Microcomputer Power

$16.95

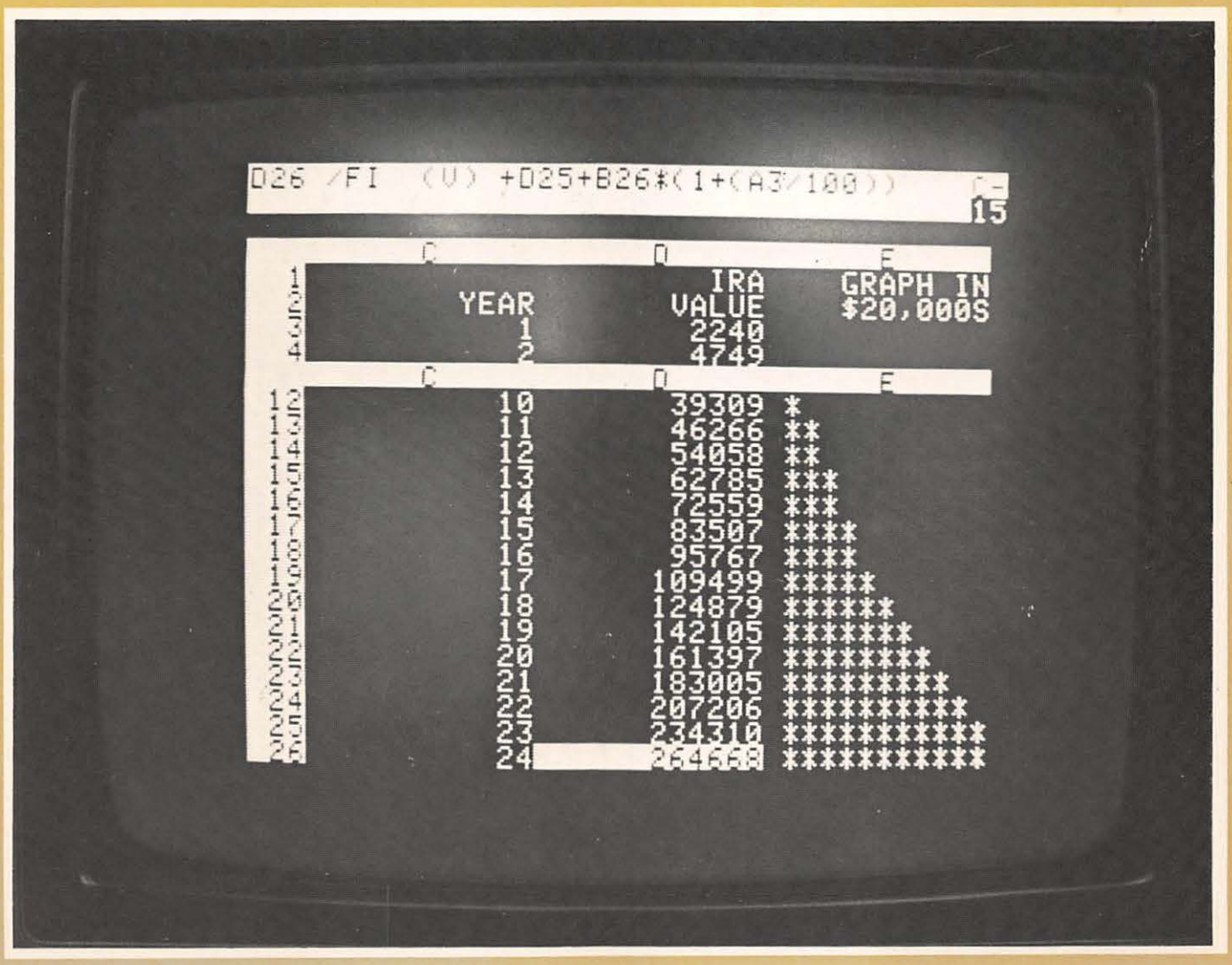
Easy to follow, step-by-step instructions
Use the VisiCalc worksheets to

- calculate sales projections
- evaluate your portfolio
- log your expenses
- compute your IRA account
- determine your net worth

Written for the novice and the professional.

VisiCalc®
for the Apple® II Plus Computer

Edouard J. Desautels



VisiCalc®

for the Apple® II Plus Computer

Microcomputer Power

VisiCalc®
for the Apple® II Plus Computer

Edouard J. Desautels
University of Wisconsin
Madison

wcb
Microcomputer Power Series
Wm. C. Brown Company Publishers
Dubuque, Iowa

Microcomputer Power Series

Second Printing, 1983

Library of Congress Catalog Card Number: 82–72690

ISBN 0–697–09968–7

Printed in the United States of America 2-09968-02

Contents

Preface

The VisiCalc® program makes a computer as easy to use as a calculator, and it gives you far more power than a calculator does. This little book is intended to show you how you can use the VisiCalc program effectively. This book is designed so you do not have to have any previous experience with computers to use the VisiCalc program.

The first chapter sketches the kind of situation in which the VisiCalc program excels. The next chapter shows you how to start using the Apple II Plus Computer. Then we start looking at the specifics of using the VisiCalc program. Although it is easier to keep reading the chapters in the order in which they appear, you can often skip ahead if you wish to look into some feature of special interest to you. A comprehensive index will help you find your way.

Most of the features of the VisiCalc program are presented while solving a sequence of realistic problems, such as evaluating the accumulation in an IRA (Individual Retirement Account). Some of these problems are solved a second time, so you can better appreciate the contrast between different approaches.

Almost everyone has to come to grips with numbers, lots of them. The VisiCalc program is such a powerful assistant in helping you manage numeric information that you should seriously consider equipping yourself with this tool. Perhaps this book will help you make such a decision.

Although this book contains detailed instructions on how you can yourself do everything that is shown, an optional diskette may be used to reduce the typing which would otherwise be required. See the appendix for further information. It is assumed that you have already purchased the appropriate version of the VisiCalc software. If not, you may obtain it from your Apple II Computer supplier.

Chapter 1

WHY USE THE VISICALC PROGRAM?

Electronic computers have been in use for over 30 years. Why is it that the VisiCalc program is one of the best-selling computer software packages since the beginning of the computer age? The Time magazine article on the VisiCalc program gives you some idea of its financial success. What makes the VisiCalc program so attractive?

The Smash Hit of Software

Daniel Bricklin, 29, and Robert Frankston, 31, a team of new-wave composers, have penned a dynamite disc that has grossed an estimated $8 million. It is not a punk-rock smash, but an unmelodic magnetic number called VisiCalc, the bestselling microcomputer program for business uses. The featherweight sliver of plastic is about the size of a greeting card, but when it is placed in a computer, the machine comes alive. A computer without a program, or "software," is like a $3,000 stereo set without any records or tapes.

Three years ago, Bricklin, then a first-year Harvard Business School student, conceived VisiCalc while struggling with financial-planning problems on his calculator. He enlisted the aid of Frankston, a longtime friend and an expert programmer, to develop a new piece of computer software that would make juggling all those figures easier.

The partnership paid off. Since late 1979 nearly 100,000 copies of nine different versions of VisiCalc have been ordered at prices ranging from $100 to $300. It is far ahead of other business programs like Data Factory and General Ledger, and even outsells the programs for Star Cruiser, Dogfight and other arcade-like computer games.

VisiCalc translates simple commands typed on a keyboard into computer language that the machine then uses to solve problems. It enables a businessman, for example, to manipulate labyrinthine equations to calculate financial trends for his company. If he changes one figure, the machine can tell quickly how that affects the other numbers. A firm that gives its workers a 10% pay hike could estimate how that action would alter its costs, sales, profits, or dividends.

The computer program is being put to a wide range of uses. It helps Allerton Cushman Jr., a New York financial analyst, to project insurance-industry profits during the week and tote up his income taxes on the weekend. The Cabot Street Cinema Theatre in Beverly, Mass., bought VisiCalc to figure out which pattern of movie show times draws the best box-office receipts. An accounting firm in Las Vegas plans to use VisiCalc to tell its gambling-house clients how to position slot machines around the floor to ensure the biggest take. VisiCalc is obviously one composition that is in no danger of fading from the charts.

Reprinted from the March 2, 1981 issue with permission of Time, Inc.

It is rarely the case that financial success is also accompanied by formal recognition of excellence by one's academic peers. The impact of the VisiCalc program is such that the ACM, the Association for Computing Machinery, the foremost professional society of computer scientists, awarded its 1981 Grace Murray Hopper award to Daniel S. Bricklin, the chairman of the board of Software Arts, Inc., which originated the VisiCalc program. He received this award for the excellence and elegance represented by

the development of the VisiCalc program. It is worth noting that this is the first time that any activity involving microcomputers or personal computers has merited the ACM's attention. It is almost as if very small computers were previously regarded as toys. This is definitely no longer the case.

What is it about the VisiCalc program that sets it apart from ordinary computer programming languages? There can be no doubt that any solution that the VisiCalc program can produce can also be produced by writing a computer program to generate that solution. For any problem you might wish to solve using the VisiCalc program, a computer program to solve that problem could be written, using some computer programming language such as BASIC, COBOL, FORTRAN, PASCAL, etc.

That is exactly the reason for using the VisiCalc program. You <u>don't</u> have to write a program to solve your problem when you use the VisiCalc program. You key in the relevant data (there is no way to avoid this completely, no matter what you are using). You then specify how the data items are related, and what answers you want calculated. At this point, the computer expert might object: "This is the same as writing a program." Theoretically that may be true; in practice, it makes all the difference in the world. Writing computer programs can be very tricky and time-consuming, and that is after you have invested your time and effort in first learning the programming language.

When you want to solve a new problem using a computer, you usually have to write a program to solve that problem, using one of the programming languages we mentioned earlier. That usually means you also have to use some kind of editing program just to type in the program you need. Then you may have to prepare a data file, with the help of the editing program.

The data file is considered <u>input</u> to the program you wrote, and the results produced by your program are called its <u>output</u>. We often depict the input as flowing into the program which has been placed into the computer's memory, and show the results produced by your program as flowing out from it, as we see in figure 1.1.

Figure 1.1 Input, program, and output relationships

After you examine the output from your program, you may come to one of several conclusions:

(1) great- let's stop computing.
(2) oops!- there must be a "bug" in the program; try to find it and fix it.
(3) it's ok, but what if ...?

With the VisiCalc program, you are more likely to get the first conclusion first. Number two is much less probable, simply because the VisiCalc program won't let you request many ridiculous computations. The VisiCalc program really shines in the third situation. You simply change the desired number or formula, and you immediately see the consequences. You don't have to fool with an editing program, or even request that your program be rerun.

Case 3 might involve wondering what would happen if some data item had a slightly larger value, or what would happen if the formula in the program was just a little different.

With the conventional approach, the input data is laid out for the convenience of the computer program (or whoever wrote the program). The actual layout of the input data has little if any spatial relationship to the results. With the VisiCalc program, you begin by putting the numbers where they should be at all times. And you decide where the results should appear, in relationship to the other input. If you should change your mind, it is a simple matter to move things around painlessly. If you then decide to change a number, you simply locate its old value where you expect to find it (not at some strange location chosen for a program's convenience). As soon as you change that number, all the other outputs which depend on that number immediately change.

Perhaps the simplest yet the most accurate way of describing how you use the VisiCalc program is to think of the computer keyboard as a pencil, with one of the keys used as an eraser, and the screen is a blank sheet of paper that seems to be as wide and as long as you need. If you forget something, it is easy to cut the sheet and paste in new material. It is not hard to move things around. When you ask for some result to be calculated, your electronic worksheet will remember how the calculation was done. So if you later change one of the values that the calculated result depends upon, the computer will recalculate the new result with no further ado. It actually makes working with numbers pleasant!

A Brief Example

The following simple example will give you a better idea of the difference between solving a problem using the VisiCalc program versus solving a problem using the conventional computer programming language approach.

Suppose you were a budding author, and had just had your manuscript accepted by a publisher. The publisher might have proposed that you be paid royalties based upon the following sliding schedule:

Earn 5% of selling price for the first 3,000 copies.
Earn 7% " " next 4,000 "
Earn 8% " " next 5,000 "
Earn 10% " " for all additional copies.

With the VisiCalc program, you would lay out the essential data in the form of a table, as we see here in table 1.1.

Rate	Copies
5	3,000
7	4,000
8	5,000
10	?

Table 1.1 Raw data

You would then make an educated guess as to what number to use for the ? in table 1.1. You might like to project your potential income (before taxes) if the book sold say 50,000 copies, assuming that it sells for $1 per copy (it is a very small book). So the ? is replaced by 50,000-(3,000+4,000+5,000), giving us 38,000. You then ask the VisiCalc program to work out the product (Rate multiplied by Copies) for each line, which results in table 1.2, since our rate figures are actually percentages.

Rate	Copies	Income
5	3,000	150
7	4,000	280
8	5,000	400
10	38,000	3,800

Table 1.2 Projected income

Of course, what you really want is a running total, so you ask the VisiCalc program to produce another column headed "Total" which is to show for each row the sum of the current and all preceding income figures. That being done, you now see the results in table 1.3.

Rate	Copies	Income	Total
5	3,000	150	150
7	4,000	280	430
8	5,000	400	830
10	38,000	3,800	4,630

Table 1.3 Projected total income

This may seem like much ado about nothing, at this point. You could have done the same thing with a calculator, or merely with paper and pencil. True--but having gone this far, you can now begin the "What If ..." phase. As the budding author, you might want to negotiate a better royalty schedule.

So you begin thinking "What if my first book is not a terrific success? I should ask for a higher percentage for the first few thousand sold, just to be safe." Suppose you settled on 6, 8 and 9%, in place of the 5, 7 and 8 that was offered; you don't want to be greedy. You can now proceed to replace the 5, 7, and 8 by the 6, 8 and 9, and, lo and behold, you immediately see the consequences, as shown in table 1.4.

Rate	Copies	Income	Total
6	3,000	180	180
8	4,000	320	500
9	5,000	450	950
10	38,000	3,800	4,750

Table 1.4 Projected total using new rates

You might think about these totals for a while, and because you are now convinced you have a best seller on your hands, perhaps it would be better to focus on negotiating a better top rate than the 10% that was offered. After all, the other rates will make very little difference if you sell 100,000 copies. So you change the 10% to 20% (dreamer) and you decide to increase the 38,000 by 50,000. The results are shown in table 1.5.

Rate	Copies	Income	Total
6	3,000	180	180
8	4,000	320	500
9	5,000	450	950
20	88,000	17,600	18,550

Table 1.5 New projected totals

If you were in the publisher's shoes, you would also like to be able to do the same juggling of figures. As the budding author, you have much less experience with the consequences of manipulating either the rates, the sales thresholds, or the number of steps in the schedule. The publishers can probably do it in their heads; you could use a little help.

You could of course have written a computer program to perform these calculations. For many computer programs, you have to lay out the numbers to be processed in a data file. The data file for this program could very well look like the following list (using the first set of rates and copies):

```
5,3000,7,4000,8,5000,10,38000
```

This is awkward to read and change. It might have looked like:

```
05 3000      or        05070810
07 4000                03000 04000 05000 38000
08 5000
1038000
```

where you find yourself putting in leading zeroes or leading

blanks with the data, because the exact spacing between these numbers may be critical, depending upon the programming language that was used in writing the program. You could easily provide data which looks correct to the naked eye, but which the program takes to be ten times larger (or ten times smaller) than what you had in mind.

The data file

```
5                3000
7               4000
8                5000
10              38000
```

seems to have the right numbers,even though the alignment for the 4000 and the 10 is a little sloppy. Some computer programs would interpret this 4000 as if you had written 40,000 because the actual position of the number on the line was critical. This is much less of a problem with the VisiCalc program. You immediately see what the VisiCalc program thinks of the number you just typed. If it is not what you intended, you can change it right away.

What About Big Problems?

Suppose you had a really big problem to solve? Could the VisiCalc program handle it? Suppose you were going to be a real-estate tycoon and you were working out a ten-year projected statement of cash flow. Such a statement is a table with at least ten columns of numeric data, and many dozens of rows, depending upon how much detail you want to include in projecting your cash disbursements. Then we also need some labels to keep track of things. If each column is to hold numbers as large as eight digits, and if we leave a little space between columns for ease of reading, the cash flow table will be about 120 characters wide. Since most computer video display terminals (VDTs) can only display 24 or 25 rows of 80 characters at one time, you would be hard pressed to squeeze all of the cash flow table onto the VDT screen at one time. Imagine how much harder it might be with an Apple II with a 40-column display! As it turns out, you can do amazing things with an Apple II equipped with the VisiCalc program.

With the VisiCalc program, you can construct and display tables with as many as 63 columns and 254 rows and you can "browse" over parts of the table very easily. If the whole table won't fit on your screen, the VisiCalc program treats your VDT screen as if it were a "window." Your screen window lets you see any part of the table you wish to see, getting as much of it as will fit on your screen at one time. Trying to do this with a conventional computer program would be far more difficult. Figure 1.2 illustrates the idea of a window.

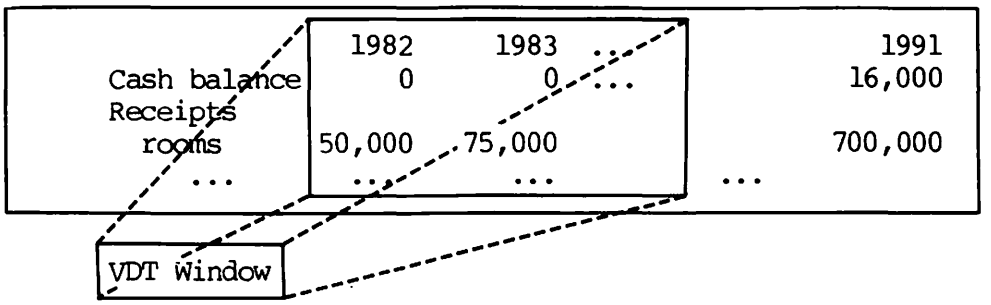


Figure 1.2 VDT Window into a large table

Coming Up

Specifics on using the VisiCalc program: how do you tell the VisiCalc program to do something? What if you made a mistake? What if you were in the middle of filling out a worksheet and had to leave suddenly: how can you save worksheets, and recall them later? These and many other features of the VisiCalc program will be described, discussed and illustrated as we proceed. It goes without saying that you will learn more, faster and better, if you can be using the VisiCalc program and trying the things we are discussing. We hope however to provide sufficiently detailed examples so that you can follow what is going on even if you don't have immediate access to a computer equipped with the VisiCalc program.

SUMMARY

The VisiCalc program is data-oriented; using it is very much like using a calculator. You begin with your own raw data, lay it out on what amounts to an electronic worksheet, using your VDT's screen as an easily erasable worksheet. Then you begin specifying the relationships between your data and the desired results. You build up to the desired end-product in a step-by-step fashion, seeing the results at every step. You always see your input data in the natural spatial relationship it is intended to have with respect to any computed results.

Problem solving with computers, using the conventional approach, is program-oriented rather than data-oriented. Most people who are not computer experts feel more at ease with the data they know well than with the use of unfamiliar computer programming languages. Most people are familiar with the everyday use of a simple worksheet. The VisiCalc program combines the ease of using a calculator and the familiarity of a worksheet with the power of a computer. It follows that most people will find that the VisiCalc program provides a very natural, user-friendly way to make the computer work for you.

Chapter 2

GETTING ACQUAINTED WITH THE APPLE II COMPUTER

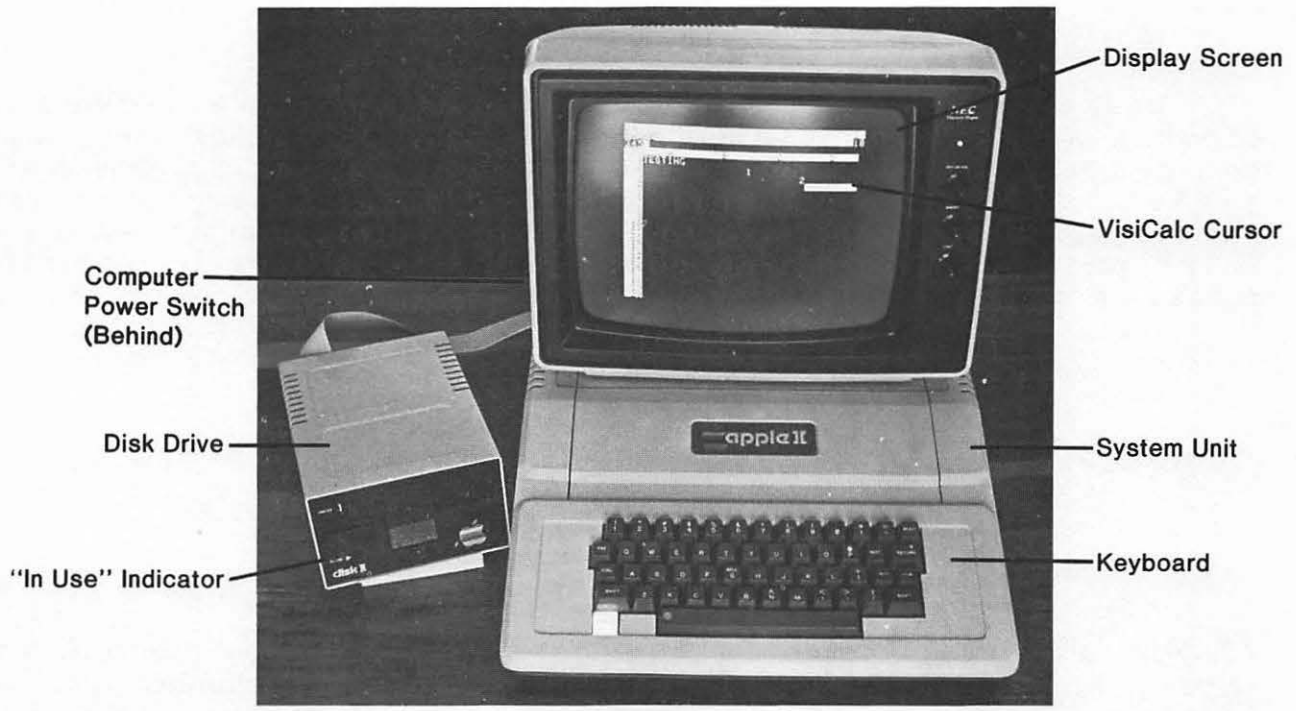
This chapter deals with the operation of the Apple II or Apple II Plus Computer and it introduces some new ideas. No previous experience with computers is required to use the VisiCalc program with the Apple II Computer. Here the rudiments of turning the computer on and setting it up so you can use the VisiCalc program effectively will be described. If you are already familiar with this computer, you might want to merely skim most of this chapter and proceed to the next.

Physical Components of an Apple II Computer

Every Apple II Computer has a system unit which includes a keyboard. The "computer" of the Apple II resides in the system unit. In order to use the VisiCalc program, the computer must have 48K (approximately 48,000 positions) of memory, and a display console and a diskette drive. The computer can have optional devices, such as a printer, as well as a second diskette drive. A diskette drive is also called a floppy disk drive, or simply a disk drive.

Figure 2.1 on the next page shows the display unit above the system unit with the keyboard, and it identifies important items. A disk drive can hold one five-inch diskette. Each diskette can record approximately 127,000 characters of information. Some of these characters may be used to represent computer programs such as the VisiCalc program and some of them are used to represent your data. The word "byte" is often used in place of the word "character"; for our purposes, these words are equivalent.

If you have two disk drives on an Apple II Computer, you can have immediate access to one-quarter of a million characters of information. If your computer has only one disk drive, it is called drive 1. If it has two disk drives, the one furthest from the computer is referred to as drive 2.

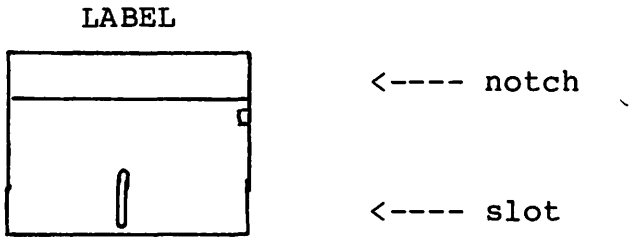


(©Allen Ruid, Photographer, 1982)

Figure 2.1 The Apple II Plus Computer

Using Diskettes

Diskettes are normally stored in protective jackets or sleeves and they should be stored in an upright position. When you are about to use a diskette you should remove its protective jacket. This will reveal a rigid black square protective envelope. Diskettes have an oval slot in the square envelope which protects the flexible (floppy) disk magnetic recording media, as shown below:



If you see a small notch on the edge, near the bottom right of the label, the information on that diskette can be changed and new information added. In such a case, we say that this diskette is "write-enabled." If on the other hand you see a piece of foil in the same position covering the notch, then the computer will refrain from changing any information on that diskette; the diskette is said to be "write-protected."

If you are inserting your one and only VisiCalc program diskette into the diskette drive, make sure its write-protect notch is covered. That will prevent accidental overwriting of your master copy. Your VisiCalc program diskette is identified as being "copy protected." That means you cannot duplicate it, so handle it with care.

When you insert the diskette into the diskette drive, the label side should be facing up and the edge furthest from the label must be inserted into the drive first. Gently push the diskette through the drive door slot. Push the drive door shut.

Each disk drive has an indicator light just below the disk drive slot, to the left, with a label "IN USE." The indicator light will glow whenever the disk drive is being used by the computer. You should not attempt to insert or extract a disk when this light is glowing. Doing so could destroy the diskette and damage the disk drive. If it seems that the indicator light won't ever stop glowing, it would be better to turn off the computer to perform this operation.

Some Preliminaries

The VisiCalc program is very easy to start up and use, as we shall see in a moment. Later on it will be helpful for us to make some use of the Apple II program known as the Disk Operating System DOS. The DOS reference manual has 200 pages. Don't be alarmed! We will discuss what we need to know about DOS in just a few pages, after we have gone further in using VisiCalc.

Operation of the Apple II Computer

We will walk through the steps required to start computing, and we will discuss the purpose of these steps, in case you are wondering about what is going on.

1. Place the diskette labeled "VisiCalc Program Diskette" in drive 1 (remember that drive 1 is the one which has its cable directly connected to the computer and for that reason it will usually be closer to the computer than drive 2).

2. Turn the computer on, by pushing the computer's on/off switch up. It is located on the left rear of the computer. Also turn on your display unit.

3. The computer will then begin reading information from the diskette. Whenever the computer is actively using a disk drive, the corresponding indicator light will glow; never remove a diskette while that light is glowing.

4. After a few dozen seconds, your display screen should greet you with a "blank worksheet" with the title "SOFTWARE ARTS" prominently visible at the top of your screen.

You have just completed "loading" the VisiCalc program. Loading is the process of transferring a copy of a program from a diskette into the computer's memory and once the copy of the program is in the computer's memory, that program starts directing the computer's operation. If you turn the computer off, the information in the computer's memory (also called RAM) will be lost. Fortunately you can retrieve a fresh copy of the program from the diskette.

Computer people have managed to invent more words or find new uses for old words than you would imagine possible, given that computers have only been around a few dozen years. One of the words you will run into is "boot" or "booting." These words are used as synonyms for loading a program.

When you wish to turn off the computer, it is wise to remove any diskettes that may still be in the disk drives. Remember that turning the computer off will erase whatever you have been typing

into the worksheet. So you may wish to save the worksheet on a diskette before you turn the computer off. How you do this and all the other things we have described in this chapter will occupy us for the next few chapters.

The Keyboard

Figure 2.2 shows you the keyboard we will be working with. It is very much like a typewriter keyboard, plus a few extra keys that will prove to be very useful as we go on.

(©Allen Ruid, Photographer, 1982)

Figure 2.2 Apple II Plus Computer Keyboard

You can correct an error while typing an entry by pushing the ESC key located in the second from the top row of the keyboard, at the far left. If you push the ESC key it will erase the last character you typed. You can then type the correct information. When the entry appears to be correct, push the RETURN key (the large key on the far right, second from the top).

In some situations you have to push the RETURN key to cause the VisiCalc program to examine your response, but as a general rule no special character will be displayed at the point you pushed the RETURN key. Note: whenever you see the word RETURN included among other items to be typed, simply push the RETURN key --do not type the 6 letters R, E, T, U, R, and N.)

Using Files

Almost anything you do with a computer involves using "files." A file is a body of information to which you have assigned a name and which you have recorded on a diskette. The VisiCalc program keeps track of which files are where by maintaining a directory of file names on each diskette. The

diskette's directory contains the names of all files on that diskette, and other information about these files. Later on we will see how you can examine a diskette's directory.

If you were running out of space on a diskette, you would look for the largest file you no longer need, and erase or delete it. We will see how the VisiCalc program lets us delete unneeded files later. You can also delete files using a DOS command. Every once in a while it is a good idea to examine a diskette's directory, to see if the diskette is filling up, and perhaps deleting files that are no longer needed.

Your only use of the VisiCalc Program Diskette will be to load the VisiCalc program into the computer's memory. When you get to the point in constructing worksheets where you wish to save your work for future use, you will save your worksheet as a file, on another diskette. Such a diskette is variously described as a storage diskette, a data diskette, or a slave diskette.

If you simply place a freshly purchased diskette in the drive so you can use it to save a worksheet, you will discover that it won't work. A fresh or "blank" diskette has to be initialized before it can be used. This process of initialization is also called formatting. We will see how to do this very soon.

Coming up

The stage is now set for us to begin using the VisiCalc program. It should be both pleasant and useful, which makes it worthwhile.

SUMMARY

We have had a brief look at the Apple II Computer system hardware, and at the process of starting (loading) the VisiCalc program.

Starting the VisiCalc proram involves:

(1) inserting the correct diskette, oval slot first, label side facing up in disk drive 1
(2) pushing the POWER switch for the computer and its display to the ON position
(3) waiting for the disk's IN USE light to go off
(4) removing the VisiCalc program diskette

Use of the Apple II Computer revolves around the creation and processing of entities known as files. We will see how to create, process and delete them when we begin using the VisiCalc program. We will see how we can find out which files are present on a diskette by using the appropriate DOS command.

Chapter 3

USING THE VISICALC PROGRAM: SOME PRELIMINARIES

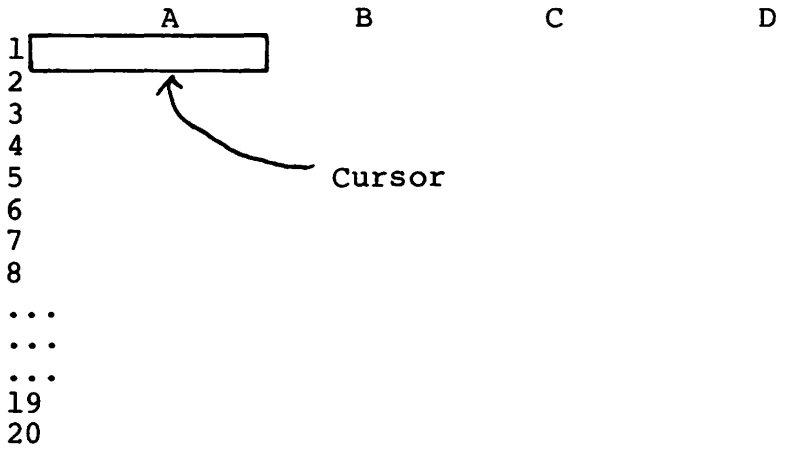
Before embarking on solving complete problems using the VisiCalc program, there are a few basic concepts and operations that we will be using over and over again, and it is better to deal with these sooner rather than later.

Entering the VisiCalc Program

The steps which follow are the same ones we used in chapter 2 to load the VisiCalc program.

(1) Place the VisiCalc program diskette in drive 1.
(2) Turn on the power switches for the display and the computer, and if you have a printer turn its power on. After a slight pause, the screen displays a blank VisiCalc worksheet.
(3) Remove the VisiCalc program diskette. You won't need it again until the next time you wish to use the VisiCalc program.

