There are, however, cases in which these two items can go hand in hand. In view of the fact that these cases occur rather seldom I'll keep this chapter quite short.

Anyone with a garden usually has a feeling for when and where he should plant his vegetables and what care they require. Should any of our readers have a garden which they love but still don't know exactly how to go about taking care of, DATAMAT can give you some help.

Your file could be called for example "garden." The record contains the names of the different plants as well as the information on their care which can be taken from any good garden book. Sort your file by date and you will have a calendar for the care and maintenance of your garden. It is as simple as that.

For the owners of large gardens, files which contain the information of which bed is to get fertilizer or to be otherwise worked on help maintain an overview of the required work. By the way: Greenhouses have used computers for this purpose for a long time; anything that is good enough for them should be good enough for us.

Farming is one area in which little use of computers has been made to date. This is something which will change. So-called part-time farmers, for example, can already

make good use of a home computer: data on the amount of harvest, prices, costs, and so on, can easily be collected and processed with the appropriate software.

One idea which partially belongs in chapter 6 but which I would like to pass along now is this: If you know what crop rotation means then you will also know that if this method is to be used successfully, it is necessary that data be collected over years (if not decades) as to what crop was planted in what bed. Make a drawing of your garden showing the crops planted. Computer graphics can also be of use if you are planting a new garden.

I will now leave you with these tips and I'm sure you'll come up with some of your own ideas from what you have read.

5.4 An inventory for a small company

My brother-in-law is a self-employed painter and whenever I have the time, I give him a hand. One day we were just getting ready to take some material out of his stock room when he complained that he never exactly knew what he had on hand. "Don't you keep an inventory list?" I asked him. "Of course, but how am I supposed to keep track of left-over paint or half a role of wall paper? I wouldn't be doing anything but writing lists."

And that's how I got the idea how to misuse DATAMAT for a very simple inventory control system. If my brother-in-law should one day increase the size of his company and hire a couple of employees, he'd have to purchase professional software anyhow. At least he will have already had his initial "EDP shock."

The data for the inventory would be entered and updated weekly. The inventory list which results is not only the documentation for the material on hand, it is also used as a formula for input. This cycle agrees with that which was done without a computer as the stock room is cleaned up every Saturday.

One remark: The file only includes that material which was not completely used and cannot be totally invoiced to the customer. I'll show you a portion of the list:

```
****************************************
       Material list - Leftovers
****************************************

Wallpaper type 6A
3 1/2 rolls 1 rolls        Leftover from
02/21/84    07/01/84       Ewing Oil
     21.00         12.00   const. site
```

Something else for workers who forget things:
Let us assume that you are a plumber and are going to lay a water pipe into a customer's garden. Do you know off the top of your head which materials and which tools (except for the normal ones) that you have to take with you? No? You simply write yourself some checklist which covers every possible case that might come up before you go to a customer and then you can check these items off point by point.

For example:

```
+++++++++++++++++++++++++++++++++++++++++
+ Order : Painting and wallpapering      +
+                                        +
+ Material: Paint                  ()    +
+           Tarps                  ()    +
+           Masking paper          ()    +
+           Glue                   ()    +
+                                        +
+ Tools:    Large roller           ()    +
+           Small roller           ()    +
```

```
+           Corner brush          ()   +
+           Sponge                ()   +
+           Paint pail (2)        ()   +
+           Water pail            ()   +
+           Small pail            ()   +
+           (for glue)                 +
++++++++++++++++++++++++++++++++++++++++
```

A simple example, of course, but my knowledge in this area is not adequate for more complicated examples.

Conclusion: With the Commodore 64 and appropriate software, any self-employed worker can get used to working with a computer. When things get a little bit bigger so that a personal computer is required, it can take care of bookkeeping, personal administration, and the invoicing and dunning letters. Bids and offers can also be made, including the required calculations, charts, and plans.

5.5 Your personal health records

Health - that's a serious topic and I'll try to keep my jokes to a minimum. Everyone is going to be sick some time or other. In most cases it is only a cold, but other times possibly something more serious. Hopefully it will never be anything incurable. Let us take the example of someone with diabetes. This sickness is, with the methods we have today, not curable. At best, it can be a condition of relative comfort. Someone with diabetes always lives with a continuous control of the body functions. It is at least possible in the meantime for them to take care of a good deal of the therapy themselves.

Did you know that a home computer could represent an enormous instrument for self help? It was not without reason that a 17-year-old student won a special prize for a diabetes software package in Germany.

With this program it is possible to measure blood sugar values, circulation, and pulse, and not only to measure them but also compute and evaluate them. Through the input of information as to how the prescribed behavior was followed, the software package can evaluate the behavior for that particular day. A person with diabetes knows for example if he receives a "0" that he has kept well within the prescribed rules set forth by his doctor. If on the other hand he receives a "12," he knows that he must change his behavior. The program also has an information file which contains general information on the sickness and some partial information on his own particular case.

This diabetes software will be available in a short time for most popular home computers, such as the C 64. If you fall within the group of potential users, you should consult your doctor or local hospital for more information.

More well known computer applications in the health sector are programs for risk calculation and life expectation. If you should find such a program listing somewhere, use it with care. Given the computing and storage capacity of home computers there is little chance that you can get really usable forecasts based on your eating, drinking, and smoking habits. The only useful effect of such programs is that if your life expectancy is deemed low enough by the program, you may quit smoking.

Many people have mixed feeling about medicine in general because of the many scandals associated with it. Some remember natural healing methods and their grandmother's homemade medicines. I find this to be a positive development. The inhalation of steam from cooked camomile is much less risky than swallowing pills. One drawback of home medicine is that its use is based primarily on experience and its effect from person to person can be quite different.

For people who really wish to use these household remedies it can be particularly important to carefully register the effects of cold and warm compresses and the different teas and herbs on the various illnesses. DATAMAT can be of great help here.

You will certainly be able to recognize a DATAMAT input

format when you see one. This one is so simple that no explanation is really required. Health and sickness records are really no different from our recipes. It is of course nonsense to turn on the '64 when you feel you're coming down with the flu. You need to do a little preparation. Enter all the symptoms and the possible therapies and out of these print a list for yourself. This will be of more use.

