Computers and Society

David D. Thornburg, Associate Editor

Computer Ethics

Over the past two years I've been hearing more and more concern about ethics in this country—with special emphasis on the ethical considerations associated with computer technology. The debate ranges from concern about the displacement of human workers by robots to the issues surrounding the copying of commercial software.

To take just one example, I recently spoke at a computing conference in which I was asked what my feelings were about the unauthorized copying and distribution of commercial software. In this particular case, the questioner was a teacher whose budget was very tight. My response was simply this: Unlike murder, which is only a state crime, the illicit copying of commercial software is a federal offense. I don't think much of murder, and I don't think much of those who deprive hard-working software companies of their just rewards for their efforts. While it might be interesting to study why otherwise law-abiding people are willing to even consider making copies of copyrighted material, that is a topic for another column.

Many people think the new technologies of the information age require more than technical skills on the part of their users—they require some thought about the ethical consequences of these technologies, both from a personal and from a societal perspective. Someday we might see the following headlines in our daily papers:

"Berserk Robot Kills Six at Auto Plant"

"Computer Failure at Hospital Threatens Safety of Hundreds"

If these disasters happened, the affected community would be outraged. But once the headlines died down, the long, drawn-out process of assessing responsibility would begin. Who was at fault? What could be done to keep this from happening again?

The Impact Of Technology

As we develop new information and automation technologies that our children will use as freely as we use paper and pencils, we should give some thought to preparing them for the complexities that arise—not just from the technology itself, but from the impact this technology can have on the people who make it, and especially on those who use it.

With this thought in mind, I was pleased to come across a book entitled *Computer Ethics* by Thomas Kemnitz and Philip Vincent ($9.95 from Trillium Press, Box 921, Madison Square Station, New York, NY 10159). It's not a book that "teaches" ethical behavior; instead, it explores the complexities of the topic for youngsters in grades seven and up.

The book consists of numerous hypothetical cases, each of which raises an interesting question for which the answer is not at all clear. Instead of presenting a point of view, the book presents a balanced view of both sides of the issue. Then it asks questions that stimulate readers to formulate their own opinions on the case and to present these opinions in a well-thought-out manner. Here's an example:

> In this day and age, one often hears talk of "human rights." People talk of certain individual's rights being violated. Some individuals maintain rights are guaranteed because of a government's "constitution." Others maintain that all human beings are, or should be guaranteed, certain rights simply because they are human beings. A common error is committed when one speaks of rights without examining the philosophical background that constitutes or guarantees these rights. Should rights be guaranteed to people? Animals? Machines?...
>
> If cognitive abilities are the criteria for granting or having rights, then humans obviously have rights...Should computers have rights? Are not computers capable of reasoning, analyzing and processing information? Are computers capable of enlightening us or other computers if they are so requested? As computer technology grows, many feel computers will be able to duplicate and exceed the thinking capabilities now dominated by man...
>
> What should be the basis for rights?

If The Headlines Happen

The activities that follow this case explore the issues that were raised in some depth, without expressing a particular position. For example: How do you distinguish between rights and privileges?

In addition to raising interesting questions about the social consequences of technology, *Computer Ethics* stimulates critical thinking skills. Given the complexity of the world into which our children are growing, this skill is one that should be nurtured and developed from an early age.

I hope we never see the kinds of headlines that would result if some of the hypothetical cases in *Computer Ethics* became real. But if we do, I hope even more that we as a society will be prepared to engage in the kind of debate that can not only resolve the issue at hand, but that can help make our world a safer and happier place in which to live.

Dr. Thornburg's most recent product is Calliope, *a nonlinear idea processor for the Apple IIe, IIc, and Macintosh computers. He welcomes letters from readers and can be reached in care of this magazine.*©

The Beginners Page

Tom R. Halfhill, Editor

String Comparisons

As we've pointed out more than once in the past few columns, computers really know nothing about our written language of alphabetic characters, punctuation marks, and symbols—they are capable of dealing only with numbers. Although this means computers have to spend a lot of time translating things for our convenience (and vice versa), it also means that computers can perform "arithmetic" on character strings.

This concept seems a little strange at first, because we're used to thinking of the written word and mathematics as two different, incompatible languages. After all, a phrase such as "The quick brown fox jumped over the lazy dogs" is just as meaningless in mathematics as the phrase "X=(Y+2)*(Z/4)" is in English. But since a computer sees "The quick brown fox..." as merely a string of numbers (character codes), we can write programs that perform a kind of arithmetic on what appears to *us* as strings of characters. Here's an example:

```
IF "A"<"B" THEN PRINT "IT WORKS!"
```

When you press RETURN, the result is the message IT WORKS!.

Notice the subtle yet vital difference between this line and the statement IF A<B THEN PRINT "IT WORKS!". Although both statements are comparing two values with an arithmetic operator (<, the less-than sign), the first statement isn't comparing two *numeric* values; it's comparing two *character* values.

At least, that's how it looks on the surface. From the computer's point of view, two numbers—character codes—are still being compared. The character A is "less than" the character B because the character code for A is a smaller number than the character code for B. You can confirm this by typing PRINT ASC("A") and PRINT ASC("B")—the character codes are 65 and 66, respectively. (See the February 1986 "Beginner's Page" for more details on ASCII character codes.) It's easy to remember that the letter A is less than the letter B, because A precedes B in the alphabet. But keep in mind that it's really the character codes, *not the alphabetical positions*, that count. Consider this example:

```
IF "A">"a" THEN PRINT "IT WORKS!"
```

From the computer's point of view, two numbers—character codes—are being compared. The character A is "less than" the character B because the character code for A is a smaller number than the character code for B.

When you enter this statement, you might expect to see the message IT WORKS!. Alphabetically, the uppercase letter *A* should take precedence over the lowercase letter *a*. But it doesn't work that way on most computers. Instead, the IF-THEN test fails; *A* is not greater than *a*. Why? Because the character codes for uppercase letters are numbered from 65 to 90, and the codes for lowercase letters are numbered from 96 to 122. (Yes, it's odd.) Therefore, *A* (65) is less than *a* (96). The statement above is really the equivalent of this:

```
IF ASC("A")>ASC("a") THEN PRINT
  "IT WORKS!"
```

which, in turn, is the equivalent of this:

```
IF 65>96 THEN PRINT "IT WORKS!"
```

As long as the computer can figure out that 65 isn't greater than 96, it doesn't have to know anything about alphabets.

Incidentally, you'll get different results if you try some of these examples on Commodore computers (except the Amiga). Commodore machines assign character codes a bit differently than other computers do. Normally, the Commodore 64, 128, and VIC-20 don't display upper/lowercase characters—you have to press the SHIFT-Commodore keys to switch to this mode. This renumbers the lowercase character set from 65 to 90 and the uppercase set from 193 to 218. So on a Commodore, the uppercase letters are indeed "greater than" the lowercase letters.

Other types of comparisons are possible with strings, too. Try these:

```
IF "OK"="OK" THEN PRINT "OK"

IF "DIAGNOSTIC TEST"<>"DIAG-
  NOSTIC TEST" THEN PRINT
  "YOU'VE GOT A HARDWARE
  PROBLEM"

IF "DOG">"CAT" THEN PRINT "TOLD
  YA SO"
```

All of the examples we've seen so far compare *string literals*. Of course, you can also compare characters stored in string variables:

```
10 DIM A$(5),B$(5):REM This line for
   Atari only
20 A$="<"
30 B$=">"
40 IF A$<B$ THEN PRINT "< IS LESS
   THAN >"
```

String arithmetic isn't limited to comparisons. Next month, we'll see how you can *add* two strings together in various versions of BASIC, and cover some remaining string functions as well. ©

INSIGHT: Atari

Bill Wilkinson

Avoiding Disk Errors

I know many of you will find this hard to believe, but I've never encountered a disk error on the Atari which I couldn't explain. Further, I have had very few DOS errors, ever. (The reasons for the few errors I have encountered, by the way, were always related to random access files—a common problem with Atari DOS 2.0 and its derivatives.) Yet after a few hundred phone calls and letters, I know that many of you have experienced the frustration of wiped-out disk files. Why? Well, I can't know each and every reason, and I can't repair damage that's already been done, but maybe I can give you some helpful hints for the future.

Hands Off That Disk

Hint 1: Never, never, never take a disk out of a drive unless the program you're using tells you to. (This goes beyond even the good advice about never removing a disk when the drive is still spinning.) In particular, never swap disks until prompted to do so. Why? Well, because the Atari disk drive has no way to tell the computer that the disk has been removed or changed.

Consider: How does any DOS know what disk sectors to allocate to a new file? Generally, a DOS keeps a list of unused disk sectors. The next time it needs to find a sector (for example, to extend a file), it takes one from this list. The list (called a *Volume Table Of Contents* or VTOC in Atari parlance) is usually kept on disk until a file is opened, when it is read into memory. It is rewritten to the disk when a file is closed.

Okay, now open a file for output, write some information, swap disks, and write more data. What happens? The list of sectors was correct for the first disk, but it's extremely unlikely that it bears any reasonable relation to what exists on the second disk. Most probably, DOS will allocate several sectors which were already part of other files on the second disk. Kablooey!

If you're using an application program, then, follow the prompts and don't swap disks unless told to. If you're programming and working with disk files, make sure you close all open files before swapping disks (END automatically closes all open files in BASIC). If you're using DOS, you should be safe as long as you change disks only at the DOS prompt. Of course, when duplicating files or disks, you must swap disks when DOS tells you to.

Beware Of RESET

Hint 2: Never hit the SYSTEM RESET button during a disk operation. For example, if you hit RESET in the middle of a SAVE, it's possible to end up with a completely blown program. In fact, if you then SAVE the program to disk again, you could end up with a blown disk file.

This results from a really subtle bug in DOS 2.0. When DOS enters what is known as *burst I/O* mode (to speed up input/output), it "copies" the memory to disk. But DOS 2.0's file organization requires that the last three bytes of each sector contain a link to the next sector in the file. How can it do this when it is writing directly from memory? Answer: By "swapping" three bytes of memory long enough to write a sector, and then restoring the bytes.

Now suppose you happen to hit RESET when those three bytes are swapped out. Oops...say goodbye to your program.

There are two ways to fix this problem. First, since DOS gets control after a RESET, it could check to see if a disk write was interrupted. If so, it could restore the three bytes. Or, second, DOS could always copy bytes to be written into a buffer, thus never disturbing the program (or data) in memory. The second approach is successfully used by DOS 2.5.

Missing Sectors

Hint 3: Avoid hitting RESET during disk operations even if you're using DOS 2.5, because you may still mess up the disk a bit. Here's one way: Open a file for write (OPEN #5,8,0,"D:FILE" in BASIC, for example), write some data, OPEN another file for write, write data to both files, CLOSE the first file, write some more data to the second file, and then hit RESET. What happens?

The VTOC says the sectors in the second file which were written before the CLOSE are now in use (and that was true when the CLOSE took place). If you add the number of free sectors remaining on the disk to the number of sectors used in all files, the total is no longer 707 in single density or 1010 in enhanced density, as it should be. You just lost some of your disk space.

Hint 4: Everything I just mentioned about RESET also applies to turning off the power. For example, if you have a power failure in the middle of a SAVE from BASIC or while there are some data files open in a business program, be prepared for some problems.

Fortunately, DOS 2.5 comes with a program called DISKFIX.COM which does a pretty good job of fixing up a "damaged" disk (either DOS 2.0 or 2.5). It allows you to undelete files as long as you haven't written any new files since the deletion. At your choice it will either try to recover or permanently remove a file which was left open for output. And, most importantly, it checks each file on the disk to make sure it is OK, and then reconstructs the VTOC to ensure that all 707 or 1010 sectors are accounted for. ©

Programming the TI

C. Regena

Animation In TI BASIC

The theme for this issue of COMPUTE! is graphics and animation, so we'll discuss some ways you can animate in TI BASIC. TI Extended BASIC adds really fun animation with the sprite features, but even in regular console BASIC you can make characters move. I'm going to suggest four ways you can animate your graphics.

Perhaps the simplest way to move something on the screen upward is to use the PRINT statement. The short program below draws a rocket at the bottom of the screen. Lines 140–160 print blank lines which move the rocket toward the top of the screen. (You may also use PRINT with colons.) Of course, any other graphics you might have on the screen also move upward as you print. This method works best with larger objects that need to be moved upward because you don't need to redraw the graphics.

```
110 CALL CLEAR
120 CALL CHAR(130,"1038383E
    38387C44")
130 CALL HCHAR(24,15,130)
140 FOR P=1 TO 21
150 PRINT
160 NEXT P
170 END
```

The next short program illustrates a way to move an object across the screen horizontally. This method involves erasing the object and then redrawing it—all the way across the screen. Unlike the previous method, this won't affect other graphics on the screen. Although it works with several characters, objects move more quickly and more smoothly if you use only one character.

The program redefines character number 130 as an arrow. In a FOR-NEXT loop that changes the column number, first a space (character number 32) is placed in the previous column to erase the existing arrow, then the new arrow is drawn in the next column. This repeatedly erases and redraws the arrow one column to the right. Run the program to see how fast the arrow moves across the screen.

```
110 CALL CLEAR
120 CALL CHAR(130,"080C0EFF
    0E0C08")
130 FOR C=3 TO 28
140 CALL HCHAR(8,C-1,32)
150 CALL HCHAR(8,C,130)
160 NEXT C
170 END
```

This method is probably the most common way to move a character. You can move it in any direction by erasing the character in the present position, then changing the row and column and redrawing it in another position. In this short example, we've erased the character with a blank space. But if the character is moving over other graphics, you might need to erase it with the appropriate graphics characters to restore the background. Otherwise, the moving character would leave behind a trail of spaces. This method of animation is rather jerky if your object consists of several characters that need to be moved, but it can be fairly quick with just one character.

CALL COLOR Motion

The next example program illustrates a different way to move an object made up of several characters. Rather than moving one character at a time, we'll use CALL COLOR to make all the characters in the set invisible at once, then another CALL COLOR to make the object in the next position visible.

This sample draws an eight-character horse. The horse is actually drawn eight times on the screen using eight different character sets. Lines 130–200 define strings for eight graphic character definitions. The loop in lines 210–250 defines the graphic characters. In each of the sets from number 9 to 16, the characters are defined using the strings A$.

Lines 260–380 are another loop. Line 270 makes the characters invisible. Lines 280–290 determine a character number and a row number depending on the set number. Lines 300–370 draw the horse. This loop draws eight horses on the screen vertically, but they are all invisible.

Lines 390–470 are the loops that create the movement. The CALL COLOR statement with a 14 defines the horse as color 14 for a particular color set. The horse moves up and down as the color set number varies and one set is made invisible and the next set made visible. All you need to do is add the rest of the carousel and the music!

This example has only one horse moving up and down. You could draw more horses on the screen—for example, with the set number 9 horse at the bottom of the screen and the set 16 horse at the top. No matter how many horses are on the screen, the CALL COLOR statement changes all the horses in a particular set. You can have several objects moving at the same time by using the CALL COLOR loops in lines 390–470.

```
110 CALL CLEAR
120 CALL SCREEN(16)
130 A$(1)="2E3F3F7F7FFFE7E7
    "
140 A$(2)="008080C0C0E0E0E"
150 A$(3)="07070707071F7FFF
    "
160 A$(4)="F1FFFFFFFFFFFFF8"
170 A$(5)="FFFDFCFCFCFCF8F8
    "
180 A$(6)="C0C0E0E0E0602"
190 A$(7)="C060301A0E"
200 A$(8)="F8783C1830606"
210 FOR C=9 TO 16
220 FOR J=1 TO 8
230 CALL CHAR(C*8+23+J,A$(J
    ))
240 NEXT J
250 NEXT C
260 FOR C=9 TO 16
270 CALL COLOR(C,1,1)
280 CH=C*8+23
290 ROW=(C-8)*3-1
300 CALL HCHAR(ROW-1,14,CH+
    1)
```

```
31Ø CALL HCHAR(ROW-1,15,CH+
    2)
32Ø CALL HCHAR(ROW,14,CH+3)
33Ø CALL HCHAR(ROW,15,CH+4)
34Ø CALL HCHAR(ROW,16,CH+5)
35Ø CALL HCHAR(ROW,17,CH+6)
36Ø CALL HCHAR(ROW+1,14,CH+
    7)
37Ø CALL HCHAR(ROW+1,16,CH+
    8)
38Ø NEXT C
39Ø FOR C=9 TO 16
40Ø CALL COLOR(C-1,1,1)
41Ø CALL COLOR(C,14,1)
42Ø NEXT C
43Ø FOR C=16 TO 9 STEP -1
44Ø CALL COLOR(C,1,1)
45Ø CALL COLOR(C-1,14,1)
46Ø NEXT C
47Ø GOTO 39Ø
48Ø END
```

CALL CHAR Animation

The last method of animation I'm going to discuss this month is using CALL CHAR. Just as CALL COLOR instantly changes the color of all characters on the screen in that color set, CALL CHAR redefines a graphic character definition of all characters of that number on the screen. For example, if you have something on the screen and execute CALL CHAR(32,"FF"), all of the characters with number 32 (all the spaces) instantly change to the new character definition, in this case a horizontal line.

The following program illustrates this technique. Lines 110–180 clear the screen and draw a simple face using keyboard symbols. You can draw a much fancier face, but this is just a sample. To type the eyes, use the function key along with the C key to get the ` mark. This is character 96. Line 190 redefines character 96 for an open eye. Lines 200–210 create a delay loop while the eye is open, then line 220 redefines character 96 as a closed eye. Lines 230–240 create another delay loop. Line 250 branches back to line 190 to open the eye.

```
11Ø CALL CLEAR
12Ø PRINT TAB(6);"@@@@@@@@@
    @"
13Ø PRINT TAB(6);"@
    @"
14Ø PRINT TAB(6);"@   `  `
    @"
15Ø PRINT TAB(6);"!      ^
    !"
16Ø PRINT TAB(6);"!      ~
    !"
17Ø PRINT TAB(6);"\
    /"
18Ø PRINT TAB(7);"\______/"
    ::::
19Ø CALL CHAR(96,"18247A7A7
    A7E81")
20Ø FOR DELAY=1 TO 5ØØ
21Ø NEXT DELAY
22Ø CALL CHAR(96,"ØØØØØØØ8
    37D2548")
23Ø FOR DELAY=1 TO 1ØØ
24Ø NEXT DELAY
25Ø GOTO 19Ø
26Ø END
```

Latest TI News

Now a few comments on the TI-99/4A world. I enjoyed a recent visit to Las Vegas to the Southern Nevada Users Group (SNUG, P.O. Box 26301) and also met several people from the Los Angeles and San Diego areas. Terri Masters, president of the L.A. 99er Computer Group (P.O. Box 3547, Gardena, CA 90247), was busy preparing for their Fest-West expo to be held March 1–2. It will be over by the time you read this, but you can plan on attending next year. Chicago holds an annual fest in October, and other groups have expos as well, so you can see the TI-99/4A is not dead.