The blank VisiCalc worksheet which is displayed looks like the following:



You will see all 20 numbers from 1 to 20 displayed on your screen. The VisiCalc program uses a very large cursor; it is the big white rectangle you see in the top left corner, under the column headed A, in row 1, just after you load the VisiCalc program (you can see this large cursor in the photograph of figure 2.1, located at position D4, in column D where it meets with row 4). The cursor indicates where a number you type will be placed on the worksheet. You can move the cursor by using the two arrow keys:

> move the cursor to the right (or down)
< move the cursor to the left (or up)

These cursor arrow keys are on the right part of the keyboard,

just above the SHIFT key. When you press the right-arrow key → you should expect the big VisiCalc program cursor to move over to the right. Then how can the same → key also move the cursor down? Having one key do two things is accomplished by your setting a direction marker. When you first start up the VisiCalc program the direction marker is set to the left-right position and that is displayed at the top-right position on your screen as a minus sign "-". If you want the alternate direction to be in effect, you push the space bar (the widest key on the keyboard). That changes the "-" to a "!" to remind you that pushing → now means ↓ and pushing ← now means ↑. Every time you push the space bar, you will change the "-" into a "!" or the "!" into a "-" and thereby change the effects of the → and ← keys.

If you hold any cursor key down while you push the key labeled REPT you will keep moving the cursor repeatedly. If you should happen to move the cursor right, beyond column D, you will see the missing part of the worksheet come into view. As column E appears on the right, column A will disappear from the left. You could go as far right as column BK, the 63rd column. The 63 columns of a worksheet have the names A, B, ..., Z, AA, AB, AC, ...AX, AY, AZ, BA, BB, BC, ..., BK.

You can return to the original screen appearance by repeated use of the arrow cursor motion key ←. However, you may find it more convenient to use what is known as the GO TO command.

The symbol > (the "greater than" key), not to be confused with the right arrow cursor motion key →, is a VisiCalc command which you have to follow with the coordinates of the worksheet entry you wish to place the cursor on. The coordinates are specified by designating the column letter, immediately followed by the row number. Thus

>A1 RETURN

immediately returns the cursor to the top-left corner of the worksheet and causes the top-left part of the worksheet to be displayed. The GO TO command lets you specify where you want to place the cursor. Your screen's view of the worksheet will be adjusted so the cursor always remains visible.

To make reading these examples easier, we will pretend the Apple II keyboard has the ↑ and ↓ arrow keys. When you see these symbols used in this book, just remember to check the top-right corner of your screen. If you see a ! then push the → to go ↓ and push the ← to go ↑. If you see a - instead of a ! touch the space bar to get the ! then proceed to use the arrow keys. Then remember to touch the space bar the next time you want to go → or ←.

If you look down the worksheet, by pushing the cursor-down arrow key v often enough, you will get to see the bottom row, which is row 254. You could also get there directly by typing

>A254 RETURN

Moving the cursor so that some information disappears from view as other information comes into view is called scrolling. Moving the cursor down far enough causes a horizontal scroll, with the screen acting as a window into the 254 row by 63 column worksheet. You can scroll up or down, left or right, as desired, using the cursor arrow keys. Scrolling does not ever cause you to lose any information. Information may disappear from view, but it will be recalled if you scroll back to it.

Keyboard Use

The ">" key should be clearly distinguished from the cursor-arrow key "→". Since the ">" shares the keycap with the period "." you must also push the SHIFT key to evoke the ">." You may avoid problems in the future by spending a few minutes examining the keys on the keyboard.

Figure 3.1 identifies the keys we have been discussing:

(©Allen Ruid, Photographer, 1982)

Figure 3.1 Apple II Plus Computer Keyboard

Writing on the Worksheet

With the cursor situated at position A1, try typing in a number, such as 345, or -765. While you are typing it, no change takes place at position A1, but you do see what you are typing being displayed above the worksheet, near the top left of the screen. This position is called the edit entry line. You can see the number 345 followed by a small white square in the photo of figure 2.1--that is where the edit entry line is. The digits 345 had just been typed, and the RETURN key had not yet been pushed, so the worksheet cursor at D4 still hovers over a blank entry.

```
345        <--- edit entry line

        A         B         C         D
1
2
3
4
5
6
7
8
9
```

From now on, we will often display only the top few rows of a worksheet when our examples will fit in that small a worksheet. We will see how you handle very large worksheets later.

If you made an error in typing, and notice it right away, you can correct it by erasing characters. Use the ESC key (second row from the top, far left) to erase characters that have just been typed. Do not try to erase characters by using the cursor arrows! Pushing a cursor arrow in this situation is the same as pushing RETURN and then some. We will discuss using the cursor arrows while entering data, in a few moments.

The information displayed on the edit entry line will be copied into the location the cursor is sitting on when you push the RETURN key. This is worth repeating: in order to place a number or a label at some spot on the worksheet, you must first position the cursor on the desired spot either by using the arrow keys or by using a > GO TO command. When you start typing the desired item, it does not go directly into the chosen spot. It is accumulated character by character on the edit entry line. The item will be transferred to the chosen spot if and when you push RETURN or one of the arrow keys. If you decide not to complete typing the item (perhaps because you had placed the cursor at the wrong worksheet location) you can discontinue the operation by simultaneously pushing the keys labeled CTRL and C. Doing so will cancel the operation and erase the item on the edit entry line. The worksheet will remain as it was.

Once a workspace entry has been set, you can change it in two ways. For the time being, you can change it by typing a new value and pushing RETURN, provided the cursor was positioned at the desired location. The edit command which we will see in chapter 5 provides a nicer way to correct errors or make changes.

Try typing something into location A1:

(a) be sure the cursor is in position A1 either by pushing the cursor arrow keys, or by typing >A1 RETURN
(b) type the desired value (perhaps correcting errors with the ESC key while typing)
(c) push RETURN and see the new value at position A1

Leaving the VisiCalc Program

If you want to stop using the VisiCalc program, just turn the computer off. If you want to save what is on the screen for later use, then keep reading until we discuss saving worksheets.

Returning to DOS

You can leave the VisiCalc program and return to DOS without turning your computer off. Insert a DOS Master Diskette in drive 1 and type a command which starts with the symbol / (the "slash" or "divide" key). The VisiCalc program has a set of commands to let you do things such as printing, moving things around, and so on. Each command starts with the symbol /. If you type a / as the first character of a new entry, you should see a prompt above the entry content line:

```
Type /
See the prompt   COMMAND: BCDEFGIMPRSTVW-
```

We can see the prompt line appearing above the worksheet column coordinates A, B, C, etc., in the following figure:

```
COMMAND: BCDEFGIMPRSTVW-    <-- prompt line
                                <-- edit entry line
        A       B       C       D
1
2
3
4
5
6
7
8
9
```

Whenever you are interacting with the VisiCalc program, the current status of your dialogue is reflected in the prompt line. In the photograph of figure 2.1 the prompt line displays the word "VALUE" because the value 345 was being typed.

Continuing our exit from the VisiCalc program, you next type one of the characters selected from the prompt line. To begin to return to DOS, type S. This puts you into the STORAGE command mode, which further prompts you with the line

```
STORAGE: L S D I Q #
```

You next type one of the characters selected from the prompt line. To return to DOS, type a Q. You will get another prompt:

QUIT: SLOT #

If you type a 6 (the usual slot number for disk drive 1), you will see your worksheet disappear, and in a second or two, you will see the DOS prompt line. If you had a DOS diskette in drive 1, the DOS software will be loaded. We will discuss why you might want to do this later. At this point, it serves to introduce the principal way in which you get the VisiCalc program to do things for you, by using the / commands.

Other Commands

The meanings of all of the letters and symbols displayed in response to your typing "/" are shown below. Each one of these commands will be discussed as we proceed, one at a time.

B	blank	erase value/label
C	clear	clear whole worksheet
D	delete	delete a row or column
E	edit	modify an item without retyping it
F	format	format an entry
G	global	set global formats, etc.
I	insert	insert a row or column
M	move	move a row or column
P	print	print segment of worksheet
R	replicate	replicate (copy)
S	storage	see "Storage" below
T	title	freeze rows or columns in place
V	version	copyright notice
W	window	split screen operations
-	repeating label	fill a position with a label

Other "Storage" Commands

The meanings of all of the letters and symbols displayed in response to your typing "/S" are shown below. Each one of these will be discussed as we proceed. You will be amazed at how natural most of these commands are.

L load a file into a worksheet

S save a worksheet as a file

D delete a file

I initialize a diskette

Q quit, leaving the VisiCalc program

load/save a file in the Data Interchange Format DIF

Initializing Diskettes

You can probably guess how this will be done. Recall that a diskette must be initialized or formatted before it can be used to store information. You can use the DOS diskette initialization procedure if you wish (we will discuss it later). However it is simpler to this as follows.

Type /S (storage command)

See the prompt STORAGE: L S D I Q #

Type I (storage-initialize command)

See the prompt INIT DISK: TYPE RETURN. ERASES DISK

If at this point you have a diskette in drive 1 and you push the RETURN key, one of two things will happen:

(1) the disk drive will whir and make noises for about 30 seconds as it erases and initializes your diskette, or

(2) the disk drive will whir for perhaps three or four seconds then stop as the message

ERROR: I/O

is displayed on the prompt line.

In the first case you have succeeded in generating an initialized diskette. Prepare a handwritten label for it. In the second case, if you examine the diskette you had in drive 1, its write-protect notch (recall the diskette diagram we saw in chapter 2) must have been covered. That generally means one of two things:

(1) either you placed the wrong diskette in drive 1, or
(2) you inserted the diskette upside down leaving the write-protect notch in the wrong position.

If you really had the intended diskette in drive 1, remove the foil covering its write-protect notch and repeat the /SI RETURN sequence.

Since the process of initializing a diskette is also called formatting, we will use the words "initialized" and "formatted" interchangeably. If you see the abbreviation I/O being used in a VisiCalc message, it stands for Input/Output and refers to some problem in fetching input or sending output in connection with the disk drive or the diskette.

Saving and Retrieving Worksheets

Suppose we have just turned our computer off. We can reload the VisiCalc program by placing the VisiCalc Program Diskette in drive 1 and turning the computer on. When the worksheet is displayed, the numbers we had typed earlier have disappeared. That was why we were given a chance earlier in using the /Q (for QUIT) command to confirm if we wanted to leave the VisiCalc program. Leaving the VisiCalc program means losing what we have typed onto the worksheet unless we take care to save a copy. You can save a copy of a worksheet by placing a formatted diskette in drive 1 and issuing the /S storage command we just saw. When it gives you the prompt

STORAGE: L S D I Q #

you can ask to save a copy of your worksheet on the diskette, by typing the letter S. You will then get another prompt:

STORAGE: FILE FOR SAVING

You should respond by making up a name with up to thirty alphabetic or numeric characters; the first character of the name must be a letter. The name of a file may have up to thirty characters in it but it may not include a comma or a RETURN code. Suppose we give this file the name TEST.VC. When the file has been copied onto the diskette, the prompt "FILE FOR SAVING" will vanish. In making up file names you should make it a practice to include the suffix ".VC" as part of the worksheet name. That will make it easier for you to keep track of the many files you may soon have. Later on we will use different suffixes when we see other ways of creating files.

If you inadvertently provide a name which is already in use when attempting to store a worksheet you will see the prompt

STORAGE: FILE EXISTS. Y TO REPLACE

If you respond with Y the old file will be replaced by your worksheet. Any other response discontinues the operation; you may issue another /SS command using a different file name if you wish.

When the worksheet has been saved (copied) on the diskette the STORAGE prompt will disappear. If you happen to see the message

ERROR: I/O

on the edit entry line, that means the worksheet was not saved. You could try again (repeat the /SS sequence). If it fails a second time you are probably using an uninitialized diskette or one which is write-protected. If neither is the case, you should try using another diskette.

Retrieving a Worksheet

Suppose we have just loaded the VisiCalc program. Once again we see a blank worksheet. We can retrieve the worksheet we called TEST.VC by typing /S, then L (for load). We will see the prompt

STORAGE: FILE TO LOAD

If we respond with the name of our file, TEST.VC, we will see the message

STORAGE: LOADING*

As soon as that message disappears our old worksheet TEST.VC has been restored, as we can well see on the screen.

Panic Button

What if we typed "/" and did not get the prompt we expected? You can either type several ESCs, or use the CTRL-C key (the notation CTRL-C means you must push the keys marked CTRL and C simultaneously). Either of these will stop whatever operation was in progress and provide you with a blank prompt line so you can type in a fresh command or a new entry value.

Using the RETURN Key

Is there any rhyme or reason with regard to using the RETURN key? Fortunately there are some simple rules that apply; these will help reduce any confusion. Whenever the VisiCalc program can reasonably anticipate that your entry or response is complete, you do not push the RETURN key. For instance, if you are trying to use a command, you can do so by typing a single character "/" whenever the entry content line is empty. You immediately see the prompt line, without pushing the RETURN key:

COMMAND: BCDEFGIMPRSTVW-

If you had typed "/S" without pausing between typing the "/" and

the "S", the preceding prompt would be immediately overwritten by the next prompt:

STORAGE: L S D I Q #

and this might happen so rapidly that you might not have even noticed the earlier prompt. If you typed "/SQ" you would see the prompt

QUIT: SLOT #

You could have typed "/SQ6" all at once and gotten the same effect. You may wish to do it in two stages; "/SQ" first, then "6" next, so that you have a final chance to reconsider, should you change your mind and wish to cancel the QUIT by pushing the CTRL-C key.

In all of the above situations, we did not use the RETURN key even once. The specific situations we were in made it possible for the VisiCalc program to correctly anticipate what we were up to. That is not the case in the following situations.

In typing

>A1

the VisiCalc program does not know whether we intend a "GO TO A1" or whether we are simply pausing before finishing the entry as in

>A10

Even then, the VisiCalc program does not know whether we intend a "GO TO A10" or whether we will continue with

>A102

At this point, the VisiCalc program should know that we cannot go any further (there are only 254 possible row numbers). However, to be consistent, each use of the ">" requires that you push RETURN or some other special key.

The rule can be made even more widely applicable: whenever you are providing a coordinate by typing in the column and row coordinates, you must signal that it is complete by using some special key. So far we have been using RETURN for this purpose. Later we will see what other special keys could be used to our advantage.

In the following chapters, we will be typing information that varies in length. These items may be numbers such as 12345, or names or labels such as SEPTEMBER or DATE. It follows that since the VisiCalc program can't reasonably be expected to know what we had in mind, the completion of each entry of a variable-length item requires a signal from us. That signal will often be

provided by using the RETURN key, as we saw in the GO TO situation.

SUMMARY

Enter the VisiCalc program by inserting the VisiCalc Program Diskette in drive 1 and turn on the display unit and computer power. Then remove the diskette when you see a blank worksheet appear on the screen.

Leave the VisiCalc program by typing /SQ6 or by turning the computer off.

Initialize (format) a diskette by using the /SI command.

Save a worksheet by inserting a formatted diskette in drive 1 then type /SS and provide a new file name.

Retrieve a saved worksheet by typing /SL and providing an old file name.

Position the cursor either by using the cursor arrow keys or by typing the GO TO command ">" followed by the letter and number coordinates of the desired entry.

Correct an entry using the ESC key, provided the entry is still on the edit entry line.

Correct a worksheet entry by placing the cursor over it and retyping the entire correct entry, followed by RETURN.

Use the CTRL-C key if you change your mind and wish to discontinue a particular operation.

Chapter 4

SOLVING A SIMPLE PROBLEM

In this chapter, we will begin working with labels, simple formulas, printing, erasing (clearing) the screen, and a very powerful technique called replication. To make things easier for us, we will reexamine a familiar problem, the author-income problem from chapter 1. Then, just to be certain that these new ideas are well understood, we will apply them to estimating an IRA's value.

Labels

If you begin typing an entry, VisiCalc will attempt to classify the entry in one of four categories:

(1) a value
(2) a command (begins with /)
(3) a GOTO request (begins with >)
(4) a label

A value is a number or it can be a formula (also called an arithmetic expression. It begins with a numeric digit or any one of the following characters: +, -, (, #, @, or a period. A label, also called a title, usually begins with a letter of the alphabet. Some typical labels are: RATE, TOTAL, COST.

Using the Cursor Arrow Keys

We will continue to use the following symbols to represent the cursor arrow keys:

```
>       move right (use > with direction -)
<       move left  (use < with direction -)
↑       move up    (use < with direction !)
↓       move down  (use > with direction !)
              (change direction by pushing space bar)
```

Try typing the labels RATE, TOTAL, and INCOME at positions A1, A2, and A3. Using the technique we saw in chapter three, you would type:

```
>A1 RETURN      (omit if the cursor was already at A1,
                or if you move the cursor using the
                arrow keys.)
RATE RETURN
>
TOTAL RETURN
>
INCOME RETURN
```

When you wish to fill in consecutive entries of a row, or consecutive entries of a column, you can avoid pushing the RETURN

key for each entry. Simply push the cursor arrow key instead. This will copy the entry you just typed into the present cursor location, and then advance the cursor to the next entry if possible. Let's try using this easier technique, but first, let's clear the screen.

Clearing the Screen

We know that turning the computer off will clear our screen. There is a much better way:

```
type /C
see the prompt  CLEAR:  TYPE Y TO CONFIRM
```

If you do type Y, the screen will be blanked out and a fresh worksheet will be displayed. All the previous information you had keyed in will be lost. Clearing the screen has no effect on worksheets that have been saved on diskettes. If you change your mind before typing the Y, you can get back to normal by pushing the CTRL-C keys.

We had the following on the screen:

```
      A        B        C        D
1    RATE    TOTAL    INCOME
2
3
4
5
6
...
9
```

Pushing /CY clears the whole worksheet; in particular, it leaves us with the labels of row one erased. The cursor will always be left in position A1 after you clear the screen with the /CY command. Now we can retype our labels a little faster, by using the following keying:

```
RATE >TOTAL >INCOME>
```

which leaves the cursor at D1, since RATE went into A1, TOTAL went into B1, and INCOME went into C1. You don't have to go from left to right; the following would work just as well:

```
>C1 RETURN INCOME < TOTAL < RATE <
```

This places INCOME in C1, TOTAL in B1, and RATE in A1, as desired. Notice that when RATE is copied into position A1 and you are trying to force the cursor left of A1 by using the < key, nothing bad happens. Position A1 is filled in with the label RATE

and the cursor stays at position A1. You will find that the VisiCalc program is very forgiving in those situations.

Let us now proceed to type in the data from table 1.1. We want the rate 5 and the number 3,000 in row 2. Position the cursor at A2, either by typing >A2 RETURN, or by pushing the cursor arrow keys, and type

5 →

Then try typing

3,000 →

The edit entry line will show you that such a number is unacceptable, by displaying the word ERROR at the cursor position. You simply have to leave the comma out. Reposition the cursor at location B2 and try typing

3000 →

Since this is a nicely-formed number it will be accepted when you push →; it will be recorded in location B2 as the cursor advances to location C2. That takes care of row 2. Using either >A3 RETURN, or

↓ (down to row 3)
← (left to column B)
← (left to column A)

you are ready to type the 7 and the 4000 in row three. Row 4 can have the 8 and 5000 keyed in. Similarly you can type the 10 and 38000 in row 5. Our table now looks like

	A	B	C	D
1	RATE	COPIES	INCOME	
2	5	3000		
3	7	4000		
4	8	5000		
5	10	38000		
6				
7				
8				
9				<C4A>*

*Note: whenever you see this kind of notation at the bottom right of a worksheet, it identifies the worksheet as one which is available on the optional diskette. This particular worksheet has the name C4A, as shown within the <...> braces. So by typing /SL C4A RETURN, you could load this worksheet from the optional diskette if you prefer not to construct it following the detailed steps we have just been through.

We have our titles in place, and the raw data has been entered. Now we want the computer to do some work for us.

Getting the VisiCalc Program to Compute

We would like to get the rate at A2 multiplied by the number of copies at B2, and have the result stored at C2. We can do this by first placing the cursor where we want the result to be placed; here we can type >C2 RETURN, or use the cursor arrow to move the cursor to C2. Then we type in an arithmetic expression depicting the formula we just described. The rate at A2 is represented by the coordinates A2. Similarly, the corresponding number of copies is represented by the coordinates B2. You use an asterisk * to indicate multiplication, and parentheses () to group things. So we can type

```
>C2 RETURN
(A2*B2) RETURN
```

As soon as we do that, the blank entry at C2 is filled in with the number 15000, which is the product of the number 5 from A2 and 3000 from B2. This is not quite what we wanted. You can't expect the VisiCalc program to read your mind; it does not know that you are using percentages for your rates. So we should revise our formula slightly: try

```
(A2*B2).01 RETURN
```

and you will be surprised to see that it is rejected. All multiplications have to be spelled out, using a *. So we try again, typing:

```
(A2*B2)*.01 RETURN
```

and we immediately see the result at C2: 150. We could have typed (A2*B2)/100 and obtained the same correct result; the / means division. You can also use + and - for addition and subtraction. The symbol ^ is used for exponentiation, as in 2^3, for two raised to the power 3. You will find the key labeled "^" sharing the "N" key.

Our table now looks like:

	A	B	C	D
1	RATE	COPIES	INCOME	
2	5	3000	150	
3	7	4000		
4	8	5000		
5	10	38000		
6				
7				
8				
9				

We can now finish specifying the relationships for the items of row 3:

```
>C3 RETURN
(A3*B3)*.01 RETURN
```

Similarly we can specify the formula for C4:

```
>D4 RETURN
(A4*B4)*.01 RETURN
```

and C5 is computed using (A5*B5)*.01. Our table now looks like:

```
        A         B          C          D
1     RATE     COPIES     INCOME
2        5       3000        150
3        7       4000        280
4        8       5000        400
5       10      38000       3800
6
7
8
9                                                    <C4B>
```

This is such an achievement that we want to record it for posterity. How do you get this table printed?

Printing Worksheets

The /P command is used to print worksheets. Before you type it though, verify that your printer has its power switch on, and that its on-line/off-line switch is in the on-line position. Then place the cursor in the A1 position, by typing >A1 RETURN, or by using the cursor arrow keys. Now you can proceed:

```
Type /P
See the prompt:  PRINT: FILE, PRINTER, # (OF SLOT)
```

Select the printer by typing P; we can ignore the other selections at this time. See the new prompt:

```
PRINT: LOWER RIGHT, "SETUP, -, &
```

You now want to position the cursor so that it will be in the bottom right corner of your screen. It does not have to be exactly in the corner. Wherever you place it, it will be taken to define a rectangle. The initial cursor position when you select the /P command defines the top left corner of the rectangle, and the final position of the cursor defines the bottom right corner of the rectangle. If you now push RETURN, the items in that rectangle will be printed. If nothing is printed, either the printer is not set up correctly, or the rectangle you defined was empty.

If we typed the following:

```
>A1 RETURN      (define top-left corner)
/PP             (get the printer service, request
                 printing)
D7 RETURN       (define bottom-right corner)
```

we should get a printout of the dashed portion of the following figure:

```
      A          B          C          D
1 ┌──────────────────────────────────────┐
2 |   RATE     COPIES     INCOME        |
3 |      5       3000        150        |
4 |      7       4000        280        |
5 |      8       5000        400        |
6 |     10      38000       3800        |
7 |                                      |
8 └──────────────────────────────────────┘
9
```

We can now reap the fruit of our efforts by seeing new results computed immediately, merely by typing in either a new rate value, or a new copies value, or both. Since we are also interested in the total income, we should add a new column, entitled TOTAL. Type

```
>D1 RETURN TOTAL ↓  (leaving the cursor at D2)
```

We want location D2 to be computed using the total income to date; this is simply the income figure found at location C2. So try typing

```
C2 RETURN
```

While you were typing C2, the VisiCalc program saw that the entry began with a letter, so it must be a label. Therefore you now see the characters "C2" in location D2, which is not quite what we wanted. We have to make the VisiCalc program believe that we mean C2 to be used as an entry coordinate in a formula. We could do this by typing

```
(C2) RETURN
```

We can also get the same effect by typing

```
+C2 RETURN
```

Our worksheet now has the following appearance:

	A	B	C	D
1	RATE	COPIES	INCOME	TOTAL
2	5	3000	150	150
3	7	4000	280	
4	8	5000	400	
5	10	38000	3800	
6				
7				
8				
9				

The formula for the next entry in the TOTAL's column, D3, is the sum of the current income C3, plus the previous total D2. So we can type

```
>D3 RETURN      (to place the cursor at D3, if necessary)
+C3+D2 ↓        (formula for C3+D2 at D3, then advance to D4)
```

As we are typing each formula, we see its result as soon as we push either the RETURN key or the cursor arrow key, which cause the formula to be both recorded and evaluated. You can examine a previous formula by moving the cursor to its location. The formula will be displayed at the top left of the worksheet, above the prompt line.

Proceeding in this fashion, we come to have our table looking as follows:

	A	B	C	D
1	RATE	COPIES	INCOME	TOTAL
2	5	3000	150	150
3	7	4000	280	430
4	8	5000	400	830
5	10	38000	3800	4630
6				
7				
8				
9				<C4C>

We can now proceed to change any of the rate or copies entries, and the consequences will be immediately computed and displayed. You should not of course try to change the entries in the income or total's column, unless you want to change the formulas.

There Must Be a Better Way--Replication

You don't mind typing in some formulas, if there are only a few involved. But even then, it's very easy to make a mistake while typing, and you might not notice it until it is too late. Some mistakes will be detected by the VisiCalc program. Others will not. If we had typed in the formula +C3+D2 for location D4, instead of +C4+D3, the computer would have no way of knowing that we had made a mistake.

So we have two needs: reducing unnecessary typing, and reducing the chance of errors. The replicate command addresses itself to both of these needs.

When we were defining the INCOME formulas, we had in mind:

```
put (A2*B2)*.01 in C2
put (A3*B3)*.01 in C3
put (A4*B4)*.01 in C4
put (A5*B5)*.01 in C5
```

You would imagine there must be some way of saying: "My formula is (Ay*By)*.01 and I want it copied into locations C2 through C5, replacing the "y" each time with the corresponding row number." This is almost, but not quite how it is done in the VisiCalc program.

First you type in the formula, which applies to some row or column, and place it at the desired location. So here you would type

```
>C2 RETURN
(A2*B2)*.01 RETURN
```

Now you type the request for the replicate command:

```
type /R
see the prompt REPLICATE: SOURCE RANGE OR RETURN
```

You want to replicate one thing, the formula located at C2. Since the cursor is at C2, you need only push RETURN. If the cursor were elsewhere, you could type C2.C2. If you did, you would see your single period replaced by three periods:

```
type C2.C2 RETURN
see C2...C2
```

Either of these will specify the desired SOURCE RANGE. In either case, you see a new prompt:

```
REPLICATE: TARGET RANGE
```

You now specify which locations you want the formula to be copied into; these are the TARGET RANGE. Here we would like it copied

into locations C3, C4, and C5. The easiest way of specifying this request is to respond by typing C3.C5:

```
type C3.C5 RETURN
see C3...C5
```

We now see a new prompt:

```
REPLICATE: N= NO CHANGE, R= RELATIVE
```

If we respond with an N, then the formula that was copied, (A2*B2)*.01, will be used as-is; it would be duplicated exactly as shown in this sentence. That is not what we want here. When the formula at position C2 is replicated (copied) into position C3, we would like the row numbers "2" in the formula (A2*B2)*.01 replaced by the row number 3. For the formula's copy in row 4, we would like the row numbers "2" in the formula (A2*B2)*.01 replaced by the row number 4, and so on. This prompt will step through the formula, highlighting each coordinate which appears in it. We have to respond each time with an N or an R. In our case, we will first see the A2 highlighted, and we should respond with an R. Then the highlight will appear on the B2, and once again we respond with an R. Since there are no other coordinates in this formula, the prompt line disappears, and lo and behold, all of the TOTALs for rows 3, 4, and 5 appear. If we move the cursor over location D5, we see the formula (A5*B5)*.01 displayed above the worksheet, at the top left of the screen.

Replication is a very powerful tool. You can use it for all kinds of things. We will be looking at other uses of it later. For the present, how could we have used it to also generate the formulas for the TOTAL income? Recall that these formulas and their intended homes were:

```
put C2 in D2
put C3+D2 in D3
put C4+D3 in D4
put C5+D4 in D5
```

It seems we can't use the formula from D2 as the target for replication. Maybe we can change it slightly. In a user-friendly system such as this one, why not have an understanding that a reference to an entry which contains a label will be taken to have the value zero. That being the case, the formula for D2 can be rewritten as:

```
put C2+D1 in D2
```

We now have a nice pattern:

```
put C2+D1 in D2
put C3+D2 in D3
put C4+D3 in D4
put C5+D4 in D5
```

So if we place the formula +C2+D1 in D2, by typing:

```
>D2 RETURN
+C2+D1 RETURN
```

then we can replicate it into locations D3, D4, and D5, by typing:

```
/R              (invoke the replicate command)
RETURN          (we are at the source location D2)
D3.D5 RETURN     (specify D3...D5, where we want the formulas)
R               (want C2 to become C3, etc.)
R               (want D1 to become D2, etc.)
```

The R response (for relative) ensures that the appropriate row numbers will be substituted as the formula is replicated.

What we have seen here involves replicating from one row to other rows. We can also replicate from one column to other columns. Suppose for some reason known only to you, you needed a column full of zeroes. You want all of column A from row 1 down to row 25 filled with zero values. Instead of typing 25 numbers and pushing as many arrow keys, you could proceed as follows:

```
>A1 RETURN
0   RETURN
/R                (replicate location A1)
RETURN            (use A1 as source)
A2.A25 RETURN     (use A2 to A25 as target)
                  (now have A1 to A25 filled with 0's)
```

There was no request made for the N (NO CHANGE) and R (RELATIVE) information because the item being replicated was just a numeric constant, so it is automatically not changed.

Estimating the Value of an IRA

Recent changes in the tax laws now permit any wage earner to participate in a tax-sheltered Individual Retirement Account (IRA). A wage earner can contribute a maximum of $2,000 per year to such an account; a married wage earner whose spouse is not a wage earner can contribute up to $2,250 per year. These accounts are attractive in that the contributions may be deducted from one's income, thereby deferring taxes on them (and their earnings) until retirement. Each bank, savings-and-loan and other approved institution can provide a different kind of IRA investment, ranging from guaranteed rate certificates to equity funds. For that reason, it is generally not possible to say what the payout on an IRA will be; rates may change from time to time. But can you project what the payout might be, based on some clearly stated assumptions?

We can set up a worksheet as follows. An assumed yearly rate of return and its label "RATE" can be placed in positions A1 and

A2. We can create a "YEARS" column in column A, indicating what the payout would be at the end of the n-th year (note that severe penalties apply if you make an IRA withdrawal before the age of 59.5). Column B starting in row 3 can be used to record the anticipated annual contribution. We should plan to make contributions early in each calendar year, to get the full benefit of interest earnings. The next column, column C, can display the total accumulation earned to date.

Our worksheet begins as follows:

```
        A          B          C
1       RATE        12
2       YEAR     CONTR     VALUE
3         1       2000
4
...
```

We can use the replicate command to fill in the numbers 2, 3, ... for as many years as we wish. Suppose we only set it up here for 20 years:

```
>A4 RETURN
+A3+1                      (formula for 2, 3, ...)
RETURN
/R
RETURN                     (replicate A4's formula)
A5.A22 RETURN              (copy into A5, A6, ..., A22)
R                          (want A3 in formula relative)
```

Assuming that we want to deposit $2000 each year, we can replicate that amount through the remainder of column B as follows:

```
>B3 RETURN                 (use the 2000 in B3)
/R
RETURN                     (copy B3)
B4.B22 RETURN              (copy into B4, B5, ..., B22)
N                          (Copy with No change)
```

All that remains is to provide a formula for the desired accumulation. At an interest rate i, a year's deposit d would be worth d+i*d after one year. If the interest rate i is given as a whole number rather than a fraction (i.e., 12% instead of .12) then we can correct the formula to read d*(1+i/100). The formula for the first year would then be +B3*(1+(B1/100)). We know when looking at a mathematical expression such as 1+i/100 that we should first divide the value of i by 100 before adding the 1. The VisiCalc program is not that sophisticated. If you give it an expression such as 1+B1/100, it will evaluate it from left to right, coming up with 1 plus B1, all divided by 100! If you want it to do otherwise, you have to use parentheses to force the order of evaluation you have in mind. Thus the need for the parentheses in 1+(B1/100).