Tip: Sports and good eating habits are a large part of the prevention of sickness. In chapter 5.8 and 5.9 you will see how your '64 can help you in these areas.

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$          Health Record                       $
$                                              $
$----------------------------------------------$
$ - 1 - Sickness        ¦Blood effusion     ¦  $
$                                              $
$       Symptoms                               $
$ - 2 - Fever           ¦no ¦  Pulse ¦===¦     $
$ - 4 - Blood pressure  ¦         ¦            $
$ - 6 - Pain            ¦                   ¦  $
$ - 7 - Nausea          ¦n  ¦ yes/no           $
$ - 7 - Diarrhea        ¦n  ¦ yes/no           $
$ - 8 - Vomiting        ¦n  ¦ yes/no           $
$ -10 - Loss of Consciousness ¦n  ¦ yes/no     $
$ -11 - Other                 ¦very painful ¦  $
$ -12 -                       ¦in places    ¦  $
$                                              $
$       Therapy                                $
$ -13-  +Compress with vinegar              +  $
$ -14-  +Cool. Rest                         +  $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

5.6 It doesn't make any difference what you collect...

I hope we are in agreement when I say that collecting, independent of what is collected, does not allow itself to be explained rationally. It is simply one of the basic human drives: Finding new objects, bringing them together, ordering and cataloging them. If you are not a hunter, at least you can be a collector. What you collect is not important and the work that goes with it is independent of the collecting itself.

Many home computer owners came to their computers by way of their collecting hobbies. There is then no reason to be surprised that the computer magazines often contain listings for applications that have to do with collections of various kinds.

At the very basic level there are two possible ways of collecting:

- with the goal of having a complete collection of something
- with the goal of having the most beautiful or most rare pieces in a particular area.

Stamp and coin collections can, for example, be placed in the first category, books and records in the second. The problems of cataloging and management are therefore somewhat different. I will show two examples of how you can use DATAMAT to work on your collection.

Stamp collecting

In order to keep it simple we will choose a fairly narrow range: How would it be if we chose sports for our theme? Are you in agreement? Then take your stamp catalog and underline all of the series which could be of interest. These data will then form the basis for our file. For the input format I would then recommend the following:

```
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
[]                  Stamp Collection                    []
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][]

  Series:      >1960 Olympics          <
  Year  :      >1959<
  Post  :      >republik brumelia      <
  complete : >no< canceled : >no<

No.  value     condition  qty      price

1)  > 0.50 <     >02<      >02<  >   7.80<
2)  > 1.00 <     >02<      >02<  >   7.80<
3)  > 1.50 <     >00<      >00<  >       <
4)  > 2.00 <     >00<      >00<  >       <
5)  >10.00 <     >02<      >04<  >   7.80<
6)  >       <    >  <      >  <  >       <
7)  >       <    >  <      >  <  >       <
8)  >       <    >  <      >  <  >       <
9)  >       <    >  <      >  <  >       <
10) >       <    >  <      >  <  >       <
```

```
[][][][][][][][][][][][][][][][][][][][][][][][][][][]
[]          Stamp Collection          []
[][][][][][][][][][][][][][][][][][][][][][][][][][][]

  Series:      >White-water rafting   <
  Year  :      >1951<
  Post  :      >Canada                <
  complete : >y < canceled : >  <

No.  value     condition  qty      price

1)  >$ 0.50<     >+9<      >03<  >   ?   <
2)  >$ 0.75<     >+9<      >03<  >   ?   <
3)  >$ 1.00<     >+9<      >03<  >   ?   <
4)  >$ 1.50<     >+9<      >03<  >   ?   <
5)  >       <    >  <      >  <  >       <
6)  >       <    >  <      >  <  >       <
7)  >       <    >  <      >  <  >       <
8)  >       <    >  <      >  <  >       <
9)  >       <    >  <      >  <  >       <
10) >       <    >  <      >  <  >       <

[][][][][][][][][][][][][][][][][][][][][][][][][][][]
```

The quantity field is useful for printing out lists of duplicates for trading. All values on hand twice or more can be selected and printed out with the corresponding information.

In the same way you can create a wish list that contains all of the stamps that you still require. You can

make a good impression in stamp collecting circles with such a well sorted-list. Here are two examples:

```
Decathlon  olympics   1960 no yes

Republik brumelia

0.10  +9 02   8.50
0.50  +9 02   8.50
1.00  +9 02  11.00
2.00     00
```

```
Decathlon olympics   1960 no no

republik brumelia

0.10  +7 01  1.30
0.50  +7 01  1.00
1.00     00
2.00     00
```

And something else. Many philatelists see their collections not only as a hobby but also as a real investment. If you keep the "price" column in your file up to date (by use of a catalog or through offers and purchase requests) you will be able to sum the list and to see what the total value of your stamps is.

Phonograph record collections

If the number of records that you have is getting close to the 500 mark, you're going to have to make an important decision: "Should I simply continue to buy and not worry about cataloging anything, or should I begin to get systematic about the thing?" If you have decided that you will use the first alternative you can now leave the chapter. If not, I would like to give you a couple of tips.

Why the number 500? Simple: From this point on it's too late to bring some type of system into your collection. The work of input would be too much and too boring. In addition to this, 500 records allow themselves to be placed in more or less reasonable categories. This separation into categories is something you should do in any event. An example: You have 100 records in a certain category, each of which has approximately 12 songs, giving us 100 data records with 12 fields each. Each field contains approximately 15 characters. The input format then contains (12X15=) 180 characters + 20 characters from the title of the record album and 20 characters for the name of the singer or group. Another 4 characters are required for the inventory number and 20 more for additional information on the year of publication and name of the record company, etc. Total: 244 characters.

A file which is created with this input format can, given that the diskette does not contain other files, contain a maximum of 700 data records. It is then useful

to distribute the complete collection over several files, possibly in the following manner:

- pop music
- jazz
- classics

or possibly more detailed

- hard rock
- reggae
- blues
- folk
- swing
- modern jazz
- baroque music
- opera
- musicals
- film music
- western

This process has one big disadvantage. Titles that are contained in different files cannot be related to one another. If you want to know all the different versions of "Slow Boat to China" the only thing that you can do is to sequentially go through all the files that could contain the song. This is not really a good solution. The only way to get around this problem is to allow multiple entries. That is, if you place Bob Dylan records not only in the folk music group but also in the pop music. You are limited here only by your imagination.