I also met Craig Miller of Millers Graphics (1475 W. Cypress Avenue, San Dimas, CA 91773), who demonstrated his GRAM Kracker, which will open up all sorts of possibilities for TI owners. This device can save a module (cartridge) program onto a disk or cassette. It also allows you to change or customize a module program—for example, change the title screen or default colors. Miller was also distributing his new book, *The Orphan Chronicles*, by Ron Albright. This book tells the history of the TI and includes a current list of TI dealers, manufacturers, and user groups.

Les Merryman (Lancaster, California), a distributor for Myarc, was also at the SNUG meeting and showed several new Myarc products, including disk controllers, a hard disk drive, and their new Extended BASIC module.

Please don't write to me about hardware products—write directly to the manufacturers and distributors. There are still companies making peripherals for the TI, and there are people who have as many as four disk drives hooked up to their machines. Even though Texas Instruments quit selling the TI-99/4A more than two and a half years ago, user groups are still going strong, and it's amazing what people are doing with their TIs. ©

Donald B. Trivette

The PC/VCR Connection

Remember those 8 mm home movies you took back in the 1960s and 70s...the ones stored away in a shoebox on the top shelf of a closet...the ones you haven't seen in years because it's too much trouble to set up the projector and screen? Now, if you have a videocassette recorder (VCR), you can show them on your TV set.

The first step is to transfer the film to tape. There are commercial firms in most major cities that specialize in this service. Ask your video dealer to recommend one or call the tape editor at your local TV station for suggestions. The cost is quite reasonable—usually just $2 to $3 for 50 feet of film, plus about $6 for the cost of a two-hour tape. Most firms give substantial price breaks for 200- and 400-foot reels.

The picture quality of the tape can actually exceed that of the original film if the transfer is properly done. This means you should avoid firms that transfer the film to tape by projecting your movies on a screen and recording the image with a video camera. Because film and video have different speeds (18 frames per second versus 30 fps), taping from a movie screen can result in horizontal interference lines and flickering. Professional transfer firms have special equipment to overcome this problem.

Once the film is copied on tape, you can add music and narration if your VCR has dubbing features. If not, consider renting a VCR with those features and, while you have the second machine available, make copies of the tape for friends or relatives. As your tapes begin to take on a professional-looking quality, you'll want to add titles, too. Here's where your IBM PC or PCjr really shines.

Simple Patchwork

All you need to make titles with your computer is a cable to connect the composite video output to the VCR. You'll need a shielded cable with a male RCA-type plug on both ends. (Electronics stores such as Radio Shack have them in different lengths for about $5. Or you can borrow one from a stereo system.) Plug one end of the cable into the video input jack on the back of the VCR. If your VCR is an older model with nothing but an antenna connection, you should rent or borrow a newer machine for best results. The other end of the cable plugs into the jack labeled V on the back of the PCjr, or into the jack on the PC's color/graphics adapter board. (If your PC only has the monochrome adapter, you lack the necessary hardware.)

Once the connection is made, you can record virtually anything that appears on your computer screen, although some color combinations that look good on an RGB monitor don't record well.

If your computer's display is a composite monitor, you'll have to disconnect it to plug the patch cable into the composite video output. That means you'll need to figure out some method for previewing the titles—the computer output. The easiest way is to connect a TV set to the VCR as usual. Then, whatever your computer is "playing" will be displayed on the TV and can be recorded by the VCR—just as though the PC were a TV station or cable system. Alternatively, you can view the computer output on an RGB monitor or TV connected directly to the PC or PCjr.

Creating Your Own Titles

The next step is to produce the titles. Things like: *Christmas 1975, Eric's 4th Birthday, Vacation in Hawaii.* You can use any program that produces text on the computer screen, preferably in a large size and in color. You'll want something that doesn't leave a menu line or blinking cursor on the screen.

For really professional results (at a professional price—$250), it's difficult to beat IBM's *PC Story Board* software. This program is designed for making animated graphics presentations. Besides having different sizes and styles of type—shadowed, outlined, and slanted in either direction—*Story Board* allows you to dissolve, wipe, explode, push, and weave from one screen to another. A whole series of titles can be stored on disk and played back automatically in a timed sequence.

Story Board is designed for corporate presentations, and although the results are spectacular, most of us can't justify spending $250 to title home videos. Fortunately, there's an economical alternative.

You can produce colorful, attractive titles with a very simple BASIC program—even if you're not a programmer. The program below produces three-line titles in colors; consult the COLOR statement (for text) in your BASIC manual to equate a color with a number (e.g., 1=blue, 2=green, etc.). Background colors must be in the range of 0 through 7; foreground (text) must be in the range of 0 through 15. Insert spaces ahead of the text to center the lines on the screen.

The INKEY$ statement in line 200 keeps the OK prompt off the bottom of the screen. On the PCjr, you can generate even larger characters by changing the number 40 in line 100 to 20. This displays 20 characters per line instead of 40.

IBM PC/PCjr Video Titler

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" in this issue of COMPUTE!.

```
EH 10 REM The 3 title lines foll
      ow
JA 20 LINE1$="                 CHRIS
      TMAS"
6E 30 LINE2$="                    19
      80"
```

```
LK 40 LINE3$="                  at Gra
      ndma's"
LD 50 REM The colors for each of
       the three lines follow
HA 60 COLR1=4            'This is re
      d for line 1
OH 70 COLR2=7            'This is wh
      ite for line 2
IM 80 COLR3=1            'This is bl
      ue for line 3
AE 90 BACKGROUND=0       'This is bl
      ack. Change 0 to 1 for blu
      e, etc.
CI 100 WIDTH 40:CLS:KEY OFF:LOCA
       TE 1,1,0
IC 110 PRINT
EH 120 COLOR COLR1,BACKGROUND
FB 130 PRINT LINE1$
BE 140 PRINT:PRINT:PRINT
GN 150 COLOR COLR2
GG 160 PRINT LINE2$
CK 170 PRINT:PRINT:PRINT
HC 180 COLOR COLR3
IL 190 PRINT LINE3$
KO 200 A$=INKEY$:IF A$="" THEN 2
       00
GB 210 WIDTH 80:COLOR 7,0
LP 220 END
```
©

CAPUTE!

MLX Mixup

If you've tried to enter the "Screen Genie" program from the April issue, you may have discovered that the version of "MLX" published in that issue (p. 123) cannot be used. The MLX program in the April issue is an accidental resurrection of the old version, which has not been used since November 1985. Screen Genie must be entered with the new version of MLX, which appeared in the December, January, and February issues.

Switchbox

The Atari version (Program 3) of this game from the March issue (p. 34) has two typos. In line 130, the AR$(1.3) should be AR$(1,3). The proper Proofreader checksum for the corrected line is NH. More significantly, in line 460 the character shown as {=} should actually be SHIFT-=, the vertical line character.

In the Apple version (Program 4), the following line should be added to ensure that no extraneous characters appear on the game screen:

```
20 1035 FOR I = 36096 TO 36103:
        POKE I,0: NEXT
```

In the Atari ST version (Program 7), a set of quotation marks is missing in line 20. The first variable definition in the line should be:

sp$(0)="\ _":.

Commodore Program Profiler

In this utility program from the February 1986 issue, two of the DATA lines contain spurious question marks and are missing characters. Lines 170 and 370 should read as follows:

```
170 DATA 160,192,32,30,171,32,
    207,255,201,13      :rem 99
370 DATA 126,192,141,127,192,1
    65,157,208,106,169 :rem 75
```

IBM PrtSc Protector & Screen Clock

These two programs—"PrtSc Protector" from the February issue (p. 81) and "Screen Clock" from the April issue (p. 107)—both have the same problem. Due to a quirk in the program we use to generate listings, the "Automatic Proofreader" checksums for the DATA lines are incorrect. That is, the programs are correct as published, but, if you attempt to type them in using our Automatic Proofreader utility, the checksums you'll get for the DATA lines will be different from those shown in the magazine. As a result, we recommend that these programs be entered directly, without using the Proofreader.

ST Doodler

A bracket character is missing in this Logo graphics program for the Atari ST in the February 1986 issue (p. 78). In the procedure BCORF at the bottom of the middle column, there should be a left bracket, [, before SETPOS. ©

Attention Programmers

COMPUTE! magazine is currently looking for quality articles on Commodore, Atari, Apple, and IBM computers (including the Commodore Amiga and Atari ST). If you have an interesting home application, educational program, programming utility, or game, submit it to COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. Or write for a copy of our "Writer's Guidelines."

COMPUTE's Author Guide

Most of the following suggestions serve to improve the speed and accuracy of publication. COMPUTE! is primarily interested in new and timely articles on the Commodore 64/128, Atari, Apple, IBM PC/PCjr, Amiga, and Atari ST. We are much more concerned with the content of an article than with its style, but articles should be clear and well-explained.

The guidelines below will permit your good ideas and programs to be more easily edited and published:

1. The upper left corner of the first page should contain your name, address, telephone number, and the date of submission.

2. The following information should appear in the upper right corner of the first page. If your article is specifically directed to one make of computer, please state the brand name and, if applicable, the BASIC or ROM or DOS version(s) involved. In addition, *please indicate the memory requirements of programs.*

3. The underlined title of the article should start about 2/3 of the way down the first page.

4. Following pages should be typed normally, except that in the upper right corner there should be an abbreviation of the title, your last name, and the page number. For example: Memory Map/Smith/2.

5. All lines within the text of the article must be double- or triple-spaced. A one-inch margin should be left at the right, left, top, and bottom of each page. No words should be divided at the ends of lines. And please do not justify. Leave the lines ragged.

6. Standard typing paper should be used (no erasable, onionskin, or other thin paper) and typing should be on one side of the paper only (upper- and lowercase).

7. Sheets should be attached together with a paper clip. Staples should not be used.

8. If you are submitting more than one article, send each one in a separate mailer with its own tape or disk.

9. Short programs (under 20 lines) can easily be included within the text. Longer programs should be separate listings. *It is essential that we have a copy of the program, recorded twice, on a tape or disk.* If your article was written with a word processor, we also appreciate a copy of the text file on the tape or disk. Please use high-quality 10 or 30 minute tapes with the program recorded on both sides. The tape or disk should be labeled with the author's name, the title of the article, and, if applicable, the BASIC/ROM/DOS version(s). Atari tapes should specify whether they are to be LOADed or ENTERed. We prefer to receive Apple programs on disk rather than tape. Tapes are fairly sturdy, but disks need to be enclosed within plastic or cardboard mailers (available at photography, stationery, or computer supply stores).

10. A good general rule is to spell out the numbers zero through ten in your article and write higher numbers as numerals (1024). The exceptions to this are: Figure 5, Table 3, TAB(4), etc. Within ordinary text, however, the zero through ten should appear as words, not numbers. Also, symbols and abbreviations should not be used within text: use "and" (not &), "reference" (not ref.), "through" (not thru).

11. For greater clarity, use all capitals when referring to keys (RETURN, TAB, ESC, SHIFT), BASIC words (LIST, RND, GOTO), and three languages (BASIC, APL, PILOT). Headlines and subheads should, however, be initial caps only, and emphasized words are not capitalized. If you wish to emphasize, underline the word and it will be italicized during typesetting.

12. Articles can be of any length—from a single-line routine to a multi-issue series. The average article is about four to eight double-spaced, typed pages.

13. If you want to include photographs, they should be either 5×7 black and white glossies or color slides.

14. We do not consider articles which are submitted simultaneously to other publishers. If you wish to send an article to another magazine for consideration, please do not submit it to us.

15. COMPUTE! pays between $70 and $800 for published articles. In general, the rate reflects the length and quality of the article. Payment is made upon acceptance. Following submission (Editorial Department, COMPUTE! Magazine, P.O. Box 5406, Greensboro, NC 27403) it will take from four to eight weeks for us to reply. If your work is accepted, you will be notified by a letter which will include a contract for you to sign and return. *Rejected manuscripts are returned to authors who enclose a self-addressed, stamped envelope.*

16. If your article is accepted and you have since made improvements to the program, please submit an entirely new tape or disk and a new copy of the article reflecting the update. We cannot easily make revisions to programs and articles. It is necessary that you send the revised version as if it were a new submission entirely, but be sure to indicate that your submission is a revised version by writing, "Revision" on the envelope and the article.

17. COMPUTE! does not accept unsolicited product reviews. If you are interested in serving on our panel of reviewers, contact the Review Coordinator for details.

Classified

SOFTWARE

TI-99/4A Software/Hardware bargains.
Hard to find items. Huge selection.
Fast Service. Free Catalog.
D.E.C., Box 690, Hicksville, NY 11801

LOTTO PICKER. Improve your chances for those Million Dollar Jackpots! Picks LOTTO, WIN-4, and Daily Numbers. All USA & CAN. games incl. Expandable! IBM/C64/TI99 $29.95. Order Now! 1-800-341-1950 Ext. 77. Mail Orders: Ridge, 170 Broadway, #201C, NYC, NY 10038. Catalog.

PROJECT PLANNING/MANAGEMENT using the C64, SX, or C128. Data sheet for SASE Program for $106.95 (CA res. add 6% s.t.) LAWCO, Dept. CL, Box 2009, Manteca, CA 95336

FREE SOFTWARE CATALOG!
Call Toll-Free 1-800-554-1162, Tevex, Inc.
Save 1/3 off retail prices. We carry SSI,
Elect. Arts, Infocom, and many more!

COMMODORE: TRY BEFORE YOU BUY. Top 25 best-selling games, utilities, new releases. Visa, MasterCard. Free brochure. Rent-A-Disk, 908 9th Ave., Huntington, WV 25701 (304) 522-1665

DISCOUNT SOFTWARE: Amiga/Apple/Atari/C64-128/IBM PC-PCjr/TRS-80/Timex/Sinclair. Free Catalog: WMJ DATA SYSTEMS, 4 Butterfly Dr., Hauppauge, NY 11788

FREE SOFTWARE for C64, C128, IBM & CPM. For info send large stamped (39¢) return envelope to: PUBLIC DOMAIN USERS GROUP, PO Box 1442-A1, Orange Park, FL 32067

TI-99/4A QUALITY SOFTWARE for Business, Home and Entertainment **BONUS SOFTWARE OFFER!** Send for FREE catalog to MICRO-BIZ HAWAII, BOX 1108, Pearl City, HI 96782

ATTENTION TI99/4A OWNERS !!!
See our ad in Product Mart.
The 99/4A National Assistance Group
(305) 583-0467

BETTER GRADES! Isn't that what you want from educational software? SPELLBOUND is a new program from ROBINSOFT that REALLY TEACHES spelling! And it's affordable too. Works with any spelling list (try foreign languages too!), uses recognized educational techniques and incorporates an adjustable difficulty level. Available on diskette for Apple II, C64, TRS-80 III/4, TI99/4A, Kaypro (CP/M). Send $14.95 or write to Robinsoft-C2, Box 666, Prineville, Oregon 97754

Games: IBM-PC/COMMODORE 64 Stocks, Black Jack and Roulette; all 3 for $25 on disk. Horserace Handicapper - produces profits! $85 for disk. Wolf-Bytes, 7833B N. 60th St., Milwaukee, WI 53223. Info: (414) 357-6410

TANDY 1000 PROGRAMS AND NEWSLETTER
Send for free information on educational & entertainment programs & newsletter.
Soda Pop Software, POB 653, Kenosha, WI 53141

ATARIWARE: THE BEST PD SOFTWARE FROM ATARI enthusiasts across the U.S.! 80 disks to choose from - $5 ea. Catalog with SASE. KD-ACE, P.O. Box 1646, Orange Park, FL 32067

DR. T's MUSIC SOFTWARE
A music composition system with true word processing capabilities for APPLE and COMMODORE computers. Also algorithmic composition programs for computer generation of sequences. Fully editable Bach, Keyboard, Bass and Drum disks. Patch librarians available for Yamaha and Casio synthesizers. Reviewed in Jan. '85 *COMPUTE!* (617) 244-6954, Dr. T's, Dept. C, 66 LOUISE RD., CHESTNUT HILL, MA 01267

LINEAR PROGRAMMING IBM 200×200 VAR/COST + LESSONS $25, SAMPLE DSK $10, TURBO PASCAL SOURCE $20, ATARI VERSION 30×30 $12, ATWRIT ERPRINTDRIVER $12, ASM LANG CONVERTER $12. ALCHEMY SYSTEM, PO BOX 694, PAOLI, PA 19301

WOODWORKERS! Program makes bidding custom work easy! C-64, Apple, IBM. $29.95. Visa, MC, check to: BID!, Box 218, Brookston, IN 47923. Info call (317) 563-3504

If you'd like information on the latest version of your software, please call or write: **Batteries Included** Customer Support, 30 Mural St., Richmond Hill, Ontario, Canada L4B 1B5 (416) 881-9816

ESCAPE FROM "DER BUNKER" TI-99/4A
An interactive Adventure on Cassette in 16K Extended Basic. Survive Hitler's Last Refuge. THEY KNOW WHO YOU ARE! "Use the Potato Masher wisely." $10 to: APEX Software, PO Box 90475, Honolulu, HI 96835

MISCELLANEOUS

Discount computer printer ribbons for all makes/models Ex: Epson 1500 Nylon $6.99. Catalog: TWS 1314 4th Ave., Coraopolis, PA 15108 (412) 262-1482 Visa or MasterCard.

IBM PCjr REPORT: THE NATIONAL NEWS-LETTER. PCjr-specific articles, reviews, Public Domain from across the nation. $18./yr. PCjr CLUB, POB 95067, Schaumberg, IL 60195

RIBBONS for MOST PRINTERS at LOW PRICES!
DELTA MICRONICS
BOX 10933, ERIE, PA 16514-0933
(814) 455-5667

INCORPORATE Without Legal Fees!
Free Booklet Tells How, Includes Forms.
Call HARVARD BUSINESS SERVICES NOW:
800-345-CORP

COMPUTE! Classified is a low-cost way to tell over 350,000 microcomputer owners about your product or service.

Rates: $25 per line, minimum of four lines. Any or all of the first line set in capital letters at no charge. Add $15 per line for boldface words, or $50 for the entire ad set in boldface (any number of lines.)