The formula for the second year would take year 1's results (in C3) and compound that with year 2's contribution (from A4), using the formula +B4+C3*(1+(B1/100)). Subsequent years follow a similar pattern. We can alter the first year formula so that it too fits the mold, giving us the following set of formulas:

```
+B3+C2*(1+(B1/100)) for C3
+B4+C3*(1+(B1/100)) for C4
+B5+C4*(1+(B1/100)) for C5
 ...
```

Now it is a simple matter to type in the formula for C3 and replicate it into C4, C5, ..., C22.

```
>C3 RETURN
+B3+C2*(1+(B1/100)) RETURN
/R
RETURN            (copy C3)
C4.C22 RETURN      (copy into C4, C5, ..., C22)
R                 (want B3 relative)
R                 (want C2 relative)
N                 (use rate in B1 so type N)
```

This leads to the following IRA table (except for some format changes that can be handled later).

```
    A        B        C
1 RATE       12
2 YEAR     CONTR     VALUE
3    1     2000       2240
4    2     2000       4749
5    3     2000       7559
...          ...      ...
22  20     2000     161398                    <C4D>
```

You can now easily evaluate the impact of a new rate, simply by placing it in position B1.

You can make it easier to compare the effect of different rates by having their consequences placed side-by-side. Suppose we use column E to display the values computed using a different rate. We can set up column E by typing:

```
>E1 RETURN
14 ↓              (rate for col. E, then down to E2)
VALUE ↓       (label, then down to E3)
+B3+E2*(1+(E1/100)) RETURN
/R                (replicate)
RETURN             (copy E3)
E4.E22 RETURN      (into E4 to E22)
R                 (want B3 relative)
R                 (want E2 relative)
N                 (keep E1, no change)
```

We now have the following table:

```
      A        B        C         D          E
1  RATE       12                            14
2  YEAR     CONTR     VALUE               VALUE
3     1      2000      2240                2280
4     2      2000      4749                4879
5     3      2000      7559                7842
...           ...       ...                 ...
22   20      2000    161398              207537                <C4E>
```

See how easy it is to make comparisons now. Of course there are many other factors to be taken into account in setting up an IRA; getting a good rate of return is just one of these factors.

Using ":"

There is another key, the colon key ":," which can be used in place of the RETURN key in two situations. You can use the colon ":" instead of the RETURN key either:

(1) to complete a GO TO entry, as in

```
>A12:
```

instead of the usual

```
>A12 RETURN
```

(2) in response to the /R (replicate command) prompts.

In the second case, instead of typing

```
>C2 RETURN
(A2*B2)*.01 RETURN
/R
C2.C2 RETURN
C3.C5 RETURN
```

we can type

```
>C2:
(A2*B2)*.01 RETURN
/R
C2.C2:
C3.C5:
```

You cannot use ":" to complete the entry of labels, expressions or values. You can only use the ":" as a RETURN key replacement in the two cases we just described. The VisiCalc program itself uses the ":" when we ask it to print out the script that created a particular worksheet. We will see how to do this when we discuss the /SS,S1 command.

SUMMARY

Fill in entries and advance the cursor to the next location by using the cursor arrow keys instead of using the RETURN key.

Values begin with digits, or the special characters +, -, (, #, @ and the period.

Labels begin with alphabetic characters.

The Clear-the-Screen command is /C. It provides a prompt and expects a Y response.

Enter formulas as values, by beginning them with a +, or a (, or one of the other special characters.

Print worksheets using the /P command. You should first select the top left corner of the rectangle to be printed. Respond to the prompt with another P, then position the cursor on the bottom right corner of the desired rectangle.

Replication: type /R and respond to several prompts. The first prompt requests the source range: we specified the beginning and end coordinates by pushing RETURN. The second prompt requests the target range; we specified the beginning and end coordinates, separated by a single period. The final prompt steps through the formula being replicated, highlighting successive worksheet coordinates. For each of these, we can respond N for no change, or R for relative. When we want them to be adjusted for the new positions, we respond with R.

The ":" key can be used in place of the RETURN key to complete the entry of a GO TO request (e.g., >B43:) and in responding to /R (replicate) prompts.

Chapter 5

USING FUNCTIONS AND COPING WITH CHANGE

In this chapter, we will introduce the use of some very handy tools known as functions. You may already be familiar with the use of high-powered functions on a calculator. They can save us a lot of time and effort. In order to better appreciate them, and also to consolidate the techniques we saw in chapter 4, we will first work out a new problem with the techniques we have in hand. Then we will redo the problem using the functions we will soon see.

We will also examine how best to cope with problems whose boundaries are not completely predetermined. How should you set things up when you don't know in advance exactly how many rows or columns you may need? This happens all the time and when dealing with a paper worksheet you do a lot of erasing or cutting and pasting. Fortunately, this has been anticipated by the VisiCalc program and we will see how to cope with it. But first let us look at a new problem.

A Teacher's Gradebook

Most gradebooks are handwritten, and they have the following appearance:

Name	Homework1	Homework2	Test1 ...	Total	Average
Alan	95	100	80	850	85
Bernice	85	95	73	820	82
Charles	60	0	50	400	40
...					
Fred	80	60	70	650	65

You might prefer to think of this example in other ways. A farmer can see it as recording dairy production by head and by month. A sale's manager sees it as a monthly sales record for each salesperson. All you have to do is change the names and the titles.

To keep our example down to a manageable size, let us have a class of five students, with the following entries:

NAME	HWK1	HWK2	TST1	TST2
ALAN	95	100	80	85
BERNICE	85	95	73	94
CHARLES	60	0	50	66
FRED	80	60	70	78
MARY	82	80	75	86

If we enter these labels, names, and numbers on the worksheet beginning in the top left corner, we get the following assignment of rows and columns:

	A	B	C	D	E	
1	NAME	HWK1	HWK2	TST1	TST2	
2	ALAN	95	100	80	85	
3	BERNICE	85	95	73	94	
4	CHARLES	60	0	50	66	
5	FRED	80	60	70	78	
6	MARY	82	80	75	86	<C5A>

We can use column F for the desired totals and column G for the average. After typing these labels in the usual way

```
>F1 RETURN
TOTAL > AV>
```

we can specify the computation of totals for row 2 using the formula +B2+C2+D2+E2 and enter that in position F2. Similarly the row 3 total is computed using +B3+C3+D3+E3. Since this pattern applies to all of the totals, we can save some effort by using the replicate command. Begin by typing

```
>F2 RETURN              (position the cursor at F2)
+B2+C2+D2+E2 RETURN     (formula for F2 total)
/R                      (replicate request)
RETURN                  (use formula in F2)
F3.F6 RETURN            (replicate into F3 to F6)
R                       (want B2 relative)
R                       (want C2 relative)
R                       (want D2 relative)
R                       (want E2 relative)
```

This generates the formulas for the totals for rows 2 through 6, in column F, and simultaneously calculates and displays the totals, as in the following figure:

	A	B	C	D	E	F	G	
1	NAME	HWK1	HWK2	TST1	TST2	TOTAL	AV	
2	ALAN	95	100	80	85	360		
3	BERNICE	85	95	73	94	347		
4	CHARLES	60	0	50	66	176		
5	FRED	80	60	70	78	288		
6	MARY	82	80	75	86	323		<C5B>

How can we compute the averages for column G? The average of a set of numbers is defined as their sum divided by the number of items. The average grade for a particular student can then be calculated using the total we just computed, and divide that total by the number of items involved, which in this case is just four.

The formulas we need are:

```
for G2, F2/4
for G3, F3/4
for G4, F4/4
for G5, F5/4
for G6, F6/4
```

Of course, each one of these needs a + in front of it, to qualify as a formula, instead of being used as a label. Once again we can use replication:

```
>G2 RETURN              (position cursor at G2)
+F2/4 RETURN            (enter formula for G2)
/R                      (request replication)
RETURN                  (use the formula at G2)
G3.G6 RETURN            (copy it into G3, G4, ..., G6)
R                       (want F2 relative)
```

Our table now looks like:

	A	B	C	D	E	F	G	
1	NAME	HWK1	HWK2	TST1	TST2	TOTAL	AV	
2	ALAN	95	100	80	85	360	90	
3	BERNICE	85	95	73	94	347	87	
4	CHARLES	60	0	50	66	176	44	
5	FRED	80	60	70	78	288	72	
6	MARY	82	80	75	86	323	81	<C5C>

If you are typing this into a worksheet as you are reading this, the averages the VisiCalc program calculates won't be rounded as shown here. For instance, Bernice's average would be displayed as 86.75. I have rounded them here just to avoid cluttering up these examples. We will see how we can get the VisiCalc program to do this for us later (if you can't wait to see, look up /GFI in the index, or read chapter 9).

Before redoing this problem using some new ideas (e.g., functions) it is worth stating how we could make more of the worksheet visible so it would be easier to deal with. The normal column width the VisiCalc program uses is nine characters wide. With a 40-character wide screen, that width only lets you see four columns at a time (e.g., A through D, or B through E, etc.). Here we are dealing with relatively small numbers, so we can request that narrower columns be used. The command /GC lets you change the column width. If you want columns that are seven characters wide, type /GC7 RETURN; that will let you see columns A through E. Typing /GC6 will let you see columns A through F.

We are ready now to redo this problem, using functions.

Using Functions

Most of us have used functions of one kind or another, perhaps in high school or college. They have many different appearances. The square root or radical sign $\sqrt{}$ is a function. So are the trigonometric functions with English-like names such as sine and cosine. Some things we don't usually think of as functions can be very useful if we modify our point of view.

In the VisiCalc program, all functions have names beginning with the at-sign symbol @. The two functions which we will use here are:

```
@SUM      compute the sum or total
@AVERAGE  compute the average
```

When you use a function, you have to specify what the function is to compute with. The items it computes with are called its arguments. Functions such as square root and sine have only one argument. The @SUM and @AVERAGE can have many arguments and these are presented as an argument list.

You can use @SUM by providing it with a list of one or more arguments, as in:

```
@SUM(C2)
```

In this case, @SUM has only one argument, the coordinate C2, and the result will be the same as if we had typed +C2 as a formula. If you type

```
@SUM(C2.C7) RETURN
```

then you will see the echo

```
@SUM(C2...C7)
```

This will compute +C2+C3+C4+C5+C6+C7. You can if you wish include other items in the list.

```
@SUM(D5.D9,100,A10.K10)
```

will add the items from rows 5 through 9 of column D, the number 100, then the items from columns A through K of row 10. The argument list can include expressions such as 2*(E1-D1), etc. The use of @AVERAGE is similar to the use of @SUM:

```
@AVERAGE(B3.B15)
```

will compute the average value of the items in rows 3 through 15 of column B, but this average will only include the non-blank numeric values. In the case of a teacher's gradebook, this means that the student who missed a test due to illness and presumably received a blank grade for it won't be penalized; all blank grades will be ignored.

Now let us redo the gradebook, using @SUM and @AVERAGE. To get totals in rows F2 through F6, type:

```
>F2 RETURN                (position cursor at F2)
@SUM(B2.E2) RETURN
/R
RETURN
F3.F6 RETURN              (target range)
```

```
R                              (want B2 relative)
R                              (want E2 relative)
```

This gives us the table:

```
     A          B        C        D        E      F    G
1  NAME       HWK1     HWK2     TST1     TST2  TOTAL  AV
2  ALAN         95      100       80       85    360
3  BERNICE      85       95       73       94    347
4  CHARLES      60        0       50       66    176
5  FRED         80       60       70       78    288
6  MARY         82       80       75       86    323      <C5D>
```

Similarly we can get the averages for rows G2 through G6 by typing:

```
>G2 RETURN
@AVERAGE(B2.E2) RETURN
/R
RETURN
G3.G6 RETURN
R
R
```

which leaves us with the following table:

```
     A          B        C        D        E      F    G
1  NAME       HWK1     HWK2     TST1     TST2  TOTAL  AV
2  ALAN         95      100       80       85    360   90
3  BERNICE      85       95       73       94    347   87
4  CHARLES      60        0       50       66    176   44
5  FRED         80       60       70       78    288   72
6  MARY         82       80       75       86    323   81   <C5E>
```

It would be very nice if, as the grades for the latest homework or test are entered, the gradebook would compute the class average for that homework or test. All we have to provide is a formula for a new bottom row, which we will label AV, for average. We can provide the label, and the formula, by typing:

```
>A7 RETURN
AV →                           (put label AV at A7)
@AVERAGE(B2.B6)                (formula for B7)
RETURN                         (place it in B7)
/R                             (begin replicating it)
RETURN
C7.G7 RETURN                   (target is C7, D7, ..., G7)
R                              (want B2 relative)
R                              (want B6 relative)
```

Our table now looks like:

	A	B	C	D	E	F	G	
1	NAME	HWK1	HWK2	TST1	TST2	TOTAL	AV	
2	ALAN	95	100	80	85	360	90	
3	BERNICE	85	95	73	94	347	87	
4	CHARLES	60	0	50	66	176	44	
5	FRED	80	60	70	78	288	72	
6	MARY	82	80	5	86	323	81	
7	AV	80	67	70	82	299	75	<C5F>

We will examine some of the other useful functions provided by the VisiCalc program later. We will now consider the next major topic of this chapter.

Coping with Change

What if part way through the school year, a new student joins the class? Does the teacher have to rewrite all the gradebook formulas? Similarly, the teacher may not know exactly how many assignments or tests will be given during the school year. So how can things be set up so that changing the number of rows or columns can be done with a minimum of effort?

Deleting Rows or Columns

Having convinced ourselves that the formulas we are using to compute the average grades are correct, we might decide to simplify our gradebook, by erasing the column headed TOTAL. How can we erase or delete an entire column (or row)? Typing

```
/D
```

produces the prompt

```
DELETE: R  C
```

You can respond with an R (to delete a row) or with a C (to delete a column) or push CTRL-C if you don't want to delete anything. What will be deleted? Whichever row or column the cursor was placed on when you invoked the delete command will disappear as soon as you type an R or a C.

So if we placed the cursor anywhere within column F, say at F1:

```
>F1 RETURN
/DC        (delete column)
```

then column F will disappear and all columns to its right will move over to fill in the empty space, leaving us with the table as it appears next.

The deletion of a row has a similar effect. Rows below the deleted row will move up to close up the gap. In all cases

(deletion of a row or column) formulas which used to refer to rows (or columns) which are moved due to a row (or column) deletion will be adjusted so the results remain correct.

Inserting Rows or Columns

Suppose we have a version of our gradebook in which the TOTAL column has been deleted, since we don't really need it now. Furthermore, suppose the second homework, HWK2, had just been graded and recorded:

```
    A        B       C        D      E     F
1 NAME     HWK1    HWK2     TST1   TST2   AV
2 ALAN       95     100                   98
3 BERNICE    85      95                   90
4 CHARLES    60       0                   30
5 FRED       80      60                   70
6 MARY       82      80                   81
7 AV         80      67                   74   <C5G>
```

A new student, Doris, has just joined the class. Suppose her name is to be entered between those of Charles and Fred. We have

```
row 4  CHARLES ...
row 5  FRED    ...
```

We want

```
row 4 CHARLES ...
row 5 DORIS    ...
row 6 FRED    ...
```

We can insert a row by using the /I (insert) command. Recall that beginning typing a new entry with a / leads to the prompt line:

```
COMMAND: BCDEFGIMPRSTVW-
```

Selecting the I command requests insertion, and the I command will prompt you:

```
INSERT: R  C
```

Here we will type R for row insertion. But how does the VisiCalc program know where to place the new row or column? If you wish to insert a row in front of row 5, place the cursor on row 5 before you use the /IR command. Similarly, if you wanted a new column to the left of an existing column, place the cursor on the existing column before you type /IC. If you find you made a mistake, you can delete the new row or the new column by typing

```
/DR    delete row the cursor is on
```

or

```
/DC    delete the column the cursor is on
```

In our gradebook, if we place the cursor on row 5 and type /IR, we will see a blank line appear in row 5 as the older rows 5, 6, etc. are pushed down and renumbered 6, 7, etc. Notice that the averages have not been affected.

	A	B	C	D	E	F	
1	NAME	HWK1	HWK2	TST1	TST2	AV	
2	ALAN	95	100			98	
3	BERNICE	85	95			90	
4	CHARLES	60	0			30	
5							
6	FRED	80	60			70	
7	MARY	82	80			81	
8	AV	80	67			74	<C5H>

We can now type in the new student's name, Doris, at A5. If we position the cursor at B8 we can examine the formula saved to calculate the average for column B. It appears on the prompt line as:

@AVERAGE(B2...B7)

But if you look back a few pages, didn't we enter it as

@AVERAGE(B2...B6)

Of course, we only typed one period; the VisiCalc program converted it into three periods. How did the B6 suddenly change into a B7? If we look at the formulas for C8, D8, etc., those too have been changed. What is going on?

In the process of inserting a row, if the new row falls into the range of some formula, that formula already includes the new row, and it will also continue to include the old rows it used to cover. That inclusion encompasses those rows which had to be assigned new row numbers due to the row insertion. So Fred and Mary, from old rows 5 and 6, are still included in the computation for AV, using their new row numbers 6 and 7. The electronic worksheet provided by the VisiCalc program behaves just as a paper worksheet should, if you squeezed in a new row.

A word of caution is warranted here. If you were using the formula +A1+B1+C1+D1 to calculate a result and you subsequently inserted a column between column B and column C, the formula +A1+B1+C1+D1 would automatically be transformed into +A1+B1+D1+E1 because the insertion forced old column D to be renamed column E and similarly old column C becomes column D. So the formula remains consistent with the changes that took place. Here is the important point: if the formula had been @SUM(A1.D1) before the insertion of a column between columns B and C, then the formula would automatically be updated to read @SUM(A1.E1) which <u>includes</u> the new column. The previous formula +A1+B1+C1+D1 was transformed into a new formula which did not include the new

column. That is one of the important reasons for using functions, because they let you specify ranges and ranges automatically include all intermediate locations as the worksheet is widened by insertion of new columns or as it is lengthened by insertion of new rows.

Notice that since @AVERAGE ignores blank entries, Doris's blank grades for HWK1 and HWK2 have no effect on the averages for the rest of the class.

Caution: you may wish to circle this paragraph, because it makes an important point about the consequences of an insertion. You must be inserting a row into (INTO) the range of a formula, not merely adding a row to one end or the other of a formula's range, if you want the formula to be automatically adjusted to include the new row. So if we had inserted Doris's entry in front of old row 7, then the formula would not have been adjusted. We will reconsider this situation in a few moments.

Adding a Column

Suppose we had our gradebook as follows:

	A	B	C	D	E	F	
1	NAME	HWK1	HWK2	TST1	TST2	AV	
2	ALAN	95	100			98	
3	BERNICE	85	95			90	
4	CHARLES	60	0			30	
5	DORIS						
6	FRED	80	60			70	
7	MARY	82	80			81	
8	AV	80	67			74	<C5I>

We have finished grading homework number three, and wish to enter it as HWK3, between HWK2 in column C and TST1 in column D. We begin by placing the cursor on column D; any row in column D will do. That ensures that the new column will be inserted immediately "in front of" column D, which is to say, immediately to its left. We should now type

```
/IC     (insert a column)
```

Our new table has the following appearance:

	A	B	C	D	E	F	G	
1	NAME	HWK1	HWK2		TST1	TST2	AV	
2	ALAN	95	100				98	
3	BERNICE	85	95				90	
4	CHARLES	60	0				30	
5	DORIS							
6	FRED	80	60				70	
7	MARY	82	80				81	
8	AV	80	67				74	<C5J>

Having graded homework number 3, we can enter its title HWK3 and the grades 95, 80, 54, 88, 60, and 90 in column D, by typing:

```
>D1 RETURN     (cursor at D1)
HWK3 ↓   (enter the label HWK3 and move cursor down)
95 ↓     (enter the grade 95, move the cursor down)
80 ↓     (enter the grade 80, move the cursor down)
54 ↓     (enter the grade 54, move the cursor down)
88 ↓     (enter the grade 88, move the cursor down)
64 ↓     (enter the grade 60, move the cursor down)
90 ↓     (enter the grade 90, move the cursor down)
```

As we type in the new grades, the row averages for all of the students are updated, except for Doris. We have to provide a formula for this new row's average, if we want it to be computed. Similarly, having entered the new column D, for homework number 3, we have to provide the formula for that column's average, if we want it to be computed. In both cases, we can either type in the desired formula, or position the cursor on the corresponding formula for an older row or column, and replicate it. Using this latter approach, for instance, we could copy the formula from C8 to D8 by typing

```
>C8      RETURN  (position over old formula)
/R               (replicate it)
RETURN           (use the formula at C8)
→                (move to D8)
RETURN
R                (relative)
R                (Relative)
```

You could of course simply have typed

```
>D8 RETURN
@AVERAGE(D2.D7) RETURN
```

Using either approach, you can also provide the formula for the average, for G5. The result in either case is:

	A	B	C	D	E	F	G
1	NAME	HWK1	HWK2	HWK3	TST1	TST2	AV
2	ALAN	95	100	95			97
3	BERNICE	85	95	80			87
4	CHARLES	60	0	54			38
5	DORIS			88			88
6	FRED	80	60	64			68
7	MARY	82	80	90			84
8	AV	80	67	79			77

<C5K>

Planning for Change

If you were setting up a new gradebook, you could anticipate having to add new students or record more tests or assignments than originally anticipated, either by setting up dummy entries, or entries which will always retain their relative positions.

The first approach goes as follows. For student names, you could have two dummy entries, one called "first," the other called "last." Then you could insert students as needed, in any row between these two, and be confident all column formulas will be adjusted accordingly. Similarly, you can safely designate a column to record the grades for a final exam, even if you never use it. Any extra tests or assignments would be recorded to its left. Your initial gradebook could look like:

```
    A      B      C      D      E     F
1 NAME    HWK1   HWK2   TST1   FINAL  AV
2 FIRST
3
4
5
6
7
8 LAST
9 AV
```

The formula for the row averages can be defined for each of rows 2 through 8, using replication. Similarly, the formula for the column averages can be defined for each of columns B through G, using replication. For your convenience, they are repeated here:

```
>B9 RETURN                        (move cursor to B9)
@AVERAGE(B2.B8) RETURN
/R
RETURN
C9.G9 RETURN                    (target range)
R
R
```

That takes care of the column averages. The row averages are generated by typing:

```
>G2  RETURN                  (cursor to G2)
@AVERAGE(B2.B8) RETURN
/R
RETURN
G3.G8 RETURN
R
R
```

The second approach is a refinement of the first, based upon the observation that blank entries are ignored, and entries containing labels are also ignored if they are used to compute a result. That being the case, we can eliminate the need for a row called "FIRST" by extending the column average computation to also include the labels of row 1. Likewise, we can simply use a blank line instead of the one labeled "LAST."

If you do this, then type in the formulas for the row and column averages, as follows:

```
>B9 RETURN
@AVERAGE(B1.B8) RETURN
/R
RETURN
C9.F9 RETURN
R
R
```

Then all will be well, except that positions D9 and E9 will contain the word ERROR. Leave it alone. Don't try changing it. If and when you provide numeric data in columns D or E, then the correct averages will be displayed. Until then, this simply reminds you that no data has been provided in those columns. It's a consequence of the attempt to calculate the average value of zero items; the sum is zero, but then you have to divide by the number of items, which is also zero. Division by zero is not defined, and the VisiCalc program reminds you of this by displaying the result "ERROR."

What if you make an error when typing in a value? The next section provides some relief.

Erasing Your Errors

If you type in an incorrect value, or simply change your mind, you can replace any value (or label) by positioning the cursor over the desired item, and typing in the new value. Unfortunately, that does not let you erase anything. You might think: "I can simply replace the old value by the number 0." That won't usually be satisfactory, for two reasons: (1) a zero will be displayed and/or printed, and it may look awful (think of the times you were not paying attention, and typed something into location A12 and you could not get rid of it) and (2) a zero counts when the @AVERAGE function is being used whereas a blank entry does not count--that could change your results. How then do you erase something?

You could try /DC; that would delete the entire column the cursor was sitting on. Similarly, if you try /DR, that will delete the entire row the cursor was sitting on. These actions may be a little drastic. If you only want to erase (blank out) one item, place the cursor on it and type

/B RETURN

The /B command erases the information at the location the cursor is sitting on if you push RETURN or any of the cursor keys. If you change your mind and wish not to erase the item you have selected, then push the CTRL-C keys.

Correcting Errors: /E Edit Command

The response to a "/" is:

COMMAND: BCDEFGIMPRSTUVW-

Pushing the E key activates the edit command. Without this command, if you wish to change a label, a value, or a formula, you have to retype the whole thing, even if the new version differs from the old in just one or two characters. Using the edit command makes it possible to retrieve a label, a value, or a formula, change any desired part or parts, and replace the old version with the new one. Let us look at a simple example.

Suppose you had earlier typed the value 12345 in location B2, and you now want to change it to 912345. You can proceed as follows:

```
type B2 RETURN
     /E                    (invoke the edit command)
see [EDIT]: VALUE          (on the prompt line)
see <1>2345                (on the edit entry line
                           just below the prompt line)
```

Having placed the cursor at B2 (using the cursor arrows, or a >B2), when you type /E the item at the current location, B2, is copied onto the edit entry line, just above the worksheet, on the left. At the same time, the prompt line which is just above the edit entry line reminds you that you are using the edit command and it identifies the item copied from B2 as either a value or a label. In this case, it shows the word VALUE. Then a miniature blinking edit cursor (much smaller than the usual VisiCalc cursor) is placed over the left-most character of the item just copied onto the edit entry line. Here in the text, we will use the pair of symbols < and > to embrace the character selected by the edit cursor (on the 1, by virtue of <1>).

Given that the edit cursor is over the left-most character, and we wish to insert a "9" just to the left of the left-most character, all we have to do is type the desired digit key. The digit keyed in will automatically be inserted to the left of the cursor position. So with

```
                <1>2345        (edit cursor at the "1")
      typing    9
produces        9<1>2345
```

Note that the edit cursor did not move in this operation. If you

typed more digits, say a 7, then a 0, the edit entry line would change as follows:

```
9<1>2345
97<1>2345
970<1>2345
```

When you are done, pushing RETURN will copy what is on the edit entry line back to the worksheet location that you had selected prior to invoking the edit command. If in this example you typed anything but a digit or some other acceptable arithmetic operator (such as a +, or @function-name, etc.), when you finally push the RETURN key to indicate you have finished changing this item, you will hear a beep and remain in the edit mode. If you start with a value (in this case it was a number) and you typed one or more characters that are not acceptable in a value, edit will not let you change values into labels, or vice-versa. If you wish to do this you can't use /E; you have to type the desired item completely. If you have such a problem, either stop the edit operation by pushing the CTRL-C keys or use the ESC key to remove the objectionable characters.

What if you had wanted to change the value 12345 into 2345? Immediately after typing /E, you would see

```
<1>2345
```

Edit lets you delete one character at a time. Pushing the ESC key erases the character immediately to the left of the edit cursor. We want to erase the 1 in 12345 to get the desired result 2345 so somehow we have to move the cursor over one character to the right.

As soon as you type /E, the standard cursor arrow keys take on new roles:

→	move the edit cursor right one char.
←	move the edit cursor left one char.

So pushing → once changes the display on the edit entry line from

```
<1>2345          to      1<2>345
```

With the edit cursor immediately to the right of the 1 we want to erase, all we have to do is push ESC once, leaving the edit entry line showing:

```
<2>345
```

Note once again that this latest operation did not move the edit cursor. To record the new value, all we need do now is push RETURN.

You can make as many insertions and deletions as you wish with any combination of edit cursor arrow moves as you desire in any single invocation of /E. Let us look at one more example.

Suppose you had earlier placed the four characters "JAN." in location C1. You now want to spell it out as the word JANUARY. Proceed as follows:

>C1 RETURN	(select worksheet location)
/E	(invoke edit)
	(edit entry line shows: <J>AN.
→	
	J<A>N.
→	
	JA<N>.
→	
	JAN<.>
→	
	JAN.< >
ESC	
	JAN< >
U	
	JANU< >
A	
	JANUA< >
R	
	JANUAR< >
Y	
	JANUARY< >
RETURN	

The rules for use of edit could not be simpler:

(1) use → and ← to position the edit cursor
(2) pushing ESC erases the character to the left of the cursor, without moving the cursor.
(3) pushing RETURN terminates the edit process, copying the item from the edit entry line back into a worksheet location, provided the nature of the original item has not been changed (i.e., if you started with a value, you must finish with a value, and if you start with a label it will remain a label)
(4) pushing any other symbolic key (letters, digits, arithmetic operators, etc.) inserts the corresponding character to the left of the edit cursor, without moving the cursor.
(5) edit cannot be used to transform a value into a label, or vice-versa; editing a value must lead to an acceptable value.

One more example should suffice. Suppose you had earlier typed a formula in position E1, +B1*C1+1/D1, and later realized

that you intended the 1 to be added to C1 before the multiplication by B1 occurs. So you have to insert parentheses which transforms the formula +B1*C1+1/D1 into +B1*(C1+1)/D1. Instead of retyping the formula (what if it were part of a very long formula!), let us edit it. Type

```
>E1 RETURN
/E
```

and see <+>B1*C1+1/D1 on the edit entry line. Move the cursor from the left-most +, so it sits on the C of the C1, by pushing → four times. Then, to insert a "(" hold the SHIFT key down while typing an 8. This does not move the cursor: our edit entry line should now be showing +B1*(<C>1+1/D1. We still need to place the matching ")" between the "1" and the "/" of the pair 1/, so we must move the cursor right four places, to sit on the /. Typing → four times does this, and holding the SHIFT down while typing a 9 inserts the missing ")" leaving us with +B1*(C1+1)</>D1. Since this exactly what we want, pushing RETURN will record this back into location E1.

Before leaving this topic we should see how you can edit something that you were in the process of typing for the first time. The editing we have finished discussing lets you retrieve a value or a formula or a label that has already been typed and placed on the worksheet. What if you were typing a new entry and noticed an error in it before pushing the RETURN key? You could of course erase as many characters as necessary using the ESC key and correct the entry. You can enter the edit mode instead, by typing CTRL-E (push the CTRL and E keys simultaneously). Then you can use the → and ← arrow keys as we did earlier to place the edit cursor as desired. Once you enter the edit mode using CTRL-E you proceed just as with any other edit operation. You will either complete it by pushing RETURN to record your finished product on the worksheet or you will cancel the operation with the CTRL-C keys.

Do not hesitate to use the edit command from now on; it should become second nature for you.

The Move Command /M

What if you had just inserted a row for a new student named Ronald. After having typed in his name and recent grades, you notice that you inserted the line at the wrong place! You could of course delete the row, then insert a new row at the correct place, then retype the information you just deleted. Somehow computers are supposed to simplify work, not complicate it--there must be a better way.

You could try moving the line to the desired position, with the help of the /M command. It prompts you very simply:

```
MOVE:  FROM ... TO
```

Before typing /M, you should place the cursor on the row you wish to move. Then, after typing /M, respond to its prompt by typing a period and moving the cursor to the row immediately below the new row position you want, and push RETURN, if you are moving a row down. So to move Ronald's entry from, say row 5, to row 7, type:

```
>A5 RETURN        (source row)
/M                (move command)
.                 (respond with a ".")
A8 RETURN     (specify row 8, to move down into row 7)
```

You could also have pushed the cursor down arrow ↓ three times (picking out row 8) instead of typing in the coordinate A8.

In moving a row up, you must select the desired target row itself (not the one below it). You can do this either by typing in its coordinate or by moving the cursor over it. Note the difference: in moving down, you select one row further down than desired. In moving up, you select the specific desired row. In either case, the coordinates (or the cursor) must remain in the same column.

Similar considerations apply to moving a column to the right (similar to "down"), and moving a column to the left (similar to "up").

SUMMARY

We have reviewed the simple uses of replication, /R.

We have introduced the use of two functions:

```
@SUM(argument list)
@AVERAGE(argument list)
```

Deletion of a row is requested using /DR; the row the cursor was sitting on will be deleted, and any formula the row was included in will be adjusted to compensate for the row's deletion.

Deletion of a column is requested using /DC, and the effect on formulas is similar to that when deleting rows.

Insertion of rows is performed by positioning the cursor on an existing row and typing /IR to create a blank row in front of (above) the older row.

Insertion of columns uses /IC and the new column will be placed left of the column the cursor was on.

Insertion causes ranges in formulas to be adjusted to include the new rows or columns, provided they were inserted within the range of a formula; specifically, formulas will not be adjusted if

you insert a row or column in front of the first row or column of a formula's range. Similarly, inserting a row or column after the last row or column of a formula's range will leave the formula unchanged.