The input format as described above:

```
########################################
#            Disco-List Pop            #
########################################
#  No.        : .001.                  #
#  Album      : .Rubber Soul        .  #
#  Singer(s)  : .Beatles            .  #
#  Label      : .EMI      . No.:.12/789. #
#                                      #
#  Side 1             Side 2           #
#                                      #
# .Michelle         . .Norwegian Wood . #
# .Run for your lif. .Drive my car    . #
# .Nowhere man      . .Girl           . #
# .In my life       . .               . #
# .                 . .               . #
# .                 . .               . #
########################################
```

As always, changes on your part are not only possible but recommended.

There are record collectors who use their hobby to make a little money on the side in that they act as mobile disc jockeys for parties and other such activities. You have to consider in such cases that there is a little more to playing records than just having a steady hand. The atmosphere of the party and the wishes of the guests need to considered. There also shouldn't be any pauses in the music.

I'll tell you a little secret: Professionals have standard lists which contain the names of the different titles and the order in which they are to be played. These lists have either proven themselves in practical application or have been tried out in the home studio. Using DATAMAT it is no trouble for the hobby DJ to make up such a tool and I am sure that it is not necessary to go into detail here on how such lists are be put together.

The professionals described above also have other lists. These are used particularly by mobile discos. These are selections out of the total repertoire which are required if you don't always want to carry the whole 10,000 records along with you.

There are many possible applications for computers in discos. Everything from computer-controlled light organs to electronically calculated and designed homemade speakers, From the 64 as a control center to a synthesizer. Anyone who wants to become a professional DJ will be able to choose what he needs.

5.7 Hobby photography

Taking pictures is the most fun if you do everything yourself. It's very difficult to explain to a vacation pocket-camera photographer what a great feeling it is when you're standing in the dark room watching your picture appear in the developer or to explain how much creativity is required to make an excellent photo out of an average snap shot. (From this point on it's just something for insiders).

Have you ever asked yourself the question what is the correct way to develop ASA 400 Kodachrome film? Aha, you've already tried. How many different variations have you been able to find? And did you write everything down in the form of notes?

In my case I saved all this information on a diskette. A DATAMAT file with the name "NEGADATA" contains information on chemicals, temperature, and developing times for films which are not to be processed in a standard manner. In the file "POSIDATA" I have saved the information on the positive process. Both files may be contained on one diskette and can be updated at any time.

```
----------------------------------------
:        === Negative Data ===         :
----------------------------------------
: (1) Film type   : Kodachrome         :
: (2) DIN/ASA     : 400 ASA            :
: (3) Contrast    : HI (HI/LO)         :
: (4) Developer   : Perceptol          :
: (5) Time        : 390 sec            :
: (6) Temperature : 84 degrees         :
: (7) Notes       : Turn often         :
:                   Control temp.      :
:                   exactly            :
----------------------------------------
```

The 64 can also be used for control of the film library, one film equals one file, one picture equals one record:

```
========================================
=            Photo Archives            =
========================================
= Number   -00129-                     =
= Date  (1) -08/17-  Date (2) -08/20-  =
= Filmtype - Tri X     - -400 ASA-     =
= Pictures:                            =
= (01)-(10)  - Screen photos for    -  =
=            - Data Becker          -  =
= (11)-(20)  - Product photos of       =
=            - computer equipment      =
= (21)--38-  - Same as 11-20           =
=            -                         =
= Notes:     - Customer Order          =
```

= - =

==========

Any picture can be reconstructed at any time. This is also a type of experience list. The information that is saved by old-time hobbyist in thick notebooks and organizers can be contained on 3 or 4 handy diskettes. That's real progress.

Further information: The possibility of using a C64 for direct control of the enlarging and development process is, as far as I have been able to determine from research, not very far developed. You should pursue specialized publications for computers and photography if you want further information.

5.8 Jogging

All those nights at the computer are not good for your health. The backbone is bent and your eyes become red. Your circulation either gets weak or begins to go into orbit. Man is in reality a hunter. The body needs activity and exercise, particularly after it spends hours in a sitting position with little or no movement.

The magic word is jogging. It is good that this form of exercise has lost its reputation as being nothing more than fad. Many other useful activities have been killed by their media image.

So let's get into our sweatsuits, put on our running shoes and get out in the park. The best times are either in the morning or in the evening after the heavy fumes of the rush hour traffic are gone. Jogging is fun and it keeps you in shape.

"And what does that have to do with my 64?" Well, in this case the computer is sort of going to play the role of our conscience. Using the computer we developed a training plan which can be hung beside our bed or in the bathroom (over the bathroom scale) and which reminds us about our fitness plan. At the same time this plan can be used to write down the data of our daily run.

We will use the many possibilities which DATAMAT offers to print out the list that we need. I don't need to say much about this. You can see for yourself:

****

JOGGING PLANNER

****

Monday			
7:30 am	Loc	Central Park	Jogging
7:00 pm	Loc		Highschool Gym
Tuesday			
7:30 am	Loc	Central Park	Jogging
:	Loc		
Wednesday			
7:30 am	Loc	Central Park	Jogging
9:00 pm	Loc		Tennis center
Thursday			
7:30 am	Loc	Central Park	Jogging
9:30 pm	Loc		Tennis center
Friday			
:	Loc		
8:00 pm	Loc		River trail
Saturday			
10:00 am	Loc	Zoo Park	with the children
5:00 pm	Loc		
Sunday			
1:00 pm	Loc		Bicycling
:	Loc		

5.9 Electronic diet plan

```
############################################################
*             Requirements                                 *
############################################################

############################################################
*             Hardware:                                    *
############################################################
*                                                          *
*             Commodore 64                                 *
*             Datasette or disk drive                      *
*             Monochrome monitor                           *
*                                                          *
############################################################

############################################################
*             Software                                     *
############################################################
*                                                          *
*             None                                         *
*                                                          *
############################################################

############################################################
*             Prerequisites                                *
############################################################
*                                                          *
*             Knowledge of Commodore BASIC                 *
*                                                          *
############################################################
```

There are so many different methods for losing weight. Potato-, egg-, rice-, fruit-only days and dehydration, 1000 calorie and 0 calorie diets. They are all tested and tried and are guaranteed not to be bad for your health (say the proponents of the different methods).