Terms: Prepayment is required. Check, money order, American Express, Visa, or MasterCard is accepted. Make checks payable to COMPUTE! Publications.

Form: Ads are subject to publisher's approval and must be either typed or legibly printed. One line equals 40 letters and spaces between words. Please underline words to be set in boldface.

General Information: Advertisers using post office box numbers in their ads must supply permanent address and telephone numbers. Orders will not be acknowledged. Ad will appear in next available issue after receipt.

Closing: 10th of the third month preceding cover date (e.g., June issue closes March 10th). Send order and remittance to: Harry Blair, Classified Manager, COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. To place an ad by phone, call Harry Blair at (919) 275-9809.

Notice: COMPUTE! Publications cannot be responsible for offers or claims of advertisers, but will attempt to screen out misleading or questionable copy.

COMPUTE!'s Guide To Typing In Programs

Computers are precise—type the program *exactly* as listed, including necessary punctuation and symbols, except for special characters noted below. We have provided a special listing convention as well as a program to check your typing—"The Automatic Proofreader."

Programs for the IBM, TI-99/4A, and Atari ST models should be typed exactly as listed; no special characters are used. Programs for Commodore, Apple, and Atari 400/800/XL/XE computers may contain some hard-to-read special characters, so we have a listing system that indicates these control characters. You will find these Commodore and Atari characters in curly braces; *do not type the braces.* For example, {CLEAR} or {CLR} instructs you to insert the symbol which clears the screen on the Atari or Commodore machines. A complete list of these symbols is shown in the tables below. For Commodore, Apple, and Atari, a single symbol by itself within curly braces is usually a control key or graphics key. If you see {A}, hold down the CONTROL key and press A. This will produce a reverse video character on the Commodore (in quote mode), a graphics character on the Atari, and an invisible control character on the Apple.

Graphics characters entered with the Commodore logo key are enclosed in a special bracket: [<A>]. In this case, you would hold down the Commodore logo key as you type A. Our Commodore listings are in uppercase, so shifted symbols are underlined. A graphics heart symbol (SHIFT-S) would be listed as S̲. One exception is {SHIFT-SPACE}. When you see this, hold down SHIFT and press the space bar. If a number precedes a symbol, such as {5 RIGHT}, {6 S̲}, or [<8 Q>], you would enter five cursor rights, six shifted S's, or eight Commodore-Q's. On the Atari, inverse characters (white on black) should be entered with the inverse video

Atari 400/800/XL/XE

When you see	Type	See	
{CLEAR}	ESC SHIFT <	↖	Clear Screen
{UP}	ESC CTRL -	↑	Cursor Up
{DOWN}	ESC CTRL =	↓	Cursor Down
{LEFT}	ESC CTRL +	←	Cursor Left
{RIGHT}	ESC CTRL *	→	Cursor Right
{BACK S}	ESC DELETE	◂	Backspace
{DELETE}	ESC CTRL DELETE		Delete character
{INSERT}	ESC CTRL INSERT		Insert character
{DEL LINE}	ESC SHIFT DELETE		Delete line
{INS LINE}	ESC SHIFT INSERT		Insert line
{TAB}	ESC TAB	▸	TAB key
{CLR TAB}	ESC CTRL TAB		Clear tab
{SET TAB}	ESC SHIFT TAB		Set tab stop
{BELL}	ESC CTRL 2		Ring buzzer
{ESC}	ESC ESC		ESCape key

Commodore PET/CBM/VIC/64/128/16/+4

When You Read:	Press:	See:	When You Read:	Press:	See:
{CLR}	SHIFT CLR/HOME		[<1>]	COMMODORE 1	
{HOME}	CLR/HOME		[<2>]	COMMODORE 2	
{UP}	SHIFT ↑ CRSR ↓		[<3>]	COMMODORE 3	
{DOWN}	↑ CRSR ↓		[<4>]	COMMODORE 4	
{LEFT}	SHIFT ← CRSR →		[<5>]	COMMODORE 5	
{RIGHT}	← CRSR →		[<6>]	COMMODORE 6	
{RVS}	CTRL 9		[<7>]	COMMODORE 7	
{OFF}	CTRL 0		[<8>]	COMMODORE 8	
{BLK}	CTRL 1		{ F1 }	f1	
{WHT}	CTRL 2		{ F2 }	SHIFT f1	
{RED}	CTRL 3		{ F3 }	f3	
{CYN}	CTRL 4		{ F4 }	SHIFT f3	
{PUR}	CTRL 5		{ F5 }	f5	
{GRN}	CTRL 6		{ F6 }	SHIFT f5	
{BLU}	CTRL 7		{ F7 }	f7	
{YEL}	CTRL 8		{ F8 }	SHIFT f7	
			←	←	

From the publishers of *COMPUTE!*

May 1986 *COMPUTE!* Disk

All the exciting programs from the past three issues of *COMPUTE!* are on one timesaving, error-free floppy disk that is ready to load on your Apple II, II+, IIe, and IIc computers. The May 1986 *COMPUTE! Disk* contains the entertaining and useful Apple programs from the March, April, and May 1986 issues of *COMPUTE!*.

The May 1986 *COMPUTE! Disk* costs $12.95 plus $2.00 shipping and handling and is available only from COMPUTE! Publications.

For added savings and convenience, you may also subscribe to the *COMPUTE! Disk.* At a cost of only $39.95 a year (a $12.00 savings), you'll receive four disks, one every three months. Each disk will contain all the programs for your machine from the previous three issues of *COMPUTE!*.

This is an excellent way to build your software library while you enjoy the quality programs from *COMPUTE!*.

Disks and subscriptions are available for Apple, Atari, Commodore 64 and 128, and IBM personal computers. Call for details.

For more information or to order the May 1986 *COMPUTE! Disk,* call toll free 1-800-346-6767 (in NY 212-265-8360) or write *COMPUTE! Disk*, P.O. Box 5038, F.D.R. Station, New York, NY 10150.

COMPUTE! Publications, Inc. abc
One of the ABC Publishing Companies

key (Atari logo key on 400/800 models).

Whenever more than two spaces appear in a row, they are listed in a special format. For example, {6 SPACES} means press the space bar six times. Our Commodore listings never leave a single space at the end of a line, instead moving it to the next printed line as {SPACE}.

Amiga program listings contain only one special character, the left arrow (←) symbol. This character marks the end of each program line. Wherever you see a left arrow, press RETURN or move the cursor off the line to enter that line into memory. Don't try to type in the left arrow symbol; it's there only as a marker to indicate where each program line ends.

The Automatic Proofreader

Type in the appropriate program listed below, then save it for future use. The Commodore Proofreader works on the Commodore 128, 64, Plus/4, 16, and VIC-20. Don't omit any lines, even if they contain unfamiliar commands or you think they don't apply to your computer. When you run the program, it installs a machine language program in memory and erases its BASIC portion automatically (so be sure to save several copies before running the program for the first time). If you're using a Commodore 128, Plus/4 or 16, do *not* use any GRAPHIC commands while the Proofreader is active. You should disable the Commodore Proofreader before running any other program. To do this, either turn the computer off and on or enter SYS 64738 (for the 64), SYS 65341 (128), SYS 64802 (VIC-20), or SYS 65526 (Plus/4 or 16). To reenable the Proofreader, reload the program and run it as usual. Unlike the original VIC/64 Proofreader, this version works the same with disk or tape.

On the Atari, run the Proofreader to activate it (the Proofreader remains active in memory as a machine language program); you must then enter NEW to erase the BASIC loader. Pressing SYSTEM RESET deactivates the Atari Proofreader; enter PRINT USR(1536) to reenable it.

The Apple Proofreader erases the BASIC portion of itself after you run it, leaving only the machine language portion in memory. It works with either DOS 3.3 or ProDOS. Disable the Apple Proofreader by pressing CTRL-RESET before running another BASIC program.

The IBM Proofreader is a BASIC program that simulates the IBM BASIC line editor, letting you enter, edit, list, save, and load programs that you type. Type RUN to activate. Be sure to leave Caps Lock on, except when typing lowercase characters.

Once the Proofreader is active, try typing in a line. As soon as you press RETURN, either a hexadecimal number (on the Apple) or a pair of letters (on the Commodore, Atari, or IBM) appears. The number or pair of letters is called a *checksum*.

Compare the value displayed on the screen by the Proofreader with the checksum printed in the program listing in the magazine. The checksum is given to the left of each line number. Just type in the program a line at a time (without the printed checksum), press RETURN or Enter, and compare the checksums. If they match, go on to the next line. If not, check your typing; you've made a mistake. Because of the checksum method used, do not type abbreviations, such as ? for PRINT. On the Atari and Apple Proofreaders, spaces are not counted as part of the checksum, so be sure you type the right number of spaces between quote marks. The Atari Proofreader does not check to see that you've typed the characters in the right order, so if characters are transposed, the checksum still matches the listing. The Commodore Proofreader catches transposition errors and ignores spaces unless they're enclosed in quotation marks. The IBM Proofreader detects errors in spacing and transposition.

IBM Proofreader Commands

Since the IBM Proofreader replaces the computer's normal BASIC line editor, it has to include many of the direct-mode IBM BASIC commands. The syntax is identical to IBM BASIC. Commands simulated are LIST, LLIST, NEW, FILES, SAVE, and LOAD. When listing your program, press any key (except Ctrl-Break) to stop the listing. If you enter NEW, the Proofreader prompts you to press Y to be especially sure you mean yes.

Two new commands are BASIC and CHECK. BASIC exits the Proofreader back to IBM BASIC, leaving the Proofreader in memory. CHECK works just like LIST, but shows the checksums along with the listing. After you have typed in a program, save it to disk. Then exit the Proofreader with the BASIC command, and load the program as usual (this replaces the Proofreader in memory). You can now run the program, but you may want to resave it to disk. This will shorten it on disk and make it load faster, but it can no longer be edited with the Proofreader. If you want to convert an existing BASIC program to Proofreader format, save it to disk with SAVE *"filename"*,A.

Program 1: Atari Proofreader

By Charles Brannon, Program Editor

```
100 GRAPHICS 0
110 FOR I=1536 TO 1700:REA
    D A:POKE I,A:CK=CK+A:N
    EXT I
120 IF CK<>19072 THEN ? "E
    rror in DATA Statement
    s.  Check Typing.":END
130 A=USR(1536)
140 ? :? "Automatic Proofr
    eader Now Activated."
150 END
160 DATA 104,160,0,185,26,
    3,201,69,240,7
170 DATA 200,200,192,34,20
    8,243,96,200,169,74
180 DATA 153,26,3,200,169,
    6,153,26,3,162
190 DATA 0,189,0,228,157,7
    4,6,232,224,16
200 DATA 208,245,169,93,14
    1,78,6,169,6,141
210 DATA 79,6,24,173,4,228
    ,105,1,141,95
220 DATA 6,173,5,228,105,0
    ,141,96,6,169
230 DATA 0,133,203,96,247,
    238,125,241,93,6
240 DATA 244,241,115,241,1
    24,241,76,205,238
250 DATA 0,0,0,0,0,32,62,2
    46,8,201
260 DATA 155,240,13,201,32
    ,240,7,72,24,101
270 DATA 203,133,203,104,4
    0,96,72,152,72,138
280 DATA 72,160,0,169,128,
    145,88,200,192,40
290 DATA 208,249,165,203,7
    4,74,74,74,24,105
300 DATA 161,160,3,145,88,
    165,203,41,15,24
310 DATA 105,161,200,145,8
    8,169,0,133,203,104
320 DATA 170,104,168,104,4
    0,96
```

Program 2: IBM Proofreader

By Charles Brannon, Program Editor

```
10 'Automatic Proofreader Vers
   ion 3.0 (Lines 205,206 adde
   d/190 deleted/470,490 chang
   ed from V2.0)
100 DIM L$(500),LNUM(500):COLO
    R 0,7,7:KEY OFF:CLS:MAX=0:
    LNUM(0)=65536!
110 ON ERROR GOTO 120:KEY 15,C
    HR$(4)+CHR$(70):ON KEY(15)
     GOSUB 640:KEY (15) ON:GOT
    O 130
120 RESUME 130
130 DEF SEG=&H40:W=PEEK(&H4A)
140 ON ERROR GOTO 650:PRINT:PR
    INT"Proofreader Ready."
150 LINE INPUT L$:Y=CSRLIN-INT
    (LEN(L$)/W)-1:LOCATE Y,1
160 DEF SEG=0:POKE 1050,30:POK
    E 1052,34:POKE 1054,0:POKE
     1055,79:POKE 1056,13:POKE
     1057,28:LINE INPUT L$:DEF
    SEG:IF L$="" THEN 150
170 IF LEFT$(L$,1)=" " THEN L$
    =MID$(L$,2):GOTO 170
```

```
180 IF VAL(LEFT$(L$,2))=Ø AND
    MID$(L$,3,1)=" " THEN L$=M
    ID$(L$,4)
200 IF ASC(L$)>57 THEN 260 'no
     line number, therefore co
    mmand
205 BL=INSTR(L$," "):IF BL=Ø T
    HEN BL$=L$:GOTO 206 ELSE B
    L$=LEFT$(L$,BL-1)
206 LNUM=VAL(BL$):TEXT$=MID$(L
    $,LEN(STR$(LNUM))+1)
210 IF TEXT$="" THEN GOSUB 54Ø
    :IF LNUM=LNUM(P) THEN GOSU
    B 56Ø:GOTO 15Ø ELSE 15Ø
220 CKSUM=Ø:FOR I=1 TO LEN(L$)
    :CKSUM=(CKSUM+ASC(MID$(L$,
    I))*I) AND 255:NEXT:LOCATE
     Y,1:PRINT CHR$(65+CKSUM/1
    6)+CHR$(65+(CKSUM AND 15))
    +" "+L$
230 GOSUB 54Ø:IF LNUM(P)=LNUM
    THEN L$(P)=TEXT$:GOTO 15Ø
    'replace line
240 GOSUB 58Ø:GOTO 15Ø 'insert
     the line
260 TEXT$="":FOR I=1 TO LEN(L$
    ):A=ASC(MID$(L$,I)):TEXT$=
    TEXT$+CHR$(A+32*(A>96 AND
    A<123)):NEXT
270 DELIMITER=INSTR(TEXT$," ")
    :COMMAND$=TEXT$:ARG$="":IF
     DELIMITER THEN COMMAND$=L
    EFT$(TEXT$,DELIMITER-1):AR
    G$=MID$(TEXT$,DELIMITER+1)
     ELSE DELIMITER=INSTR(TEXT
    $,CHR$(34)):IF DELIMITER T
    HEN COMMAND$=LEFT$(TEXT$,D
    ELIMITER-1):ARG$=MID$(TEXT
    $,DELIMITER)
280 IF COMMAND$<>"LIST" THEN 4
    1Ø
290 OPEN "scrn:" FOR OUTPUT AS
     #1
300 IF ARG$="" THEN FIRST=Ø:P=
    MAX-1:GOTO 34Ø
310 DELIMITER=INSTR(ARG$,"-"):
    IF DELIMITER=Ø THEN LNUM=V
    AL(ARG$):GOSUB 54Ø:FIRST=P
    :GOTO 34Ø
320 FIRST=VAL(LEFT$(ARG$,DELIM
    ITER)):LAST=VAL(MID$(ARG$,
    DELIMITER+1))
330 LNUM=FIRST:GOSUB 54Ø:FIRST
    =P:LNUM=LAST:GOSUB 54Ø:IF
    P=Ø THEN P=MAX-1
340 FOR X=FIRST TO P:N$=MID$(S
    TR$(LNUM(X)),2)+" "
350 IF CKFLAG=Ø THEN A$="":GOT
    O 37Ø
360 CKSUM=Ø:A$=N$+L$(X):FOR I=
    1 TO LEN(A$):CKSUM=(CKSUM+
    ASC(MID$(A$,I))*I) AND 255
    :NEXT:A$=CHR$(65+CKSUM/16)
    +CHR$(65+(CKSUM AND 15))+"
     "
370 PRINT #1,A$+N$+L$(X)
380 IF INKEY$<>"" THEN X=P
390 NEXT :CLOSE #1:CKFLAG=Ø
400 GOTO 13Ø
410 IF COMMAND$="LLIST" THEN O
    PEN "lpt1:" FOR OUTPUT AS
    #1:GOTO 3ØØ
420 IF COMMAND$="CHECK" THEN C
    KFLAG=1:GOTO 29Ø
430 IF COMMAND$<>"SAVE" THEN 4
    5Ø
440 GOSUB 6ØØ:OPEN ARG$ FOR OU
    TPUT AS #1:ARG$="":GOTO 3Ø
    Ø
450 IF COMMAND$<>"LOAD" THEN 4
    9Ø
460 GOSUB 6ØØ:OPEN ARG$ FOR IN
    PUT AS #1:MAX=Ø:P=Ø
470 WHILE NOT EOF(1):LINE INPU
    T #1,L$:BL=INSTR(L$," "):B
    L$=LEFT$(L$,BL-1):LNUM(P)=
    VAL(BL$):L$(P)=MID$(L$,LEN
    (STR$(VAL(BL$)))+1):P=P+1:
    WEND
480 MAX=P:CLOSE #1:GOTO 13Ø
490 IF COMMAND$="NEW" THEN INP
    UT "Erase program - Are yo
    u sure";L$:IF LEFT$(L$,1)=
    "y" OR LEFT$(L$,1)="Y" THE
    N MAX=Ø:LNUM(Ø)=65536!:GOT
    O 13Ø:ELSE 13Ø
500 IF COMMAND$="BASIC" THEN C
    OLOR 7,Ø,Ø:ON ERROR GOTO Ø
    :CLS:END
510 IF COMMAND$<>"FILES" THEN
    52Ø
515 IF ARG$="" THEN ARG$="A:"
    ELSE SEL=1:GOSUB 6ØØ
517 FILES ARG$:GOTO 13Ø
520 PRINT"Syntax error":GOTO 1
    3Ø
540 P=Ø:WHILE LNUM>LNUM(P) AND
     P<MAX:P=P+1:WEND:RETURN
560 MAX=MAX-1:FOR X=P TO MAX:L
    NUM(X)=LNUM(X+1):L$(X)=L$(
    X+1):NEXT:RETURN
580 MAX=MAX+1:FOR X=MAX TO P+1
     STEP -1:LNUM(X)=LNUM(X-1)
    :L$(X)=L$(X-1):NEXT:L$(P)=
    TEXT$:LNUM(P)=LNUM:RETURN
600 IF LEFT$(ARG$,1)<>CHR$(34)
     THEN 52Ø ELSE ARG$=MID$(A
    RG$,2)
610 IF RIGHT$(ARG$,1)=CHR$(34)
     THEN ARG$=LEFT$(ARG$,LEN(
    ARG$)-1)
620 IF SEL=Ø AND INSTR(ARG$,".
    ")=Ø THEN ARG$=ARG$+".BAS"
630 SEL=Ø:RETURN
640 CLOSE #1:CKFLAG=Ø:PRINT"St
    opped.":RETURN 15Ø
650 PRINT "Error #";ERR:RESUME
     15Ø
```