Erasing (blanking out) an entry is done by positioning the cursor on it and typing /B.

Editing an entry instead of retyping the entire entry is possible by using the /E command. Edit allows insertion and deletions one character at a time. A new entry which is still on the edit entry line can be edited with all of the edit command facility by typing CTRL-E to enter the edit mode.

Moving a row (or a column) is accomplished by using the /M command.

Chapter 6

WHAT IF IT WON'T FIT ON THE SCREEN?

In this chapter we will examine those features of the VisiCalc program that permit you to more easily manipulate much larger worksheets than you can display on your screen. How you can do this and how you can print out worksheets that require wider paper than your printer can handle will provide the substance of this chapter.

It bears repeating that the VisiCalc program makes it possible for smaller computers such as you are using to do things that are very conspicuous by their absence on much larger computers. The fact is that few large computers provide the features available from the VisiCalc program in such an easy-to-use fashion. This is particularly applicable to what we are about to discuss.

As you recall, a worksheet can extend from column A to column BK (that provides for 63 columns) and from rows 1 to 254. Suppose you filled out a worksheet with a row of eleven labels, as you would obtain by typing:

```
>A1 RETURN
> NAME > H1 > H2 > T1 > T2 > ...  T8 RETURN
```

At the point where the label T2 has been typed, the label NAME disappears from view, leaving your screen as follows:

```
     B      C      D      E
1    H1     H2     T1     T2                              <C6A>
```

Similarly, if you filled in the next 21 rows with names and grades, the titles in row 1 would disappear from your screen as you began entering the items for row 21. Suppose we had the following names and grades, some of which we saw earlier:

NAME	H1	H2	T1	T2	T3	T4	T5	T6	T7	T8	
ALAN		95	100	88	55	88	85	79	82	76	83
BERNICE		85	95	72	88	66	74	80	74	69	78
CHARLES		60	0	84	92	89	78	70	83	84	71
RUTH		66	76	75	77	78	79	73	78	77	75
MARY		82	80	76	66	76	83	78	80	89	79
FRED		80	60	52	98	83	68	79	83	55	73
SARA		98	66	88	69	45	88	74	51	66	72
PHIL		88	71	49	94	77	59	88	77	50	73
PATRICK		90	87	86	78	75	81	78	90	94	84
SUE		66	68	72	74	77	83	90	92	95	80
MARIA		75	78	67	72	77	80	82	78	85	77
JIM		66	69	64	72	77	70	74	78	78	72
JACK		44	49	70	65	68	72	76	72	78	66
NICOLE		66	68	78	62	70	80	78	88	84	75
ROGER		69	73	80	71	76	90	81	84	88	79
LARRY		88	81	78	73	77	78	80	79	84	80
ADAM		56	67	87	72	66	45	69	78	88	70
HELENE		66	78	88	89	90	88	90	88	94	86
RONALD		66	68	87	72	66	78	83	71	77	74
FRANCINE		66	78	88	92	88	86	80	85	88	83
EDDY		50	52	45	67	76	62	56	55	58	58
AVERAGE		72	70	75	76	75	77	78	78	79	76

<C6B>

How can we easily work with worksheets that do not fit on the screen? Before you get upset and start blaming yourself for having purchased a microcomputer instead of buying a bigger computer, the state of affairs with a larger computer is not much different. Your screen size is smaller (in terms of the numbers of characters per line) than is typical of the screen size used on larger computers. However, your VisiCalc program makes your very small computer much more useful than is the case with many of the fancy terminals connected to very large computers. One advantage of working with fewer characters per line is that the character sizes are much larger and this makes them easier to read. A word of caution is warranted here: you may be aware of hardware enhancements for the Apple II computer which make it possible to see 80 characters per line (provided you use a monitor display--you cannot use a home television set). Adding such hardware will not help you if the software does not know about it. As this book was going to press, the VisiCalc program for the Apple II had not yet been adapted to use the 80-column hardware.

If you had a very large screen, you probably would make more errors. Having a reasonably sized screen lets you concentrate on the few things the human mind can handle well at one time.

Using Narrower Columns

If you are dealing with small numbers then a simple way of making better use of the limited capacity of your screen presents itself. As mentioned in a previous chapter the default width of the worksheet columns is nine characters per column. You can change this at any time without losing any information by typing

/GCn (global column width changed to value n)

where n is a number between 3 and 37. The appearance of numbers or labels seen on the screen may change but they retain the forms you originally recorded and any calculations will use the values you recorded regardless of the column width selected. We will return to this topic when we examine all of the global commands invoked using /G later.

Printing Long Worksheets

If a worksheet won't fit on the screen because it is too long it still can be easily printed in one operation. Suppose we had a roster with 30 rows, with the names in column A, and the badge numbers in column B:

```
                A           B
        1       NAME        BADGE NUMBER
        2       SAM           321
        3       JAMES         700
      ...       ...           ...
        30      CLOUSEAU      123                  <C6C>
```

To print this roster, place the cursor at A1, by typing the following:

```
>A1 RETURN
/PP
B30                      (define bottom right corner)
RETURN
```

You could have typed C30 instead of B30 to define the bottom right corner of the rectangle which has the initial cursor position (here, it was A1) as its top left corner. The content of that rectangle will be printed, if it fits on the printer paper. If you wanted just the names printed, without titles or badge numbers, you could type:

```
>A2 RETURN
/PP
A30
RETURN
```

This will cause column A to be printed, starting at row 2, continuing up to row 30.

Printing Wide Worksheets

If your worksheet is too wide to fit on the screen, as was the case with the first example of this chapter (it had 11 wide columns), you may still be able to print it using a single command; it depends on the printer you are using, and the paper it is printing on (you might be using 8 inch wide paper on a printer capable of using 15 inch wide paper).

Suppose your printer only accommodates 80-character lines. Our example can be printed in two segments, as shown below:

NAME	H1	H2	T1	T2	T3	
ALAN		95	100	88	55	88
BERNICE		85	95	72	88	66
CHARLES		60	0	84	92	89
RUTH		66	76	75	77	78
MARY		82	80	76	66	76
FRED		80	60	52	98	83
SARA		98	66	88	69	45
PHIL		88	71	49	94	77
PATRICK		90	87	86	78	75
SUE		66	68	72	74	77
MARIA		75	78	67	72	77
JIM		66	69	64	72	77
JACK		44	49	70	65	68
NICOLE		66	68	78	62	70
ROGER		69	73	80	71	76
LARRY		88	81	78	73	77
ADAM		56	67	87	72	66
HELENE		66	78	88	89	90
RONALD		66	68	87	72	66
FRANCINE		66	78	88	92	88
EDDY		50	52	45	67	76
AVERAGE		72	70	75	76	75

Segment 1 is produced using

```
>A1 RETURN          (select top left corner)
/PP                 (print command)
F24                 (select bottom right corner)
RETURN              (begin printing)
```

Segment 2, shown below, is produced by typing

```
>G1 RETURN          (select top left corner)
/PP
K24 RETURN          (select bottom right corner)
```

T4	T5	T6	T7	T8	
	85	79	82	76	83
	74	80	74	69	78
	78	70	83	84	71
	79	73	78	77	75
	83	78	80	89	79
	68	79	83	55	73
	88	74	51	66	72
	59	88	77	50	73
	81	78	90	94	84
	83	90	92	95	80
	80	82	78	85	77
	70	74	78	78	72
	72	76	72	78	66
	80	78	88	84	75
	90	81	84	88	79
	78	80	79	84	80
	45	69	78	88	70
	88	90	88	94	86
	78	83	71	77	74
	86	80	85	88	83
	62	56	55	58	58
	77	78	78	79	76

A wider worksheet can be printed by slicing it up into the widest slices your printer can handle. You can reconstruct the desired end product with some cutting and pasting or taping of the printout segments.

If you expect to be dealing with very large worksheets frequently, consider investing in a printer with a wider print line. The typical print line sizes are 80, 120 and 132 characters. There are larger sizes available, specifically for financial reports. Just because your screen only accommodates 40-character lines does not mean you can only print 40-character wide segments. The maximum width of the segment depends on your printer and the commands you use; it does not depend on your screen width.

Freezing Titles

When you are working with a problem so large that not all of it fits on the screen at one time, it is very easy to make a mistake. As you scroll far over to the right, information disappears from the left of the screen. Similarly, if you scroll down far enough, information at the top of the screen disappears. Such a situation is depicted on the next page. Suppose we have the following on our screen:

```
    A           B       C       D
 1  NAME        H1      H2      T2
 2  ALAN        95      100     88
 3  BERNICE     85      95      72
...
19  HELENE      66      78      88
20  RONALD      66      68      87      <C6D>
```

Suppose we now move the cursor from its position on column D over to the right so that we can see column E. If we do that, either by pushing the cursor arrow key → or by typing >E1 RETURN, then column A will disappear from view. The names in column A have been forced off the screen. This is a very annoying situation. How can we avoid it?

The /T command (T for title) allows you to fix (or freeze) several rows or several columns, or both, in place. Suppose we wanted to ensure that the names in our example would always be on the screen? Typing

```
/T
```

generates the prompt

```
TITLES:  H V B N
```

Responding with a V (V for vertical) fixes those columns which are at and to the left of the cursor in the space they presently occupy. If you change your mind, type

```
/TN
```

The N (N for negate) undoes any previous use of the title command. So, in order to freeze the names of column A in place, we can type

```
>A1 RETURN
/TV
```

Now, if we scroll far enough to the right, instead of column A being forced off the screen, column B will be forced off instead, as shown in the next worksheet:.

```
   A            C                E
 1 NAME         H2               T3
 2 ALAN         100      ...     55
 3 BERNICE      95       ...     88
...

19 HELENE       66       ...     89
20 RONALD       68       ...     72     <C6D>
```

In the same fashion, we could have elected to freeze the labels of row 1 by typing

```
>A1 RETURN
/TH                    (H for horizontal)
```

Then all rows at or above the cursor position are frozen in place. So now, no matter how far down we scroll, row 1 will not disappear, as shown below:

```
    A            B       C       D
 1  NAME         H1      H2      T2
 3  BERNICE      85      95      72
...
20  RONALD       66      68      87
21  FRANCINE     66      78      88        <C6D>
```

Notice that row 2 (with Alan's grades) is temporarily off the screen and that we now see as far down as row 21.

You can fix more than one row or column in place . How many rows or columns are fixed in place is determined by the position of the cursor when you type the /T command. You can fix both a set of rows and a set of columns in place, simultaneously, by typing

```
/TB        (B for both)
```

Then all rows at or above the current cursor position and all columns at and to the left of the cursor will be fixed in place.

As we stated earlier, you can override any consequence of a /T command by typing

```
/TN        (negate, nullify previous /T request)
```

The VisiCalc Program in Stereo: Split Screens

If you think of your screen as a window into a large workspace, you might like to have a larger window. The next best thing would be to have a pair of smaller windows. In fact, a pair of smaller windows may be even better than a single larger window.

With the VisiCalc program, you can set up two windows which will allow you to concentrate on those parts of your worksheet of immediate concern. You retain the usual freedom to browse, scroll, change values and labels, etc. You can invoke this capability by typing

```
/W         (W for window)
```

It will respond with the prompt

```
WINDOW: H V 1 S U
```

Typing H lets you split your screen horizontally at the current

cursor position. Typing V gives you a vertically split screen at the current cursor position. You can only split your screen one way at a time--vertically or horizontally--but not both simultaneously. If you change your mind about which way to split it or where to split it you can return to the normal single screen by typing the digit "1" in response to another /W request. Do not confuse the digit "1" with the "eye" key "i" or with the letter "l" (pronounced el).

Once you have split your screen you can use any of the usual commands, on either of your two screens. If you change a value on one screen then any values which depend on it will change in both screens. Since you still have only one cursor to work with, typing the character ";" (semicolon) will select the other screen. So you can type ";" to move the cursor from one screen to the other.

The position the cursor had on one screen is saved when you type ";" so that the next time you type a ";" the cursor will return to the same position it had previously.

You might prefer not having the cursor positions in each of the two screens change independently. You might prefer that as you move the cursor around in one screen, that typing a ";" would place the cursor on the corresponding row or column of the other screen. If so, you can select what is called synchronized scrolling by typing

```
/WS       (synchronized scrolling)
```

You can disable the synchronized scrolling effected by a /WS, by typing

```
/WU       (unsynchronized scrolling)
```

The unsynchronized scrolling mode selected by the /WU command is automatically selected when you initially use /W to split your screen.

You can save a file even when you are using split screens. You do so using the same /SS command we saw earlier. All values will be saved, not just the visible ones.

If you try printing a file while using split screens, you probably won't print what you are looking at. For instance, suppose you have your screen split vertically, as shown below:

```
    A     B          F       G
1                  1
2                  2
3                  3
...
```

If you try to print the segment defined by the corners A1 and G10, by typing

```
>A1 RETURN
/PP
G10 RETURN
```

your printout will include columns C, D, and E, even though they were not visible on your screen when you set out to do this printing. The print command works on the single-screen interpretation of your worksheet.

The photograph on the cover of this book uses a horizontally split screen (command /WH). See the section entitled "IRA Revisited" in the next chapter for details on how it was set up.

How Large Can a Worksheet Be?

The size limitations on a worksheet depend on what the worksheet contains and on how much memory (RAM) your Apple II Computer has. In order to use the VisiCalc program your computer is expected to have at least 48K of RAM memory. When you load the VisiCalc program, the number displayed at the top-right corner of the worksheet shows you how much memory is available for your use. As you construct a worksheet, that number slowly decreases. If it ever reaches zero, then it will be replaced by a flashing M; that means your worksheet has filled the available memory.

On a 48K Apple II Computer, when you start with a blank worksheet, the "available memory" number at the top-right of the worksheet should read "18". This means you have approximately 18K, or 18,000 memory locations available for your use. Each location can store the equivalent of one digit or or character. Creating a worksheet involves storing each character you type plus the information about which locations are being used, which formats are in effect, what the column width is, etc.

There are so many factors involved in storing a worksheet in the computer's memory that we cannot give a simple formula to express the relationship between a worksheet and how much memory it needs. So the next best thing is to give a few examples. A worksheet with 25 columns and 40 rows filled with the single digit "0" uses up 10K of memory. A similarly-sized table of 40 columns with 25 rows uses the same amount of space. Obviously worksheets with many formulas would need more memory (remember that each position which has a formula associated with it must have its own formula stored in memory).

If you find that the uses you are making of the VisiCalc program cannot be supported with a 48K memory, you can purchase a 16K memory card which should allow you to have almost twice as much memory (18K + 16K) as before for your worksheets. The VisiCalc program can be used as-is with either a 48K or a 64K

Apple II Computer. If you feel you need more than a 64K memory, shop carefully. Using the VisiCalc program on an Apple II Computer with more than 64K of memory requires modifying the VisiCalc program. Make sure this can easily be done before you invest in additional memory beyond the normal 64K limit.

SUMMARY

Printing is performed using the /PP command. This command expects to print the values and labels contained in a rectangular segment which we define. We designate the top-left corner of the desired rectangle by positioning the cursor, before typing /PP. We define the bottom right corner in response to the prompt from /PP.

Titles or other information can be frozen in place using the /T command. We can lock into place all rows above a given row, using /TH. Similarly, all columns to the left of a given column can be locked into view using /TV. Frozen rows or columns can be unfrozen using /TN.

Split screen viewing can be requested using the window command /W. /WH will split the screen horizontally. /WV will split the screen vertically. The single cursor can be made to jump from one screen to the other by typing a ";." Split screens can be unsplit by typing /W1.

Chapter 7

A PICTURE FROM THE VISICALC PROGRAM

Sometimes a picture conveys a result more clearly than a set of numbers might. You can direct the VisiCalc program to produce some simple images or graphics, which is what we will do in this chapter. You can display these graphics on your screen; if you have a printer, you can print these graphics.

Getting Graphical Output

Some computers have elaborate equipment to produce very fancy graphical output. The VisiCalc program assumes that all you have on your computer is an ordinary alphanumeric video display, and if you want the graphics printed, that you will use an ordinary alphanumeric printer. That being the case, the kind of graphical output you can produce is the same kind you could produce on an ordinary typewriter, namely horizontal bar charts, also known as histograms. For instance, given the following table:

YEAR	PRECIP	
1970	5	
1971	4	
1972	5	
1973	3	
1974	7	
1975	1	
1976	4	<C7A>

we can arrange to produce either of the following tables:

YEAR	PRECIP	
1970	*****	
1971	****	
1972	*****	
1973	***	
1974	*******	
1975	*	
1976	****	<C7B>

Table 7.1

YEAR	PRECIP		
1970	5	*****	
1971	4	****	
1972	5	*****	
1973	3	***	
1974	7	*******	
1975	1	*	
1976	4	****	<C7C>

Table 7.2

How can we get the results shown in table 7.1? The command /F (F for format) allows you to specify a format for any desired position on the worksheet, and that format will affect the appearance of any value displayed in that position. Note the use of the word "appearance;" the underlying value will not be changed, so you can always change your mind about which format to use without worrying about losing something. Typing /F generates the prompt line

FORMAT: D G I L R $ *

The meanings of each of the letters or symbols is:

D default format as specified by /GF

G general format (labels left, numbers right)

I integer format (rounded)

L left-adjusted

R right-adjusted

$ dollar format; uses units of .01

* graphics format

We will of course examine and discuss each of these format commands as we go forward.

Selecting the * (asterisk) response will cause the worksheet position the cursor is currently situated at to display zero, one, or more asterisks, depending on the value at that position. If the value is less than one, then no asterisks will be displayed. The number of asterisks displayed is determined by truncating the value (for display purposes only; the underlying value remains intact). That simply means any fractional part of a number will be ignored. If the position is not wide enough to display the appropriate number of asterisks, the excess will not be displayed; we can if we wish enlarge the width of our columns using the /GC command. Recall that the default width of a column is nine characters. When displaying numbers or *'s, one of the nine character positions will always be left blank so that you can see where the columns are separated.

To obtain the result in table 7.1, we should begin with a worksheet which has the following appearance:

```
    A         B
1 YEAR      PRECIP
2 1970      5
3 1971      4
4 1972      5
5 1973      3
6 1974      7
7 1975      1
8 1976      4                          <C7D>
```

We would like asterisks in place of the numbers of column B (that is the only choice you get, asterisks or nothing). So let us move the cursor to B2 and request the * format, by typing

```
>B2 RETURN
/F*
```

We immediately see the number 5 in location B2 replaced by five asterisks. We then proceed to step down through the remaining positions of column B and repeat the process:

```
                  (resuming from position B2)
↓                 (down to B3)
/F*
↓                 (down to B4)
/F*
↓                 (down to B5)
/F*
etc.              ....
↓                 (down to B8)
/F*
```

You might be tempted to replicate the /F* format associated with position B2. Unfortunately, the value or formula in B2 will also be replicated. In this case, all values in column B would change to 5 (B2's value). This is not what we intended. If you began with a blank column B, then you could assign the format F* to position B2, and replicate it from B3 through B8, in the usual way:

```
>B2 RETURN
/F*                (set F* format in B2)
/R                 (replicate it)
RETURN             (use B2 as source)
B3.B8 RETURN       (target)
```

Now, if we begin typing in the values for B2, B3, etc., they will immediately be displayed with asterisks, as in table 7.1.

What if you decide you don't like this bar graph? Suppose you would like to resume displaying your numbers instead of the astersisks. You could do something drastic: delete column B, by typing:

```
>B1 RETURN
/DC              (delete column)
```

This will remove the asterisks; it will also erase your stored values. If you prefer restoring the usual numeric display, then you can restore the default format which is in effect when you start with a new worksheet. Typing

```
/FD              (default format)
```

restores the original display format. So now if you want your numbers to reappear in column B, type

```
>B2 RETURN
/FD              (default format)
↓                (down to B3)
/FD
etc.             ...
↓                (down to B8)
/FD
```

Once again, the temptation to replicate the format using /R must be resisted, because it would also lead to replicating B2's value.

Table 7.2 can be obtained by formatting column C as follows:

```
>C2 RETURN
/F*
+B2 RETURN       (Use +B2 as the formula for C2)
/R
RETURN           (use C2 as source)
C3.C8 RETURN
R                (want B2 relative)
```

We immediately see the results as shown in table 7.2, which is repeated here:

	A	B	C
1	YEAR	PRECIP	
2	1970	5	*****
3	1971	4	****
4	1972	5	*****
5	1973	3	***
6	1974	7	*******
7	1975	1	*
8	1976	4	**** <C7E>

It is worth repeating that using the F* display format does not change any values you have recorded. As you move the cursor over any positions with asterisks, you can see the original value intact and displayed on the entry content line above your worksheet.

Changing Column Widths

We have used this command before but now we can discuss some of the more subtle ramifications. The standard column width is nine characters. You can change the width of your columns by using the global command /G. It will prompt you:

```
GLOBAL: C O R F
```

Selecting the C response lets you change the width of all columns in your window (if you are using a split screen), or of all columns if you are using a single screen. You type a "C" followed by the desired width and push RETURN. If you wanted your columns to be 15 characters wide, instead of the normal nine characters wide, type

```
/GC15 RETURN
```

The narrowest column width you can request is three characters wide; this lets you see columns A through Y simultaneously. The widest column width is limited by the screen width; you can get up to 37 characters in the widest column you can view on a screen with a 40 character line.

Longer Titles

If you have felt constrained in having to use abbreviated titles or labels, you can now use longer ones, by choosing an appropriate column width, using /GC. Note that you can store longer titles than can be displayed. Changing column widths only affects the visible display. It has no effect on stored values or labels or titles.

You can take advantage of the fact that all available character positions in a column can be used when displaying labels. This means you can display one long label using two or more adjacent column positions across the same row without having it broken up by spaces. For instance, if you have the standard nine character wide column, you can still display all of a title such as "MISSISSIPPI," say in A1 and A2, by typing:

```
>A1 RETURN
MISSISSIP >          (first 9 characters)
PI RETURN            (last two characters)
```

Error Flag

If you should happen to see

```
>>>
```

fill up a worksheet position, this indicates the value at that position cannot be displayed because the column is too narrow. It

has no effect on any calculations, but it serves as a reminder that perhaps you should increase the column width, say to the value n, by typing /GCn.

Changing to a narrower column width may cause the VisiCalc program to change the way it displays values. In our earlier example, where we had the years 1970, 1971, etc. in column A, we could reduce the column width down to five (using /GC5) without affecting the appearance of the numbers 1970, 1971, etc. However if we reduce the width to four, using /GC4, suddenly the years change to "2E3." What happened?

A four-character column cannot display the four-digit number "1970" if one of the four characters has to remain blank to separate adjacent columns. So instead of displaying either "197" or "970" both of which would be misleading, the display reverts to scientific notation, with which we can approximate the intended values. The display "2E3" is read as: "The number 2 multiplied by ten raised to the power 3," which is the best approximation to 1970, or 1.97E3, that can be managed with three characters.

It bears repeating that controlling column widths does not go beyond the window or screen level. You cannot have a variety of column widths in the same window or on the same screen.

Scaling for Graphs

Not all tables of numeric information can lead to a manageable bar graph. For instance, suppose you had the following table:

Date	Market
Q1	990
Q2	940
Q3	890
Q4	920

<C7F>

You would not want to have bars with over 800 asterisks printed. For openers, you could set up a worksheet that would erase the first 850 asterisks from each line. But this is still not manageable because the "spread" or "range" in this example is 100 (the largest number is 990 and the smallest is 890). Suppose you wanted to display no more than twenty asterisks per line. You could begin by setting the column width to 21, by typing /GC21 (if you want as many as 20 asterisks to be displayed then the column must be 20+1 characters wide).

You want to either stretch or squeeze the numbers you are working with so the display can fit in the space you have designated. This process of stretching or squeezing is called scaling. Here you want to find the largest value (the maximum value) and the smallest value (the minimum value) so you can know what the range or spread is. Since you expect to save this worksheet, and change the quarterly market values, you need a

formula that finds the maximum and minimum values for you. Once again, we get to use functions.

@MAX and @MIN

The function @MAX(argument list) will choose the largest value associated with the items in its argument list. Similarly, @MIN(argument list) will select the smallest value.

In our market example, we can use column C to store computed values which will drive our graphical display. Prior to writing down a formula, let us take a closer look at the numbers we are dealing with: 990, 940, 890, and 920.

The largest number here is 990. The smallest is 890. So the range (Largest-Smallest) is 990-890, or 100. We don't want to display as many as one hundred asterisks, so let us choose a "squeezing" or "scaling" factor, say ten. For every increase in value of ten, we will display one asterisk. In order to give the bar graph an appearance of being the top part of a full-size chart let us add in a bias so we have a solid base to look at. A bias of five would not be unreasonable here. We can see what that would do to each of our numbers:

```
Value  Bias + Spread/Scale              Result

990 -> 5 + (990-MIN)/10 -> 5+100/10 = 15
940 -> 5 + (940-MIN)/10 -> 5+ 50/10 = 10
890 -> 5 + (890-MIN)/10 -> 5+  0/10 =  5
920 -> 5 + (920-MIN)/10 -> 5+ 30/10 =  8
```

The corresponding formula is:

bias + (current value - min)/scale factor.

If we agree to use a bias of five, and a scale factor of ten, this leads us to the formula:

5 + (current value -min)/10

If we plan to use column C for the bar graph, we can begin by selecting the * format:

```
>C2 RETURN
/F*
```

We must not forget to enlarge the column width from its normal size of 9 characters, to, say 21 characters. We can do this by typing

```
/GC21 RETURN
```

Then we can define the formula for position C2:

```
5+((B2-@MIN(B2.B5))/10) RETURN
```

We can now replicate both the format and the formula which we placed in C2:

```
/R
RETURN                    (copy from C2)
C3.C5 RETURN              (into C3, C4 and C5)
R                         (want B2 relative)
N                         (want the @MIN not relative)
N                         (so keep B2 and B5)
```

The fruits of our labor are shown below:

```
     A       B        C
1 Date   Market
2   Q1      990 ***************
3   Q2      940 **********
4   Q3      890 *****
5   Q4      920 ********              <C7G>
```

You could make it easier to modify this display for other values by making the bias (currently equal to 5) and the scaling factor (currently .1, the inverse of 10) variables instead of constants.

Since no columns beyond column C are being used, you could use position D1 to store the bias, and position D2 to store the scaling factor. Then the formula would become

```
+D1+D2*(B2-@MIN(B2.B5))
```

OOPS! If you do this, you have fallen into a common trap. The VisiCalc program evaluates expressions strictly from left to right, unless parentheses are used to override this evaluation priority. So here the current value from D1 would be added to the value from D2 before D2 is multiplied by the remaining expression. For our example when coming to the smallest number, 890, the sum D1+D2 would be multiplied by zero! We can correct this by using parentheses:

```
+D1+( D2*(B2-@MIN(B2.B5) ) )
```

It is now easy to try out various scaling factors and biases, without having to retype lengthy formulas. You need only change the scaling factor in D2 and change the bias in D1.

IRAs Revisited

The IRA chart on this book's cover was prepared using a slight variation of the IRA formula we saw back in chapter 4. Once again we will use yearly interest rates, with interest computed once per year. If you want to compute your interest quarterly, monthly, or daily, see the case study dealing with interest calculations. This IRA chart gives us the opportunity to

use several of the VisiCalc facilities in combination with each other. We will build this worksheet one step at a time.

The last thing we will do is adjust the column width to allow for the dramatic graph shown on the cover. It is easy to change the column width at any time, and trying to build a worksheet while using wide columns means you have to do a lot of scrolling for no good purpose. So let us begin by laying out the labels we want, and a few initial values, to obtain the following worksheet.

	A	B	C	D	E
1				IRA	GRAPH IN
2	RATE	CONTRIB	YEAR	VALUE	$20,000S
3	12	2000	1	?	?
4		?	?	?	?
...					
32		?	?	?	?

Question marks appear where we want the VisiCalc program to fill in the results. Instead of replicating the $2,000 per year contribution directly, we can place the formula +B3 in position B4, then replicate it into B5, B6, up to B32, in the relative mode, so each year's contribution is a copy of the preceding year's entry. That way, we could easily assess the impact of reducing the contribution from any point on. The typing for the B column entries is:

```
>B3 RETURN
2000 ↓          (enter 2000, advance to B4)
+B3 RETURN      (place formula in B4)
/R              (replicate formula)
RETURN          (use location B4 as source)
B5.B32 RETURN   (range is B5 to B32)
R               (make coordinate B3 relative)
```

We can fill in the YEAR column, column C, as follows:

```
>C2 RETURN
YEAR ↓          (advance to C3)
1 ↓             (advance to C4)
+C3+1 RETURN    (formula for next year)
/R RETURN       (use position C4 as source)
C5.C32 RETURN   (replicate into C5 to C32)
R               (make coordinate C3 relative)
```

For the VALUE column, column D, we don't need the results to full precision, so let us at the outset agree to display them as integers (i.e., as whole numbers; we will have a full discussion of all the formatting options in chapter 9). We can fill in column D as follows:

```
>D2 RETURN
VALUE ↓                 (advance to D3)
/FI                     (specify integer format)
```

```
+D2+B3*(1+(A3/100)) RETURN
/R
RETURN                          (use position D3 as source)
D4.D32 RETURN                   (replicate into D4 to D32)
R                               (want D2 relative)
R                               (want B3 relative)
N     ,                         (want rate A3 not-relative)
```

We should now see the numbers as they appear on the cover. Getting the graph into column E goes as follows: we could replicate the formulas of column D into column E, but the format would be copied with the formula, so you would have to enter the /F* command to request graphical output 30 times, by hand. Besides, the formula has to be changed anyway, otherwise you will get far more *s than you can display! So the simple thing to do is: assign the desired format to the first graphical position of column E, position E3, using /F*. Then assign the very simple formula +D3/20000, to position E3. So here it is:

```
>E3 RETURN
/F*                             (request display of *s)
+D3/20000 RETURN                (formula for E3)
/R
RETURN                          (use location E3 as source)
E4.E32 RETURN                   (use E4 to E32 as target)
R                               (want coordinate D3 relative)
```

The number 20,000 is used to shrink or scale our dollar values into numbers small enough to be represented by no more than 20 to 30 asterisks each. So each asterisk represents an increment of $20,000. So no asterisks are displayed for the first few years, until we reach year 7, when the accumulation with interest first exceeds 20,000. When we reach year 21, from then on, we will only see nine asterisks, since all values from year 21 onward exceed $180,000 and the default column width is nine columns. Before changing the column width, we can double check that our invisible numbers are correct. Place the cursor on position D32, type the command /FI, and you should see the integer value 27 displayed. Since the value for year 30 is $540,585, as you can see in the adjacent column, the result of dividing that by 20,000, displayed as an integer, is 27. That should convince you that the formula for the graph is reasonable. Now restore the position's format by typing /F*.

You can now set the column width as you wish. If you use /GC28 to see the maximum number of asterisks, you will be disappointed because then you can only display one column (a forty-character-wide screen, with the worksheet coordinates, can only display one such very wide column). With a little trial-and-error, you will see that /GC12 lets you display three columns, which is what we did for the cover.

To be able to see as far down as year 30, we run the risk of having our top rows scrolled off the top of the screen. We can prevent that in two ways. We could freeze these titles into place

using the /T command we saw in chapter 6. Here we chose to use the window command /WH. Placing the cursor on row 5, then typing /WH gave us a horizontally-split screen showing rows 1-4 above, and 5-11 below. We would like to leave rows 1-4 visible, and bring in as many years up to year 30 into view. So we need to jump out of the top window, into the bottom window. This is done by typing a semicolon ";." With the cursor now on the bottom window, scroll down until year 30 comes into view, and you have essentially duplicated the cover. Just a few editorial changes were made for appearances sake; the title "YEAR" was assigned the format /FR, as was "VALUE," and the title "IRA" was typed in with leading blanks by preceding it with a single " because the VisiCalc program will take a leading " to mean that all characters which follow are part of a label, up to a RETURN.

The optional diskette has the IRA worksheet we just constructed under the name C7H. If you load it, you will observe that it is displayed just as it appears on the cover, in the split screen mode, with the cursor near the bottom, in the bottom window. You can flip the cursor to the top window by pushing the ";" key. You can eliminate the split screen by using the /W1 command and you can if you wish use narrower columns to see more of them; try various values with the /GC command.