Everyone at least agrees on one thing: Only through a combination of the right food and enough physical activities can we get rid of our beer bellies. We have already used our '64 for the physical activities part and now we come to the diet program.

I have in front of me a small plain brochure with the title "Calorie Tables." This book contains an (almost) complete list of the chemical make-up of almost every type of food. That is to say not only calories or joules as it is called today but also the component parts of vitamins, minerals, protein, fat, and carbohydrates.

These tables should be a basis for almost every type of diet program, but first a little theory.

There are all sorts of strange numbers floating around about the number of calories per day which you require. Sometimes 2000, sometimes 1400, and occasionally 3000 calories per day are considered to be the correct amounts. Nonsense! The daily requirement of calories must be individually computed and in our program that will be taken care of automatically. Just one piece of basic information: A normal white collar worker should not require more than 2600 calories if his weight is normal. This is about 20 glasses of beer, 4 bags of potato chips or

three-quarters of a pound of walnuts. You can also of course get your energy in the form of 20 cups of fruit yogurt, if
you'd rather.

The following program is more or less the first stage in a diet program. It computes the normal weight from the input or in the case of overweight, it computes the number of calories which will cause you to lose weight.

I have tried to bring the whole thing in a form which I hope will be fun because with such a sensitive issue the computer has got to show that he's a person just like anyone else.

The second stage would be a program which can further process the diet data. That is to say, a program which could select the correct diet plan out of the list of all your favorite foods and drinks.

I have not included a listing of this program, as I don't think we should get into the problem of the different types of diet forms. We would spend so much time with that issue we wouldn't take off any weight.

By the way: our program "Pounds off" is a good thing to have fun with at a party. You know, for those people who are always asking: "What do you actually do with your computer?" They like those kinds of things.