Program 3: Commodore Proofreader

By Philip Nelson, Assistant Editor

```
10 VEC=PEEK(772)+256*PEEK(773)
   :LO=43:HI=44
20 PRINT "AUTOMATIC PROOFREADE
   R FOR ";:IF VEC=42364 THEN
   {SPACE}PRINT "C-64"
30 IF VEC=5Ø556 THEN PRINT "VI
   C-2Ø"
40 IF VEC=35158 THEN GRAPHIC C
   LR:PRINT "PLUS/4 & 16"
50 IF VEC=17165 THEN LO=45:HI=
   46:GRAPHIC CLR:PRINT"128"
60 SA=(PEEK(LO)+256*PEEK(HI))+
   6:ADR=SA
70 FOR J=Ø TO 166:READ BYT:POK
   E ADR,BYT:ADR=ADR+1:CHK=CHK
   +BYT:NEXT
80 IF CHK<>2Ø57Ø THEN PRINT "*
   ERROR* CHECK TYPING IN DATA
    STATEMENTS":END
90 FOR J=1 TO 5:READ RF,LF,HF:
   RS=SA+RF:HB=INT(RS/256):LB=
   RS-(256*HB)
100 CHK=CHK+RF+LF+HF:POKE SA+L
    F,LB:POKE SA+HF,HB:NEXT
110 IF CHK<>22Ø54 THEN PRINT "
    *ERROR* RELOAD PROGRAM AND
    {SPACE}CHECK FINAL LINE":EN
    D
120 POKE SA+149,PEEK(772):POKE
     SA+15Ø,PEEK(773)
130 IF VEC=17165 THEN POKE SA+
    14,22:POKE SA+18,23:POKESA+
    29,224:POKESA+139,224
140 PRINT CHR$(147);CHR$(17);"
    PROOFREADER ACTIVE":SYS SA
150 POKE HI,PEEK(HI)+1:POKE (P
    EEK(LO)+256*PEEK(HI))-1,Ø:N
    EW
160 DATA 12Ø,169,73,141,4,3,16
    9,3,141,5,3
170 DATA 88,96,165,2Ø,133,167,
    165,21,133,168,169
180 DATA Ø,141,Ø,255,162,31,18
    1,199,157,227,3
190 DATA 2Ø2,16,248,169,19,32,
    21Ø,255,169,18,32
200 DATA 21Ø,255,16Ø,Ø,132,18Ø
    ,132,176,136,23Ø,18Ø
210 DATA 2ØØ,185,Ø,2,24Ø,46,2Ø
    1,34,2Ø8,8,72
220 DATA 165,176,73,255,133,17
    6,1Ø4,72,2Ø1,32,2Ø8
230 DATA 7,165,176,2Ø8,3,1Ø4,2
    Ø8,226,1Ø4,166,18Ø
240 DATA 24,165,167,121,Ø,2,13
    3,167,165,168,1Ø5
250 DATA Ø,133,168,2Ø2,2Ø8,239
    ,24Ø,2Ø2,165,167,69
260 DATA 168,72,41,15,168,185,
    211,3,32,21Ø,255
270 DATA 1Ø4,74,74,74,74,168,1
    85,211,3,32,21Ø
280 DATA 255,162,31,189,227,3,
    149,199,2Ø2,16,248
290 DATA 169,146,32,21Ø,255,76
    ,86,137,65,66,67
300 DATA 68,69,7Ø,71,72,74,75,
    77,8Ø,81,82,83,88
310 DATA 13,2,7,167,31,32,151,
    116,117,151,128,129,167,136
    ,137
```

Program 4: Apple Proofreader

By Tim Victor, Editorial Programmer

```
10 C = Ø: FOR I = 768 TO 768 +
   68: READ A:C = C + A: POKE I
   ,A: NEXT
20 IF C < > 7258 THEN PRINT "ER
   ROR IN PROOFREADER DATA STAT
   EMENTS": END
30 IF PEEK (19Ø * 256) < > 76 T
   HEN POKE 56,Ø: POKE 57,3: CA
   LL 1ØØ2: GOTO 5Ø
40 PRINT CHR$ (4);"IN#A$3ØØ"
50 POKE 34,Ø: HOME : POKE 34,1:
    VTAB 2: PRINT "PROOFREADER
   INSTALLED"
60 NEW
100 DATA 216,32,27,253,2Ø1,141
110 DATA 2Ø8,6Ø,138,72,169,Ø
120 DATA 72,189,255,1,2Ø1,16Ø
130 DATA 24Ø,8,1Ø4,1Ø,125,255
140 DATA 1,1Ø5,Ø,72,2Ø2,2Ø8
150 DATA 238,1Ø4,17Ø,41,15,9
160 DATA 48,2Ø1,58,144,2,233
170 DATA 57,141,1,4,138,74
180 DATA 74,74,74,41,15,9
190 DATA 48,2Ø1,58,144,2,233
200 DATA 57,141,Ø,4,1Ø4,17Ø
210 DATA 169,141,96
```

©

Managing Files From Atari ST BASIC

William Sanders

This excerpt from COMPUTE! Books' new title The Elementary Atari ST *demonstrates how to work with the disk system and how to create sequential text files and random access files. A simple program for keeping and updating an address book illustrates various file-handling techniques.*

While programming, you will often find that subroutines which you wrote for one application can be used in subsequent programs with no modifications or with minor changes. ST BASIC provides the MERGE command so that you can transfer these subroutines between programs without retyping them. You have to make sure your subroutines do not contain the same line numbers as other routines with which they will be MERGEd, but, otherwise, MERGE is a simple procedure. For example, enter and save this program:

```
10 CLEARW 2 : FULLW 2
20 FOR X = 1 TO 110 STEP 10
30 NA$ = "NAME# " + STR$(X)
40 GOSUB 200
50 NEXT
60 END
```

Now enter:

```
SAVE "PART1"
```

After you have done that, enter NEW and do the next program.

```
200 REM *****************
210 REM CENTERING ROUTINE
220 REM *****************
230 L = 40 -INT( LEN(NA$)/2 )
240 PRINT TAB(L)NA$
250 RETURN
```

Next enter:

```
SAVE "PART2"
```

You now have two programs saved as files. Neither program will work by itself. If you tried to load them with the LOAD or OLD command, as soon as you loaded the second one, the first one would be wiped out. However, with the MERGE command, you can load them separately. Once they are both loaded, you can run the combined program, and, if you want, you can even save it as a BAS file. Key in this sequence:

```
MERGE "PART1"
MERGE "PART2"
RUN
```

At this point everything should work just fine. Now enter:

```
SAVE "COMBINE"
```

You now have a BASIC file made up of the two combined files. As you collect useful subroutines, you can keep a record of their line number ranges, and it will be possible to write a program simply by MERGEing several subroutines.

Sequential Text Files

Of the two kinds of text files we will discuss, sequential files are simpler to work with. Random access files are a little trickier, but can be accessed faster than sequential files. Using sequential and random access files, it is possible to enter data from a program and store it as a text file. You can add to it, change it, and retrieve data from the file.

Creating sequential files. The first step is to write a *formatting program* which will create a sequential file. To create a file:

```
OPEN "O",File#,"FILENAME"
PRINT# or PRINT# USING or WRITE#
CLOSE
```

These statements take care of everything we need in a sequential file.

Now let's begin writing our address book program, which will store the names of a known number of people who sent us Christmas cards. (Then next Christmas we can check the file to see to whom we should send cards.)

```
10 FULLW 2 : CLEARW 2
20 GOTOXY 1,10 : INPUT "HOW MANY
   ENTRIES";N%
30 DIM NA$(N%) : CLEARW 2
40 FOR X=1 TO N%
50 GOTOXY 1,10 : PRINT "NAME#
   ";X;SPACE$(40)
60 GOTOXY 9,10 : INPUT NA$(X)
70 NEXT X
100 REM ****************************
110 REM PUT DATA INTO SEQUEN-
   TIAL FILE
120 REM ****************************
130 OPEN "O",#1,"XMAS.DAT"
140 FOR X=1 TO N%
150 PRINT#1,NA$(X)
160 NEXT X
170 CLOSE
```

After you enter the program, run it, and remember the number of names you entered. Save the program under the name MAKEFILE; we will come back to it later. Enter *FILES* from BASIC to make sure there is a file called XMAS.DAT that was created by our MAKEFILE program.

The next step is to read our files, using:

```
OPEN "I","FILENAME"
INPUT# or LINE INPUT#
EOF check
CLOSE
```

The next program will OPEN XMAS, INPUT the file, check EOF (end-of-file), CLOSE the file, and then PRINT out the contents to the screen. Notice the similarities and differences between it and our previous program for writing files:

```
10 FULLW 2 : CLEARW 2
20 REM ************************
30 REM READ FROM SEQUENTIAL
   FILE
40 REM ************************
50 OPEN "I",#1,"XMAS.DAT"
60 WHILE NOT EOF(1)
70 INPUT#1,NA$
80 PRINT NA$
90 WEND
100 CLOSE
```

After you run the program, save it under the filename READFILE. Using the EOF function, you do not have to know the number of files you entered. If there are more files, EOF(1) (with 1 being the file number), then the value of NOT EOF(1) will equal 0. If the value of NOT EOF(1) is equal to −1, then the program has found the end-of-file.

Using the WHILE–WEND statement, we check for the case where EOF is *not* true. When this condition is met, the program exits the loop. Notice that as soon as we INPUT# the data from the XMAS file, we printed it to the screen using the normal PRINT statement.

Appending sequential files. So far, so good. We have a program that outputs a list of names into a data file and one that inputs those names back to us. What happens, though, if we want to add some names to our file? Some versions of BASIC have an Append statement along with Open and Input. However, while ST BASIC does not have such a statement, it is a simple matter to append a sequential file. It involves two steps:

1. Count the number of elements in a file and put them into an array.
2. Enter the new elements at the end of the array, and overwrite the old file with the combined data in the array.

```
10 FULLW 2 : CLEARW 2
20 REM ************************
30 REM READ FROM SEQUENTIAL
   FILE
40 REM ************************
50 INPUT "HOW MANY NEW NAMES
   TO ADD";NN
60 OPEN "I",#1,"XMAS.DAT"
70 WHILE NOT EOF(1)
80 INPUT#1,NA$
90 N=N+1
100 WEND
110 CLOSE
200 REM *******************
210 REM LOAD DATA INTO ARRAY
220 REM *******************
230 OPEN "I",#1,"XMAS.DAT"
240 DIM NA$(N+NN)
250 FOR X=1 TO N
260 INPUT#1,NA$(X)
270 NEXT
280 CLOSE
300 REM ***********
310 REM ADD NEW DATA
320 REM ***********
330 FOR X=1 TO NN
340 INPUT "NAME PLEASE";NA$(X+N)
350 NEXT
400 REM ******************
410 REM COMBINE OLD AND NEW
420 REM ******************
430 N%=N+NN
440 OPEN "O",#1,"XMAS.DAT"
450 FOR X=1 TO N%
460 PRINT#1,NA$(X)
470 NEXT
480 CLOSE
```

You can use this method of appending files for simple record keeping. If you're really ambitious, it is not too difficult to edit the array while it is in memory and change the data. However, we will soon be discussing random access files, and the random files are probably better suited for creating files that will require a good deal of manipulation.

Now we've seen how to output and input elements of a single file. However, since filenames are essentially nothing but strings, we could use variables to do much of the work automatically. This next example, "File Manager," will create, append, and read any text file you want. It handles only a single string element, but you can change that if you want. Save the program under the name FILEMAN.

A shortcut. FILEMAN is relatively long, but certain parts of it are very similar to earlier routines we have written. Rather than retyping everything, we will use the MERGE, EDIT, and RENUM functions. First, enter lines 10–120 from FILEMAN below, the MENU block, and save these lines as FILEMAN. Load MAKEFILE, your program to create sequential files. Then enter:

```
RENUM 130,10
```

and then enter:

```
MERGE FILEMAN
```

Compare the listings of previous programs we have written to each block in the following listing. When you find a match, follow the procedure described above to merge the routines into their proper places in FILEMAN as you build your File Manager program.

File Manager

```
10 REM ****
20 REM MENU
30 REM ****
40 FULLW 2 : CLEARW 2 : RESTORE :
   CLEAR
50 FOR X=1 TO 4 : READ CHOICE$
60 GOTOXY 5,X*3 : PRINT X;".
   ";CHOICE$
70 NEXT X
80 GOTOXY 3,17 : PRINT "CHOOSE BY
   NUMBER";
90 A$=INPUT$(1) : A=VAL(A$)
100 ON A GOSUB CREATE,APPEND,
    VIEW, EXIT
```

```
110 DATA CREATE NEW FILE,ADD TO
    FILE,READ FILE,QUIT
120 GOTO 40
130 REM ***************
140 REM CREATE NEW FILE
150 REM ***************
160 CREATE:
170 FULLW 2 : CLEARW 2
180 INPUT "NAME OF FILE";NF$
190 GOTOXY 1,10 : INPUT "HOW
    MANY ENTRIES";N%
200 DIM NA$(N%) : CLEARW 2
210 FOR X=1 TO N%
220 GOTOXY 1,10 : PRINT "NAME#
    ";X;SPACE$(40)
230 GOTOXY 9,10 : INPUT NA$(X)
240 NEXT X
250 REM ******************************
260 REM PUT DATA INTO SEQUEN-
    TIAL FILE
270 REM ******************************
280 OPEN "O",#1,NF$
290 FOR X=1 TO N%
300 PRINT#1,NA$(X)
310 NEXT X
320 CLOSE
330 RETURN
340 REM ********************
350 REM ADD TO EXISTING FILE
360 REM ********************
370 APPEND:
380 FULLW 2 : CLEARW 2
390 REM **********************
400 REM READ FROM EXISTING FILE
410 REM **********************
420 INPUT "NAME OF FILE TO
    APPEND";NF$
430 INPUT "HOW MANY NEW NAMES
    TO ADD";NN
440 OPEN "I",#1,NF$
450 WHILE NOT EOF(1)
460 INPUT#1,NA$
470 N=N+1
480 WEND
490 CLOSE
500 REM ********************
510 REM LOAD DATA INTO ARRAY
520 REM ********************
530 OPEN "I",#1,NF$
540 DIM NA$(N+NN)
550 FOR X=1 TO N
560 INPUT#1,NA$(X)
570 NEXT
580 CLOSE
590 REM ************
600 REM ADD NEW DATA
610 REM ************
620 FOR X=1 TO NN
630 INPUT "NAME PLEASE";NA$(X+N)
640 NEXT
650 REM ******************
660 REM COMBINE OLD AND NEW
670 REM ******************
680 N%=N+NN
690 OPEN "O",#1,NF$
700 FOR X=1 TO N%
710 PRINT#1,NA$(X)
720 NEXT
730 CLOSE
740 RETURN
750 REM ************************
760 REM READ FROM SEQUENTIAL
    FILE
770 REM ************************
780 VIEW:
790 FULLW 2 : CLEARW 2
800 INPUT "FILE TO READ";NF$
810 OPEN "I",#1,NF$
820 WHILE NOT EOF(1)
830 INPUT#1,NA$
840 PRINT NA$
850 WEND
860 CLOSE
870 PRINT :PRINT "HIT ANY KEY TO
    RETURN TO MENU"
880 W$=INPUT$(1)
890 RETURN
900 REM ************
910 REM QUIT PROGRAM
920 REM ************
930 EXIT:
940 END
```

Hand Me A Line

LINE INPUT and LINE INPUT # can be very handy commands for reading and writing sequential files. For example, let's say you want to enter a name, address, and phone number into an array, store the array on disk, and later read it back. With LINE INPUT, it is possible to use a single string or string array variable to put all that information in at once. Likewise, when retrieving information from the disk, you can get a whole line by using LINE INPUT #.

This is especially useful when you are reading a file with an unknown format. For example, let's say that you want to read the contents of a disk, but don't know whether it is composed of strings or numeric values, and you don't know their order. By using LINE INPUT # and a string variable, you can read the file line by line rather than variable by variable.

To see how LINE INPUT works, enter the following program. When you run it, be sure to include commas between the name, address, and phone number. Unlike the INPUT statement, commas entered from the keyboard when using LINE INPUT will not result in an error message.

```
10 REM **********
20 REM LINE INPUT
30 REM **********
40 FULLW 2 : CLEARW 2
50 GOTOXY 5,5 : INPUT "HOW MANY
   ENTRIES";N%
60 DIM NAP$(N%)
70 CLEARW 2
80 FOR X = 1 TO N%
90 LINE INPUT "Name, Address, Phone
   ";NAP$(X)
100 NEXT X
200 REM **********************
210 REM PRINT RESULTS TO SCREEN
220 REM **********************
230 CLEARW 2
240 FOR X = 1 TO N%
250 PRINT NAP$(X)
260 NEXT
```

The program does not do anything with files, but it would be a simple matter to have it PRINT # to the disk instead of to the screen. Change the block beginning at line 200 to write the file to disk. The name, address, and phone are in one string with the delimiters preserved.

Now, we're going to write information to disk using several variables, and then, using LINE INPUT #, we are going to read the disk with a single string variable. This will show you how to read a line of variables that were stored either as separate variables or as a single LINE INPUTed variable.

```
10 FULLW 2 : CLEARW 2
20 GOTOXY 5,5 : INPUT "HOW MAN
   Y ENTRIES";N%
30 GOTOXY 5,5 : PRINT SPACE$(40): DI
   M NA$(N%),AD$(N%),PH$(N%)
40 FOR X=1 TO N%
50 INPUT "NAME";NA$(X)
60 INPUT "ADDRESS";AD$(X)
70 INPUT "PHONE";PH$(X)
80 NEXT X
100 REM **************
110 REM OUTPUT TO DISK
120 REM **************
130 OPEN "O",#1,"NAMEAD"
140 FOR X=1 TO N%
150 PRINT#1,NA$(X);",";AD$(X);",";PH$(
    X)
160 NEXT X
170 CLOSE
200 REM ********************
210 REM READ WITH LINE INPUT#
220 REM ********************
230 CLEARW 2 : GOTOXY 5,5 : PRINT "Hi
    t any key to continue";
240 AN$=INPUT$(1) : CLEARW 2
250 OPEN "I",#1,"NAMEAD"
260 ON ERROR GOTO 300
270 LINE INPUT#1,NAP$
280 PRINT NAP$
290 GOTO 270
300 CLOSE
310 LOOK$=INPUT$(1)
320 END
```

As you saw when the program executed, the variables, along with their printed format established in line 150, were read and displayed with a single string variable. This is where LINE INPUT # can save time and guessing. Of course, it would have been even simpler to use LINE INPUT when we entered the information originally, but the program was designed to show you how LINE INPUT # works when reading files created with several variables.