SUMMARY

The /F* command displays as many asterisks as are indicated by the integer part of the value associated with the current worksheet coordinate. The integer value is obtained by truncating a copy of the position's value. The original value is not changed; only the display is affected.

You can "turn off" the effect of the /F* command by using the /FD command.

Column widths can be changed as desired by typing /GCsize, where "size" is the number which indicates the desired column width; it must be at least three. All columns in the current window will be affected.

>>> is displayed when a column is too narrow to properly display the value it contains.

The @MAX(argument list) and @MIN(argument list) functions select the largest and smallest values in their respective argument lists.

Chapter 8

MORE COMPLEX CALCULATIONS

We have seen some of the functions supported by the VisiCalc program, such as @AVERAGE, @SUM, @MAX, etc. There are several other built-in functions provided. This chapter introduces many of these other functions.

More Functions

Some of these new functions are very simple; for instance

@COUNT(argument list)

generates the number of non-blank entries in the locations or the values in its argument list. You could use it to find out how many students took a particular exam, or how many customers have sent in a payment towards their outstanding balances.

If you had in row 1, beginning at A1:

	A	B	C	D	E
1	12	-	24	-	13

(where the - represent blank entries), then @COUNT(A1.E1) would return the value 3.

All of the functions we have seen so far can examine several items in their argument lists. The next set of functions only allow a single item as their argument. This item may be a worksheet coordinate; it can also be a formula.

The first of these functions is @ABS. @ABS(arg) returns the absolute value of its argument. That is, it strips away the arithmetic sign, so that its result is always positive.

Example: for

	A	B	C
1	77	-33	0

@ABS(A1) returns 77
@ABS(B1) returns 33
@ABS(C1) returns 0

The function @INT(arg) ignores the fraction part of the value associated with its argument "arg" and it returns the integer part of the argument as its result.

Example: for

	A	B	C
1	2.1	2.9	-3.8

@INT(A1) returns 2

@INT(B1) returns 2
@INT(C1) returns -3

The function @SQRT(arg) returns the square root of its argument if the argument value is not negative.

Example: for

	A	B	C	D
1	0	4	-4	2

@SQRT(A1) returns 0
@SQRT(B1) returns 2
@SQRT(C1) returns ERROR
@SQRT(D1) returns 1.414214
@SQRT(@ABS(C1) +1) returns 2.236

The square root of the value stored at C1, -4, is not defined, since that value is negative. The result "ERROR" will be displayed whenever impossible computations are requested.

Mathematical and Engineering Functions

Many of the functions used in mathematical and engineering problems are available in the VisiCalc program. These functions are:

@EXP(arg) returns the "arg" power of the natural number "e"

Example: @EXP(1) returns the value 2.718282 ("e" to the power 1)

@LOG10(arg) returns the base 10 logarithm of its argument.

Example: @LOG10(100) returns the value 2.

@LN(arg) returns the base e logarithm, or natural logarithm ofits argument.

Example: @LN(100) returns 4.605170
@LN(2.718282) returns 1.

Trigonometric Functions

The usual complement of trigonometric functions is supported by the VisiCalc program. For each of these, the argument is expressed in radians. Recall that 2*PI radians equals 360 degrees. The names used for these functions are:

@SIN	@COS	@TAN
@ASIN	@ACOS	@ATAN

The first row refers to the sine, cosine and tangent functions.

The second row refers to the arc-sine, arc-cosine and arc-tangent functions. The arguments must be expressed in radians.

Other Functions

There are three functions which have no arguments. These are:

@ERROR @PI @NA

You can use @ERROR to test very complicated sequences of formulas. Suppose the coordinates X1, X2, and X3 are each supposed to depend on a value at Y1, among other values. You can verify this dependency experimentally by giving Y1 the value @ERROR. All positions depending on the value at Y1 should display the result ERROR. In particular, positions X1, X2, and X3 should display the result ERROR. If they do not, you know something is very wrong. Of course this check does not prove you used the correct formula; it only shows that you did or did not reference Y1 in providing the formulas for X1, X2, and X3.

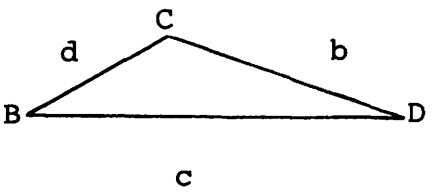
The function @PI returns the value 3.141593 which you know under the name pi (pronounced pie) in the famous formula 2 * pi * r = circumference, for a circle with radius r.

The function @NA should be used when data is not available. That alerts formulas that may be referencing empty positions to ignore them. For instance, if you were using @AVERAGE(A1.A9) at A10, and had not yet filled in any values for A1 through A9, then position A10 would display the result ERROR, because the sum of zero non-blank entries, divided by the count of non-blank entries, causes an attempted division by zero. Since division by zero is never meaningful, the result ERROR is returned. So filling unused positions with the "value" @NA would cause results which depend on these positions to be displayed as "NA" which is somewhat less alarming than seeing the message "ERROR."

Using the VisiCalc program as a Programmable Printing Calculator

There are calculations that you may be performing using a calculator that could be performed using the VisiCalc program. Why consider using the VisiCalc program instead of a calculator? It might be helpful to get a hard-copy record which you may not be getting from a non-printing calculator. Also, it may be easier for you to show another person how to enter the data using the VisiCalc program than might be the case with some calculators.

For instance, suppose you often need to compute triangular relationships. Specifically, you may need to know how long a side of a triangle is, given the lengths of its other two sides and their opposing angles. Using the names b, c, and d to stand for the lengths of sides b, c, and d, and letting the opposing angles be designated as B, C, and D,



we can use the relationship

$$d = b * COS(C) + c * COS(B)$$

to find the length of side a, given the lengths b, and c, and the angles B, and C. We can set up a worksheet as follows:

```
          A       B          C       D
1       SIDE      3          3
2       ANGLE   1.05       1.05                    <C8A>
```

for a triangle with sides b and c, of length 3, and opposing angles B and C, of 60 degrees. Here we are entering the angles in radian measure and have converted them from degrees to radians by multiplying the number of degrees by pi/180, which is .0174533. We can now enter the formula for the length of side d, in position D1:

```
>D1 RETURN
+B1 * @COS(C2) + ( C1 * @COS(B2) ) RETURN
```

The spaces in the formula will be ignored if you type them; they are present here for ease of reading. Note especially how the second part of the formula, C1 * @COS(B2), is surrounded by a pair of parentheses. The original formula,

$$d = b * COS(C) + c * COS(B)$$

did not need the extra parentheses because people are accustomed to performing multiplications before additions. But the VisiCalc program takes arithmetic operations strictly from left to right unless you override this sequencing by using parentheses.

We can build-in the conversion from degree measure to radian measure by including it in the formula for D1. The formula would then read:

```
+B1 * @COS(C2 * @PI/180)+ ( C1 * @COS(B2 * @PI/180) )
```

You could replace the expression @PI/180 by its numeric value, if you prefer, .0174533, which leads to the following worksheet:

```
          A       B         C       D
1       SIDE      3         3       3
2       ANGLE    60        60                    <C8B>
```

This lets us express the angles in degrees. The previous worksheet expected angles to be given in radians.

Accuracy

Just how accurate are the results produced using the VisiCalc program? Try the following experiments. Type /GC15 so you can see larger columns, then type

```
>A1 RETURN
123456789 >        (place 9-digit number in A1)
1+A1 RETURN        (place formula 1+A1 in B1)
```

Location B1 should be displaying the result of adding 1 to 123456789, which is 123456790. So we seem to be getting results with at least nine decimal digits of accuracy. Try using a larger value in A1, say 1234567890. Once again the result in B1 is correct; it should be 1234567891. We are getting ten-digit accuracy. Now try placing 12345678900 in A1; again the B1 result is correct. We are getting eleven-digit precision! Try a twelve-digit number, say 123456789000. The result in B1 is one larger, as it should be. But if you try another twelve-digit number, say 987654321000, the result in B1 will be off by one. So the best we can hope for is eleven-digit accuracy, which seems adequate for most uses.

Such a situation does not necessarily apply when you are using the exponentiation operation ^ and some of the mathematical functions. Consider the following table:

Formula	Result
2^2	4
2^3	7.9999999806
@SQRT(100000000)	10000
@SQRT(4)	1.9999999998

The first use of ^ has no error in its result. The second example has an error in the ninth digit. The first use of @SQRT has no error in its result. The second example has an error in the eleventh digit. So if you are going to be making extensive use of these operations and the other mathematical functions, be careful.

Printing Formulas

When you save a worksheet using the /SS command, the formulas for that worksheet are saved with it. So far, the only way you can get to see an old formula is to load the worksheet it was saved with by using the /SL command. Then you can place the cursor over the position associated with the desired formula and see it displayed on the entry content line above your screen worksheet. Isn't it possible to have formulas printed for future reference? Yes, it is possible, by using the strange file name

",S1" if your printer is accessed through slot 1 (if not, replace the digit 1 by the appropriate slot number). If you were using the worksheet we had a few pages back, the one which computed b*Cos(C)+c*Cos(B), and were to type the storage command

```
/SS
,S1 RETURN          (Use "file name" ,S1)
```

you would not create a file but cause the following printing to occur:

```
>C2:1.05
>B2:1.05
>A2:"Angle
>D1:+B1@COS(C2)+(C1*@COS(B2))
>C1:3
>B1:3
>A1:"Side
/W1
/GOC
/GRA
/GC9
/X-A1:>A1:
```

Recall that the character colon ":" can be used in place of the RETURN key, to complete the entry of a GO-TO coordinate. So what we see in this printout are the keystrokes which define everything about our worksheet. In particular, the line beginning ">D1:" displays the formula we wanted printed. If you were to examine a file created using the /SS command, by using an appropriate program with the help of the DOS software, you would see the same kind of information we see here. We will show how this is done in chapter 10.

Faster Computing

You have become accustomed to getting immediate recalculation of the current worksheet entries whenever you change any entry. There are situations where you would prefer to have the recalculation deferred until you had changed several entries. For instance, in the triangle computation we just looked at, you usually don't want the result for side d until you have typed in all the data. In some larger problems, a recalculation may take several seconds, and you can't be typing in new entries during that time. So you could save your time by having the recalculation postponed. How do you do this?

Typing /GRM puts you in the Global Recalculation Manual mode. In this mode, you can type as many entries as you wish, without any calculations being done. When you finally want the worksheet recalculated, you request this action by typing an exclamation point "!."

You can resume the usual automatic recalculation mode by typing

/GRA (global recalculation automatic)

This mode is in effect when you start out with a fresh worksheet.

The @CHOOSE Function

The @CHOOSE function uses the first item in its argument list to choose from or select one of the remaining items in its list, so it can return the value associated with that item. If you had the formula @CHOOSE(1,A5,B3) at some worksheet location, it would always return the value associated with location A5 because "A5" is the first item in the remainder of the list which follows the first argument of @CHOOSE. Similarly, @CHOOSE(2,A5,B3) always returns the value associated with position B3 because the "2" selects the second value of the remainder of the argument list. Of course, we don't need to use @CHOOSE to get the above results, but this sets the stage for the more interesting and useful cases.

If we had typed @CHOOSE(D1, A5+1, @SUM(B3.B10), 77) then if the location D1 had the value 1, this use of @CHOOSE would have whatever value A5+1 has. Similarly, if D1 had the value 2 or 3, the function would return the value corresponding to @SUM(B3.B10), or the value 77, respectively. What if D1 had some other value?

D1 Value	Result Obtained Using
1	A5+1
2	@SUM(B3.B10)
3	77
less than 1	NA
4 or more	NA
1 to 1.99...9	A5+1
2 to 2.99...9	@SUM(B3.B10)
3 to 3.99...9	77

So now if we look at the general form

@CHOOSE (index, a1, a2, ..., an)

the formula or value associated with the first argument "index" is evaluated, then it is truncated (the fractional part is discarded). The remaining integer, if it has a value in the range from 1 to n, is used to select the corresponding one of the n remaining items, one of a1, a2, ..., an. If the truncated value of "index" is less than 1 or greater than n, none of the remaining arguments can be selected, so the result is "NA."

What makes @CHOOSE particularly useful is the fact that you can specify ranges as arguments. Suppose we had the following table:

```
Item    Quantity          Code      Value
A        20               1         ?
B        10               5         ?
X         1               2         ?
...     ...               ...       ...
```

where, if an item has code i, then its value is found by selecting the i'th entry from a code table, and multiplying that entry by the quantity recorded for that item. In some applications, you would substitute Total Weight or Volume for the heading Value. If you had 50 codes, it would be painful to have to type @CHOOSE(A2,Q1,Q2,Q3,...,Q49,Q50) for a 50-entry code table in column Q. Fortunately @CHOOSE accepts a range as representing as many items as it encompasses. So we can instead write @CHOOSE(A2,Q1.Q50).

The following worksheet has columns A to C prepared in the usual way. For the VALUE column, column D, we typed

```
>D2 RETURN
@CHOOSE(C2,F1.F10)*B2 RETURN
/R
RETURN                      (use D2 as source)
D3.D4 RETURN                (D3 to D4 as target)
R                           (C2 relative)
N                           (F1 not relative)
N                           (F10 not relative)
R                           (B2 relative)
```

and we used positions F1, F2, ..., F10 to record the ten code values we needed. You can see how easy it would be to use longer code tables.

```
     A        B        C         D         E         F
1    ITEM     QUANTITY CODE      VALUE     CODES      90
2    A        20         1       1800                100
3    B        10         5        800                 65
4    X         1         2        100                 40
5                                                     80
6                                                     50
7                                                     65
8                                                    125
9                                                     64
10                                                    75 <C8C>
```

The @CHOOSE function makes it possible for you to construct far more intricate formulas.

Logical Functions

The functions we have been using either produce a numeric result or one of the messages "ERROR" or "NA." The functions we

are about to examine return a logical value. There are only two logical values: TRUE and FALSE.

The simplest way to obtain these values is to use the functions @TRUE and @FALSE. They have no arguments and you can easily guess which one produces which of the two logical values TRUE and FALSE. The more interesting way to generate a logical value is to use a comparison operator. Let us look at just one of these comparison operators here, the "less than" operator for which we use the symbol "<." If you typed the formula

```
A1 < Y7
```

then the result will be "TRUE" if the value at location A1 is less-than the value at location Y7. Otherwise the result will be the value "FALSE." That may not seem too exciting or useful. However the comparison operators turn out to be quite useful when used with the function @IF. The general form of the @IF is

```
@IF(logical value, value 1, value 2)
```

The @IF always has exactly three arguments. The first argument must always turn out to have a TRUE/FALSE value. The other two arguments, "value 1" and "value 2," are expected to have arithmetic values. When the "logical value" turns out to be "TRUE," the @IF produces the value of "value 1". When the "logical value" turns out to be "FALSE," the @IF returns the value of "value 2". So

```
@IF(@TRUE,A1,B1) produces A1's value
@IF(X1,A1,B1) produces A1's value when X1 is TRUE
@IF(X1,A1,B1) produces B1's value when X1 is FALSE
```

You can think of an @IF(condition,first,second) being read as: "If this condition is true, then use the first thing, otherwise use the second thing." The "first" and "second" things can be as complicated as you want them to be, as long as they make sense as arithmetic expressions. So we could have the following:

```
@IF( L2 < K4+1, Q1, @SUM(R1.R15) )
```

Each side of the comparison operator "<" is evaluated, then the two values are compared, and if the value of L2 is less than the value of K4 plus 1, the @IF selects Q1's value. Otherwise it will evaluate the last argument @SUM(R1.R15) and produce its value.

The comparison operators we can use are:

Symbol	Name
<	less than
>	greater than
=	equal to
>=	greater than or equal to
<=	less than or equal to
<>	not equal to

You can now handle situations such as: if an employee works no more than 40 hours in a week the normal pay rate applies and for hours in excess of 40 hours per week the normal rate is increased by a specified factor. Suppose the factor we use for overtime pay is 1.5, as is the case in the following worksheet.

	A	B	C	D	
1	NAME	HOURS	RATE	PAY	
2	JOE	40.00	5.00	200.00	
3	SAM	50.00	5.00	275.00	
4	MARY	55.00	5.00	312.50	
5	DORA	35.00	7.00	245.00	
6	BILL	50.00	4.50	247.50	
7	HARRY	34.00	5.50	187.00	
8	JOAN	50.00	6.00	330.00	
9	MIKE	45.50	5.00	241.25	
10	RONNY	52.00	5.50	319.00	
11					
12			TOTAL	2357.25	<C8D>

The pay formula can be expressed as

hours * rate + (if hours > 40, multiply extra hours by .5)

Using the coordinates we see above, we can express the pay for position D2 as:

```
+B2 * C2 + @IF( B2 <= 40, 0, (B2-40)*C2*.5 )
```

which literally reads as "multiply the hours by the rate and if the hours were less than or equal to forty then add zero for overtime; otherwise add in the result of the following (subtract forty from the hours and multiply that difference by the usual rate and multiply that by .5)." After entering this formula in position D2, we can replicate it:

```
/R                  (use D2 as source)
RETURN
D3.D10 RETURN       (use D3 to D10 as targets)
R                   (B2 relative)
R                   (C2 relative)
R                   (B2 relative)
R                   (B2 relative)
R                   (C2 relative)
```

Then we can obtain the total pay, for D12:

```
>D12 RETURN
@SUM(D2.D11) RETURN
```

What if, in using @IF, its first argument does not have a logical value? If the arguments value is "NA" then the result of the @IF is also "NA." Any other non-logical value will produce the result "ERROR."

Other Logical Functions

There are several more logical functions. Some of them may be familiar, since they are in widespread use. @NOT, @OR, and @AND may already be familiar to you; we will in any case introduce them here. The @NOT needs a single logical argument. It returns the logical complement of its argument's logical value:

Use	Result
@NOT(true)	FALSE
@NOT(false)	TRUE

So if you have the formula @NOT(A1 < B1), the result should be "TRUE" whenever A1 is greater than or equal to B1, because the complement or opposite of "<" is ">=."

The two logical functions @OR and @AND each allow you to specify several logical arguments. @OR(larg1, larg2, ..., largn) returns "TRUE" if at least one of its logical arguments larg1, larg2, ..., largn is "TRUE." @AND(larg1, larg2,..., largn) returns "FALSE" unless all of its logical arguments are "TRUE," in which case it then returns the value "TRUE."

You can combine functions as you wish, as long as it makes sense. @IF(@NOT(@AND(A1, @OR(B1,B2))), 0, 1) can produce results as shown in this table:

B1	B2	@OR(B1,B2)	A1	@AND(...)	@NOT	@IF
T	T	T	T	T	F	1
T	F	T	T	T	F	1
F	T	T	T	T	F	1
F	F	F	T	F	T	0
T	T	T	F	F	T	0
T	F	T	F	F	T	0
F	T	T	F	F	T	0
F	T	F	F	F	T	0

There are a few more logical functions; these are special-purpose functions:

@ISNA(arg) returns "TRUE" when its "arg" has the value "NA"; otherwise it returns the value "FALSE"

@ISERROR(arg) returns "TRUE" when its "arg" has the value "ERROR"; otherwise it returns the value "FALSE"

@NPV, Net Present Value Function

The time value of money has always been an important consideration. This is even more so when double-digit interest rates are the norm. The @NPV function calculates the present value of a series of future amounts FA1, FA2, ..., FAn, for years 1, 2, ..., n, at an assumed rate of return i. The general form in using @NPV is

@NPV(rate i, range of locations for future amounts)

The assumed rate of return i is specified as a decimal number; so 10% is written as .10. Let us examine a few simple cases. They are simple enough to be mentally verified. If we assume a rate of return of 10% per year (compounded annually) we should expect that a payment of $110 one year from now should be worth $100 today. We can verify this by typing

```
>A1 RETURN
110 >                          (year 1 amount in A1)
.10 >                          (rate in B1)
@NPV(B1, A1.A1) RETURN         (result in C1)
```

If you typed a plain A1 instead of A1.A1 as the second argument of @NPV, you would see the result ERROR displayed. @NPV can only have two arguments; the second argument must be a range of locations, even if only one location is involved.

Suppose we expected payments of $110 at the end of year one and $121 at the end of year two. What is the present value of such a series of payments if we continue to assume the rate of return is 10%? If we modify our formula in C1 to include both the locations for the values 110 and 120, we should see

	A	B	C	D
1	110	.10	200	
2	120			
3				

on the worksheet, where the C1 formula is @NPV(B1,A1.A2)

You can place the list of future amounts in any desired row or column. The assumed rate of return uses simple annual discounting. Since the principal use of @NPV involves estimating present value based on an educated guess as to what average interest rates will prevail for one or more years, there is not much point to using fancier interest compounding formulas. For a range of values $v_1, v_2, \ldots, v_n$ if we let d be 1+ rate of return, the net present value is calculated using the formula

$$v_1/(d)^1 + v_2/(d)^2 + \ldots + v_n/(d)^n$$

@LOOKUP

We saw the @CHOOSE function a little while ago. It lets you pick out the i'th item from a list. The @LOOKUP function goes a step further. It lets you select the i'th item from a table based upon a search argument that need not have the value "i." The table consists of two adjacent rows (or two adjacent columns). The top row (or left column)is called the search list. The row below (or column on the right) provides the search values.

You provide @LOOKUP with two arguments, as in

@LOOKUP(arg, table).

The first argument "arg" is used to examine the search list. The second argument, "table," provides the range which indicates the first and last locations of the search list.

Suppose we had the following worksheet:

	A	B	C	D
1	SEARCH	SEARCH		
2	LIST	VALUES		
3	10	31		
4	15	2		
5	25	-3		
6	45	14		

The "table" we have here fills A3-A6 and B3-B6. If we want to use it with @LOOKUP, we will specify the range A3.A6 for @LOOKUP's second argument. Whatever value is provided for @LOOKUP's first argument, the "search argument" is used to search the "search list." If we call the search argument x, then if x is less than the smallest search list value (here, the smallest search list value is 10), @LOOKUP will return the value NA. If x has the value 10, or greater than 10, but less than the next search list value 15, @LOOKUP returns the result chosen from the corresponding search value position next to the 10. In this case a value of x between 10 and 15, possibly equal to 10, but less than 15, will have @LOOKUP return the search value from row 3, which is the number 31.

In the general case, where we have a search list l_1, l_2, ..., l_n, the use of @LOOKUP requires that these values be in ascending order. Since only the first of any duplicated search list items would ever be used, as a general rule we should have our search list values obey the relationship

$$l_1 < l_2 < \ldots < l_n$$

In such a case, if we set up the table

Search list	Search values
l_1	v_1
l_2	v_2
...	...
l_n	v_n

then @LOOKUP(x, table) would operate as follows:

if $x < l_1$	then return NA
if $l_1 <= x < l_2$	then return v_1
if $l_2 <= x < l_2$	then return v_2
...	
if $l_n <= x$	then return v_n
if x is blank	then return NA
if x is ERROR	then return ERROR

If the search list includes blank values they are ignored unless the last part of the table is blank. In such a case @LOOKUP returns the value 0 if the search argument exceeds the last nonblank entry in the search list.

We can settle many uncertainties by examining the results shown in the next worksheet. The search arguments used in column D lead to the results shown in the corresponding locations in column E. The formula for position E3 is @LOOKUP(D3,A3.A6). Similarly, the formula for position E4 is @LOOKUP(D4,A3.A6), and so on for locations E5, E6, etc. The formula for location E14 differs from the others. Due to a (deliberate) typographical error it went in as @LOOKUP(D14,A3.A7). Since our table only goes as far as row 6, the search argument 75 returns the value corresponding to location B7, converting it to 0.

	A	B	C	D	E
1	SEARCH	SEARCH		ARGS	RESULTS
2	LIST	VALUES		-9	NA
3	10	31		10	31
4	15	2		12	31
5	25	-3		15	2
6	45	14		16	2
7				24	-3
8				25	-3
9				44	-3
10				45	14
11				100	14
12					NA
13				ERROR	ERROR
14				75	0

In these examples we had our search table set up in adjacent columns. You can set them up in adjacent rows if you prefer. When a search table is set up in adjacent rows, the top row holds

the search list, with the smallest values on the left. The row immediately beneath the search list row will be expected to have the search values.

SUMMARY

The following functions were discussed:

@COUNT @ABS @INT @SQRT

Mathematical and engineering functions were discussed:

@EXP	@LOG10	@LN
@SIN	@COS	@TAN
@ASIN	@ACOS	@ATAN

Special functions were introduced:

@ERROR	useful in testing
@NA	to designate "Not Available" data
@PI	provides the constant 3.1415...

The error message "ERROR" is displayed when attempting undefined arithmetic calculations.

/GRM command puts the global recalculation manual mode in effect. New entries will not cause any recalculations until you request them, by typing an exclamation point.

/GRA command forces reversion to the normal global recalculation automatic mode.

The /SS,S1 storage command produces a printout of the worksheet currently in use, showing exactly what keystrokes could have produced it. This is a way of printing the formulas you are using.

The following features were introduced:

the @CHOOSE(index, list) function
logical values TRUE, FALSE
comparison operators <, >, =, <=, >=, <>
logical functions @IF(logical-arg,value1,value2),
@NOT(logical-arg)
@OR(list of logical-args)
@AND(list of logical-args)
@ISNA(arg), @ISERROR(arg)

Finally, the net present value and lookup functions @NPV and @LOOKUP were presented.

Chapter 9

CONTROLLING FORMATS

The visual disposition of items on a worksheet is controlled by use of what are known as format commands. Except for a brief look at the "*" format for graphics which we saw back in chapter 7, we have been ignoring this aspect of the VisiCalc program until now because it has little to do with any "what if" situation. However, it is important in that the results you are producing might look very sloppy and this may lead to misinterpretations or misunderstandings. Since the format controls are few and easy to use, let us proceed.

When we dealt with the gradebook example back in chapter 5, it was mentioned that some of the average grades would actually be displayed with fraction parts showing, unless you wanted to prevent this. If you were to type in the values and labels shown in the following worksheet:

	A	B	C	D	E
1	NAME	HWK1	HWK2	HWK3	AVERAGE
2	ALAN	95	100	90	
3	BERNICE	79	95	90	
4	CHARLES	83	0	85	
5	FRED	82	80	79	
6	MARY	80	60	88	
7	AVERAGE				

the actual worksheet screen appearance, in the absence of any format controls, would be as follows:

	A	B	C	D	E	
1	NAME	HWK1	HWK2	HWK3	AVERAGE	
2	ALAN	95	100	90	95	
3	BERNICE	79	95	90	88	
4	CHARLES	83	0	85	56	
5	FRED	82	80	79	80.33333	
6	MARY	80	60	88	76	
7	AVERAGE	83.8	67	86.4	79.06667	<C9A>

Two things stand out: (1) the titles do not seem to fit over the columns as well as they should, and (2) some of the calculated averages (e.g. 80.33333) stick out like a sore thumb. Why does the VisiCalc program present the kind of display it does? How can we make the actual worksheet more presentable? This leads us to consider global formats.

Global Formats

The VisiCalc program normally tries to display each number and each calculated result to its full accuracy within the column width allowed. The default column width is nine characters. Whenever you are displaying a number, one of those characters is

always reserved for a blank character, so numbers of up to eight digits will be displayed if it takes eight digits to represent the best approximation. You can of course increase or decrease the column width using the /GC command, in which case more or fewer digits will be displayed. Similarly, for labels, as much of a label as can fit in the given column (including all character positions) will be displayed, but:

(1) numbers will be right-adjusted
(2) labels will be left-adjusted

We see that clearly in column A: the names line up nicely with the column label "NAME;" all of them are left-adjusted, by default. The B column label "HWK1" is left-adjusted, but the numbers in column B are right-adjusted, all by default. When we get to the calculated results, some of them look "nice," because they only need two digits to be represented exactly. For instance, Alan's average grade of 95 comes from (95+100+90)/3, or 285/3, which is exactly 95. On the other hand, Mary's grade does not come out as a whole number, so it is displayed in seven digits plus a decimal point. Some of the column averages come out as whole numbers and others (e.g. 83.8 for column B) are exact if displayed with an extra digit.

What we have just described, along with its ramifications, is called the general format, which is the default global format in effect when you start with a blank sheet. What other global formats are there and how do we invoke them?

Changing Global Formats

We can change a global format by typing /GF. We used /G earlier to get at /GC, to change column widths. Recall the prompt generated by /G is:

GLOBAL: C O R F

Selecting the F leads to the prompt:

FORMAT: D G I L R $ *

If we then select I, for integer, all numbers on display will be rounded to the nearest integer. Note the phrase "all numbers on display." The underlying values are not affected. You will continue to compute with full accuracy, but you only see a display to the accuracy you select. If we typed /GFI, our worksheet would now look as follows:

```
     A              B          C          D          E
1  NAME       HWK1       HWK2       HWK3    AVERAGE
2  ALAN             95        100         90         95
3  BERNICE          79         95         90         88
4  CHARLES          83          0         85         56
5  FRED             82         80         79         80
6  MARY             80         60         88         76
7  AVERAGE          84         67         86         79  <C9B>
```

This looks much better and it also makes much more sense. When the raw data is only accurate to the nearest unit, what sense does it make to report averages accurate to the nearest tenth, hundredth, or thousandth?

Now, how do we get our labels to fit more nicely? Typing /GFR selects the right-adjust option. This will force all items not otherwise being controlled to be right-adjusted in their respective columns. So our worksheet now looks like:

```
     A              B          C          D          E
1      NAME       HWK1       HWK2       HWK3    AVERAGE
2      ALAN         95        100         90         95
3  BERNICE          79         95         90         88
4  CHARLES          83          0         85         56
5      FRED         82         80         79   80.33333
6      MARY         80         60         88         76
7  AVERAGE        83.8         67       86.4   79.06667  <C9C>
```

The titles certainly look like they fit more nicely, but why suddenly did "80.33333" come back along with a few other numbers to ruin things. After all, didn't we specify we want a global integer format just before we asked for a global right-adjusted format? What is going on? Well, it turns out that the global formats are mutually exclusive.

You can select any one of the /GF options; whichever one you select undoes any previous global format selection. That may seem to limit our freedom to such an extent that we should quit now. Fortunately there is an out, as we shall see shortly.

What are the meanings of the remaining global format selections? The ones we saw earlier are included as well.

I	Integer
R	Right-adjust
L	Left-adjust
$	Display in dollars.cents
G	General format
*	Display *'s (graph)
D	Revert to default (no effect here)

We have discussed I and R; L for left-adjust is the opposite of R for right-adjust. The $ format will display any and all

numbers with two decimal points, so 80, 83.8, and 79.06667 are displayed as 80.00, 83.80, and 79.07 respectively. The dollar format is convenient for data entry of any financial transactions, as you do not have to key in the ".00" for exact dollar amounts such as $123. Of course you should not be typing the $ with the 123, nor will a $ be displayed.

The G for general-format was discussed earlier (labels left-adjusted, numbers right-adjusted, to eight-digit accuracy). The * format has the same effect as the one we saw when discussing graphical output in chapter 7. As you recall, it caused the value at an entry to be truncated to an integer number, and that number of asterisks to be displayed, up to the column width capacity. If we used /GF* here, almost all our numbers would be replaced by ********, because almost all of them are greater than eight.

The /GF "D" option for default has no effect here; it is simply a reflection of the fact that the /G command shares all but one option with the /F command, which is the next command we shall look at.

Format Command

The /G global command affects all positions for which no specific format has been provided. You can provide a specific format to a position by using the /F format command. Typing /F gives the prompt:

FORMAT: D G I L R $ *

The selections are exactly those we saw for the /GF global format command. The meanings are also exactly the same (except for D, default). What has changed? Here you assign a format using /F to a specific worksheet position. Before typing the /F command, place the cursor on the position to which you wish to assign a format. Such a specifically assigned format will override any format previously assigned using the /GF global format command. If you then replicate a position, then its format is also replicated.