Note: The initial input of the weight is in pounds, which the program then converts to kilograms.

```
100 rem ****************************************
110 rem *                                      *
120 rem *           Take it off                *
130 rem *           fun with a serious         *
140 rem *           background                 *
150 rem *           out of the data becker     *
160 rem *           idea book (c) 1984         *
170 rem *                                      *
180 rem *                                      *
190 rem ****************************************
200 rem
210 poke 53280,7
220 poke 53281,7
230 poke 53272,23
240 print chr$(149)
250 print chr$(147); : print "";
259 rem **********************
260 rem *** program start ***
261 rem **********************
300 print "HELLO" : print
310 print "My name is 'pounds off'"
320 print "Call me OFF! for short" : print
330 print "Have you ever used"
340 input "this program (y/n) "; in$
350 if in$="y" or in$="Y" then 11000
360 goto 10000
999 rem  ********************************
1000 rem ***    keyboard subroutine    ***
1001 rem ********************************
1010 poke 214,23 : sys 58640
1020 print "Please strike any >key<      "
1030 get t$ : if t$="" then 1030
1040 print chr$(147); : print "" ;
```

```
1050 return
1099 rem ******************************
1100 rem ***   subprogram color    ***
1101 rem ******************************
1110 poke 53280,12
1120 poke 53281,15
1130 print chr$(151)
1140 input "Better (y/n) ";in$
1150 if in$="y" then return
1160 print : print "Always complaining! Ok..."
1170 poke 53280,7
1180 poke 53281,7
1190 print chr$(149)
1200 return
1299 rem ********************************
1300 rem ***   ask name - subroutine   ***
1301 rem ********************************
1310 input " First name "; in$
1320 ha$=in$:an$(1)="you":an$(2)="your":an$(3)="you":
     an$(4)=""
1330 print : input "Male or female  (m/f) ";in$
1340 if in$="f" then an$(4)=""
1350 return
1399 rem ***********************************
1400 rem ***   first name - subroutine   ***
1401 rem ***********************************
1410 rem
1420 input "Shall we go by first names (y/n) ";in$
1430 if in$="y" then return
1440 print : print "Too bad. then I need your "
1450 input "last name "; in$
1460 an$(1)="You": an$(2)="Your" : an$(3) ="You"
1470 na$=na$+" "+in$
```

```
1480 return
1499 rem ***************************
1500 rem ***   age subroutine   ***
1501 rem ***************************
1510 print an$(2) ; : input " Age in years ";in$
1520 in=val(in$) : if in >10 and in<100 then 1550
1530 if in<=10 then print "Come back again in 2 years !"
1540 if in>=100 then print "Anyone that old can be fat!"
1545 gosub 1000 : goto 1510
1550 al%=in : return
1599 rem ******************************
1600 rem ***   weight subroutine   ***
1601 rem ******************************
1610 print an$(2) ; : input " weight in pounds";in$
1620 in=val(in$)*2.2 : if in>20 and in<200 then 1650
1630 if in<=20 then print  "Please don't cheat !"
1640 if in>=200 then print "I don't believe you!"
1645 gosub 1000 : goto 1710
1650 gw%=in : return
1699 rem ******************************
1700 rem ***   height subroutine   ***
1701 rem ******************************
1710 print "And what is ";an$(2); : input "height ";in$
1720 in=val(in$) : if in>20 and in<240 then 1770
1730 if in<=20 then print "even dwarfs started small."
1740 if in<=20 then print "but that small? no cheating !"
1750 if in>=240 then print "I'm not prepared for  "
1760 if in>=240 then print "giants ."
1765 gosub 1000 : goto 1810
1770 gr%=in : return
1799 rem ********************************************
1800 rem ***   overweight calculation subroutine   ***
1801 rem ********************************************
```

```
1805 if an$(4)="r" then bo%=0 : if an$(4)="" then bo%=10
1810 ng%=(gr%-(100-bo%))+int(al%/20):ig%=ng%-(ng%/20):
     ug%=gw%-ng%
1820 if ug%<=0 then 1870
1830 if ug%>0 then ug$="over "
1840 if ug%>0  and ug%<3 then mu$="we'll get that off fast."
1850 if ug%>3 and ug% <10 then mu$=" That's bad for your
     heart: lose weight!"
1860 if ug%>=10 then mu$="Your life is in danger."
1870 if ug%<=0 then ug$= "under"
1880 if ug%<=0 then mu$="this is not a feeding program." :
     ug%=ug%*(-1)
1890 return
1999 rem ****************************
2000 rem ***   physical stress   ***
2001 rem ****************************
2010 print : Print " 1 = No physical activity   "
2020 print " 2 = Blue collar worker or housewife "
2030 print " 3 = Hard worker"
2040 print " 4 = Woodsman or Mason"
2050 print " 5 = Professional sportsman"
2060 print : input "your evaluation"; in$ : in=val(in$)
2070 if in<0 or in>5 then print "" : goto 2060
2080 if in=5 then print" Professional and overweight?"
2090 if in=5 then print "Wait'll your coach finds out..."
     : in=3
2095 kb%=in : return
2099 rem ****************************
2100 rem ***   sports subroutine   ***
2101 rem ****************************
2110 print : print " 1 = Me and sports?!"
2120 print "  2 = Bowling and lifting six-packs"
2130 print "  3 = A little bicycling or swimming "
```

```
2140 print "  4 = Sure! At least 3 hrs/wk "
2150 print "  5 = Very athletic"
2160 print : input "And      ??????????????";in$
2170 in=val(in$) : if in<1 or in>5 then print "" :
     goto 2160
2180 sp%=in : return
2199 rem ***************************************
2200 rem ***   compute calories subroutine   ***
2201 rem ***************************************
2210 kb%=(kb%-1)*400 : sp%=(sp%-2)*150 : en%=2400+kb%+sp%
2220 if al%<15 then en%=en%+200
2230 if  al%>15 and al%<19 then en%=en%+200
2240 di%=100*ug% : di%=en%-di%
2250 if di%<1000 then di%=1000
2260 if en%-di%<200 then km$="go for your ideal weight."
2270 if di%<1200 then km$="what must be must be." return
2280 if di%>1200 and di%<1500 then km$="courage!": return
2290 if di%<1500 and di%<1800 then  km$="no problem. or ?"
     : return
2295 if di%>=1800 then km$="live the good life": return
2999 rem ******************************
3000 rem ***   tables subroutine   ***
3001 rem ******************************
3010 pd$(1)=str$(al%) : pd$(2)=str$(gw%) : d$(3)=str$(gr%)
3020 pd$(4)=str$(ug%) : pd$(5)=ug$+"weight": pd$(6)=
     str$(kb%)
3030 pd$(7)=str$(sp%) : pd$(8)=str$(en%) :
     pd$(9)=str$(di%) : pd$(10)=str$(ng%)
3040 return
3099 rem ************************************
3100 rem ***   display tables subroutine   ***
3101 rem ************************************
3110 gosub 1000
```

```
3120 print : print "Pounds-off table for ";na$ : print
3130 print ">>>"+km$+"<<<" : print
3140 for n=1 to 10
3150 :   read n$
3160 : print n$;pd$(n)
3170 next
3180 restore : return
3190 data "age      =","weight    =","height ="
3191 data "=","","increase 1 =","increase 2 ="
3192 data "Requirement =","Diet =","Nml. Wgt ="
3199 rem ***********************************
3200 rem ***   save tables subroutine   ***
3201 rem ***********************************
3210 ff$="                "
3220 if len(na$)>15 then dn$=left$(na$,15)
3230 if len(na$)<15 then dn$=na$+left$(ff$,15-len(na$))
3240 print ">"+ff$"<"
3250 print ""+dn$ : print "the table will be saved  "
3260 print "under this name. Please be patient!"
3270 open 1,8,2,u$+dn$+"s,w"
3280 for n=1 to 10 : print#1,pd$(n) : next
3290 close 1 : return
3299 rem ***********************************
3300 rem ***   read tables subroutine   ***
3301 rem ***********************************
3310 ff$="              "
3320 print "What was your name ?"
3330 input na$
3340 print : print "I first have to read the "
3350 print "table into memory"
3360 if len(na$)>15 then dn$=left$(na$,15)
3370 if len(na$)<15 then dn$=na$+left$(ff$,15-len(na$))
3380 open 1,8,2,dn$+"s,r"
```

```
3390 for n=1 to 10
3400 :   input#1,pd$(n)
3410 next n
3420 al%=val(pd$(1)) : gw%=val(pd$(2)) : gr%=val(pd$(3))
3430 kb%=val(pd$(6)) : sp%=val(pd$(7)) : en%=0 : di%=0
3440 close 1 : gosub 3500
3450 if ff=0 then return
3460 if ff=62 then 3320
3499 rem ************************
3500 rem ***   error channel   ***
3501 rem ************************
3510 open 15,8,15
3520 input#15,ff,fb$,sp,se
3530 close 15
3540 if ff<>0 then print ff;fb$;sp;se : gosub 1000 : return
3560 if ff=0 then return
9999 rem  ****************************
10000 rem ****   main program (1)   ***
10001 rem ****************************
10010 rem
10020 u$=""
10030 print : print " You want to lose weight!"
10040 print "Good I'll help you. I need"
10050 print "some information."
10060 print "(just like a computer, isn't it...)"
10070 gosub 1000 : gosub 1300 : gosub 1000 : gosub 1400
      : gosub 1000
10080 print "by the way" ;an$(3);" is the picture":
      input "too colorful (y/n)  ";in$
10090 if in$="y" then gosub 1100
10100 gosub 1000 : print "now I need ": gosub 1500
10110 gosub 1000 : print "the normal info... " : gosub 1600
10120 gosub 1000 : gosub 1700 : gosub 1800 : gosub 1000
```

```
10130 if an$(1)="you" then ha$="have"
10150 print "ok ";an$(1);" ";ha$;" ";ug%;" kg";ug$+"weight"
10160 print : print an$(2);" normal weight is";ng%;"Kilo!"
10170 print "or";ig%; "ideal weight."
10180 print : print "Dear "+an$(4)+" "+na$;",this is:"
10190 print mu$
10200 if mu$="This is not a feeding program." then 10490
10210 gosub 1000 : print "Everyone has a different
      lifestyle and";
10220 print "a different physical stress level."
10230 print "how is it with you"; an$(3); "?": gosub 2000
10240 gosub 1000:  print "interesting.  I know sports are
      tough,"
10250 print "but good for you"
10260 print "I need ";an$(2);" answer."
10270 gosub 2100 : gosub 2200
10280 gosub 1000 : print "one more time:"
10290 print : print ug$+"weight=";ug%;"kilo"
10300 print "normal weight=";ng%;"kg ("ng%*.454"pounds)"
10310 print " ideal weight=";ig%;"kg ("ig%*.454"pounds)"
10320 print : print "If your normal daily"
10330 print "energy requirement is";en%;"thousand calories"
10340 print "I would recommend consumption"
10350 print "of no more than;
10360 print di%;" cal/day."
10370 print "My comments:"
10380 print : print ">>>"+km$+"<<<"
10390 gosub 1000: gosub 3000
10400 input "should I show the table again";in$
10410 if in$="n" then 10430
10420 gosub 3100
10430 print : input "shall I print the table";in$
10440 if in$="n" then 10460
```

```
10450 open4,4,2: cmd 4: gosub 3120 : print#4 : close 4
10460 input "and/or store on diskette";in$
10470 if in$="n" then 10490
10480 gosub 3200
10490 gosub 1000 : print "well that's it."
10500 print "please honor me again with your presence!!"
10510 rem
10520 gosub 1000 : print "***END***" : end
10999 rem ****************************
11000 rem ***   main program (2)   ***
11001 rem ****************************
11010 gosub 1000 : gosub 3300 : gosub 3100
11020 if left$(dn$,2)<>"Mr" and left$(dn$,3)<>"Mrs" then
      gosub 11900
11030 if left$(dn$,2)="Mr" or left$(dn$,3)="Mrs" then
      gosub 12000
11040 print an$(2);" data."
11050 gosub 1000 : input "Has anything changed (y/n) ";in$
11060 if in$="n" then 10430
11070 print : print " 1 = age"
11080 print "     2 = weight"
11090 print "     3 = height "
11100 print "     4 = physical stress"
11110 print "     5 = sports"
11190 print : input "What";in$ : in=val(in$)
11200 if in>0 and in<6 then 11220
11210 print "you made a mistake.":gosub 1000 : goto 11070
11220 on in goto 11230,11270,11340,11380,11430
11230 al%=0 : gosub 1000: print "happy birthday!";
11240 gosub 1500 : gosub 1000 : input "Has anything else
      changed  (y/n) ";in$
11250 if in$="n" then 11480
11260 print chr$(147);:print "";:goto 11070 : gosub 1000
```

```
11270 gw%=0 : input "gained weight (y/n) ";in$
11280 if in$="y" then print : print "you didn't watch your
      diet?"
11290 if in$="n" then print : print "you see it works!!!"
11300 print "I'd like to know your new weight.":gosub 1600
11310 gosub 1000 : input "anything else?";in$
11320 if in$="n" then 11480
11330 print chr$(147) ; : print ""; : goto 11070
11340 gr%=0:"stood out in the May rain":gosub 1700 : gosub
      1000
11350 input "has anything else changed ";in$
11360 if in$="n" then 11480
11370 print chr$(147); : print "";: goto 11070
11380 kb%=0 : print "I'd like a new job"
11390 print "too. What is it?" : gosub 2000
11400 gosub 1000 : input "but except for that everything
      is the same";in$
11410 if in$="y" then 11480
11420 print chr$(147); : print ""; : goto 11070
11430 sp%=0 : print "well we'll see if it's"
11440 print "gotten better." : gosub 2100
11450 gosub 1000 : input "anything else (y/n) ";in$
11460 if in$="n" then 11480
11470 print chr$(147); : print ""; : goto 11070
11480 gosub 1000 :  print "we'll see what comes out  this
      time."
11490 gosub 1800 : gosub 2200 : u$="@:" : goto 10420 :end
11900 an$(1)="you" : an$(2)="your" : an$(3)="you"
11910 return
12000 an$(1)="you" : an$(2)="your" : an$(3)="you "
12010 return
```