We also introduced another way to determine the end of a file. While the EOF statement is the preferred method, you can also use ON ERROR GOTO to jump out of an error. When the *end-of-file* error occurs, the program jumps to the line that CLOSEs the file. Be careful in using ON ERROR GOTO, because there will be times when some bug in your program will cause an error rather than the error condition you intended to trap for.

PRINT # USING And Files

A final way to store information on disks is with PRINT # USING, which sends data to the disk much like the PRINT USING statement outputs to the screen. The format is slightly different, but the statement works essentially the same way. PRINT # USING is very handy in programs which process formatted numeric data:

```
10 REM ******************
20 REM ENTER NUMERIC DATA
30 REM ******************
40 FULLW 2 : CLEARW 2
50 GOTOXY 5,5 : INPUT "HOW MAN
   Y ENTRIES";N%
60 GOTOXY 5,5 : PRINT SPACE$(20) : DI
   M AMOUNT(N%)
70 FOR X=1 TO N%
80 GOTOXY 10,5 : PRINT SPACE$(2
   0) : GOTOXY 5,5 :INPUT "HOW MUC
   H";AMOUNT(X)
90 NEXT X
100 REM *****************************
    ***
110 REM WRITE TO DISK WITH PRIN
    T #, USING
120 REM *****************************
    ***
130 GOTOXY 5,5 : PRINT SPACE$(20)
140 OPEN "O",#1,"EXPENSES"
150 FOR X=1 TO N%
160 TOTAL=TOTAL+AMOUNT(X)
170 PRINT #1, USING "####.##";AM
    OUNT(X),TOTAL
180 NEXT X
190 CLOSE
200 PRINT "Hit any key" : LOOK$=INP
    UT$(1)
```

When you read your file, all of the data will be formatted for you. Instead of using the variables you originally employed, use LINE INPUT #. Thus, the following program will read and display your information as you wrote it to disk:

```
300 OPEN "I",#1,"EXPENSES"
310 ON ERROR GOTO 350
320 LINE INPUT#1,EXPENSES$
330 PRINT EXPENSES$
340 GOTO 320
350 CLOSE
360 PRINT "Hit any key" : LOOK$=
    INPUT$(1)
```

Random Access Files

Random access files are like containers of equal size into which you store data. You first decide how big a container you will need, based on the maximum size of the material you will be putting in the box. Each character in a string takes one byte. Therefore, if your maximum length for a given string is ten, it will be necessary to allocate a total of ten bytes. With numbers, storage is different. Here's a chart for quick reference on how much memory space to allocate for the different kinds of data:

Type	Allocation
String	1 byte per character
Integer	2 bytes per number
Single-precision	4 bytes per number
Double-precision	8 bytes per number

All entries into a random access file must be in string format, including numbers; we will examine the functions for doing that later. For now, we will concentrate on entering data as normal strings.

For the most part, the process of creating and reading random access files looks very much like sequential files, but there are important differences. For instance, when you OPEN a random access file, you must include the length of the file. First, as we did with sequential files, we OPEN the file and place the filename in quotes. However, instead of writing the mode, we indicate the file number and the length of our file. Here is the format:

```
OPEN "R",#1,"NAMEFI",128
```

With this statement we can either write to the disk file or read from it. Unlike with sequential files, we do not indicate whether the mode is output or input when we OPEN a random access file.

Random access files can be undivided or divided. Undivided files use the same length for every entry. For the most part, it is pointless to use undivided files unless you are entering a single field, such as a list of names with no other information, or when you put all the information into a single string as we did with LINE INPUT #. It is more useful to divide random access files into sections called *fields*, with each field having a maximum length. The FIELD statement expects a file number, width, and string variable:

```
FIELD #1, 20 AS A$, 10 AS B$, 2 AS C$
```

The above statement sets the width of A$ to 20, B$ to 10, and C$ to 2. When the file is OPENed, the LENgth value (the last value entered in the OPEN statement) must equal the sum of the FIELD values. In the above example, the length must be 20 + 10 + 2 = 32. When OPENing the R file (R is used for both input and output), the last value would be 32.

```
OPEN "R",#1,"FILENAME",32
```

To illustrate using random access files, let's modify our address book program. We will call the file we create HOMETOWN, using three strings. Before we can enter the data into a random access file, we have to use the LSET statement to store our records in their respective fields. Moreover, the variable names we LSET cannot be the same ones we INPUT. Therefore, we have two sets of variables, one for INPUT and one for LSET. The nice thing about LSET is that it automatically pads the strings with sufficient spaces to fit the field exactly, or it truncates the string if it is too long.

Here is a list of our variable names:

NA$ for a person's name	LSET = N$
CT$ for the city's name	LSET = C$
SC$ for the state's mailing code	LSET = S$

Since we'll be dealing with the names of people and cities and thus the fields will be of differing lengths, we'll have to decide on a maximum-size name. Longer names will be truncated to this specified size. This process is extremely important in working with random access files since we are limited to the number of bytes specified when we OPEN a file. Without the truncate feature, entries over the maximum length would spill over into the next record. Therefore, we will limit the length of a name to 20, a city to 10, and states to the two-character abbreviations employed by the post office:

N$	= 20
C$	= 10
S$	= 2
Total	= 32

Using these values, we can now write a program to enter a single record into a random access file:

```
10 CLEARW 2 : NR%=1
20 GOTOXY 1,4 : INPUT "NAME";NA$
30 GOTOXY 1,6 : INPUT "CITY";CT$
40 GOTOXY 1,8 : INPUT "STATE CODE";SC$
100 REM ******************
110 REM WRITE SINGLE RECORD
120 REM ******************
130 OPEN "R",#1,"HOMETOWN",32
140 FIELD #1, 20 AS N$,10 AS C$,2 AS S$
150 LSET N$=NA$
160 LSET C$=CT$
170 LSET S$=SC$
180 PUT #1,NR%
190 CLOSE
200 END
```

That was a lot of work to enter one simple record, but be patient and we will do more. Now, we will GET# a record from a random access file. As in writing to random access files, we must OPEN the file with a specified length and read it in terms of a specified record. The following program will read record 1 in the HOMETOWN file:

```
10 CLEARW 2 : NR%=1
20 REM ******************
30 REM READ SINGLE RECORD
40 REM ******************
50 OPEN "R",#1,"HOMETOWN",32
60 FIELD #1, 20 AS N$,10 AS C$,2 AS S$
70 GET #1,NR%
80 PRINT N$:PRINT C$;",";S$
90 CLOSE
100 END
```

We had to write quite a lot just to write and read a single record, but this illustrates how random access files operate. Now we can deal with multiple records with our HOMETOWN example.

Our next task is to create a *sequential* file to keep track of our pointers in the random access file. Basically, a pointer routine will check the sequential file and tell us which record number was the last one we wrote, and then it will move the pointer to the next record number. For instance, if there are ten records in a random access file, we want the pointer 10 to be stored somewhere we can easily get it. When we want to add to a random access file, we can then find the value 10, add 1 to it, and begin writing our record at position 11. We will call this file HOMEPOINT.

```
10 CLEARW 2: GOTOXY 1,1
20 INPUT "How many new entries";NE%
30 FOR X=1 TO NE%
40 GOTOXY 4,4 : PRINT SPACE$(25) : GOTOXY 1,4 : INPUT "NAME";NA$
50 GOTOXY 4,6 : PRINT SPACE$(25) : GOTOXY 1,6 : INPUT "CITY";CT$
60 GOTOXY 7,8 : PRINT SPACE$(25) : GOTOXY 1,8 : INPUT "STATE CODE";SC$
70 IF X=1 THEN GOSUB 100 ELSE GOSUB 200
80 NEXT X
90 GOTO 400
100 REM *****************
110 REM FIND LAST POINTER
120 REM *****************
130 OPEN "I",#1,"POINT"
140 INPUT #1,POINTER%
150 CLOSE
200 REM *****************
210 REM WRITE RANDOM FILE
220 REM *****************
230 POINTER%=POINTER%+1
240 OPEN "R",#2,"HOMETOWN",32
250 FIELD #2, 20 AS N$,10 AS C$,2 AS S$
260 LSET N$=NA$
270 LSET C$=CT$
280 LSET S$=SC$
290 PUT #2,POINTER%
300 CLOSE
310 RETURN
400 REM **************
410 REM UPDATE POINTER
420 REM **************
430 OPEN "O",#1,"POINT"
440 PRINT#1,POINTER%
450 CLOSE
460 END
```

The first time you run this program, you must *initialize* the POINT file. To do this, enter GOTO 400 the first time you run the program.

Now that we have several records in our file, we will need a way to get them out again. Here's where our counter variable POINTER comes in handy. First, we will read POINTER to see how many records there are and then loop through the records to GET# them all. Notice that in line 60, we first INPUT#1 POINTER and after INPUTing it into memory, it is used in the FOR-NEXT loop in line 150 to pull all the records out.

```
10 FULLW 2 : CLEARW 2
20 REM *****************
30 REM FIND LAST POINTER
40 REM *****************
50 OPEN "I",#1,"POINT"
60 INPUT #1,POINTER%
70 CLOSE
100 REM *****************
110 REM READ RANDOM FILE
120 REM *****************
130 OPEN "R",#2,"HOMETOWN",32
140 FIELD #2, 20 AS N$,10 AS C$,2 AS S$
150 FOR X=1 TO POINTER%
160 GET #2,X
170 PRINT N$:PRINT C$;",";S$:PRINT
180 NEXT
190 CLOSE
200 LOOK$=INPUT$(1)
```

By adding a few lines and calculating a few more bytes, you can expand our example program into a very useful, customized address list. The program already enters names, cities, and states. All you have to add are addresses and zip codes, and there you have it. By attaching a subroutine to send it to your printer, you could generate your own mailing list program.

More File-Handling Commands

Most file applications deal with strings, but there are many applications which require the use of numbers instead. Instead of using STR$ to convert numbers into strings, we can use MKI$, MKS$, and MKD$. The *I*, *S*, and *D* in the commands stand for Integer, Single-precision, and Double-precision number conversions. Unfortunately, these conversions translate your numbers into ASCII code, so you have to convert them back to numbers using CVI, CVS, and CVD before you use them in arithmetic operations.

Let's see how they are used with files. First, create a numeric variable TOTAL. For a single-precision number, TOTAL would take four bytes, or two for integers and eight for double precision. We will call our string SUM$, so we would define our FIELD as:

```
FIELD #2, 4 AS SUM$
```

Then with LSET, we would put:

```
LSET SUM$ = MKS$(TOTAL)
```

Finally, once we read SUM$ from our file, if we want to reconvert it to a numeric variable, we would enter:

```
TOTAL = CVS(SUM$)
```

By making the conversions to and from string and numeric variables, we can store strings in random access files, yet use numeric variables in programs where the values are used as real numbers. This applies only to random access files, for we saw how we could store and update numeric variables in sequential files with no conversions.

There is a great deal more you can do with files; this introductory look at them just scratches the surface. It is possible to make database systems that search for individual records, change individual records, sort records, and more. ©

BASIC Equivalents In C

Harley M. Templeton

One of the hottest programming topics these days is the C language. C is the language of choice for many professional programmers because it's easy to write and produces compact, efficient machine language code. Another plus is that C programs are easy to transport from one computer to another. For those who may be new to C, here's an article that describes similarities between BASIC and C. It's excerpted from a chapter in From BASIC To C, *currently available from COMPUTE! Books.*

From a beginner's viewpoint, one of the reassuring aspects of the C language is that it has many things in common with BASIC. You can write a large part of your C program using statements that are just like or very similar to BASIC statements. Of course, they have no line numbers, are written with lowercase letters, and end with a semicolon. But they use the same keywords as BASIC statements, and perform the same or nearly the same operations. These are the C language statements that are equivalent to BASIC statements:

1. Assignment statement
2. if statement
3. for loop
4. while loop
5. goto statement

Assignment Statement

The *assignment statement* assigns a value to a variable. The value may be that of a constant, another variable, an expression, or a function. An assignment statement in BASIC is:

```
100 ITEM = 4875
```

The C equivalent is:

```
item = 4875;
```

Same thing ... almost. BASIC is very secretive about the *type* of a variable. ITEM is automatically assigned single-precision type, and the constant 4875 is assigned integer type. The integer is converted to single precision and stored as a single-precision variable.

C does not assign a type to a variable automatically. The example shown would result in an error message unless it had been preceded by a *declarator*. This particular declarator assigns type **float** to variable *item*:

```
float item;
```

C's **float** variable is identical to the single-precision variable in BASIC. It is a real number providing six to seven digits of precision. The constant is assigned integer type in C also. It's converted to **float** type and stored as the value of *item*.

The statements are the same in both languages. The difference is that C requires you to declare variables, which shows you at a glance which variables you're using and what type they are. Automatic assignment of types in BASIC sometimes provides strange answers; specific declaration of variables in C puts you in the driver's seat.

C, like some versions of BASIC, includes a type of assignment statement that assigns the same value to more than one variable:

```
sum1 = sum2 = sum3 = 0;
```

This statement assigns the value of 0 to variable *sum3* first. Then it assigns the value of *sum3* to variable *sum2*. Last of all, it assigns the value of *sum2* to variable *sum1*. You can use any variable or expression to the right of the rightmost equal sign (instead of the 0).

Here's another example in BASIC:

```
380 AVE = (VAL1 + VAL2 + VAL3) / 3
```

The C language statement is:

```
ave = (val1 + val2 + val3) / 3;
```

The BASIC statement adds three single-precision values, converts the integer 3 to single precision, and performs single-precision division. The result is stored as a single-precision variable.

Assuming that *val1, val2, val3,* and *ave* have been declared as **float** type, the C statement provides the same result, but in a different way. A C program performs no **float** type computations. Instead, the program that contains this statement converts *val1, val2, val3,* and integer 3 to type **double**, and performs the computations. Type **double** is a real-number type that provides 16 to 17 digits of precision. When it's converted and stored, the result is more accurate (potentially, at least) than if the result had been type **float**.

If *val1, val2,* and *val3* are declared as integers, you should use a **float** type constant:

```
ave = (val1 + val2 + val3) / 3.0;
```

The program adds the integers, converts the sum and constant to type **double**, performs the computations, and converts the result to **float**. Using 3 instead of 3.0 would have caused all the numbers to be treated as integers, and an integer result would have been converted to **float** and stored. This would probably not be accurate enough.

Here's a more complex BASIC statement that includes a function:

```
500 SIDE1 = SIN(A) * HYP
```

The C equivalent is:

```
side1 = sin(a) * hyp;
```

The BASIC example calls the SIN function. Some versions of the interpreter return a double-precision result, but most return a single-precision result. The result is a single-precision value in variable SIDE1.

In C, the result is the same, but the computations are **double** type. The function, however, is in one of the libraries that came with your C compiler. The statement causes the compiler to ask the link program to get the function from the library and include it in your program. The advantage is that you don't have to use the C compiler or libraries each time you run the program. The library function becomes as much a part of your program as the functions you write and compile.

C assignment statements are very much like BASIC assignment statements. All you need to do is omit the line number, change the letters to lowercase, and add a semicolon.

if Statement

The C **if** statement never needs a "then" and cannot transfer control to another part of the program. It uses parentheses around the relational or logical expression. Otherwise, it is the BASIC IF in lowercase letters and ending with a semicolon. Here's an example in BASIC:

```
450 IF YEAR MOD 4 THEN FEB = 28
    ELSE FEB = 29
```

In C, you'll need four statements:

```
if (year % 4)
   feb = 28;
else
   feb = 29;
```

Both versions use modulo division to identify leap years. The percent sign (%) is the modulo division operator in C. When the result is not zero, the variable *feb* is set to 28. When the result is zero, *feb* is set to 29. Like BASIC, C considers a zero value as false and a nonzero value as true. C replaces the BASIC THEN by using parentheses. Whatever comes after the closing parentheses is considered to be the statement to be executed if the expression is true. Now for the bad news. C has no equivalent for this BASIC statement:

```
500 IF YEAR < 1984 THEN 600 ELSE 650
```

But the news is really not that bad, because C has a better way to do the same thing. In BASIC, line 600 and lines following are statements to be executed for years prior to 1984. Lines 650 and following are statements that apply to subsequent years. In C, you can put those statements right in the **if** statement:

```
if (year < 1984) {
   rate = .25;
   base = 2500;
   surcharge = .50;
   }
else {
   rate = .26;
   base = 2000;
   surcharge = .52;
   }
```

Whatever it is, it went up in 1984. The important thing to notice is the left brace following the parenthesis. This brace is the beginning of a *compound statement,* or *block,* that is executed for years prior to 1984. The right brace ends the block. All the statements you need for years prior to 1984 go right here instead of somewhere else in your program. Similarly, a block following **else** contains the statements for years 1984 and later—all right here together, where you can't miss them.

In BASIC, you can leave off the ELSE when you don't need it:

```
400 IF YEAR > 1983 THEN RATE =
    RATE + .01:BASE
    = BASE - 500: SURCHARGE =
    SURCHARGE + .02
```

You can in C, too:

```
if (year > 1983) {
   rate = rate + .01;
   base = base - 500;
   surcharge = surcharge + .02;
   }
```

You would have to set *rate, base,* and *surcharge* to the values that apply before 1984, or this statement would not give you the same answer. If you set the variables to the 1983 values (with assignment statements), this statement is more efficient in either language. In either version, the statement does nothing if the year is 1983 or earlier.

C's **if** statement works like BASIC's IF statement, and it can make your program more readable by including blocks of statements that otherwise would be in some other part of the program.

for Loop

C's **for** loop is much more versatile than BASIC's FOR-NEXT loop.