Returning to our previous example, in the last round we had:

	A	B	C	D	E	
1	NAME	HWK1	HWK2	HWK3	AVERAGE	
2	ALAN	95	100	90	95	
3	BERNICE	79	95	90	88	
4	CHARLES	83	0	85	56	
5	FRED	82	80	79	80.33333	
6	MARY	80	60	88	76	
7	AVERAGE	83.8	67	86.4	79.06667	<C9D>

We had just gotten our titles for columns A through E, right-adjusted, using /GFR. This undid our previous rounding to integers using /GFI. We could now go through and assign specific

integer formats to the calculated results of column E and row 7 by typing /FI for each of these positions as we move the cursor from one to the other. It would be less work to force the whole sheet to revert to integer format, using /GFI. Of course this causes our titles to revert to being left-adjusted. But since there are only four titles we want to position, as opposed to nine calculated results, we are slightly better off this way. To recapitulate,

(1) use /GFI to round all values to the nearest integer
(2) use /FR for positions B1 through E1
(3) use /FL for positions A1 through A7

You can type:

```
>B1 RETURN
/FR > /FR >                          (B1, C1)
/FR > /FR RETURN                     (D1, E1)
>A1 RETURN
/FL ↓ /FL ↓ /FL ↓                    (A1, A2, A3)
/FL ↓ /FL ↓ /FL ↓ /FL ↓                              (A4-A7)
```

and now we see:

	A	B	C	D	E	
1	NAME	HWK1	HWK2	HWK3	AVERAGE	
2	ALAN	95	100	90	95	
3	BERNICE	79	95	90	88	
4	CHARLES	83	0	85	56	
5	FRED	82	80	79	80	
6	MARY	80	60	88	76	
7	AVERAGE	84	67	86	79	<C9E>

Why not simply replicate the /FR? If you typed:

```
/B1 RETURN
/FR
/R RETURN                    (replicate B1)
C1.E1 RETURN                 (target range)
```

not only will you replicate the R format you just assigned to B1, you will also replicate the value or expression assigned to B1. So you would inadvertently replace the labels HWK2, HWK3, and AVERAGE with the label HWK1. The new labels would be nicely right-adjusted, but they would not be the labels we want! So, if you have a situation where values have already been entered, as is the case here, use the cursor-moving technique we just used:

```
>B1 RETURN
/FR > ...
/FR > ...
```

Now that you know that replication of a format destroys the values in the target locations, you can plan ahead. In those

cases where the appearance of a worksheet matters to you, lay out the specific formats in your head. Then, go ahead and assign formats to positions which you will later fill in with labels or values; you can safely use replication here, before you fill in the values. Then for all positions where you expect to use formulas where you very likely will want to use replication of these formulas, replicate both the formulas and formats simultaneously.

Do not:	enter values into A1, A2, A3, ... and then replicate a format from A1 into A2, A3 ,...
Do:	assign a format for A1, replicate it into positions for which it is desired, and then proceed to assign values or labels
You may:	enter values into A1, A2, A3 and then later type in a format for each one

The formats you can assign using /F are the same as those you can assign using /GF and they have the same meanings except for D, default. Typing /GFD has no effect. Typing /FD assigns the default format to a specific position. If you have not used any other /GF commands at that point, the /FD will have no particular effect. If you do use some /GF command (other than /GFD), that global command will apply to each location which has been assigned the /FD format. There is a simpler way of thinking about this. Suppose you have assigned a format to some location, say format I, but you change your mind about it later. You would like that location to revert to whatever global format you might decide to use. You might try blanking out the location, using /B, hoping that would force it to revert to the default global format. But /B only erases the value or label; it does not affect whatever specific format may be in effect. Thus the need for and use of /FD.

Another Approach

Some people find it easy to plan ahead, so they have no trouble in working up both the desired formulas and the formats to best display the results, all at the same time. Other people, myself included, don't find it easy to think of formats until we have already generated the results. By then, it would seem to be necessary to type in a format for each and every position that should not revert to the global format then in effect. Fortunately, there is a better way out of this situation. It is based upon the following observation: when you save a worksheet using /SS, it stores the worksheet in a VC format file. We saw what such a worksheet looks like, in chapter 8, when we used the /SS,S1 command. Those commands which suffice to rebuild the worksheet are stored in the file. No record is made of the fact that you used replication. Each position is individually recorded with its current value and format, if any. So how does this help us?

Suppose you prepare a new worksheet by first typing in the data then next typing in formulas and labels, freely using replication as desired, with no concern for formats. When you have all the right labels and formulas and numbers where you want them and have decided to start providing "nice" formats, at that point, save your format-free worksheet using /SS. Then go ahead and type in whatever formats you want, using /F. Feel free to use replication, even if it destroys some of your data or formulas or labels. When the formatting is complete, reload the worksheet you just saved, using /SL.

Won't this undo our most recent formatting efforts? Not at all, if you in fact saved a worksheet that had no formatting associated with it. We can count on two things working in our favor:

(1) loading a worksheet using /SL does not clear the screen, so it won't erase a format we have assigned, unless a format had been saved when we created the file.
(2) loading a value or label into a position does not destroy that position's format.

A brief example will help illustrate this technique. Suppose we typed in the data as shown in A1 and B1, and a formula to place their sum in C1:

	A	B	C
1	2.1	3	5.1

and now we wanted to try out various formats. We can save this worksheet using /SS. Then we can assign the desired formats using replication if we want to. Suppose this was part of a much larger set of numbers and we wanted A1, B1, and C1 to use the $ format but we wanted everything else to be in the global default general format. We can force A1 to use the $ format and replicate it over B1, and C1, even if it changes the values and destroys the formula at C1. Type

```
>A1 RETURN
/F$                (use $ format for A1)
/R                 (replicate it)
RETURN             (copy A1's format and value)
B1.C1 RETURN       (target range)
```

The result will be:

	A	B	C
1	2.10	2.10	2.10

We can now restore the original values and the desired formulas by reloading the worksheet that we saved before assigning formats (using /SL). This will leave us with:

	A	B	C
1	2.10	3.00	5.10

If you save a worksheet that has some positions with assigned formats these positions will of course be reloaded with both the value and format when you reload the saved worksheet. We will take advantage of the freedom to defer assigning formats when we later discuss an interesting application with the title "Sales Projections."

Changes Within a Window

If you change an entry's value, or the formula which generates this value, or the specific format (assigned with /F) which controls the entry's display in one window, it will affect the entry when it comes into view in the other window. After all, each entry is unique, in spite of the fact that we can see two of them. An entry has a single value, and only one specific format, if one has been assigned.

However, if you assign a global format using /GF while in one window, it will have no effect on the other window's global format. The same holds true for any change made to column widths using /GC. You could have 5-character-wide columns in one window and 16-character-wide columns in the other. If and when you revert to a single window using /W1 the global settings in effect for the window you were examining will be used for the single window.

SUMMARY

The /GF command permits you to select one of the following formats:

I	Integer
R	Right-adjust
L	Left-adjust
$	Display in dollars.cents
G	General format
*	Display *'s (graph)
D	Revert to default (no effect here)

The default format in effect when you begin is G; it forces labels to be left-adjusted, and numbers are right-adjusted, displayed to full accuracy within the column width provided. The global format selections are mutually exclusive.

The /F command permits you to select from the same list, and have the selected format apply to the current position. The /FD default selection causes that position's format to revert to whatever global format is in effect.

Blanking an entry will not erase a format assigned using /F.

Formats may be replicated. Replicating a format will also replicate the source position's value.

Saving a worksheet before any formats have been assigned makes it easy to replicate formats without worrying about destroying any values or formulas.

Format changes while using windows (i.e., split screens) behave differently from format changes made when using a single screen.

Chapter 10

USING DISK FILES

Recall that in introducing you to the operation of the Apple II Computer, we stressed how most uses of the computer revolved around the creation and processing of files. The worksheets you have saved using the /SS command are saved as files which can be accessed by the Apple II disk operating system DOS. The importance of files is reflected in the VisiCalc program by its providing you with three ways of creating files. The one we have been using, /SS, saves a whole worksheet, which includes all values, labels, formulas, and even command mode settings such as /WH, if you happened to be using a split screen when you saved your worksheet. Before going further, let us look at a new command which creates a new kind of file and see how we can use it to rearrange values on our worksheet.

Transposing Data

Transposing data means transforming a group of rows into a group of columns, or vice-versa. Suppose you were creating a new worksheet, and had typed

	A	B	C	
1	11	12	13	...
2	21	22	23	...

when it occurred to you that you really had intended the numbers 11, 12, 13, etc. to be placed in column A, and the numbers 21, 22, 23 to go into column B. You had intended to fill in columns instead of rows, so it would have the following appearance:

	A	B
1	11	21
2	12	22
3	13	23
4	...	...

How can we get from the first worksheet to the second without clearing the screen and starting all over again? You can look in vain for commands to move rows to columns. The move command moves rows to rows, or columns to columns. What to do?

You can use a different kind of /S command. Recall that the prompt following typing the /S command was:

STORAGE: L S D I Q #

If you select the # response, this further prompts you:

DATA: SAVE LOAD

You should have placed the cursor at the top-left of the rectangular segment of the worksheet you wish to save, before responding to this last prompt; if you did not, abort the command using the CTRL-C keys, position the cursor, then retype the /S# command. If you now type S for save, the next prompt is:

DATA SAVE: FILE FOR SAVING

Respond by making up a file name; here we will use "TEST.DIF." Your response is then:

TEST.DIF RETURN

The next prompt is:

DATA SAVE: LOWER RIGHT

You should now place the cursor at the lower right position which defines the rectangular segment you wish to save, then push RETURN. If this series of prompts seems familiar, it should be; this is the pattern we followed when we used the /PP print command. Here you want to be saving as much of rows one and two as you have already filled in. Just when you think you're done, one more prompt appears:

DATA SAVE: R, C OR RETURN

We can push R or RETURN to save the segment we have selected, row by row. Otherwise, typing C saves it column by column. Suppose we respond by pushing RETURN (or typing R). After the few seconds it takes to copy these worksheet values onto the disk, we can clear our screen using /CY, then reload the values we just saved, either as they were (by row), or choose to reload them by column, even if they had been stored by row. If we choose this last option, we will have transformed our old rows into new columns, as desired. To do this, we can type:

```
/S#L          (# load command)
TEST.DIF RETURN  (to load file "TEST.DIF")
C             (to load by columns)
```

The values you had saved will be replaced on the worksheet, filling in the segment in which the cursor position designates the top-left corner. Assuming we typed the previous lines while the cursor was at position A1, we will have transformed the original worksheet

```
   A  B    C
1 11 12   13    ... Original
2 21 22   23    ... Worksheet
```

into the transposed version

```
    A    B
1  11   21
2  12   22
3  13   23
4  ... ...
```

Two things must be noted. First, the positions which are not filled in when you use the /S#L command will not be cleared; you have to clear them if you want them cleared. This means, among other things, that you could duplicate a particular pattern of values by loading the pattern several times, each time setting the initial cursor position in the desired location. Second, no formulas are saved when you use a /S#S command. Consequently, no formulas will be loaded when you use a /S#L command. We will reexamine the uses of /S#S and /S#L later in this chapter.

Saving a File for Printing or Processing

The form in which the VisiCalc program stores a file when you use the /SS command is illustrated here.

```
>B2:12345
>A2:1
>B1:"AMOUNT
>A1:"MONTH
/W1
/GOC
/GRA
/GC9
/X-A1:>A1:
```

The colon which appears following most GO TO commands is accepted in place of using the RETURN key. We obtained this printout as follows: while in the VisiCalc program, with the worksheet as follows:

```
     A       B
1  MONTH   AMOUNT
2    1     12345
```

we typed

```
/SS
ABC.VC RETURN
```

This creates a file named "ABC.VC." We can execute a new program using DOS which can "read" the file ABC.VC and display it on the screen (we will do this later in this chapter). If you examine this output carefully, you will see it is the same as that obtained by typing

```
/SS,S1    (print worksheet with formulas)
```

Files created by the /SS command are said to be in VC format, and you should add the suffix VC to remind you of that fact. The /SL command can only load files prepared in the VC format. This format contains everything that the VisiCalc program needs to know to recreate a particular worksheet, but it is cumbersome for those who may wish to do either of the following:

(1) use another computer program to generate values for subsequent processing by the VisiCalc program
(2) use another computer program to process the values in a VisiCalc worksheet.

Program 1 -> File 1 -> VisiCalc program -> File 2 -> Prog 2

The general case is illustrated above. You might have a program called Program 1 (say it is written in BASIC). It generates a set of results which it stores in a file named File 1. You would like to feed these results into the VisiCalc program. Then it might be the case that you have some results in your VisiCalc worksheet that you would like to save as File 2, so they could be processed further by another program, here called Prog 2. It would be possible to write programs such as Program 1 and Prog 2 which create and accept files in the VC format, but that would be difficult. So the VisiCalc program provides two much simpler formats for these purposes. They are the PF and DIF formats. We will discuss these two new file formats and their uses after we examine how the Apple II disk operating system could help us.

Using the Apple II Disk Operating System DOS

It will be helpful for us to make some use of the Apple II software known as the Disk Operating System DOS. The DOS reference manual has 200 pages. Don't be alarmed! We can discuss what we need to know about DOS in just a few pages.

The computer program which usually manages all of the computing resources used on or by the Apple II Computer is called the Disk Operating System (DOS). DOS makes it easier to manage your programs and files, even those you created using the VisiCalc program. We will discuss a few DOS facilities that will make our use of the VisiCalc program more productive.

Loading DOS

You can load the DOS software in exactly the same way you have been loading the VisiCalc program, except that you will use a different diskette.

(1) Place the diskette labeled "Apple II DOS System Master" in drive 1.
(2) Turn the display and computer power switches on.
(3) After a few dozen seconds, your display screen should

greet you with the message

```
DOS VERSION 3.3                    08/25/80
APPLE II PLUS OR ROMCARD       SYSTEM MASTER
```

and display the prompt character "]."

When the IN USE indicator on the disk drives stops glowing, remove the System Master diskette and insert one of the storage diskettes you have been using with the VisiCalc program. Type the word CATALOG, followed by a push of the RETURN key. This is the DOS CATALOG command. If your diskette contained the worksheets named TEST.VC, TEST.PF, and ABC.DIF, the DOS response to the CATALOG command would look like

```
        DISK VOLUME 254
T         002     TEST.VC
T         002     TEST.PF
A         008     TRIALRUN
T         012     ABC.DIF
```

where the number after the word VOLUME identifies the particular diskette. Each file on the diskette is described by one of the lines in this table. All of the files produced using the VisiCalc program are identified with a T for text. It is possible to produce text files in other ways so we cannot guarantee that a T always means the file was produced using the VisiCalc program. File TRIALRUN is marked with an A to identify is as an AppleSoft BASIC program. The other numbers (e.g., 002) indicate how much space the file occupies on the diskette, in units of a sector. A sector corresponds to 256 characters of information. One diskette can hold 496 sectors.

If you were running out of space on a diskette, you would look for the largest file you no longer need, and erase or delete it. We will see how the VisiCalc program lets us delete unneeded files later in this chapter. You can also delete files using the DOS command DELETE. For instance, typing

```
DELETE ABC.DIF
```

would "erase" file ABC.DIF and free up 12 sectors of space. Every once in a while it is a good idea to examine a diskette's directory using the CATALOG command, to see if the diskette is filling up, and perhaps deleting files that are no longer needed.

The DOS command INIT initializes a diskette in the same way the VisiCalc command /SI does. Diskettes initialized using DOS can be used as VisiCalc storage diskettes just as diskettes initialized using the VisiCalc /SI command can be used to store files and/or programs while using DOS.

Loading DOS When Using the VisiCalc Program

If you happen to be using the VisiCalc program and decide to switch to using the DOS software, you could turn the computer off and proceed as we saw in the previous section. However if you recall the storage-quit sequence we saw a long time ago, typing /SQ leads to a prompt requesting a slot number. If you place your DOS System Master diskette in drive 1 and respond with its slot number (usually number 6) then the DOS software will be loaded without your having to turn the computer off, then on again.

Using DOS to Examine VisiCalc Files

We have seen how to find out which files are on a diskette by using the DOS command CATALOG. Getting the same kind of information is rather awkward when using the VisiCalc program; it would have you pushing the → key to see the file names one at a time. How can we actually see what a particular file looks like? Doing this will help us understand the possible uses for DIF-formatted files.

We are going to type in a short program which is written in the language called BASIC. All the instructions for doing this and for using the program will be given here.

First, if you have not loaded the DOS software, do so now (instructions for doing this appeared a few pages back). If you have loaded it, you should be seeing the DOS prompt "]" on your screen. Type the DOS command NEW (end all command lines and all other lines by pushing the RETURN key). Now type the following lines

```
10 REM "EXAMINE" PROGRAM
20 D$ = CHR$(4)
30 INPUT "ENTER FILE NAME:" ; F$
40 PRINT D$; "OPEN"; F$
50 PRINT D$; "READ"; F$
60 B$ = ""
70 FOR J = 1 TO 100
80 GET A$
90 IF A$ = CHR$(13) THEN GOTO 120
100 B$ = B$ + A$
110 NEXT J
120 PRINT CHR$(1); B$
130 GOTO 60
140 END
```

and finish this by placing one of your storage diskettes in drive 1 and type the DOS command SAVE EXAMINE. A copy of the BASIC program you have just typed will be saved on your diskette under the name EXAMINE. Now type the DOS command RUN. That will cause

the program you just typed to take control of the Apple II Computer. The program will prompt you with the request

```
ENTER FILE NAME:
```

Respond with the name of a VisiCalc worksheet which you have stored on the storage diskette you just placed in drive 1. If you can't remember any, type a silly name and the program will give up. Then use the CATALOG command to see what you have in drive 1. Repeat the RUN command and type in the name of one of your VisiCalc worksheets.

The program will start showing you line after line of information. It will eventually stop with the messages

```
END OF DATA
BREAK IN LINE 80
```

From now on you can use this program whenever you want to examine any of the files you have created using the VisiCalc program. Since you now have a copy of this program on a storage diskette you no longer have to type in the 14 lines of the BASIC program. Here is the procedure you should follow the next time you wish to use this program:

(1) Load the DOS software as described earlier.
(2) Place the storage diskette which has a copy of the EXAMINE program in drive 1.
(3) Use the DOS command LOAD EXAMINE.
(4) Verify the program is intact using the DOS command LIST. It should display the same 14 lines you had saved.
(5) Use the DOS command RUN to have your program perform.

If the VisiCalc file you examine has any GO TO commands (lines of the form >A1:xxx) in it then you are examining a VC-formatted file. If it only has the labels and values you normally see on a worksheet then you are looking at a PF-formatted file, the kind of file we will discuss next. If it is none of the preceding then it must be a DIF-formatted file, which we will discuss last.

PF Printer Format Files

When you print a copy of your display screen using the /PP command, you see exactly what would be stored in a file if you had used the /PF command, as in:

```
/PF                 (create a "print file")
TEST.PF RETURN      (name it TEST.PF)
```

You could now write programs in BASIC which accept such a simple format; it is just a table of numbers and labels. In some cases, this format is too simple. It might not provide the kind of information that would be useful to a program. Do not bother

thinking of writing programs to generate such PF-formatted tables because there is no way to feed them into the VisiCalc program. You can produce them with the VisiCalc program but you cannot feed them back in. For these reasons we need something more flexible. This leads us to the next (and last) of the kinds of files we will deal with.

DIF™ Files

DIF is a trademark of Software Arts, Inc. The DIF data interchange format was the one used when we transposed rows into columns earlier in this chapter, using the /S#S and /S#L commands. It is similar to the PF format in that no formulas are saved but it retains more information than the ultra-simple PF format does. It spells out explicitly how many rows and columns have been saved and it identifies each position's content as either a label or a number.

The /S#S command leads to creating a file in the DIF format. Consider the following worksheet:

```
    A     B
1  ABC   123
2  XXX  -321
```

If we save it in the DIF format, by typing:

```
/S#S
TEST.DIF RETURN
```

and then print the file "TEST.DIF" by using our EXAMINE program (this requires loading DOS of course) you will see the somewhat strange printout:

```
TABLE
0,1
""
VECTORS
0,2
""
TUPLES
0,2
""
DATA
0,0
""
-1,0
BOT
1,0
"ABC"
1,0
"XXX"
-1,0
BOT
0,123
V
0,-321
V
-1,0
EOD
```

Such a file always begins with the word TABLE. The word VECTOR refers to each row, and the word TUPLE refers to each column. The data in a DIF file is stored in the order TUPLE 1, TUPLE 2, ... Any numeric entry is always identified by having a " 0, " preceding it. A label (e.g. ABC) is always preceded by a line of the form "1,0".

You can load in a file (i.e., read in the corresponding worksheet, keeping in mind that it cannot provide any formulas or formats) prepared in the DIF format by using the command /S#L. Return to the beginning of this chapter if you wish to review the prompts associated with both the /S#S and /S#L commands.

Using DIF Files

The VisiCalc reference manual has a special section on DIF files. It provides three sample programs written in BASIC.

The first of the sample programs has the title "Dumping a DIF File." Dumping is the elegant word computer people use to describe the process of displaying the content of a file. You can use this program to look a DIF-formatted files just as we did earlier with our EXAMINE program. To do so, follow a similar procedure: load DOS, type the command NEW, type the one-and-one-half pages of the BASIC program just as they appear in appendix B of your VisiCalc reference manual. Then save a copy on one of your storage diskettes by using the DOS command SAVE followed by a nice name such as DIFDUMP. You can "execute" the program by typing the DOS command RUN. The program will ask you

for the name of the DIF file you wish to examine. You can later reload DOS and use the DIFDUMP program you have just saved on a diskette by typing the DOS command LOAD DIFDUMP (you should have inserted the diskette containing your DIFDUMP program in drive 1), followed by a RUN command.

The second sample program in the VisiCalc reference manual is entitled "Printing a Worksheet from a DIF File." Haven't we already done this? We have used the EXAMINE program to display a DIF file and seen the words TABLE, TUPLES, etc. appear. In this second program, the prompts you might see are depicted below:

```
FILE NAME:
COLUMN WIDTH:
SAVED BY ROW OR COLUMN (R OR C) :
PRINT THE WORKSHEET (Y OR N) :
...
```

The point in showing you the second BASIC program is to give you a model which you can modify to suit your needs. This sample program illustrates how a program can interpret a DIF file and extract the useful information it contains. Reverting to an earlier sketch,

Program 1 -> File 1 -> VisiCalc program -> File 2 -> Prog 2

a suitably modified "Creating a DIF File" program (discussed next) can serve as Program 1. Similarly, a suitably modified "Printing a Worksheet from a DIF File" program can serve as Program 2. Both File 1 and File 2 can and should use the DIF format.

The last of these programs, entitled "Creating a DIF File" will, when executed, ask you several questions:

```
WRITE THE FILE (Y OR N):
FILE NAME:
ROW       COLUMN
...
...
```

If you examined the file created by using this program, using the EXAMINE program, you would see a DIF-formatted file, with the words TABLE, TUPLES, VECTORS, along with the data you typed in. This file could be loaded by the VisiCalc program, using its /S#L command.

Deleting Files

You can delete a file using the /SD command. It will prompt you to request a file name, which you can type, followed by RETURN. If you can't remember the exact name, push the → cursor arrow key. This will cause the first of your file names to be displayed. Keep pushing → until the right name appears, then push RETURN to have that file deleted. You can also use the

→ cursor arrow key with the /SL and /S#L commands, instead of typing a file name.

When using any of the commands which store a worksheet or worksheet values (/SS, /S#S, /PF) the VisiCalc program does check to see if the file name you are assigning is already being used. This helps prevent you from inadvertently destroying an older file of the same name. Keep in mind that names which have different suffixes are considered different. Thus TEST.VC, TEST.DIF, and TEST.PF are treated as three different file names.

SUMMARY

/SS stores all the information necessary to construct a worksheet including formulas and formats in a diskette file. Such a file can be processed by programs operating under the DOS software but it is aukward.

/S# stores labels, values, and descriptive information such as the number of rows and columns (but no formulas or formats), in a file specifically to make it easier to handle with a program operating under DOS.

/PF only stores labels and values, just as would appear on a printout generated using a /PP command. The file created using /PF should have the suffix PF attached to its name.

Loading files (restoring worksheets): you can load (restore) a VC format file using /SL. You can load a DIF format file using /S#L. You can't load a file which is in the PF format.

The printout generated by a /SS,S1 command shows what is contained in the VC file that would have been created using /SS. The printout generated by a /PP command shows what is contained in the PF file that would have been created using /PF. The displaygenerated by using the EXAMINE program after DOS has been loaded shows what is contained in the DIF file that would have been created using /S#S.

The rows of a worksheet can be transposed into columns, or vice-versa, by saving the worksheet in the DIF format, by using the /S#S command. If the file is created by rows, then reloading the worksheet by columns, using /S#L, will cause the desired transposition.

/DF is used to delete files.

An attempt is made to prevent you from destroying an existing file by overwriting it with a new file.

Chapter 11

CASE STUDIES

This chapter consists of several case studies. This will provide us with an opportunity to review many of the things we have been discussing while developing some useful applications. We will also see some new features of the VisiCalc program.

Case Study A: Expense Log

You would like to keep track of your major expenses. How can the VisiCalc program help you in this task? If you were doing it by hand on a paper worksheet, it might look like this:

Date	Item	Cost	Total
1/20	battery	55.98	55.98
1/21	suit	133.00	188.98
2/3	desk	250.00	338.98

The Total column is the running total of expenses incurred to date. You can set up a worksheet by typing the titles in row 1:

```
>A1 RETURN
DATE > ITEM > COST > TOTAL RETURN
```

Then you can try entering the first item into row 2:

```
>A2 RETURN
1/20 > BATTERY > 55.98 RETURN
```

As you look up at your screen, you will be in for a surprise:

```
     A     B          C        D
1  DATE  ITEM      COST     TOTAL
2  .05   BATTERY   55.98                    <C11A>
```

Surprise! The date "1/20" for January 20 does not appear in position A2. What you see is the number .05 which, upon reflection, you connect with the arithmetic expression 1/20 (one divided by twenty). When you were typing 1/20 into position A2, the VisiCalc program interpreted it as a value because it started with a digit. Out of desperation, you might try changing the entry to Jan 20. That is not necessary. You can override the standard interpretation by the VisiCalc program simply by first typing the character " (quotation mark) immediately followed by your numbers. The " character will not be stored or displayed. Its only use is to condition the VisiCalc program to treat the remainder of the entry as a label, instead of taking it as a value. Do not confuse the quotation mark " with the apostrophe '. Two apostrophes '' will not work any better either. So we can reenter our date 1/20 into A2 by typing

```
>A2 RETURN "1/20 RETURN
```

and location A2 will now display the title 1/20.

We want the worksheet to maintain a running total for us in column D. So the running totals can be computed as follows:

```
use +C2 for D2
use +D2+C3 for D3
use +D3+C4 for D4
```

We can use +D1+C2 for D2, and have a nice pattern to replicate. The worksheet can be completed as follows:

```
>D2 RETURN
+C2+D1 RETURN
/R                  (replicate)
RETURN              (use D2 as target)
D3.D4 RETURN        (copy into D3 and D4)
R                   (want C2 relative)
R                   (want D1 relative)
```

We can fill in the values and description for the remaining two items and our worksheet now has the appearance:

```
    A      B          C        D
1   DATE   ITEM       COST     TOTAL
2   1/20   BATTERY     55.98    55.98
3   1/21   SUIT       133.00   188.98
4   2/3    DESK       250.00   438.98          <C11B>
```

If we now want to add a new expense, say a lamp for 42.00 purchased on 2/15, we can type this information and finish the line by replicating the running-total formula:

```
>A5 RETURN
"2/15 → LAMP → 45.00 →
 ↑                  (up to D4)
 /R                 (replicate)
RETURN              (and use it as source)
↓                   (use position D5 as target)
 R                  (relative)
 R                  (relative)
```

Our worksheet now has the following appearance:

```
    A      B          C        D
1   DATE   ITEM       COST     TOTAL
2   1/20   BATTERY     55.98    55.98
3   1/21   SUIT       133.00   188.98
4   2/3    DESK       250.00   438.98
5   2/15   LAMP        45.00   483.98          <C11C>
```

Dollar Format F$

In keeping track of expenses all the values are in dollars and cents. That being the case, we can save some typing by letting the VisiCalc program fill in zeroes for any missing pennies if we use the /GF$ command. This will cause numeric entries in all positions not specifically formatted by a /F command to adopt the $ format which always displays numbers in the form dd...ddd.pp.

Type /GF$. From then on, typing a value such as 250 will cause it to be displayed as 250.00. Typing a value 34.6 causes it to be displayed as 34.60. If you were working with mills (one tenth of a penny, a common measure when speaking of property tax rates) and you typed 12.342, the value 12.34 would be displayed. As always, the higher precision numbers you type will be used in all calculations even though the results may only be displayed to two decimal points of accuracy, at your request.

Another Approach

If you do not use the preceding expense log very often you might forget the steps involved in replicating the running total formula as you add new items to the log. We will examine another approach that is easier to use. It involves eliminating column D which displays the running total and having two extra rows at the bottom, one of which displays the grand total, as shown below:

```
      A      B              C
1  DATE   ITEM           COST
2  1/20   BATTERY         55.98
3  1/21   SUIT           133.00
4  2/3    DESK           250.00
5
6         TOTAL          438.98            <C11D>
```

The only formula we need is the one for position C6:

```
>C6 RETURN
@SUM(C2.C5) RETURN
```

Now, as we wish to add new expense items, we can insert a new row just above the blank row and the formula for the total will be automatically adjusted to include the new cost we just inserted. So, if we were using this latest worksheet and wanted to log the expense for the lamp we saw earlier, we could type:

```
>A5 RETURN   (select the blank row)
/IR          (push it down by inserting a new row)
"2/15 > LAMP > 45.00 RETURN
```

When you insert a new row (or column), the cursor is left sitting on the new row (or column). Our worksheet now looks as follows:

	A	B	C	
1	DATE	ITEM	COST	
2	1/20	BATTERY	55.98	
3	1/21	SUIT	133.00	
4	2/3	DESK	250.00	
5	2/15	LAMP	45.00	
6				
7		TOTAL	483.98	<C11E>

In the previous version of the worksheet where we kept a running total, why couldn't we use the same technique, that of inserting a row, and get a similar effect? Let us reexamine the earlier worksheet:

	A	B	C	D	
1	DATE	ITEM	COST	TOTAL	
2	1/20	BATTERY	55.98	55.98	
3	1/21	SUIT	133.00	188.98	
4	2/3	DESK	250.00	438.98	<C11F>

We could have created this worksheet with a totals row, as shown below:

	A	B	C	D	
1	DATE	ITEM	COST	TOTAL	
2	1/20	BATTERY	55.98	55.98	
3	1/21	SUIT	133.00	188.98	
4	2/3	DESK	250.00	438.98	
5			TOTAL	438.98	<C11G>

But if we use the /IR (insert row) command to add a new item between rows 4 and 5, we would get the following result after we filled in the entries for the new item, in the new row 5:

	A	B	C	D
1	DATE	ITEM	COST	TOTAL
2	1/20	BATTERY	55.98	55.98
3	1/21	SUIT	133.00	188.98
4	2/3	DESK	250.00	338.98
5	2/17	INK	20.00	
6			TOTAL	438.98

The total at the bottom will be correct because the row insertion operation introduces a new row which is automatically included in the range of the formula that was at old position D5. However no formulas for the new row were duplicated. That is why the running total in position D5 is blank.

Expense Log with Categories; Drawing Lines

If you want to distinguish between your expenses, say between those that are tax-deductible and those that are not, you could try to set up a worksheet such as:

```
    A     B           C       D
1   DATE  ITEM        TAXD    NOTTD
2   1/20  BATTERY     55.98
3   1/21  SUIT                133.00
4   2/3   DESK       250.00
5   ------------------------------
6        TOTALS      305.98   133.00          <C11H>
```

Any time you wish to draw a horizontal line, you can use the <u>repeating label</u> command. Recall that the prompt line upon typing a / is:

```
COMMAND: BCDEFGIMPRSTVW-
```

We have finally come to the - (dash or minus sign) part, even at the cost of skipping over a few of the other items. If you select the - any characters you type next will be repeated within the current worksheet position. You can then replicate that entry's content as far across as you wish. In the above worksheet we obtained the dashed line in row 5 by typing:

```
>A5 RETURN
/-         (select repeating label command)
-          (use "-" as the desired label)
RETURN
/R RETURN
B5.D5 RETURN
```

If we had wanted a fancier line in row 5, such as:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

we could have set up position A5 using:

```
/-
+- RETURN
```

and replicated that position across row 5.