5.10 An intelligent dictionary

```
#########################################################
*             Requirements :                            *
#########################################################

#########################################################
*             Hardware :                                *
#########################################################
*                                                       *
*             Commodore 64                              *
*             1541 disk drive                           *
*             Monochrome monitor                        *
*                                                       *
#########################################################

#########################################################
*             Software :                                *
#########################################################
*                                                       *
*             None                                      *
*                                                       *
#########################################################

#########################################################
*             Prerequisites :                           *
#########################################################
*                                                       *
*             Good knowledge of BASIC                   *
*             Experience with use of relative files     *
*                                                       *
#########################################################
```

I am going to assume 2 things here: First, kids (computer fans under the age of 18) know more about programming than do older people and secondly, games are out and education is in.

It could be that I'm wrong, but I want to talk about computer-aided instruction nonetheless. In the case of the following program we will be dealing with a self-learning dictionary. Just what do I mean by that?

One example from the area of German-English dictionaries: We enter the German word "Klobuerste" and want to know what it means in English. The program answers: "I don't know. What does that mean?" After entering "toilet brush," the program automatically places the new word in its inventory.

The program can also do the following:

- ask for vocabulary
- store vocabulary

This example is perhaps a little esoteric. Not every student is going to need a foreign language dictionary. As an excuse I can only say that this example has the benefit that only relatively small amounts of words need be handled. In reality this application is not all that unusual if we think about similar problems, such as technical dictionaries. The listing can also be changed without too much difficulty to meet the requirements of any similar type of problem.

The largest and unfortunately unavoidable problem is the amount of data which can be handled: If we consider a record of 20 characters for the English as well the German word, we cannot get more than 300 different words in one file (that is to say, on one diskette). This is relatively little. According to the experts, a basic vocabulary in English contains between 5,000 and 12,000 words. If the program is to be used less for translations than as a learning aid, this will not be a problem. We can simply use more than one file: one for verbs, one for nouns and one for idioms, etc.