The BASIC loop starts with a FOR statement and ends with a NEXT:

```
800 SQ = 1: ODD = 1
810 FOR R = 1 TO 15
820 PRINT SQ, R
830 ODD = ODD + 2
840 SQ = SQ + ODD
850 NEXT R
```

The C loop has no need for NEXT:

```
sq = odd = 1;
for (r = 1; r <= 15; r = r + 1) {
   printf("%d    %d\n",sq, r);
   odd = odd + 2;
   sq = sq + odd;
   }
```

Notice that the C statement consists of keyword **for** followed by three expressions enclosed in parentheses and separated by semicolons. The first expression initializes *r;* it corresponds to the FOR R = 1 portion of the BASIC statement. The next expression is evaluated before each repetition of the statements in the loop. This corresponds to the TO 15 portion of the BASIC statement. The third expression is executed for each repetition of the loop, after the last statement. This expression corresponds to the STEP 1 option implied in the BASIC statement.

The loop itself is a block, described in the section on the **if** statement. The loop could consist of a single statement. The statements in this block print 15 perfect squares and their roots without multiplying or calling a square root function. How about that?

Whether the loop consists of a single statement or a compound statement (block), no NEXT statement is needed. Either the semicolon that ends the statement or the brace that encloses the block tells the compiler what belongs in the loop.

The **for** statement of C does not look like the FOR statement of BASIC, but **for** statements like the one shown in this section work exactly like BASIC FOR-NEXT loops.

while Loop

The C **while** loop is similar to the WHILE–WEND loop of BASIC. The C version doesn't need the WEND statement for the same reason that the **for** loop does not need a NEXT. Here's an example from BASIC:

```
250 WHILE N <> 21
260 INPUT N
270 N = N + 5
280 WEND
```

The C version is:

```
while (n != 21) {
   scanf("%d",n);
   n = n + 5;
   }
```

The **while** keyword is followed by an expression within parentheses. In the example, the expression is a relational one, having a true value (1) or a false value (0). Notice that **!=** means not equal to. The expression could be a numeric expression; in that case, it is considered true when its value is not equal to zero. A zero value is considered false. In either case, the loop is executed as long as the expression is true.

In the C version, the statements of the loop are enclosed in braces and include the **scanf( )** function to accept a decimal number that you type in. These statements are executed until the user types 21; then the loop terminates. In either language, it could happen that typing 21 would not get you out of the loop. This would happen if variable *n* were a real number type and the automatic type conversions introduced a fractional result (21.00001, for example). Declare *n* as an integer to avoid this problem.

Notice the contrast between **for** and **while** loops. The **for** loop initializes, tests, and increments. The **while** loop only tests. The loop must include some way for the variable to acquire the value required for the test. Otherwise, the loop is repeated until you get tired of it and turn off your computer. Here's the **while** loop version of the **for** loop example:

```
r = sq = odd = 1;
while (r <= 15) {
   printf("%d    %d\n",sq, r);
   odd = odd + 2;
   sq = sq + odd;
   r = r + 1;
   }
```

The initialization expression moved to the statement preceding the loop, the incrementing expression moved into the loop, and the keyword changed. Otherwise, the examples are identical.

goto Statement

BASIC's GOTO statement allows you to write unmanageable programs, yet it is unavoidable in many BASIC programs. C's **goto** statement is seldom required. You should avoid using the **goto** statement to keep your programs understandable.

Here's the culprit in BASIC:

```
400 GOTO 450
```

In C:

```
goto there;
```

Since C has no line numbers, the **goto** statement has to be different, but it works the same. However, the program must include a statement with "there" as its label:

```
there:year = year + 1; /* You may
   label any statement */
```

Leave the **goto** statement for use only in dire emergencies.

A Program Example

You can program solutions for many problems using only these statements along with input/output functions. Program 1 demonstrates this with a very simple checkbook balancer.

Notice the first line of the program. It is a *compiler control line*, required for the input/output statements in the example. The number sign (#) of a compiler control line must be in column 1 of the line, the leftmost character position.

The input/output functions required are **printf( )**, which displays prompting messages on the screen, **scanf( )**, which accepts the numbers you type in, and **getche( )**, which accepts a single character from the keyboard and displays it on the screen. The other executable statements, except **return**, are all like BASIC statements. The **return** statement just returns control to the operating system.

The program first asks for the balance and stores it after you type it in. Then it asks if there are any outstanding deposits. Using the familiar **if** statement and the character input function, the program accepts the first character you type and requests a deposit amount if you have typed Y. After displaying the prompting message, the program stores the deposit value and enters a **while** loop. Until you type 0 for the deposit amount, the loop is repeated, adding the deposit to the balance and getting another deposit value.

When you type any letter other than Y (and that includes y), the program skips to process the checks, subtracting each check from the previous balance. When you type 0 (for no more checks), the program displays the balance.

The program is not very user-friendly, because it skips the processing of deposits if you type any letter but Y. It should at least recognize y. It could reject all letters but Y, y, N, and n, telling you to try again.

This program shows that it's possible to program solutions to common problems using only the statements described in this article. But you can do a lot more with C: things that are done differently from BASIC and things you cannot do in BASIC.

Program 1: Checkbook in C

```
#include <stdio.h>
main () /* Balance your checkbook? */
{
double bal, chk, dep;
char in[12];
int c;

printf("Type in statement balance:"); /* Display function */
scanf("%lf", &bal); /* Formatted input function */
printf("Any outstanding deposits? (Y or N) ");
if ((c = getche()) == 'Y') { /* Character input function */
   printf("\nType outstanding deposit: $");
   scanf("%lf", &dep);
   while (dep > 0) {
      bal = bal + dep;
      printf("Type outstanding deposit: $");
      scanf("%lf", &dep);
      }
   }
printf("\nType outstanding check: $");
scanf("%lf", &chk);
while (chk > 0) {
   bal = bal - chk;
   printf("Type outstanding check: $");
   scanf("%lf", &chk);
   }
printf("Your checkbook balance should be $%.2lf",bal);
return;
}
```

©

Hard Disk For Atari ST

Hard disks for the Atari 520ST and 1040ST in 10-, 20-, 30, and 60-megabyte configurations have been introduced by Supra Corporation. The SupraDrive system connects to the computer's high-speed DMA port and can significantly improve disk transfer speeds. The drive is compatible with TOS and will work with other standard DMA bus peripherals. The ST can be booted directly from the hard disk.

The SupraDrive system, which ranges in price from $799 for the 10 mb system to $1,995 for the 60 mb unit includes format, backup, and partition utilities that allow the user to created up to four separate logical drives for file storage.

Supra Corporation, 1133 Commercial Way, Albany, OR 97321.

Circle Reader Service Number 218.

Supra's hard disk for the Atari ST is available in four sizes.

New Tools For Amiga

Brown-Wagh Publishing has released three new productivity tools for the Amiga, designed by Micro-Systems Software Inc. of Florida.

Analyze! is a spreadsheet program that can be used for financial analysis and planning, bookkeeping, home budgets, check registers, and professional-sized spreadsheets.

Using *OnLine!*, a telecommunications program, you can link up with commercial information services, send Telex messages and electronic mail, and exchange data with other computers. This program comes equipped with user-defined macrokeys to transmit frequently used commands and script files for automated operation.

The electronic bulletin board system, *BBS-PC*, interfaces to a hard disk and a 2400 bps modem. It enables other users to call your Amiga and read messages, leave you messages, send you a file, or take a file you have left for them. *BBS-PC* works in the background so the Amiga can answer the phone while other users are working on their projects.

Analyze! and *BBS-PC* retail for $99.95, and *OnLine!* retails for $69.95.

Brown-Wagh Publishing, 100 Verona Ct., Los Gatos, CA 95030.

Circle Reader Service Number 219.

ST Graphic Arts

Progressive Computer Applications, Inc., has announced *The Graphic Artist*, a graphic arts package for the Atari ST. This software package combines computer-aided design, typesetting, spreadsheet, and word processing capabilities for use with the color graphics features of the ST.

The Graphic Artist Language FiG-GAL is an auxiliary package which allows the user to create complex custom applications for *The Graphic Artist*. It offers if-then logic, branching, looping, and variables.

Suggested retail price for *The Graphic Artist* is $495.00, and $245.00 for *The Graphic Artist Language*.

Progressive Computer Applications, Inc., 2002 McAuliffe Dr., Rockville, MD 20851.

Circle Reader Service Number 220.

Adventure Game New For The Mac, ST

Search Transylvania for Princess Sabrina with Penguin Software's graphic adventure game *Transylvania* in a new Macintosh version. You must look through the forest and castle, encounter creatures, and put the clues together in order to find and rescue the princess.

Transylvania now uses the new *Comprehend* advanced parser—the portion of the adventure that analyzes your commands—as well as colorful graphics. It is also available for the Atari ST, Apple II (64K required), and Commodore 64.

The Macintosh version retails for $39.95.

Penquin Software, 830 Fourth Ave., P.O. Box 311, Geneva, IL 60134.

Circle Reader Service Number 221.

Educational Software

Learning Technologies has released ten educational software packages for pre-kindergarten through grade six. All software is compatible with the Apple II-series and Commodore 64 and 128.

For preschool through grade two:

Animal Hotel develops specific recall, visual memory, visual discrimination, and analysis of the whole.

Bike Hike develops specific recall and visual memory, number recognition, and counting and visual discrimination.

Lion's Workshop also develops skills in visual discrimination as well as pattern recognition and analysis of part-whole relationships.

Visual discrimination, matching, observation, and deductive reasoning skills are developed in *Same or Different*.

Both *Shutterbug's Pictures* and *Shutterbug's Patterns* develop skills in visual discrimination and analysis of part-whole relationships, while *Patterns* adds pattern recognition.

For ages eight and above:

Number Please develops specific recall and sequential memory.

Thinking skills such as observing details, comparing and contrasting, classifying, defining a problem, determining a solution, and evaluating outcomes are developed in *Gremlin Hunt*.

Pipeline teaches such thinking skills as defining a problem, experimenting with possible solutions, evaluating outcomes, recognizing patterns, and determining part-whole relationships.

For math instruction for preschool through sixth grade:

To receive additional information from advertisers in this issue, use the handy reader service cards in the back of the magazine.

Math in a Nutshell helps users develop skills in counting, addition, subtraction, multiplication, and division of single-digit numbers.

Each program retails for $19.95 and includes a redemption card for a free *Learning Kit*. The *Learning Kit* includes a color poster, a custom lesson plan, worksheets, a progress chart, and award certificates.

Learning Technologies, 4255 LBJ, Suite 265, Dallas, TX 75244.

Circle Reader Service Number 222.

In Pursuit Of Computer Trivia

King Chip, from XYLYX Computer Entertainment Limited, is a board game designed to test your knowledge in many areas of the computer industry. Similar to *Trivial Pursuit* in format, it contains more than 4,000 questions in six categories: data communications, history and current events, hardware, jargon, and acronyms, potpourri, and software. Questions in each category can be selected from five levels of difficulty, and vary in format among multiple choice, fill-in-the blank, true or false, and one- or two-word answers.

The object of the game is to answer enough questions correctly so that you can attain the throne of King Chip. Once there, you'll have to keep answering correctly to maintain it.

What sets *King Chip* apart from many other trivia-type games is its depth of responses. The back of each question card contains not only each correct answer but, where appropriate, an explanation of the answer. So it's a bit of a tutorial as well as just a game.

King Chip retails for $39.95.

XYLYX Computer Entertainment Limited, 20 Torbay Rd., Markham, Ontario, Canada L3R 1G6.

Circle Reader Service Number 223.

Apple II, Commodore, IBM Telecommunications Package

The Information Connection, from Grolier Electronic Publishing, combines a telecommunications program, text editor, and tutorial software on one disk for the Apple II-series, IBM PC or PCjr, and Commodore 64 or 128 in 64 mode. This package for beginners teaches the fundamentals of telecommunications and features a simulated on-line practice session. There is also an alarm and automatic shut-off to help control telecommunications costs.

Apple and IBM versions retail for $59.95 and the Commodore 64 version costs $39.95.

Grolier Electronic Publishing, Inc., 95 Madison Ave., New York, NY 10016.

Circle Reader Service Number 224.

World Series Baseball

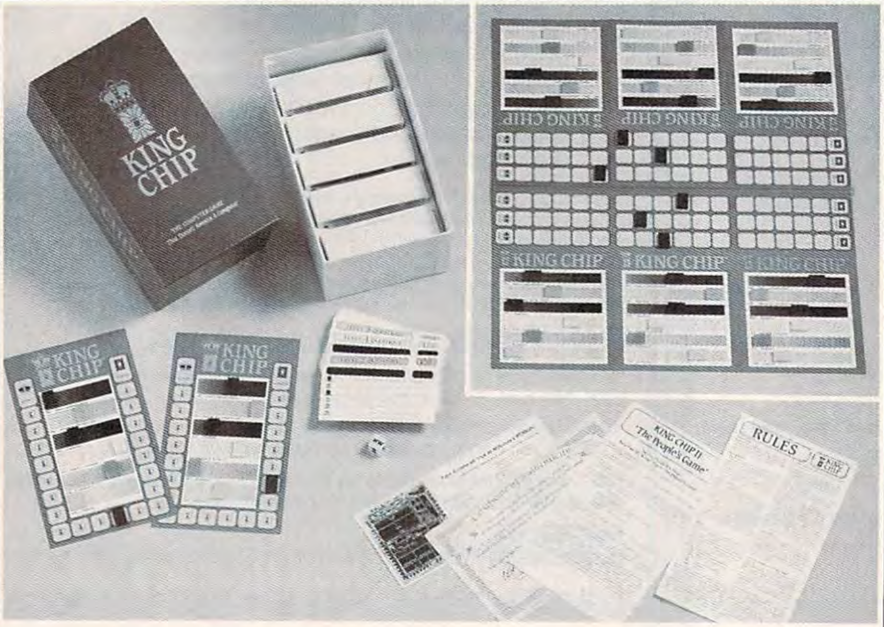
Manage your own baseball team and play the team of your choice with *The World's Greatest Baseball Game* from Epyx, an upgraded version of the popular original. This strategy game features over 75 teams, complete rosters for the 1984 and 1985 seasons, statistics from actual All Star and World Series teams, the ability to trade players, and a scoreboard that asks baseball trivia questions.

Users can custom-design their own teams by picking lineups from actual major league baseball rosters, and then challenge the team of their choice in a championship game.

For the Commodore 64/128, Apple II-series, and IBM at prices ranging from $24.95 to $34.95.

Epyx Computer Software, Inc., Sunnyvale, CA 94089.

Circle Reader Service Number 225.



King Chip *is a trivia game that tests your knowledge of computers and computing—it's a computer game that doesn't require a computer.*

Two New Teaching Aids

Gamco Industries, Inc. has released two new software packages to help teachers explain calendars and simple geometry. Both packages hold up to 200 student files which automatically record each student's records.

Calendar helps students learn the days and months; seasons, special days, and holidays; and how to use a calendar.

Perimeter, Area, & Volume offers simple geometry formulas and practice in using them. There are several levels in each lesson.

In either package, when the student achieves a certain score, he or she may play an arcade-style game as a reward.

Calendar and *Perimeter, Area, & Volume* are available for $39.95 each for the Apple II-series and the Commodore 64.

Gamco Industries, Inc., Box 1911, Big Spring, TX 79721.

Circle Reader Service Number 226.

Simple Graphics Program

The Graphics Magician Junior from Polarware is a graphics program for novice computer artists using Apple II computers. It utilizes 108 colors and patterns with a wide variety of computer "brushes." Pictures can then be saved to disk or printed out.

Polarware is a division of Penguin software.

Retail price is $34.95.

Polarware, 2600 Keslinger Rd., P.O. Box 311, Geneva, IL 60134.

Circle Reader Service Number 227.

Apple Graphics Utilities

Dark Star Systems has designed two printing utilities for Apple II computers.

MousePrintz lets *MousePrint* users print their pictures directly to an Epson or other non-Apple dot-matrix printer, plus offers a menu of screen-editing features, including full-screen viewing, inversion, mirror image, and cropping; rotation, shading, chart recording, and

ATARI 130XE

ATARI 130XE Super Computer Package
130XE Computer
1050 Disk Drive
1027 Printer
Atariwriter +
Call for individual & super package price

ATARI PRINTER INTERFACES

Uprint A	54.95
Uprint AW/16K	79.95
Uprint AW/64K	99.95
MPP 1150	59.95

INDUS GT DISK DRIVE...Call

ATARI 130XE SUPER PRINTER PKGS.

SG-10 Printer and U-Print A	275
Panasonic 1091 and U-Print A	309

Super Printer Packages have no extra shipping charges or credit card surcharges when shipped in Continental USA

ATARI 130XE SOFTWARE

BRODERBUND

Print Shop	28.95
Karateka	20.95
Print Shop Graph. I, II, or III	19.95
Print Shop Comp.	27.95

INFOCOM
See Commodore 64 section for items and prices

ELECTRONIC ARTS

Archon II	21.95
Archon	16.95
Seven Cit. of Gold	21.95
Skyfox	24.95
Pinball Const.	16.95
One on One	21.95

MICROPROSE

Silent Service	23.95
Gunship	23.95
Accrojet	23.95
F-15 Strike Eagle	23.95
Kennedy Approach	23.95

OSS

Basic XE-Cart	52.95
MAC 65 XL-Cart	49.95
Action-Cart	49.95
Basic XL-Cart	39.95
All Tool Kits	20.95

BATTERIES INCLUDED

Home Pak	34.95
Paper Clip	39.95
B-Graph	34.95

SYNAPSE

Syncalc	32.95
Synfile	32.95
Syntrend	25.95
Syncalc Templates	16.95
Loderunner Rescue	20.95
Mindwheel	27.95
Essex	27.95
Brimstone	27.95

SSI
See Commodore 64 section for items and prices

MISCELLANEOUS 130XE

Hacker	19.95
Amer. Cross Ctry.	19.95
Flight Simulator II	34.95
Ultima II	37.95
Ultima III	37.95
Universe	69.95
Letter Perfect	39.95
Data Perfect	39.95
Halley Project	27.95
Ultima I	23.95
Ultima IV	41.95
MMG Basic Comp.	69.95

ATARI 520ST*

Atari 520ST-RGB System...Call
Atari 520ST-Monochrome Sys...Call
SF314DS/DD 1 Megabyte Disk Drive........Call

We warranty all 520ST computers purchased from ComputAbility for ninety days.