SUMMARY

We have seen two approaches to setting up a worksheet for use in a situation in which you will expect to be adding new items at the bottom (or new columns to the right). It is somewhat simpler to use row insertion (/IR) for this purpose if you set up your worksheet anticipating this.

/GF$ is the global format dollar command which lets you enter dollars and cents amounts without typing the rightmost zeroes for pennies. The values will always be displayed in the form ddd...dd.pp provided that the column width is adequate.

The repeating label command /- lets you provide one or more characters which will be duplicated to fill the current location. You can extend these horizontally across your worksheet by using replication.

Case Study B: Portfolio Evaluation

From time to time, you may wish to see where you stand in terms of your current investment holdings. Suppose you owned five shares of the ABC company, 100 shares of WOW, and 20 shares of XYZ. You could set up the following worksheet:

	A	B	C	D
1	Stock	Shares	Price	Worth
2	ABC	5	110	550
3	WOW	100	45	4500
4	XYZ	20	75	1500
5			Total	6550

Periodically you can check the current stock prices and update those that have changed. Suppose WOW goes up to 65. Then typing

```
>C3 RETURN
65 RETURN
```

will change the 45 into a 65 and simultaneously update the Worth entry to 6500 and the bottom Total value to 8550.

We can set a similar worksheet as follows. First we can enter the titles in row 1 beginning at position A1:

```
>A1 RETURN
STOCK → SHARES → PRICE → WORTH →
```

Then enter the ABC stock information in row 2 by typing

```
>A2 RETURN
ABC → 5 → 110 →
```

Similarly we can enter the WOW and XYZ information in rows 3 and 4 by typing

```
>A3 RETURN
WOW → 100 → 45 →
>A4 RETURN
XYZ → 20 → 75 →
```

We can create a dashed line to separate the bottom TOTAL row from the other rows by typing

```
>A5 RETURN
/-                  (repeating label command)
-                   (use dash to fill in position)
RETURN
/R                  (replicate to create a line)
RETURN
B5.D5               (extend into B5, up to D5)
RETURN
```

Then the extensions (Shares held * Prices) for column D are defined by:

```
place B2 * C2 in D2
place B3 * C3 in D3
place B4 * C4 in D4
```

Let us enter the formula for D2:

```
>D2 RETURN
+B2 * C2 RETURN
```

and then we can replicate it for D3 to D4.

```
/R
RETURN     (replicate from D2)
D3.D4      (copy into D3 and D4)
RETURN
R          (relative)
R          (relative)
```

Finally we can enter the label "TOTAL"" in C6 and request a total for the Worth column by typing:

```
>C6 RETURN TOTAL >
@SUM(D2.D5) RETURN
```

We deliberately chose to include D5 in the summation. It is currently filled with dashes, so its value will be ignored. But if we later choose to add stocks to our list, a row insertion where the dashed line is located will adjust the TOTAL formula to include the new stock's worth.

Our worksheet now has the following appearance:

```
      A       B       C       D
1  STOCK   SHARES  PRICE   WORTH
2  ABC        5     110      550
3  WOW      100      45     4500
4  XYZ       20      75     1500
5 --------------------------
6                  TOTAL   6550                <C11I>
```

If we now buy a new stock, say 50 shares of SUPER at $33, we can insert this stock in front of WOW, which is in row 3:

```
>A3 RETURN
/IR  (insert row, pushing down rows 3 through 6)
SUPER > 50 > 33 > +B3*C3 RETURN
```

and now we see

```
    A       B       C       D
1   STOCK   SHARES  PRICE   WORTH
2   ABC       5     110      550
3   SUPER    50      33     1650
4   WOW     100      45     4500
5   XYZ      20      75     1500
6  ---------------------------
7                   TOTAL   8200                    <C11J>
```

We could avoid typing in the worth formula each time we add a new stock by replicating an existing row and correcting the name and values. For instance, before we introduced SUPER, we could have proceeded as follows:

```
>A3 RETURN          (select WOW row 3)
/IR                 (push WOW down 1 row)
↓                   (WOW now in row 4)
/R
.D4                 (replicate A4 to D4)
RETURN              (replicate the worth formula)
A3 RETURN
R                   (relative)
R                   (relative)
```

It's a tossup whether you prefer to retype a formula such as +B3*C3 RETURN, or type /R RETURN D3 RETURN. Clearly replication is preferred if you have a more complicated formula. Couldn't we have proceeded in another way? Why not fill column D with the worth formula, right from the beginning. Anticipating that we won't have more than 50 rows in our worksheet, we could fill D2 with a formula, then replicate it through positions D3, D4, ..., D51. This will work, except that since most of the rows will have no data in columns B or C, and the product of two blank positions is 0, and you will see a column of zeroes down column D. If you try to erase them by typing /B (to blank out an entry) you will also erase the corresponding formula. So it appears we will make do either with retyping formulas as needed, or using replication.

Portfolio Evaluation with Gains and Losses

Suppose we wanted to keep track of our gains or losses on each stock. We need a column headed "COST" to record our original cost, and one headed "GAIN." We can insert the "COST" column just to the left of the PRICE column which is in column C. We are starting with our earlier worksheet:

```
     A        B       C        D
1  STOCK   SHARES   PRICE   WORTH
2  ABC        5      110     550
3  SUPER     50       33    1650
4  WOW      100       45    4500
5  XYZ       20       75    1500
6  -------------------------
7                  TOTAL    8200                    <C11K>
```

To insert a COST column between the SHARE and PRICE columns, we can type:

```
>C1 RETURN
/IC        (insert column to the left)
```

Since the cursor remains in its old position when you perform an insertion the cursor is now still in position C1, so we can proceed to type in the title "COST" and fill in the costs for our four stocks:

```
COST ↓ 100 ↓ 33 ↓ 50 ↓ 60 ↓ /-- RETURN
```

The /-- fills the gap in the dashed line. Our worksheet now looks like

```
     A        B      C       D        E
1  STOCK   SHARES   COST   PRICE   WORTH
2  ABC        5      100    110     550
3  SUPER     50       33     33    1650
4  WOW      100       50     45    4500
5  XYZ       20       60     75    1500
6  ---------------------------------
7                           TOTAL   8200          <C11L>
```

We would like the "GAIN" column to be right-most so let us place the title "GAIN" in F1. The gain for a stock is defined as the current worth minus the original cost. For the purpose of this example we will ignore any other expenses such as broker commissions. The formula

```
current worth - (shares * share cost)
```

translates into our coordinates for row 2 as:

```
+E2 - (B2 * C2)
```

We can assign this formula to F2 and the title "GAIN" to F1 by typing:

```
>F1 RETURN GAIN ↓
+E2 - (B2 * C2) RETURN
```

Then we can replicate for rows 3 through 5 by typing:

```
/R
RETURN
F3.F5 RETURN
R          (want all 3 coordinates relative)
R
R
```

We can ask for a grand total for our gains, in F7, by typing

```
>F7 RETURN
@SUM(F2.F6) RETURN
```

Our table now looks like:

```
     A       B       C       D      E      F
1  STOCK   SHARES   COST   PRICE   WORTH  GAIN
2  ABC        5      100    110      550     50
3  SUPER     50       33     33     1650      0
4  WOW      100       50     45     4500   -500
5  XYZ       20       60     75     1500    300
6  ---------------------------------
7                          TOTAL  8200        <C11M>
```

It is a simple matter to provide the net gain inposition F7 and if we also fill in the dashed line we obtain the following worksheet:

```
     A       B       C       D      E      F
1  STOCK   SHARES   COST   PRICE   WORTH  GAIN
2  ABC        5      100    110      550     50
3  SUPER     50       33     33     1650      0
4  WOW      100       50     45     4500   -500
5  XYZ       20       60     75     1500    300
6  ---------------------------------------
7                          TOTAL  8200   -150    <C11N>
```

Replication Revisited

We have been building up our worksheets one feature at a time so we could more easily follow how and why they were constructed that way. Suppose we knew exactly what we wanted. Couldn't we set up worksheets with less effort?

As a matter of fact, that will often be the case. When you need to replicate more than one thing for a single worksheet it may be possible to replicate them all at once. This is the case when you want to replicate things that happen to be in adjoining columns (or adjoining rows).

Suppose we were just setting up our WORTH and GAIN columns in columns E and F:

```
>E2 RETURN
```

```
+B2 * D2          (formula for WORTH, in E2)
>                 (advance to F2)
+E2 - (B2 * C2)   (GAIN for F2)
<                         (back to E2)
```

We can now replicate both formulas simultaneously into positions E3, E4 and E5 for the WORTH formula in E2, and into F3, F4 and F5, for the GAIN formula in E2:

```
/R
.F2         (source range is E2 to F2)
RETURN
E3.E5       (target range for col E)
RETURN
R           (for B2)
R           (for D2)
R           (for E2)
R           (for B2)
R           (for C2)
```

When you are replicating two or more adjacent items you must specify the beginning and ending coordinates of the source row (or column). Here it is simply E2 and F2, both in row 2. From that information the VisiCalc program can determine that you want to replicate two things (namely, the things in E2 and its immediate neighbor F2).

For the target range you only specify the target range for the first source item. The VisiCalc program will replicate the first source item as usual, into the specified target range. Then when it comes to replicate the next source item, it knows how many times to replicate it and which row (or column) to begin in, since that information is implicit in the response you gave to the request for target range coordinates. Think of painting into a rectangle:

```
top edge: source coordinates
left side: target coordinates

E2 ..............F2
E3
.
.
.
E5                x
```

It takes little imagination to see that "x" marks where the bottom edge and the right side of this rectangle are located. That is the mental model you need when trying to replicate several things at a time, intending to fill several adjacent columns (or rows).

In the process of replicating you will be prompted as usual to indicate whether the coordinate used in a formula is to be relative or not-relative.

What if you bought stocks at different times, presumably at different costs? Then you should keep track of each purchase almost as if it were a different stock.

Avoiding Retyping Data

Suppose you had bought shares of ABC on five different occasions at different costs. You would like to enter them in your portfolio worksheet. Suppose you had purchased the following quantities at the indicated prices:

Number	Cost
5	100
5	110
10	101
20	90
25	80

Whenever you care to evaluate your portfolio you have to enter ABC's current price five times. Suppose today it was $123 per share. The worksheet might have started looking like:

	A	B	C	D	E
1	STOCK	SHARES	COST	PRICE	WORTH
2	ABC	5	100	110	550
3	ABC	5	110		
4	ABC	10	101		
5	ABC	20	90		
6	ABC	25	80		
7	SUPER	50	33	33	1650
8	WOW	100	50	45	4500
9	XYZ	20	60	75	1500
10					
11				TOTAL	8200

<C110>

You can update ABC's price of 123 by typing into positions D2, D3, ..., D6, as follows:

```
>D2 RETURN
123 ↓ 123 ↓ 123 ↓
123 ↓ 123 RETURN
```

or by typing

```
>D2 RETURN
123 RETURN
/R                  (replicate)
RETURN
D3.D6
RETURN
```

Copying Values with #

There is another way. It can be used when replication would not be possible. The character # (SHIFT 3 key) can be used to copy a value. Typing a # will copy the numerical value of a worksheet position into the edit line which appears above the worksheet, at the left. You can use it here as follows: enter the first copy of the number 123 as usual:

```
>D2 RETURN
123 RETURN          (or advance with ↓)
```

Then if you wish to fill in a value for D3 advance the cursor to it and type a +. Move the cursor back to D2 and type a #; the number 123, in position D2, will appear on the edit line. Return the cursor to D3 and push RETURN (or any cursor arrow). The edit entry "123" will be copied into location D3. The steps are:

```
>D2 RETURN
123 ↓               (original value)
+                   (+ for D3)
↑                   (back to D2)
#                   (copies the D2 value to edit line)
↓                   (back to D3)
RETURN              (123 now in D3)
```

You can then repeat the process, copying from D3 to D4, and so on. In this example, replication in column D may seem simpler. This technique using the # character is not restricted to copying into a column or a row. You can jump around as desired.

Unfortunately you can't use the # to copy titles or formulas. It can only be used to copy numbers.

Another Variation

If you expect to have many instances of a given stock bought at different prices it is a nuisance to update its current price several times, even with the help of "#." You could avoid having to do this by typing a simple formula for every "PRICE" entry for a given stock, save one.

For instance, the first time you record your holdings of ABC stock bought at 100, with a current value of 123, you would type

```
>A2 RETURN
ABC → 5 → 100 → 123 → +B2*D2 RETURN
```

The next time you buy ABC stock, say at 110, you can type

```
>A3 RETURN
/IR                 (insert row above old row 3)
```

```
↑                    (up to row 2)
/R
.E2 RETURN
A3 RETURN
R                          (make B2 relative)
R                          (make D2 relative)
↓                          (down to A3)
ABC → 5 → 110 → +D2 RETURN
```

Then, whenever you update ABC's current price (in D2) the other ABC entries will reflect the same current price in their column D spot. This is how we produced the following worksheet:

```
   A       B      C       D      E
1  STOCK   SHARES  COST   PRICE  WORTH
2  ABC        5     100    123     615
3  ABC        5     110    123     615
4  ABC       10     101    123    1230
5  ABC       20      90    123    2460
6  ABC       25      80    123    3075
7  SUPER     50      33     33    1650
8  WOW      100      50     45    4500
9  XYZ       20      60     75    1500
10           ------------------------
11                      TOTAL 15645          <C11P>
```

SUMMARY

Portfolio evaluation uses simple formulas and the @SUM function. Adding new stock entries involves using row insertion with the /IR command and either retyping a simple formula or using /R to replicate it.

We have seen how several replications can be accomplished with one replicate command. This is possible when you wish to replicate items which appear in adjacent rows, or in adjacent columns.

The character # can be used to copy numeric values from one position to another.

Case Study C: Computing Your Net Worth

Your "net worth" is an illusive beast. It keeps changing, even if you remember to include all of your relevant assets and liabilities, simply because the values of the items which make up your net worth can fluctuate almost daily. The problem of defining what net worth means is compounded by the differing approaches taken in the accounting community. Some would evaluate your assets based on their original cost while others would evaluate them based on current market value; still others may use replacement cost, and so on. Be that as it may, once you choose an approach that meets your needs, the same kind of VisiCalc worksheet can be used in all cases. So let us proceed.

Your net worth is the sum of all of your assets, minus the sum of all of your liabilities. So we can begin with the simple formula:

Assets - Liabilities -> Net Worth

We can begin by concentrating first on your assets, then worry about the liabilities later on. We could start out by listing the major categories for your assets. You may wish to refine or change this later.

```
ASSETS
  Cash
    Savings account
    Money fund
  Real estate
    Home
  Securities
    Stocks
    Bonds
  Personal property
    Furniture
    Clothing
    Car
  Long-term
    Insurance
    Pension
-----------
TOTAL
```

We can see that if we want to avoid very cryptic abbreviations, that we will need wider columns to store these labels. We can start out by requesting fifteen character wide columns, which allow us to display fifteen character labels or fourteen character values. We can change this width again later if we wish to. Getting fifteen character columns is accomplished by typing:

/GC15 RETURN

We can start typing the labels in column 1. In order to enhance the readability it is nice to provide some indentation of the labels. If you attempt to do so simply by typing one or more spaces at the beginning of a label you wish to store, the spaces will be stripped off by the VisiCalc program. What are we to do? Recall the use of the quotation mark character ". We used it to let us enter numbers as labels, so the VisiCalc program wouldn't try to treat them as numbers. To enter the date 11/23 as a label, we typed it as "11/23 and found it stored as the label 11/23; the leading " was discarded, as we expected. You can use the " to force the VisiCalc program to accept a label with leading spaces. So by typing

```
>A3 RETURN
"  SAVINGS ACCT ↓
```

the two leading spaces in front of the word SAVINGS will be stored as part of the label and the leading character " will be discarded.

We can fill in the labels for column A by typing:

```
>A1 RETURN
ASSETS ↓ CASH ↓ "  SAVINGS ACCT ↓ ...
... ↓  PERSONAL PROP ↓ " FURNITURE ↓ ...
... ↓ "  PENSION ↓ ↓ TOTAL RETURN
```

We could have filled in the dashed line following PENSION instead of skipping directly to the TOTAL label, but it might be easier to follow if we do it separately. Assuming that PENSION was placed in A16, we can fill in the dashed line we want in A17 by typing:

```
>A17 RETURN
/-
-                    (repeat "-" as a label)
→                    (advance to column B)
/--                  (fill B17 with dashes)
RETURN
```

The formula for the total can go into position B18:

```
>B18 RETURN
@SUM(B1.B17) RETURN
```

If we subsequently wish to insert any new asset items above row 17, their values will automatically be included in the total. Using /IR to insert a new row at or above row 17 (presently occupied by our dashed line) will automatically adjust the argument range in @SUM above to include all of the old positions as well as the new one.

Our worksheet now has the following appearance (some numbers have been already entered in column B.

```
ASSETS
CASH
  SAVINGS                2000
  MONEY FUND            12000
REAL ESTATE
  HOME                 125000
SECURITIES
  STOCKS                32500
  BONDS                  2000
PERSONAL PROP
  FURNITURE              5000
  CAR                    1500
  CLOTHING               2000
LONG-TERM
  INSURANCE              1800
  PENSION               24000
---------------------------
TOTAL                  207800                <C11Q>
```

We can proceed to prepare the liabilities part of the worksheet in the same way. If we choose to have that part to the right of the assets portion, say in columns C and D, we may later have an annoying problem on our hands. Each time you wish to insert a new row for some asset you had forgotten, or each time you need a new row to account for an yet another liability, both sides of your worksheet will be affected. When you insert a new row, it cuts across the entire worksheet. You can avoid this problem by setting up your liabilities segment below the assets segment.

It would not matter which columns you chose as long as the first row is below that of the assets' TOTAL row (presently row 18). So let us choose to place the liabilities segment in columns A and B. This has the advantage that we can print the whole worksheet with a single /PP command and not have any cutting and pasting to do. Leaving row 19 blank, we can fill in the liabilities segment by typing:

```
>A20 RETURN
LIABILITIES ↓ BILLS ↓ " CHARGE ACCT ↓
" MEDICAL ↓ MORTGAGES ↓ ...
...
" OTHER ↓ /-- ↓ TOTAL RETURN
```

Here we chose to fill in the dashed line as we entered the labels. We can extend the dashed line over into column B by repositioning the cursor over the position with the dashes and replicating it into the position on its right:

```
↑          (up to dashed entry)
/R         (replicate)
RETURN     (use present location as source)
→          (move right to identify target)
RETURN     (target range)
```

We now have dashed lines extending from column A through column B. It might have been simpler to type /-- to fill in the B column entry but what we have seen shows how you can use the cursor arrows to identify coordinates instead of typing them in. The formula for the sum of the liabilities can now go into position B36.

We can complete our worksheet by repeating the totals at the bottom in rows 38 and 39 and calculating their difference in row 41. To obtain the following worksheet:

ASSETS	
CASH	
SAVINGS	2000
MONEY FUND	12000
REAL ESTATE	
HOME	125000
SECURITIES	
STOCKS	32500
BONDS	2000
PERSONAL PROP	
FURNITURE	5000
CAR	1500
CLOTHING	2000
LONG-TERM	
INSURANCE	1800
PENSION	24000
TOTAL	207800
LIABILITIES	
BILSS	
MEDICAL	3400
CHARGE ACCT	1850
MORTGAGES	
HOME	39600
COTTAGE	15800
TAXES	
FEDERAL	6000
STATE	890
PROPERTY	1900
LOAN PAYMENTS	
CAR	1200
COLLEGE	8500
OTHER	2000
TOTAL	81140
ASSETS	207800
- LIABILITIES	81140
NET WORTH	126660

<C11R>

we typed:

```
>A38 RETURN
" ASSETS ↓ "-LIABILITIES ↓ /-= ↓
NET WORTH RETURN
```

for the four bottommost lines of column A, and

```
>B38 RETURN
+B18 ↓                  (copy ASSETS total)
+B36 ↓                  (copy LIABILITIES total)
/-= ↓                   (fill in ='d line)
+B18-B36 RETURN         (difference)
```

You can now insert new asset lines or new liability lines as desired, using /IR to insert a new row immediately above the row you select with the cursor. Similarly, you can remove any line which is no longer needed by typing /DR. You can update any values that have changed just by placing the cursor on the outdated value and typing in the new value. The appropriate total and the net-worth figure will automatically be updated.

If you feel you should want to rearrange the worksheet, say by moving the "CHARGE ACCT" line, row 23, so that it would precede rather than follow the "MEDICAL" line, row 22, you can do so by using the move command /M. Type

```
>A23 RETURN             (select row to be moved)
/M                      (see prompt MOVE:FROM...TO)
↑                       (move cursor to desired row)
RETURN
```

This will remove row 23 and place it in front of old row 22. All rows at or below the new row 22 will be renumbered and all value references to items within them will be adjusted so the results don't change. We can see the "before" and "after" views of the surrounding rows:

Before moving row 23:

```
21        BILLS
22          MEDICAL            3400
23          CHARGE ACCT        1850
24        MORTGAGES
```

After moving row 23 to a position preceding old row 22:

21	BILLS	
22	CHARGE ACCT	1850
23	MEDICAL	3400
24	MORTGAGES	

SUMMARY

Computing your net worth uses @SUM for two subtotals: one for the assets and the other for liabilities. In a worksheet with two essentially independent segments which may vary in size from one month to the next (that is, you may wish to insert new information in one part without changing the other part), it is better to have one segment below the other rather than side by side.

If wish to enter titles with leading blanks or spaces, first type a quotation mark ". The quotation mark will be discarded but the leading spaces will be retained.

Moving a row (or a column) can be done using the move command /M. The move command makes it possible to rearrange a worksheet without having to start all over again.

Case Study D: Sales Projection

Suppose you are selling cars and trucks and wish to project your future sales based on your past sales. After all, is there any better indicator? Depending on your outlook regarding recessions and depressions you might prefer using some other indicator. In any case, here is how you could proceed.

You have been in business for three years, so your dollar sales per year record to date is:

	1978	1979	1980
cars	42323	51891	65123
trucks	45671	46128	49088

You would like to calculate the growth rate for the sales, obtaining the average yearly compounded rate. What you would really like almost fills the rest of this page. How do we produce this report?

	-ACTUAL-			GROWTH		TOTAL
	1978	1979	1980	RATE	AVERAGE	(000's)
CARS	42323	51891	65123	24.04	53112	159.34
TRUCKS	45671	46128	49088	3.67	46962	140.89
TOTAL	87994	98019	114211	13.93	100075	300.22
% CARS	48.10	52.94	57.02	8.88	52.69	158.06
% TRUCKS	51.90	47.06	42.98	-9.00	47.31	141.94
TOTAL	100.00	100.00	100.00		100.00	300.00

-PROJECTED-				
1981	1982	1983	1984	1985
80782	100206	124300	154188	191262
50891	52761	54699	56708	58791
131673	152966	178999	210896	250053
61.35	65.51	69.44	73.11	76.49
38.65	34.49	30.56	26.89	23.51
100.00	100.00	100.00	100.00	100.00

<C11S>

We will build up this report one step at a time, not worrying about any formatting for the time being. We can begin by building the top half of the report first, starting from the left. We can also defer some of the headings without creating more work for ourselves.

If you start with Year 1's sales being S, and let the unknown yearly average interest rate be i, then for Year 2 and Year 3, you would expect:

Year 2 sales are $S*(1+i)$
Year 3 sales are (Year 2 sales)$*(1+i)$, or

$S*(1+i)*(1+i)$, or
$S*(1+i)^2$

We can solve for i since the total increase in sales, T, is known to us and it must also be equal to

$$S*(1+i)^2 - S = T$$

$$S*(1+i)^2 = T + S$$

$$(1+i)^2 = (T+S)/S$$

$$= (\text{Year 3 sales})/S$$

$$(1+i) = \text{square root of } (\text{Year 3 sales}/S)$$

If we begin to lay out our worksheet as follows:

	A	B	C	D	E
1		1978	1979	1980	
2	CARS	42323	51891	65123	
3	TRUCKS	45671	46128	49088	<C11T>

we can rephrase our solution for i in terms of our coordinates. Assuming we want the growth rate for car sales in E2, type

```
>E1 RETURN
RATE ↓
( @SQRT( D2 / B2) -1 ) )*100 RETURN
```

The *100 converts the fraction into a percentage. Then we can replicate this for the trucks and assuming we want the growth rate for total sales, we can fill in row 4 as follows and then replicate the growth rate from E2 into E3 and E4. Doing the replication first would also work, except you might be surprised by seeing the result ERROR in positions E4 due to not having any values yet for B4.

```
>A4 RETURN
TOTAL > @SUM(B2.B3) RETURN
/R                  (now relicate the formula)
RETURN              (use B4 as source)
C4.D4               (range is C4 to D4)
RETURN
R                   (want coordinates relative)
R
```

and now we can replicate the rate formula from E2 into E3 and E4:

```
>E2 RETURN
/R
RETURN
E3.E4 RETURN
R
R
```

and our worksheet now looks as follows:

```
      A            B            C        D        E
1                1978         1979     1980     RATE
2  CARS         42323        51891    65123    24.04
3  TRUCKS       45671        46128    49088    3.667
4  TOTAL        87994        98019   114211    13.93 <C11U>
```

The average we want in the next column, column F, is just the sum of the yearly sales divided by the number of years; we can use the @AVERAGE function for this as follows:

```
>F1 RETURN
AVERAGE ↓
@AVERAGE(B2.D2) RETURN          (for F2)
/R
RETURN                          (replicate F2)
F3.F4 RETURN                    (into F3 and F4)
R                               (relative coordinates)
R
```

The next column is the total sales in thousands of dollars. We can set it up as follows:

```
>G1 RETURN
"(000's) ↓                      (need " here)
@SUM(B2.D2)/1000 RETURN
/R
RETURN
G3.G4 RETURN
R
R
```

We need the " to begin entering the label (000's), otherwise it will be taken as an arithmetic expression and since it is not a valid arithmetic expression, it will be rejected.

The next five columns are the sales projections we want calculated. Assuming that the average growth rate for sales is maintained and that it compounds yearly, if we have sales S now we expect S*(1+i) next year, and S*(1+i)*(1+i) the following year and so on. We can set this up for cars, in year 1981, and replicate it horizontally. Unfortunately, it won't quite work out that way.

Let us provide the projected sales for cars, in position H2:

```
>H2 RETURN
+D2 * (1 + E2/100)      oops!- wrong! Push CTRL-C to cancel
+D2 * (1 + (E2/100) )
RETURN
```

The rate in E2 is a percentage so we have to divide by 100. But if we write 1+E2/100, we will compute (1+E2)/100, which is not our intention. So we have to use another pair of parentheses to force (E2/100). If we now try to replicate this formula for the next few years of sales projections we run into a trap. All of the years for a given row should use the same growth rate, from E2, and we easily get this by responding with N, for not-relative, if we use replication. But what of D2? Clearly we want that to change, so we can't respond with N if we replicate this formula. But if we respond with R we would get, for position I2:

```
+E2*(1+(E2/100))
```

The D2 from our formula for position H2, if it is identified as relative when we replicate H2 into the next position I2, will be converted to E2; but E2 is not the prior-year sales! Position E2 holds a growth rate. We simply have to expect that replication with relative coordinates only works well when you are dealing with contiguous rows or contiguous columns. Unfortunately sales for the year 1980 and those projected for 1981 are separated by three other columns.

So, let's not replicate the H2 formula. We can type in the one we need for I2, and replicate it without any problems.

```
>I2 RETURN
+H2 * (1 + (E2/100) ) RETURN
/R
RETURN
J2.L2 RETURN
R                          (want H2 relative)
N                          (keep E2 as-is)
```

You can now replicate H2 down its column, to define H3 and H4, but you may be in for a nasty surprise. H4 is supposed to be the total projected sales for 1981. You would expect the sales breakdown for that year to add up to the total. But if you compute H4 using

```
+D4 * (1 + (E4/100) )
```

the result is 130,117 which is not the total of 80,782 and 50,891 (total of 131,673). We seem to be off by 1,556, which is about a one percent error. When there are two ways of deriving a result, you might expect some discrepancy, especially when functions such as square root are involved. Calculating the total projected sales figure by using the compounding formula will lead to the totals not agreeing. It would be better in this instance to use a simple sum and avoid the problem. So let us use the compounding

formula for the trucks row and a simple sum in the totals row. We can fill in column H as follows:

```
>H3 RETURN
+D3 * (1 + (E3/100)) ↓
@SUM(H2.H3) RETURN
```

This last entry can be replicated for the remaining totals projections:

```
>H4 RETURN        (want to be at H4, if you were not)
/R
RETURN
I4.L4 RETURN
R                 (want both coordinates relative)
R
```

We can proceed as before to fill in the remaining projections for truck sales.

```
>I3 RETURN
+H3 * (1 + (E3/100) ) RETURN
/R
RETURN
G3.L3 RETURN
R
N
```

We have now completed the top half of our report, save for some labels and the formatting. The bottom half consists almost entirely of percentages. It is only the two growth rate items that break up this nice pattern. That being the case, we can fill in all of these positions as if they were all percentages, dealing with their relative columns using replication. Then we can go back and patch up the few exceptions. We can start with labels for A6, A7, and A8:

```
>A6 RETURN
"% CARS ↓ "% TRUCKS ↓ TOTAL RETURN
```

Then we can compute percentages for B6 and C6:

```
>B6 RETURN
+B2/B4*100 ↓
+B3/B4*100 RETURN
```

and we can replicate these two formulas simultaneously into positions C6 through L6, and C7 through L7, respectively. But we should first fill in the bottom formula and then we can replicate all three simultaneously.

```
>B8 RETURN
@SUM(B6.B7) RETURN
>B6 RETURN
/R
```

```
.B8 RETURN                    (source is B6, B7, B8)
C6.L6 RETURN                  (target is C6 to  L6 and
                              C7-L7, C8-L8 are implied)
R                             (both coordinates in first
R                             formula are relative)
                    (C6 to L6 have now been filled in)
R                             (both coordinates in second
R                                   formula are relative)
                    (C7 to L7 have now been filled in)
R                             (both coordinates in last
R                                   formula are relative)
                    (C8 to L8 have now been filled in)
```

We can now amend positions E6, E7 and E8. Position E8 is easy to fix; we just want it blank so we type /B RETURN. Positions E6 and E7 reflect the rate of growth of the sales percentages. We have been grossing more from car sales at the expense of truck sales, so we should expect the first rate to be positive and the next to be negative. We could either replicate an existing rate formula from above, also in column E, or simply type them in again, as we show here:

```
>E6 RETURN
(@SQRT(D6/B6)-1)*100 >        (formula for E6)
(@SQRT(D7/B7)-1)*100 >        (formula for E7)
/B RETURN                     (erase E8)
```

Formatting This Report

We have now defined everything that needs to be calculated, so we should now set out to make the report more presentable by formatting it. We can begin by saving the worksheet, say in a file called "DATA.VC" by typing

```
/SS
DATA.VC RETURN
```

We can now examine our screen and see what needs to be done. For the numeric entries, it seems that they should either be displayed as integers (for sales projections), or using two-digit fractions (for percentages). We could start out by forcing all numbers into the $ format, since that provides the two-digit fraction representation. So we type

```
/GF$    (set global format $)
```

Depending on how we typed in the years 1978, 1979, etc., if these were typed in as labels (by typing "1978, "1979, etc.) then they will not be affected. If they were typed in as numbers, they will be affected by the /GF$ and appear as 19.78, 19.79, etc. In that case, instead of reentering these items as labels you can assign "I" formats to them, using /FI for each of the positions.

Having taken care of all the percentages, we can make the sales and sales projection formats all "I" by assigning /FI to B2, B3, and B4, and then replicating it across as far as column F, by typing:

```
>B2 RETURN
/FI ↓              (assign I format to B2, advance to B3)
/FI ↓              (assign I format to B3, advance to B4)
/FI                (assign I format to B4)
>B2 RETURN         (return to B2)
/R                 (replicate  from B2 to B4)
B2.B4 RETURN       (use B2 to B4 as source)
C2.L2 RETURN       (replicate spanning columns C to L)
```

Since B2-B4 only contain the numbers we typed in, there will not be any prompts for N or R as is usually the case with replication. In copying our I format, we will have destroyed numbers and formulas in columns C to L. We can recover these later. We also reset the growth rate results in column E and G, rows 2, 3, and 4. Since we do want these in $ format, you can restore the format by typing

```
>E2 RETURN
/F$ ↓ /F$ ↓ /F$
>G2 RETURN
/F$ ↓ /F$ ↓ /F$
```

We now have the formatting as we like it, so let us reload the correct data and formulas by typing:

```
/SL DATA.VC RETURN
```

We now have almost exactly the report that was shown in the first page of this case study. There are just a few loose ends we must attend to.