If you have not yet used a program which makes us of relative files, you should get yourself a good book, such as The Anatomy of the 1541 Disk Drive from Abacus Software, before you try to change the following listings.

And now the listing.

```
1 rem * * * * * * * * * * * * * * * * * * * *
2 rem       foreign language dictionary
3 rem *                                       *
4 rem   from the data becker "idea book 64"
5 rem *                                       *
6 rem      version 3              6/1/84
7 rem * * * * * * * * * * * * * * * * * * * *
8 rem
9 rem
10 poke 53272,23          : rem upper/lower case
20 poke 53280,12          : rem border color medium grey
30 poke 53281,12          : rem background color light grey
40 print chr$(151)        : rem text color dark grey
50 rem
60 ff$="                                        "
70 fr$=ff$+"                                   "
80 cr$=chr$(13)           : rem >>return<<
90 dim ix$(1500)          : rem index table
99 goto 50000
10000 rem * * * titles * * *
10001 rem
10010 print chr$(147);
10020 print "***-----------------------------***";
10030 print "***                             ***";
10040 print "***     Foreign Dictionary      ***";
10050 print "***                             ***";
10060 print "***-----------------------------***";
10070 print : print
10080 return
11000 rem * * * input/command line * * *
```

```
11001 rem
11010 cz=23 : cs=0
11020 poke 214,cz : poke 211,cs : sys 58640
11030 return
12000 rem * * *  read keys  * * *
12001 rem
12010 gosub 11000
12020 print "          >press key<          ";
12030 get g$ : if g$="" then 12030
12040 print "" : print ff$;
12050 return
20000 rem * * *  read error channel  * * *
20001 rem
20010 open 15,8,15
20020 input#15,ff,fb$,sp,se
20030 close 15
20040 fm$=str$(ff)+" "+fb$+" "+str$(sp)+" "+str$(se)
20050 return
21000 rem * * * format disk * * *
21001 rem
21010 gosub 11000 : print ff$; : gosub 11000
21020 print "  Disk is being formatted     ";
21030 open 15,8,15
21040 print#15,"n:lexicon,11"
21050 close 15
21060 gosub 20000
21070 if ff=0 then 22000
21080 gosub 11000
21090 print "error message: ";fm$;
21100 gosub 12000
21110 gosub 11000
21120 print "Take a new diskette, repeat procedure";
21130 gosub 12000
```

```
21140 goto 21000
22000 rem * * * create relative file * * *
22001 rem
22010 gosub 10000
22020 poke 214,9 : poke 211,0 : sys 58640
22030 input "  number of foreign words? 1500";in$
22040 in=val(in$) : if in>1500 then 22020
22050 print: print "the file is called >lexicon< "
22060 print " A record has  max.  >85< characters".
22070 gosub 11000
22080 print "File will be created ";
22090 rn%=in : hb=int(rn%/256) : lb=rn%-hb*256
22100 open 1,8,2, "lexicon,l,"+chr$(86)
22110 open.2,8,15,"p"+chr$(2)+chr$(lb)+chr$(hb)+chr$(1)
22120 print#1,chr$(255)
22130 close 2 : close 1
22140 gosub 20000
22150 if ff=0 or ff=50 then return
22160 gosub 11000
22170 print "error message:";fm$;
22180 gosub 12000
22190 gosub 11000
22200 print "deleting files. repeat procedure";
22210 gosub 12000
22220 open 14,8,15
22230 print#14,"s:lexicon"
22240 print#14,"v"
22250 close 14
22260 gosub 20000
22270 if ff<>1 then 21000
22280 return
23000 rem * * * load index file * * *
23001 rem
```

```
23010 gosub 10000
23020 gosub 11000
23030 print "index file being loaded ";
23040 open 3,8,2, "index,s,r"
23050 x=1
23060 input#3,ix$(x)
23070 if st<>64 then x=x+1 : goto 23060
23080 close 3
23090 gosub 20000
23100 if ff<>0 then 23120
23110 return
23120 gosub 11000
23130 print "error message:";fm$
23140 gosub 12000
23150 gosub 11000
23160 print "Looking up meaning. Starting from beginning."
23170 gosub 12000
23180 print chr$(147)
23190 end
24000 rem * * * store index file * * *
24001 rem
24010 gosub 10000
24020 gosub 11000
24030 print "  Index file is being stored    "
24040 open 3,8,2,"@:index,s,w"
24050 for x=1 to rn%
24060 :   print#3,ix$(x)
24070 next x
24080 close 3
24090 gosub 20000
24100 if ff<>0 then 24120
24110 return
24120 gosub 11000
```

```
24130 print "error message:";fm$
24140 gosub 12000
24150 gosub 11000
24160 print "Looking up meaning. Starting from beginning."
24170 gosub 12000
24180 print chr$(147)
24190 end
25000 rem * * *  output file parameter * * *
25001 gosub 23000
25020 gosub 11000
25030 print ff$
25040 gosub 10000
25060 print "      *** File parameters *** "
25070 print : print
25080 print   "The  file  >lexicon<  contains   ";x;"
      records."
25090 gosub 12000
25100 return
30000 rem * * * disk/file menu * * *
30001 rem
30010 gosub 10000
30020 print " *** Preparations ***"
30030 print
30040 input "   (1) disk inserted Y";in$
30050 if in$<>"n" and in$<>"N" then 30080
30060 print " > insert disk <"
30070 gosub 12000 : poke 214,10 : poke 211,0 : sys 58640
30080 input "   (2) diskette formatted Y";in$
30090 if in$<>"n" and in$<>"N" then 30125
30100 gosub 21000
30110 return
30120 gosub 12000
30125 poke 214,13 : poke 211,0 : sys 58640
```

```
30130 input " (3) File lexicon exists Y";in$
30140 if in$<>"n" and in$<>"N" then 30160
30150 gosub 22000
30160 gosub 23000
30170 return
40000 rem * * *  Dictionary mode * * *
40001 rem
40005 gosub 25000
40010 open 1,8,2,"lexicon,l,"+chr$(86)
40015 open 2,8,15
40020 gosub 10000
40030 print "   *** Dictionary mode *** "
40040 gosub 11000
40050 print " input foreign words "
40060 poke 214,10 : poke 211,0 : sys 58640
40070 input fw$
40080 fw=len(fw$) : if fw>20 then 40060
40090 if fw=20 then 40110
40100 fw$=fw$+left$(fr$,20-fw)
40110 for i=1 to x
40120 :   if fw$=left$(ix$(i),20) then 40160
40130 next i
40140 fl=1 : rn%=x+1 : goto 45110
40150 fl=0 : goto 40280
40160 ix=len(ix$(i))
40170 i$=right$(ix$(i),ix-20)
40180 rn%=val(i$)
40190 hb=int(rn%/256) : lb=rn%-hb*256
40200 print#2,"p"+chr$(2)+chr$(lb)+chr$(hb)+chr$(22)
40210 input#1,ue$
40220 for j=1 to 63
40230 :   j$=mid$(ue$,1,j)
40240 :   if j$=" " then ue$=left$(ue$,j) : goto 40260
```

```
40250 next j
40260 poke 214,12 : poke 211,0 : sys 58640
40270 print ue$
40280 gosub 11000 : print ff$; : gosub 11000
40290 input "More questions Y";in$
40300 if in$="n" or in$="N" then 40320
40310 goto 40050
40320 close 2 : close 1
40330 return
45000 rem * * * expand dictionary * * *
45001 rem
45005 gosub 25000
45010 rn%=x+1 : fl=0
45015 open 1,8,2, "lexicon,l,"+chr$(86)
45020 open 2,8,15
45025 gosub 10000
45030 print " *** EXPAND DICTIONARY *** No. : ";rn%
45040 gosub 11000
45050 print " enter foreign word "
45060 poke 214,10 : poke 211,0: sys 58640
45070 input fw$
45080 fw=len(fw$) : if fw>20 then 45060
45090 if fw=20 then 45110
45100 fw$=fw$+left$(fr$,20-fw)
45110 gosub 11000
45120 print " enter translation "
45130 poke 214,12 : poke 211,0 . sys 58640
45140 input ue$
45150 ue=len(ue$) : if ue>64 then 45130
45160 if ue=64 then 45180
45170 ue$=ue$+left$(fr$,64-ue)
45180 ix$(rn%)=fw$+str$(rn%)
45190 le$=fw$+cr$+ue$
```

```
45200 hb=int(rn%/256) : lb=rn%-hb*256
45210 print#2,"p"+chr$(2)+chr$(lb)+chr$(hb)+chr$(1)
45220 print#1,le$
45230 if fl=1 then 40150
45240 gosub 11000 : print ff$ : gosub 11000
45 250 input " Other inputs   Y";in$
45260 if in$="N" or in$="n" then 45300
45270 poke 214,7 : poke 211,35 : sys 58640
45280 rn%=rn%+1 : print rn%
45290 goto 45040
45300 close 2 : close 1
45310 return
49000 rem * * * end program * * *
49001 rem
49010 gosub 10000
49020 gosub 24000
49030 gosub 11000
49040 input "End program N";in$
49050 if in$="y" or in$="Y" then 49070
49060 goto 50000
49070 print chr$(147)
49080 poke 214,10 : poke 211,12 : sys 58640
49090 print " *** End ***"
49100 end
50000 rem * * * main program * * *
50001 rem
50010 gosub 10000
50020 print "           *** menu ***"
50030 print
50040 print "    (1) Disk/file preparation"
50050 print "    (2) Dictionary mode"
50060 print "    (3) Expand dictionary"
50070 print "        ---------------------"
```

```
50080 print "    (0)  End program              "
50090 gosub 11000
50100 input "      >Please choose<  1";in$
50110 in=val(in$): if in>3 then 50090
50120 if in=0 then 49000
50130 on in gosub 30000, 40000, 45000
50140 goto 50000
50150 rem
50160 rem * * * * * * * * * * * * * * * * * *
50170 rem *          end of program          *
50180 rem * * * * * * * * * * * * * * * * * *

ready.
```