***Please call for stock availability on Atari ST products before ordering by mail.**

MISCELLANEOUS ST

VIP Professional	129.95
Ultima II	39.95
Perry Mason	34.95
Degas	27.95
Farenheit 451	34.95
Amazon	34.95
Hacker	29.95
The Final Word	94.95
Deja Vu	39.95
PC/Intercom	89.95
Hex	27.95
Crimson Crown	27.95
Mudpies	23.95
King's Quest II	34.95
Gato	34.95
Borrowed Time	34.95
Personal Prolog	79.95
Personal Pascal	64.95
Zoomracks	59.95
Mi-Term	54.95
Regent Word	34.95
Sundog	27.95
Regent Spell	34.95
Winnie/Pooh	19.95
Isgur Portfolio	Call
Michtron Utilities	41.95
Music Studio	Call
Silent Service	Call

INFOCOM ST

Deadline	34.95
Starcross	34.95
Zork I, II, or III	29.95
Witness	27.95
Suspended	34.95
Planetfall	27.95
Sorcerer	29.95
Seastalker	27.95
Cutthroats	27.95
Hitchiker	27.95
Suspect	29.95
Wishbringer	27.95
Infidel	29.95
Enchanter	27.95
Spellbreaker	34.95
Mind Forever Voy.	34.95

AMIGA

Call for Hardware and add-on peripherals prices

AMIGA SOFTWARE

Hacker	29.95
Mindshadow	29.95
Archon	27.95
VIP Professional	139.95
Maxicom	39.95
Maxidesk	49.95
Maxiplan	119.95
One on One	29.95
7 Cities/Gold	29.95
Skyfox	29.95
Marble Madness	29.95
Return/Atlantis	29.95
Archon	29.95
Deluxe Paint	59.95
Borrowed Time	29.95
Infocom	Call
Fin. Cookbook	34.95

APPLE

PRACTICAL PERIPHERALS

We carry the complete line of products
Call for Current Prices

APPLE PRINTER INTERFACES AND BOARDS

Apricorn Parallel w/Graphics	69.95
Apricorn 16K Expansion Board	82.95
Apricorn 80 Column Board	64.95
Apricorn RS232 Interface	69.95
U-Print-Apple IIC w/64K	109.95
U-Print-Apple IIC w/16K	89.95

APPLE SOFTWARE

BRODERBUND

Print Shop	33.95
Print Shop Graphics I, II, or III	17.95
Print Shop Comp.	27.95
Karateka	21.95
Carmen Sandiego	25.95
Science Tool Kit	39.95
Bank Street Writer	44.95
Fantavision	27.95

ELECTRONIC ARTS

Adventure Const.	34.95
Archon II	27.95
Bard's Tale	29.95
Imagic Football	24.95
Auto-Duel	34.95
Skyfox	27.95
Lords of Conquest	27.95
One on One	27.95
Ultima III	39.95
Ultima IV	39.95
Moebius	39.95

EPYX

Ballblazer	24.95
Winter Games	24.95
Summer Games II	24.95
World's Great/Football	24.95
Eidolon	24.95
Apshai Trilogy	24.95
Koronis Rift	24.95

INFOCOM
See Atari 520ST section for items and prices.

MICROPROSE
See Atari 130XE section for items and prices.

MINDSCAPE

Color Me	20.95
Crossword Magic	34.95
Halley Project	29.95
A View To Kill	27.95
Racter	29.95
The Mist	27.95
Perfect Score	49.95
Voodoo Island	27.95
Rambo	27.95
American Challenge	Call
Goldfinger	Call

SIMON & SCHUSTER

Typing Tutor III	34.95
Kobiashi Adv	29.95
Webster Spell Chk.	39.95
Webster Thesaurus	84.95
Lovejoy SAT	49.95

SIR-TECH

Wizardry/Diam.	23.95
Wizardry/Legacy	27.95
Wizardry/Proving	33.95
Wizardry/Wernda	29.95
Wiziprint	19.95

SSI
See Commodore 64 section for items & prices.

APPLE MISCELLANEOUS

Beachead	23.95
Beachead II	23.95
Gamemaker	27.95
Hacker	27.95
Hardball	24.95
Sundog	27.95
Star League Base.	23.95
Microleague Base.	29.95
PFS Software	Call
Newsroom	39.95
Clip Art Vol. I	20.95
Gato	27.95
Superbase	69.95
Comp. Gardening	39.95
Kings Quest II	34.95
Dambusters	27.95
Alternate Reality	27.95
Prince	47.95
Movie Maker	39.95
Zorro	27.95
Goonies	27.95
Commando	25.95
Kung Fu Master	25.95
Karate Champ	25.95
Hulk	24.95
The Hobbit	24.95
Strip Poker	23.95
Pixit	34.95
Blazzing Paddles	34.95
Beagle Bros.	Call
Math Blaster	34.95
Fight Night	27.95

We stock hundreds of programs for the Apple, Atari, C-64 and IBM. If you don't see it listed here, don't hesitate to call.

NO SURCHARGE FOR MASTERCARD AND VISA

—EST. 1982—

COMMODORE 128

C-128 Computer	Call
1571 Disk Drive	Call
1902 Monitor	Call
1670 Modem	Call

In Stock Now

IBM PC

IBM PC SYSTEMS
Configured to your specific needs
Call for lowest price on IBM-PC, IBM-XT or IBM-AT

Corona PC-400 Compatible .. Call
Corona Portable PC Compatible .. Call

PC Multifunction Boards
We carry the complete line of **AST, Hercules, Paradise, STB, and Quadram**
Call for current prices

IBM PC SOFTWARE

Print Shop	39.95
Print Shop Graph. I	27.95
Bank Street Writer	49.95
Ancient Art of War	29.95

BORLAND

Sidekick	37.95
Turbo Pascal	49.95

BLUE CHIP

Baron	34.95
Squire	34.95
Millionaire	34.95
Tycoon	34.95

DIGITAL RESEARCH
Call for items and prices.

INFOCOM
See Atari 520ST for items and prices

LEADING EDGE

Nutshell	69.95
LE/WP Basic	67.95
LE/Word Proc +Speller	169.95

MICROPROSE

F-15 Strike Eagle	23.95
Kennedy Approach	27.95
Acrojet	27.95
Silent Service	27.95

MICROSOFT

Flight Simulator	38.95
Word	249.00
Multiplan	134.95

MINDSCAPE
See Apple Section for items and prices.

SIERRA

King's Quest	34.95
King's Quest II	34.95
Ultima II	39.95

THOUGHTWARE
Call for items and prices.

IBM MISCELLANEOUS

PFS	Call
Gato	27.95
Wizardry	39.95
Strip Poker	27.95
Electric Desk	204.95
D-Base III	Call
Sideways	39.95
Home Pak	34.95
Sargon III	34.95
Peachtree	Call
Jet	34.95
BPI Business	Call
Newsroom	39.95

COMMODORE 128 SOFTWARE

Multiplan 64/128	44.95
Consultant	52.95
Paper Clip/Spell	54.95
Swiftcalc	49.95
Wordwriter	49.95
Data Manager	49.95
Fleet System II	44.95
Superbase 128	69.95
Mach V/128	34.95

COMMODORE 64

C-64 Computer	Call
1541 Disk Drive	Call
1660 Modem	49.95
1670 Modem	169

C-64 SUPER PRINTER PKGS.

SG-10 & Xetec Supergraph	279
Panasonic 1091 & Xetec Supergraph	315
Legend 1080 & Xetec Supergraph	269

Super Printer Packages have no added shipping or charge card surcharges when shipped in Continental USA

B.I. 80 Column Card . . . 59.95
While Supplies Last

Cardco G-Wiz . . . 54.95

GENERAL HARDWARE

star

SG-10	209
SG-15	369
SD-10	339
SD-15	449
SR-10	489
SR-15	Call

PRINTERS

Panasonic 1091	245
Legend 808	169
Legend 1080	209
Powertype	309
Juki 5510	389
Epson	Call
Panasonic 1080	215

PRINTER BUFFERS

Microfazer	From 169
U-Buff 16K	79.95
U-Buff 64K	99.95

MODEMS

US Robotics 2400	469
Volksmodem 1200	189
Prometheus 1200	319
Password 1200	209
Novation	Call
PC Modems	Call

MONITORS

Commodore 1902	Call
Commodore 1802	169
Sakata SC-100	169
Samsung 12" Green	79.95
Samsung 12" Amb.	79.95
Taxan	Call
Amdek	Call

COMMODORE 64 SOFTWARE

ACCESS

Beach-Head	21.95
Beach-Head II	24.95
Raid/Moscow	24.95
Mach V-Cart	21.95

INFOCOM

Zork I	24.95
Zork II, or III	27.95
Deadline	29.95
Starcross	29.95
Witness	29.95
Planetfall	24.95
Hitchiker	24.95
Enchanter	24.95
Cutthroats	24.95
Sorcerer	29.95
Spellbreaker	29.95

SSI

Battalion Commander	24.95
Battle of Antietnam	32.95
Fighter Command (No Atari)	37.95
Norway 85 (No Atari)	21.95
Panzer Grenidier	24.95
USAAF	37.95
Breakthrough/Ardennes	37.95
Kampfgruppe	37.95
Phantasie (No Atari)	24.95
Broadsides	24.95
Carrier Force	37.95
Comp. Ambush	37.95
Mech Brigade (No Atari)	37.95
Field of Fire (No Apple)	24.95
Op. Mkt. Garden	32.95
Pro Tour Golf (No Atari)	24.95
Gemstone Warrior	21.95
Imp. Galactum	24.95
Computer Baseball	24.95
Comp. Quarterback	24.95

ELECTRONIC ARTS

Adv. Construction	29.95
Mail Order Monster	24.95
Racing Destruction	24.95
Ultima III	37.95
Ultima IV	41.95
Bard's Tale	27.95

See Atari 130XE section for rest of items & prices.

EPYX

Eidolon	24.95
Koronis Rift	24.95
Winter Games	24.95
Apshai Trilogy	24.95
Fast Load-Cart	24.95
Program/Tool Kit	29.95

MICROPROSE
See Atari 130XE section for items and prices.

MISCELLANEOUS COMMODORE 64

Print Shop	28.95
Cal-Kit	34.95
Superbase 64	47.95
Karateka	20.95
Hacker	20.95
Gamemaker	27.95
Ultima II	37.95
Karate Champ	25.95
Essex	27.95
Kung Fu/Kung Fu Master	25.95
9 Princes of Amber	22.95
Spy vs. Spy Vol. II	23.95
Maxi Golf	24.95
Internat'l Hockey	19.95
The Fourth Proto.	23.95
Blazing Paddles	24.95
Mirage Word	34.95
Mirage Database	34.95
Welcome Aboard	19.95
Super Huey	14.95
Spell It	34.95
Math Blaster	34.95
Word Attack	34.95
Odesta Chess	49.95
Brimstone	27.95

P.O. Box 17882, Milwaukee, WI 53217
ORDER LINES OPEN
Mon-Fri 11 a.m. - 7 p.m. CST • Sat. 12 p.m. - 5 p.m. CST

ComputAbility™

To Order Call Toll Free
800-558-0003

For Technical Info., Order Inquiries, or for Wisc. Orders
414-351-2007

ORDERING INFORMATION: Please **specify system.** For fast delivery send cashier's check or money order. Personal and company checks allow 14 business days to clear. School P.O.'s welcome. **C.O.D. charges are $3.00.** In Continental U.S.A. include $3.00 for software orders, 4% shipping for hardware, minimum $4.00. Master Card and Visa orders please include card #, expiration date and signature. WI residents please include 5% sales tax. HI, AK, FPO, APO, Puerto Rico and Canadian orders, please add 5% shipping, **minimum $5.00.** All other **foreign orders add 15% shipping, minimum $10.00.** All orders shipped outside the Continental U.S.A. are shipped first class insured U.S. mail. If foreign shipping charges exceed the minimum amount, you will be charged the additional amount to get your package to you quickly and safely. All goods are new and include factory warranty. Due to our low prices all sales are final. **All defective returns must have a return authorization number.** Please call (414) 351-2007 to obtain an R.A.# or your return will not be accepted. Prices and availability subject to change without notice.

automatic centering and adjustment of margins. For the 128K Apple IIe and Apple IIc.

The *ScreenSnapper* is designed for use with *Applesoft* and machine code programs running on the Apple II+, IIe, and IIc. This program lets you print to the screen in a variety of ways, including enlargement, rotation, inversion, and shading. It shows you what a printout will look like before printing. *ScreenSnapper* will not work with copy-protected software.

Suggested retail price for *MousePrintz* is $35.00, and $40.00 for *ScreenSnapper.*

Available from *Greengate Productions, Inc., 2041 Pioneer Ct. #15, San Mateo, CA 94403.*

Circle Reader Service Number 228.

256K RAM For The Amiga

A 256K display RAM card for the Amiga installs behind the front panel and expands available memory for programs and graphics to 512K. The card comes with a one year warranty, manual, and schematics. It's designed for compatibility with software and hardware.

Suggested retail price is $120.00

Starpoint Software, 122 South Broadway, Yreka, CA 96097-2902.

Circle Reader Service Number 229.

CompuServe And MCI Interconnect

CompuServe Inc. and MCI Communications Corporation have interconnected their electronic mail services. Subscribers to either of the two systems can instantaneously communicate with each other using the same methods and commands as before. CompuServe's InfoPlex and EasyPlex are both included.

Circle Reader Service Number 230.

High-Level Language For Amiga

Designed by Professor Niklaus Wirth, the creator of Pascal, *Modula-2* is a high-level language which encourages the user to write programs in modules. This method of programming makes it easy to design, write, and maintain software. Programmers familiar with Pascal should be able to learn the language in a few hours.

Modula-2 for the Amiga features full interface to the ROM Kernel, *Intuition,* and AmigaDOS; 32-bit native code implementation; separate compilation of modules with version control; a CODE statement for in-line assembly code; and the ability to quickly locate and identify errors in source code. It also supports transcendental functions and real numbers.

Modula-2, published by TDI Software, is not copy-protected. Suggested retail price for the regular version is $89.95; the developer's version is $149.95. Both come with a 300-page manual.

TDI Software, Inc., 10410 Markison Rd., Dallas, TX 75238.

Circle Reader Service Number 231.

Commemorative Version Of Mac Challenger

Profits from the sale of the commemorative version of Aegis Development's *Mac Challenger* flight simulator for the Macintosh will go to the Challenger space shuttle's Children's Fund and Rebuild the Space Shuttle Fund. The commemorative version has a sticker on the front of the package and is dedicated to the seven-member crew of the Challenger space shuttle.

Suggested retail price is $49.95.

Aegis Development Inc., 2210 Wilshire Blvd., #277, Santa Monica, CA 90403.

Circle Reader Service Number 232.

Home Accounting Package For Apple II

Schmidt Enterprises has introduced a sophisticated, easy-to-use accounting package appropriate for use in the home or small business. *The Accountant* can access an unlimited number of transactions, with no limit to the number of accounts and categories used. The user can instantly retrieve, print, or delete any transaction or group of transactions. The Printed Transaction Summary feature allows the user to print and total a selected group of transactions. A profit/loss statement can be created by subtracting debits from credits.

The Accountant comes with a manual containing sample disk files and a tutorial. An on-screen help menu is available at all times. No command phrases are used, as all functions can be activated by a single keypress. The program accesses the disk only for loading the program and saving data, which makes search and retrieval functions execute in seconds.

The Accountant retails for $120.

Schmidt Enterprises, 7448 Newcastle Ave., Reseda, CA 91335.

Circle Reader Service Number 233.

Telecommunications Package For Mac

Software Ventures Corporation has begun shipping *MicroPhone,* a telecommunication program for the Macintosh designed for both novice and experienced computer users.

MicroPhone automates the telecommunications process, allowing the user to do things like access stock quotes hourly or send and receive electronic mail without striking a single key, freeing him or her to do other work simultaneously. The program can record complete telecommunications sessions, remembering any series of keyboard commands and system prompts. The entire sequence can be called up at any later time with a single keystroke.

MicroPhone retails for $74.95, and carries a 30-day unconditional money-back guarantee.

Software Ventures Corporation, 2907 Claremont Ave., Suite 220, Berkeley, CA 94705.

Circle Reader Service Number 234.

Penmanship Practice On Your Computer Screen

Beginning writers quickly get the "feel" of correct letter formation with *Touch 'n Write,* a new penmanship program that lets students practice handwriting directly on the computer screen. A complete 23-lesson curriculum based on the Palmer Method of manuscript writing, the program operates with the Touch Window, a portable touch screen easily attached to the screen.

Using their fingers or the pen included with Touch Window, youngsters first learn to duplicate basic letter strokes. Next, they trace and then "fingerpaint" letters in colors chosen from their own on-screen paint set. As students move from one section of a lesson to the next, they get rest and reward through short animations based on the theme of their choice -- nature, fantasy, or the circus. When a letter or number lesson is completed, they can "touch 'n color" in an electronic picture book related to one of the three themes. And after a lesson is completed, a reward certificate can be printed out.

Touch 'n Write, available for 64K Apple II computers, costs $69 (software only; the Touch Window is additional).

Sunburst Communications, Inc., 39 Washington Ave., Pleasantville, NY 10570.

Circle Reader Service Number 235.

Cross Assembler Program For Atari ST

Lamar Micro has developed a 65C02 cross assembler program for the Atari 520 ST. The *C02 Cross Assembler* allows the ST to serve as a software development system for Apple, Atari, or Commodore computers that use the 6502 or the 65C02 microprocessor.

Suggested retail price is $89.95.

Lamar Micro, 2107 Artesia Blvd., Redondo Beach, CA 90278.

Circle Reader Service Number 236.

Nutrition Analysis Program

Micromedx has announced availability of *Macnutriplan*, a nutrition analysis program for the Apple Macintosh.

The program asks the user to type in the type and amount of food that he or she has had for each meal. It keeps a running tally of the meal's caloric content plus the user's choice of any two other elements, like cholesterol, saturated fat, vitamin B-6, or potassium. In addition, maximum values for any of these tallies can be set, causing an audible/visual alarm to go off when exceeded. If the user requests further analysis, the program will ask for the user's name and age, and calculate what percentage of his or her recommended daily allowance of key nutrients the meal will supply.

The program comes with a built-in directory of caloric and nutritional content of more than 400 common foods; up to 200 more can be added.

Macnutriplan requires 150K memory in a 512K Macintosh, one disk drive, and Microsoft BASIC 2.0. Suggested retail price is $75.

Micromedx, 187 Gardiners Ave., Levittown, NY 11756.

Circle Reader Service Number 237.

Stock Update Package For *Managing Your Money*

The Micro Education Corporation of America has expanded its offerings by introducing *Managing The Market*, a stock price update package for the IBM-PC that automatically accesses Dow Jones News/Retrieval Service.

Managing The Market is a cost-effective communications program that allows you to update securities automatically via modem. The program can be used in conjunction with Andrew Tobias' financial package *Managing Your Money*, or by itself with spreadsheets like *Lotus 1-2-3*. An onscreen stopwatch helps keep track of time spent online. Once connected to the service, all data is stored in a file so the user can log off quickly (reducing phone charges) and review the information at his or her leisure. Users can also create customized "hot lists" of up to 225 securities to check key prices at a glance, saving the time and expense of running through their entire list of securities.