We are missing the few labels that should sit over the whole report. We can make room for a new row 1 by typing:

```
>A1 RETURN
/IR                (insert row)
```

To obtain a label that spans more than one column, such as -PROJECTED-, type the left nine characters of it in I1, as -PROJECTE (do remember to type this as "-PROJECTE, because if you leave off the leading ", entry of -PROJECTE will be rejected as a badly-formed formula!), and type the remainder of it in J1, as D-. If you change the global column width using /GC, you will have to fiddle with these extra-long labels.

Now that you have the worksheet in the shape you want, you should save a copy of it, say by typing /SS SALESPR.VC, and you can print a copy by using /PP. If your printer cannot accommodate such a wide report, print it out in two segments, as we did back at the start of this case study. We printed the segment defined

by the corners A1 and F9, then we printed the segment defined by the corners G1 and L9.

Another Approach

If you had to do this sales projection all over again (say you lost the file you just saved, by spilling coffee on your diskette or by writing on it with a hard pen), you could in the light of our previous discussion save some typing. By planning ahead, which is always easier after a bit of experience, we can see that if we assigned formats as we defined formulas that were going to be replicated, we could make one replication do the work of two.

For this report, one could begin by assigning the global format "I" since we won't be replicating into the actual-sales positions. Then we can assign /F$ formats as we proceed to type formulas.

A Word of Caution

Reports generated by a computer seem to carry a lot of weight. The press often presents stories with the phrase "A computer analysis of ... shows that ..." We just finished a computer forecast (a sales projection) and we should take it with some skepticism. Suppose you clear the screen and load in the simplest part of the sales projection, the one corresponding to worksheet C11U. Now change the car-sales figure for the second year (it is the 51891 in position C2 of worksheet C11U). Change it to almost double, say 100000. The growth rate of 24.04% in E2 does not change! Try a sales figure that is almost half of the old one. Typing 25000 into C2 still leaves you with a growth rate of 24.04%. What is going on? Recall how we originally formulated the problem. The growth-rate formula we derived depends only on the difference between the first-year and final-year sales. This is exactly how the worksheet is behaving. If you use simple-minded formulations, you will get the corresponding results. The computer should neither get any credit or any blame.

Obviously you would like a more sensitive projection formula. The simplest approach that takes into account intervening years involves computing the year-to-year sales increments and deriving the year-to-year growth rates. Then an overall growth rate can be calculated as the average of the year-to-year rates. If we had used this approach, then using original sales figures, the final growth rate for car sales would have been 24.05% instead of 24.04%, an insignificant difference. However if the sales history had been more erratic, say the second-year car sales had been 30,000 instead of 51,891, then our reformulation would produce an average growth rate of 44%, which is significantly different from 24.05%. If you pursue the art of using the past as a guide to the future you will start looking into time-series

analysis and other sophisticated statistical methods which are beyond the scope of this book.

The moral of the story is: the computer is a useful tool and a computer equipped with the VisiCalc program is an even more useful tool, but tools are not a substitute for thinking through what it is you really want to use these tools for.

SUMMARY

The work of defining a complicated report can be made simpler by treating the formulas and the formats independently of each other, thanks to the use of files.

The typing effort in preparing a complicated report can be reduced by careful planning, so that as formulas are replicated, the appropriate formats are also replicated.

Care must be taken to ensure that results labeled as "totals" are in fact computed so that they represent correct totals.

Computer projections of the future should be treated with a great deal of skepticism.

Case Study E: Interest Computations

When looking into an investment possibility, you need to have a good idea as to what it might cost you to borrow the money to make the investment. You should know whether or not you could make the monthly payments on a new loan. You would also like to have some idea of how agressively you should negotiate for a better rate. What difference will a 0.5% or 1% per year difference in the loan rate make to you in terms of monthly payments? You probably have the same problem I have, of not being able to find the interest tables when you need them. It is even more frustrating to find the interest tables that cover every rate from 0.5% to 10% in steps of .5% when you need to investigate possibilities in the 16 to 20% range.

In such a case, it is nice to have a few VisiCalc worksheets available so you can create the tables you need when you need them.

Monthly Payments

Suppose you want to borrow an amount A, which you wish to repay in n equal monthly payments P, at a monthly interest rate of i%. Your monthly payments are then given by the formula

$$P = A * i / (1 - (1+i)^{-n})$$

The operation x^m, which is called exponentiation, is read as "x to the power m." It means that x is to be multiplied by itself m times. The symbol used by the VisiCalc program to denote exponentiation is ^ (the character obtained by typing SHIFT N). We can set up a worksheet to compute these monthly payments as follows:

use column A for the desired amount A
use column B for the monthly interest rate i
use column C for the loan duration in months
use column D to display the resulting monthly payment P

We can set up titles in row 1:

```
>A1 RETURN
AMOUNT > INT > MONTHS > PAYMENT RETURN
```

Then we can enter the data in row 2. For a $100 loan at 12% per year, compounded monthly, we would enter 1% (12% divided by 12 months) as the interest rate. If we intended to pay it off in one month we could fill in row 2 as follows:

```
>A2 RETURN
100 > 1 > 1 >
```

We should now enter the payment formula in position D2:

```
>D2 RETURN       (place formula in D2)
+A2*B2*.01/(1-((1+(B2*.01))^-C2)) RETURN
```

The worksheet should display the following items:

```
          A        B          C          D
1         AMOUNT   INT        MONTHS     PAYMENT
2         100      1          1          100.9999    <C11V>
```

You would expect the payment to be 101.00. The VisiCalc program produces the result 100.999903, which is tolerable. The use of the exponentiation operation ^ introduces greater inaccuracy than is usually the case. You can of course ask that the results be displayed to the nearest penny by specifying the $ format for position D2, by typing /F$.

It seems that we have many more parentheses in the D2 formula than we had in the original formula. Let us place them next to each other. The original formula was:

$$P = A * i /(1 - (1+i)^{-n})$$

The translation of this formula for use with the VisiCalc program is:

```
+A2*B2*.01/(1-((1+(B2*.01))^-C2))
```

Why do we need so many parentheses? Let us number them in pairs, so we can explain why each pair is required.

```
+A2*B2*.01/(1-((1+(B2*.01))^-C2))
           1 23  4      43    21
```

The need for the pair of parentheses numbered 1 is clear; those parentheses come from the original formula. Pair number 4 is needed to prevent B2 from being added to the 1 on its left. Pairs number 2 and 3 prevent "1+(B2*.01)" from being subtracted from the 1 on its left, until it has been exponentiated by the power -C2. It is so easy to make a mistake in typing a lengthy formula that you should always have some very simple test case on hand to verify that the result produced by the formula is reasonable.

In this case, two simple tests can be applied. If you borrowed zero dollars, you should expect the payment to be zero. Setting A2 to zero should cause D2 to become zero. The other simple test is: for a $100 loan at 1% per month, payable in one month, you would expect to pay $100 plus 1% of $100, for a single payment of $101. Test by setting A2, B2, and C2 to 100, 1, and 1 respectively; these values should cause D2 to display 101.

Interest Tables

You may have noticed that the monthly payment for a loan of amount 2*A is exactly twice that for a loan of amount A, given the same terms. So we can simplify things by considering a unit loan, where A is always 1. Then if you need to know what the payment would be for any particular size loan, you can look up the unit payment for the desired interest rate and period of time, and multiply that amount by the loan amount. The payment formula reduces to

$$P = i / (1 - (1+i)^{-n})$$

To make things simpler, we can modify this formula so that it can work with annual interest rates and always specify the loan period in years. Since we still want monthly compounding the revised formula becomes:

$$P = (i/12) / (1 - (1+(i/12))^{-(n*12)})$$

Suppose you want a table showing the payment schedule for a five year unit loan, for interest rates 10%, 11%, ..., 14%. We can enter titles in A1 and B1:

```
>A1 RETURN
RATE > PAYMENT RETURN
```

We can enter the desired interest rates in A2, A3, etc. by typing them in or, as we show here, by using replication so that we can then easily change the interest rates by changing the lowest of the rates.

```
>A2 RETURN
10 %              (10% in A2)
+A2+1             (compute 10% +1, for A3)
RETURN
/R                (replicate A3 formula)
RETURN
A4.A6 RETURN       (fill in A4 to A6)
R                 (want +A2 relative)
```

The payment formula can be placed in B2 and replicated through B3, B4, ..., B6:

```
>B2 RETURN
.01*A2/12/(1-((1+(.01*A2/12)^-(5*12))))
RETURN
/R
RETURN
B3.B6 RETURN
R                      (want A2 relative)
R                      (want A2 relative)
```

We can see the fruits of our labor below:

```
     A          B
1    RATE       PAYMENT
2    10         .0212470
3    11         .0217424
4    12         .0222444
5    13         .0227531
6    14         .0232683                          <C11W>
```

It would be convenient to be able to easily obtain such a table for periods other than 5 years. This can be achieved by using some position, say C2, to hold the desired number of years and then changing the last part of our formula, the (5*12), into (C2*12). While we are at it, we can go further, as the next section discusses.

Multi-Year Interest Tables

It would be useful to have a table showing the monthly payments for both a range of interest rates and a range of loan periods, as in:

```
     A     B          C          D          E          F
1                   TERM IN YEARS
2  RATE     1          2          3          4          5
3    10   .0879      .0461      .0323      .0254      .0212
4    11   .0884      .0466      .0327      .0258      .0217
5    12   .0888      .0471      .0332      .0263      .0222
6    13   .0893      .0475      .0337      .0268      .0228
7    14   .0898      .0480      .0342      .0273      .0233 <C11X>
```

We can build such a table by using our previous formula and making good use of replication. Let us begin by setting the column width to 6 and placing a title in positions C1 and D1.

```
/GC6                    (set column width to 6)
>C1 RETURN
TERM I>N YEAR>S RETURN
```

Then we can initialize column A as follows:

```
>A2 RETURN RATE ↓
10 ↓                    (place 10 in A3)
+A3+1                   (place formula in A4)
RETURN
/R
RETURN                  (replicate A4)
A5.A7 RETURN
R
```

This sets up the rate column with the values 10, 11, ..., 14. Merely by changing the 10 in A2, say to a 16, our table would then reflect the rates (and corresponding payments) for 16%, 17%, etc. The loan period can similarly be set up in row 2, from B2 to F2:

```
>B2 RETURN
1 >          (place a 1 in B2, advance to C2)
+B2+1        (place formula in C2)
RETURN
/R           (replicate it)
RETURN
D2.F2
RETURN
R            (relative)
```

Once again it is worth noting that the loan period or term range can be changed from the 1-to-5 we have here merely by changing the first value, the 1, into the starting value for a new range of years. The other term periods will be calculated by repeatedly adding one. If you wish, you can also change the formula for C2. It now increases successive values by one; you could change that to steps of 0.5 or any other desired value.

Finally we can fill in a payment formula. We can proceed as follows:

```
>B3 RETURN              (payment formula)
.01*A3/12/(1-((1+(.01*A3/12)^-(B2*12))))
RETURN
```

It is hard enough to type the parentheses correctly once; imagine how difficult it would be to type this formula once for each of 5 terms and then do that again for each of 5 interest rates. You would type it 25 times to fill in the table we want. Fortunately we need only type it in once (correctly) and then use replication to create the other 24 copies, suitably altered. We can start by replicating the payments formula across row 3:

```
/R           (assuming we are at B3, replicate
RETURN        the payment formula)
C3.F3        (fill in C3 to F3)
RETURN
N            (use rate in A3)
N            (use rate in A3)
R            (use term in new column)
```

This fills up positions C3 to F3 with the appropriate payment formulas and displays the results in those positions. We can now replicate all of the formulas in row 3, from positions B3 to F3, into the adjacent rows 4, 5, on up to row 7, as follows:

```
>B3 RETURN
/R
B3.F3        (replicate B3, C3, ..., F3)
RETURN
B4.B7        (into the next 4 rows)
RETURN
R            (use A3 for rate)
R            (use A3 for rate)
```

```
N         (use this column's term)
          (completes filling in col B)
R         (use A3 for rate)
R         (use A3 for rate)
N         (use this column's term)
          (completes filling in col C)

    ...   (for cols. D, E, ...)

R         (use A3 for rate)
R         (use A3 for rate)
N         (use this column's term)
          (completes filling in col F)
```

SUMMARY

We have finished building the multiyear loan payment table we showed at the beginning of this section. It has been constructed so that the interest rate ranges and term ranges can easily be changed. Similarly, you can extend the table to cover many more term periods or a wider range of interest rates, or both, simply by specifying a wider range when replicating the factors involved.

Chapter 12

WHEN TO AVOID VISICALC

The VisiCalc program is a marvelous tool when dealing with numbers in a situation where you have to "cut and try." You could, however, get carried away and become so infatuated with this tool that you find yourself creating work rather than reducing it, because you are using a good tool for the wrong job.

Consider the worksheet we set up some time ago to maintain an expense log, with expenses divided into two categories:

Date	Item	Category1	Category2
1/23	battery		55
1/29	suit	125	

What if we wanted an expense journal with a dozen or more categories? It might look like:

Date	Item	C1	C2 ...	C12
1/23	battery		55	
1/29	suit	125		
2/2	heat			197
...				

It would be a nuisance to have to skip over so many columns to fill in new entries. What if you wanted to set up a table using four columns, with a code for each category, as in:

Date	Item	Category	Cost
1/23	battery	B	55
1/29	suit	A	125
2/2	heat	L	197
...			

There is no way to ask the current version of the VisiCalc program to do the following:

(1) if an item has category code A, then add its cost to the total for Category-A
(2) if an item has category code B, then add its cost to the total for Category-B
(3) ... similarly for categories C, D, ...

Could we not get the same effect by using numeric category codes 1, 2, 3, ... instead of codes A, B, C, ...? Not with this version of the VisiCalc program.

The inability of having the VisiCalc program perform according to "if this situation is in effect, then do that" <u>in all its generality</u> separates it fundamentally from the high level computer programming languages such as BASIC, COBOL, FORTRAN, PASCAL, etc. They all include the decision-making ability implied by an "if this, then do that" capability. Note that the

@IF function does not solve this problem. The @IF lets you choose one of two values--it does not let you choose where the result will be placed (@CHOOSE has a similar constraint). This inability of the VisiCalc program is both a weakness and a virtue. We can see how it limits you, but we should appreciate that not having this ability presents a tremendous simplification. You may already feel that the VisiCalc program has too many rules; you need only examine books which describe programming in languages such as BASIC or FORTRAN and you will be amazed at how much more complicated life can get.

You can still sometimes get the best of both worlds. Take the example we just saw. The VisiCalc program makes it so easy to fill in the entries that we might want to use the VisiCalc program just to do that: type in expense items. Then we can save the expense worksheet in the DIF format (remember /S#S). Whenever we need or want a current total for each of the cost categories A, B, C, ..., or categories 1, 2, 3, ..., we can use the Apple II's disk operating system DOS to run a program we have had written for this purpose. This program will process our DIF-formatted file without changing it. This program will display and/or print the category totals we need.

Other Problems

Some data processing applications require more print control than the VisiCalc program can provide. For instance, you might have to prepare a report in which the first column uses five characters, the second column uses 13 characters, the third column uses eight characters, and so on. The only control we have over column width is provided by /GC. It allows you to set at most two column widths, and then only if you happen to be using two windows. Even then, when you come to print a worksheet, only a single common column width will apply, regardless of your use of windows.

The most serious problem we encounter reflects the way the world works. Most things we deal with require a lot of information. In computer jargon, decision-making requires many inputs. Being very specific, much of the data you want to base your decisions on is accumulated in different computer files. You may have an accounts receivable file, an accounts payable file, a customer name-address file, a year-to-date sales file, etc. There is no easy way to extract the desired data from multiple files to send on to a VisiCalc worksheet. You would have to find or write a program to do this. The simplest method is for you to do a lot of thinking about what really matters. It has to matter so much that you are willing to type the required data in.

That may be the best criterion for deciding whether or not some problem is worthy of being attacked with the VisiCalc program. If it is worth your time to type it in, then the VisiCalc program may be the tool to use, and then the search for a program to help with the use of existing data files will certainly

be warranted. Should no such program turn up, you can then consider writing one yourself or contracting for one.

If you can find some way of letting another program provide the data you need for cogitating with the VisiCalc program, or create a program that will extract the data you need from existing files for the same purpose, then you can avoid all the data errors that sneak in due to excessive data transcription and you can use some of the time that has been saved to do some more thinking, so you can concentrate on making the right decisions.

Other VisiCalc Program Versions

It is comforting to know that almost everything we have seen in this book applies verbatim to the versions of the VisiCalc program which are available for other personal computers.

It is conceivable that new versions of the VisiCalc program will evolve, most probably as supersets of what we have seen. So it should not be necessary to unlearn or undo anything--it becomes a matter of expanding the scope of things that possible new versions might let us tackle.

The background you have developed by working this far should make it easy for you to learn how to use these enhancements. Happy computing.

Chapter 13

SUMMARY OF VISICALC COMMANDS

The / Command

/ --> prompt COMMAND: B C D E F G I M P R S T U V W -

B	blank	erase value/label, 50
C	clear	clear whole worksheet, 26
D	delete	delete a row or column, 44
E	edit	edit an entry without retyping it, 51
F	format	see Format on next page, 68, 93
G	global	see Global on next page, 41, 93
I	insert	insert a row or column, 45
M	move	move a row or column, 54, 132
P	print	print segment of worksheet, 29, 59
R	replicate	replicate (copy), 32, 98
S	storage	see Storage on next page, 18, 102
T	title	freeze rows or columns in place, 61
V	version	copyright notice
W	window	split screen operations, 63
-	repeating label	fill a position with a label, 117

Format Command /F

/F --> prompt FORMAT: D G I L R $ *

D default format as specified by /GF, 70, 98

G general format (labels left, numbers right), 94

I integer format (rounded), 94

L left-adjusted, 95

R right-adjusted, 93

$ dollar format; uses units of .01, 95

* graphics format, 68

Global Command /G

/G --> prompt GLOBAL: C O R F

C column width, 71

O reevaluation order (normally by column, from A1), see the discussion in the VisiCalc reference manual.

R recalculation (automatic/manual), 83

F format (see Format Command, above)

Storage Command /S

/S --> prompt STORAGE: L S D I Q #

L load a file into a worksheet, 22, 102

S save a worksheet as a file, 21, 102

D delete a file, 111

I initialize (format) a diskette, 20

Q quit, leave the VisiCalc program, 18

load/save a file in the Data Interchange Format DIF, 101

Appendix: Using the Optional Diskette

The optional diskette which is being made available is a data-only diskette. This optional diskette assumes you have already purchased the VisiCalc program for the Apple II Computer.

If you follow the instructions in this book carefully, you will have constructed many VisiCalc worksheets. Some of these you may wish to save for future use (such generally-useful worksheets are often called "templates"). In order to reduce the time you might have to spend typing values or labels or formulas, each of the significant worksheets we discuss, as well as those used to try out some command or some technique, are being made available on an optional diskette. The worksheets available on the diskette are identified in the text by the notation "<worksheet name>." Each worksheet has a name of the form

Chapter number Upper-case letter

where the chapter number is always preceded by the letter C. Thus <C4A> identifies the first such worksheet, from Chapter 4, (C4), on page 27.

Loading a Worksheet

All the worksheets were prepared using the /SS command, so each one of them can be loaded using the /SL command. Keep in mind that loading a worksheet does not erase those entries on your screen that are not being affected by the new worksheet. So you should precede each /SL command with a /CY command to clear the screen before loading a new worksheet.

Load the VisiCalc program in the usual way (see chapter 3 if you don't remember how this was done) and place the optional diskette in disk drive 1. To load worksheet C4A, type the following:

/SL C4A RETURN

For most worksheets the loading is very quick. For a few (e.g., C11X, multi-year interest tables) it seems to take much longer (about thirty seconds) even though the resulting worksheet has only some forty entries. Why does it take so long? You have to keep in mind that the file which represents a worksheet does not contain any calculated results. Those files contain the formulas used to calculate the desired results. So whenever you load a worksheet, all the formulas are evaluated. In the case of C11X, that represents a substantial amount of computing.

If you followed these directions carefully, you should see the following information displayed within a few seconds:

RATE	COPIES	INCOME	C4A
6	3000		
8	4000		
9	5000		
10	38000		

Each worksheet includes its name in some location, so you can be certain that you loaded what you wanted to load. The name will usually appear in the top right position; sometimes it will appear on the bottom left.

Diskette Content

All the worksheets of general use are included (e.g., IRA, portfolio evaluation, net worth, sales projection, interest calculations, etc.). The worksheets on the optional diskette and the subject matter are:

C4A-C	Author-Income
C4D-E	IRA
C5A-K	Gradebook-alias-Production/Sales Records
C6A-D	Handling big worksheets
C7A-E	Precipitation graphs
C7F-G	Market graphs
C7H	IRA graph
C8A-B	Using trig functions
C8C	Using @CHOOSE to handle coded items
C8D	Using @IF for a payroll
C9A-E	Gradebook-alias-Production/Sales Records
C11A-H	Case study A (expense log)
C11I-P	Case study B (portfolio evaluation)
C11Q-R	Case study C (net worth)
C11S-U	Case study D (sales projection)
C11V-X	Case study E (interest calculations)

The optional diskette may be helpful for those of you who can't spare the time to do any avoidable typing, or those of you who will do anything to avoid typing. Happy computing!

INDEX

Introducing. . . An exciting new breed of Computer Guidebooks designed to help you work SMARTER instead of HARDER:

The MICROCOMPUTER POWER Series

Four fascinating Guidebooks packed with practical, usable **TECHNIQUES—plus 208 BASIC PROGRAMS** to make your personal computer really Valuable!

Microcomputer Power

Series by John P. Grillo and J. D. Robertson

GUIDE TO SYSTEMS APPLICATIONS with Programs for TRS–80© Models I and III
61 programs/288 pages/$17.95

TECHNIQUES OF BASIC for the TRS–80© Models I and III
61 programs/272 pages/$18.95

INTRODUCTION TO GRAPHICS for the TRS–80© Models I and III
38 programs/144 pages/$15.95

DATA MANAGEMENT TECHNIQUES with programs for TRS–80© Models I and III
48 programs/208 pages/$16.95

And remember, you can 'preview' any or all four Guidebooks in your own home for a full 15 days—absolutely RISK-FREE!

wcb **Wm. C. Brown Company Publishing**
2460 Kerper Blvd., Dubuque, Iowa 52001

ORDER TODAY!

Complete Order Form at right and mail to:

Wm. C. Brown Co. Pub.
Direct Marketing Div.
2460 Kerper Blvd.
Dubuque, Iowa 52001

Gentlemen: Please send me the book(s) indicated. I understand that if I am not delighted with my selection(s), I may return it/them within fifteen days of receipt for a full and immediate refund—or that my charge card account will be credited for the full amount of any book(s) returned.

Total amount of order: $ _____

Tax (California residents add 6%) $ _____

Postage and Handling ($1.00 per book) $ _____

TOTAL enclosed/to be charged $ _____

Name _____

Street Address _____

City _____ State _____ Zip Code _____

Please check method of payment below

__ Check or Money Order Enclosed

__ VISA __ Master Charge M.C. Interbank No. __ __ __ __

Account No. .....

Exp. Date: _____ Signature: _____

POWER UP with VisiCalc® today!

A revolution in computer software, VisiCalc can make your microcomputer as easy to use as a calculator—with far more power!

To help you get into business with VisiCalc, Professor Edouard Desautels has specially prepared several electronic worksheets to accompany VisiCalc for Apple II Plus Computer. The worksheets are stored on a data-only diskette for use with the disk drive units of the Apple II Plus. Alternate diskettes, also prepared by Professor Desautels, are available for TRS-80 Model II, Model III, and the IBM Personal Computer.

At $39.95 it's an excellent way to learn and save valuable time while putting VisiCalc to work for you. The worksheets for use with the Apple II Plus cover:

- **Author—Income (C4A-C)**
- **IRA (C4D-E)**
- **Gradebook-alias-Production/Sales Record (C5A-J)**
- **Handling Big Worksheets (C6A-D)**
- **Precipitation Graphs (C7A-E)**
- **Market Graphs (C7F-G)**
- **IRA Graph (C7H)**
- **Using Trig. Functions (C8A-C8B)**
- **Gradebook-alias-Production/Sales Record (C9A-C9E)**
- **Case Study A. Expense Log (C11A-C11H)**
- **Case Study B. Portfolio Evaluation (C11I-C11P)**
- **Case Study C. Net Worth (C11Q-C11R)**
- **Case Study D. Sales Projection (C11S-C11U)**
- **Case Study E. Interest Calculations (C11V-C11X)**

Mail to:
Wm. C. Brown Co. Pub.
P.O. Box 539
Dubuque, Iowa 52001

☐ **YES, please send me the diskette containing specially prepared worksheets to accompany VisiCalc for the Apple II Plus Computer, by Edouard Desautels. I understand that if I am not satisfied with the diskette, I may return it within 30 days without further obligation.**

Diskette	**$39.95**
Tax (California residents add 6%)	**$_____**
Postage and handling	**$_____**
Total amount	**$_____**

Please indicate method of payment:

☐ **Check/Money Order enclosed (WCB pays postage and handling)**
☐ **Charge my credit account:**

☐ Visa ☐ Master card

Account No. ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ MC Bank # ☐☐☐☐

Exp. Date ____ Signature _____
(required for all charges)

Name _____

Address _____

City _____

State _____ **ZIP** _____

Phone _____

POWER UP with VisiCalc® today!

A revolution in computer software, VisiCalc can make your microcomputer as easy to use as a calculator—with far more power!

To help you get into business with VisiCalc, Professor Edouard Desautels has specially prepared several electronic worksheets to accompany VisiCalc for Apple II Plus Computer. The worksheets are stored on a data-only diskette for use with the disk drive units of the Apple II Plus. Alternate diskettes, also prepared by Professor Desautels, are available for TRS-80 Model II, Model III, and the IBM Personal Computer.

At $39.95 it's an excellent way to learn and save valuable time while putting VisiCalc to work for you. The worksheets for use with the Apple II Plus cover:

- **Author—Income (C4A-C)**
- **IRA (C4D-E)**
- **Gradebook-alias-Production/Sales Record (C5A-J)**
- **Handling Big Worksheets (C6A-D)**
- **Precipitation Graphs (C7A-E)**
- **Market Graphs (C7F-G)**
- **IRA Graph (C7H)**
- **Using Trig. Functions (C8A-C8B)**
- **Gradebook-alias-Production/Sales Record (C9A-C9E)**
- **Case Study A. Expense Log (C11A-C11H)**
- **Case Study B. Portfolio Evaluation (C11I-C11P)**
- **Case Study C. Net Worth (C11Q-C11R)**
- **Case Study D. Sales Projection (C11S-C11U)**
- **Case Study E. Interest Calculations (C11V-C11X)**

Mail to:
Wm. C. Brown Co. Pub.
P.O. Box 539
Dubuque, Iowa 52001

☐ **YES, please send me the diskette containing specially prepared worksheets to accompany VisiCalc for the Apple II Plus Computer, by Edouard Desautels. I understand that if I am not satisfied with the diskette, I may return it within 30 days without further obligation.**

Diskette	**$39.95**
Tax (California residents add 6%)	**$_____**
Postage and handling	**$_____**
Total amount	**$_____**

Please indicate method of payment:

☐ **Check/Money Order enclosed (WCB pays postage and handling)**
☐ **Charge my credit account:**

☐ Visa ☐ Master card

Account No. ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ MC Bank # ☐☐☐☐

Exp. Date ____ Signature _____
(required for all charges)

Name _____

Address _____

City _____

State _____ **ZIP** _____

Phone _____

THE MICROCOMPUTER POWER SERIES

A growing library of guidebooks filled with helpful information, money-saving tips, and useful programs, THE MICROCOMPUTER POWER SERIES is the best way to power up your micro!

VisiCalc® for the TRS-80™ Model II and Model 16 Computers

by Edouard Desautels
160 pages
ISBN 09955-5
$16.95

VisiCalc® for the TRS-80™ Model I and Model III Computers

by Edouard Desautels
160 pages
ISBN 09956-3
$16.95

VisiCalc® for the IBM Personal Computer

by Edouard Desautels
176 pages
ISBN 09967-9
$16.95

VisiCalc® for the Apple II® Computer

by Edouard Desautels
160 pages
ISBN 09968-7
$16.95

To get you acquainted with the hottest software package on the market, these unique guidebooks provide easy, step-by-step instructions for running the VisiCalc software on your particular system.

Desautels is remarkably thorough in coverage as well as in the selection and development of examples. He carefully explains how the VisiCalc software interacts with the underlying operating system and its file structure. Desautels doesn't just tell you what a microcomputer does but gets you involved in actually working with it.

With the VisiCalc program's modeling approaches to financial data, you'll soon be calculating financial trends, making sales and cash flow projections, keeping an expense log, computing your net worth, and more.

VisiCalc® is a registered trademark of VisiCorp.
TRS-80™ is a trademark of the Tandy Corp.
Apple II® is a registered trademark of Apple Computer, Inc.

INTRODUCTION TO GRAPHICS

John P. Grillo
J. D. Robertson
144 pages/38 programs
ISBN 09953-9
$15.95

INTRODUCTION TO GRAPHICS will show you the kinds of graphics your micro can generate while giving you complete instructions to creating them. This helpful guidebook progresses from simple graphics techniques produced by a series of print statements to more sophisticated displays produced by table driven programs using special graphics characters.

GUIDE TO SYSTEMS APPLICATIONS

John P. Grillo
J. D. Robertson
288 pages/61 programs
ISBN 09952-0
$17.95

An amazingly clear presentation of the essential hardware and software features of a microcomputer system, GUIDE TO SYSTEMS APPLICATIONS illustrates how complete systems can be designed and implemented. Three systems are developed, with examples of how they can be adapted and expanded. All of the programs are immediately usable and demonstrate a wide range of applications.

TECHNIQUES OF BASIC

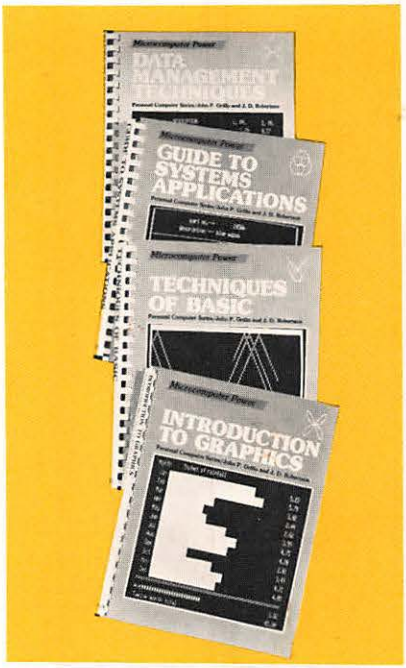
John Grillo
J. D. Robertson
272 pages/61 programs
ISBN 09951-2
$18.95

TECHNIQUES OF BASIC demonstrates techniques for writing sound, structured programs for business, educational, scientific, and home applications. Well-illustrated examples cover diverse topics such as simple data analysis, conversational statistics, isotherm graphing, text processing, line renumbering, banner printing, message managing, shell sorts, binary search trees, and more.

DATA MANAGEMENT TECHNIQUES

John Grillo
J. D. Robertson
208 pages/48 programs
ISBN 09954-7
$16.95

Designed for intermediate to advanced programmers, DATA MANAGEMENT TECHNIQUES really helps you use your micro more effectively. The authors demonstrate and explain the many ways to manage data in memory and on disk or tape files. Included are list and array processing as well as queuing, stack, and tree processing. Every technique is illustrated in simple, straightforward BASIC.



THE MICROCOMPUTER POWER SERIES

is available in better bookstores and through:
Wm. C. Brown Company, Publishers
2460 Kerper Boulevard
Dubuque, Iowa 52001

(Use the order card in the back of this book)

WM. C. BROWN COMPANY PUBLISHERS
Dubuque, Iowa