5.11 Literature data bank

Very important. This type of data bank can be very useful for anyone who is studying, from a high school senior to a candidate for a masters degree.

Persons who must learn intensively and wish to receive good grades need some kind of card file for formulas, vocabularies, and other references. A literature data base is the final answer to the question "I read that somewhere. Where was it?"

Data base? We have to redefine that term just a bit since I have already admitted that a real data base cannot be managed with DATAMAT. It is, however, possible to simulate a data base because of the ability to access different files with various formats. I don't think it would be very useful at this point to present a universal guide for the solution of this problem. The various requirements would be much too different. We will therefore simply present some general information so that you can carry on yourself.

```
::::::::::::::::::::::::::::::::::::::::::::
::          Literature File                 ::
::            for the C64                   ::
::::::::::::::::::::::::::::::::::::::::::::
::                                          ::
:: Title  :  |Idea Book 64          |       ::
:: in     :  |      | No. |        |       ::
:: Author :  |R. Bartel            |       ::
:: Publish:  |Abacus Software      |       ::
:: Year   :  |84|                          ::
::                                          ::
:: Page(s)   |          |                  ::
:: Short desc.|diverse applications|        ::
:: Contents  |Ideas for word proc.|        ::
::           |music, data proc.,  |        ::
::           |graphics, calculat. |        ::
::           |8 programs          |        ::
:: References|other DB books      |        ::
::::::::::::::::::::::::::::::::::::::::::::
```

The main problem is the choice of the key or search criteria. They must be short because of the search times but also tell us enough so that we don't delete it because we think it contains something which we don't need any more (key: "OLD.19.55." is a little too cryptic.)

It is also not all that easy to put the required information into the room available. The best thing to do is to begin by experiment, either with a small or fictitious file.

If you don't want to do without your old-fashioned card file, you can print the records on appropriately sized labels which can then be placed on the cards.

Just one extra tip: The file should always

a) be brought up to date on a routine basis
b) copied to a second backup diskette

Let me then wish you a lot fun in your studies and an A in every course.


1st
FIRST PUBLISHING LTD