Managing The Market runs on the IBM-PC and PCjr (with 256K), and requires DOS 2.0 or any later version.

Suggested retail price is $149.95, which includes a free password for Dow Jones News/Retrieval Service (a $75 value) and an hour's free connect time.

Micro Education Corporation of America, 285 Riverside Ave., Westport, CT 06880.

Circle Reader Service Number 238.

Chess Tutorial And Game For Apple And Commodore

Enlightenment, Inc., has announced the availability of *Paul Whitehead Teaches Chess,* a combination chess tutorial/opponenent for IBM, Apple II, and Commodore 64 computers.

The tutorial is designed to take the user from not knowing anything at all about chess to the point where he or she can beat a middle-level chess player. It is driven by a database that contains a tree-like structure where the user's choices guide the branching of the presentation. This allows the user to skip over information that he or she already knows. Each branch contains both moves and comments integrated together. The commentary appears and disappears every half move, so you can see an anylisis of your game while the players are on the screen.

Once the user is ready to take on an opponent, the *Coffeehouse Chess Monster* is a formidable one. The Apple and Commodore versions of this game portion of the program were written by international master Julio Kaplan. The IBM version is a customization of the SPOC program from Cypress Software. All three versions retail for $49.95.

Enlightenment, Inc., 1240 Sanchez St., San Francisco, CA 94114.

Circle Reader Service Number 239.

True BASIC Prices Reduced

Addison-Wesley has cut its site licensing fee for *True BASIC* from $5,000 to $1,250 in hopes of attracting more interest from high schools and colleges. In addition, the publisher is introducing *True BASIC* packages designed for students.

True BASIC Student's Reference Kit, available for the IBM PC or the Macintosh, includes full-function software and a reference manual for $49.95. *True BASIC Student's User Kit,* being sold for $41.95, includes the software and user manual.

True BASIC, a product of John Kemeny and Thomas Kurtz, the original creators of BASIC, features increased speed, multi-line functions, named subroutines, windows, and portability between the Macintosh and IBM PC.

A student calculus program, *Calculus Student's Toolkit,* for the Apple II and IBM PC also is available. The $19.95 program assists students in mastering topics such as limits, differentiation, and integration.

Addison-Wesley Publishing Company, Educational Media Systems Division, Reading, MA 01867.

Circle Reader Service Number 240.

Submarine Simulation

Overt Strategic Simulations has announced *OPERATION: keystone,* a submarine simulation for IBM PCs and compatibles with at least 200K RAM.

The program, which sells for $69, features randomly generated mission assignments and stresses strategic planning over quick reflexes.

Overt Strategic Simulations, P.O. Box 66424, Los Angeles, CA 90066.

Circle Reader Service Number 241.

Software Catalog On Disk

Electronic Courseware Systems, Inc. has made available a disk version of its software catalog. The disks, available now for Apple and Commodore 64 with an IBM version expected, catalog the company's instructional software offerings, including music, MIDI, math, science, language arts, and utility programs.

The disk is available for $2.99, which is refundable if software is purchased from the catalog. Paper versions of the catalog are available at no charge.

Electronic Courseware Systems, Inc, 1210 Lancaster Dr., Champaign, IL 61821.

Circle Reader Service Number 242.

Computerized Classic

A computer version of *Treasure Island* has been indroduced by Classics on Computer. The game, designed for students in grades 5–9, is intended to help students rediscover the joys of reading. A player's progress in the game depends on reading comprehension and vocabulary-building skills.

This adaptation of the Robert Louis Stevenson novel is available for Apple II series computers for $39.95.

Classics on Computer, 5150 Wilshire Blvd., Suite 502, Los Angeles, CA 90036.

Circle Reader Service Number 243.

Gato For 64

Spectrum Holobyte, Inc., has introduced a Commodore 64 version of the popular World War II submarine simulation game, *Gato,* previously available for the Apple and IBM computers.

Gato puts you in the captain's seat of a World War II "Gato" class submarine, as you play against the computer to decide who controls the seas. The 64 version includes eight missions, five difficulty levels, and three ships. And the Commodore version uses the 64's sound capabilities to add realism, including a digitized voice to receive mission assignments from SUBCOM.

Suggested retail price is $29.95.

Spectrum HoloByte, Inc., 1050 Walnut, Suite 325, Boulder, CO 80302.

Circle Reader Service Number 244.

128 Program Generator

OMNISoft & Associates has introduced *OMNICodeI,* a program that generates source code in writing BASIC programs and subroutines to handle screen formatting, input, and compiled output. The code generated is modular, REMarked, and compiler-compatible.

OMNICodeI has a user interface designed so that even a novice will be comfortable in the operating environment. For the experienced programmer, the package can save hours of work.

The Commodore 128 version requires at least one 1541 or 1571 disk drive and an 80-column display (either color or monochrome). It writes Commodore BASIC 7.0, and is compatible with the BLITZ!-128 BASIC Compiler from Skyles Electric Works. Retailing for $89.95, the package includes *OMNI Merge-128,* which allows the user to merge tokenized BASIC programs and subroutines.

OMNISoft & Associates, P.O. Box 280, Rogers, AZ 72756.

Circle Reader Service Number 245.

Geopolitical Simulation For IBM

Mindscape has announced that its highly-acclaimed *Balance of Power,* previously available only for the Macintosh, will be available in IBM format this June.

Written by noted software designer Chris Crawford, this one- or two-player strategy game allows players to assume the role of either the President of the United States or the General Secretary of the Soviet Union for a fictional eight-year period. Each leader must work to enhance his or her country's prestige, yet avoid nuclear war. They can support friendly governments, move against unfriendly governments, and try to foil the same efforts of the opposing superpower.

Balance of Power's vast database of information on the 62 nations represented in the game helps players make their strategic decisions. Players can learn about a country's political stability, GNP, literacy rate, and financial assistance to and from other nations. As players develop strategies, all data must be considered in light of international events presented as news items. The scenario is constantly changing.

With Microsoft Windows as the user interface, the IBM version creates a gameplay environment virtually identical to that of the Macintosh computer.

The IBM version of *Balance of Power* is $49.95.

Mindscape, Inc., 3444 Dundee Rd., Northbrook, IL 60062.

Circle Reader Service Number 246.

Commodore I/O Controller Card

The BH100 General Purpose I/O Card is an intelligent input-output device from Intelligent I/O, Inc., for the VIC-20, Commodore 64, and 128. The card provides a total of eight 8-bit parallel ports (32 separate input and 32 separate output lines). Since the ports are memory mapped, data is sent or retrieved by a single POKE or PEEK command.

The card can be used in a home control application, controlling lights, appliances, relays, motors, heating/cooling systems, and other electrical devices. It can also be used for more sophisticated applications, like laboratory data acquisition, automated testing/experimentation, and security systems, and can be connected to analog-to-digital and digital-to-analog converters.

Suggested retail price is $129.

Intelligent I/O, Inc., 30 Lawrence Ave., Potsdam, NY 13676.

Circle Reader Service Number 247.

Programming Utility For Amiga

Gimpel Software has announced the availability of *Amiga-Lint*, a diagnostic facility for the C programming language running on the Commodore Amiga. It's similar to the *Lint* that runs on the Unix operating system.

Amiga-Lint will analyze C programs and report on bugs, glitches, and inconsistencies. It helps develop reliable programs and port programs over from other machines and operating systems.

Some of the types of errors reported by *Amiga-Lint* include parameter-argument mismatches, library usage irregularities, variables declared but not used, and suspicious use of operators and unreachable code. The program's features include full K & R support, fast one-pass operation, no fixed-size tables to overflow, and special *Lint*-style comments to suppress errors. *Amiga-Lint* runs under the Amiga's CLI interface.

Suggested retail price is $98.

Gimpel Software, 3207 Hogarth Ln., Collegeville, PA 19426.

Circle Reader Service Number 248.

Apple, IBM Math Tutorials

Mindplay has introduced two programs that help children learn about math while they're having fun.

In *Campaign Math*, players practice their math skills as they research issues, raise funds, and choose the advertising media that will help them win an election. This helps them hone not only their math skills, but their knowledge of political science elements like platform issues, surveying techniques, population size, and fundraising. In *RoboMath*, arcade action inspires robo-mathematicians to practice multiplication and division as they close down trashbot factories. Players choose a quick-answer method or use the screen to work out more difficult problems with a unique step-by-step process that prompts development of long division or multiplication skills.

Both programs are available for IBM PC and Apple II computers, and retail for $39.95 each.

Mindplay Software, Methods & Solutions, Inc., 82 Montvale Ave., Stoneham, MA 02180.

Circle Reader Service Number 249. ©

HOTWARE: Software Best Sellers

This Month	Last Month	Title	Publisher	Remarks	Apple	Atari	Commodore	IBM	Macintosh
Entertainment									
1.	3.	*Ultima IV*	Origin Systems	Fantasy game	•	•	•		
2.	5.	*F-15 Strike Eagle*	MicroProse	Air combat simulation	•	•	•	•	
3.	1.	*Jet*	SubLogic	Jet simulation			•		
4.	4.	*Karateka*	Brøderbund	Action karate game	•	•	•		
5.	2.	*Silent Service*	MicroProse	Submarine simulation	•	•	•	•	
Education									
1.	1.	*Typing Tutor III*	Simon & Schuster	Typing instruction program	•		•	•	•
2.	2.	*Math Blaster!*	Davidson	Introductory math program, ages 6-12	•	•	•	•	
3.	4.	*Music Construction Set*	Electronic Arts	Music composition program	•	•	•		
4.	3.	*New Improved MasterType*	Scarborough	Typing instruction program	•	•	•	•	•
5.	5.	*I Am The C-64*	Creative/Activision	Introduction to the C-64			•		
Home Management									
1.	1.	*Print Shop*	Brøderbund	Do-it-yourself print shop	•	•	•		
2.	2.	*The Newsroom*	Springboard	Do-it-yourself newspaper	•		•	•	
3.	3.	*Bank Street Writer*	Brøderbund	Word processor	•	•	•	•	
4.	5.	*Print Shop Graphics Library*	Brøderbund	100 additional graphics	•	•	•		
5.		*Swiftax*	Timeworks	Tax preparation program	•		•	•	

Advertisers Index

To Our Readers:

COMPUTE! Publications is a part of the ABC Consumer Magazines group of ABC Publishing, Inc. and recently we consolidated many of our operations and moved our Customer Service Department to the New York ABC headquarters. If you have any questions regarding back issues, disk orders, book orders, or how to place an order, call toll free **1-800-346-6767**. New York residents should call 212-887-8525.

If you want to order a subscription to COMPUTE!, COMPUTE!'s GAZETTE, COMPUTE!'s GAZETTE DISK, or the COMPUTE! DISK, call **1-800-247-5470** or in Iowa call 1-800-532-1272.

Our Editorial Offices remain in Greensboro, North Carolina. If you have trouble using a program in one of the books or magazines, or if you wish to submit an article for publication, write us at COMPUTE! Publications, Inc., P.O. Box 5406, Greensboro, NC 27403.

We thank you for your interest and continued support of COMPUTE! Publications.

COMPUTE! Publications, Inc. abc

Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies

Reader-Friendly!

COMPUTE!

SAVE 32%
OFF THE $35.40 COVER PRICE
☐ 1 YEAR (12 ISSUES) $24

SAVE 36%
OFF THE $70.80 COVER PRICE
☐ 2 YEARS (24 ISSUES) $45

NAME ______________________________

ADDRESS ______________________ APT. ______

CITY/STATE/ZIP ______________________________

☐ PAYMENT ENCLOSED ☐ PLEASE BILL ME ☐ CHECK HERE IF RENEWAL

PLEASE ALLOW 4-6 WEEKS FOR DELIVERY OF YOUR FIRST ISSUE. FOREIGN AND CANADIAN PLEASE ADD $6 (U.S.) PER YEAR POSTAGE. OFFER SUBJECT TO CHANGE WITHOUT NOTICE. S

J1395

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 2312 GREENSBORO, NC

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE!
P.O. Box 914
Farmingdale, NY 11737

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 7551 DES MOINES, IA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE!'s DISK
P.O. BOX 10036
DES MOINES, IA 50347

COMPUTE!'s APPLE APPLICATIONS SPECIAL

☐ *COMPUTE!'s Apple Applications Special,* Spring 1986 **$3.95**
(In U.S. add $1.00 shipping and handling charges; outside U.S. add $2.00)

☐ Companion Disk to *COMPUTE!'s Apple Applications Special,* Spring 1986 **$16.95**
(Add $2.00 shipping and handling charges)

To order, send in this card with your payment or call toll-free **800-346-6767**.
ALL ORDERS MUST BE PREPAID IN U.S. FUNDS. (in NY 212-887-8525.)

☐ Payment enclosed (check or money order).
☐ Charge: ☐ Visa ☐ MasterCard

Account No.________________ Exp. Date____/____ (Required)

Signature________________ (Required when using credit card.)

Name________________ (Please print)

Address________________

City__________ State_____ Zip______

AAC586

PLACE
STAMP
HERE

COMPUTE!'s
APPLE APPLICATIONS SPECIAL
6th FLOOR SALES DEPARTMENT
825 Seventh Ave.
New York, N.Y. 10019

COMPUTE!'s FREE Reader Information Service

Use these cards to request FREE information about the products advertised in this issue. Clearly print or type your full name and address. Only one card should be used per person. Circle the numbers that correspond to the key number appearing in the advertisers index.

Send in the card and the advertisers will receive your inquiry. Although every effort is made to insure that only advertisers wishing to provide product information have reader service numbers, COMPUTE! cannot be responsible if advertisers do not provide literature to readers.

Please use these cards *only* for subscribing or for requesting product information. Editorial and customer service inquiries should be addressed to: COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. Check the expiration date on the card to insure proper handling.

Use these cards and this address only for COMPUTE!'s Reader Information Service. Do not send with payment in any form.

COMPUTE!

101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117
118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134
135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151
152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168
169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185
186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202
203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236
237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253

Please let us know. Do you

own:		plan to buy:
☐ 270	Apple	☐ 271
☐ 272	Atari	☐ 273
☐ 274	Commodore	☐ 275
☐ 276	IBM	☐ 277
☐ 278	TI-99/4A	☐ 279
☐ 280	Other ____ (specify model)	☐ 281

Please print or type name and address.
Limit one card per person.

Name ______________________

Address ______________________

City ______________________

State/Province ______________ Zip ________

Country ______________________

Phone ______________________

Please Include ZIP Code   Expiration Date 6/30/86   CO586

SUBSCRIBE TO COMPUTE!

For Fastest Service,
Call Our **Toll-Free**
US Order Line
800-247-5470
In IA call 1-800-532-1272

My Computer Is:

01 ☐ Apple 02 ☐ Atari 03 ☐ Commodore 64
04 ☐ VIC-20 05 ☐ IBM 06 ☐ TI-99/4A
99 ☐ Other________ ☐ Don't yet have one.

☐ $24.00 One Year US Subscription
☐ $45.00 Two Year US Subscription

Name ______________________

Address ______________________

City ______________ State ______________ Zip ________

☐ Payment Enclosed ☐ Bill me
Charge my: ☐ VISA ☐ MasterCard ☐ American Express
Account No. ______________ Expires ____ / ____

Your subscription will begin with the next available issue. Please allow 4-6 weeks for delivery of first issue. Subscription prices subject to change at any time. Outside the U.S.A. please add $6 for each subscription.

Place
Stamp
Here

COMPUTE! Reader Service
P.O. Box 2141
Radnor, PA 19089

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 7478 DES MOINES, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE!
PO BOX 10954
DES MOINES, IOWA 50347

SUPER HUEY

America's #1 Helicopter Flight Simulator

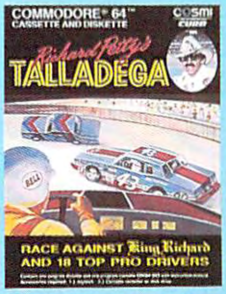
Forget the disappointment of other flight simulators . . . SUPER HUEY has eliminated them! – *Commodore Power Play Magazine*

"One of the best flight simulators ever tested!" – *Erv Bobo, Run Magazine*

Handles like a real helicopter. All your flying skill will be needed.

FOUR ACTION-PACKED ADVENTURES!

Solo Flight • Rescue
Explore • Combat



Race against King Richard Petty and 18 top drivers. A real life 3-D main event.
Atari® • Commodore® 64/128™

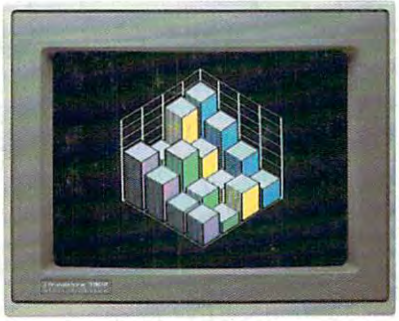
Your skill as an archer can protect you from the giant spiders, snakes, dragons, frogs, phantoms and, finally, the demogorgon.
Atari® • Commodore® 64/128™

Atari® • Apple® • Commodore® 64/128™

COSMI where the action is!

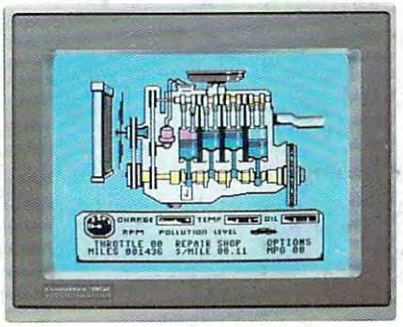
415 North Figueroa Street, Wilmington, CA 90744 • (213) 835-9687

All you need to do this



graph a spreadsheet

write a novel



fix an engine

compose a song

paint a picture

your banking

learn to fly

organize a data base

tell a story

forecast sales

When it comes to personal computers, you want the smartest, at a price that makes sense.

The new Commodore 128™ system has a powerful 128K memory, expandable by 512K. An 80-column display and 64, 128 and CP/M® modes for easy access to thousands of educational, business and home programs. And a keyboard, with built-in numeric keypad, that operates with little effort.

Or if the Commodore 128 is more machine than you had in mind, you can pick up the Commodore 64.® The Commodore 64 is our lower-priced model geared to more fundamental, basic needs.

Discover personal computers that do more for you. At prices you've been waiting for. From the company that sells more personal computers than IBM® or Apple.®

COMMODORE 128 AND 64 PERSONAL COMPUTERS

A Higher Intelligence